

Administrator's Guide

VMware Infrastructure (VI) Toolkit (for Windows)

BETA



Administrator's Guide
Revision: 20080311

You can find the most up-to-date technical documentation on our Web site at:

<http://www.vmware.com/support/>

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback on:

docfeedback@vmware.com

© 2008 VMware, Inc. All rights reserved. Protected by one or more U.S. Patent Nos. 6,397,242, 6,496,847, 6,704,925, 6,711,672, 6,725,289, 6,735,601, 6,785,886, 6,789,156, 6,795,966, 6,880,022, 6,944,699, 6,961,806, 6,961,941, 7,069,413, 7,082,598, 7,089,377, 7,111,086, 7,111,145, 7,117,481, 7,149,843, 7,155,558, 7,222,221, 7,260,815, 7,260,820, 7,269,683, 7,275,136, 7,277,998, 7,277,999, 7,278,030, 7,281,102, and 7,290,253; patents pending.

VMware, the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies.

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

About This Book	3
1 Getting Started with VI Toolkit (for Windows)	5
Introduction to the VI Toolkit Cmdlets	5
Command-Line Syntax	5
Launching VI Toolkit (for Windows) Console	6
List All VI Toolkit (for Windows) Cmdlets	6
Displaying Help for Any Cmdlet	6
Connecting to a Server	6
2 Basic Cmdlet Usage	9
PowerShell Cmdlet Usage	9
Pipelines	9
Wildcards	9
Common Parameters	10
Examples of Basic Usage of the VI Toolkit Cmdlets	10
3 Advanced Cmdlet Usage	11
Examples of Advanced Cmdlet Usage	11
Using the VI Toolkit Cmdlets	11
Web Service Access Cmdlets	13
Mixed Usage of VI Toolkit and Web Service Access Cmdlets	15
The Inventory Provider	15

BETA

About This Book

This book, the *Administrator's Guide*, provides information about using the VI Toolkit cmdlets (pronounced “commandlets”) set that ships with the VI Toolkit (for Windows) for managing, monitoring, automating, and handling life-cycle operations for VMware Infrastructure components—virtual machines, datacenters, storage, networks, and so on.

Revision History

This guide is revised with each release of the product or when necessary. A revised version can contain minor or major changes. [Table 1](#) summarizes the significant changes in each version of this guide.

Table 1. Revision History

Revision	Description
20080313	First version of the documentation for the VI Toolkit (for Windows) 1.0.

To view the most current version of this guide, see the VMware Web site documentation page at:

<http://www.vmware.com/support/pubs>

Intended Audience

This book is intended for administrators with different levels of familiarity with VMware Infrastructure administration and Windows PowerShell. There are two categories of users for the VI Toolkit (for Windows):

- **Basic** administrators can use PowerShell commands included in the VI Toolkit (for Windows) to manage their VMware virtual infrastructure from the command line.
- **Advanced** administrators can develop PowerShell scripts that may be reused by other administrators or integrated into other applications.

NOTE All VI Toolkit (for Windows) users are expected to be familiar with the details of VMware Infrastructure administration and the Windows operating system. Solution developers are expected to be familiar with the .NET infrastructure and the VIM object model as well.

Document Feedback

VMware welcomes your suggestions for improving our documentation. Send your feedback to:

docfeedback@vmware.com

Technical Support and Education Resources

The following sections describe the technical support resources available to you. To access the current versions of other VMware manuals, go to:

<http://www.vmware.com/support/pubs>

Online Support

You can submit questions or post comments to the Developer Community: SDKs and APIs forum, which is monitored by VMware technical support and product teams. To access the forum, go to:

<http://communities.vmware.com/community/developer>

Support Offerings

Find out how VMware support offerings can help meet your business needs. Go to:

<http://www.vmware.com/support/services>

VMware Education Services

VMware courses offer extensive hands-on labs, case study examples, and course materials designed to be used as on-the-job reference tools. For more information about VMware Education Services, go to:

<http://mylearn1.vmware.com/mgrreg/index.cfm>

Getting Started with VI Toolkit (for Windows)

1

The VI Toolkit (for Windows) provides easy-to-use C# and PowerShell interface to VMware Infrastructure APIs. It ships with a number of cmdlets that you can use to perform various administration tasks on VMware infrastructure components. This chapter explains how to get started using the VI Toolkit cmdlets.

This chapter covers the following topics:

- [“Introduction to the VI Toolkit Cmdlets”](#) on page 5
- [“Launching VI Toolkit \(for Windows\) Console”](#) on page 6

Introduction to the VI Toolkit Cmdlets

Microsoft PowerShell is both a command-line and scripting environment, designed for Windows. It leverages the .NET object model and provides administrators with management and automation capabilities. Working with PowerShell, like with any other console environment, is done by typing commands. In PowerShell commands are called cmdlets, which term we will use throughout this guide.

The VI Toolkit (for Windows) 1.0 ships with 101 cmdlets:

- 97 PowerShell-based cmdlets.
- 4 .NET cmdlets for use through PowerShell—the so called Web Service Access Cmdlets. See [“Web Service Access Cmdlets”](#) on page 13.

VI Toolkit cmdlets are created to answer the specific needs of the VMware Infrastructure administration and management. The VI Toolkit cmdlets are found in the `VMware.VimAutomation.Core` snapin.

Command-Line Syntax

VI Toolkit command-line syntax is the same as generic PowerShell syntax.

PowerShell cmdlets use a consistent verb-noun structure, where the verb specifies the action and the noun specifies the object to operate on. PowerShell cmdlets follow consistent naming patterns, which makes it easy to figure out how to construct a command if you know the object you want to work with.

All command categories take parameters and arguments. A parameter starts with a hyphen and is used to control the behavior of the command. An argument is a data value consumed by the command.

A simple PowerShell command looks like the following:

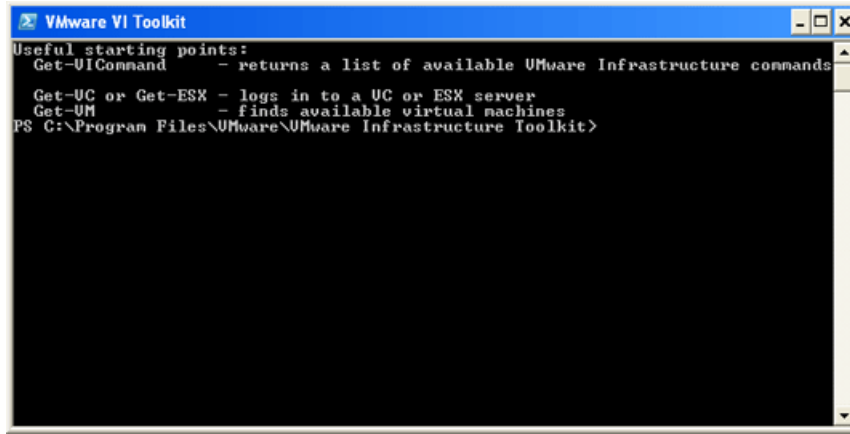
```
command -parameter1 -parameter2 argument1 argument2
```

Launching VI Toolkit (for Windows) Console

You can launch the VI Toolkit (for Windows) Console in one of two ways:

- From the **Start** menu, click **Programs > VMware > VMware Infrastructure Toolkit > VMware VI Toolkit**.
- Double-click the desktop icon (if you have chosen the option at installation).

The VI Toolkit (for Windows) console window opens in a PowerShell window.



List All VI Toolkit (for Windows) Cmdlets

If you are new to the VI Toolkit (for Windows), one thing you want to know is what cmdlets are available to you. To get a list of all VI Toolkit cmdlets, use the `Get-Command` cmdlet with the `-PSSnapin` parameter in the following way:

```
Get-Command -PSSnapin VMware.VimAutomation.Core
```

The VI Toolkit cmdlets are listed in the console window as one long, scrolling topic. You can view them a single page at a time by piping the results of the `Get-Command` cmdlet to the `more` option in the following way:

```
Get-Command -PSSnapin VMware.VimAutomation.Core | more
```

Displaying Help for Any Cmdlet

You can get help for a specific cmdlet by supplying the `Get-Help` command in the VI Toolkit Console. For example, for information on the `Add-VMHost` cmdlet, run the `Get-Help` command as follows:

```
Get-Help Add-VMHost
```

For more detailed information, add the `-full` parameter:

```
Get-Help Add-VMHost -full
```

Alternatively, you can use the `help` alias with any cmdlet:

```
help Add-VMHost
```

To view detailed help information page by page, pipe the `help` cmdlet to the `more` cmdlet:

```
help Add-VMHost -full | more
```

Connecting to a Server

To run specific VI Toolkit cmdlets and perform administration or monitoring tasks, first establish a connection to an ESX Server host or a VirtualCenter server.

In the VI Toolkit console window, type the following cmdlet:

```
Get-VIServer -Server <Server_Address> -Protocol <http_or_https> -User <user_name> -Password <user_password>
```


You should provide specific values for the server address of the target VirtualCenter Server or ESX Server host, the protocol (http or https), as well as the user name and password required for log in. For example:

```
Get-VIServer -Server 192.168.10.10 -Protocol http -User admin -Password sck9p84
```

After a connection is established, you are ready to run the VI Toolkit cmdlets.

For example, you can start a specific virtual machine using the following cmdlet:

```
Get-VM -name <virtual_machine_name> | Start-VM
```

For example, to run a virtual machine named MyVM, run:

```
Get-VM -name MyVM | Start-VM
```

BETA

BETA

Basic Cmdlet Usage

This chapter explores the basics of the VI Toolkit cmdlets usage.

The chapter discusses the following topics:

- [“PowerShell Cmdlet Usage”](#) on page 9
- [“Examples of Basic Usage of the VI Toolkit Cmdlets”](#) on page 10

NOTE This chapter does not discuss PowerShell basics. You are expected to have knowledge of PowerShell and its command-line and scripting conventions

PowerShell Cmdlet Usage

In this section, some of the cmdlets syntax and usage basic concepts are described.

Pipelines

A pipeline is a series of commands separated by the pipe operator `|`. Each command in the pipeline receives an object from the previous command, performs some operation on it, and then passes it along to the next command in the pipeline. Objects are output from the pipeline as soon as they become available. You can type a pipeline on a single line, or spread it across multiple lines. You can cycle backwards through command history using the up arrow, so it is easier to repeat pipelines if you type them on a single line.

Wildcards

PowerShell has a number of pattern matching operators called wildcards, which work on strings. For example, to display all files with a `.txt` extension, run:

```
dir *.txt
```

In this example, the asterisk `*` operator matches any combination of characters.

Wildcard patterns allow you to specify character ranges as well. For example, to display all files that start with the letter S or T and have a `.txt` extension, run:

```
dir [st]*.txt
```

You can use the question mark `?` wildcard to match any single character within a sequence of characters. For example, to display all `.txt` files whose names consist of ‘string’ and one more character in the end, run:

```
dir string?.txt
```

All wildcard expressions can be used with the VI Toolkit cmdlets.

Common Parameters

The Windows PowerShell engine implements a set of reserved parameter names, referred to as common parameters. All PowerShell cmdlets, including the VI Toolkit cmdlets, support them. Common parameters are: `Verbose`, `Debug`, `ErrorAction`, `ErrorVariable`, `OutVariable`, and `OutBuffer`. Respectively, the following aliases are reserved for these parameters: `vb`, `db`, `ea`, `ev`, `ov`, and `ob`.

In addition, there are two risk mitigation parameters in PowerShell: `WhatIf` and `Confirm`. `WhatIf` is used when you want to see the effects of a command without executing it. `Confirm` is used when a cmdlet performs an operation that stops a program or service or deletes data.

For more details on the usage of common parameters, use the following command:

```
Get-Help about_CommonParameters
```

Examples of Basic Usage of the VI Toolkit Cmdlets

This section provides some examples of basic VI Toolkit cmdlets usage.

Example 1: Connecting to a Server

The following cmdlet establishes a connection to a local server and asks for credentials (user name and password), as they are not passed as parameters:

```
Get-VIServer -Server <VI_server_address>
```

For example:

```
Get-VIServer -Server esx3.example.com
```

Example 2: Retrieving a List of Virtual Machines

After establishing a connection to a server, you can list all virtual machines on the target system by running the following cmdlet:

```
Get-VM
```

Example 3: Retrieving a List of the Hosts

To list all hosts on a target VMware Infrastructure server you have established a connection with, use:

```
Get-VMHost
```

Advanced Cmdlet Usage

This chapter provides examples of advanced usage of the VI Toolkit cmdlets.

The chapter discusses these topics:

- [“Examples of Advanced Cmdlet Usage”](#) on page 11
- [“The Inventory Provider”](#) on page 15

Examples of Advanced Cmdlet Usage

This section contains examples of using the VI Toolkit cmdlets, the Web Service Access cmdlets, and the inventory provider functionality for retrieving and managing VMware Infrastructure objects.

Using the VI Toolkit Cmdlets

The following examples illustrate how to use advanced functionality provided by the VI Toolkit cmdlets:

Example 1: Creating VI Objects

The following example illustrates common methods for creating folders, datacenters, clusters, resource pools and virtual machines using VI Toolkit cmdlets.

- 1 Establish a connection to a VirtualCenter server using the `Get-VIServer` command:

```
Get-VIServer -Server <Server_Address>
```

For example:

```
Get-VIServer -Server 192.168.10.10
```

When prompted, provide the administrator's user name and password to authenticate access on the server.

- 2 Get the inventory root folder and create a new folder called `USA` in it:

```
$usaFolder = Get-Folder -NoRecursion | New-Folder -Name USA
```

Note that the information about the location of the new folder is specified through the pipeline.

- 3 Create a new datacenter called `OrlandoDC` in the `USA` folder:

```
New-Datacenter -Location $usaFolder -Name OrlandoDC
```

- 4 Create a folder called `Engineering` under `OrlandoDC`:

```
Get-Datacenter OrlandoDC | New-Folder -Name Engineering  
$engineeringFolder = Get-Folder -Name engineering
```

NOTE Search in PowerShell is not case-sensitive.

- 5 Create a new cluster `ResearchAndDevelopmentCluster` in the `Engineering` folder:

```
New-Cluster -Location $engineeringFolder -Name ResearchAndDevelopmentCluster -DRSEnabled
-DRSMODE FullyAutomated
```

NOTE DRS (distributed resource scheduler) is a facility that allows automatic allocation of cluster resources.

- 6 Add a host in the cluster using the `Add-VMHost` command, which will prompt you for credentials:

```
$hostInCluster1 = Add-VMHost -Name <host IP address> -Location ( Get-Cluster
ResearchAndDevelopmentCluster )
```

The parentheses interpolate the object returned by the `Get-Cluster` command into `Location` parameter.

- 7 Create a resource pool in the cluster's root resource pool:

```
$clusterRootRP = Get-ResourcePool -Location ( Get-Cluster ResearchAndDevelopmentCluster )
-Name Resources
```

```
New-ResourcePool -Location $clusterRootRP -Name DevelopmentResources
-CpuExpandableReservation $true -CpuReservationMhz 500 -CpuSharesLevel high
-MemExpandableReservation $true -MemReservationMB 500 -MemSharesLevel high
```

- 8 Create a virtual machine synchronously:

```
New-VM -Name DevelopmentVM1 -VMHost $hostInCluster1 -ResourcePool ( Get-ResourcePool
DevelopmentResources ) -DiskMB 4000 -MemoryMB 256
```

- 9 Create a virtual machine asynchronously:

```
$vmCreationTask = New-VM -Name DevelopmentVM2 -VMHost $hostInCluster1 -ResourcePool
( Get-ResourcePool DevelopmentResources ) -DiskMB 4000 -MemoryMB 256 -RunAsync
```

The `-RunAsync` parameter specifies that the command will be executed asynchronously. This means that in contrast to a synchronous operation, you do not have to wait for the process to complete before supplying the next command in the command line.

Example 2: Using Virtual Machine Templates

A virtual machine template is a reusable image created from a virtual machine. The template, as a derivative of the source virtual machine, includes virtual hardware components, an installed guest operating system, and software applications.

This example illustrates how to create virtual machines templates and convert them to virtual machines. It uses the VMware Infrastructure objects created in the previous example.

- 1 Add an additional 2GB hard disk to the `DevelopmentVM2` virtual machine:

```
$vmCreationTask | Wait-Task | New-HardDisk -CapacityKB ( 2 * 1024 * 1024 )
```

- 2 Create a template from the `DevelopmentVM1` virtual machine:

```
New-Template -VM ( Get-VM DevelopmentVM1 ) -Name WebTemplate -Location (Get-Datacenter
orlandoDC )
```

A virtual machine template is a reusable image created from a virtual machine. The template, as a derivative of the source virtual machine, includes virtual hardware components, an installed guest operating system and software applications.

- 3 Convert this template for a use by a virtual machine named `DevelopmentVM3`:

```
Get-Template WebTemplate | Set-Template -ToVM -Name DevelopmentVM3
```

- 4 Create a template from the `DevelopmentVM2` virtual machine:

```
New-Template -VM ( Get-VM DevelopmentVM2 ) -Name DatabaseTemplate -Location (Get-Datacenter
orlandoDC )
```

- 5 Convert this template to a virtual machine named `DevelopmentVM4`:

```
Get-Template DatabaseTemplate | Set-Template -ToVM -Name DevelopmentVM4
```

- 6 Move the virtual machines into the `DevelopmentResources` resource pool:

```
Get-VM DevelopmentVM? | Move-VM -Destination ( Get-ResourcePool DevelopmentResources )
```

Use `?` as a wildcard to match just one symbol. The command returns all virtual machines whose names start with `DevelopmentVM` and have one more symbol at the end. In this example, `DevelopmentVM1`, `DevelopmentVM2`, `DevelopmentVM3`, and `DevelopmentVM4` are retrieved and moved into the `DevelopmentResources` resource pool.

- 7 Start the virtual machines in the `DevelopmentResources` resource pool:

```
Get-ResourcePool DevelopmentResources | Get-VM | Start-VM
```

Example 3: Making Snapshots

A snapshot captures the memory, disk, and settings state of a virtual machine at a particular moment. When you revert to a snapshot, you return all these items to the state they were in at the time you took that snapshot.

The following example illustrates taking a snapshot of virtual machines and then reverting the virtual machines to it. It uses the objects created in Example 1.

- 1 Take a snapshot of all virtual machines in the `DevelopmentResources` resource pool:

```
Get-ResourcePool DevelopmentResources | Get-VM | New-Snapshot -Name InitialDevConfig
```

The `-Location` parameter takes arguments of the `VIContainer` type, on which `Cluster`, `Datacenter`, `Folder`, `ResourcePool`, and `VMHost` object types are based. Therefore, the `-Location` parameter can use arguments of all these types.

- 2 Revert all virtual machines in the `DevelopmentResources` resource pool to the `InitialDevConfig` snapshot:

```
$VMs = Get-VM -Location ( Get-ResourcePool DevelopmentResources )
foreach( $vm in $VMs ) { Set-VM -VM $vm -Snapshot ( Get-Snapshot -VM $vm -Name
InitialDevConfig ) }
```

Example 4: Removing VI Objects

In this example you will remove some of the VMware Infrastructure objects created in the previous examples.

- 1 Retrieve the virtual machines using the `'?'` wildcard symbol in the `Name` parameter and remove them along with their disks:

```
Get-VM DevelopmentVM? | Remove-VM -DeleteFromDisk
```

- 2 Remove the `USA` folder:

```
Remove-Folder -Folder ( Get-Folder usa )
```

By executing the `Remove-Folder` command, all child objects for the folder—datacenters, folders, clusters, hosts, resource pools, and so on—will be also removed.

Web Service Access Cmdlets

The VI Toolkit (for Windows) 1.0 list of cmdlets includes four Web Service Access cmdlets included in the VI Toolkit (for Windows) 1.0 list of cmdlets:

- `Find-EntityView`
- `Find-EntityViews`
- `Get-View`
- `Get-Views`

They enable access to the programming model of the .NET Toolkit from PowerShell and can be used to initiate .NET Toolkit objects. Each object:

- Is a static copy of a server-side managed object and is not automatically updated when the object on the server changes.
- Includes properties and methods that correspond to the properties and operations of the server-side managed object. For more information about server-side object methods and properties, check the *VMware Infrastructure (VI) API Reference Guide* (http://www.vmware.com/support/pubs/sdk_pubs.html).

Generally, the Web Service Access cmdlets provide an alternative tool for low-level VMware Infrastructure management. Using the Web Service Access cmdlets requires some knowledge of both PowerShell scripting and the VMware Infrastructure API.

Example 1: Filtering VI Objects

This example illustrates the use of the `Find-EntityViews` cmdlet in combination with a filter. The filter parameter is a `NameValueCollection` containing one or more pairs of filter criteria. Each of the criteria consists of a property path and a value that represents a regular expression pattern used to match the property.

The filter in this example gets a list of the virtual machines that are powered on and their guest operating system names contain "Windows XP". For each item in the list, the `Find-EntityViews` cmdlet will initiate shutdown of the guest operating system.

- 1 Create a filter:


```
$filter = New-Object -TypeName "System.Collections.Specialized.NameValueCollection"
```
- 2 Modify the filter to include the power state of the virtual machines:


```
$filter.Add("Runtime.PowerState", "poweredOn")
```
- 3 Include guest operating system name in the filter:


```
$filter.Add("Config.GuestFullName", "Windows XP")
```
- 4 Get a list of the virtual machines using the created filter and call the `ShutdownGuest` method for each virtual machine in the list:


```
Find-EntityViews -ViewType "VirtualMachine" -Filter $filter | foreach{$_}.ShutdownGuest() }
```

Example 2: Populating a View Object

This example illustrates how to populate a view object from an already retrieved managed object using the `Get-View` cmdlet.

- 1 Populate the host object of a virtual machine:


```
$hostView = Get-View -MoRef (Find-EntityView -ViewType "VirtualMachine").Runtime.Host
```

Use the `-MoRef` parameter to specify the managed object reference.
- 2 Retrieve runtime information:


```
$hostView.Summary.Runtime
```

Example 3: Updating the State of a Server-Side Object

This example illustrates how to update the state of server-side objects:

- 1 Create a filter by name:


```
$filter = New-Object -TypeName "System.Collections.Specialized.NameValueCollection"
$filter.Add("Name", "DevelopmentVM2")
$vm = Find-EntityView -ViewType "VirtualMachine" -Filter $filter
```
- 2 Print the current power state:


```
$vm.Runtime.PowerState
```


- 3 Change the power state of the virtual machine:

```
If ($vm.Runtime.PowerState -ne "PoweredOn") {
    $vm.PowerOnVM($vm.Runtime.Host)
} else {
    $vm.PowerOffVM()
}
```

- 4 Print the value of \$vm power state (the powerstate is still not updated because the vm property values are not updated automatically):

```
$vm.Runtime.PowerState
```

- 5 Update the view:

```
$vm.UpdateViewData($null)
```

- 6 Show the actual power state of the virtual machine:

```
$vm.Runtime.PowerState
```

Mixed Usage of VI Toolkit and Web Service Access Cmdlets

To get more advantages of the usability and functionality of the VI Toolkit cmdlets and the Web Service Access cmdlets you can combine their usage. The following example illustrates this:

- 1 Use the Get-VMHost cmdlet to get a virtual machine host by host name:

```
$vmhost = Get-VMHost -Name "host.domain.com"
```

"host.domain.name" is the IP address or the DNS name of the ESX server as shown in the inventory.

- 2 Get a host view by VmHost ID:

```
$hostView = Get-View -MoRef $vmhost.ID
```

- 3 Call the reboot method of the Host view object to reboot the host:

```
$hostView.RebootHost()
```

The Inventory Provider

The inventory provider is designed to expose a raw inventory view of the inventory items from a server. It enables interactive navigation and file-style management of the VMware Infrastructure inventory. By creating a PowerShell drive based on a managed object (such as datacenter), you obtain a view of its contents and the relationships between the items. In addition, you are able to manipulate objects (move, rename or delete them) by running commands from the VI Toolkit console.

Example 1: Basic Functions of the Inventory Provider

The following example illustrates some basic functions of the inventory provider:

- 1 Establish a connection to the server using the Get-VIServer command:

```
Get-VIServer -Server <Server_Address>
```

For example:

```
Get-VIServer -Server 192.168.10.10
```

When prompted, provide the administrator user name and password to authenticate access on the server.

- 2 Get the root folder of the server:

```
$root = Get-Folder -NoRecursion
```

- 3 Create a PowerShell drive named VI, based on the server root folder. You can use the built-in New-PSDrive cmdlet.

```
New-PSDrive -Location $root -Name vi -PSProvider VimInventory -Root '\'
```

In this release, a single backslash is the required value for the -Root parameter.

- 4 Access the new drive by running the following command:

```
cd vi:
```

To list the drive content, run

```
dir
```

`dir` is an alias of the `Get-ChildItem` cmdlet.

- 5 Navigate through your server inventory using the `cd` command with the full path to the host. For a fictional VMware Infrastructure inventory, it might look like the following:

```
cd Folder01\DataCenter01\host\Web\LiveHost01
```

- 6 List the content of the host using the `ls` command:

```
ls
```

`ls` is the UNIX style alias of the `Get-ChildItem` cmdlet.

This command will return the virtual machines and the root resource pool of the host. To view only the virtual machines of the host, run the following cmdlet:

```
Get-VM
```

When called within the inventory drive, `Get-VM` retrieves only the virtual machines on the current drive location.

- 7 To delete a virtual machine named `LiveVm01`, type:

```
del LiveVm01
```

- 8 To rename a virtual machine from `LiveVm01New` to `LiveVm01`, type:

```
ren LiveVm01New LiveVm01
```

- 9 To start all virtual machines whose names start with `LiveVm`, type:

```
dir LiveVm* | Start-VM
```

Note that the provider file-style operations work on the same inventory objects as the toolkit cmdlets.