Windows PowerShell Get-Help on Cmdlet 'Get-Job'

PS:\>Get-HELP Get-Job -Full

NAME

Get-Job

SYNOPSIS

Gets PowerShell background jobs that are running in the current session.

SYNTAX

Get-Job [[-Id] <System.Int32[]>] [-After <System.DateTime>] [-Before <System.DateTime>] [-ChildJobState {NotStarted | Running | Completed | Failed | Stopped | Blocked | Suspended | Disconnected | Suspending | Stopping | AtBreakpoint}] [-HasMoreData <System.Boolean>] [-IncludeChildJob] [-Newest <System.Int32>] [<CommonParameters>]

Get-Job [-After <System.DateTime>] [-Before <System.DateTime>] [-ChildJobState {NotStarted | Running | Completed | Failed | Stopped | Blocked | Suspended | Disconnected | Suspending | Stopping | AtBreakpoint}] [-Command <System.String[]>] [-HasMoreData <System.Boolean>] [-IncludeChildJob] [-Newest <System.Int32>] [<CommonParameters>]

Get-Job [-InstanceId] <System.Guid[]> [-After <System.DateTime>] [-Before <System.DateTime>] [-ChildJobState

{NotStarted | Running | Completed | Failed | Stopped |

Blocked | Suspended | Disconnected | Suspending | Stopping | AtBreakpoint}] [-HasMoreData <System.Boolean>] [-IncludeChildJob] [-Newest <System.Int32>]

[<CommonParameters>]

Get-Job [-Name] <System.String[]> [-After <System.DateTime>] [-Before <System.DateTime>] [-ChildJobState {NotStarted | Running | Completed | Failed | Stopped |

Blocked | Suspended | Disconnected | Suspending | Stopping | AtBreakpoint}] [-HasMoreData <System.Boolean>] [-IncludeChildJob] [-Newest <System.Int32>]

[<CommonParameters>]

Get-Job [-State] {NotStarted | Running | Completed | Failed | Stopped | Blocked | Suspended | Disconnected | Suspending | Stopping | AtBreakpoint} [-After

<System.DateTime>] [-Before <System.DateTime>] [-ChildJobState {NotStarted | Running | Completed | Failed | Stopped | Blocked | Suspended | Disconnected | Suspending

| Stopping | AtBreakpoint}] [-HasMoreData <System.Boolean>] [-IncludeChildJob] [-Newest <System.Int32>] [<CommonParameters>]

Get-Job [-Filter] <System.Collections.Hashtable> [<CommonParameters>]

DESCRIPTION

The `Get-Job` cmdlet gets objects that represent the background jobs that were started in the current session. You can use `Get-Job` to get jobs that were started by

using the `Start-Job` cmdlet, or by using the AsJob parameter of any cmdlet.

Without parameters, a `Get-Job` command gets all jobs in the current session. You can use the parameters of `Get-Job` to get particular jobs.

The job object that `Get-Job` returns contains useful information about the job, but it does not contain the job results. To get the results, use the `Receive-Job`

cmdlet.

A Windows PowerShell background job is a command that runs in the background without interacting with the current session. Typically, you use a background job to run

a complex command that takes a long time to finish. For more information about background jobs in Windows PowerShell, see about_Jobs (./about/about_Jobs.md).

Beginning in Windows PowerShell 3.0, the `Get-Job` cmdlet also gets custom job types, such as workflow jobs and instances of scheduled jobs. To find the job type of a

job, use the PSJobTypeName property of the job.

To enable `Get-Job` to get a custom job type, import the module that supports the custom job type into the session before you run a `Get-Job` command, either by using

the `Import-Module` cmdlet or by using or getting a cmdlet in the module. For information about a particular custom job type, see the documentation of the custom job

type feature.

PARAMETERS

-After <System.DateTime>

Gets completed jobs that ended after the specified date and time. Enter a DateTime object, such as one returned by the `Get-Date` cmdlet or a string that can be

converted to a DateTime object, such as `Dec 1, 2012 2:00 AM` or `11/06`.

This parameter works only on custom job types, such as workflow jobs and scheduled jobs, that have an EndTime property. It does not work on standard background

jobs, such as those created by using the `Start-Job` cmdlet. For information about support for this parameter, see the help topic for the job type.

This parameter was introduced in Windows PowerShell 3.0.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Before <System.DateTime>

Gets completed jobs that ended before the specified date and time. Enter a DateTime object.

This parameter works only on custom job types, such as workflow jobs and scheduled jobs, that have an EndTime property. It does not work on standard background

jobs, such as those created by using the `Start-Job` cmdlet. For information about support for this parameter, see the help topic for the job type.

This parameter was introduced in Windows PowerShell 3.0.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-ChildJobState <System.Management.Automation.JobState>

Gets only the child jobs that have the specified state. The acceptable values for this parameter are:

- NotStarted

- Running

- Completed

- Failed

- Stopped

- Blocked

- Suspended

- Disconnected

- Suspending

- Stopping

By default, `Get-Job` does not get child jobs. By using the IncludeChildJob parameter, `Get-Job` gets all child jobs. If you use the ChildJobState parameter, the

IncludeChildJob parameter has no effect.

This parameter was introduced in Windows PowerShell 3.0.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Command <System.String[]>

Specifies an array of commands as strings. This cmdlet gets the jobs that include the specified commands. The default is all jobs. You can use wildcard characters

to specify a command pattern.

Required? false

Position? named

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? true

-Filter <System.Collections.Hashtable>

Specifies a hash table of conditions. This cmdlet gets jobs that satisfy all of the conditions. Enter a hash table where the keys are job properties and the

values are job property values.

This parameter works only on custom job types, such as workflow jobs and scheduled jobs. It does not work on standard background jobs, such as those created by

using the `Start-Job` cmdlet. For information about support for this parameter, see the help topic for the job type.

This parameter was introduced in Windows PowerShell 3.0.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-HasMoreData <System.Boolean>

Indicates whether this cmdlet gets only jobs that have the specified HasMoreData property value. The HasMoreData property indicates whether all job results have

been received in the current session. To get jobs that have more results, specify a value of `$True`. To get jobs that do not have more results, specify a value

of `$False`.

To get the results of a job, use the `Receive-Job` cmdlet.

When you use the `Receive-Job` cmdlet, it deletes from its in-memory, session-specific storage the results that it returned. When it has returned all results of

the job in the current session, it sets the value of the HasMoreData property of the job to `$False`) to indicate that it has no more results for the job in the

current session. Use the Keep parameter of `Receive-Job` to prevent `Receive-Job` from deleting results and changing the value of the HasMoreData property. For

more information, type `Get-Help Receive-Job`.

The HasMoreData property is specific to the current session. If results for a custom job type are saved outside of the session, such as the scheduled job type,

which saves job results on disk, you can use the `Receive-Job` cmdlet in a different session to get the job results again, even if the value of HasMoreData is

`$False`. For more information, see the help topics for the custom job type.

This parameter was introduced in Windows PowerShell 3.0.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-Id <System.Int32[]>

Specifies an array of IDs of jobs that this cmdlet gets.

The ID is an integer that uniquely identifies the job in the current session. It is easier to remember and to type than the instance ID, but it is unique only in

the current session. You can type one or more IDs separated by commas. To find the ID of a job, type `Get-Job` without parameters.

Required? false

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-IncludeChildJob <System.Management.Automation.SwitchParameter>

Indicates that this cmdlet returns child jobs, in addition to parent jobs.

This parameter is especially useful for investigating workflow jobs, for which `Get-Job` returns a container parent job, and job failures, because the reason for

the failure is saved in a property of the child job.

This parameter was introduced in Windows PowerShell 3.0.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-InstanceId <System.Guid[]>

Specifies an array of instance IDs of jobs that this cmdlet gets. The default is all jobs.

An instance ID is a GUID that uniquely identifies the job on the computer. To find the instance ID of a job, use `Get-Job`.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Name <System.String[]>

Specifies an array of instance friendly names of jobs that this cmdlet gets. Enter a job name, or use wildcard characters to enter a job name pattern. By default,

`Get-Job` gets all jobs in the current session.

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? true

-Newest <System.Int32>

Specifies a number of jobs to get. This cmdlet gets the jobs that ended most recently.

The Newest parameter does not sort or return the newest jobs in end-time order. To sort the output, use the `Sort-Object` cmdlet.

This parameter was introduced in Windows PowerShell 3.0.

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-State <System.Management.Automation.JobState>

Specifies a job state. This cmdlet gets only jobs in the specified state. The acceptable values for this parameter are:

- NotStarted

- Running

- Completed

- Failed

- Stopped

- Blocked

- Suspended

- Disconnected

- Suspending

- Stopping

By default, `Get-Job` gets all the jobs in the current session.

For more information about job states, see JobState Enumeration (/dotnet/api/system.management.automation.jobstate).

Required? true

Position? 0

Default value None

Accept pipeline input? True (ByPropertyName)

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

None

You can't pipe objects to this cmdlet.

OUTPUTS

System.Management.Automation.RemotingJob

This cmdlet returns objects that represent the jobs in the session.

NOTES

Windows PowerShell includes the following aliases for `Get-Job`:

- `gjb`

The PSJobTypeName property of jobs indicates the job type of the job. The property value is determined by the job type author. The following list shows common job

types.

- BackgroundJob . Local job started by using `Start-Job`. - RemoteJob . Job started in a PSSession by using the AsJob parameter of the `Invoke-Command` cmdlet.

- PSWorkflowJob . Job started by using the AsJob common parameter of workflows.

Example 1: Get all background jobs started in the current session

```
Get-Job

Id     Name            PSJobTypeName   State         HasMoreData     Location             Command
--     ----            -------------   -----         -----------     --------             -------
1      Job1            BackgroundJob   Completed     True            localhost            $env:COMPUTERNAME
```

-------- Example 2: Stop a job by using an instance ID --------

```
$j = Get-Job -Name Job1
$ID = $j.InstanceID
$ID

Guid
----
03c3232e-1d23-453b-a6f4-ed73c9e29d55
```

Stop-Job -InstanceId $ID

----- Example 3: Get jobs that include a specific command -----

Get-Job -Command '*Get-Process*'

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
3	Job3	BackgroundJob	Running	True	localhost	Get-Process

Example 4: Get jobs that include a specific command by using the pipeline

[pscustomobject]@{Command='*Get-Process*'} | Get-Job

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
3	Job3	BackgroundJob	Running	True	localhost	Get-Process

-------- Example 5: Get jobs that have not been started --------

Get-Job -State NotStarted

---- Example 6: Get jobs that have not been assigned a name ----

Get-Job -Name Job*

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
1	Job1	BackgroundJob	Completed	True	localhost	$env:COMPUTERNAME

Example 7: Use a job object to represent the job in a command

```
Start-Job -ScriptBlock {Get-Process} -Name MyJob
$j = Get-Job -Name MyJob
$j

Id     Name            PSJobTypeName   State         HasMoreData     Location             Command
--     ----            -------------   -----         -----------     --------             -------
6      MyJob           BackgroundJob   Completed     True            localhost            Get-Process

Receive-Job -Job $j

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    124       4    13572      12080    59          1140 audiodg
    783      16    11428      13636   100           548 CcmExec
     96       4     4252       3764    59          3856 ccmsetup
...
```

Example 8: Get all jobs including jobs started by a different method

```
Start-Job -ScriptBlock {Get-EventLog -LogName System}
Invoke-Command -ComputerName S1 -ScriptBlock {Get-EventLog -LogName System} -AsJob
Invoke-Command -ComputerName S2 -ScriptBlock {Start-Job -ScriptBlock {Get-EventLog -LogName System}}
Get-Job

Id     Name       PSJobTypeName   State         HasMoreData     Location       Command
--     ----       -------------   -----         -----------     --------       -------
1      Job1       BackgroundJob   Running       True            localhost      Get-EventLog System
2      Job2       RemoteJob       Running       True            S1             Get-EventLog System
```

```
$Session = New-PSSession -ComputerName S2
Invoke-Command -Session $Session -ScriptBlock {Start-Job -ScriptBlock {Get-EventLog -LogName System}}
Invoke-Command -Session $Session -ScriptBlock {Get-Job}

 Id      Name            PSJobTypeName   State         HasMoreData     Location             Command
PSComputerName
--     ----          -------------   -----       -----------    --------          -------              --------------
1      Job1          BackgroundJob   Running       True           localhost          Get-EventLog -LogName Sy. S2



---------- Example 9: Investigate a failed job ----------

PS> Start-Job -ScriptBlock {Get-Process}
Id     Name       PSJobTypeName   State      HasMoreData     Location           Command
--     ----       -------------   -----      -----------     --------           -------
1      Job1       BackgroundJob   Failed     False           localhost          Get-Process

PS> (Get-Job).JobStateInfo | Format-List -Property *
State  : Failed
Reason :

PS> Get-Job | Format-List -Property *
HasMoreData   : False
StatusMessage :
Location      : localhost
Command       : get-process
JobStateInfo  : Failed
Finished      : System.Threading.ManualReset
EventInstanceId    : fb792295-1318-4f5d-8ac8-8a89c5261507
Id            : 1
Name          : Job1
ChildJobs     : {Job2}
```

Output : {}

Error : {}

Progress : {}

Verbose : {}

Debug : {}

Warning : {}

StateChanged :

PS> (Get-Job -Name job2).JobStateInfo.Reason

Connecting to remote server using WSManCreateShellEx api failed. The async callback gave the following error message: Access is denied.

---------- Example 10: Get filtered results ----------

PS> Workflow WFProcess {Get-Process}

PS> WFProcess -AsJob -JobName WFProcessJob -PSPrivateMetadata @{MyCustomId = 92107}

PS> Get-Job -Filter @{MyCustomId = 92107}

Id	Name	State	HasMoreData	Location	Command
1	WFProcessJob	Completed	True	localhost	WFProcess

--------- Example 11: Get information about child jobs ---------

PS> Get-Job

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
2	Job2	BackgroundJob	Completed	True	localhost	.\Get-Archive.ps1
4	Job4	RemoteJob	Failed	True	Server01, Server02	.\Get-Archive.ps1
7	UpdateHelpJob	PSScheduledJob	Completed	True	localhost	Update-Help
8	UpdateHelpJob	PSScheduledJob	Completed	True	localhost	Update-Help

```
9      UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help
10     UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help

PS> Get-Job -IncludeChildJob

Id     Name          PSJobTypeName   State        HasMoreData     Location           Command
--     ----          -------------   -----        -----------     --------           -------
2      Job2          BackgroundJob   Completed    True            localhost          .\Get-Archive.ps1
3      Job3                          Completed    True            localhost          .\Get-Archive.ps1
4      Job4          RemoteJob       Failed       True            Server01, Server02 .\Get-Archive.ps1
5      Job5                          Failed       False           Server01           .\Get-Archive.ps1
6      Job6                          Completed    True            Server02           .\Get-Archive.ps1
7      UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help
8      UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help
9      UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help
10     UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help

PS> Get-Job -Name Job4 -ChildJobState Failed

Id     Name          PSJobTypeName   State        HasMoreData     Location           Command
--     ----          -------------   -----        -----------     --------           -------
2      Job2          BackgroundJob   Completed    True            localhost          .\Get-Archive.ps1
4      Job4          RemoteJob       Failed       True            Server01, Server02 .\Get-Archive.ps1
5      Job5                          Failed       False           Server01           .\Get-Archive.ps1
7      UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help
8      UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help
9      UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help
10     UpdateHelpJob   PSScheduledJob  Completed     True          localhost          Update-Help

PS> (Get-Job -Name Job5).JobStateInfo.Reason

Connecting to remote server Server01 failed with the following error message:

Access is denied.
```

For more information, see the about_Remote_Troubleshooting (./about/about_Remote_Troubleshooting.md)Help topic.

RELATED LINKS

Online Version:

https://learn.microsoft.com/powershell/module/microsoft.powershell.core/get-job?view=powershell-5.1&WT.mc_id=ps-gethelp

Invoke-Command

Receive-Job

Remove-Job

Resume-Job

Start-Job

Stop-Job

Suspend-Job

Wait-Job

about_Jobs

about_Job_Details

about_Remote_Jobs

about_Scheduled_Jobs