



python



PowerShell

FPDF Library  
PDF generator

*Full credit is given to the above companies including the OS that this PDF file was generated!*

### **PowerShell Get-Help on command 'Get-Counter'**

**PS C:\Users\wahid> Get-Help Get-Counter**

#### NAME

Get-Counter

#### SYNOPSIS

Gets performance counter data from local and remote computers.

#### SYNTAX

```
Get-Counter [[-Counter] <System.String[]>] [-ComputerName <System.String[]>]  
[-Continuous] [-MaxSamples <System.Int64>] [-SampleInterval <System.Int32>]  
[<CommonParameters>]
```

```
Get-Counter [-ListSet] <System.String[]> [-ComputerName <System.String[]>]  
[<CommonParameters>]
```

#### DESCRIPTION

The `Get-Counter` cmdlet gets performance counter data directly from the performance monitoring instrumentation in the Windows family of operating systems. `Get-Counter` gets performance data from a local computer or remote computers.

You can use the ``Get-Counter`` parameters to specify one or more computers, list the performance counter sets and the instances they contain, set the sample intervals, and specify the maximum number of samples. Without parameters, ``Get-Counter`` gets performance counter data for a set of system counters.

Many counter sets are protected by access control lists (ACL). To see all counter sets, open PowerShell with the Run as administrator option.

> [!NOTE] > Performance counter names are localized. The examples shown here use the English names of the > performance objects, counters, and instances. The names will be different on a system that uses > another language. Use the ``Get-Counter -ListSet`` command to see the localized names.

## PARAMETERS

`-ComputerName <System.String[]>`

Specifies one computer name or a comma-separated array of remote computer names. Use the NetBIOS name, an IP address, or the computer's fully qualified domain name.

To get performance counter data from the local computer, exclude the `ComputerName` parameter. For output such as a `ListSet` that contains the `MachineName` column, a dot (``.`) indicates the local computer.

``Get-Counter`` doesn't rely on PowerShell remoting. You can use the `ComputerName` parameter even if your computer isn't configured to run remote commands.

`-Continuous <System.Management.Automation.SwitchParameter>`

When `Continuous` is specified, ``Get-Counter`` gets samples until you press `<kbd>CTRL</kbd>+<kbd>C</kbd>`. Samples are obtained every second for each

specified performance counter. Use the `SampleInterval` parameter to increase the interval between continuous samples.

`-Counter <System.String[]>`

Specifies the path to one or more counter paths. Paths are input as a comma-separated array, a variable, or values from a text file. You can send counter path strings down the pipeline to ``Get-Counter``.

Counter paths use the following syntax:

```
`\ComputerName\CounterSet(Instance)\CounterName`
```

```
`\CounterSet(Instance)\CounterName`
```

For example:

```
`\Server01\Processor(*)\% User Time`
```

```
`\Processor(*)\% User Time`
```

The ``\ComputerName`` is optional in a performance counter path. If the counter path doesn't include the computer name, ``Get-Counter`` uses the local computer.

An asterisk (``*``) in the instance is a wildcard character to get all instances of the counter.

`-ListSet <System.String[]>`

Lists the performance counter sets on the computers. Use an asterisk (``*``) to specify all counter sets. Enter one name or a comma-separated string of counter set names. You can send counter set names down the pipeline.

To get a counter sets formatted counter paths, use the `ListSet` parameter.

The Paths and PathsWithInstances properties of each counter set contain the individual counter paths formatted as a string.

You can save the counter path strings in a variable or use the pipeline to send the string to another `Get-Counter` command.

For example to send each Processor counter path to `Get-Counter`:

```
`Get-Counter -ListSet Processor | Get-Counter`
```

#### **-MaxSamples <System.Int64>**

Specifies the number of samples to get from each specified performance counter. To get a constant stream of samples, use the Continuous parameter.

If the MaxSamples parameter isn't specified, `Get-Counter` only gets one sample for each specified counter.

To collect a large data set, run `Get-Counter` as a PowerShell background job. For more information, see `about_Jobs` (`../Microsoft.PowerShell.Core/About/about_Jobs.md`).

#### **-SampleInterval <System.Int32>**

Specifies the number of seconds between samples for each specified performance counter. If the SampleInterval parameter isn't specified, `Get-Counter` uses a one-second interval.

#### **<CommonParameters>**

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see `about_CommonParameters` (<https://go.microsoft.com/fwlink/?LinkID=113216>).

Get-Counter -ListSet \*

CounterSetName : Processor

MachineName : .

CounterSetType : MultiInstance

Description : The Processor performance object consists of counters that measure aspects ...

computer that performs arithmetic and logical computations, initiates ...

computer can have multiple processors. The processor object represents ...

Paths : {\Processor(\*)\% Processor Time, \Processor(\*)\% User Time, ...

PathsWithInstances : {\Processor(0)\% Processor Time, \Processor(1)\% Processor Time, ...

Counter : {\Processor(\*)\% Processor Time, \Processor(\*)\% User Time, ...

`Get-Counter` uses the ListSet parameter with an asterisk (`\*`) to get the list of counter sets. The dot (`.`) in the MachineName column represents the local computer.

----- Example 2: Specify the SampleInterval and MaxSamples -----

Get-Counter -Counter "\Processor(\_Total)\% Processor Time" -SampleInterval 2 -MaxSamples 3

Timestamp	CounterSamples
-----------	----------------

-----

-----

6/18/2019 14:39:56	\\Computer01\processor(_total)\% processor time :
	20.7144271584086

6/18/2019 14:39:58	\\Computer01\processor(_total)\% processor time :
--------------------	---

10.4391790575511

```
6/18/2019 14:40:01    \\Computer01\processor(_total)\% processor time :
                        37.5968799396998
```

`Get-Counter` uses the Counter parameter to specify the counter path  
`\\Processor(\_Total)\% Processor Time`. The SampleInterval parameter sets a  
two-second interval to check the counter. MaxSamples determines that three is  
the maximum number of times to check the counter.

----- Example 3: Get continuous samples of a counter -----

```
Get-Counter -Counter "\\Processor(_Total)\% Processor Time" -Continuous
```

Timestamp	CounterSamples
-----------	----------------

-----	-----
6/19/2019 15:35:03	\\Computer01\processor(_total)\% processor time :
	43.8522842937022

6/19/2019 15:35:04	\\Computer01\processor(_total)\% processor time :
	29.7896844697383

6/19/2019 15:35:05	\\Computer01\processor(_total)\% processor time :
	29.4962645638135

6/19/2019 15:35:06	\\Computer01\processor(_total)\% processor time :
	25.5901500127408

`Get-Counter` uses the Counter parameter to specify the `\\Processor\  
Processor Time` counter. The Continuous parameter specifies to get samples  
every second until the command is stopped with <kbd>CTRL</kbd>+<kbd>C</kbd>.

----- Example 4: Alphabetical list of counter sets -----

```
Get-Counter -ListSet * | Sort-Object -Property CounterSetName | Format-Table
```

-AutoSize

CounterSetName	MachineName	CounterSetType	Description
-----	-----	-----	
.NET CLR Data	.	SingleInstance	.Net CLR Data
.NET Data Provider for SqlServer System.Data.SqlClient	.	SingleInstance	Counters for
AppV Client Streamed Data Percentage streamed to disk ...	.	SingleInstance	Size of data
Authorization Manager Applications Counters for ...	.	SingleInstance	The set of
BitLocker Drive Encryption ...	.	MultInstance	BitLocker
Bluetooth Device related to a remote ...	.	SingleInstance	Counters
Cache performance object ...	.	SingleInstance	The Cache
Client Side Caching counters for SMB ...	.	SingleInstance	Performance

`Get-Counter` uses the ListSet parameter with an asterisk (`\*`) to get a complete list of counter sets. The CounterSet objects are sent down the pipeline. `Sort-Object` uses the Property parameter to specify that the objects are sorted by CounterSetName . The objects are sent down the pipeline to `Format-Table`. The AutoSize parameter adjusts the column widths to minimize truncation.

The dot (`. `) in the MachineName column represents the local computer.

----- Example 5: Run a background job to get counter data -----

```
Start-Job -ScriptBlock {Get-Counter -Counter "\LogicalDisk(_Total)\% Free  
Space" -MaxSamples 1000}
```

```

Id   Name PSJobTypeName State HasMoreData Location Command
--   -
1   Job1 BackgroundJob Running True    localhost Get-Counter
-Counter

```

`Start-Job` uses the ScriptBlock parameter to run a `Get-Counter` command.

`Get-Counter` uses the Counter parameter to specify the counter path

`\LogicalDisk(\_Total)\% Free Space`. The MaxSamples parameter specifies to get 1000 samples of the counter.

----- Example 6: Get counter data from multiple computers -----

```
$DiskReads = "\LogicalDisk(C:)\Disk Reads/sec"
```

```
$DiskReads | Get-Counter -ComputerName Server01, Server02 -MaxSamples 10
```

```

Timestamp          CounterSamples
-----
6/21/2019 10:51:04  \\Server01\logicaldisk(c:)\disk reads/sec :
                    0

                    \\Server02\logicaldisk(c:)\disk reads/sec :
                    0.983050344269146

```

The `\$DiskReads` variable stores the `\LogicalDisk(C:)\Disk Reads/sec` counter path. The `\$DiskReads` variable is sent down the pipeline to `Get-Counter`.

Counter is the first position parameter and accepts the path stored in

`\$DiskReads`. ComputerName specifies the two computers and MaxSamples specifies to get 10 samples from each computer.

Example 7: Get a counter's instance values from multiple random computers

```
$Servers = Get-Random (Get-Content -Path C:\Servers.txt) -Count 50
```

```
$Counter = "\Processor(*)\% Processor Time"
```

```
Get-Counter -Counter $Counter -ComputerName $Servers
```



Timestamp	CounterSamples
-----	-----
6/20/2019 12:20:35	\\Server01\processor(0)\% processor time : 6.52610319637854
	\\Server01\processor(1)\% processor time : 3.41030663625782
	\\Server01\processor(2)\% processor time : 9.64189975649925
	\\Server01\processor(3)\% processor time : 1.85240835619747
	\\Server01\processor(_total)\% processor time : 5.35768447160776

The `Get-Random` cmdlet uses `Get-Content` to select 50 random computer names from the `Servers.txt` file. The remote computer names are stored in the `\$Servers` variable. The `\\Processor(\*)\% Processor Time` counter's path is stored in the `\$Counter` variable. `Get-Counter` uses the Counter parameter to specify the counters in the `\$Counter` variable. The ComputerName parameter specifies the computer names in the `\$Servers` variable.

- Example 8: Use the Path property to get formatted path names -

```
(Get-Counter -ListSet Memory).Paths | Where-Object { $_ -like "*Cache*" }
```

\\Memory\Cache Faults/sec

\\Memory\Cache Bytes

\\Memory\Cache Bytes Peak

\\Memory\System Cache Resident Bytes

\\Memory\Standby Cache Reserve Bytes

\\Memory\Standby Cache Normal Priority Bytes

\Memory\Standby Cache Core Bytes

\Memory\Long-Term Average Standby Cache Lifetime (s)

`Get-Counter` uses the ListSet parameter to specify the Memory counter set.

The command is enclosed in parentheses so that the Paths property returns each path as a string. The objects are sent down the pipeline to `Where-Object`.

`Where-Object` uses the variable `\$\_` to process each object and uses the `-like` operator to find matches for the string `Cache`. The asterisks (`\*`) are wildcards for any characters.

Example 9: Use the PathsWithInstances property to get formatted path names

```
(Get-Counter -ListSet PhysicalDisk).PathsWithInstances
```

\PhysicalDisk(0 C:)\Current Disk Queue Length

\PhysicalDisk(\_Total)\Current Disk Queue Length

\PhysicalDisk(0 C:)\% Disk Time

\PhysicalDisk(\_Total)\% Disk Time

\PhysicalDisk(0 C:)\Avg. Disk Queue Length

\PhysicalDisk(\_Total)\Avg. Disk Queue Length

\PhysicalDisk(0 C:)\% Disk Read Time

\PhysicalDisk(\_Total)\% Disk Read Time

`Get-Counter` uses the ListSet parameter to specify the PhysicalDisk counter set. The command is enclosed in parentheses so that the PathsWithInstances property returns each path instance as a string.

Example 10: Get a single value for each counter in a counter set

```
$MemCounters = (Get-Counter -ListSet Memory).Paths
```

```
Get-Counter -Counter $MemCounters
```

Timestamp	CounterSamples
-----------	----------------

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

6/19/2019 12:05:00	\\Computer01\memory\page faults/sec :
--------------------	---------------------------------------

868.772077545597

\\Computer01\memory\available bytes :  
9031176192

\\Computer01\memory\committed bytes :  
8242982912

\\Computer01\memory\commit limit :  
19603333120

`Get-Counter` uses the ListSet parameter to specify the Memory counter set.  
The command is enclosed in parentheses so that the Paths property returns each  
path as a string. The paths are stored in the `\$MemCounters` variable.

`Get-Counter` uses the Counter parameter to specify the counter paths in the  
`\$MemCounters` variable.

----- Example 11: Display an object's property values -----

```
$Counter = "\\Server01\Process(Idle)\% Processor Time"  
$Data = Get-Counter $Counter  
$Data.CounterSamples | Format-List -Property *
```

```
Path      : \\Server01\process(idle)\% processor time  
InstanceName  : idle  
CookedValue   : 198.467899571389  
RawValue      : 14329160321003  
SecondValue   : 128606459528326201  
MultipleCount : 1  
CounterType   : Timer100Ns  
Timestamp     : 6/19/2019 12:20:49  
Timestamp100NSec : 128606207528320000  
Status        : 0  
DefaultScale  : 0
```

TimeBase : 10000000

The counter path is stored in the ``$Counter`` variable. ``Get-Counter`` gets one sample of the counter values and stores the results in the ``$Data`` variable.

The ``$Data`` variable uses the `CounterSamples` property to get the object's properties. The object is sent down the pipeline to ``Format-List``. The `Property` parameter uses an asterisk (``*``) wildcard to select all the properties.

----- Example 12: Performance counter array values -----

```
$Counter = Get-Counter -Counter "\Processor(*)\% Processor Time"
$Counter.CounterSamples[0]
```

Path	InstanceName	CookedValue
----	-----	-----
\\Computer01\processor(0)\% processor time	0	1.33997091699662

``Get-Counter`` uses the `Counter` parameter to specify the counter ``\Processor(*)\% Processor Time``. The values are stored in the ``$Counter`` variable. ``$Counter.CounterSamples[0]`` displays the array value for the first counter value.

----- Example 13: Compare performance counter values -----

```
$Counter = Get-Counter -Counter "\Processor(*)\% Processor Time"
$Counter.CounterSamples | Where-Object { $_.CookedValue -lt "20" }
```

Path	InstanceName	CookedValue
----	-----	-----
\\Computer01\processor(0)\% processor time	0	12.6398025240208
\\Computer01\processor(1)\% processor time	1	15.7598095767344

``Get-Counter`` uses the `Counter` parameter to specify the counter ``\Processor(*)\% Processor Time``. The values are stored in the ``$Counter``

variable. The objects stored in `\$Counter.CounterSamples` are sent down the pipeline. `Where-Object` uses a script block to compare each objects value against a specified value of `20`. The `\$\_CookedValue` is a variable for the current object in the pipeline. Counters with a CookedValue that is less than 20 are displayed.

----- Example 14: Sort performance counter data -----

```
$Procs = Get-Counter -Counter "\Process(*)\% Processor Time"
$Procs.CounterSamples | Sort-Object -Property CookedValue -Descending |
  Format-Table -Property Path, InstanceName, CookedValue -AutoSize
```

Path	InstanceName	CookedValue
-----	-----	-----
\\Computer01\process(_total)\% processor time	_total	395.464129650573
\\Computer01\process(idle)\% processor time	idle	389.356575524695
\\Computer01\process(mssense)\% processor time	mssense	3.05377706293879
\\Computer01\process(csrss#1)\% processor time	csrss	1.52688853146939
\\Computer01\process(microsoftedgecp#10)\% processor time	microsoftedgecp	1.52688853146939
\\Computer01\process(runtimebroker#5)\% processor time	runtimebroker	0
\\Computer01\process(settingsynchost)\% processor time	settingsynchost	0
\\Computer01\process(microsoftedgecp#16)\% processor time	microsoftedgecp	0

`Get-Counter` uses the Counter parameter to specify the `\\Process(\*)\%`

Processor Time` counter for all the processes on the local computer. The result is stored in the `\$Procs` variable. The `\$Procs` variable with the CounterSamples property sends the PerformanceCounterSample objects down the pipeline. `Sort-Object` uses the Property parameter to sort the objects by CookedValue in Descending order. `Format-Table` uses the Property parameter to select the columns for the output. The AutoSize parameter adjusts the column widths to minimize truncation.

## REMARKS

To see the examples, type: "get-help Get-Counter -examples".

For more information, type: "get-help Get-Counter -detailed".

For technical information, type: "get-help Get-Counter -full".

For online help, type: "get-help Get-Counter -online"