PowerShell Get-Help on command 'Get-WinEvent'

PS C:\Users\wahid> Get-Help Get-WinEvent

```
NAME
    Get-WinEvent

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


SYNTAX
    Get-WinEvent [[-LogName] <System.String[]>] [-ComputerName <System.String>]
    [-Credential <System.Management.Automation.PSCredential>] [-FilterXPath
    <System.String>] [-Force] [-MaxEvents <System.Int64>] [-Oldest]
    [<CommonParameters>]

    Get-WinEvent [-ListLog] <System.String[]> [-ComputerName <System.String>]
    [-Credential <System.Management.Automation.PSCredential>] [-Force]
    [<CommonParameters>]

    Get-WinEvent [-ListProvider] <System.String[]> [-ComputerName <System.String>]
    [-Credential <System.Management.Automation.PSCredential>] [<CommonParameters>]
    Get-WinEvent [-ProviderName] <System.String[]> [-ComputerName <System.String>]
    [-Credential <System.Management.Automation.PSCredential>] [-FilterXPath
    <System.String>] [-Force] [-MaxEvents <System.Int64>] [-Oldest]
    [<CommonParameters>]

    Get-WinEvent [-FilterHashtable] <System.Collections.Hashtable[]>
    [-ComputerName <System.String>] [-Credential
    <System.Management.Automation.PSCredential>] [-Force] [-MaxEvents
    <System.Int64>] [-Oldest] [<CommonParameters>]

    Get-WinEvent [-FilterXml] <System.Xml.XmlDocument> [-ComputerName
    <System.String>] [-Credential <System.Management.Automation.PSCredential>]
    [-MaxEvents <System.Int64>] [-Oldest] [<CommonParameters>]

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

DESCRIPTION

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 <kbd>CTRL</kbd>+<kbd>C</kbd>. 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.

PARAMETERS

-ComputerName <System.String>

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.

-Credential <System.Management.Automation.PSCredential>

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.

-FilterHashtable <System.Collections.Hashtable[]>

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 `.etl`, `.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.

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[]>`

-FilterXml <System.Xml.XmlDocument>

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 (/windows/win32/wes/queryschema-schema)and the XML Event Queries section of Event Selection (/previous-versions/aa385231(v=vs.85)).

-FilterXPath <System.String>

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

For more information about the XPath language, see XPath Reference (/previous-versions/dotnet/netframework-4.0/ms256115(v=vs.100))and the Selection Filters section of Event Selection (/previous-versions/aa385231(v=vs.85)).

-Force <System.Management.Automation.SwitchParameter>

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.

-ListLog <System.String[]>

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

-ListProvider <System.String[]>

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.

-LogName <System.String[]>

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.

> [!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 }`

-MaxEvents <System.Int64>

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.

-Oldest <System.Management.Automation.SwitchParameter>

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 `.etl` 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.

-Path <System.String[]>

    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 `.etl` file
    name extensions. You can include events from different files and file
    types in the same command.

-ProviderName <System.String[]>

    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.

<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: Get all the logs from a local computer ------

```
Get-WinEvent -ListLog *

LogMode   MaximumSizeInBytes RecordCount LogName
-------   ------------------ ----------- -------
Circular            15532032       14500 Application
Circular             1052672         117 Azure Information Protection
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 -------------

```
Get-WinEvent -ListLog Setup | Format-List -Property *
```

```
FileSize                       : 69632
IsLogFull                      : False
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
IsClassicLog                   : False
SecurityDescriptor             : O:BAG:SYD: ...
LogFilePath                    : %SystemRoot%\System32\Winevt\Logs\Setup.evtx
MaximumSizeInBytes             : 1052672
LogMode                        : Circular
OwningProviderName             : Microsoft-Windows-Eventlog
ProviderNames                  : {Microsoft-Windows-WUSA,
Microsoft-Windows-ActionQueue...
ProviderLevel                  :
ProviderKeywords               :
ProviderBufferSize             : 64
ProviderMinimumNumberOfBuffers : 0
ProviderMaximumNumberOfBuffers : 64
ProviderLatency                : 1000
```

```
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 --------

```
$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 log!'
   $ErrMsg += ' Try running this script with administrator privileges. '
   $ErrMsg += $_.Exception.Message
   Write-Error $ErrMsg
}
```

```
FileSize                       : 69632
IsLogFull                      : False
LastAccessTime                 : 3/13/2019 09:41:46
LastWriteTime                  : 3/13/2019 09:41:46
OldestRecordNumber             : 1
RecordCount                    : 23
LogName                        : Security
LogType                        : Administrative
LogIsolation                   : Custom
IsEnabled                      : True
IsClassicLog                   : True
SecurityDescriptor             : O:BAG:SYD: ...
LogFilePath                    :
```

```
%SystemRoot%\System32\Winevt\Logs\Security.evtx
MaximumSizeInBytes             : 1073741824
LogMode                        : Circular
OwningProviderName             :
ProviderNames                  : {Microsoft-Windows-WUSA,
Microsoft-Windows-ActionQueue...
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 -----------

```
Get-WinEvent -ListLog * -ComputerName localhost | Where-Object {
$_.RecordCount }

LogMode   MaximumSizeInBytes RecordCount LogName
-------   ------------------ ----------- -------
Circular            15532032       14546 Application
Circular             1052672         117 Azure Information Protection
Circular             1052672        2990 CxAudioSvcLog
Circular             1052672           9 MSFTVPN Setup
Circular             1052672         282 OAlerts
```

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 -------

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

LogMode MaximumSizeInBytes RecordCount LogName     ComputerName
 ------- ------------------ ----------- -------     ------------
Circular          15532032       14577 Application Server01
Circular          15532032        9689 Application Server02
Circular          15532032        5309 Application Server03
```

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

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

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

```
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

```
(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

```
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 --

```
(Get-WinEvent -ListProvider Microsoft-Windows-GroupPolicy).Events |
Format-Table Id, Description

Id  Description
  --  -----------
1500  The Group Policy settings for the computer were processed successfully...
```

```
1501  The Group Policy settings for the user were processed successfully...
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 -

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

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

```
Count  Name
-----  ----
    1  Error, PowerShellCore/Operational
   26  Information, Microsoft-Windows-Kernel-WHEA/Operational
  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 -Path 'C:\Test\Windows PowerShell.evtx'

ProviderName: PowerShell

TimeCreated              Id LevelDisplayName  Message
-----------              -- ----------------  -------
3/15/2019 13:54:13      403 Information       Engine state is changed from
Available to Stopped...
3/15/2019 13:54:13      400 Information       Engine state is changed from
None to Available...
3/15/2019 13:54:13      600 Information       Provider "Variable" is Started...
3/15/2019 13:54:13      600 Information       Provider "Function" is Started...
3/15/2019 13:54:13      600 Information       Provider "FileSystem" is
Started...
```

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

```
Get-WinEvent -Path 'C:\Test\PowerShellCore Operational.evtx' -MaxEvents 100
```

```
ProviderName: PowerShellCore

TimeCreated                 Id   LevelDisplayName  Message
-----------                 --   ----------------  -------
3/15/2019 09:54:54        4104   Warning           Creating Scriptblock text
(1 of 1):...
3/15/2019 09:37:13       40962   Information       PowerShell console is ready
for user input
3/15/2019 07:56:24        4104   Warning           Creating Scriptblock text
(1 of 1):...
...
3/7/2019 10:53:22        40961   Information       PowerShell console is
starting up
3/7/2019 10:53:22         8197   Verbose           Runspace state changed to
Opening
3/7/2019 10:53:22         8195   Verbose           Opening RunspacePool
```

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 ------------

```
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 --------

```
Get-WinEvent -Path 'C:\Tracing\TraceLog.etl', 'C:\Test\Windows
PowerShell.evtx' -Oldest |
  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 -------------

```
# Using the Where-Object cmdlet:
$Yesterday = (Get-Date) - (New-TimeSpan -Day 1)
Get-WinEvent -LogName 'Windows PowerShell' | Where-Object { $_.TimeCreated -ge
$Yesterday }

# Using the FilterHashtable parameter:
$Yesterday = (Get-Date) - (New-TimeSpan -Day 1)
Get-WinEvent -FilterHashtable @{ LogName='Windows PowerShell'; Level=3;
StartTime=$Yesterday }

# Using the FilterXML parameter:
$xmlQuery = @'
<QueryList>
  <Query Id="0" Path="Windows PowerShell">
```

```
    <Select Path="System">*[System[(Level=3) and
        TimeCreated[timediff(@SystemTime) &lt;= 86400000]]]</Select>
  </Query>
</QueryList>
'@
Get-WinEvent -FilterXML $xmlQuery

# Using the FilterXPath parameter:
$XPath = '*[System[Level=3 and TimeCreated[timediff(@SystemTime) &lt;=
86400000]]]'
Get-WinEvent -LogName 'Windows PowerShell' -FilterXPath $XPath
```

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

```
$Date = (Get-Date).AddDays(-2)
Get-WinEvent -FilterHashtable @{ LogName='Application'; StartTime=$Date;
Id='1003' }
```

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 --

```
$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.

REMARKS

To see the examples, type: "get-help Get-WinEvent -examples".

For more information, type: "get-help Get-WinEvent -detailed".

For technical information, type: "get-help Get-WinEvent -full".

For online help, type: "get-help Get-WinEvent -online"