PowerShell Get-Help on command 'Register-EngineEvent'

PS C:\Users\wahid> Get-Help Register-EngineEvent

```
NAME
    Register-EngineEvent


SYNOPSIS
    Subscribes to events that are generated by the PowerShell engine and by the
    `New-Event` cmdlet.


SYNTAX
    Register-EngineEvent [-SourceIdentifier] <System.String> [[-Action]
    <System.Management.Automation.ScriptBlock>] [-Forward] [-MaxTriggerCount
    <System.Int32>] [-MessageData <System.Management.Automation.PSObject>]
    [-SupportEvent] [<CommonParameters>]


DESCRIPTION
    The `Register-EngineEvent` cmdlet subscribes to events that are generated by
    the PowerShell engine and the `New-Event` cmdlet. Use the SourceIdentifier
    parameter to specify the event.

    You can use this cmdlet to subscribe to the OnIdle or Exiting engine events
```

and events generated by the `New-Event` cmdlet. These events are automatically added to the event queue in your session without subscribing. However, subscribing lets you forward the events, specify an action to respond to the events, and cancel the subscription.

When you subscribe to an event, an event subscriber is added to your session. To get the event subscribers in the session, use the `Get-EventSubscriber` cmdlet. To cancel the subscription, use the `Unregister-Event` cmdlet, which deletes the event subscriber from the session.

When the subscribed event is raised, it is added to the event queue in your session. To get events in the event queue, use the `Get-Event` cmdlet.

PARAMETERS

-Action <System.Management.Automation.ScriptBlock>

Specifies commands to handle the events. The commands in the Action run when an event is raised, instead of sending the event to the event queue. Enclose the commands in braces (`{}`) to create a script block.

The value of the Action parameter can include the `$Event`, `$EventSubscriber`, `$Sender`, `$EventArgs`, and `$Args` automatic variables, which provide information about the event to the Action script block. For more information, see about_Automatic_Variables (../Microsoft.PowerShell.Core/About/about_Automatic_Variables.md).

When you specify an action, `Register-EngineEvent` returns an event job object that represents that action. You can use the Job cmdlets to manage the event job.

-Forward <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet sends events for this subscription to the session on the local computer. Use this parameter when you are registering for events on a remote computer or in a remote session.

-MaxTriggerCount <System.Int32>

Specifies the maximum number of times that the action is executed for the event subscription.

-MessageData <System.Management.Automation.PSObject>

Specifies additional data associated with the event. The value of this parameter appears in the MessageData property of the event object.

-SourceIdentifier <System.String>

Specifies the source identifier of the event to which you are subscribing. The source identifier must be unique in the current session. This parameter is required.

The value of this parameter appears in the value of the SourceIdentifier property of the subscriber object and of all event objects associated with this subscription.

The value is specific to the source of the event. This can be an arbitrary value you created to use with the `New-Event` cmdlet. The PowerShell engine supports the PSEngineEvent values PowerShell.Exiting and PowerShell.OnIdle .

-SupportEvent <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet hides the event subscription. Add this parameter when the current subscription is part of a more complex event registration mechanism and it should not be discovered independently.

To view or cancel a subscription that was created with the SupportEvent parameter, add the Force parameter to the `Get-EventSubscriber` or `Unregister-Event` cmdlets.

<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: Register a PowerShell engine event on remote computers

```
$S = New-PSSession -ComputerName "Server01, Server02"
Invoke-Command -Session $S {
  Register-EngineEvent -SourceIdentifier
([System.Management.Automation.PsEngineEvent]::Exiting) -Forward
}
```

`New-PSSession` creates a user-managed session (PSSession) on each of the remote computers.The `Invoke-Command` cmdlet runs the `Register-EngineEvent` command in the remote sessions. `Register-EngineEvent` uses the SourceIdentifier parameter to identify the event. The Forward parameter tell the engine to forward the events from the remote session to the local session.

Example 2: Take a specified action when the Exiting event occurs

```
Register-EngineEvent -SourceIdentifier PowerShell.Exiting -SupportEvent
-Action {
    Get-History | Export-Clixml $HOME\history.clixml
}
```

The SupportEvent parameter is added to hide the event subscription. When PowerShell exits, in this case, the command history from the exiting session is exported an XML file in the user's `$HOME` directory.

--- Example 3: Create and subscribe to a user-defined event ---

```
Register-EngineEvent -SourceIdentifier MyEventSource -Action {
    "Event: {0}" -f $event.messagedata | Out-File c:\temp\MyEvents.txt -Append
}

Start-Job -Name TestJob -ScriptBlock {
   While ($True) {
      Register-EngineEvent -SourceIdentifier MyEventSource -Forward
      Start-Sleep -seconds 2
      "Doing some work..."
      New-Event -SourceIdentifier MyEventSource -Message ("{0} -  Work
done..." -f (Get-Date))
   }
}
Start-Sleep -seconds 4
Get-EventSubscriber
Get-Job

SubscriptionId   : 12
SourceObject     :
EventName        :
SourceIdentifier : MyEventSource
Action           : System.Management.Automation.PSEventJob
HandlerDelegate  :
SupportEvent     : False
ForwardEvent     : False

Id     Name            PSJobTypeName   State         HasMoreData     Location
        Command
--     ----            -------------   -----         -----------     --------
        -------
18     MyEventSource                   Running       True
        .
19     TestJob         BackgroundJob   Running       True            localhost
        .
```

`Register-EngineEvent` created Job Id 18. `Start-Job` created Job Id 19. In Example #4, we remove the event subscription and the jobs, then inspect the log file.

-------- Example 4: Unregister events and clean up jobs --------

```
PS> Start-Sleep -seconds 10
PS> Get-EventSubscriber | Unregister-Event
PS> Get-Job

Id     Name            PSJobTypeName   State         HasMoreData     Location
       Command
--     ----            -------------   -----         -----------     --------
       -------
18     MyEventSource                   Stopped       False
       .
19     TestJob         BackgroundJob   Running       True            localhost
       .

PS> Stop-Job -Id 19
PS> Get-Job | Remove-Job
PS> Get-Content C:\temp\MyEvents.txt
Event: 2/18/2020 2:36:19 PM -  Work done...
Event: 2/18/2020 2:36:21 PM -  Work done...
Event: 2/18/2020 2:36:23 PM -  Work done...
Event: 2/18/2020 2:36:25 PM -  Work done...
Event: 2/18/2020 2:36:27 PM -  Work done...
Event: 2/18/2020 2:36:29 PM -  Work done...
Event: 2/18/2020 2:36:31 PM -  Work done...
```

The `Unregister-Event` cmdlet stops the job associated with the event subscription (Job Id 18). Job Id 19 is still running and creating new events. We use the Job cmdlets stop the job and remove the unneeded job objects. `Get-Content` displays the contents of the log file.

REMARKS

To see the examples, type: "get-help Register-EngineEvent -examples".

For more information, type: "get-help Register-EngineEvent -detailed".

For technical information, type: "get-help Register-EngineEvent -full".

For online help, type: "get-help Register-EngineEvent -online"