Creating Get-WinEvent queries with FilterHashtable

To read the original June 3, 2014 **Scripting Guy** blog post, see Use FilterHashTable to Filter Event Log with PowerShell.

This article is an excerpt of the original blog post and explains how to use the `Get-WinEvent` cmdlet's **FilterHashtable** parameter to filter event logs. PowerShell's `Get-WinEvent` cmdlet is a powerful method to filter Windows event and diagnostic logs. Performance improves when a `Get-WinEvent` query uses the **FilterHashtable** parameter.

When you work with large event logs, it's not efficient to send objects down the pipeline to a `Where-Object` command. Prior to PowerShell 6, the `Get-EventLog` cmdlet was another option to get log data. For example, the following commands are inefficient to filter the **Microsoft-Windows-Defrag** logs:

PowerShellCopy
```
Get-EventLog -LogName Application | Where-Object Source -Match defrag

Get-WinEvent -LogName Application | Where-Object { $_.ProviderName -Match 'defrag' }
```

The following command uses a hash table that improves the performance:

PowerShellCopy
```
Get-WinEvent -FilterHashtable @{
   LogName='Application'
   ProviderName='*defrag'
}
```

Blog posts about enumeration

This article presents information about how to use enumerated values in a hash table. For more information about enumeration, read these **Scripting Guy** blog posts. To create a function that returns the enumerated values, see Enumerations and Values. For more information, see the Scripting Guy series of blog posts about enumeration.

Hash table key-value pairs

To build efficient queries, use the `Get-WinEvent` cmdlet with the **FilterHashtable** parameter. **FilterHashtable** accepts a hash table as a filter to get specific information from Windows event logs. A hash table uses **key-value** pairs. For more information about hash tables, see about_Hash_Tables.

If the **key-value** pairs are on the same line, they must be separated by a semicolon. If each **key-value** pair is on a separate line, the semicolon isn't needed. For example, this article places **key-value** pairs on separate lines and doesn't use semicolons.

This sample uses several of the **FilterHashtable** parameter's **key-value** pairs. The completed query includes **LogName**, **ProviderName**, **Keywords**, **ID**, and **Level**.

The accepted **key-value** pairs are shown in the following table and are included in the documentation for the Get-WinEvent **FilterHashtable** parameter.

The following table displays the key names, data types, and whether wildcard characters are accepted for a data value.

Key name	Value data type	Accepts wildcard characters?
LogName	`<String[]>`	Yes
ProviderName	`<String[]>`	Yes
Path	`<String[]>`	No
Keywords	`<Long[]>`	No
ID	`<Int32[]>`	No
Level	`<Int32[]>`	No
StartTime	`<DateTime>`	No
EndTime	`<DateTime>`	No
UserID	`<SID>`	No
Data	`<String[]>`	No
`<named-data>`	`<String[]>`	No

The `<named-data>` key represents a named event data field. For example, the Perflib event 1008 can contain the following event data:

XMLCopy

```xml
<EventData>
  <Data Name="Service">BITS</Data>
  <Data Name="Library">C:\Windows\System32\bitsperf.dll</Data>
  <Data Name="Win32Error">2</Data>
</EventData>
```

You can query for these events using the following command:

PowerShellCopy

```powershell
Get-WinEvent -FilterHashtable @{LogName='Application'; 'Service'='Bits'}
```

Note

The ability to query for `<named-data>` was added in PowerShell 6.

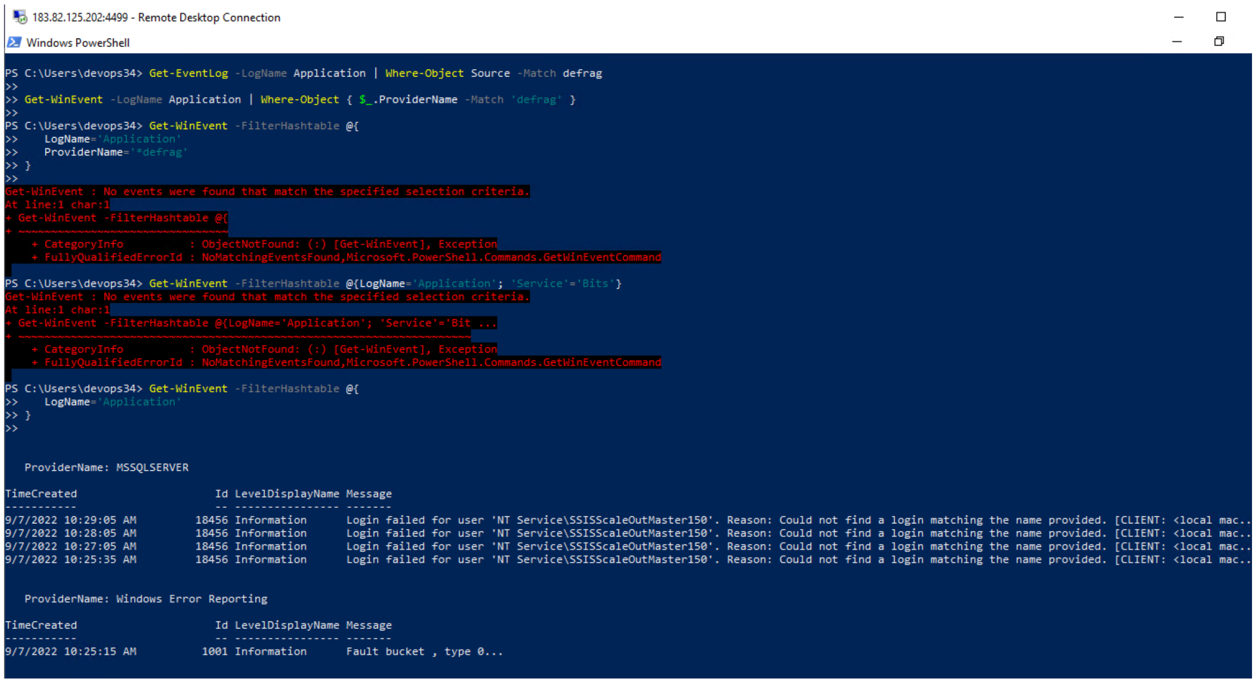
Building a query with a hash table

To verify results and troubleshoot problems, it helps to build the hash table one **key-value** pair at a time. The query gets data from the **Application** log. The hash table is equivalent to `Get-WinEvent -LogName Application`.

To begin, create the `Get-WinEvent` query. Use the **FilterHashtable** parameter's **key-value** pair with the key, **LogName**, and the value, **Application**.

PowerShellCopy
```
Get-WinEvent -FilterHashtable @{
   LogName='Application'
}
```

```
Output :
```



```
183.82.125.202:4499 - Remote Desktop Connection
Windows PowerShell

PS C:\Users\devops34> Get-EventLog -LogName Application | Where-Object Source -Match defrag
>>
>> Get-WinEvent -LogName Application | Where-Object { $_.ProviderName -Match 'defrag' }
>>
PS C:\Users\devops34> Get-WinEvent -FilterHashtable @{
>>     LogName='Application'
>>     ProviderName='*defrag'
>> }
>>
Get-WinEvent : No events were found that match the specified selection criteria.
At line:1 char:1
+ Get-WinEvent -FilterHashtable @{
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand

PS C:\Users\devops34> Get-WinEvent -FilterHashtable @{LogName='Application'; 'Service'='Bits'}
Get-WinEvent : No events were found that match the specified selection criteria.
At line:1 char:1
+ Get-WinEvent -FilterHashtable @{LogName='Application'; 'Service'='Bit ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand

PS C:\Users\devops34> Get-WinEvent -FilterHashtable @{
>>     LogName='Application'
>> }
>>


   ProviderName: MSSQLSERVER

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
9/7/2022 10:29:05 AM         18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local mac..
9/7/2022 10:28:05 AM         18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local mac..
9/7/2022 10:27:05 AM         18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local mac..
9/7/2022 10:25:35 AM         18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local mac..


   ProviderName: Windows Error Reporting

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
9/7/2022 10:25:15 AM          1001 Information      Fault bucket , type 0...
```

PowerShellCopy
```
Get-WinEvent -FilterHashtable @{
   LogName='Application'
   ProviderName='.NET Runtime'
}
```
 Note

For some event providers, the correct **ProviderName** can be obtained by looking on the **Details** tab in **Event Properties**. For example, events where the **Source** field shows `Defrag`, the correct **ProviderName** is `Microsoft-Windows-Defrag`.

If your query needs to get data from archived event logs, use the **Path** key. The **Path** value specifies the full path to the log file. For more information, see the **Scripting Guy** blog post, Use PowerShell to Parse Saved Event Logs for Errors.

Using enumerated values in a hash table

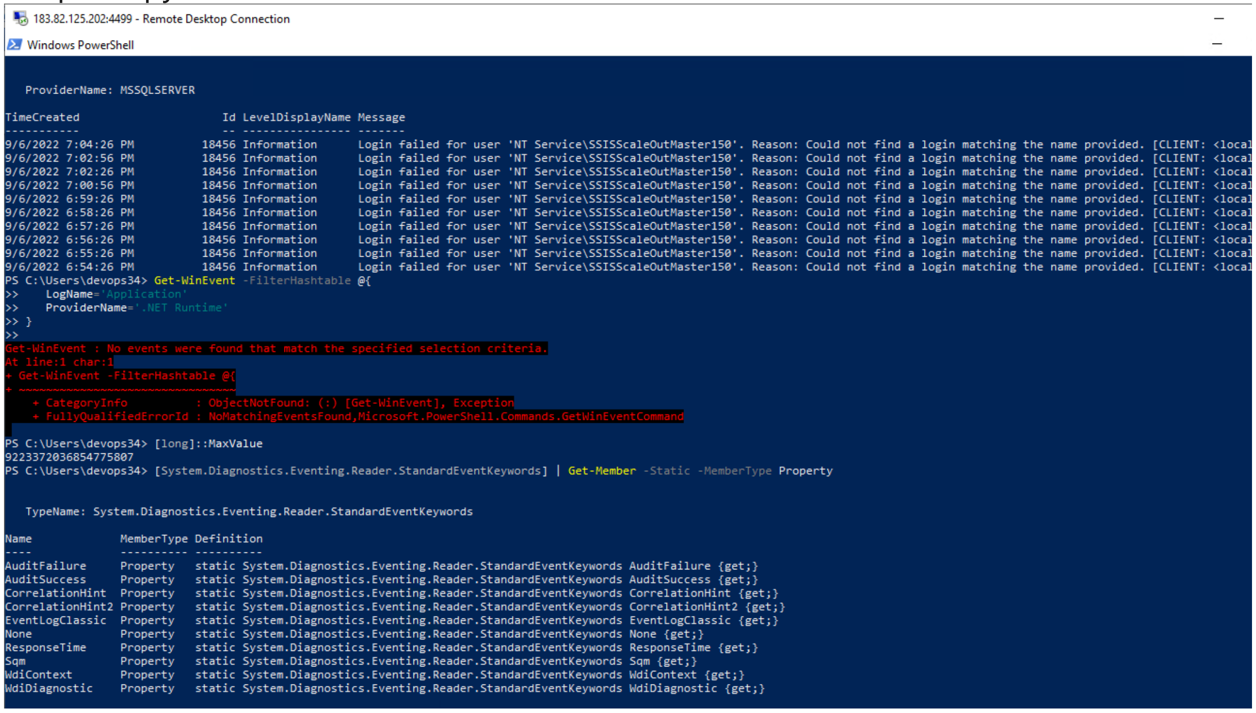
Keywords is the next key in the hash table. The **Keywords** data type is an array of the `[long]` value type that holds a large number. Use the following command to find the maximum value of `[long]`:

PowerShellCopy
```
[long]::MaxValue
```

Use the following command to display the `StandardEventKeywords` property names.

PowerShellCopy
```
[System.Diagnostics.Eventing.Reader.StandardEventKeywords] | Get-Member -Static -
MemberType Property
```

OutputCopy



183.82.125.202:4499 - Remote Desktop Connection

Windows PowerShell

```
   ProviderName: MSSQLSERVER

TimeCreated                     Id LevelDisplayName Message
-----------                     -- ---------------- -------
9/6/2022 7:04:26 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 7:02:56 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 7:02:26 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 7:00:56 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 6:59:26 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 6:58:26 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 6:57:26 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 6:56:26 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 6:55:26 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
9/6/2022 6:54:26 PM          18456 Information      Login failed for user 'NT Service\SSISScaleOutMaster150'. Reason: Could not find a login matching the name provided. [CLIENT: <local
PS C:\Users\devops34> Get-WinEvent -FilterHashtable @{
>>     LogName='Application'
>>     ProviderName='.NET Runtime'
>> }
>>
Get-WinEvent : No events were found that match the specified selection criteria.
At line:1 char:1
+ Get-WinEvent -FilterHashtable @{
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand

PS C:\Users\devops34> [long]::MaxValue
9223372036854775807
PS C:\Users\devops34> [System.Diagnostics.Eventing.Reader.StandardEventKeywords] | Get-Member -Static -MemberType Property


   TypeName: System.Diagnostics.Eventing.Reader.StandardEventKeywords

Name             MemberType Definition
----             ---------- ----------
AuditFailure     Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords AuditFailure {get;}
AuditSuccess     Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords AuditSuccess {get;}
CorrelationHint  Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords CorrelationHint {get;}
CorrelationHint2 Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords CorrelationHint2 {get;}
EventLogClassic  Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords EventLogClassic {get;}
None             Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords None {get;}
ResponseTime     Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords ResponseTime {get;}
Sqm              Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords Sqm {get;}
WdiContext       Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords WdiContext {get;}
WdiDiagnostic    Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords WdiDiagnostic {get;}
```

Update the hash table and include the **key-value** pair with the key, **Keywords**, and the **EventLogClassic** enumeration value, **36028797018963968**.

PowerShellCopy
```
Get-WinEvent -FilterHashtable @{
   LogName='Application'
   ProviderName='.NET Runtime'
   Keywords=36028797018963968
}
```

Keywords static property value (optional)

The **Keywords** key is enumerated, but you can use a static property name in the hash table query. Rather than using the returned string, the property name must be converted to a value with the **Value__** property.

For example, the following script uses the **Value__** property.

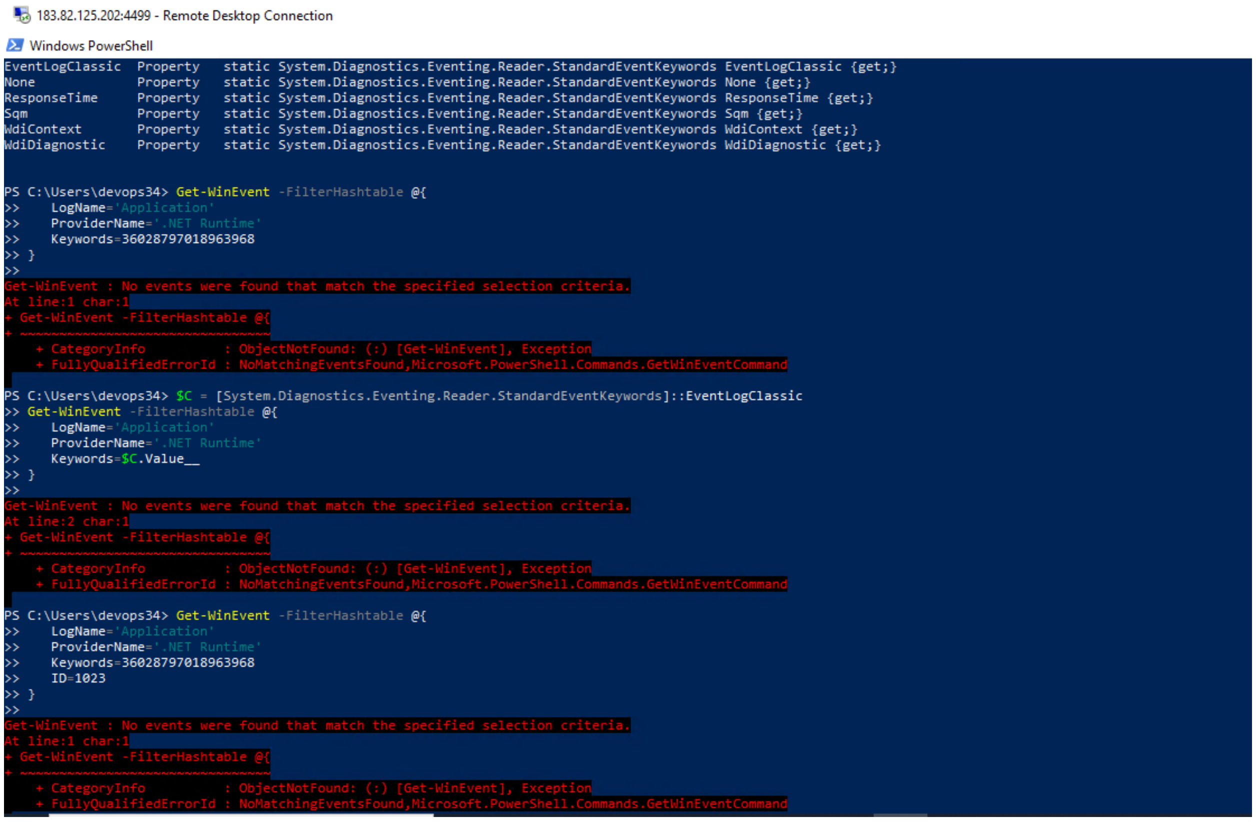
PowerShellCopy
```
$C = [System.Diagnostics.Eventing.Reader.StandardEventKeywords]::EventLogClassic
Get-WinEvent -FilterHashtable @{
   LogName='Application'
   ProviderName='.NET Runtime'
   Keywords=$C.Value__
}
```

Filtering by Event Id

To get more specific data, the query's results are filtered by **Event Id**. The **Event Id** is referenced in the hash table as the key **ID** and the value is a specific **Event Id**. The **Windows Event Viewer** displays the **Event Id**. This example uses **Event Id 1023**.

Update the hash table and include the **key-value** pair with the key, **ID** and the value, **1023**.

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



```
EventLogClassic  Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords EventLogClassic {get;}
None             Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords None {get;}
ResponseTime     Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords ResponseTime {get;}
Sqm              Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords Sqm {get;}
WdiContext       Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords WdiContext {get;}
WdiDiagnostic    Property   static System.Diagnostics.Eventing.Reader.StandardEventKeywords WdiDiagnostic {get;}


PS C:\Users\devops34> Get-WinEvent -FilterHashtable @{
>>     LogName='Application'
>>     ProviderName='.NET Runtime'
>>     Keywords=36028797018963968
>> }
>>
Get-WinEvent : No events were found that match the specified selection criteria.
At line:1 char:1
+ Get-WinEvent -FilterHashtable @{
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand

PS C:\Users\devops34> $C = [System.Diagnostics.Eventing.Reader.StandardEventKeywords]::EventLogClassic
>> Get-WinEvent -FilterHashtable @{
>>     LogName='Application'
>>     ProviderName='.NET Runtime'
>>     Keywords=$C.Value__
>> }
>>
Get-WinEvent : No events were found that match the specified selection criteria.
At line:2 char:1
+ Get-WinEvent -FilterHashtable @{
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand

PS C:\Users\devops34> Get-WinEvent -FilterHashtable @{
>>     LogName='Application'
>>     ProviderName='.NET Runtime'
>>     Keywords=36028797018963968
>>     ID=1023
>> }
>>
Get-WinEvent : No events were found that match the specified selection criteria.
At line:1 char:1
+ Get-WinEvent -FilterHashtable @{
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand
```

Filtering by Level

To further refine the results and include only events that are errors, use the **Level** key. **Windows Event Viewer** displays the **Level** as string values, but they are enumerated values. In the hash table, if you use the **Level** key with a string value, an error message is displayed.

Level has values such as **Error**, **Warning**, or **Informational**. Use the following command to display the `StandardEventLevel` property names.

PowerShellCopy

```
[System.Diagnostics.Eventing.Reader.StandardEventLevel] | Get-Member -Static -MemberType Property
```

PowerShellCopy

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

Level static property in enumeration (optional)

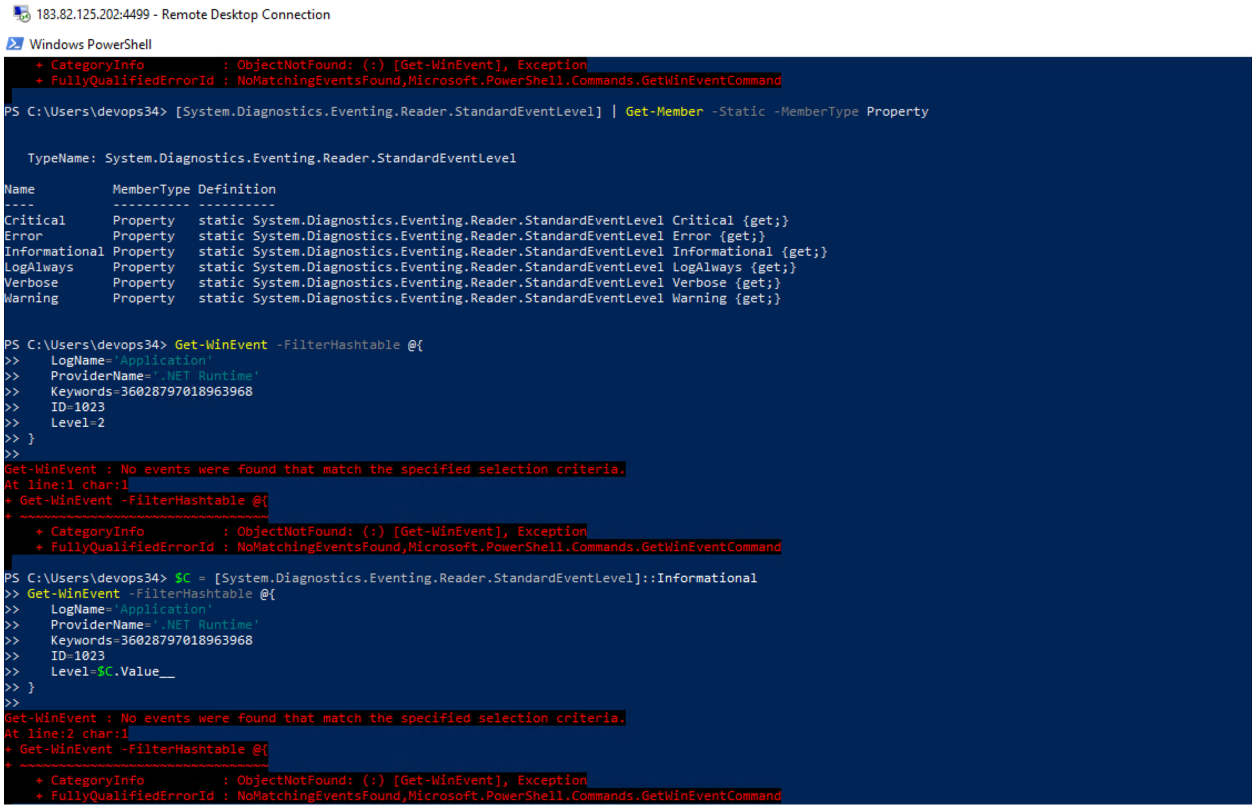
The **Level** key is enumerated, but you can use a static property name in the hash table query. Rather than using the returned string, the property name must be converted to a value with the **Value__** property.

For example, the following script uses the **Value__** property.

PowerShellCopy

```powershell
$C = [System.Diagnostics.Eventing.Reader.StandardEventLevel]::Informational
Get-WinEvent -FilterHashtable @{
    LogName='Application'
    ProviderName='.NET Runtime'
    Keywords=36028797018963968
    ID=1023
    Level=$C.Value__
}
```

Output Copy:



183.82.125.202:4499 - Remote Desktop Connection

Windows PowerShell

```
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand

PS C:\Users\devops34> [System.Diagnostics.Eventing.Reader.StandardEventLevel] | Get-Member -Static -MemberType Property


   TypeName: System.Diagnostics.Eventing.Reader.StandardEventLevel

Name          MemberType Definition
----          ---------- ----------
Critical      Property   static System.Diagnostics.Eventing.Reader.StandardEventLevel Critical {get;}
Error         Property   static System.Diagnostics.Eventing.Reader.StandardEventLevel Error {get;}
Informational Property   static System.Diagnostics.Eventing.Reader.StandardEventLevel Informational {get;}
LogAlways     Property   static System.Diagnostics.Eventing.Reader.StandardEventLevel LogAlways {get;}
Verbose       Property   static System.Diagnostics.Eventing.Reader.StandardEventLevel Verbose {get;}
Warning       Property   static System.Diagnostics.Eventing.Reader.StandardEventLevel Warning {get;}


PS C:\Users\devops34> Get-WinEvent -FilterHashtable @{
>>     LogName='Application'
>>     ProviderName='.NET Runtime'
>>     Keywords=36028797018963968
>>     ID=1023
>>     Level=2
>> }
>>
Get-WinEvent : No events were found that match the specified selection criteria.
At line:1 char:1
+ Get-WinEvent -FilterHashtable @{
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand

PS C:\Users\devops34> $C = [System.Diagnostics.Eventing.Reader.StandardEventLevel]::Informational
>> Get-WinEvent -FilterHashtable @{
>>     LogName='Application'
>>     ProviderName='.NET Runtime'
>>     Keywords=36028797018963968
>>     ID=1023
>>     Level=$C.Value__
>> }
>>
Get-WinEvent : No events were found that match the specified selection criteria.
At line:2 char:1
+ Get-WinEvent -FilterHashtable @{
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (:) [Get-WinEvent], Exception
    + FullyQualifiedErrorId : NoMatchingEventsFound,Microsoft.PowerShell.Commands.GetWinEventCommand
```