Register-WmiEvent (Microsoft.PowerShell.Management) - PowerShell

By sdwheeler

Archived: 2026-04-05 13:11:01 UTC

In this article

Subscribes to a Windows Management Instrumentation (WMI) event.

Syntax

class (Default)

```
Register-WmiEvent
   [-Class] <String>
   [[-SourceIdentifier] <String>]
   [[-Action] <ScriptBlock>]
   [-Namespace <String>]
   [-Credential <PSCredential>]
   [-ComputerName <String>]
   [-Timeout <Int64>]
   [-MessageData <PSObject>]
   [-SupportEvent]
   [-Forward]
   [-MaxTriggerCount <Int32>]
   [<CommonParameters>]
```

query

```
Register-WmiEvent
   [-Query] <String>
   [[-SourceIdentifier] <String>]
```

```
    [[-Action] <ScriptBlock>]
    [-Namespace <String>]
    [-Credential <PSCredential>]
    [-ComputerName <String>]
    [-Timeout <Int64>]
    [-MessageData <PSObject>]
    [-SupportEvent]
    [-Forward]
    [-MaxTriggerCount <Int32>]
    [<CommonParameters>]
```

Description

The `Register-WmiEvent` cmdlet subscribes to Windows Management Instrumentation (WMI) events on the local computer or on a remote computer.

When the subscribed WMI event is raised, it is added to the event queue in your local session even if the event occurs on a remote computer. To get events in the event queue, use the `Get-Event` cmdlet.

You can use the parameters of `Register-WmiEvent` to subscribe to events on remote computers and to specify the property values of the events that can help you identify the event in the queue. You can also use the **Action** parameter to specify actions to take when a subscribed event is raised.

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.

New Common Information Model (CIM) cmdlets, introduced Windows PowerShell 3.0, perform the same tasks as the WMI cmdlets. The CIM cmdlets comply with WS-Management (WSMan) standards and with the CIM standard, which enables the cmdlets to use the same techniques to manage computers that run the Windows operating system and those that run other operating systems. Instead of using `Register-WmiEvent`, consider using the Register-CimIndicationEvent cmdlet.

Examples

Example 1: Subscribe to events generated by a class

This command subscribes to the events generated by the **Win32_ProcessStartTrace** class. This class raises an event whenever a process starts.

```
Register-WmiEvent -Class 'Win32_ProcessStartTrace' -SourceIdentifier "ProcessStarted"
```

Example 2: Subscribe to creation events for a process

This command uses a query to subscribe to Win32_process instance creation events.

```
 $wmiParameters = @{
  Query = "select * from __instancecreationevent within 5 where targetinstance isa 'Win32_Process'"
  SourceIdentifier = "WMIProcess"
  MessageData = "Test 01"
  TimeOut = 500
}
Register-WmiEvent @wmiParameters
```

Example 3: Use an action to respond to an event

This example shows how to use an action to respond to an event. In this case, when a process starts, any `Start-Process` commands in the current session are written to an XML file.

```
 $action = {
    Get-History |
    Where-Object { $_.CommandLine -like "*Start-Process*" } |
    Export-CliXml "commandHistory.clixml"
}
Register-WmiEvent -Class Win32_ProcessStartTrace -SourceIdentifier ProcessStarted -Action $action
```

```
 Id    Name            State      HasMoreData   Location  Command
--    ----            -----      -----------   --------  -------
1     ProcessStarted  NotStarted False                   Get-History | where {...
```

When you use the **Action** parameter, `Register-WmiEvent` returns a background job that represents the event action. You can use the **Job** cmdlets, such as `Get-Job` and `Receive-Job`, to manage the event job.

For more information, see about_Jobs.

Example 4: Register for events on a remote computer

This example registers for events on the Server01 remote computer.

```
 Register-WmiEvent -Class 'Win32_ProcessStartTrace' -SourceIdentifier "Start" -ComputerName Server01
Get-Event -SourceIdentifier "Start"
```

WMI returns the events to the local computer and stores them in the event queue in the current session. To retrieve the events, run a local `Get-Event` command.

Parameters

-Action

Specifies commands that handle the events. The commands in the **Action** parameter 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 **Action** 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.

When you specify an action, `Register-WmiEvent` returns an event job object that represents that action. You can use the cmdlets that contain the **Job** noun (the **Job** cmdlets) to manage the event job.

Parameter properties

Type:	ScriptBlock
Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ (All)

Position:	101
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-Class

Specifies the event to which you are subscribing. Enter the WMI class that generates the events. A **Class** or **Query** parameter is required in every command.

Parameter properties

Type:	String
Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ class

Position:	0
Mandatory:	True
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-ComputerName

Specifies the name of the computer on which the command runs. The default is the local computer.

Type the NetBIOS name, an IP address, or a fully qualified domain name of the computer. To specify the local computer, type the computer name, a dot (`.`), or localhost.

This parameter does not rely on Windows PowerShell remoting. You can use the **ComputerName** parameter even if your computer is not configured to run remote commands.

Parameter properties

Type:	String
Default value:	None
Supports wildcards:	False
DontShow:	False
Aliases:	Cn

Parameter sets

▼ (All)

Position:	Named
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-Credential

Specifies a user account that has permission to perform this action. The default is the current user.

Type a user name, such as User01 or Domain01\User01, or enter a **PSCredential** object, such as one generated by the `Get-Credential` cmdlet. If you type a user name, this cmdlet prompts you for a password.

Parameter properties

Type:	PSCredential
Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ (All)

Position:	Named
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-Forward

Indicates that this 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.

Parameter properties

Type:	SwitchParameter
Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ (All)

Position:	Named
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-MaxTriggerCount

Specifies the maximum trigger count.

Parameter properties

Type:	Int32
Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ (All)

Position:	Named
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-MessageData

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.

Parameter properties

Type:	PSObject

Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ (All)

Position:	Named
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-Namespace

Specifies the namespace of the WMI class.

Parameter properties

Type:	String
Default value:	None
Supports wildcards:	False
DontShow:	False
Aliases:	NS

Parameter sets

▼ (All)

Position:	Named
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-Query

Specifies a query in WMI Query Language (WQL) that identifies the WMI event class, such as: `select * from __InstanceDeletionEvent`.

Parameter properties

Type:	String
Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ query

Position:	0
Mandatory:	True
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-SourceIdentifier

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 Windows PowerShell assigns.

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.

Parameter properties

Type:	String
Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ (All)

Position:	100
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-SupportEvent

Indicates that this cmdlet hides the event subscription. Use 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 by using the **SupportEvent** parameter, specify the **Force** parameter of the `Get-EventSubscriber` and `Unregister-Event` cmdlets.

Parameter properties

Type:	SwitchParameter
Default value:	None
Supports wildcards:	False
DontShow:	False

Parameter sets

▼ (All)

Position:	Named
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

-Timeout

Specifies how long Windows PowerShell waits for this command to finish.

The default value, 0 (zero), means that there is no time-out, and it causes Windows PowerShell to wait indefinitely.

Parameter properties

Type:	Int64
Default value:	None
Supports wildcards:	False
DontShow:	False
Aliases:	TimeoutMSec

Parameter sets

▼ (All)

Position:	Named
Mandatory:	False
Value from pipeline:	False
Value from pipeline by property name:	False
Value from remaining arguments:	False

CommonParameters

This cmdlet supports the common parameters: -Debug, -ErrorAction, -ErrorVariable, -InformationAction, -InformationVariable, -OutBuffer, -OutVariable, -PipelineVariable, -ProgressAction, -Verbose, -WarningAction, and -WarningVariable. For more information, see about_CommonParameters.

Inputs

None

You cannot pipe objects to this cmdlet.

Outputs

None

This cmdlet does not generate any output.

Notes

To use this cmdlet in Windows Vista or a later version of the Windows operating system, start Windows PowerShell by using the Run as administrator option.

Events, event subscriptions, and the event queue exist only in the current session. If you close the current session, the event queue is discarded and the event subscription is canceled.
