


Understanding pipelines

08/23/20183 minutes to read 

In this article

The PowerShell pipeline

Objects in the pipeline

Pipelines act like a series of connected segments of pipe. Items moving along the pipeline pass through each segment. To create a pipeline in PowerShell, you connect commands together with the pipe operator "|". The output of each command is used as input to the next command.

The notation used for pipelines is similar to the notation used in other shells. At first glance, it may not be apparent how pipelines are different in PowerShell. Although you see text on the screen, PowerShell pipes objects, not text, between commands.

The PowerShell pipeline

Pipelines are arguably the most valuable concept used in command-line interfaces. When used properly, pipelines reduce the effort of using complex commands and make it easier to see the flow of work for the commands. Each command in a pipeline (called a pipeline element) passes its output to the next command in the pipeline, item-by-item. Commands don't have to handle more than one item at a time. The result is reduced resource consumption and the ability to begin getting the output immediately.

For example, if you use the `Out-Host` cmdlet to force a page-by-page display of output from another command, the output looks just like the normal text displayed on the screen, broken up into pages:

Get-ChildItem -Path C:\WINDOWS\System32 |
Out-Host -Paging

Output

Copy

Directory: C:\WINDOWS\system32

Mode		LastWriteTime
Length	Name	
----		-----
d-----		
0409	4/12/2018 2:15 AM	
d-----	5/13/2018 11:31 PM	
1033		
d-----	4/11/2018 4:38 PM	
AdvancedInstallers		
d-----	5/13/2018 11:13 PM	
af-ZA		
d-----	5/13/2018 11:13 PM	
am-et		
d-----	4/11/2018 4:38 PM	
AppLocker		
d-----	5/13/2018 11:31 PM	
appmgmt		
d-----	7/11/2018 2:05 AM	
appraiser		
d---s-	4/12/2018 2:20 AM	
AppV		
d-----	5/13/2018 11:10 PM	
ar-SA		
d-----	5/13/2018 11:13 PM	
as-IN		
d-----	8/14/2018 9:03 PM	
az-Latn-AZ		

d-----	5/13/2018	11:13 PM
be-BY		
d-----	5/13/2018	11:10 PM
BestPractices		
d-----	5/13/2018	11:10 PM
bg-BG		
d-----	5/13/2018	11:13 PM
bn-BD		
d-----	5/13/2018	11:13 PM
bn-IN		
d-----	8/14/2018	9:03 PM
Boot		
d-----	8/14/2018	9:03 PM
bs-Latn-BA		
d-----	4/11/2018	4:38 PM
Bthprops		
d-----	4/12/2018	2:19 AM
ca-ES		
d-----	8/14/2018	9:03 PM
ca-ES-valencia		
d-----	5/13/2018	10:46 PM
CatRoot		
d-----	8/23/2018	5:07 PM
catroot2		
<SPACE> next page; <CR> next line; Q quit		
...		

Paging also reduces CPU utilization because processing transfers to the Out-Host cmdlet when it has a complete page ready to display. The cmdlets that precede it in the pipeline pause execution until the next page of output is available.

You can see how piping impacts CPU and memory usage in the Windows Task Manager by comparing the following commands:

- Get-ChildItem C:\Windows -Recurse

- `Get-ChildItem C:\Windows -Recurse | Out-Host -Paging`

Note

The **Paging** parameter is not supported by all PowerShell hosts. For example, when you try to use the **Paging** parameter in the PowerShell ISE, you see the following error:

Output

Copy

```
out-lineoutput : The method or operation
is not implemented.
At line:1 char:1
+ Get-ChildItem C:\Windows -Recurse |
Out-Host -Paging
+ ~~~~~
    + CategoryInfo          : NotSpeci-
fied: (:) [out-lineoutput], NotImplemented-
Exception
    + FullyQualifiedErrorId :
System.NotImplementedException,Microsoft-
.PowerShell.Commands.OutLineOutputCommand
```

Objects in the pipeline

When you run a cmdlet in PowerShell, you see text output because it is necessary to represent objects as text in a console window. The text output may not display all of the properties of the object being output.

For example, consider the `Get-Location` cmdlet. If you run `Get-Location` while your current location is the root of the

C drive, you see the following output:

Copy

```
PS> Get-Location
```

```
Path
```

```
----
```

```
C:\
```

The text output is a summary of information, not a complete representation of the object returned by `Get-Location`. The heading in the output is added by the process that formats the data for onscreen display.

When you pipe the output to the `Get-Member` cmdlet you get information about the object returned by `Get-Location`.

PowerShell

Copy

[Get-Location](#) | [Get-Member](#)

Output

Copy

```
TypeName: System.Management.Automation.-  
PathInfo
```

Name	MemberType	Definition
----	-----	-----
Equals	Method	bool Equals(System.Object obj)
GetHashCode	Method	int GetHashCode()
GetType	Method	type GetType()
ToString	Method	string ToString()
Drive	Property	System.Management.Au-

```
tomation.PSDriveInfo Drive {get;}
Path          Property    string Path {get;}
Provider      Property    System.Management.Au-
tomation.ProviderInfo Provider {get;}
ProviderPath  Property    string ProviderPath
{get;}
```

Get-Location returns a **PathInfo** object that contains the current path and other information.