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

PowerShell Get-Help on command 'Enable-JobTrigger'

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

NAME

Enable-JobTrigger

SYNOPSIS

Enables the job triggers of scheduled jobs.

SYNTAX

Enable-JobTrigger [-InputObject]

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

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

DESCRIPTION

The `Enable-JobTrigger` cmdlet re-enables job triggers of scheduled jobs, such as those that were disabled by using the `Disable-JobTrigger` cmdlet. Enabled and re-enabled job triggers can start scheduled jobs immediately; there is no need to restart Windows or Windows PowerShell.

To use this cmdlet, use the `Get-JobTrigger` cmdlet to get the job triggers. Then pipe the job triggers to `Enable-JobTrigger` or use its InputObject

parameter.

To enable a job trigger, the `Enable-JobTrigger` cmdlet sets the Enabled property of the job trigger to `$true`.

`Enable-ScheduledJob` 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

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

Specifies the job trigger to enable. 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 `Enable-JobTrigger`.

-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.

-Confirm <System.Management.Automation.SwitchParameter>

Prompts you for confirmation before running the cmdlet.

-WhatIf <System.Management.Automation.SwitchParameter>

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

<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: Enable a job trigger ----------

```
Get-JobTrigger -Name Backup-Archives -TriggerID 1 | Enable-JobTrigger
```

This command enables the first trigger (ID=1) of the Backup-Archives scheduled
job on the local computer.

The command uses the `Get-JobTrigger` cmdlet to get the job trigger. A
pipeline operator sends the job trigger to the `Enable-JobTrigger` cmdlet,
which enables it.

---------- Example 2: Enable all job triggers ----------

```
Get-ScheduledJob | Get-JobTrigger | Enable-JobTrigger
```

The command uses the `Get-ScheduledJob` cmdlet to get the scheduled jobs on
the local computer. A pipeline operator (`|`) sends the scheduled jobs to the
`Get-JobTrigger` cmdlet, which gets all job triggers of the scheduled jobs.
Another pipeline operator sends the job triggers to the `Enable-JobTrigger`
cmdlet, which enables them.

Example 3: Enable the job trigger of a scheduled job on a remote computer

```
Invoke-Command -ComputerName Server01 {Get-JobTrigger -Name DeployPackage |
Where-Object {$_.Frequency -eq "AtLogon"} | Enable-JobTrigger}
```

This command re-enables the AtLogon job triggers on the DeployPackage
scheduled job on the Server01 remote computer.

The command uses the `Invoke-Command` cmdlet to run the commands on the Server01 computer. The remote command uses the `Get-JobTrigger` cmdlet to get the job triggers of the DeployPackage scheduled job. A pipeline operator sends the job triggers to the `Where-Object` cmdlet which returns only AtLogon job triggers. A pipeline operator sends the AtLogon job triggers to the `Enable-JobTrigger` cmdlet, which enables them.

---------- Example 4: Display disabled job triggers ----------

```
Get-ScheduledJob | Get-JobTrigger | where {!$_.Enabled} | Format-Table Id,
Frequency, At, DaysOfWeek, Enabled,
@{Label="JobName";Expression={$_.JobDefinition.Name}}
Id Frequency At                  DaysOfWeek Enabled JobName
-- --------- --                  ---------- ------- -------
 1    Weekly 9/28/2011 3:00:00 AM   {Monday}   False   Backup-Archive
 2    Daily  9/29/2011 1:00:00 AM              False   Backup-Archive
 1    Weekly 10/20/2011 11:00:00 PM {Friday}   False   Inventory
 1    Weekly 11/2/2011 2:00:00 PM   {Monday}   False   Inventory
```

This command displays all disabled job triggers of all scheduled jobs in a table. You can use a command like this one to discover job triggers that might need to be enabled.

The command uses the `Get-ScheduledJob` cmdlet to get the scheduled jobs on the local computer. A pipeline operator (`|`) sends the scheduled jobs to the `Get-JobTrigger` cmdlet, which gets all job triggers of the scheduled jobs. Another pipeline operator sends the job triggers to the `Where-Object` cmdlet, which returns only job triggers that are disabled, that is, where the value of the Enabled property of the job trigger is not (`!`) true.

Another pipeline operator sends the disabled job triggers to the `Format-Table` cmdlet, which displays the selected properties of the job triggers in a table. The properties include a new JobName property that displays the name of the scheduled job in the JobDefinition property of the

job trigger.

REMARKS

To see the examples, type: "get-help Enable-JobTrigger -examples".

For more information, type: "get-help Enable-JobTrigger -detailed".

For technical information, type: "get-help Enable-JobTrigger -full".

For online help, type: "get-help Enable-JobTrigger -online"