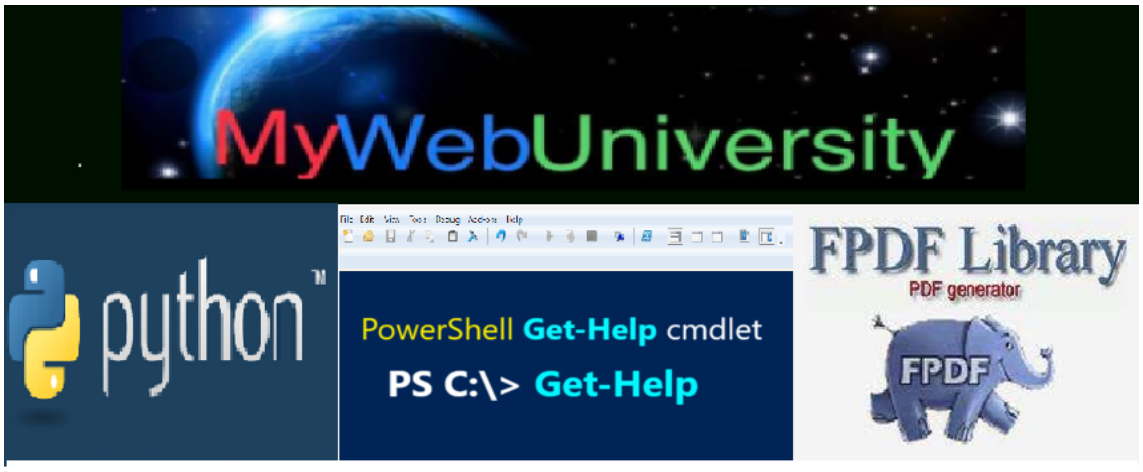

Full credit is given to all the above companies including the Operating System that this PDF file was generated!

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

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

NAME

Stop-Job

SYNOPSIS

Stops a PowerShell background job.

SYNTAX

Stop-Job [-Filter] <System.Collections.Hashtable> [-PassThru] [-Confirm] [-WhatIf] [<CommonParameters>]

Stop-Job [-Id] <System.Int32[]> [-PassThru] [-Confirm] [-WhatIf] [<CommonParameters>]

Stop-Job [-InstanceId] <System.Guid[]> [-PassThru] [-Confirm] [-WhatIf] [<CommonParameters>]

Stop-Job [-Job] <System.Management.Automation.Job[]> [-PassThru] [-Confirm] [-WhatIf] [<CommonParameters>]

Stop-Job [-Name] <System.String[]> [-PassThru] [-Confirm] [-WhatIf] [<CommonParameters>]

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

[-Confirm] [-WhatIf] [<CommonParameters>]

DESCRIPTION

The `Stop-Job` cmdlet stops PowerShell background jobs that are in progress. You can use this cmdlet to stop all jobs or stop selected jobs based on their name, ID,

instance ID, or state, or by passing a job object to `Stop-Job`.

You can use `Stop-Job` to stop background jobs, such as those that were started by using the `Start-Job` cmdlet or the AsJob parameter of any cmdlet. When you stop a

background job, PowerShell completes all tasks that are pending in that job queue and then ends the job. No new tasks are added to the queue after this command is

submitted.

This cmdlet does not delete background jobs. To delete a job, use the `Remove-Job` cmdlet.

Starting in Windows PowerShell 3.0, `Stop-Job` also stops custom job types, such as WorkflowJobs and instances of ScheduledJobs . To enable `Stop-Job` to stop a job

with custom job type, import the module that supports the custom job type into the session before you run a `Stop-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

-Filter <System.Collections.Hashtable>

Specifies a hash table of conditions. This cmdlet stops jobs that satisfy every condition. 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 WorkflowJobs and ScheduledJobs . 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

-Id <System.Int32[]>

Specifies the IDs of jobs that this cmdlet stops. The default is all jobs in the current session.

The ID is an integer that uniquely identifies the job in the current session. It is easier to remember and 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`.

Required? true

Position? 0

Default value All jobs

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-InstanceId <System.Guid[]>

Specifies the instance IDs of jobs that this cmdlet stops. 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 All jobs

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Job <System.Management.Automation.Job[]>

Specifies the jobs that this cmdlet stops. Enter a variable that contains the jobs or a command that gets the jobs. You can also use a pipeline operator to submit

jobs to the `Stop-Job` cmdlet. By default, `Stop-Job` deletes all jobs that were started in the current session.

Required? true

Position? 0

Default value All jobs

Accept pipeline input? True (ByPropertyName, ByValue)

Accept wildcard characters? false

-Name <System.String[]>

Specifies friendly names of jobs that this cmdlet stops. Enter the job names in a comma-separated list or use wildcard characters (`*`) to enter a job name

pattern. By default, `Stop-Job` stops all jobs created in the current session.

Because the friendly name is not guaranteed to be unique, use the WhatIf and Confirm parameters when stopping jobs by name.

Required? true

Position? 0

Default value All jobs

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? true

-PassThru <System.Management.Automation.SwitchParameter>

Returns an object representing the item with which you are working. By default, this cmdlet does not generate any output.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-State <System.Management.Automation.JobState>

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

- `NotStarted`

- `Running`

- `Completed`

- `Failed`

- `Stopped`

- `Blocked`

- `Suspended`

- `Disconnected`

- `Suspending`

- `Stopping`

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

Required? true

Position? 0

Default value All jobs

Accept pipeline input? True (ByPropertyName)

Accept wildcard characters? false

-Confirm <System.Management.Automation.SwitchParameter>

Prompts you for confirmation before running the cmdlet.

Required? false

Position? named

Default value False

Accept pipeline input? False

Accept wildcard characters? false

-WhatIf <System.Management.Automation.SwitchParameter>

Shows what would happen if the cmdlet runs. The cmdlet is not run.

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.Management.Automation.RemotingJob

You can pipe a job object to this cmdlet.

OUTPUTS

None

By default, this cmdlet returns no output.

System.Management.Automation.PSRemotingJob

When you use the PassThru parameter, this cmdlet returns a job object.

NOTES

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

- `spjb`

Example 1: Stop a job on a remote computer with Invoke-Command

```
$s = New-PSSession -ComputerName Server01 -Credential Domain01\Admin02
$j = Invoke-Command -Session $s -ScriptBlock {Start-Job -ScriptBlock {Get-EventLog -LogName System}}
Invoke-Command -Session $s -ScriptBlock { Stop-job -Job $Using:j }
```

This example shows how to use the `Stop-Job` cmdlet to stop a job that is running on a remote computer.

Because the job was started with the `Invoke-Command` cmdlet to run a `Start-Job` command remotely, the job object is stored on the remote computer. You must use

another `Invoke-Command` command to run a `Stop-Job` command remotely. For more information about remote background jobs, see about_Remote_Jobs.

The first command creates a PowerShell session (PSSession) on the Server01 computer, and then stores the session object in the `$s` variable. The command uses the

credentials of a domain administrator.

The second command uses the `Invoke-Command` cmdlet to run a `Start-Job` command in the session. The command in

the job gets all of the events in the System event

log. The resulting job object is stored in the `$j` variable.

The third command stops the job. It uses the `Invoke-Command` cmdlet to run a `Stop-Job` command in the PSSession on Server01. Because the job objects are stored in

`$j`, which is a variable on the local computer, the command uses the Using scope modifier to identify `$j` as a local variable. For more information about the Using

scope modifier, see about_Remote_Variables (about/about_Remote_Variables.md).

When the command finishes, the job is stopped and the PSSession in `$s` is available for use.

---------- Example 2: Stop a background job ----------

```
Stop-Job -Name "Job1"
```

This command stops the `Job1` background job.

---------- Example 3: Stop several background jobs ----------

```
Stop-Job -Id 1, 3, 4
```

This command stops three jobs. It identifies them by their IDs .

---------- Example 4: Stop all background jobs ----------

```
Get-Job | Stop-Job
```

This command stops all of the background jobs in the current session.

--------- Example 5: Stop all blocked background jobs ---------

```
Stop-Job -State Blocked
```

This command stops all the jobs that are blocked.

---------- Example 6: Stop a job by instance ID ----------

```
Get-Job | Format-Table ID, Name, Command, @{Label="State";Expression={$_.JobStateInfo.State}},
```

InstanceID -Auto

Id Name Command State InstanceId

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

1 Job1 start-service schedule Running 05abb67a-2932-4bd5-b331-c0254b8d9146

3 Job3 start-service schedule Running c03cbd45-19f3-4558-ba94-ebe41b68ad03

5 Job5 get-service s* Blocked e3bbfed1-9c53-401a-a2c3-a8db34336adf

Stop-Job -InstanceId e3bbfed1-9c53-401a-a2c3-a8db34336adf

These commands show how to stop a job based on its InstanceID .

The first command uses the `Get-Job` cmdlet to get the jobs in the current session. The command uses a pipeline operator (`|`) to send the jobs to a `Format-Table`

command, which displays a table of the specified properties of each job. The table includes the InstanceID of each job. It uses a calculated property to display the

job state.

The second command uses a `Stop-Job` command that has the InstanceID parameter to stop a selected job.

---------- Example 7: Stop a job on a remote computer ----------

$j = Invoke-Command -ComputerName Server01 -ScriptBlock {Get-EventLog -LogName System} -AsJob

$j | Stop-Job -PassThru

Id Name State HasMoreData Location Command

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

5 Job5 Stopped True user01-tablet Get-EventLog -LogName Sy...

This example shows how to use the `Stop-Job` cmdlet to stop a job that is running on a remote computer.

Because the job was started with the AsJob parameter of the `Invoke-Command` cmdlet, the Job object is located on the local computer, even though the job runs on the

remote computer. Therefore, you can use a local `Stop-Job` command to stop the job.

The first command uses the `Invoke-Command` cmdlet to start a background job on the Server01 computer. The command uses the AsJob parameter to run the remote command

as a background job.

This command returns a job object, which is the same job object that the `Start-Job` cmdlet returns. The command saves the job object in the `$j` variable.

The second command uses a pipeline operator to send the job in the `$j` variable to `Stop-Job`. The command uses the PassThru parameter to direct `Stop-Job` to return

a job object. The job object display confirms that the state of the job is Stopped.

For more information about remote background jobs, see about_Remote_Jobs (About/about_Remote_Jobs.md).

RELATED LINKS

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

Get-Job

Invoke-Command

Receive-Job

Remove-Job

Resume-Job

Start-Job

Suspend-Job

Wait-Job

about_Job_Details

about_Remote_Jobs

about_Remote_Variables

about_Jobs

about_Scopes