PowerShell Get-Help on command 'Register-ObjectEvent'

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

NAME

Register-ObjectEvent

SYNOPSIS

Subscribes to the events that are generated by a Microsoft .NET Framework object.

SYNTAX

```
Register-ObjectEvent [-InputObject] <System.Management.Automation.PSObject>
[-EventName] <System.String> [[-SourceIdentifier] <System.String>] [[-Action]
<System.Management.Automation.ScriptBlock>] [-Forward] [-MaxTriggerCount
<System.Int32>] [-MessageData <System.Management.Automation.PSObject>]
[-SupportEvent] [<CommonParameters>]
```

DESCRIPTION

The `Register-ObjectEvent` cmdlet subscribes to events that are generated by .NET objects on the local computer or on a remote computer.

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.

You can use the parameters of `Register-ObjectEvent` to specify property values of the events that can help you to identify the event in the queue. You can also use the Action parameter to specify actions to take when a subscribed event is raised and the Forward parameter to send remote events to the event queue in the local session.

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.

PARAMETERS

-Action <System.Management.Automation.ScriptBlock>

Specifies the commands to handle the event. 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. These variables 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-ObjectEvent` returns an event job object that represents that action. You can use the Job cmdlets to manage the event job.

-EventName <System.String>

Specifies the event to which you are subscribing.

The value of this parameter must be the name of the event that the .NET
object exposes. For example, the ManagementEventWatcher class has events
named EventArrived and Stopped . To find the event name of an event, use
the `Get-Member` cmdlet.

-Forward <System.Management.Automation.SwitchParameter>

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

-InputObject <System.Management.Automation.PSObject>

Specifies the .NET object that generates the events. Enter a variable that
contains the object, or type a command or expression that gets the object.

-MaxTriggerCount <System.Int32>

Specifies the maximum number of times an event can be triggered.

-MessageData <System.Management.Automation.PSObject>

Specifies any additional data to be associated with this event
subscription. The value of this parameter appears in the MessageData
property of all events associated with this subscription.

-SourceIdentifier <System.String>

Specifies a name that you select for the subscription. The name that you
select must be unique in the current session. The default value is the
GUID that PowerShell assigns.

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

-SupportEvent <System.Management.Automation.SwitchParameter>

Indicates that the cmdlet hides the event subscription. Use this parameter

    when the current subscription is part of a more complex event registration
    mechanism and should not be discovered independently.

    To view or cancel a subscription that was created with the SupportEvent
    parameter, use the Force parameter of the `Get-EventSubscriber` and
    `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: Subscribe to events when a new process starts ---

```
$queryParameters = '__InstanceCreationEvent', (New-Object TimeSpan 0,0,1),
    "TargetInstance isa 'Win32_Process'"
$Query = New-Object System.Management.WqlEventQuery -ArgumentList
$queryParameters
$ProcessWatcher = New-Object System.Management.ManagementEventWatcher $Query
Register-ObjectEvent -InputObject $ProcessWatcher -EventName "EventArrived"
```

----- Example 2: Specify an action to respond to an event -----

```
$queryParameters = '__InstanceCreationEvent', (New-Object TimeSpan 0,0,1),
    "TargetInstance isa 'Win32_Process'"
$Query = New-Object System.Management.WqlEventQuery -ArgumentList
$queryParameters
$ProcessWatcher = New-Object System.Management.ManagementEventWatcher $query
$newEventArgs = @{
    SourceIdentifier = 'PowerShell.ProcessCreated'
    Sender = $Sender
    EventArguments = $EventArgs.NewEvent.TargetInstance
}
$Action = { New-Event @newEventArgs }
Register-ObjectEvent -InputObject $ProcessWatcher -EventName "EventArrived"
-Action $Action

Id   Name              PSJobTypeName   State      HasMoreData   Location
Command
--   ----              -------------   -----      -----------   --------
-------
 5   3db2d67a-efff-...                 NotStarted   False
New-Event @newEventArgs
```

The action uses the `$Sender` and `$EventArgs` automatic variables which are populated only for event actions.

The `Register-ObjectEvent` command returns a job object that represents the action, which runs as a background job. You can use the Job cmdlets, such as `Get-Job` and `Receive-Job`, to manage the background job. For more information, see about_Jobs (../Microsoft.PowerShell.Core/About/about_Jobs.md).

-- Example 3: Subscribe to object events on remote computers --

```
# ProcessCreationEvent.ps1
function  Enable-ProcessCreationEvent {
    $queryParameters = "__InstanceCreationEvent", (New-Object TimeSpan 0,0,1),
        "TargetInstance isa 'Win32_Process'"
    $Query = New-Object System.Management.WqlEventQuery -ArgumentList
$queryParameters

    $objectEventArgs = @{
        Input = New-Object System.Management.ManagementEventWatcher $Query
        EventName = 'EventArrived'
        SourceIdentifier = 'WMI.ProcessCreated'
        MessageData = 'Test'
        Forward = $True
    }
    Register-ObjectEvent @objectEventArgs
}

$S = New-PSSession -ComputerName "Server01, Server02"
Invoke-Command -Session $S -FilePath ProcessCreationEvent.ps1
Invoke-Command -Session $S { Enable-ProcessCreationEvent }
```

The first we create PSSessions on two remote computers and save them in the `$S` variable. Next, the `Invoke-Command` cmdlet run the `ProcessCreationEvent.ps1` script in the each of the PSSessions in `$S`. This action creates the `Enable-ProcessCreationEvent` function in the remote sessions. Finally, we run the `Enable-ProcessCreationEvent` function in the remote sessions.

The function includes a `Register-ObjectEvent` command that subscribes to instance creation events on the Win32_Process object through the ManagementEventWatcher object and its EventArrived event.

-- Example 4: Use the dynamic module in the PSEventJob object --

```
$Timer = New-Object Timers.Timer
$Timer.Interval = 500
$Timer.Enabled = $True
$objectEventArgs = @{
    InputObject = $Timer
    EventName = 'Elapsed'
    SourceIdentifier = 'Timer.Random'
    Action = {$Random = Get-Random -Min 0 -Max 100}
}
$Job = Register-ObjectEvent @objectEventArgs
$Job | Format-List -Property *
```

```
& $Job.module {$Random}
& $Job.module {$Random}

State         : Running
Module        : __DynamicModule_53113769-31f2-42dc-830b-8749325e28d6
StatusMessage :
HasMoreData   : True
Location      :
Command       : $Random = Get-Random -Min 0 -Max 100
JobStateInfo  : Running
Finished      : System.Threading.ManualResetEvent
InstanceId    : 47b5ec9f-bfe3-4605-860a-4674e5d44ca8
Id            : 7
Name          : Timer.Random
ChildJobs     : {}
PSBeginTime   : 6/27/2019 10:19:06 AM
PSEndTime     :
PSJobTypeName :
Output        : {}
Error         : {}
Progress      : {}
Verbose       : {}
Debug         : {}
Warning       : {}
Information   : {}
60
47
```

The PSEventJob has a Module property that contains a dynamic script module that implements the action. Using the call operator (`&`), we invoke the command in the module to display the value of the `$Random` variable.

For more information about modules, see about_Modules

(../Microsoft.PowerShell.Core/About/about_Modules.md).

REMARKS

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

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

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

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