PowerShell Get-Help on command 'Set-JobTrigger'

PS C:\Users\wahid> Get-Help Set-JobTrigger

NAME

Set-JobTrigger

SYNOPSIS

Changes the job trigger of a scheduled job.

SYNTAX

Set-JobTrigger [-InputObject]

<Microsoft.PowerShell.ScheduledJob.ScheduledJobTrigger[]> [-At

<System.DateTime>] [-AtLogOn] [-AtStartup] [-Daily] [-DaysInterval

<System.Int32>] [-DaysOfWeek {Sunday | Monday | Tuesday | Wednesday | Thursday

| Friday | Saturday}] [-Once] [-PassThru] [-RandomDelay <System.TimeSpan>]

[-RepeatIndefinitely] [-RepetitionDuration <System.TimeSpan>]

[-RepetitionInterval <System.TimeSpan>] [-User <System.String>] [-Weekly]

[-WeeksInterval <System.Int32>] [<CommonParameters>]

DESCRIPTION

The `Set-JobTrigger` cmdlet changes the properties of the job triggers of

scheduled jobs. You can use it to change the time or frequency at which the jobs start or to change from a time-based schedules to schedules that are triggered by a logon or startup.

A job trigger defines a recurring schedule or conditions for starting a scheduled job. Although job triggers are not saved to disk, you can change the job triggers of scheduled jobs, which are saved to disk.

To change a job trigger of a scheduled job, begin by using the `Get-JobTrigger` cmdlet to get the job trigger of a scheduled job. Then, pipe the trigger to `Set-JobTrigger` or save the trigger in a variable and use the InputObject parameter of `Set-JobTrigger` cmdlet to identify the trigger. Use the remaining parameters of `Set-JobTrigger` to change the job trigger.

When you change the type of a job trigger, such as changing a job trigger from a daily or weekly trigger to an AtLogon trigger, the original trigger properties are deleted. However, if you change the values of the trigger, but not its type, such as changing the days in a weekly trigger, only the properties that you specify are changed. All other properties of the original job trigger are retained.

`Set-JobTrigger` is one of a collection of job scheduling cmdlets in the PSScheduledJob module that is included in Windows PowerShell.

For more information about Scheduled Jobs, see the About topics in the PSScheduledJob module. Import the PSScheduledJob module and then type: `Get-Help about_Scheduled*` or see about_Scheduled_Jobs (About/about_Scheduled_Jobs.md).

This cmdlet was introduced in Windows PowerShell 3.0.

PARAMETERS

-At <System.DateTime>

Starts the job at the specified date and time. Enter a DateTime object, such as one that the `Get-Date` cmdlet returns, or a string that can be converted to a time, such as `April 19, 2012 15:00`, `12/31/2013 9:00 PM`, or `3am`.

If you don't specify an element of the DateTime object, such as seconds, that element of the job trigger is not changed. If the original job trigger didn't include a DateTime object and you omit an element, the job trigger is created with the corresponding element from the current date and time.

When using the Once parameter, set the value of the At parameter to a particular date and time. Because the default date in a DateTime object is the current date, setting a time before the current time without an explicit date results in a job trigger for a time in the past. DateTime objects, and strings that are converted to DateTime objects, are automatically adjusted to be compatible with the date and time formats selected for the local computer in Region and Language in Control Panel.

-AtLogOn <System.Management.Automation.SwitchParameter>

Starts the scheduled job when the specified users log on to the computer. To specify a user, use the User parameter.

-AtStartup <System.Management.Automation.SwitchParameter>

Starts the scheduled job when Windows starts.

-Daily <System.Management.Automation.SwitchParameter>

Specifies a recurring daily job schedule. Use the other parameters in the Daily parameter set to specify the schedule details.

-DaysInterval <System.Int32>

Specifies the number of days between occurrences on a daily schedule. For example, a value of `3` starts the scheduled job on days `1`, `4`, `7` and so on. The default value is `1`.

-DaysOfWeek <System.DayOfWeek[]>

Specifies the days of the week on which a weekly scheduled job runs. Enter day names, such as `Monday`, `Thursday`, integers `0`-`6`, where `0` represents Sunday, or an asterisk (`*`) to represent every day. This parameter is required in the Weekly parameter set.

Day names are converted to their integer values in the job trigger. When you enclose day names in quotation marks in a command, enclose each day name in separate quotation marks, such as `"Monday", "Tuesday"`. If you enclose multiple day names in a single quotation mark pair, the corresponding integer values are summed. For example, `"Monday, Tuesday"` (`1 + 2`) results in a value of `Wednesday` (`3`).

-InputObject <Microsoft.PowerShell.ScheduledJob.ScheduledJobTrigger[]>

Specifies the job triggers. Enter a variable that contains ScheduledJobTrigger objects or type a command or expression that gets ScheduledJobTrigger objects, such as a `Get-JobTrigger` command. You can also pipe a ScheduledJobTrigger object to `Set-JobTrigger`.

If you specify multiple job triggers, `Set-JobTrigger` makes the same changes to all job triggers.

-Once <System.Management.Automation.SwitchParameter>

Specifies a non-recurring (one time) schedule.

-PassThru <System.Management.Automation.SwitchParameter>

Returns the job triggers that changed. By default, this cmdlet does not generate any output.

-RandomDelay <System.TimeSpan>

Enables a random delay that begins at the scheduled start time, and sets

the maximum delay value. The length of the delay is set pseudo-randomly
for each start and varies from no delay to the time specified by the value
of this parameter. The default value, zero (`00:00:00`), disables the
random delay.

Enter a timespan object, such as one returned by the `New-TimeSpan`
cmdlet, or enter a value in `<hours>:<minutes>:<seconds>` format, which is
automatically converted to a timespan object.

-RepeatIndefinitely <System.Management.Automation.SwitchParameter>

This parameter, available starting in Windows PowerShell 4.0, eliminates
the necessity of specifying a TimeSpan.MaxValue value for the
RepetitionDuration parameter to run a scheduled job repeatedly, for an
indefinite period.

-RepetitionDuration <System.TimeSpan>

Repeats the job until the specified time expires. The repetition frequency
is determined by the value of the RepetitionInterval parameter. For
example, if the value of RepetitionInterval is 5 minutes and the value of
RepetitionDuration is 2 hours, the job is triggered every five minutes for
two hours.

Enter a timespan object, such as one that the `New-TimeSpan` cmdlet
returns or a string that can be converted to a timespan object, such as
`1:05:30`.

To run a job indefinitely, add the RepeatIndefinitely parameter instead.

To stop a job before the job trigger repetition duration expires, set the
RepetitionDuration value to zero (`0`).

To change the repetition duration or repetition interval of a Once job
trigger, the command must include both the RepetitionInterval and

RepetitionDuration parameters. To change the repetition duration or
repetition intervals of other types of job triggers, the command must
include the Once , At , RepetitionInterval and RepetitionDuration
parameters.

-RepetitionInterval <System.TimeSpan>

Repeats the job at the specified time interval. For example, if the value
of this parameter is 2 hours, the job is triggered every two hours. The
default value, `0`, does not repeat the job.

Enter a timespan object, such as one that the `New-TimeSpan` cmdlet
returns or a string that can be converted to a timespan object, such as
`1:05:30`.

To change the repetition duration or repetition interval of a Once job
trigger, the command must include both the RepetitionInterval and
RepetitionDuration parameters. To change the repetition duration or
repetition intervals of other types of job triggers, the command must
include the Once , At , RepetitionInterval and RepetitionDuration
parameters.

-User <System.String>

Specifies the users who trigger an AtLogon start of a scheduled job. Enter
the name of a user in `<UserName>` or `<Domain><Username>` format or enter
an asterisk (`*`) to represent all users. The default value is all users.

-Weekly <System.Management.Automation.SwitchParameter>

Specifies a recurring weekly job schedule. Use the other parameters in the
Weekly parameter set to specify the schedule details.

-WeeksInterval <System.Int32>

Specifies the number of weeks between occurrences on a weekly job
schedule. For example, a value of `3` starts the scheduled job on weeks

`1`, `4`, `7` and so on. The default value is `1`.

<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: Change the days in a job trigger ---------

```
Get-JobTrigger -Name "DeployPackage"

Id        Frequency      Time                  DaysOfWeek
Enabled
--        ---------      ----                  ----------
-------
1         Weekly         9/29/2011 12:00:00 AM  {Wednesday, Saturday}   True

Get-JobTrigger -Name "DeployPackage" | Set-JobTrigger -DaysOfWeek "Wednesday",
"Sunday" -Passthru

Id        Frequency      Time                  DaysOfWeek
Enabled
--        ---------      ----                  ----------
-------
1         Weekly         9/29/2011 12:00:00 AM  {Wednesday, Sunday}     True
```

The first command uses the `Get-JobTrigger` cmdlet to get the job trigger of the `DeployPackage` scheduled job. The output shows that the trigger starts the job at midnight on Wednesdays and Saturdays.

The second command uses the `Get-JobTrigger` cmdlet to get the job trigger of the `DeployPackage` scheduled job. A pipeline operator (`|`) sends the trigger to the `Set-JobTrigger` cmdlet, which changes the job trigger so that it

starts the `DeployPackage` job on Wednesdays and Sundays. The command uses the

Passthru parameter to return the trigger after the change.

This command is not required; it is included only to show the effect of the

trigger change.

------------ Example 2: Change the job trigger type ------------

```
Get-JobTrigger -Name "Inventory"

Id         Frequency       Time                    DaysOfWeek
Enabled
--         ---------       ----                    ----------
-------
1          Daily           9/27/2011 11:00:00 PM                          True
2          AtStartup                                              True

Get-JobTrigger -Name "Inventory" -TriggerID 2 | Set-JobTrigger -Weekly
-WeeksInterval 4 -DaysOfWeek Monday -At "12:00 AM"

Id         Frequency       Time                    DaysOfWeek
Enabled
--         ---------       ----                    ----------
-------
1          Daily           9/27/2011 11:00:00 PM                          True
2          Weekly          10/31/2011 12:00:00 AM {Monday}                True
```

The first command uses the `Get-JobTrigger` cmdlet to get the job trigger of

the `Inventory` scheduled job. The output shows that the job has two triggers

a daily trigger and an AtStartup trigger.

The second command uses the `Get-JobTrigger` cmdlet to get the AtStartup job

trigger of the `Inventory` job. The command uses the TriggerID parameter to

identify the job trigger. A pipeline operator (`|`) sends the job trigger to
the `Set-JobTrigger` cmdlet, which changes it to a weekly job trigger that
runs every four weeks on Monday at midnight. The command uses the Passthru
parameter to return the trigger after the change.

This command is not required; it is included only to show the effect of the
trigger change.

------ Example 3: Change the user on a remote job trigger ------

```
Invoke-Command -ComputerName "Server01" -ScriptBlock {Get-ScheduledJob |
Get-JobTrigger | Where-Object {$_.User} | Set-JobTrigger -User
"Domain01/Admin02"}
```

This command changes the user in all AtLogon job triggers of scheduled jobs on
the Server01 computer.

The command uses the `Invoke-Command` cmdlet to run a command on the Server01
computer.

The remote command begins with a `Get-ScheduledJob` command that gets all
scheduled jobs on the computer. The scheduled jobs are piped to the
`Get-JobTrigger` cmdlet, which gets the job triggers of the scheduled jobs.
Each job trigger contains a JobDefinition property that contains the scheduled
job, so the trigger remains associated with the scheduled job even when it is
changed.

The job triggers are piped to the `Where-Object` cmdlet, which gets job
triggers that have the User property. The selected job triggers are piped to
the `Set-JobTrigger` cmdlet, which changes the user to `Domain01\Admin02`.

---------- Example 4: Change one of many job triggers ----------

```
Get-JobTrigger -Name "SecurityCheck"
```

```
Id        Frequency      Time                   DaysOfWeek
Enabled
--        ---------      ----                   ----------
-------
1         Daily          4/24/2013 3:00:00 AM                         True
2         Weekly         4/24/2013 4:00:00 PM   {Sunday}              True
3         Once           4/24/2013 4:00:00 PM                         True

Get-JobTrigger -Name "SecurityCheck" -TriggerID 3 | Format-List -Property *


At                 : 4/24/2012 4:00:00 PM
DaysOfWeek         :
Interval           : 1
Frequency          : Once
RandomDelay        : 00:00:00
RepetitionInterval : 01:00:00
RepetitionDuration : 1.00:00:00
User               :
Id                 : 3
Enabled            : True
JobDefinition      : Microsoft.PowerShell.ScheduledJob.ScheduledJobDefinition

Get-JobTrigger -Name "SecurityCheck" -TriggerId 3 | Set-JobTrigger
-RepetitionInterval (New-TimeSpan -Minutes 90)
Get-JobTrigger -Name "SecurityCheck" -TriggerID 3 | Format-List -Property *


At                 : 4/24/2012 4:00:00 PM
DaysOfWeek         :
Interval           : 1
Frequency          : Once
RandomDelay        : 00:00:00
RepetitionInterval : 01:30:00
RepetitionDuration : 1.00:00:00
```

```
    User              :
    Id                : 3
    Enabled           : True
    JobDefinition     : Microsoft.PowerShell.ScheduledJob.ScheduledJobDefinition
```

The commands in this example changes the repetition interval of the Once job trigger of `SecurityCheck` scheduled job from every 60 minutes to every 90 minutes. The `SecurityCheck` scheduled job has three job triggers, so the commands use the TriggerId parameter of the `Get-JobTrigger` cmdlet to identify the job trigger that is being changed.

The first command uses the `Get-JobTrigger` cmdlet to get all job triggers of the `SecurityCheck` scheduled job. The output, which displays the IDs of the job triggers, reveals that the Once job trigger has an ID of `3`.

The second command uses the TriggerID parameter of the `Get-JobTrigger` cmdlet to get the Once trigger of the `SecurityCheck` scheduled job. The command pipes the trigger to the `Format-List` cmdlet, which displays all of the properties of the Once job trigger. The output shows that the trigger starts the job once every hour (RepetitionInterval is 1 hour) for one day (RepetitionDuration is 1 day).

The third command changes the repetition interval of the job trigger from one hour to 90 minutes. The command does not return any output.

The fourth command displays the effect of the change.The output shows that the trigger starts the job once every 90 minutes (RepetitionInterval is 1 hour, 30 minutes) for one day (RepetitionDuration is 1 day).

REMARKS

To see the examples, type: "get-help Set-JobTrigger -examples".
For more information, type: "get-help Set-JobTrigger -detailed".
For technical information, type: "get-help Set-JobTrigger -full".
For online help, type: "get-help Set-JobTrigger -online"