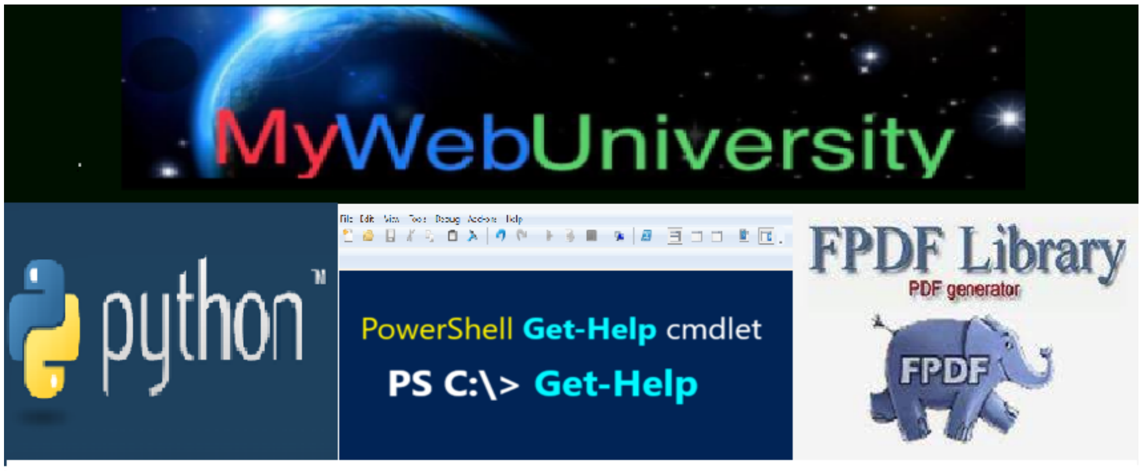

Full credit is given to all the above companies including the Operating System that this PDF file was generated!

Windows PowerShell Get-Help on Cmdlet 'Import-Counter'

PS:\>Get-HELP Import-Counter -Full

NAME

Import-Counter

SYNOPSIS

Imports performance counter log files and creates the objects that represent each counter sample in the log.

SYNTAX

Import-Counter [-Path] <System.String[]> [-Counter <System.String[]>] [-EndTime <System.DateTime>] [-MaxSamples <System.Int64>] [-StartTime <System.DateTime>]

[<CommonParameters>]

Import-Counter [-Path] <System.String[]> -ListSet <System.String[]> [<CommonParameters>]

Import-Counter [-Path] <System.String[]> [-Summary] [<CommonParameters>]

DESCRIPTION

The `Import-Counter` cmdlet imports performance counter data from performance counter log files and creates objects for each counter sample in the file. The

PerformanceCounterSampleSet objects that it creates are identical to the objects that `Get-Counter` returns when it collects performance counter data.

You can import data from comma-separated value (`.csv`), tab-separated value (`.tsv`), and binary performance log (`.blg`) performance log files. If you are using
`.blg` files, you can import up to 32 files in each command. You can use the parameters of `Import-Counter` to filter the data that you import.

Along with the `Get-Counter` and `Export-Counter` cmdlets, this feature lets you collect, export, import, combine, filter, manipulate, and re-export performance
counter data within Windows PowerShell.

PARAMETERS

-Counter <System.String[]>

Specifies, as a string array, the performance counters. By default, `Import-Counter` imports all data from all counters in the input files. Enter one or more
counter paths. Wildcards are permitted in the Instance part of the path.

Each counter path has the following format. The `ComputerName` value is required in the path. For instance:

- `\<ComputerName><CounterSet>(<Instance>)<CounterName>`

For example:

- `\Server01\Processor(2)\% User Time`

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

Required?	false
Position?	named
Default value	All counter
Accept pipeline input?	False

Accept wildcard characters? true

-EndTime <System.DateTime>

Specifies an end date and time that this cmdlet imports counter data between the StartTime and this parameter timestamps. Enter a DateTime object, such as one

created by the `Get-Date` cmdlet. By default, `Import-Counter` imports all counter data in the files specified by the Path parameter.

Required?   false

Position?   named

Default value   No end time

Accept pipeline input?   False

Accept wildcard characters? false

-ListSet <System.String[]>

Specifies the performance counter sets that are represented in the exported files. Commands with this parameter do not import any data.

Enter one or more counter set names. Wildcards are permitted. To get all counter sets in the file, type `Import-Counter -ListSet *`.

Required?   true

Position?   named

Default value   None

Accept pipeline input?   False

Accept wildcard characters? true

-MaxSamples <System.Int64>

Specifies the maximum number of samples of each counter to import. By default, `Get-Counter` imports all of the data in the files specified by the Path parameter.

Required?   false

Position?   named

Default value   No maximum

Accept pipeline input?   False

Accept wildcard characters? false

-Path <System.String[]>

Specifies the file paths of the files to be imported. This parameter is required.

Enter the path and file name of a, `.csv`, `.tsv`, or `.blg` file that you exported by using the `Export-Counter` cmdlet. You can specify only one `.csv` or

`.tsv` file, but you can specify multiple `.blg` files (up to 32) in each command. You can also pipe file path strings (in quotation marks) to `Import-Counter`.

Required?   true

Position?   1

Default value   None

Accept pipeline input?   True (ByPropertyName, ByValue)

Accept wildcard characters? true

-StartTime <System.DateTime>

Specifies the start date and time in which this cmdlet gets counter data. Enter a DateTime object, such as one created by the `Get-Date` cmdlet. By default,

`Import-Counter` imports all counter data in the files specified by the Path parameter.

Required?   false

Position?   named

Default value   No start time

Accept pipeline input?   False

Accept wildcard characters? false

-Summary <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets a summary of the imported data, instead of getting individual counter data samples.

Required?   false

Position?   named

Default value   False

Accept pipeline input?   False

Accept wildcard characters? false

<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).

INPUTS

System.String

You can pipe performance counter log paths to this cmdlet.

OUTPUTS

Microsoft.PowerShell.Commands.GetCounter.PerformanceCounterSampleSet, Microsoft.PowerShell.Commands.GetCounter.CounterSet,

Microsoft.PowerShell.Commands.GetCounter.CounterFileInfo

This cmdlet returns a Microsoft.PowerShell.Commands.GetCounter.PerformanceCounterSampleSet . If you use the ListSet parameter, this cmdlet returns a

Microsoft.PowerShell.Commands.GetCounter.CounterSet object. If you use the Summary parameter, this cmdlet returns a

Microsoft.PowerShell.Commands.GetCounter.CounterFileInfo object.

NOTES

- This cmdlet does not have a ComputerName parameter. However, if the computer is configured for Windows PowerShell remoting, you can use the `Invoke-Command`

cmdlet to run an `Import-Counter` command on a remote computer.

-------- Example 1: Import all counter data from a file --------

```
$data = Import-Counter -Path ProcessorData.csv
```

This command imports all counter data from the `ProcessorData.csv` file into the `$data` variable.

----- Example 2: Import specific counter data from a file -----

```
$i = Import-Counter -Path "ProcessorData.blg" -Counter "\\SERVER01\Processor(_Total)\Interrupts/sec"
```

This command imports only the "Processor(_total)\Interrupts/sec" counter data from the `ProcessorData.blg` file into the `$i` variable.

Example 3: Select data from a performance counter then export it to a file

```
$data = Import-Counter .\ProcessorData.blg
$data[0].CounterSamples | Format-Table -Property Path

Path
----
\\SERVER01\Processor(_Total)\DPC Rate
\\SERVER01\Processor(1)\DPC Rate
\\SERVER01\Processor(0)\DPC Rate
\\SERVER01\Processor(_Total)\% Idle Time
\\SERVER01\Processor(1)\% Idle Time
\\SERVER01\Processor(0)\% Idle Time
\\SERVER01\Processor(_Total)\% C3 Time
\\SERVER01\Processor(1)\% C3 Time

$intCtrs = $Data[0].Countersamples | Where-Object {$_.Path -like "*Interrupts/sec"} | ForEach-Object {$_.Path}
$intCtrs

\\SERVER01\Processor(_Total)\Interrupts/sec
\\SERVER01\Processor(1)\Interrupts/sec
```

```
\\SERVER01\Processor(0)\Interrupts/sec


$i = Import-Counter -Path .\ProcessorData.blg -Counter $intCtrs
$i | Export-Counter -Path .\Interrupts.csv -Format CSV
```

The first command uses `Import-Counter` to import all of the performance counter data from the `ProcessorData.blg` files. The command saves the data in the `$data`
variable.

The second command displays the counter paths in the `$data` variable. To get the display shown in the command output, the example uses the `Format-Table` cmdlet to
format as a table the counter paths of the first counter in the `$data` variable.

The third command gets the counter paths that end in `Interrupts/sec` and saves the paths in the `$intCtrs` variable. It uses the `Where-Object` cmdlet to filter the
counter paths and the `ForEach-Object` cmdlet to get only the value of the Path property of each selected path object.

The fourth command displays the selected counter paths in the `$intCtrs` variable.

The fifth command uses the `Import-Counter` cmdlet to import the data. It uses the `$intCtrs` variable as the value of the Counter parameter to import only data for
the counter paths in `$intCtrs`.

The sixth command uses the `Export-Counter` cmdlet to export the data to the `Interrupts.csv` file.

Example 4: Display all counter paths in a group of imported counter sets

```
Import-Counter -Path ProcessorData.csv -ListSet *


CounterSetName     : Processor
MachineName        : \\SERVER01
CounterSetType     : MultiInstance
Description        :
Paths              : {\\SERVER01\Processor(*)\DPC Rate, \\SERVER01\Processor(*)\% Idle Time, \\SERVER01
\Processor(*)\% C3 Time, \\SERVER01\Processor(*)\% Interrupt Time...}

PathsWithInstances : {\\SERVER01\Processor(_Total)\DPC Rate, \\SERVER01\Processor(1)\DPC Rate, \\SERVER01

\Processor(0)\DPC Rate, \\SERVER01\Processor(_Total)\% Idle Time...}

Counter            : {\\SERVER01\Processor(*)\DPC Rate, \\SERVER01\Processor(*)\% Idle Time, \\SERVER01

\Processor(*)\% C3 Time, \\SERVER01\Processor(*)\% Interrupt Time...}


Import-Counter -Path ProcessorData.csv -ListSet * | ForEach-Object {$_.Paths}


\\SERVER01\Processor(*)\DPC Rate
\\SERVER01\Processor(*)\% Idle Time
\\SERVER01\Processor(*)\% C3 Time
\\SERVER01\Processor(*)\% Interrupt Time
\\SERVER01\Processor(*)\% C2 Time
\\SERVER01\Processor(*)\% User Time
\\SERVER01\Processor(*)\% C1 Time
\\SERVER01\Processor(*)\% Processor Time
\\SERVER01\Processor(*)\C1 Transitions/sec
\\SERVER01\Processor(*)\% DPC Time
\\SERVER01\Processor(*)\C2 Transitions/sec
\\SERVER01\Processor(*)\% Privileged Time
\\SERVER01\Processor(*)\C3 Transitions/sec
\\SERVER01\Processor(*)\DPCs Queued/sec
\\SERVER01\Processor(*)\Interrupts/sec
```

The first command uses the ListSet parameter of the `Import-Counter` cmdlet to get all of the counter sets that are represented in a counter data file.

The second command gets all of the counter paths from the list set.

-- Example 5: Import counter data from a range of timestamps --

```
Import-Counter -Path ".\disk.blg" | Format-Table -Property Timestamp
$start = [datetime]"7/9/2008 3:47:00 PM"; $end = [datetime]"7/9/2008 3:47:59 PM"
Import-Counter -Path Disk.blg -StartTime $start -EndTime $end
```

The first command lists in a table the timestamps of all of the data in the `ProcessorData.blg` file.

The second command saves particular timestamps in the `$start` and `$end` variables. The strings are cast to DateTime objects.

The third command uses the `Import-Counter` cmdlet to get only counter data that has a time stamp between the start and end times (inclusive). The command uses the
StartTime and EndTime parameters of `Import-Counter` to specify the range.

Example 6: Import a specified number of the oldest samples from a performance counter log file

```
Import-Counter -Path "Disk.blg" -MaxSamples 5
(Import-Counter -Path Disk.blg)[-1 .. -5]
```

The first command uses the `Import-Counter` cmdlet to import the first (oldest) five samples from the `Disk.blg` file. The command uses the MaxSamples parameter to
limit the import to five counter samples.

The second command uses array notation and the Windows PowerShell range operator (`..`) to get the last five counter samples from the file. These are the five newest
samples.

----- Example 7: Get a summary of counter data from a file -----

```
Import-Counter "D:\Samples\Memory.blg" -Summary

OldestRecord          NewestRecord          SampleCount
------------          ------------          -----------
7/10/2008 2:59:18 PM    7/10/2008 3:00:27 PM    1000
```

This command uses the Summary parameter of the `Import-Counter` cmdlet to get a summary of the counter data in the `Memory.blg` file.

------- Example 8: Update a performance counter log file -------

```
$counters = Import-Counter OldData.blg -ListSet * | ForEach-Object {$_.PathsWithInstances}
Get-Counter -Counter $Counters -MaxSamples 20 | Export-Counter C:\Logs\NewData.blg
```

The first command uses the ListSet parameter of `Import-Counter` to get the counters in `OldData.blg`, an existing counter log file. The command uses a pipeline
operator (`|`) to send the data to a `ForEach-Object` command that gets only the values of the PathsWithInstances property of each object

The second command gets updated data for the counters in the `$counters` variable. It uses the `Get-Counter` cmdlet to get a current sample, and then export the
results to the `NewData.blg` file.

Example 9: Import performance log data from multiple files and then save it

```
$counters = "D:\test\pdata.blg", "D:\samples\netlog.blg" | Import-Counter
```

This command imports performance log data from two logs and saves the data in the `$counters` variable. The command uses a pipeline operator to send the performance
log paths to Import-Counter, which imports the data from the specified paths.

Notice that each path is enclosed in quotation marks and that the paths are separated from each other by a comma.

RELATED LINKS

Online Version: https://learn.microsoft.com/powershell/module/microsoft.powershell.diagnostics/import-counter?view=powershell-5.1&WT.mc_id=ps-gethelp

Export-Counter

Get-Counter