Troubleshoot Using Event Log Mining

Jeff Hicks
One tried-and-true troubleshooting technique is to search the Windows event logs for useful information. Windows 7 and Windows Server 2008 provide even better event log management tools provided you know how to effectively mine them for information. In this article I’ll focus on using the event log tools in Windows 7. As you’ll see you can not only find a wealth of information from a local system but querying remote computers is practically no different.

The Event Log Management Console
You can launch the event log viewer by selecting the shortcut under Administrative tools. Or you can save some mouse clicks and do what I do: Click Start – Run and enter `eventvwr`.

The first thing that I immediately like with the new viewer is the summary in the center top panel. I can see at a glance what types of events have been logged over specific time frames. I can see that this desktop has recorded 132 errors in the last 7 days. This information is summarized from all event logs. Click the + symbol to expand the error type.
You can resize the columns but now I can see what log the error is from, the event id and source. I can double click any entry to see all the instances. The result is shown in Figure 3.

![Figure 3: Error Instances](image)

The other way to view this information is to click Administrative Events under Custom Views in the left side panel.

![Figure 4: Administrative Events](image)

As you select different events, the details are shown in the bottom center panel. These summaries are great for looking at recent events. But you may need to dig deeper. Let’s first look at the traditional event logs you are most likely familiar with.
Expand the Windows Logs link in the left panel. Select any log. I’ll select Application as depicted in Figure 5.

![Figure 5: Application Event Log](image)

I’ll come back to filtering and custom view in a moment. What I want to show next are all the new event logs. Expand Microsoft under Applications and Services and you’ll discover a wealth of new logs. In Figure 6 I’ve selected a log for Bits-Client.

![Figure 6: Advanced Windows Logs](image)
Creating Filters

As you can see, there is a lot of information at your disposal that you can use for troubleshooting as well as proactive tasks. The challenge is finding relevant information amidst this huge haystack.

Let’s imagine I want to find all events in the System event log related to the Time Service because I’m troubleshooting a Kerberos problem and wrong clocks can be problematic. I want to see all non-information events within the last 30 days. After I’ve selected the System event log I click “Filter Current Log” from the right hand panel.

I check the boxes for Event Levels of Critical, Error and Warning. From the Logged drop down I select 30 days. Under Event Sources I check Time Service. You can build a filter for multiple sources if necessary but I’m going to stick to a single source. I end up with what you see in Figure 7.

Figure 7: Creating a Filter
When I click OK, the results are filtered.

![Filtered Results](image)

**Figure 8: Filtered Results**

Now it is much easier to find the information I need. This log will stay filtered until I click Clear Filter. But what if I want to reuse this filter? Easy. Turn it into a custom view.

**Creating Custom Views**

In the right hand panel select Save Filter to Custom View. You’ll be prompted for a name and description as I’ve done in Figure 9.

![Saving a Filter to a View](image)

**Figure 9: Saving a Filter to a View**
If you save a lot of view, you can create folders under Custom Views to keep everything organized.

Now that I have a custom view, I can clear the filter in the System event log. Whenever I want to see my Time Service events, I simply use the new view. This is not a copy of the event logs but a live filter. As new events occur that match the filter, they will be displayed in the view.

If you would like to use a custom view on another machine or share with other administrators, you can export the view to an XML file. On another machine, you would then import the custom view. You should see Export and Import commands in the right side panel.

**Querying Remote Machines**

You’re probably thinking how wonderful this is and how much information you can mine from this mother lode of event logs. But you don’t want to have to make a visit to an end user’s desktop every time there is a problem. You don’t have to. Right click where it says Event Viewer (Local) in the management console and select “Connect to another computer”

![Select Computer](image)

**Figure 10: Connecting to another computer**

Enter the computer name or browse Active Directory. I like that you can even specify alternate credentials. In Figure 10 I’m connecting to SERVER01.

Once connected, everything I did locally applies to the remote computer. In fact, even custom views like the Time Service view I created earlier.
You can create additional filters and views but when you save them, they are saved locally. This is all fine and dandy when looking at a single computer. But what if you are looking for information across multiple computers?

**Event Log Subscriptions**

The answer is to setup an event log subscription. Select Subscriptions from the bottom of the left hand panel. From the right hand side select Create a Subscription. You might get a message about starting the Windows Event Collector service. You’ll need it. In the dialog box, give your subscription a name and select the log you want to aggregate from the drop down list.

Next, define the computers to query. You can either configure the subscription to pull information (Collector Initiated), in which case you’ll need to manually add the computers you want to poll. Or you can use Group Policy (Source Initiated). For the sake of this article I’ll setup a collector initiated subscription.

The last step is to define a query filter. I recommend creating a custom view and choosing it after clicking Select Events. Figure 12 shows a new subscription.
Figure 12: Creating a new event log subscription

Depending on your configuration, you might have to set subscription credentials under Advanced. The WinRM and Windows Event Collector services must also be running on the remote computers.

Events that meet the filtering criteria on the remote computers are added to local event log specified in subscription. In my example I have the events written to the local system event log. This is handy when you want to aggregate results across multiple computers. Otherwise you might be better off send subscriptions to the Forwarded Events log. Figure 13 shows the results of another subscription for system events 7036.

Figure 13: Forwarded Events

It would be nice if we could create additional logs to handle multiple subscriptions. In lieu of that, simply create custom views.
Using Windows PowerShell

The event log viewer management console is indeed a powerful tool. But it is not the only tool at our disposal. We can also use Windows PowerShell to query local and remote event logs. PowerShell 2.0 has two cmdlets, depending on what we need to query.

Get-Eventlog

The Get-EventLog cmdlet is designed to work with the legacy event logs like System and Application. At its most basic, all you need to do is specify a log file name.

```
PS C:\> get-eventlog system
```

You’ll get event log objects for everything. I’m assuming you need something more specific. The cmdlet supports a number of parameters that you can mix and match as necessary.

Let’s say I want to find the 10 most recent errors and warning in the System event log. All it takes is a one line command.

```
PS C:\> get-eventlog system -newest 10 -entrytype error,warning
```

You can see the results in Figure 14.

![Figure 14: Get-Eventlog](image)

Remember we’re looking at event log object so we can slice and dice however we need to. If I want a closer look at these events, I might try an expression like this.

```
PS C:\> get-eventlog system -newest 10 -entrytype error,warning | Format-Table TimeWritten,Source,EventID,Message -Autosize -Wrap
```

Not only is it easy to retrieve information but you can analyze the logs to identify trends or get a handle on problems. Look at this one line expression.
I’m retrieving all errors and warnings from the system event log and piping them to the Group-Object which is organizing them into groups based on the Source property. Then I sort the group object by its Count property in descending order. Finally I select the first 10 which gives me Top 10 problem sources for this computer.

If you have a subscription writing to one of the legacy event logs, then that information will be included in any query or analysis.

But you aren’t limited to subscriptions. You can use Get-Eventlog to connect to one or more remote computers. Figure 16 depicts a PowerShell command querying SERVER01 for all Errors in the Application log that were logged after July 1, 2010.

Figure 15: Event log analysis

Figure 16: Querying a Remote Computer
If you query more than one machine you’ll need to tweak the output to include the Machinename. Figure 17 shows a query for the newest 5 DNS entries from 2 servers.

```powershell
PS C:\> get-eventlog -LogName "dns server" -entrytype error -newest 5 -ComputerName server01,coredc01 | Select Machinename,TimeWritten,EventID,Message | format-table -auto
```

**Figure 17: Querying multiple remote computers**

### Get-WinEvent

The other cmdlet you can use is Get-WinEvent. You need this cmdlet to query any of the new event logs. Although you can also specify a legacy event log as well. The event log object is just about the same. What is really different is how we filter. The –MaxEvents parameter is similar to the –Newest parameter from Get-Eventlog. But to truly filter we’ll need to use some XML.

Unless you are a native XML speaker, I find the best approach is to use the Event Viewer management console to create a filter as we did earlier, then click the XML tab.
You can select the query and press Ctrl-C to copy. To use in PowerShell you need to edit it as a one line expression, or paste it in as a here string as I did in Figure 19.

The filter has all the information so all I need to do is specify it with the –FilterXML parameter. Figure 20 shows a more complete application of this technique.
If you think you’ll want to re-use the filter in PowerShell, I recommend saving the filter to a file.

```
PS C:\> $filter | out-file c:\work\gpfilter.xml
```

Then the next time you need it, get the xml file’s contents and save to a variable.

```
PS C:\> $filter= get-content c:\work\gpfilter.xml
```

There are other ways to filter information but I don’t have space here to go into more detail. I encourage you to look at help examples for Get-WinEvent.

I hope you realize how much information you can mine from local and event logs, all from the comfort of your own cubicle, or corner office. The event viewer management console is the easiest tool. PowerShell requires a little more work and setting up complex monitoring and reporting definitely is not for the beginner or faint of heart. If you have very simple needs, you should be able to accomplish event log related tasks with the tools I’ve discussed. But some administrators and those with larger more complex environments may want to look to dedicated 3rd party applications.