VMware vSphere PowerCLI User's Guide

vSphere PowerCLI 5.5 Release 2 for Tenants

EN-001417-00

vmware®

You can find the most up-to-date technical documentation on the VMware 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 to:

docfeedback@vmware.com

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

Contents

VMware vSphere PowerCLI for Tenants User's Guide

The *VMware vSphere PowerCLI for Tenants User's Guide* provides information about installing and using the vSphere PowerCLI for Tenants cmdlets (pronounced "commandlets") for managing, monitoring, automating, and handling lifecycle operations for VMware® vCloud Director systems.

To help you start with vSphere PowerCLI for Tenants, this documentation includes descriptions of specific concepts and features of vSphere PowerCLI for Tenants. In addition, it provides a set of usage examples and sample scripts.

Intended Audience

This documentation is intended for all users and administrators who need to install and use vSphere PowerCLI for Tenants. This documentation is written for vCloud Director tenant users who are familiar with virtual machine technology, Windows PowerShell, and vCloud Director.

Tenants are organization administrators and users in vCloud Director systems.

Depending on their role in the organization, they can populate the organization with users and groups, create and manage access privileges, create and manage catalogs, media, vApps, and vApp templates, and update some organization policies and properties. **Tenants** cannot create or remove organizations.

- Basic tenants can use cmdlets included in vSphere PowerCLI for Tenants to manage organizations and their resources from the command line.
- With vSphere PowerCLI for Tenants advanced tenants can develop PowerShell scripts that can be reused by other administrators or integrated into other applications.

Introduction to VMware vSphere PowerCLI for Tenants

1

VMware vSphere PowerCLI for Tenants is a snap-in of cmdlets based on Microsoft PowerShell for automating the administration of organizations in vCloud Director. It provides C# and PowerShell interfaces to VMware vCloud API.

- Microsoft PowerShell Basics on page 7

  vSphere PowerCLI for Tenants is based on Microsoft PowerShell and uses the PowerShell basic syntax and concepts.

- Concepts of vSphere PowerCLI for Tenants on page 8

  vSphere PowerCLI for Tenants cmdlets are created to automate administration in cloud organizations and to introduce some specific features in addition to the PowerShell concepts.

Microsoft PowerShell Basics

vSphere PowerCLI for Tenants is based on Microsoft PowerShell and uses the PowerShell basic syntax and concepts.

Microsoft PowerShell is both a command-line and scripting environment, designed for Windows. It uses the .NET object model and provides administrators with system administration and automation capabilities. To work with PowerShell, you run commands named cmdlets.

- PowerShell Command-Line Syntax on page 7

  PowerShell cmdlets use a consistent verb-noun structure, where the verb represents the action and the noun represents the object to operate on.

- PowerShell Pipelines on page 8

  A pipeline is a series of commands separated by the pipe operator `|`.

- PowerShell Wildcards on page 8

  PowerShell has a number of pattern-matching operators named wildcards that you can use to substitute one or more characters in a string, or substitute the complete string.

- PowerShell Common Parameters on page 8

  The Windows PowerShell engine retains a set of parameter names, referred to as common parameters. All PowerShell cmdlets, including the vSphere PowerCLI for Tenants cmdlets, support them.

PowerShell Command-Line Syntax

PowerShell cmdlets use a consistent verb-noun structure, where the verb represents the action and the noun represents the object to operate on.

PowerShell cmdlets follow consistent naming patterns, ensuring that construction of a command is easy if you know the object that 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 has the following syntax:

```
command –parameter1 –parameter2 argument1, argument2
```

PowerShell 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 to the next command in the pipeline. Objects are output from the pipeline as soon as they become available.

PowerShell Wildcards

PowerShell has a number of pattern-matching operators named wildcards that you can use to substitute one or more characters in a string, or substitute the complete string.

All wildcard expressions can be used with the vSphere PowerCLI for Tenants cmdlets. For example, you can view a list of all files with a `.txt` extension by running `dir *.txt`. In this case, the asterisk * operator matches any combination of characters.

With wildcard patterns you can indicate character ranges as well. For example, to view all files that start with the letter S or T and have a `.txt` extension, you can run `dir [st]*.txt`.

You can use the question mark ? wildcard to match any single character within a sequence of characters. For example, to view all `.txt` files with names that consist of `string` and one more character at the end, run `dir string?.txt`.

PowerShell Common Parameters

The Windows PowerShell engine retains a set of parameter names, referred to as common parameters. All PowerShell cmdlets, including the vSphere PowerCLI for Tenants cmdlets, support them.

Some of the PowerShell common parameters are `Verbose`, `Debug`, `ErrorAction`, `ErrorVariable`, `OutVariable`, and `OutBuffer`. For a full list of the common parameters and more details on their usage, run `Get–Help about_CommonParameters`.

PowerShell offers two risk mitigation parameters: `WhatIf` and `Confirm`.

`WhatIf`	Displays the effects of a command without running it.
`Confirm`	Prompts for confirmation before running a command that stops a program or service, or deletes data.

Concepts of vSphere PowerCLI for Tenants

vSphere PowerCLI for Tenants cmdlets are created to automate administration in cloud organizations and to introduce some specific features in addition to the PowerShell concepts.

- Providing Login Credentials on page 9

  When you provide login credentials in the command prompt or in a script file, a PowerShell limitation might prevent vSphere PowerCLI for Tenants from processing non-alphanumeric characters correctly. To prevent login problems, escape the non-alphanumeric characters in your credentials.

- Selecting Objects in vSphere PowerCLI for Tenants on page 9

  In vSphere PowerCLI for Tenants, you can pass strings and wildcards to all parameters that take vCloud Director objects, datastores, or `CIServer` objects as arguments. In vSphere PowerCLI for Tenants, this approach is named Object-by-Name (OBN) selection.

- Running Cmdlets Asynchronously in vSphere PowerCLI for Tenants on page 10

  By default, vSphere PowerCLI for Tenants cmdlets return an output only after completion of the requested tasks. If you want a cmdlet to return to the command line immediately, without waiting for the tasks to complete, you can use the `RunAsync` parameter.

- Managing Default Server Connections on page 10

  By default, vSphere PowerCLI for Tenants cmdlets run on the vCloud Director servers you are connected to, if no target servers can be determined from the provided parameters.

- Views in vSphere PowerCLI for Tenants on page 11

  The vSphere PowerCLI for Tenants list of cmdlets includes the `Get-CIView` cmdlet, which enables access to vSphere PowerCLI views for vCloud.

- About Articles in vSphere PowerCLI for Tenants on page 11

  You can learn more about some features and concepts of vSphere PowerCLI for Tenants from the built-in help articles named about articles. You can access them through a running vSphere PowerCLI for Tenants process.

Providing Login Credentials

When you provide login credentials in the command prompt or in a script file, a PowerShell limitation might prevent vSphere PowerCLI for Tenants from processing non-alphanumeric characters correctly. To prevent login problems, escape the non-alphanumeric characters in your credentials.

To escape non-alphanumeric characters in vSphere PowerCLI for Tenants, you need to place the expression that contains them in single quotes (').

NOTE When you provide your login credentials in the Specify Credential dialog box, you do not need to escape non-alphanumeric characters.

Example: Connecting to Your Organization

This example illustrates how to escape non-alphanumeric characters when connecting to a selected organization with the `Adminis!ra!or` user name and the `pa$$word` password.

```
Connect-CIServer -Server cloud.example.com -Username 'Adminis!ra!or' -Password 'pa$$word' -Org
'MyOrganization'
```

Selecting Objects in vSphere PowerCLI for Tenants

In vSphere PowerCLI for Tenants, you can pass strings and wildcards to all parameters that take vCloud Director objects, datastores, or `CIServer` objects as arguments. In vSphere PowerCLI for Tenants, this approach is named Object-by-Name (OBN) selection.

Instead of assigning an object name to a cmdlet parameter, users can pass the object through a pipeline or a variable. For example, the following three commands are interchangeable:

- `Remove-CIVApp -VApp 'My VApp'`
- `Get-CIVApp -Name 'My VApp' | Remove-CIVApp`
- `Remove-CIVApp -VApp (Get-CIVApp -Name 'My VApp')`

NOTE In vSphere PowerCLI for Tenants, passing strings as pipeline input is not supported.

If you provide a non-existing object name, an OBN failure occurs. In such cases, vSphere PowerCLI for Tenants generates a non-terminating error and runs the cmdlet ignoring the invalid name.

For more details about OBN, run `help about_OBN`.

Example: An OBN failure

This example illustrates the occurrence of an OBN failure.

```
Set-CIVApp -VApp 'MyVApp1', 'MyVApp2', 'MyVApp3' -Server $ciServer1, $ciServer2 -Description
"This vApp belongs to the MyOrg organization."
```

If the vApp named *MyVApp2* does not exist on either of the selected servers, vSphere PowerCLI for Tenants generates a non-terminating error and applies the command only on the vApps named *MyVApp1* and *MyVApp3*.

Running Cmdlets Asynchronously in vSphere PowerCLI for Tenants

By default, vSphere PowerCLI for Tenants cmdlets return an output only after completion of the requested tasks. If you want a cmdlet to return to the command line immediately, without waiting for the tasks to complete, you can use the `RunAsync` parameter.

When you use the `RunAsync` parameter, the cmdlet returns `Task` objects instead of its usual output. The `Status` property of a returned `Task` object contains a snapshot of the initial state of the task. This state is not updated automatically and has the values `Error`, `Queued`, `Running`, or `Success`. You can refresh a task state by retrieving the task object with the `Get-Task` cmdlet. If you want to observe the progress of a running task and wait for its completion before running other commands, use the `Wait-Task` cmdlet.

NOTE In vSphere PowerCLI for Tenants, the `RunAsync` parameter affects only the invocation of a cmdlet and does not control whether the initiated tasks run consecutively or in parallel. For example, the `Remove-CIVApp` cmdlet might remove the selected vApps simultaneously or consecutively depending on the internal design of vSphere PowerCLI for Tenants. To make sure that tasks initiated by a cmdlet run consecutively, run the cmdlet in a loop, each time applying it to a single object.

Example: Running Remove-CIVApp with and without the RunAsync parameter

```
Remove-CIVApp $vAppList
```

The command returns no output when all vApps stored in the *$vAppList* variable are removed, irrespective of whether they are removed simultaneously.

```
Remove-CIVApp $vAppList -RunAsync
```

The command returns an output that consists of one or more `Task` objects immediately.

Managing Default Server Connections

By default, vSphere PowerCLI for Tenants cmdlets run on the vCloud Director servers you are connected to, if no target servers can be determined from the provided parameters.

When you connect to a vCloud Director system by using `Connect-CIServer`, the server connection is stored in the *$DefaultCIServers* array variable. This variable contains all connected servers for the current session. To remove a server from the *$DefaultCIServers* variable, you can either use `Disconnect-CIServer` to close all active connections to this server, or modify the value of *$DefaultCIServers* manually.

Views in vSphere PowerCLI for Tenants

The vSphere PowerCLI for Tenants list of cmdlets includes the `Get-CIView` cmdlet, which enables access to vSphere PowerCLI views for vCloud.

Views in vSphere PowerCLI for Tenants are .NET objects that correspond to server-side objects. Each operation defined by VMware vCloud API has a corresponding view method.

A vSphere PowerCLI view object has the following characteristics:

- It includes properties and methods that correspond to the properties and operations of the server-side objects.
- It is a static copy of a server-side object and is not updated automatically when the object on the server changes.
- It includes additional methods other than the operations offered in the server-side managed object.

NOTE Using `Get-CIView` for low-level vCloud Director management requires some knowledge of both PowerShell scripting and the vCloud API.

About Articles in vSphere PowerCLI for Tenants

You can learn more about some features and concepts of vSphere PowerCLI for Tenants from the built-in help articles named about articles. You can access them through a running vSphere PowerCLI for Tenants process.

Running `Help About_*` lists all built-in Windows PowerShell and vSphere PowerCLI for Tenants about articles.

Table 1-1. Accessing Built-In Help Articles for vSphere PowerCLI for Tenants

Article Title	Command	Article Description
Handling Invalid Certificates	`Help About_Invalid_Certificates`	When you try to connect to a vCloud Director server and the server cannot recognize any valid certificates, the Invalid Certificate prompt appears.
Object-by-Name (OBN)	`Help About_OBN`	To help you save time and effort, vSphere PowerCLI for Tenants lets you select objects by their names.
Using the RunAsync Parameter	`Help About_RunAsync`	When you set the `RunAsync` parameter, you indicate that you want to run the command asynchronously.
Authenticating with a vCloud Director Server	`Help About_Server_Authentication`	To authenticate with a vCloud Director server, you can provide a user name and password through the User and Password parameters, or a PSCredential object through the Credential parameter.

Installing VMware vSphere PowerCLI for Tenants

2

VMware vSphere PowerCLI for Tenants lets you manage organizations and their resources from the command line. You can install vSphere PowerCLI for Tenants on all supported Windows operating systems.

After installing the package on your machine, you can connect to your vCloud Director system by providing valid authentication credentials.

Supported Operating Systems

You can install vSphere PowerCLI for Tenants only on supported Windows operating systems.

VMware vSphere PowerCLI 5.5 Release 2 for Tenants is supported on the following operating systems:

OS Type	64-Bit
Server	■ Windows Server 2012 R2 ■ Windows Server 2008 R2 Service Pack 1
Workstation	■ Windows 8.1 ■ Windows 7 Service Pack 1

Supported VMware Environments

You can use the vSphere PowerCLI for Tenants components to manage supported vCloud Director environments.

VMware vSphere PowerCLI 5.5 Release 2 for Tenants is compatible with VMware vCloud Director 1.5.1, VMware vCloud Director 5.1, and VMware vCloud Director 5.5.

NOTE You can automate only vCloud Director 5.1 features against vCloud Director 5.5.

Supported Windows PowerShell Versions

vSphere PowerCLI for Tenants is compatible with multiple versions of Windows PowerShell, but requires specific PowerShell 2.0 components to function properly.

You must verify that you have PowerShell 2.0 before installing a newer version of PowerShell.

VMware vSphere PowerCLI 5.5 Release 2 for Tenants is compatible with the following PowerShell versions:

- Windows PowerShell 2.0
- Windows PowerShell 3.0
- Windows PowerShell 4.0

Prerequisites for Installing and Running vSphere PowerCLI for Tenants

Before installing and running vSphere PowerCLI for Tenants, verify that you have installed the required software on the same machine.

If you want to work with VMware vSphere PowerCLI 5.5 Release 2 for Tenants, make sure the following software is present on your system:

- Windows PowerShell 2.0
- A supported version of .NET Framework
  - .NET Framework 2.0 with Service Pack 2
  - .NET Framework 3.0 or .NET Framework 3.0 with Service Pack 1, or Service Pack 2
  - .NET Framework 3.5 or .NET Framework 3.5 with Service Pack 1

Install vSphere PowerCLI for Tenants

After installing the package on your machine, you can run vSphere PowerCLI for Tenants cmldets to manage your organization and its resources in the cloud.

Prerequisites

- Before installing vSphere PowerCLI for Tenants, see "Prerequisites for Installing and Running vSphere PowerCLI for Tenants," on page 14.
- Verify that you have uninstalled VMware vSphere PowerCLI from your system.

Procedure

1 Download the latest version of vSphere PowerCLI for Tenants from the VMware Web site.

2 Navigate to the folder that contains the vSphere PowerCLI for Tenants installer file you downloaded and double-click the executable file.

3 On the Welcome page, click **Next**.

4 Accept the license agreement terms and click **Next**.

5 On the Destination Folder page, select the location to install vSphere PowerCLI for Tenants and click **Next**.

6 On the Ready to Install the Program page, click **Install** to proceed with the installation.

7 Click **Finish** to complete the installation process.

What to do next

Enable remote signing. See "Set the Properties to Support Remote Signing," on page 15.

Set the Properties to Support Remote Signing

If you want to run scripts and load configuration files with vSphere PowerCLI for Tenants, you must set the execution policy of Windows PowerShell to `RemoteSigned`.

For security reasons, Windows PowerShell supports an execution policy feature. It determines whether scripts are allowed to run and whether they must be digitally signed. By default, the execution policy is set to `Restricted`, which is the most secure policy. For more information about the execution policy and script digital signing in Windows PowerShell, run `Get-Help About_Signing`.

You can change the execution policy by using the `Set-ExecutionPolicy` cmdlet.

Procedure

1 From your Windows taskbar, select **Start > All Programs > VMware > VMware vSphere PowerCLI for Tenants**.

The console window of vSphere PowerCLI for Tenants opens.

2 In the console window, run `Set-ExecutionPolicy RemoteSigned`.

Uninstall vSphere PowerCLI for Tenants

You can uninstall vSphere PowerCLI for Tenants from your Windows system by using the default uninstall tool of the operating system.

Prerequisites

Close the vSphere PowerCLI for Tenants application.

Procedure

1 Select the default uninstall tool for your Windows system from Control Panel.

2 Select VMware vSphere PowerCLI for Tenants from the list and click **Change**.

3 On the Program Maintenance page, select **Remove** and click **Next**.

4 Click **Remove**.

Configuring VMware vSphere PowerCLI for Tenants

3

To extend and customize the features of VMware vSphere PowerCLI for Tenants, you can configure the application settings for different users and user groups, modify the script configuration file of vSphere PowerCLI for Tenants, and add custom scripts.

This chapter includes the following topics:

Scoped Settings of vSphere PowerCLI for Tenants

In vSphere PowerCLI for Tenants you can set the scope of the settings to enhance security and personalize the configuration.

Configuring the Scope of the Settings of vSphere PowerCLI for Tenants

Scoped configuration enhances system security and prevents nonadministrator users from introducing global changes to the configuration of vSphere PowerCLI for Tenants.

For greater control over the configuration of vSphere PowerCLI for Tenants, the `Set-PowerCLIConfiguration` cmdlet provides the `Scope` parameter.

Table 3-1. Valid Values for the Scope Parameter

Parameter Value	Description
`Session`	Configures settings for the current session and does not modify any configuration files of vSphere PowerCLI for Tenants on your system.
`User`	Configures settings for the current Windows user and modifies some configuration files of vSphere PowerCLI for Tenants on your system.
`AllUsers`	Configures settings for all users and modifies some configuration files of vSphere PowerCLI for Tenants on your system.

Priority of Settings Scopes in vSphere PowerCLI for Tenants

vSphere PowerCLI for Tenants loads the program configuration based on the scope that you select for each setting.

Table 3-2. Scope Impact on the Behavior of vSphere PowerCLI

Scope	Priority	Impact
`Session`	High	■ When started, vSphere PowerCLI for Tenants tries to load settings with the Session scope first. ■ Session settings override `User` and `All Users` settings. ■ Session settings are valid for the current session only.
`User`	Medium	■ When vSphere PowerCLI for Tenants cannot detect `Session` settings, the program tries to load `User` settings from the configuration files of vSphere PowerCLI for Tenants. ■ User settings override `AllUsers` settings. ■ User settings are automatically detected from the configuration files of vSphere PowerCLI for Tenants.
`AllUsers`	Low	■ When vSphere PowerCLI for Tenants cannot detect `Session` and `User` settings, the program loads `AllUsers` settings. ■ `AllUsers` settings do not override `Session` and `User` settings. ■ `AllUsers` settings are automatically detected from the configuration files of vSphere PowerCLI for Tenants.

Configuration Files of vSphere PowerCLI for Tenants

The copies of the `PowerCLI_settings.xml` file on your system contain User and All Users settings for vSphere PowerCLI for Tenants.

Installing vSphere PowerCLI for Tenants creates two copies of `PowerCLI_settings.xml` on your system. The version of your Windows operating system determines the location of the copies of `PowerCLI_settings.xml`.

Table 3-3. Location of the Copies of PowerCLI_settings.xml

Windows OS Version	Location	Description
Windows Vista and later	`%APPDATA%\VMWare\PowerCLI`	Contains settings for the current Windows user only.
	`%SYSTEMDRIVE %\ProgramData\VMware\PowerCLI`	Contains settings for all users.
Earlier Windows versions	`%SYSTEMDRIVE%\Documents and Settings\[Username]\Application Data\VMware\PowerCLI`	Contains settings for the current Windows user only.
	`%SYSTEMDRIVE%\Documents and Settings\All Users\Application Data\VMware\PowerCLI`	Contains settings for all users.

Users with advanced knowledge and understanding of Windows PowerShell and VMware vSphere PowerCLI, can manually modify the contents of `PowerCLI_settings.xml` to change the settings of vSphere PowerCLI for Tenants. Modifying `PowerCLI_settings.xml` might require administrator privileges.

NOTE If you modify the contents of `PowerCLI_settings.xml` manually while vSphere PowerCLI for Tenants is running, you need to restart vSphere PowerCLI for Tenants for the changes to take effect.

Script Configuration Files of vSphere PowerCLI for Tenants

Starting vSphere PowerCLI for Tenants automatically loads the script configuration file located in the `Scripts` folder in the installation directory of vSphere PowerCLI for Tenants.

Default Script Configuration File

The default script configuration file of vSphere PowerCLI for Tenants is `Initialize-PowerCLIEnvironment.ps1`. Loading the file provides access to vSphere PowerCLI for Tenants cmdlets aliases and to other configuration settings.

`Initialize-PowerCLIEnvironment.ps1` is included in the installation package of vSphere PowerCLI for Tenants.

Custom Script Configuration File

If you want to load custom vSphere PowerCLI for Tenants settings automatically, you can create a script configuration file named `Initialize-PowerCLIEnvironment_Custom.ps1` in the `Scripts` folder. The application recognizes and loads the custom file after the default script configuration file.

Load the Script Configuration File in Other PowerShell Tools

If you want to work with vSphere PowerCLI for Tenants from another PowerShell-based tool, such as PowerShell Plus or PowerGUI, you must load the default script configuration file manually.

Procedure

1 Run the PowerShell-based tool you have installed on your system.

2 At the command line, change the active directory to the folder where you have installed vSphere PowerCLI for Tenants.

3 In the command line, type **`.\Scripts\Initialize-PowerCLIEnvironment.ps1`** and press Enter.

After the tool loads the default script configuration file, custom script configuration files, if any, load automatically.

Customizing vSphere PowerCLI for Tenants with Script Configuration Files

You can edit and extend the configuration script of vSphere PowerCLI for Tenants to set up the environment, set startup actions for vSphere PowerCLI for Tenants, or define cmdlets aliases.

Creating a Custom Script Configuration File

If you want to load custom vSphere PowerCLI for Tenants settings automatically, you can create a script configuration file named `Initialize-PowerCLIEnvironment_Custom.ps1` in the `Scripts` folder. The application recognizes and loads the custom file after loading the default script configuration file.

Signing the Script Configuration File

When the execution policy of Windows PowerShell is set to `RemoteSigned`, you do not need to sign the script configuration file after editing.

When the execution policy of your system is set to `AllSigned`, you need to sign the script configuration file after editing. If you do not sign the file, vSphere PowerCLI for Tenants will not load your modified configuration.

To learn more about setting the execution policy, see "Set the Properties to Support Remote Signing," on page 15.

Sample Scripts for Managing vCloud Director with VMware vSphere PowerCLI for Tenants

4

To help you get started with VMware vSphere PowerCLI for Tenants, this documentation provides a set of sample scripts that illustrate basic and advanced tasks in cloud administration.

- Create vApp Networks for a Selected vApp on page 27

  To define how the virtual machines in a vApp connect to each other and access other networks, you need to create a vApp network. When creating the vApp network, you can select the settings for the network, or adopt them from an organization policy.

- Modify or Remove vApp Networks on page 28

  Based on the type of the vApp network, you can configure various network settings, such as DNS, static IP pools, and firewalls. If you no longer need a vApp network, you can remove it.

Connect to an Organization on a vCloud Director Server

To run vSphere PowerCLI for Tenants cmdlets on a vCloud Director server and perform administration or monitoring tasks, first establish a connection to your organization on the vCloud Director server.

If your login credentials contain special characters, you might need to escape them. For more information, see "Providing Login Credentials," on page 9.

Prerequisites

If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for vSphere PowerCLI for Tenants tasks to complete running.

NOTE If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- Run `Connect-CIServer` with the server name, the name of your organization, and valid credentials.

```
Connect-CIServer -Server cloud.example.com -User 'MyAdministratorUser' -Password
'MyPassword' -Org 'MyOrganization'
```

Import a vApp Template from the Local Storage

To make an OVF package from your local storage available to other cloud users, you can import the package and save it as a vApp template in a catalog.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve the catalog to which you want to add the imported vApp template.

```
$myCatalog = Get-Catalog -Name 'MyCatalog'
```

2 Retrieve the organization virtual datacenter (vDC) to which you want to add the imported vApp template.

```
$myOrgVdc = Get-OrgVdc -Name 'MyOrgVdc'
```

3 Import a virtual machine from your local storage and save it as a vApp template in the cloud.

```
Import-CIVAppTemplate -SourcePath 'C:\OVFs\WindowsXP\WindowsXP.ovf' -Name
'MyWindowsXPVAppTemplate' -OrgVdc $myOrgVdc -Catalog $myCatalog
```

Create a vApp Template from a vApp

Creating vApp templates from vApps in the cloud might minimize future efforts for cloning vApps. You can use the templates later to create new vApps that are based on the source vApp.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve the source vApp for the vApp template that you want to create.

```
$myVApp = Get-CIVApp -Name 'MyVApp'
```

2 If the source vApp is running, stop it.

```
$myVApp = Stop-CIVApp -VApp $myVApp
```

3 Retrieve the catalog to which you want to add the new vApp template.

```
$myCatalog = Get-Catalog -Name 'MyCatalog'
```

4 Retrieve the organization vDC to which you want to add the new vApp template.

```
$myOrgVdc = Get-OrgVdc -Name 'MyOrgVdc'
```

5 Create the new vApp template.

```
New-CIVAppTemplate -Name 'MyVAppTemplate' -VApp $myVApp -OrgVdc $myOrgVDC -Catalog $myCatalog
```

6 Start the source vApp.

```
$myVApp = Start-CIVApp -VApp $myVApp
```

What to do next

Create a vApp from the template and modify the vApp. See"Create and Modify a vApp," on page 23.

Create and Modify a vApp

You can use vApp templates to instantiate vApps. After creating the vApp, you can modify its settings to minimize the consumption of computing and storage resources.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve the organization vDC to which you want to add the new vApp.

```
$myOrgVdc = Get-OrgVdc -Name 'MyOrgVdc'
```

2 Retrieve the source vApp template for your new vApp.

```
$myVAppTemplate = Get-CIVAppTemplate -Name 'MyVAppTemplate'
```

3 Create your new vApp.

```
$myVApp = New-CIVApp -Name 'MyVApp' -VAppTemplate $myVAppTemplate -OrgVdc $myOrgVDC
```

By default, the vApp is powered off.

4 Renew the runtime lease for the new vApp and set it to 12 hours.

```
Set-CIVApp -VApp $myVApp -RuntimeLease "12:0:0" -RenewLease
```

To set leases, you can use the *days.hours:minutes:seconds* syntax.

5 Start the new vApp.

```
Start-VApp -VApp $myVApp
```

Manage Virtual Machines with vApps

For a large-scale approach to administration, you can start, stop, or restart virtual machines or their guest operating systems by running cmdlets on the associated vApps.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Power on all virtual machines in all vApps with names starting with *MyVApp*.

```
Get-CIVApp -Name 'MyVApp*' | Start-CIVApp
```

2 Suspend all virtual machines in all vApps with names starting with *YourVApp*.

```
Get-CIVapp -Name 'YourVApp*' | Suspend-CIVApp
```

3 Power off all virtual machines in the vApp named *MyVApp1*.

```
Get-CIVapp -Name 'MyVApp1' | Stop-CIVApp
```

4 Shut down the guest operating systems of all virtual machines in the vApp named *MyVApp2*.

```
Get-CIVapp -Name 'MyVApp2' | Stop-CIVAppGuest
```

5 Restart the guest operating systems of all virtual machines in the vApp named *MyVApp3*.

```
Get-CIVapp -Name 'MyVApp3' | Restart-CIVAppGuest
```

6 Reset all virtual machines in the vApp.

```
Get-CIVapp -Name 'MyVApp4' | Restart-CIVApp
```

Manage Virtual Machines and Their Guest Operating Systems

For a targeted approach to administration, you can use the `CIVM` and `CIVMGuest` cmdlets to handle lifecycle operations for one or more virtual machines.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve all virtual machines with names starting with *MyVM* and power them on.

```
Get-CIVM -Name 'MyVM*' | Start-CIVM
```

2 Suspend all virtual machines with names starting with *YourVM*.

```
Get-CIVM -Name 'YourVM*' | Suspend-CIVM
```

3 Power off the virtual machine named *MyVM1*.

```
Get-CIVM -Name 'MyVM1' | Stop-CIVM
```

4 Shut down the guest operating system of the virtual machine named *MyVM2*.

```
Get-CIVM -Name 'MyVM2' | Stop-CIVMGuest
```

5 Restart the guest operating system of the virtual machine named *MyVM3*.

```
Get-CIVM -Name 'MyVM3' | Restart-CIVMGuest
```

6 Reset the nonresponsive virtual machine named *MyVM4*.

```
Get-CIVM -Name 'MyVM4' | Restart-CIVM
```

Retrieve a List of the Internal and External IP Addresses of Virtual Machines in vApps

When managing vApps in the cloud, you might need to obtain information about the NIC settings of the associated virtual machines.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve the organization for which you want to generate the report.

```
$myOrg = Get-Org -Name 'MyOrg'
```

2 Retrieve all vApps in the organization.

```
$vApps = Get-CIVApp -Org $myOrg
```

3 Populate an array with the information that you want to report.

```
$vAppNetworkAdapters = @()
foreach ($vApp in $vApps) {
        $vms = Get-CIVM -VApp $vApp
        foreach ($vm in $vms) {
                $networkAdapters = Get-CINetworkAdapter -VM $vm
                foreach ($networkAdapter in $networkAdapters) {
                        $vAppNicInfo = New-Object "PSCustomObject"
                        $vAppNicInfo | Add-Member -MemberType NoteProperty -Name VAppName -
Value $vApp.Name
                        $vAppNicInfo | Add-Member -MemberType NoteProperty -Name VMName   -
Value $vm.Name
                        $vAppNicInfo | Add-Member -MemberType NoteProperty -Name NIC      -
Value ("NIC" + $networkAdapter.Index)
                        $vAppNicInfo | Add-Member -MemberType NoteProperty -Name ExternalIP -
Value $networkAdapter.IpAddress
                        $vAppNicInfo | Add-Member -MemberType NoteProperty -Name InternalIP -
Value $networkAdapter.ExternalIpAddress

                        $vAppNetworkAdapters += $vAppNicInfo
                }
        }
}
```

Running this script retrieves the names of the virtual machines and their associated vApp, the IDs of the NICs of the virtual machines, and external, and internal IP addresses of the NICs.

4 Display the report on the screen.

```
$vAppNetworkAdapters
```

Create and Manage Access Control Rules

By defining access control rules you can assign levels of access to separate users, user groups, or everyone in the organization. You can define access control rules for catalogs and vApps.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Create a new rule for accessing the vApp named *MyVApp*.

```
New-CIAccessControlRule -Entity 'MyVApp' -EveryoneInOrg -AccessLevel "Read"
```

All users in the organization have read-only access to the vApp.

2 Modify the access rule for a trusted user who needs full control over *MyVApp*.

```
New-CIAccessControlRule -Entity 'MyVApp' -User "MyAdvancedUser" -AccessLevel "FullControl"
```

3 Restrict the full control access of *MyAdvancedUser* to read/write access.

```
$accessRule = Get-CIAccessControlRule -Entity 'MyVApp' -User 'MyAdvancedUser'
$accessRule | Set-CIAccessControlRule -AccessLevel "ReadWrite"
```

4 Remove the custom rule that you created earlier for *MyAdvancedUser*.

```
$accessRule | Remove-CIAccessControlRule
```

Filter and Retrieve vApp Networks

To generate reports about vApp networks, you need to retrieve the respective vApp networks. You can use search criteria to filter the results returned by `Get-CIVAppNetwork`.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

- Get the vApp network named *MyVAppNetwork*.

```
Get-CIVAppNetwork -Name 'VAppNetwork'
```

- Get all vApp networks for the vApp named *MyVApp*.

```
Get-CIVapp -Name 'MyVApp'| Get-CIVAppNetwork
```

- Get all vApp networks of connection type direct and direct fenced.

```
Get-CIVAppNetwork -ConnectionType Direct, DirectFenced
```

- Get all direct vApp networks that connect to the organization network named *MyOrgNetwork*.

```
Get-OrgNetwork -Name 'MyOrgNetwork' | Get-CIVAppNetwork -ConnectionType Direct
```

Filter and Retrieve Organization Networks

To generate reports about organization networks, you need to retrieve the respective organization networks. You can use search criteria to filter the results returned by `Get-OrgNetwork`.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

- Get all organization networks for the organization named *MyOrg*.

  ```
  Get-Org -Name 'MyOrg' | Get-OrgNetwork
  ```

- Get all organization networks that are connected to the external network named *MyExternalNetwork*.

  ```
  Get-OrgNetwork -ExternalNetwork 'MyExternalNetwork'
  ```

- Get the organization network that is named *MyInternalOrgNetwork* and is connected to the network pool named *MyNetworkPool*.

  ```
  Get-NetworkPool -Name 'MyNetworkPool' | Get-OrgNetwork -Name 'MyInternalOrgNetwork'
  ```

Create vApp Networks for a Selected vApp

To define how the virtual machines in a vApp connect to each other and access other networks, you need to create a vApp network. When creating the vApp network, you can select the settings for the network, or adopt them from an organization policy.

To address multiple networking scenarios for a vApp, you can create multiple vApp networks.

- Create an Isolated vApp Network on page 27

  When you do not want the virtual machines in a vApp to connect to objects outside the vApp, you must create an isolated vApp network.

- Create an NAT Routed vApp Network on page 28

  To provide a vApp network with DHCP, firewall, NAT, and VPN services, you must create it as an NAT routed vApp network.

- Create a Direct vApp Network on page 28

  To establish a network connection between the virtual machines in a vApp and an organization network, you need to create a direct vApp network.

Create an Isolated vApp Network

When you do not want the virtual machines in a vApp to connect to objects outside the vApp, you must create an isolated vApp network.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve the vApp for which you want to create a vApp network.

  ```
  $myVApp = Get-CIVapp -Name 'MyVApp'
  ```

2 Create the new vApp network with a selected gateway and network mask.

  ```
  New-CIVAppNetwork -VApp $myVApp -Name 'MyVAppInternalNetwork' -Routed -Gateway '192.168.2.1'
  -Netmask '255.255.255.0' -ParentOrgNetwork $null
  ```

  By default, the vApp network has an enabled firewall.

Create an NAT Routed vApp Network

To provide a vApp network with DHCP, firewall, NAT, and VPN services, you must create it as an NAT routed vApp network.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve the vApp for which you want to create a vApp network.

```
$myVApp = Get-CIVapp -Name 'MyVApp'
```

2 Retrieve the organization network to which you want to connect the vApp network.

```
$myOrgNetwork = Get-OrgNetwork -Name 'MyOrgNetwork'
```

3 Create the new vApp network with a gateway and network mask, defined pool of static IP addresses, and a disabled firewall.

```
New-CIVAppNetwork -VApp $myVApp -ParentOrgNetwork $myOrgNetwork -Name
'MyVAppInternalNetwork' -Routed -Gateway '192.168.2.1' -Netmask '255.255.255.0' -
DisableFirewall -StaticIPPool "192.168.2.100 - 192.168.2.199"
```

If you do not run `New-CIVAppNetwork` with the `DisableFirewall` parameter, the new vApp network has an enabled firewall by default.

Create a Direct vApp Network

To establish a network connection between the virtual machines in a vApp and an organization network, you need to create a direct vApp network.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve the vApp for which you want to create a vApp network.

```
$myVApp = Get-CIVapp -Name 'MyVApp'
```

2 Retrieve the organization network that you want to connect to.

```
$myOrgNetwork = Get-OrgNetwork -Name 'MyOrgNetwork'
```

3 Create a direct vApp network that connects to the selected organization network.

```
New-CIVAppNetwork -VApp $myVapp -Direct -ParentOrgNetwork $myOrgNetwork
```

By default, the new vApp network has an enabled firewall.

Modify or Remove vApp Networks

Based on the type of the vApp network, you can configure various network settings, such as DNS, static IP pools, and firewalls. If you no longer need a vApp network, you can remove it.

Prerequisites

Verify that you are connected to a vCloud Director server.

Procedure

1 Retrieve the vApp for which you want to modify vApp networks.

```
$myVApp = Get-CIVApp -Name 'MyVApp'
```

2 Modify the settings for DNS and static IP pool for the vApp network named *MyVAppNetwork*.

```
Get-CIVAppNetwork -VApp $myVApp -Name 'MyVAppNetwork' | Set-CIVAppNetwork -PrimaryDns
10.17.0.94 -SecondaryDns 10.17.0.95 -DnsSuffix 'my.domain.com' -StaticIPPool "10.151.168.1 -
10.151.169.240"
```

3 (Optional) Remove *MyVAppNetwork*.

```
$myVApp | Get-CIVAppNetwork -Name 'MyVAppNetwork' | Remove-CIVAppNetwork
```

4 (Optional) Remove all isolated vApp networks for the vApp named *MyVapp*.

```
$myVApp | Get-CIVAppNetwork -ConnectionType Isolated | Remove-CIVAppNetwork
```

5 Retrieve the organization network named *MyOrgNetwork1*.

```
$myOrgNetwork1 = Get-OrgNetwork -Name 'MyOrgNetwork1'
```

6 Retrieve the organization network named *MyOrgNetwork2*.

```
$myOrgNetwork2 = Get-OrgNetwork -Name 'MyOrgNetwork2'
```

7 Redirect all vApp networks that connect to *MyOrgNetwork1* to connect to *MyOrgNetwork2*.

```
Get-CIVAppNetwork -ParentOrgNetwork $myOrgNetwork1 | Set-CIVAppNetwork -ParentOrgNetwork
$myOrgNetwork2 -NatEnabled $false -FirewallEnabled $false
```

The operation disables the firewall and NAT routing for all vApp networks that are connected to *MyOrgNetwork1*.

Index

U

V

W