**Manipulating items directly**

* Article
* 09/18/2021
* 5 minutes to read
* 4 contributors

The elements that you see in Windows PowerShell drives, such as the files and folders in the file system drives, and the registry keys in the Windows PowerShell registry drives, are called **items** in Windows PowerShell. The cmdlets for working with them item have the noun **Item** in their names.

The output of the Get-Command -Noun Item command shows that there are nine PowerShell item cmdlets.

PowerShellCopy

Get-Command -Noun Item

OutputCopy

CommandType Name Definition

----------- ---- ----------

Cmdlet Clear-Item Clear-Item [-Path] <String[]...

Cmdlet Copy-Item Copy-Item [-Path] <String[]>...

Cmdlet Get-Item Get-Item [-Path] <String[]> ...

Cmdlet Invoke-Item Invoke-Item [-Path] <String[...

Cmdlet Move-Item Move-Item [-Path] <String[]>...

Cmdlet New-Item New-Item [-Path] <String[]> ...

Cmdlet Remove-Item Remove-Item [-Path] <String[...

Cmdlet Rename-Item Rename-Item [-Path] <String>...

Cmdlet Set-Item Set-Item [-Path] <String[]> ...

**Creating new Items (New-Item)**

To create a new item in the file system, use the New-Item cmdlet. Include the **Path** parameter with path to the item, and the **ItemType** parameter with a value of file or directory.

For example, to create a new directory named New.Directory in the C:\Temp directory, type:

PowerShellCopy

New-Item -Path c:\temp\New.Directory -ItemType Directory

OutputCopy

Directory: Microsoft.Windows PowerShell.Core\FileSystem::C:\temp

Mode LastWriteTime Length Name

---- ------------- ------ ----

d---- 2006-05-18 11:29 AM New.Directory

To create a file, change the value of the **ItemType** parameter to file. For example, to create a file named file1.txt in the New.Directory directory, type:

PowerShellCopy

New-Item -Path C:\temp\New.Directory\file1.txt -ItemType file

OutputCopy

Directory: Microsoft.Windows PowerShell.Core\FileSystem::C:\temp\New.Directory

Mode LastWriteTime Length Name

---- ------------- ------ ----

-a--- 2006-05-18 11:44 AM 0 file1

You can use the same technique to create a new registry key. In fact, a registry key is easier to create because the only item type in the Windows registry is a key. (Registry entries are item **properties**.) For example, to create a key named "\_Test" in the CurrentVersion subkey, type:

PowerShellCopy

New-Item -Path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\\_Test

OutputCopy

Hive: Microsoft.PowerShell.Core\Registry::HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion

SKC VC Name Property

--- -- ---- --------

0 0 \_Test {}

When typing a registry path, be sure to include the colon (:) in the PowerShell drive names, HKLM: and HKCU:. Without the colon, PowerShell does not recognize the drive name in the path.

**Why Registry Values are not Items**

When you use the Get-ChildItem cmdlet to find the items in a registry key, you will never see actual registry entries or their values.

For example, the registry key HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run usually contains several registry entries that represent applications that run when the system starts.

However, when you use Get-ChildItem to look for child items in the key, all you will see is the **OptionalComponents** subkey of the key:

PowerShellCopy

Get-ChildItem HKLM:\Software\Microsoft\Windows\CurrentVersion\Run

OutputCopy

Hive: Microsoft.PowerShell.Core\Registry::HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run

SKC VC Name Property

--- -- ---- --------

3 0 OptionalComponents {}

Although it would be convenient to treat registry entries as items, you cannot specify a path to a registry entry in a way that ensures that it is unique. The path notation does not distinguish between the registry subkey named **Run** and the **(Default)** registry entry in the **Run** subkey. Furthermore, because registry entry names can contain the backslash character (\), if registry entries were items, then you could not use the path notation to distinguish a registry entry named Windows\CurrentVersion\Run from the subkey that is located in that path.

**Renaming existing Items (Rename-Item)**

To change the name of a file or folder, use the Rename-Item cmdlet. The following command changes the name of the file1.txt file to fileOne.txt.

PowerShellCopy

Rename-Item -Path C:\temp\New.Directory\file1.txt fileOne.txt

The Rename-Item cmdlet can change the name of a file or a folder, but it cannot move an item. The following command fails because it attempts to move the file from the New.Directory directory to the Temp directory.

PowerShellCopy

Rename-Item -Path C:\temp\New.Directory\fileOne.txt c:\temp\fileOne.txt

OutputCopy

Rename-Item : Cannot rename because the target specified is not a path.

At line:1 char:12

+ Rename-Item <<<< -Path C:\temp\New.Directory\fileOne c:\temp\fileOne.txt

**Moving Items (Move-Item)**

To move a file or folder, use the Move-Item cmdlet.

For example, the following command moves the New.Directory directory from the C:\temp directory to the root of the C: drive. To verify that the item was moved, include the **PassThru** parameter of the Move-Item cmdlet. Without **Passthru**, the Move-Item cmdlet does not display any results.

PowerShellCopy

Move-Item -Path C:\temp\New.Directory -Destination C:\ -PassThru

OutputCopy

Directory: Microsoft.Windows PowerShell.Core\FileSystem::C:\

Mode LastWriteTime Length Name

---- ------------- ------ ----

d---- 2006-05-18 12:14 PM New.Directory

**Copying Items (Copy-Item)**

If you are familiar with the copy operations in other shells, you might find the behavior of the Copy-Item cmdlet in Windows PowerShell to be unusual. When you copy an item from one location to another, Copy-Item does not copy its contents by default.

For example, if you copy the New.Directory directory from the C: drive to the C:\temp directory, the command succeeds, but the files in the New.Directory directory are not copied.

PowerShellCopy

Copy-Item -Path C:\New.Directory -Destination C:\temp

If you display the contents of C:\temp\New.Directory, you will find that it contains no files:

Copy

PS> Get-ChildItem -Path C:\temp\New.Directory

PS>

Why doesn't the Copy-Item cmdlet copy the contents to the new location?

The Copy-Item cmdlet was designed to be generic; it is not just for copying files and folders. Also, even when copying files and folders, you might want to copy only the container and not the items within it.

To copy all of the contents of a folder, include the **Recurse** parameter of the Copy-Item cmdlet in the command. If you have already copied the directory without its contents, add the **Force** parameter, which allows you to overwrite the empty folder.

PowerShellCopy

Copy-Item -Path C:\New.Directory -Destination C:\temp -Recurse -Force -Passthru

OutputCopy

Directory: Microsoft.Windows PowerShell.Core\FileSystem::C:\temp

Mode LastWriteTime Length Name

---- ------------- ------ ----

d---- 2006-05-18 1:53 PM New.Directory

Directory: Microsoft.Windows PowerShell.Core\FileSystem::C:\temp\New.Directory

Mode LastWriteTime Length Name

---- ------------- ------ ----

-a--- 2006-05-18 11:44 AM 0 file1

**Deleting Items (Remove-Item)**

To delete files and folders, use the Remove-Item cmdlet. Windows PowerShell cmdlets, such as Remove-Item, that can make significant, irreversible changes will often prompt for confirmation when you enter its commands. For example, if you try to remove the New.Directory folder, you will be prompted to confirm the command, because the folder contains files:

PowerShellCopy

Remove-Item C:\temp\New.Directory

OutputCopy

Confirm

The item at C:\temp\New.Directory has children and the -recurse parameter was not

specified. If you continue, all children will be removed with the item. Are you

sure you want to continue?

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help

(default is "Y"):

Because Yes is the default response, to delete the folder and its files, press the Enter key. To remove the folder without confirming, use the **Recurse** parameter.

PowerShellCopy

Remove-Item C:\temp\New.Directory -Recurse

**Executing Items (Invoke-Item)**

PowerShell uses the Invoke-Item cmdlet to perform a default action for a file or folder. This default action is determined by the default application handler in the registry; the effect is the same as if you double-click the item in File Explorer.

For example, suppose you run the following command:

PowerShellCopy

Invoke-Item C:\WINDOWS

An Explorer window that is located in C:\Windows appears, just as if you had double-clicked the C:\Windows folder.

If you invoke the Boot.ini file on a system prior to Windows Vista:

PowerShellCopy

Invoke-Item C:\boot.ini

If the .ini file type is associated with Notepad, the boot.ini file opens in Notepad.