#### Get-WinEvent (Microsoft.PowerShell.Diagnostics) - PowerShell | Microsoft Learn - 31/10/2024 19:26

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.diagnostics/get-winevent?view=powershell-7.4&viewFallbackFrom=powershell-7.3

We use optional cookies to improve your experience on our websites, such as through social media connections, and to display personalized advertising based on your online activity. If you reject optional cookies, only cookies necessary to provide you the services will be used. You may change your selection by

clicking "Manage Cookies" at the bottom of the page. Privacy Statement ThirdAccept

Reject

Manage cookies

## Microsoft Ignite

Party Cookies

Nov 19-22, 2024

Register now >



### **Get-WinEvent**

Reference Feedback

Module: Microsoft.PowerShell.Diagnostics

#### In this article

**Syntax** 

Description

Examples

**Parameters** 

Show 4 more

Gets events from event logs and event tracing log files on local and remote computers.

### **Syntax**

```
Get-WinEvent
  [[-LogName] <String[]>]
  [-MaxEvents <Int64>]
  [-ComputerName <String>]
  [-Credential <PSCredential>]
  [-FilterXPath <String>]
  [-Force]
  [-Oldest]
  [<CommonParameters>]
```

```
Get-WinEvent
  [-ListLog] <String[]>
  [-ComputerName <String>]
  [-Credential <PSCredential>]
  [-Force]
  [<CommonParameters>]
```

```
Get-WinEvent
  [-ListProvider] <String[]>
  [-ComputerName <String>]
  [-Credential <PSCredential>]
  [<CommonParameters>]
```

```
Get-WinEvent
  [-ProviderName] <String[]>
  [-MaxEvents <Int64>]
  [-ComputerName <String>]
  [-Credential <PSCredential>]
```

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.diagnostics/get-winevent?view=powershell-7.4&viewFallbackFrom=powershell-7.3

```
[-FilterXPath <String>]
[-Force]
[-Oldest]
[<CommonParameters>]
```

```
Get-WinEvent
  [-Path] <String[]>
  [-MaxEvents <Int64>]
  [-Credential <PSCredential>]
  [-FilterXPath <String>]
  [-Oldest]
  [<CommonParameters>]
```

```
Get-WinEvent
  [-MaxEvents <Int64>]
  [-ComputerName <String>]
  [-Credential <PSCredential>]
  [-FilterHashtable] <Hashtable[]>
  [-Force]
  [-Oldest]
  [<CommonParameters>]
```

```
Get-WinEvent
  [-MaxEvents <Int64>]
  [-ComputerName <String>]
  [-Credential <PSCredential>]
  [-FilterXml] <XmlDocument>
  [-Oldest]
  [<CommonParameters>]
```

### Description

This cmdlet is only available on the Windows platform.

The Get-WinEvent cmdlet gets events from event logs, including classic logs, such as the **System** and **Application** logs. The cmdlet gets data from event logs that are generated by the Windows Event Log technology introduced in Windows Vista and events in log files generated by **Event Tracing for Windows (ETW)**. By default, Get-WinEvent returns event information in the order of newest to oldest.

Get-WinEvent lists event logs and event log providers. To interrupt the command, press CTRL+C. You can get events from selected logs or from logs generated by selected event providers. And, you can combine events from multiple sources in a single command. Get-WinEvent allows you to filter events using XPath queries, structured XML queries, and hash table queries.

If you're not running PowerShell as an Administrator, you might see error messages that you cannot retrieve information about a log.

### **Examples**

## Example 1: Get all the logs from a local computer

This command gets all the event logs on the local computer. Logs are listed in the order that Get-WinEvent gets them. Classic logs are retrieved first, followed by the new Windows Event logs. It's possible for a log's **RecordCount** to be null, which is blank, or zero.

Get-WinEv	ent -ListLog *		
LogMode	MaximumSizeInBytes	RecordCount	LogName
Circular	15532032	14500	Application
Circular	1052672	117	Azure Information
Circular	1052672	3015	CxAudioSvcLog

Circular	20971520	ForwardedEvents
Circular	20971520	0 HardwareEvents

The Get-WinEvent cmdlet gets log information from the computer. The ListLog parameter uses the asterisk (\*) wildcard to display information about each log.

### **Example 2: Get the classic Setup log**

This command gets an **EventLogConfiguration** object that represents the classic **Setup** log. The object includes information about the log, such as file size, provider, file path, and whether the log is enabled.

```
Get-WinEvent -ListLog Setup | Format-List -Property *
FileSize
                            : 69632
                            : False
IsLogFull
LastAccessTime
                           : 3/13/2019 09:41:46
LastWriteTime
                           : 3/13/2019 09:41:46
OldestRecordNumber
                            : 1
RecordCount
                           : 23
LogName
                           : Setup
LogType
                           : Operational
LogIsolation
                           : Application
IsEnabled
                           : True
                           : False
IsClassicLog
SecurityDescriptor
                           : 0:BAG:SYD: ...
LogFilePath
                           : %SystemRoot%\System32\Wine\
MaximumSizeInBytes
                           : 1052672
LogMode
                           : Circular
OwningProviderName
                            : Microsoft-Windows-Eventlog
                           : {Microsoft-Windows-WUSA, Mi
ProviderNames
ProviderLevel
ProviderKeywords
                     : 64
ProviderBufferSize
ProviderMinimumNumberOfBuffers: 0
ProviderMaximumNumberOfBuffers: 64
                    : 1000
ProviderLatency
ProviderControlGuid
```

The Get-WinEvent cmdlet uses the **ListLog** parameter to specify the **Setup** log. The object is sent down the pipeline to the Format-List cmdlet. Format-List uses the **Property** parameter with the asterisk (\*) wildcard to display each property.

# Example 3: Configure the classic Security log

This command gets an **EventLogConfiguration** object that represents the classic **Security** log. The object is then used to configure settings for the log, such as max file size, file path, and whether the log is enabled.

```
$log = Get-WinEvent -ListLog Security
$log.MaximumSizeInBytes = 1gb
try{
  $log.SaveChanges()
  Get-WinEvent -ListLog Security | Format-List -Property *
}catch [System.UnauthorizedAccessException]{
   $ErrMsg = 'You do not have permission to configure this
   $ErrMsg += ' Try running this script with administrator p
   $ErrMsg += $_.Exception.Message
  Write-Error $ErrMsg
}
FileSize
                              : 69632
IsLogFull
                              : False
LastAccessTime
                              : 3/13/2019 09:41:46
                             : 3/13/2019 09:41:46
LastWriteTime
OldestRecordNumber
RecordCount
                              : 23
                              : Security
LogName
LogType
                              : Administrative
LogIsolation
                              : Custom
                              : True
IsEnabled
IsClassicLog
                             : True
SecurityDescriptor
                              : 0:BAG:SYD: ...
LogFilePath
                              : %SystemRoot%\System32\Wine\
                              : 1073741824
MaximumSizeInBytes
LogMode
                              : Circular
OwningProviderName
ProviderNames
                               : {Microsoft-Windows-WUSA, Mi
```

```
ProviderLevel :
ProviderKeywords :
ProviderBufferSize : 64
ProviderMinimumNumberOfBuffers : 0
ProviderMaximumNumberOfBuffers : 64
ProviderLatency : 1000
ProviderControlGuid :
```

The Get-WinEvent cmdlet uses the ListLog parameter to specify the Security log. The object is saved to a variable. The MaximumSizeInBytes property is set to 1 gigabyte on the object. The SaveChanges method is called to push the change to the system inside of a try block to handle access violations. The Get-WinEvent cmdlet is called again on the Security log and piped to the Format-List cmdlet to verify that the MaximumSizeInBytes property has been saved on the machine.

### Example 4: Get event logs from a server

This command only gets event logs on the local computer that contain events. It's possible for a log's **RecordCount** to be null or zero. The example uses the \$\_ variable. For more information, see about Automatic Variables.

The Get-WinEvent cmdlet gets log information from the computer. The ListLog parameter uses the asterisk (\*) wildcard to display information about each log. The ComputerName parameter specifies to get the

logs from the local computer, **localhost**. The objects are sent down the pipeline to the Where-Object cmdlet. Where-Object uses \$\_.RecordCount to return only logs that contain data. \$\_ is a variable that represents the current object in the pipeline. **RecordCount** is a property of the object with a non-null value.

## Example 5: Get event logs from multiple servers

This example gets objects that represent the **Application** event logs on three computers: Server01, Server02, and Server03. The **ForEach** keyword is used because the **ComputerName** parameter accepts only one value. For more information, see about\_Foreach.

```
$S = 'Server01', 'Server02', 'Server03'
ForEach ($Server in $S) {
 Get-WinEvent -ListLog Application -ComputerName $Server |
   Select-Object LogMode, MaximumSizeInBytes, RecordCount,
     @{name='ComputerName'; expression={$Server}} |
   Format-Table -AutoSize
}
LogMode MaximumSizeInBytes RecordCount LogName
                                            ComputerN
-----
                15532032
Circular
                              14577 Application Server01
Circular
                15532032
                             9689 Application Server02
                               5309 Application Server03
Circular
                15532032
```

The variable \$\$ stores the names three servers: Server01, Server02, and Server03. The ForEach statement uses a loop to process each server, (\$Server in \$\$). The script block in the curly braces ({ }) runs the Get-WinEvent command. The ListLog parameter specifies the Application log. The ComputerName parameter uses the variable \$Server to get log information from each server.

The objects are sent down the pipeline to the <code>Select-Object</code> cmdlet. <code>Select-Object</code> gets the properties <code>LogMode</code>, <code>MaximumSizeInBytes</code>, <code>RecordCount</code>, <code>LogName</code>, and uses a calculated expression to display the <code>ComputerName</code> using the <code>\$Server</code> variable. The objects are sent down the pipeline to the <code>Format-Table</code> cmdlet to display the output in the PowerShell console. The <code>AutoSize</code> parameter formats the output to fit the screen.

## Example 6: Get event log providers and log names

This command gets the event log providers and the logs to which they write.

```
Get-WinEvent -ListProvider *

Name : .NET Runtime
LogLinks : {Application}
Opcodes : {}
Tasks : {}

Name : .NET Runtime Optimization Service
LogLinks : {Application}
Opcodes : {}
Tasks : {}
```

The Get-WinEvent cmdlet gets log information from the computer. The ListProvider parameter uses the asterisk (\*) wildcard to display information about each provider. In the output, the Name is the provider and LogLinks is the log that the provider writes to.

# Example 7: Get all event log providers that write to a specific log

This command gets all of the providers that write to the **Application** log.

```
(Get-WinEvent -ListLog Application).ProviderNames

.NET Runtime
.NET Runtime Optimization Service
Application
Application Error
Application Hang
Application Management
```

The Get-WinEvent cmdlet gets log information from the computer. The ListLog parameter uses Application to get objects for that log.

ProviderNames is a property of the object and displays the providers that write to the Application log.

# Example 8: Get event log provider names that contain a specific string

This command gets the event log providers with names that include a specific string in the provider's name.

```
Get-WinEvent -ListProvider *Policy*

Name : Group Policy Applications
LogLinks : {Application}
Opcodes : {}
Tasks : {}

Name : Group Policy Client
LogLinks : {Application}
Opcodes : {}
Tasks : {}

Name : Group Policy Data Sources
LogLinks : {Application}
```

```
Opcodes : {}
Tasks : {}
```

The Get-WinEvent cmdlet gets log information from the computer. The ListProvider parameter uses the asterisk (\*) wildcard to find Policy anywhere within the provider's name.

## Example 9: Get Event Ids that the event provider generates

This command lists the Event Ids that the **Microsoft-Windows- GroupPolicy** event provider generates along with the event description.

```
(Get-WinEvent -ListProvider Microsoft-Windows-GroupPolicy).E

Id Description
-------
1500 The Group Policy settings for the computer were proces
1501 The Group Policy settings for the user were processed
4115 Group Policy Service started.
4116 Started the Group Policy service initialization phase.
4117 Group Policy Session started.
```

The Get-WinEvent cmdlet gets log information from the computer. The ListProvider parameter specifies the provider, Microsoft-Windows-GroupPolicy. The expression is wrapped in parentheses and uses the Events property to get objects. The objects are sent down the pipeline to the Format-Table cmdlet. Format-Table displays the Id and Description of the event objects.

# Example 10: Get log information from event object properties

This example shows how to get information about a log's contents using event object properties. Event objects are stored in a variable and

then grouped and counted by **Event Id** and **Level**.

```
$Event = Get-WinEvent -LogName 'Windows PowerShell'
$Event.Count
$Event | Group-Object -Property Id -NoElement | Sort-Object
$Event | Group-Object -Property LevelDisplayName -NoElement

195

Count Name
-----
147 600
22 400
21 601
3 403
2 103

Count Name
-----
2 Warning
193 Information
```

The Get-WinEvent cmdlet uses the LogName parameter to specify the Windows PowerShell event log. The event objects are stored in the \$Event variable. The Count property of \$Event shows the total number of logged events.

The \$Event variable is sent down the pipeline to the Group-Object cmdlet. Group-Object uses the Property parameter to specify the Id property and counts the objects by the event Id value. The NoElement parameter removes other properties from the objects output. The grouped objects are sent down the pipeline to the Sort-Object cmdlet. Sort-Object uses the Property parameter to sort the objects by Count. The Descending parameter displays the output by count, from highest to lowest. In the output, the Count column contains the total number of each event. The Name column contains the grouped event Id numbers.

The \$Event variable is sent down the pipeline to the Group-Object cmdlet. Group-Object uses the Property parameter to specify the LevelDisplayName property and counts the objects by LevelDisplayName. The objects are grouped by the levels such as Warning and Information. The NoElement parameter removes other properties from the output. In the output, the Count column contains the total number of each event. The Name column contains the grouped LevelDisplayName.

## Example 11: Get error events that have a specified string in their name

This example uses a comma-separated string of log names. The output is grouped by the level such as error or warning and the log name.

```
Get-WinEvent -LogName *PowerShell*, Microsoft-Windows-Kernel
Group-Object -Property LevelDisplayName, LogName -NoElemer
Format-Table -AutoSize

Count Name
----

1 Error, PowerShellCore/Operational
26 Information, Microsoft-Windows-Kernel-WHEA/Operationa
488 Information, Microsoft-Windows-PowerShell/Operational
77 Information, PowerShellCore/Operational
9835 Information, Windows PowerShell
19 Verbose, PowerShellCore/Operational
444 Warning, Microsoft-Windows-PowerShell/Operational
512 Warning, PowerShellCore/Operational
```

The Get-WinEvent cmdlet gets log information from the computer. The LogName parameter uses a comma-separated string with the asterisk (\*) wildcard to specify the log names. The objects are sent down the pipeline to the Group-Object cmdlet. Group-Object uses the Property parameter to group the objects by LevelDisplayName and LogName. The NoElement parameter removes other properties from the output.

The grouped objects are sent down the pipeline to the Format-Table cmdlet. Format-Table uses the AutoSize parameter to format the columns. The Count column contains the total number of each event. The Name column contains the grouped LevelDisplayName and LogName.

# Example 12: Get events from an archived event log

Get-WinEvent can get event information from saved log files. This sample uses an archived PowerShell log that is stored on the local computer.

```
Get-WinEvent -Path 'C:\Test\Windows PowerShell.evtx'
ProviderName: PowerShell
TimeCreated
                      Id LevelDisplayName Message
-----
3/15/2019 13:54:13 403 Information Engine state i
3/15/2019 13:54:13
                    400 Information
                                        Engine state i
                     600 Information
                                        Provider "Vari
3/15/2019 13:54:13
3/15/2019 13:54:13
                     600 Information
                                         Provider "Func
                      600 Information
                                          Provider "File
3/15/2019 13:54:13
```

The Get-WinEvent cmdlet gets log information from the computer. The Path parameter specifies the directory and file name.

# Example 13: Get a specific number of events from an archived event log

These commands get a specific number of events from an archived event log. Get-WinEvent has parameters that can get a maximum number of events or the oldest events. This sample uses an archived

PowerShell log that is stored in C:\Test\PowerShellCore Operational.evtx.

Get-WinEvent -Path 'C:\Test\PowerShellCore Operational.evtx'			
ProviderName: PowerShellCore			
TimeCreated	Id	LevelDisplayName	Message
3/15/2019 09:54:54	4104	Warning	Creating
3/15/2019 09:37:13	40962	Information	PowerShe
3/15/2019 07:56:24	4104	Warning	Creating
•••			
3/7/2019 10:53:22	40961	Information	PowerShe
3/7/2019 10:53:22	8197	Verbose	Runspace
3/7/2019 10:53:22	8195	Verbose	Opening N

The Get-WinEvent cmdlet gets log information from the computer. The Path parameter specifies the directory and filename. The MaxEvents parameter specifies that 100 records are displayed, from newest to oldest.

### **Example 14: Event Tracing for Windows**

Event Tracing for Windows (ETW) writes events to the log as events occur. The events are stored in the order of oldest to newest. An archived ETW file is saved as an <code>.etl</code> such as **TraceLog.etl**. The events are listed in the order in which they are written to the log, so the *Oldest* parameter is required.

```
Get-WinEvent -Path 'C:\Tracing\TraceLog.etl' -Oldest |
Sort-Object -Property TimeCreated -Descending |
Select-Object -First 100
```

The Get-WinEvent cmdlet gets log information from the archived file. The Path parameter specifies the directory and file name. The Oldest parameter is used to output events in the order they are written, oldest to newest. The objects are sent down the pipeline to the Sort-Object cmdlet Sort-Object sorts the objects in descending order by the value of the TimeCreated property. The objects are sent down the pipeline to the Select-Object cmdlet that displays the 100 newest events.

# Example 15: Get events from an event trace log

This example shows how to get the events from an event trace log file (.et1) and an archived Windows PowerShell log file (.evtx). You can combine multiple file types in a single command. Because the files contain the same type of .NET Framework object, EventLogRecord, you can filter them with the same properties. The command requires the Oldest parameter because it is reading from an .et1 file, but the Oldest parameter applies to each file.

```
Get-WinEvent -Path 'C:\Tracing\TraceLog.etl', 'C:\Test\Window
Where-Object { $_.Id -eq '403' }
```

The Get-WinEvent cmdlet gets log information from the archived files. The Path parameter uses a comma-separated list to specify each files directory and file name. The Oldest parameter is used to output events in the order they are written, oldest to newest. The objects are sent down the pipeline to the Where-Object cmdlet. Where-Object uses a script block to find events with an Id of 403. The \$\_ variable represents the current object in the pipeline and Id is the Event Id property.

### **Example 16: Filter event log results**

This example shows a variety of methods to filter and select events from an event log. All of these commands get events that occurred in the last 24-hours from the **Windows PowerShell** event log. The filter methods are more efficient than using the Where-Object cmdlet. Filters are applied as the objects are retrieved. Where-Object retrieves all of the objects, then applies filters to all of the objects.

```
# Using the Where-Object cmdlet:
$Yesterday = (Get-Date) - (New-TimeSpan -Day 1)
Get-WinEvent -LogName 'Windows PowerShell' | Where-Object {
# Using the FilterHashtable parameter:
$Yesterday = (Get-Date) - (New-TimeSpan -Day 1)
Get-WinEvent -FilterHashtable @{ LogName='Windows PowerShell
# Using the FilterXML parameter:
$xmlQuery = @'
<QueryList>
  <Query Id="0" Path="Windows PowerShell">
    <Select Path="System">*[System[(Level=3) and
        TimeCreated[timediff(@SystemTime) <= 86400000]]]
  </Query>
</QueryList>
Get-WinEvent -FilterXML $xmlQuery
# Using the FilterXPath parameter:
$XPath = '*[System[Level=3 and TimeCreated[timediff(@System]
Get-WinEvent -LogName 'Windows PowerShell' -FilterXPath $XPa
```

# Example 17: Use FilterHashtable to get events from the Application log

This example uses the **FilterHashtable** parameter to get events from the **Application** log. The hash table uses **key/value** pairs. For more information about the **FilterHashtable** parameter, see Creating Get-WinEvent queries with FilterHashtable. For more information about hash tables, see about\_Hash\_Tables.

```
$Date = (Get-Date).AddDays(-2)
Get-WinEvent -FilterHashtable @{ LogName='Application'; Star
```

The Get-Date cmdlet uses the AddDays method to get a date that is two days before the current date. The date object is stored in the \$Date variable.

The Get-WinEvent cmdlet gets log information. The FilterHashtable parameter is used to filter the output. The LogName key specifies the value as the Application log. The StartTime key uses the value stored in the \$Date variable. The Id key uses an Event Id value, 1003.

# Example 18: Use FilterHashtable to get application errors

This example uses the **FilterHashtable** parameter to find Internet Explorer application errors that occurred within the last week.

```
$StartTime = (Get-Date).AddDays(-7)
Get-WinEvent -FilterHashtable @{
  Logname='Application'
  ProviderName='Application Error'
  Data='iexplore.exe'
  StartTime=$StartTime
}
```

The Get-Date cmdlet uses the **AddDays** method to get a date that is seven days before the current date. The date object is stored in the \$StartTime variable.

The Get-WinEvent cmdlet gets log information. The FilterHashtable parameter is used to filter the output. The LogName key specifies the value as the Application log. The ProviderName key uses the value,

Application Error, which is the event's Source. The Data key uses the value iexplore.exe The StartTime key uses the value stored in \$StartTime variable.

# Example 19: Use SuppressHashFilter to filter application errors

Like Example 16 above, this example uses the **FilterHashtable** parameter to get events from the **Application** log. However, we add the **SuppressHashFilter** key to filter out **Information** level events.

```
$Date = (Get-Date).AddDays(-2)
$filter = @{
   LogName='Application'
   StartTime=$Date
   SuppressHashFilter=@{Level=4}
}
Get-WinEvent -FilterHashtable $filter
```

In this example, Get-WinEvent gets all events from the **Application** log for the last two days except those that have a **Level** of 4 (Information).

### **Parameters**

#### -ComputerName

Specifies the name of the computer that this cmdlet gets events from the event logs. Type the NetBIOS name, an IP address, or the fully qualified domain name (FQDN) of the computer. The default value is the local computer, **localhost**. This parameter accepts only one computer name at a time.

To get event logs from remote computers, configure the firewall port for the event log service to allow remote access.

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

#### Expand table

Туре:	String
Aliases:	Cn
Position:	Named
Default value:	Local computer
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

#### -Credential

Specifies a user account that has permission to perform this action. The default value 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, you are prompted for a password. If you type only the parameter name, you are prompted for both a username and a password.

#### **Expand table**

Туре:	PSCredential
Position:	Named
Default value:	Current user
Required:	False

Accept pipeline input:	False
Accept wildcard characters:	False

#### -FilterHashtable

Specifies a query in hash table format to select events from one or more event logs. The query contains a hash table with one or more **key/value** pairs.

Hash table queries have the following rules:

- Keys and values are case-insensitive.
- Wildcard characters are valid only in the values associated with the LogName and ProviderName keys.
- Each key can be listed only once in each hash table.
- The Path value takes paths to .et1, .evt, and .evtx log files.
- The **LogName**, **Path**, and **ProviderName** keys can be used in the same query.
- The UserID key can take a valid security identifier (SID) or a domain account name that can be used to construct a valid System.Security.Principal.NTAccount object.
- The **Data** value takes event data in an unnamed field. For example, events in classic event logs.
- <named-data> key represents a named event data field.

When Get-WinEvent cannot interpret a **key/value** pair, it interprets the key as a case-sensitive name for the event data in the event.

The valid Get-WinEvent key/value pairs are as follows:

- LogName= <String[]>
- ProviderName = <String[]>
- Path= <String[]>
- Keywords= <Long[]>
- **ID**= <Int32[]>
- Level= <Int32[]>

- StartTime = < DateTime >
- EndTime = < DateTime >
- UserID= <SID>
- Data = <String[]>
- <named-data> = <String[]>
- SuppressHashFilter= <Hashtable>

**Expand table** 

Туре:	Hashtable[]
Position:	0
Default value:	None
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	False

#### -FilterXml

Specifies a structured XML query that this cmdlet selects events from one or more event logs.

To generate a valid XML query, use the **Create Custom View** and **Filter Current Log** features in Windows Event Viewer. Use the items in the dialog box to create a query, and then click the XML tab to view the query in XML format. You can copy the XML from the XML tab into the value of the **FilterXml** parameter. For more information about the Event Viewer features, see Event Viewer Help.

Use an XML query to create a complex query that contains several XPath statements. The XML format also allows you to use a **Suppress XML** element that excludes events from the query. For more information about the XML schema for event log queries, see

Query Schema and the XML Event Queries section of Event Selection.

You may also create a **Suppress** element using the **FilterHashtable** parameter.

#### **Expand table**

Туре:	XmlDocument
Position:	0
Default value:	None
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	False

#### -FilterXPath

Specifies an XPath query that this cmdlet select events from one or more logs.

For more information about the XPath language, see XPath Reference and the Selection Filters section of Event Selection.

#### **Expand table**

Type:	String
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

#### -Force

Gets debug and analytic logs, in addition to other event logs. The **Force** parameter is required to get a debug or analytic log when the value of the name parameter includes wildcard characters.

By default, the Get-WinEvent cmdlet excludes these logs unless you specify the full name of a debug or analytic log.

#### **Expand table**

Туре:	SwitchParameter
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

#### -ListLog

Specifies the event logs. Enter the event log names in a commaseparated list. Wildcards are permitted. To get all the logs, use the asterisk (\*) wildcard.

#### **Expand table**

Туре:	String[]
Position:	0
Default value:	None
Required:	True
Accept pipeline input:	False

Accept wildcard characters:	True
'	

#### -ListProvider

Specifies the event log providers that this cmdlet gets. An event log provider is a program or service that writes events to the event log.

Enter the provider names in a comma-separated list. Wildcards are permitted. To get the providers of all the event logs on the computer, use the asterisk (\*) wildcard.

#### **Expand table**

Туре:	String[]
Position:	0
Default value:	None
Required:	True
Accept pipeline input:	False
Accept wildcard characters:	True

#### LogName

Specifies the event logs that this cmdlet get events from. Enter the event log names in a comma-separated list. Wildcards are permitted. You can also pipe log names to the Get-WinEvent cmdlet.

#### ① Note

PowerShell does not limit the amount of logs you can request. However, the Get-WinEvent cmdlet queries the Windows API which has a limit of 256. This can make it difficult to filter through all of your logs at one time. You can work around this

```
by using a foreach loop to iterate through each log like this:
Get-WinEvent -ListLog * | ForEach-Object{ Get-WinEvent
-LogName $_.Logname }
```

#### **Expand table**

Туре:	String[]
Position:	0
Default value:	None
Required:	False
Accept pipeline input:	True
Accept wildcard characters:	True

#### -MaxEvents

Specifies the maximum number of events that are returned. Enter an integer such as 100. The default is to return all the events in the logs or files.

#### **Expand table**

Туре:	Int64
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

#### -Oldest

Indicate that this cmdlet gets the events in oldest-first order. By default, events are returned in newest-first order.

This parameter is required to get events from .et1 and .evt files and from debug and analytic logs. In these files, events are recorded in oldest-first order, and the events can be returned only in oldest-first order.

#### **Expand table**

Туре:	SwitchParameter
Position:	Named
Default value:	None
Required:	False
Accept pipeline input:	False
Accept wildcard characters:	False

#### -Path

Specifies the path to the event log files that this cmdlet get events from. Enter the paths to the log files in a comma-separated list, or use wildcard characters to create file path patterns.

Get-WinEvent supports files with the .evt, .evtx, and .et1 file name extensions. You can include events from different files and file types in the same command.

#### **Expand table**

Туре:	String[]
Aliases:	PSPath
Position:	0

Default value:	None
Required:	True
Accept pipeline input:	True
Accept wildcard characters:	True

#### -ProviderName

Specifies, as a string array, the event log providers from which this cmdlet gets events. Enter the provider names in a comma-separated list, or use wildcard characters to create provider name patterns.

An event log provider is a program or service that writes events to the event log. It is not a PowerShell provider.

**Expand table** 

Туре:	String[]
Position:	0
Default value:	None
Required:	True
Accept pipeline input:	True
Accept wildcard characters:	True

### Inputs

#### **String**

You can pipe a LogName (string) to this cmdlet.

#### **XmlDocument**

You can pipe a FilterXML query to this cmdlet.

#### Hashtable

You can pipe a FilterHashtable query to this cmdlet.

### **Outputs**

#### **EventLogConfiguration**

With the **ListLog** parameter, this cmdlet returns **EventLogConfiguration** objects.

#### EventLogRecord

By default, this cmdlet returns **EventLogRecord** objects.

#### **ProviderMetadata**

With the **ListProvider** parameter, this cmdlet returns **ProviderMetadata** objects.

### **Notes**

Get-WinEvent is designed to replace the Get-EventLog cmdlet on computers running Windows Vista and later versions of Windows. Get-EventLog gets events only in classic event logs. Get-EventLog is retained for backward compatibility.

The Get-WinEvent and Get-EventLog cmdlets are not supported in Windows Pre-installation Environment (Windows PE).

### **Related Links**

- about\_Automatic\_Variables
- about\_Foreach
- about\_Hash\_Tables
- Creating Get-WinEvent queries with FilterHashtable

- Format-Table
- Group-Object
- Sort-Object
- Where-Object

### Collaborate with us on **GitHub**

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see our contributor guide.



### **PowerShell** feedback

PowerShell is an open source project. Select a link to provide feedback:

🖔 Open a documentation issue

Provide product feedback

Senglish (United States)

**✓** ✓ Your Privacy Choices

☆ Theme ∨

Manage cookies **Previous Versions** Blog ☑ Contribute Privacy ☑ Terms of Use Trademarks ☑

© Microsoft 2024