



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

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

NAME

Start-Job

SYNOPSIS

Starts a PowerShell background job.

SYNTAX

```

        Start-Job [-ScriptBlock] <System.Management.Automation.ScriptBlock> [[-InitializationScript]
<System.Management.Automation.ScriptBlock>] [-ArgumentList
    <System.Object[]>] [-Authentication {Default | Basic | Negotiate | NegotiateWithImplicitCredential | Credssp | Digest |
Kerberos}] [-Credential
    <System.Management.Automation.PSCredential>] [-InputObject <System.Management.Automation.PSObject>] [-Name
<System.String>] [-PSVersion <System.Version>] [-RunAs32]
    [<CommonParameters>]

```

```

        Start-Job [[-InitializationScript] <System.Management.Automation.ScriptBlock>] [-ArgumentList <System.Object[]>]
[-Authentication {Default | Basic | Negotiate |
        NegotiateWithImplicitCredential | Credssp | Digest | Kerberos}] [-Credential
<System.Management.Automation.PSCredential>] [-InputObject

```

<System.Management.Automation.PSObject>] [-LiteralPath <System.String> [-Name <System.String>] [-PSVersion <System.Version>] [-RunAs32] [<CommonParameters>]

Start-Job [-FilePath] <System.String> [[-InitializationScript] <System.Management.Automation.ScriptBlock>] [-ArgumentList <System.Object[]>] [-Authentication {Default | Basic | Negotiate | NegotiateWithImplicitCredential | Credssp | Digest | Kerberos}] [-Credential <System.Management.Automation.PSCredential>] [-InputObject <System.Management.Automation.PSObject>] [-Name <System.String>] [-PSVersion <System.Version>] [-RunAs32] [<CommonParameters>]

Start-Job [-DefinitionName] <System.String> [[-DefinitionPath] <System.String>] [[-Type] <System.String>] [<CommonParameters>]

DESCRIPTION

The `Start-Job` cmdlet starts a PowerShell background job on the local computer.

A PowerShell background job runs a command without interacting with the current session. When you start a background job, a job object returns immediately, even if

the job takes an extended time to finish. You can continue to work in the session without interruption while the job runs.

The job object contains useful information about the job, but it doesn't contain the job results. When the job finishes, use the `Receive-Job` cmdlet to get the

results of the job. For more information about background jobs, see `about_Jobs` (`./About/about_Jobs.md`).

To run a background job on a remote computer, use the `AsJob` parameter that is available on many cmdlets, or use the `Invoke-Command` cmdlet to run a `Start-Job` command on the remote computer. For more information, see `about_Remote_Jobs` (`./About/about_Remote_Jobs.md`).

Starting in PowerShell 3.0, `Start-Job` can start instances of custom job types, such as scheduled jobs. For information about how to use `Start-Job` to start jobs

with custom types, see the help documents for the job type feature.

The default working directory for jobs is hardcoded. The Windows default is ``\$HOME\Documents`` and on Linux or macOS the default is ``\$HOME``. The script code running in the background job needs to manage the working directory as needed.

PARAMETERS

`-ArgumentList <System.Object[]>`

Specifies an array of arguments, or parameter values, for the script that is specified by the `FilePath` parameter or a command specified with the `ScriptBlock` parameter.

Arguments must be passed to `ArgumentList` as single-dimension array argument. For example, a comma-separated list. For more information about the behavior of

`ArgumentList`, see [about_Splatting \(about/about_Splatting.md#splatting-with-arrays\)](#).

Required? false

Position? named

Default value None

Accept pipeline input? False

Accept wildcard characters? false

`-Authentication <System.Management.Automation.Runspaces.AuthenticationMechanism>`

Specifies the mechanism that is used to authenticate user credentials.

The acceptable values for this parameter are as follows:

- Default

- Basic

- Credssp

- Digest

- Kerberos
- Negotiate
- NegotiateWithImplicitCredential

The default value is Default.

CredSSP authentication is available only in Windows Vista, Windows Server 2008, and later versions of the Windows operating system.

For more information about the values of this parameter, see [AuthenticationMechanism \(/dotnet/api/system.management.automation.runspaces.authenticationmechanism\)](#).

> [!CAUTION] > Credential Security Support Provider (CredSSP) authentication, in which the user's credentials are > passed to a remote computer to be authenticated, is designed for commands that require > authentication on more than one resource, such as accessing a remote network share. This mechanism > increases the security risk of the remote operation. If the remote computer is compromised, the > credentials that are passed to it can be used to control the network session.

Required?	false
Position?	named
Default value	Default
Accept pipeline input?	False
Accept wildcard characters?	false

-Credential <System.Management.Automation.PSCredential>

Specifies a user account that has permission to perform this action. If the Credential parameter isn't specified, the command uses the current user's credentials.

Type a user name, such as User01 or Domain01\User01 , or enter a PSCredential object generated by the ``Get-Credential`` cmdlet. If you type a user name, you're prompted to enter the password.

Credentials are stored in a PSCredential (`/dotnet/api/system.management.automation.pscredential`) object and the password is stored as a SecureString (`/dotnet/api/system.security.securestring`).

> [!NOTE] > For more information about SecureString data protection, see > How secure is SecureString? (`/dotnet/api/system.security.securestring#how-secure-is-securestring`).

Required?	false
Position?	named
Default value	Current user
Accept pipeline input?	False
Accept wildcard characters?	false

`-DefinitionName` <System.String>

Specifies the definition name of the job that this cmdlet starts. Use this parameter to start custom job types that have a definition name, such as scheduled jobs.

When you use ``Start-Job`` to start an instance of a scheduled job, the job starts immediately, regardless of job triggers or job options. The resulting job

instance is a scheduled job, but it isn't saved to disk like triggered scheduled jobs. You can't use the `ArgumentList` parameter of ``Start-Job`` to provide values

for parameters of scripts that run in a scheduled job. For more information, see `about_Scheduled_Jobs` (`../PSScheduledJob/About/about_Scheduled_Jobs.md`).

This parameter was introduced in PowerShell 3.0.

Required?	true
Position?	0

Default value None
Accept pipeline input? False
Accept wildcard characters? false

-DefinitionPath <System.String>

Specifies path of the definition for the job that this cmdlet starts. Enter the definition path. The concatenation of the values of the DefinitionPath and

DefinitionName parameters is the fully qualified path of the job definition. Use this parameter to start custom job types that have a definition path, such as
scheduled jobs.

For scheduled jobs, the value of the DefinitionPath parameter is
`\$HOME\AppData\Local\Windows\PowerShell\ScheduledJob`.

This parameter was introduced in PowerShell 3.0.

Required? false
Position? 1
Default value None
Accept pipeline input? False
Accept wildcard characters? false

-FilePath <System.String>

Specifies a local script that `Start-Job` runs as a background job. Enter the path and file name of the script or use the pipeline to send a script path to

`Start-Job`. The script must be on the local computer or in a folder that the local computer can access.

When you use this parameter, PowerShell converts the contents of the specified script file to a script block and runs the script block as a background job.

Required? true
Position? 0
Default value None

Accept pipeline input? False

Accept wildcard characters? false

-InitializationScript <System.Management.Automation.ScriptBlock>

Specifies commands that run before the job starts. To create a script block, enclose the commands in curly braces (`{ }`).

Use this parameter to prepare the session in which the job runs. For example, you can use it to add functions, snap-ins, and modules to the session.

Required? false

Position? 1

Default value None

Accept pipeline input? False

Accept wildcard characters? false

-InputObject <System.Management.Automation.PSObject>

Specifies input to the command. Enter a variable that contains the objects, or type a command or expression that generates the objects.

In the value of the ScriptBlock parameter, use the ``$input`` automatic variable to represent the input objects.

Required? false

Position? named

Default value None

Accept pipeline input? True (ByValue)

Accept wildcard characters? false

-LiteralPath <System.String>

Specifies a local script that this cmdlet runs as a background job. Enter the path of a script on the local computer.

``Start-Job`` uses the value of the LiteralPath parameter exactly as it's typed. No characters are interpreted as wildcard characters. If the path includes escape

characters, enclose it in single quotation marks. Single quotation marks tell PowerShell not to interpret any characters as escape sequences.

Required?	true
Position?	named
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	false

-Name <System.String>

Specifies a friendly name for the new job. You can use the name to identify the job to other job cmdlets, such as the ``Stop-Job`` cmdlet.

The default friendly name is ``Job#``, where ``#`` is an ordinal number that is incremented for each job.

Required?	false
Position?	named
Default value	None
Accept pipeline input?	True (ByPropertyName)
Accept wildcard characters?	false

-PSVersion <System.Version>

Specifies a version. ``Start-Job`` runs the job with the version of PowerShell. The acceptable values for this parameter are: ``2.0`` and ``3.0``.

This parameter was introduced in PowerShell 3.0.

Required?	false
Position?	named
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	false

`-RunAs32 <System.Management.Automation.SwitchParameter>`

Indicates that ``Start-Job`` runs the job in a 32-bit process. RunAs32 forces the job to run in a 32-bit process, even on a 64-bit operating system.

On 64-bit versions of Windows 7 and Windows Server 2008 R2, when the ``Start-Job`` command includes the RunAs32 parameter, you can't use the Credential parameter to specify the credentials of another user.

Required?	false
Position?	named
Default value	False
Accept pipeline input?	False
Accept wildcard characters?	false

`-ScriptBlock <System.Management.Automation.ScriptBlock>`

Specifies the commands to run in the background job. To create a script block, enclose the commands in curly braces (`{ }`). Use the ``$input`` automatic variable to access the value of the InputObject parameter. This parameter is required.

Required?	true
Position?	0
Default value	None
Accept pipeline input?	False
Accept wildcard characters?	false

`-Type <System.String>`

Specifies the custom type for jobs started by ``Start-Job``. Enter a custom job type name, such as PSScheduledJob for scheduled jobs or PSWorkflowJob for workflows jobs. This parameter isn't valid for standard background jobs.

This parameter was introduced in PowerShell 3.0.

Required?	false
-----------	-------

Position? 2
Default value None
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 \(https://go.microsoft.com/fwlink/?LinkID=113216\)](https://go.microsoft.com/fwlink/?LinkID=113216).

INPUTS

System.String

You can pipe an object with the Name property to the Name parameter to this cmdlet. For example, you can pipe a FileInfo object from ``Get-ChildItem``.

OUTPUTS

System.Management.Automation.PSRemotingJob

This cmdlet returns a PSRemotingJob object representing the job that it started.

NOTES

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

- ``sajb``

To run in the background, ``Start-Job`` runs in its own session in the current session. When you use the ``Invoke-Command`` cmdlet to run a ``Start-Job`` command in a session on a remote computer, ``Start-Job`` runs in a session in the remote session.

----- Example 1: Start a background job -----

```
Start-Job -ScriptBlock { Get-Process -Name powershell }
```

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

`Start-Job` uses the ScriptBlock parameter to run `Get-Process` as a background job. The Name parameter specifies to find PowerShell processes, `powershell`. The job information is displayed and PowerShell returns to a prompt while the job runs in the background.

To view the job's output, use the `Receive-Job` cmdlet. For example, `Receive-Job -Id 1`.

----- Example 2: Start a job using Invoke-Command -----

```
$jobWRM = Invoke-Command -ComputerName (Get-Content -Path C:\Servers.txt) -ScriptBlock {  
    Get-Service -Name WinRM } -JobName WinRM -ThrottleLimit 16 -AsJob
```

A job that uses `Invoke-Command` is created and stored in the `\$jobWRM` variable. `Invoke-Command` uses the ComputerName parameter to specify the computers where the job runs. `Get-Content` gets the server names from the `C:\Servers.txt` file.

The ScriptBlock parameter specifies a command that `Get-Service` gets the WinRM service. The JobName parameter specifies a friendly name for the job, WinRM . The

ThrottleLimit parameter limits the number of concurrent commands to 16. The AsJob parameter starts a background job that runs the command on the servers.

----- Example 3: Get job information -----

```
$j = Start-Job -ScriptBlock { Get-WinEvent -Log System } -Credential Domain01\User01  
$j | Select-Object -Property *
```

State : Completed

HasMoreData : True

StatusMessage :

Location : localhost

Command : Get-WinEvent -Log System

JobStateInfo : Completed

Finished : System.Threading.ManualResetEvent

InstanceId : 27ce3fd9-40ed-488a-99e5-679cd91b9dd3

Id : 18

Name : Job18

ChildJobs : {Job19}

PSBeginTime : 8/8/2019 14:41:57

PSEndTime : 8/8/2019 14:42:07

PSJobTypeName : BackgroundJob

Output : {}

Error : {}

Progress : {}

Verbose : {}

Debug : {}

Warning : {}

Information : {}

`Start-Job` uses the ScriptBlock parameter to run a command that specifies `Get-WinEvent` to get the System log. The Credential parameter specifies a domain user

account with permission to run the job on the computer. The job object is stored in the `\$j` variable.

The object in the `\$j` variable is sent down the pipeline to `Select-Object`. The Property parameter specifies an asterisk (*) to display all the job object's properties.

----- Example 4: Run a script as a background job -----

```
Start-Job -FilePath C:\Scripts\Sample.ps1
```

`Start-Job` uses the FilePath parameter to specify a script file that's stored on the local computer.

----- Example 5: Get a process using a background job -----

```
Start-Job -Name PShellJob -ScriptBlock { Get-Process -Name PowerShell }
```

`Start-Job` uses the Name parameter to specify a friendly job name, PShellJob . The ScriptBlock parameter specifies `Get-Process` to get processes with the name

PowerShell .

-- Example 6: Collect and save data by using a background job --

```
Start-Job -Name GetMappingFiles -InitializationScript {Import-Module -Name MapFunctions} -ScriptBlock {  
    Get-Map -Name * | Set-Content -Path D:\Maps.tif } -RunAs32
```

`Start-Job` uses the Name parameter to specify a friendly job name, GetMappingFiles . The InitializationScript parameter runs a script block that imports the

MapFunctions module. The ScriptBlock parameter runs `Get-Map` and `Set-Content` saves the data in the location specified by the Path parameter. The RunAs32 parameter

runs the process as 32-bit, even on a 64-bit operating system.

----- Example 7: Pass input to a background job -----

```
Start-Job -ScriptBlock { Get-Content -Path $input } -InputObject "C:\Servers.txt"
```

```
Receive-Job -Name Job45 -Keep
```

Server01

Server02

Server03

Server04

`Start-Job` uses the ScriptBlock parameter to run `Get-Content` with the `\$input` automatic variable. The `\$input` variable gets objects from the InputObject

parameter. `Receive-Job` uses the Name parameter to specify the job and outputs the results. The Keep parameter saves the job output so it can be viewed again during

the PowerShell session.

Example 8: Use the ArgumentList parameter to specify an array

```
Start-Job -ScriptBlock { Get-Process -Name $args } -ArgumentList powershell, pwsh, notepad
```

Id	Name	PSJobTypeName	State	HasMoreData	Location	Command
--	----	-----	-----	-----	-----	-----
1	Job1	BackgroundJob	Running	True	localhost	Get-Process -Name \$args

The ``Start-Job`` cmdlet uses the `ScriptBlock` parameter to run a command. ``Get-Process`` uses the `Name` parameter to specify the automatic variable ``$args``. The

`ArgumentList` parameter passes the array of process names to ``$args``. The process names `powershell`, `pwsh`, and `notepad` are processes running on the local computer.

To view the job's output, use the ``Receive-Job`` cmdlet. For example, ``Receive-Job -Id 1``.

RELATED LINKS

Online

Version:

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

[about_Arrays](#)

[about_Automatic_Variables](#)

[about_Jobs](#)

[about_Job_Details](#)

[about_Remote_Jobs](#)

[Get-Job](#)

[Invoke-Command](#)

[Receive-Job](#)

[Remove-Job](#)

[Resume-Job](#)

[Stop-Job](#)

[Suspend-Job](#)

[Wait-Job](#)