**Working with Registry Keys**

* Article
* 10/08/2021
* 3 minutes to read
* 2 contributors

Because registry keys are items on PowerShell drives, working with them is very similar to working with files and folders. One critical difference is that every item on a registry-based PowerShell drive is a container, just like a folder on a file system drive. However, registry entries and their associated values are properties of the items, not distinct items.

**Listing All Subkeys of a Registry Key**

You can show all items directly within a registry key by using Get-ChildItem. Add the optional **Force** parameter to display hidden or system items. For example, this command displays the items directly within PowerShell drive HKCU:, which corresponds to the HKEY\_CURRENT\_USER registry hive:

PowerShellCopy

Get-ChildItem -Path HKCU:\ | Select-Object Name

OutputCopy

Hive: Microsoft.PowerShell.Core\Registry::HKEY\_CURRENT\_USER

Name

----

HKEY\_CURRENT\_USER\AppEvents

HKEY\_CURRENT\_USER\Console

HKEY\_CURRENT\_USER\Control Panel

HKEY\_CURRENT\_USER\DirectShow

HKEY\_CURRENT\_USER\dummy

HKEY\_CURRENT\_USER\Environment

HKEY\_CURRENT\_USER\EUDC

HKEY\_CURRENT\_USER\Keyboard Layout

HKEY\_CURRENT\_USER\MediaFoundation

HKEY\_CURRENT\_USER\Microsoft

HKEY\_CURRENT\_USER\Network

HKEY\_CURRENT\_USER\Printers

HKEY\_CURRENT\_USER\Software

HKEY\_CURRENT\_USER\System

HKEY\_CURRENT\_USER\Uninstall

HKEY\_CURRENT\_USER\WXP

HKEY\_CURRENT\_USER\Volatile Environment

These are the top-level keys visible under HKEY\_CURRENT\_USER in the Registry Editor (regedit.exe).

You can also specify this registry path by specifying the registry provider's name, followed by ::. The registry provider's full name is Microsoft.PowerShell.Core\Registry, but this can be shortened to just Registry. Any of the following commands will list the contents directly under HKCU:.

PowerShellCopy

Get-ChildItem -Path Registry::HKEY\_CURRENT\_USER

Get-ChildItem -Path Microsoft.PowerShell.Core\Registry::HKEY\_CURRENT\_USER

Get-ChildItem -Path Registry::HKCU

Get-ChildItem -Path Microsoft.PowerShell.Core\Registry::HKCU

Get-ChildItem HKCU:

These commands list only the directly contained items, much like using DIR in **Cmd.exe** or ls in a UNIX shell. To show contained items, you need to specify the **Recurse** parameter. To list all registry keys in HKCU:, use the following command.

PowerShellCopy

Get-ChildItem -Path HKCU:\ -Recurse

Get-ChildItem can perform complex filtering capabilities through its **Path**, **Filter**, **Include**, and **Exclude** parameters, but those parameters are typically based only on name. You can perform complex filtering based on other properties of items by using the Where-Object cmdlet. The following command finds all keys within HKCU:\Software that have no more than one subkey and also have exactly four values:

PowerShellCopy

Get-ChildItem -Path HKCU:\Software -Recurse |

Where-Object {($\_.SubKeyCount -le 1) -and ($\_.ValueCount -eq 4) }

**Copying Keys**

Copying is done with Copy-Item. The following example copies the CurrentVersion subkey of HKLM:\SOFTWARE\Microsoft\Windows\ and all of its properties to HKCU:\.

PowerShellCopy

Copy-Item -Path 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion' -Destination HKCU:

If you examine this new key in the registry editor or by using Get-ChildItem, you notice that you do not have copies of the contained subkeys in the new location. In order to copy all of the contents of a container, you need to specify the **Recurse** parameter. To make the preceding copy command recursive, you would use this command:

PowerShellCopy

Copy-Item -Path 'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion' -Destination HKCU: -Recurse

You can still use other tools you already have available to perform filesystem copies. Any registry editing tools—including reg.exe, regini.exe, regedit.exe, and COM objects that support registry editing, such as **WScript.Shell** and WMI's **StdRegProv** class can be used from within Windows PowerShell.

**Creating Keys**

Creating new keys in the registry is simpler than creating a new item in a file system. Because all registry keys are containers, you do not need to specify the item type; you simply supply an explicit path, such as:

PowerShellCopy

New-Item -Path HKCU:\Software\_DeleteMe

You can also use a provider-based path to specify a key:

PowerShellCopy

New-Item -Path Registry::HKCU\Software\_DeleteMe

**Deleting Keys**

Deleting items is essentially the same for all providers. The following commands will silently remove items:

PowerShellCopy

Remove-Item -Path HKCU:\Software\_DeleteMe

Remove-Item -Path 'HKCU:\key with spaces in the name'

**Removing All Keys Under a Specific Key**

You can remove contained items by using Remove-Item, but you will be prompted to confirm the removal if the item contains anything else. For example, if we attempt to delete the HKCU:\CurrentVersion subkey we created, we see this:

PowerShellCopy

Remove-Item -Path HKCU:\CurrentVersion

OutputCopy

Confirm

The item at HKCU:\CurrentVersion\AdminDebug 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"):

To delete contained items without prompting, specify the **Recurse** parameter:

PowerShellCopy

Remove-Item -Path HKCU:\CurrentVersion -Recurse

If you wanted to remove all items within HKCU:\CurrentVersion but not HKCU:\CurrentVersion itself, you could instead use:

PowerShellCopy

Remove-Item -Path HKCU:\CurrentVersion\\* -Recurse