PowerShell Get-Help on command 'Get-Process'

PS C:\Users\wahid> Get-Help Get-Process

```
NAME
    Get-Process

SYNOPSIS
    Gets the processes that are running on the local computer or a remote computer.


SYNTAX
    Get-Process [[-Name] <System.String[]>] [-ComputerName <System.String[]>]
    [-FileVersionInfo] [-Module] [<CommonParameters>]

    Get-Process [-ComputerName <System.String[]>] [-FileVersionInfo] -Id
    <System.Int32[]> [-Module] [<CommonParameters>]

    Get-Process [-ComputerName <System.String[]>] [-FileVersionInfo] -InputObject
    <System.Diagnostics.Process[]> [-Module] [<CommonParameters>]

    Get-Process -Id <System.Int32[]> -IncludeUserName [<CommonParameters>]

    Get-Process [[-Name] <System.String[]>] -IncludeUserName [<CommonParameters>]
```

Get-Process -IncludeUserName -InputObject <System.Diagnostics.Process[]>
[<CommonParameters>]

DESCRIPTION

The `Get-Process` cmdlet gets the processes on a local or remote computer.

Without parameters, this cmdlet gets all of the processes on the local computer. You can also specify a particular process by process name or process ID (PID) or pass a process object through the pipeline to this cmdlet.

By default, this cmdlet returns a process object that has detailed information about the process and supports methods that let you start and stop the process. You can also use the parameters of the `Get-Process` cmdlet to get file version information for the program that runs in the process and to get the modules that the process loaded.

PARAMETERS

-ComputerName <System.String[]>

Specifies the computers for which this cmdlet gets active processes. The default is the local computer.

Type the NetBIOS name, an IP address, or a fully qualified domain name (FQDN) of one or more computers. To specify the local computer, type the computer name, a dot (`.`), or `localhost`.

This parameter does not rely on Windows PowerShell remoting. You can use the ComputerName parameter of this cmdlet even if your computer is not configured to run remote commands.

-FileVersionInfo <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets the file version information for the program that runs in the process.

On Windows Vista and later versions of Windows, you must open PowerShell with the **Run as administrator** option to use this parameter on processes that you do not own.

You cannot use the FileVersionInfo and ComputerName parameters of the `Get-Process` cmdlet in the same command.

To get file version information for a process on a remote computer, use the `Invoke-Command` cmdlet.

Using this parameter is equivalent to getting the MainModule.FileVersionInfo property of each process object. When you use this parameter, `Get-Process` returns a FileVersionInfo object System.Diagnostics.FileVersionInfo , not a process object. So, you cannot pipe the output of the command to a cmdlet that expects a process object, such as `Stop-Process`.

-Id <System.Int32[]>

Specifies one or more processes by process ID (PID). To specify multiple IDs, use commas to separate the IDs. To find the PID of a process, type `Get-Process`.

-IncludeUserName <System.Management.Automation.SwitchParameter>

Indicates that the UserName value of the Process object is returned with results of the command.

-InputObject <System.Diagnostics.Process[]>

Specifies one or more process objects. Enter a variable that contains the objects, or type a command or expression that gets the objects.

-Module <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet gets the modules that have been loaded by the processes.

On Windows Vista and later versions of Windows, you must open PowerShell with the **Run as administrator** option to use this parameter on processes that you do not own.

To get the modules that have been loaded by a process on a remote computer, use the `Invoke-Command` cmdlet.

This parameter is equivalent to getting the Modules property of each process object. When you use this parameter, this cmdlet returns a ProcessModule object System.Diagnostics.ProcessModule , not a process object. So, you cannot pipe the output of the command to a cmdlet that expects a process object, such as `Stop-Process`.

When you use both the Module and FileVersionInfo parameters in the same command, this cmdlet returns a FileVersionInfo object with information about the file version of all modules.

-Name <System.String[]>

Specifies one or more processes by process name. You can type multiple process names (separated by commas) and use wildcard characters. The parameter name (`Name`) is optional.

<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).

Example 1: Get a list of all active processes on the local computer

```
Get-Process
```

This command gets a list of all active processes running on the local computer. For a definition of each column, see the Notes (#notes)section.

Example 2: Get all available data about one or more processes

```
Get-Process winword, explorer | Format-List *
```

This command gets all available data about the Winword and Explorer processes on the computer. It uses the Name parameter to specify the processes, but it omits the optional parameter name. The pipeline operator (`|`) passes the data to the `Format-List` cmdlet, which displays all available properties (`*`) of the Winword and Explorer process objects.

You can also identify the processes by their process IDs. For instance, `Get-Process -Id 664, 2060`.

Example 3: Get all processes with a working set greater than a specified size

```
Get-Process | Where-Object {$_.WorkingSet -gt 20000000}
```

This command gets all processes that have a working set greater than 20 MB. It uses the `Get-Process` cmdlet to get all running processes. The pipeline operator (`|`) passes the process objects to the `Where-Object` cmdlet, which selects only the object with a value greater than 20,000,000 bytes for the WorkingSet property. WorkingSet is one of many properties of process objects. To see all of the properties, type `Get-Process | Get-Member`. By default, the values of all amount properties are in bytes, even though the default display lists them in kilobytes and megabytes.

Example 4: List processes on the computer in groups based on priority

```
$A = Get-Process
$A | Get-Process | Format-Table -View priority
```

These commands list the processes on the computer in groups based on their priority class. The first command gets all the processes on the computer and then stores them in the `$A` variable.

The second command pipes the Process object stored in the `$A` variable to the `Get-Process` cmdlet, then to the `Format-Table` cmdlet, which formats the processes by using the Priority view.

The Priority view, and other views, are defined in the PS1XML format files in the PowerShell home directory (`$pshome`).

Example 5: Add a property to the standard Get-Process output display

```
Get-Process powershell | Format-Table `
    @{Label = "NPM(K)"; Expression = {[int]($_.NPM / 1024)}},
    @{Label = "PM(K)"; Expression = {[int]($_.PM / 1024)}},
    @{Label = "WS(K)"; Expression = {[int]($_.WS / 1024)}},
    @{Label = "VM(M)"; Expression = {[int]($_.VM / 1MB)}},
    @{Label = "CPU(s)"; Expression = {if ($_.CPU) {$_.CPU.ToString("N")}}},
    Id, ProcessName, StartTime -AutoSize

NPM(K)  PM(K) WS(K)   VM(M)  CPU(s)    Id ProcessName StartTime
------  ----- -----   -----  ------    -- ----------- ---------
   143 239540 259384 2366162 22.73  12720 powershell  12/5/2022 3:21:51 PM
   114  61776 104588 2366127 11.45  18336 powershell  12/5/2022 7:30:53 AM
   156  77924  82060 2366185 10.47  18812 powershell  12/5/2022 7:30:52 AM
    85  48216 115192 2366074 1.14   24428 powershell  12/8/2022 9:14:15 AM
```

This example retrieves processes from the local computer. The retrieved processes are piped to the `Format-Table` command that adds the StartTime property to the standard `Get-Process` output display.

------- Example 6: Get version information for a process -------

```
Get-Process powershell -FileVersionInfo
```

```
ProductVersion   FileVersion      FileName
--------------   -----------      --------
6.1.6713.1       6.1.6713.1 (f...
C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.exe
```

This command uses the FileVersionInfo parameter to get the version information for the `powershell.exe` file that is the main module for the PowerShell process.

To run this command with processes that you do not own on Windows Vista and later versions of Windows, you must open PowerShell with the Run as administrator option.

--- Example 7: Get modules loaded with the specified process ---

```
Get-Process SQL* -Module
```

This command uses the Module parameter to get the modules that have been loaded by the process. This command gets the modules for the processes that have names that begin with `SQL`.

To run this command on Windows Vista and later versions of Windows with processes that you do not own, you must start PowerShell with the Run as administrator option.

------------ Example 8: Find the owner of a process ------------

```
Get-Process pwsh -IncludeUserName

Handles      WS(K)   CPU(s)     Id UserName            ProcessName
-------      -----   ------     -- --------            -----------
    782     132080     2.08   2188 DOMAIN01\user01     powershell

$p = Get-WmiObject Win32_Process -Filter "name='powershell.exe'"
```

```
$p.GetOwner()


__GENUS          : 2
__CLASS          : __PARAMETERS
__SUPERCLASS     :
__DYNASTY        : __PARAMETERS
__RELPATH        :
__PROPERTY_COUNT : 3
__DERIVATION     : {}
__SERVER         :
__NAMESPACE      :
__PATH           :
Domain           : DOMAIN01
ReturnValue      : 0
User             : user01
```

The first command shows how to find the owner of a process. The IncludeUserName parameter requires elevated user rights (Run as Administrator). The output reveals that the owner is `Domain01\user01`.

The second and third command are another way to find the owner of a process.

The second command uses `Get-WmiObject` to get the PowerShell process. It saves it in the `$p` variable.

The third command uses the GetOwner method to get the owner of the process in `$p`. The output reveals that the owner is `Domain01\user01`.

Example 9: Use an automatic variable to identify the process hosting the current session

```
Get-Process powershell


Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
```

```
-------  ------    -----      ----- -----   ------     -- -----------
308      26       52308      61780   567     3.18   5632 powershell
377      26       62676      63384   575     3.88   5888 powershell
```

Get-Process -Id $PID

```
Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
396      26       56488      57236   575     3.90   5888 powershell
```

These commands show how to use the `$PID` automatic variable to identify the process that is hosting the current PowerShell session. You can use this method to distinguish the host process from other PowerShell processes that you might want to stop or close.

The first command gets all of the PowerShell processes in the current session.

The second command gets the PowerShell process that is hosting the current session.

Example 10: Get all processes that have a main window title and display them in a table

Get-Process | Where-Object {$_.mainWindowTitle} | Format-Table Id, Name, mainWindowtitle -AutoSize

This command gets all the processes that have a main window title, and it displays them in a table with the process ID and the process name.

The mainWindowTitle property is just one of many useful properties of the Process object that `Get-Process` returns. To view all of the properties, pipe the results of a `Get-Process` command to the `Get-Member` cmdlet `Get-Process | Get-Member`.

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

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

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

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