

# Querying Event Logs

I noticed that there is a huge speed difference between using an XML Query and PowerShell Get-EventLog piped through Where-Object to filter event logs. Thanks to [this article](#), I learned how to use the XML Query via PowerShell, so you get the best of both worlds.

## Know your version

Here's different commands that will show you which version PowerShell you're running.

```
$PSVersionTable.PSVersion
```

```
Get-Host
```

```
$host
```

```
$host.version
```

## General concepts

There are two different cmdlets for accessing Windows Event Logs. Get-WinEvent is a newer version of Get-EventLog.

### Get-WinEvent

- You have access to more information
- Because you have more information, it might take more effort to filter the data

### Get-EventLog

- One clear advantage: you can use the **-After** and **-Before** attributes to easily filter results by date

## Filtering results

If you want to know how to filter the results, simply pipe the cmdlet to Get-Member:

```
Get-EventLog system -newest 1 | Get-Member
```

The output of the command clearly shows you the methods and properties returned:

```
PS C:\> Get-EventLog system -newest 1 | Get-Member
```

TypeName: System.Diagnostics.EventLogEntry#system/nhi/1074012975

Name	MemberType	Definition
----	-----	-----
Disposed	Event	System.EventHandler Disposed(System.Object, System.EventArgs)
CreateObjRef	Method	System.Runtime.Remoting.ObjRef CreateObjRef(type requestedType)
Dispose	Method	void Dispose(), void IDisposable.Dispose()
Equals	Method	bool Equals(System.Diagnostics.EventLogEntry otherEntry), bool Equals(System.Object obj)
GetHashCode	Method	int GetHashCode()
GetLifetimeService	Method	System.Object GetLifetimeService()
GetObjectData	Method	void ISerializable.GetObjectData(System.Runtime.Serialization.SerializationInfo info, System.Runtime.Serialization.StreamingContext context)
GetType	Method	type GetType()
InitializeLifetimeService	Method	System.Object InitializeLifetimeService()
ToString	Method	string ToString()
Category	Property	string Category {get;}
CategoryNumber	Property	int16 CategoryNumber {get;}
Container	Property	System.ComponentModel.IContainer Container {get;}
Data	Property	byte[] Data {get;}
EntryType	Property	System.Diagnostics.EventLogEntryType EntryType {get;}
Index	Property	int Index {get;}
InstanceId	Property	long InstanceId {get;}
MachineName	Property	string MachineName {get;}
Message	Property	string Message {get;}
ReplacementStrings	Property	string[] ReplacementStrings {get;}
Site	Property	System.ComponentModel.ISite Site {get;set;}
Source	Property	string Source {get;}
TimeGenerated	Property	datetime TimeGenerated {get;}
TimeWritten	Property	datetime TimeWritten {get;}

UserName	Property	string UserName {get;}
EventID	ScriptProperty	System.Object EventID {get=\$this.get_EventID() -band 0xFFFF;}

# Get-EventLog Examples

# Show available event logs and stats

```
Get-EventLog -List
```

# get the most recent 10 system log entries

# just change the LogName from System to Application, Security, etc. to access other logs

```
Get-EventLog -LogName System -Newest 10
```

# get all system logs from the last 4 hours

```
Get-EventLog -LogName System -After (Get-Date).AddHours(-4)
```

# get all system logs from the last 24 hours

```
Get-EventLog -LogName System -After (Get-Date).AddDays(-1)
```

# View specific event using the event Index

```
Get-EventLog -LogName System -Index [Event_Index_Number] | Format-List
```

# get the most recent 10 entries from a specific source

```
Get-EventLog -LogName System -Source Kerberos -Newest 10
```

```
Get-EventLog -LogName System -Source Microsoft-Windows-WLAN-AutoConfig -Newest 10
```

# Get system logs from the last 24 hours from Source WLAN-AutoConfig

```
Get-EventLog -LogName system -After (Get-Date).AddDays(-1) -Source Microsoft-Windows-WLAN-AutoConfig
```

# get the most recent 10 Error entries

```
Get-EventLog -LogName Application -EventType Error -Newest 10
```

```
Get-EventLog -LogName Security -EventType Error -Newest 10
```

```
Get-EventLog -LogName System -EventType Error -Newest 10
```

```
# Get list of Event Log Sources from the System log from the last 8 hours sorted by log count
Get-EventLog -LogName System -after (Get-Date).AddHours(-8) | Group-Object -Property Source -NoElement |
Select-Object -Property Count, Name | Sort-Object -Descending Count
```

```
# Find logins in the last 24 hours
Get-EventLog system -after (get-date).AddDays(-1) | where {$_.InstanceId -eq 7001}

# Find last computer start
$today = get-date -Hour 0 -Minute 0;
Get-EventLog system -after $today | sort -Descending | select -First 1
```

```
# Find logins and logoffs in the last 7 days
$logs = get-eventlog system -source Microsoft-Windows-Winlogon -After (Get-Date).AddDays(-7);
$res = @(); ForEach ($log in $logs) {if($log.instanceid -eq 7001) {$type = "Logon"} Elseif ($log.instanceid -eq
7002){$type="Logoff"} Else {Continue} $res += New-Object PSObject -Property @{Time = $log.TimeWritten;
"Event" = $type; User = (New-Object System.Security.Principal.SecurityIdentifier
$Log.ReplacementStrings[1]).Translate([System.Security.Principal.NTAccount])}};
$res
```

# Get-WinEvent Examples

```
# Get user Logon / Logoff events
Get-WinEvent -FilterHashtable @{
    LogName='System'
    ProviderName='Microsoft-Windows-Winlogon'
    ID=7001,7002
}
```

```
Get-WinEvent -FilterHashtable @{
    LogName='Application'
    ProviderName='.NET Runtime'
    Keywords=36028797018963968
    ID=1023
    Level=2
}
```

# Formatting output

## Source

You can see what formatters are available on any system using the following command

```
Get-Command -Verb Format -Module Microsoft.PowerShell.Utility
```

Below is the output on Windows 11 as of 10/31/2023

```
PS C:\> Get-Command -Verb Format -Module Microsoft.PowerShell.Utility
```

CommandType	Name	Version	Source
-----	----	-----	-----
Function	Format-Hex	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Format-Custom	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Format-List	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Format-Table	3.1.0.0	Microsoft.PowerShell.Utility
Cmdlet	Format-Wide	3.1.0.0	Microsoft.PowerShell.Utility

You also have access to the following cmdlets for other output formats

## Export-CliXml

```
# Export-Clixml exports an XML representation of an object or objects and stores it in a file
Get-Acl C:\Windows | Export-CliXml -Path .\c-windows-acl.xml
```

```
# You can use Import-CliXml to save the stored object or objects back to a variable
$WindowsFolderACL = Import-CliXml -Path .\c-windows-acl.xml
```

## Export-Csv

```
# Export-Csv - Add an example later
```

# Redirecting data with Out-\* cmdlets

## Source

```
Out-Host -Paging
```

```
Get-Process | Out-Host -Paging | Format-List
```

```
Get-Process | Format-List | Out-Host -Paging
```

```
Get-Command | Out-Null
```

```
Get-Command Get-Command | Out-Printer -Name 'Microsoft Office Document Image Writer'
```

```
Get-Process | Out-File -FilePath C:\temp\processlist.txt
```

```
Get-Command | Out-File -FilePath c:\temp\output.txt -Width 2147483647
```

```
Get-EventLog -LogName System -After (Get-Date).AddDays(-1) -EntryType Error | Out-GridView
```

# Grouping output

Remember to use the Get-Member cmdlet to see what properties you can use with Sort-Object and -GroupBy

```
Get-Service -Name win* | Sort-Object StartType | Format-Table -GroupBy StartType -AutoSize
```

```
PS C:\> Get-Service -Name win* | Sort-Object StartType | Format-Table -GroupBy StartType -AutoSize
```

StartType: Automatic

Status	Name	DisplayName
--------	------	-------------

Running	Winmgmt	Windows Management Instrumentation
---------	---------	------------------------------------

Running	WinDefend	Microsoft Defender Antivirus Service
---------	-----------	--------------------------------------

StartType: Manual

Status	Name	DisplayName
--------	------	-------------

Stopped	WinRM	Windows Remote Management (WS-Management)
---------	-------	---

Running	WinHttpAutoProxySvc	WinHTTP Web Proxy Auto-Discovery Service
---------	---------------------	--

# Querying for specific logs

## System uptime related logs

For the actual current system uptime via PowerShell, [look here](#). The code below will show actual related event log entries.

Use this XML Filter in the Windows Event Viewer to create a custom filtered view of Kernel-General "The operating system started at system time..." events.

Event ID	Description
12	
13	
41	The system has rebooted without cleanly shutting down first. This error could be caused if the system stopped responding, crashed, or lost power unexpectedly.
1074	Logged when an app (ex: Windows Update) causes the system to restart, or when a user initiates a restart or shutdown.
6006	Logged as a clean shutdown. It gives the message "The Event log service was stopped".

6008

Logged as a dirty shutdown. It gives the message "The previous system shutdown at time on date was unexpected".

```
$query = @"
<QueryList>
  <Query Id="0" Path="System">
    <Select Path="System">*[System[(EventID='12')]]</Select>
    <Select Path="System">*[System[(EventID='13')]]</Select>
    <Select Path="System">*[System[(EventID='41')]]</Select>
    <Select Path="System">*[System[(EventID='1074')]]</Select>
    <Select Path="System">*[System[(EventID='6006')]]</Select>
    <Select Path="System">*[System[(EventID='6008')]]</Select>
  </Query>
</QueryList>
"@
```

Get-WinEvent -FilterXml \$query | Format-List

# Finding account lockouts.

## XML Query

Use this XML Filter in the Windows Event Viewer to create a custom filtered view displaying account lockouts.

```
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">
      *[
        System[(EventID='4740')]
      ]
    </Select>
  </Query>
</QueryList>
```

## PowerShell Script - Slow method



```
Get-EventLog -LogName Security | Where-Object {$_.EventID -eq 4740} |  
Select-Object -Property TimeGenerated, Source, EventID, InstanceId, Message
```

## PowerShell Script - Fast method

```
$query = @"  
<QueryList>  
  <Query Id="0" Path="Security">  
    <Select Path="Security">  
      *[*  
        System[(EventID='4740')]  
      ]  
    </Select>  
  </Query>  
</QueryList>  
"@  
  
Get-WinEvent -FilterXml $query | Format-List
```

# Finding account lockouts for a particular user.

## XML Query

Use this XML Filter in the Windows Event Viewer to create a custom filtered view displaying account lockouts for the administrator user.

```
<QueryList>  
  <Query Id="0" Path="Security">  
    <Select Path="Security">  
      *[*  
        EventData[Data[@Name='TargetUserName']='administrator']  
        and  
        System[(EventID='4740')]  
      ]  
    </Select>  
  </Query>  
</QueryList>
```

# PowerShell Script - Fast method

```
$query = @"
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">
      *
      EventData[Data[@Name='TargetUserName']='administrator']
      and
      System[(EventID='4740')]
    </Select>
  </Query>
</QueryList>
"@
```

```
Get-WinEvent -FilterXml $query | Format-List
```

## NPS + Azure MFA Logs - XML Query

XML Filter for custom filtered view that suppresses accounting event logs.

```
<QueryXML>
  <QueryList>
    <Query Id="0" Path="System">
      <Select Path="System">*[System[Provider[@Name='NPS']]]</Select>
      <Select Path="Security">*[System[Provider[@Name='Microsoft-Windows-Security-Auditing'] and Task = 12552]]</Select>
      <Suppress Path="Security">*[System[Provider[@Name='Microsoft-Windows-Security-Auditing'] and Task = 12552 and (Data='Network Policy Server discarded the accounting request for a user.')]</Suppress>
      <Select Path="Security">*[System[Provider[@Name='Microsoft-Windows-Security-Auditing']]] and
      *[EventData[Data[@Name='LogonProcessName'] and (Data='IAS')]]</Select>
      <Select Path="AuthNOptCh">*</Select>
      <Select Path="AuthZAdminCh">*</Select>
      <Select Path="AuthZOptCh">*</Select>
    </Query>
  </QueryList>
</QueryXML>
```

# NPS Logs - XML Query

XML Filter for custom filtered view that suppresses accounting event logs.

```
<QueryXML>
  <QueryList>
    <Query Id="0" Path="System">
      <Select Path="System">*[System[Provider[@Name='NPS']]]</Select>
      <Select Path="Security">*[System[Provider[@Name='Microsoft-Windows-Security-Auditing'] and Task = 12552]]</Select>
      <Suppress Path="Security">*[System[Provider[@Name='Microsoft-Windows-Security-Auditing'] and Task = 12552 and (Data='Network Policy Server discarded the accounting request for a user.')]</Suppress>
      <Select Path="Security">*[System[Provider[@Name='Microsoft-Windows-Security-Auditing']]] and
      *[EventData[Data[@Name='LogonProcessName'] and (Data='IAS')]]</Select>
    </Query>
  </QueryList>
</QueryXML>
```

## Disk logs

### XML Query

XML Filter for custom filtered view for disk events.

```
<QueryList>
  <Query Id="0" Path="System">
    <Select Path="System">*[System[Provider[@Name='disk']]]</Select>
  </Query>
</QueryList>
```

## VPN Client Logs

PowerShell Query

```
$query = @'
<QueryList>
  <Query Id="0" Path="Application">
    <Select Path="Application">*[System[Provider[@Name='RasAuto' or @Name='RasCfg' or
```

```

@Name='RasClient' or @Name='Rasman' or @Name='Microsoft-Windows-RasServer' or @Name='Microsoft-
Windows-RasSstp' or @Name='Microsoft-Windows-EapMethods-RasChap' or @Name='Microsoft-Windows-
NcdAutoSetup' or @Name='Microsoft-Windows-NCSI' or @Name='Microsoft-Windows-
NetworkProfile']]]</Select>

    <Select Path="System">*[System[Provider[@Name='RasAuto' or @Name='RasCfg' or @Name='RasClient' or
@Name='Rasman' or @Name='Microsoft-Windows-RasServer' or @Name='Microsoft-Windows-RasSstp' or
@Name='Microsoft-Windows-EapMethods-RasChap' or @Name='Microsoft-Windows-NcdAutoSetup' or
@Name='Microsoft-Windows-NCSI' or @Name='Microsoft-Windows-NetworkProfile']]]</Select>

</Query>
</QueryList>

"@

$vpnEvents = Get-WinEvent -FilterXml $query -Oldest

# Displays events from the last 24 hours grouped by ProviderName
# This is the best view for easily browsing
$vpnEvents | ?{$_.TimeCreated -ge (Get-Date).Addhours(-24)}

```

```

# Displays events from the last 24 hours as a time sorted list
$vpnEvents | ?{$_.TimeCreated -ge (Get-Date).Addhours(-24)} | Format-List

```

## XML Query

```

<QueryList>
  <Query Id="0" Path="Application">
    <Select Path="Application">*[System[Provider[@Name='RasAuto' or @Name='RasCfg' or
@Name='RasClient' or @Name='Rasman' or @Name='Microsoft-Windows-RasServer' or @Name='Microsoft-
Windows-RasSstp' or @Name='Microsoft-Windows-EapMethods-RasChap' or @Name='Microsoft-Windows-
NcdAutoSetup' or @Name='Microsoft-Windows-NCSI' or @Name='Microsoft-Windows-
NetworkProfile']]]</Select>

    <Select Path="System">*[System[Provider[@Name='RasAuto' or @Name='RasCfg' or @Name='RasClient' or
@Name='Rasman' or @Name='Microsoft-Windows-RasServer' or @Name='Microsoft-Windows-RasSstp' or
@Name='Microsoft-Windows-EapMethods-RasChap' or @Name='Microsoft-Windows-NcdAutoSetup' or
@Name='Microsoft-Windows-NCSI' or @Name='Microsoft-Windows-NetworkProfile']]]</Select>

  </Query>
</QueryList>

```

# Searching for Wired/WLAN-AutoConfig related errors

[Original source](#)

## Wired-AutoConfig

```
#Powershell
$addhours = 12;

# Setup filter for error only logs
$filter = @{ LogName = "Microsoft-Windows-Wired-AutoConfig/Operational"
StartTime = [DateTime]::Now.AddHours($addhours*-1)
EndTime = [DateTime]::Now
Level = 2
}

Write-Host ([DateTime]::Now.AddHours($addhours*-1))

Write-Host ([DateTime]::Now)

$Events = Get-Winevent -FilterHashtable $filter


# Parse out the event message data
ForEach ($Event in $Events) {
    # Convert the event to XML
    $eventXML = [xml]$Event.ToXml()
    # Iterate through each one of the XML message properties
    For ($i=0; $i -lt $eventXML.Event.EventData.Data.Count; $i++) {
        # Append these as object properties
        Add-Member -InputObject $Event -MemberType NoteProperty -Force -Name
        $eventXML.Event.EventData.Data[$i].name -Value $eventXML.Event.EventData.Data[$i].'#text'
    }
}
```

```
# Show results stored in variable
```

```
$Events | Format-List
```

## WLAN-AutoConfig

```
#Powershell
```

```
$addhours = 12;
```

```
# Setup filter for error only logs
```

```
$filter = @{ LogName = "Microsoft-Windows-WLAN-AutoConfig/Operational"
```

```
StartTime = [DateTime]::Now.AddHours($addhours*-1)
```

```
EndTime = [DateTime]::Now
```

```
Level = 2
```

```
}
```

```
Write-Host ([DateTime]::Now.AddHours($addhours*-1))
```

```
Write-Host ([DateTime]::Now)
```

```
$Events = Get-Winevent -FilterHashtable $filter
```

```
# Parse out the event message data
```

```
ForEach ($Event in $Events) {
```

```
    # Convert the event to XML
```

```
    $eventXML = [xml]$Event.ToXml()
```

```
    # Iterate through each one of the XML message properties
```

```
    For ($i=0; $i -lt $eventXML.Event.EventData.Data.Count; $i++) {
```

```
        # Append these as object properties
```

```
        Add-Member -InputObject $Event -MemberType NoteProperty -Force -Name
```

```
$eventXML.Event.EventData.Data[$i].name -Value $eventXML.Event.EventData.Data[$i].'#text'
```

```
    }
```

```
}
```

```
$Events | Select-Object id, MachineName, ProcessId, TimeCreated, Adapter, LocalMac, SSID, Cipher, Auth,
```

```
# Show results stored in variable
```

```
$Events | Select-Object id, MachineName, ProcessId, TimeCreated, Adapter, LocalMac, SSID, Cipher, Auth,  
PeerMac | Format-List
```

## Show available wireless profiles and available wireless networks

```
# show profiles
```

```
netsh wlan show profiles
```

```
# show available networks
```

```
netsh wlan show networks
```

## Duo Security Events

```
# Get Duo Security related events
```

```
Get-WinEvent -FilterHashtable @{  
    LogName='Application'  
    ProviderName='Duo Security'  
}
```

Sources: [1](#)

Revision #24

Created 1 January 2021 23:57:13 by bluecrow76

Updated 1 July 2024 21:28:42 by bluecrow76