Integrate Windows PowerShell

EventTracker Enterprise
Abstract

This guide provides instructions to enable Microsoft PowerShell logging for EventTracker.

Scope

The configurations detailed in this guide are consistent with EventTracker Enterprise version 7.X and later, and PowerShell 3.0 and later.

Audience

Administrators, who wish to monitor PowerShell command or script execution using EventTracker.
# Table of Contents

Abstract .................................................................................................................................................. 1  
Scope .................................................................................................................................................. 1  
Audience ............................................................................................................................................. 1  
What is PowerShell ................................................................................................................................ 4  
Enable PowerShell logging .................................................................................................................... 4  
Enable PowerShell logging ................................................................................................................ 4  
Configure Event viewer ...................................................................................................................... 8  
Configure EventTracker Event Filter .................................................................................................. 11  
  Event ID - 4103 .................................................................................................................................. 13  
  Event ID - 4100 .................................................................................................................................. 13  
  Event ID - 6 ...................................................................................................................................... 14  
  Event ID - 8 ...................................................................................................................................... 14  
  Event ID - 161 ................................................................................................................................... 15  
  Event ID - 169 ................................................................................................................................... 15  
EventTracker Knowledge Pack (KP) ..................................................................................................... 16  
  Reports ............................................................................................................................................. 17  
  Alerts ............................................................................................................................................... 17  
  Filter ............................................................................................................................................... 17  
Import Windows PowerShell Knowledge Pack into EventTracker ..................................................... 18  
  Import Parsing Rules ....................................................................................................................... 18  
  Import Alerts .................................................................................................................................... 19  
  Import Flex Reports .......................................................................................................................... 21  
  Import Filters .................................................................................................................................... 22  
Verify Windows PowerShell knowledge pack in EventTracker ........................................................... 23  
  Verify Parsing Rules ......................................................................................................................... 23  
  Verify Alerts .................................................................................................................................... 24  
  Verify Flex Reports ........................................................................................................................... 24  
  Verify Event Filters ............................................................................................................................ 25  
Create Dashboards in EventTracker .................................................................................................... 26
What is PowerShell?

**Windows PowerShell** is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and associated scripting language built on the .NET Framework. PowerShell comes in two versions: Console and Integrated Scripting Environment (ISE). PowerShell also features SSH like remote shell capability through **Windows Remote Management** (WinRM).

EventTracker amasses and examines logs generated by PowerShell to help an administration to monitor remote session’s establishment and execution of rogue scripts or commands.

Enable PowerShell logging

1. Open **Group Policy Editor** in Windows.

![Figure 1](image-url)
2. Navigate to **Computer Configuration>Administrative Templates>Windows Components>Windows PowerShell**.

For **PowerShell 3.0 and 4.0**, settings are shown as follows:

![Figure 2](image)

For **PowerShell 5.0**, settings are shown as follows:
3. Click **Turn on Module Logging** setting.
3. Select Enabled.
4. In **Module Names** section, select **Show** to enable logging for selected modules.
5. Configure **Module Names** as shown below.
6. Select **OK** and **Apply** to save the changes.

**NOTE-**
- It is not advised to enable script logging options as it might result into high log volume.
- Select value as ‘*’ in Module Names pane to enable logging for all available modules.

**Configure Event viewer**

1. Open **Event Viewer** in Windows.
2. Right-click **Saved Logs** and select **Open Saved Log...** option.
3. Navigate to `C:\Windows\System32\winevt\Logs` and select following logs.
   a. Microsoft-Windows-PowerShell%40Operational.evtx
   b. Microsoft-Windows-WinRM%40Operational.evtx
4. For both log types, compose **Open Saved Log** dialog settings per convenience. 
5. Select **OK** to confirm.
6. PowerShell and WinRM logs can be observed in the right pane.

**Configure EventTracker Event Filter**

1. Launch **EventTracker Control Panel**.

![EventTracker Control Panel](image)

**Figure 9**

2. Double click **EventTracker Agent Configuration**.
EventTracker: Integrate Windows PowerShell

3. Navigate to Event Filters > Filter Exception.

4. Click New and compose Edit Event Details. Configure settings for relevant events as shown below.
Event ID - **4103**

![Event ID 4103](image)

**Figure 12**

Event ID - **4100**

![Event ID 4100](image)

**Figure 13**
Event ID - 6

Figure 14

Event ID - 8

Figure 15
Event ID - 161

Event ID - 169
5. Review the changes and click **OK** to confirm.

![EventTracker Agent Configuration](image)

**Figure 18**

6. Click **Close** and **Save** to apply the changes.

**EventTracker Knowledge Pack (KP)**

Once logs are received in to EventTracker, Alerts, Reports and Dashboards can be configured into EventTracker. The following Knowledge Packs are available in EventTracker to support Windows PowerShell monitoring.
Reports

1. **Windows PowerShell-Command execution details**– This report provides information related to command execution on PowerShell which includes User Name, Host Type, Command Executed and Command Parameters fields.

2. **Windows PowerShell-Script execution details**– This report provides information related to command execution through script on PowerShell which includes User Name, Host Type, Script Path, Command Executed and Command Parameters fields.

3. **Windows PowerShell-Command execution error details**– This report provides information related to command execution errors by script or CLI on PowerShell which includes User Name, Host Type, Script Path, Command Executed and Command Parameters fields.

4. **Windows PowerShell-Remote session creation details**– This report provides information related to PowerShell remote session initialization which includes Computer, User Name and Remote Host fields.

5. **Windows PowerShell-Remote session authentication success details**– This report provides information related to successful PowerShell remote session authentication which includes Computer, Remote User Name and Authentication Method fields.

6. **Windows PowerShell-Remote session authentication failure details**– This report provides information related to unsuccessful PowerShell remote session authentication which includes Computer, Event User and Reason fields.

Alerts

1. **Windows PowerShell-Command execution failed**– This alert is generated when command execution on PowerShell fails.

2. **Windows PowerShell-Remote session initiated**– This alert is generated when PowerShell remote session is initialized.

3. **Windows PowerShell-Remote session user authentication failed**– This alert is generated when PowerShell user authentication fails.

Filter

1. **Windows PowerShell-EventTracker script filter**– This filter excludes events generated by EventTracker scripts.
Import Windows PowerShell Knowledge Pack into EventTracker

1. Launch EventTracker Control Panel.
2. Double click Import Export Utility, and then click the Import tab.

![EventTracker Control Panel](image)

**Figure 19**

Import Parsing Rules, Alerts, Reports and Filter as given below.

**Import Parsing Rules**

1. Click Token Value option, and then click the browse button.
2. Locate All Windows PowerShell group of tokens.istoken file, and then click the Open button.
3. To import token value, click the **Import** button. EventTracker displays success message.

4. Click **OK**, and then click the **Close** button.

**Import Alerts**

1. Click **Alerts** option, and then click the **browse** button.
2. Locate **All Windows PowerShell group alerts.isalt** file, and then click the **Open** button.

![Figure 22](image)

3. To import alerts, click the **Import** button.

EventTracker displays success message.

![Figure 23](image)

4. Click **OK**, and then click the **Close** button.
Import Flex Reports

1. Click **Reports** option, and then click the ‘*browse*' button.
2. Locate **All Windows PowerShell group reports.issch** file, and then click the **Open** button.

![Figure 24](image)

3. To import scheduled reports, click the **Import** button.
   
   EventTracker displays success message.

![Figure 25](image)

4. Click **OK**, and then click the **Close** button.
Import Filters

1. Click **Reports** option, and then click the ‘**browse**’ button.

2. Locate **Windows PowerShell Filter.isfil** file, and then click the **Open** button.

3. To import filters, click the **Import** button.

   EventTracker displays success message.

4. Click **OK**, and then click the **Close** button.
Verify Windows PowerShell knowledge pack in EventTracker

Verify Parsing Rules

1. Logon to EventTracker Enterprise.

2. Click the Admin menu, and then click Parsing Rule.

3. In Token Value Group Tree to view imported token values, scroll down and click Windows PowerShell group folder.

Token values are displayed in the token value pane.
Verify Alerts

1. Logon to EventTracker Enterprise.
2. Click the Admin menu, and select Alerts.
3. In Search field, type 'powershell', and then click the button.

Alert Management page will display all the imported PowerShell alerts.

4. To activate the imported alerts, select the respective checkbox in the Active column.

EventTracker displays message box.

5. Click OK, and then click the Activate Now button.

NOTE: Please specify appropriate systems in alert configuration for better performance.

Verify Flex Reports

1. Logon to EventTracker Enterprise.
2. Click the **Reports** menu and select **Configuration**.
3. Select **Defined** in report type.

   Imported reports are displayed in the Reports Configuration pane.

---

**Figure 31**

**Verify Event Filters**

1. Logon to **EventTracker Enterprise**.
2. Click the **Admin** menu, and select **Event Filters**.
3. In **Search** field, type ‘powershell’, and then click the **Search** button.

   Event Filters page will display all the imported PowerShell filter.
EventTracker: Integrate Windows PowerShell

EVENT FILTERS

4. To activate the imported alerts, select the respective checkbox in the **Active** column. EventTracker displays message box.

![Message Box]

**Figure 32**

5. Click **OK**, and then click the **Activate Now** button.

**NOTE:** Please specify appropriate **systems** in **filter wizard** for better performance.

Create Dashboards in EventTracker

Schedule Reports

1. Open **EventTracker** in browser and logon.
2. Navigate to **Reports>Configuration**.

3. Select ‘**Windows PowerShell**’ in report groups. Check **defined** dialog box.

4. Click on ‘**schedule**’ to plan a report for later execution.
5. Choose appropriate time for report execution and in Step 8 check Persist data in Eventvault explorer box.
6. Check column names to persist using PERSIST checkboxes beside them. Choose suitable Retention period.
7. Proceed to next step and click Schedule button.
8. Wait for scheduled time or generate report manually.

Create Dashlets

1. EventTracker 8 is required to configure flex dashboard.
2. Open EventTracker in browser and logon.
3. Navigate to Dashboard > Flex.
   Flex Dashboard pane is shown.

4. Click to add a new dashboard.
   Flex Dashboard configuration pane is shown.
5. Fill fitting title and description and click **Save** button.

6. Click 🗯 to configure a new flex dashlet. Widget configuration pane is shown.

**WIDGET CONFIGURATION**

![Widget Configuration](image)

7. Locate earlier scheduled report in **Data Source** dropdown.

8. Select **Chart Type** from dropdown.

9. Select extent of data to be displayed in **Duration** dropdown.

10. Select computation type in **Value Field Setting** dropdown.

11. Select evaluation duration in **As Of** dropdown.

12. Select comparable values in **X Axis** with suitable label.

13. Select numeric values in **Y Axis** with suitable label.

14. Select comparable sequence in **Legend**.

15. Click **Test** button to evaluate. Evaluated chart is shown.
16. If satisfied, click **Configure** button.

17. Click ‘customize’ to locate and choose created dashlet.

18. Click to add dashlet to earlier created dashboard.
Sample Dashboards

- Windows PowerShell-Remote session created in last 24 hrs

Figure 44
## Sample Reports

- Windows PowerShell-Command execution error details

<table>
<thead>
<tr>
<th>LogTime</th>
<th>Computer</th>
<th>User Name</th>
<th>Host Type</th>
<th>ScriptPath</th>
<th>Command Executed</th>
<th>Error Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>02/29/2015 02:47:59 PM</td>
<td>MGR23</td>
<td>ADMIN\Jim</td>
<td>Windows PowerShell ISE Host</td>
<td>C:\Downloads\Web\hCheck.ps1</td>
<td>Out-File Command</td>
<td>Error Message = Could not find a part of the path ‘C:\Scripts\Test.htm’. Fully Qualified Error ID = FileOpenFailure,Microsoft.PowerShell.Commands:OutFile Command</td>
</tr>
<tr>
<td>02/29/2015 02:48:08 PM</td>
<td>MGR23</td>
<td>ADMIN\Jim</td>
<td>Windows PowerShell ISE Host</td>
<td>C:\Downloads\Web\hCheck.aspx</td>
<td>Out-File Command</td>
<td>Error Message = Could not find a part of the path ‘C:\Scripts\Test.htm’. Fully Qualified Error ID = FileOpenFailure,Microsoft.PowerShell.Commands:OutFile Command</td>
</tr>
<tr>
<td>02/29/2015 02:53:00 PM</td>
<td>HR43</td>
<td>ADMIN\Jim</td>
<td>RemoteHost</td>
<td>Invoke-WebRequest</td>
<td>Network Access Message: The page cannot be displayed Explanation: There is a problem with the page you are trying to reach and it cannot be displayed.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 45
- Windows PowerShell-Script execution details

<table>
<thead>
<tr>
<th>LogTime</th>
<th>Computer</th>
<th>User Name</th>
<th>Host Type</th>
<th>Script Path</th>
<th>Command Executed</th>
<th>Command Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>02/29/2016 02:41:45 PM</td>
<td>MCR44</td>
<td>ADMINJim</td>
<td>RemoteHost</td>
<td>C:\Users\Jim\PowerShell.ps1</td>
<td>Get-Content</td>
<td>ParameterBinding(New-Object); name=&quot;Properties&quot;; value=&quot;System.Collections.Generic.Dictionary<code>2</code>&lt;String,String&gt;`</td>
</tr>
<tr>
<td>02/29/2016 02:47:28 PM</td>
<td>HR23</td>
<td>ADMINJim</td>
<td>WindowsPowerShellHost</td>
<td>C:\HealthCheck.ps1</td>
<td>Get-Content</td>
<td>ParameterBinding(Get-Content); name=&quot;Path&quot;; value=&quot;C:\scripts\test.html&quot;</td>
</tr>
<tr>
<td>02/29/2016 02:47:28 PM</td>
<td>HR23</td>
<td>ADMINJim</td>
<td>WindowsPowerShellHost</td>
<td>C:\HealthCheck.ps1</td>
<td>Get-Content</td>
<td>ParameterBinding(Get-Content); name=&quot;Path&quot;; value=&quot;C:\scripts\test.html&quot;</td>
</tr>
<tr>
<td>02/29/2016 02:47:55 PM</td>
<td>HR23</td>
<td>ADMINJim</td>
<td>WindowsPowerShellHost</td>
<td>C:\HealthCheck.ps1</td>
<td>Out-File</td>
<td>ParameterBinding(Out-File); name=&quot;FilePath&quot;; value=&quot;C:\scripts\test.html&quot;</td>
</tr>
<tr>
<td>02/29/2016 02:47:59 PM</td>
<td>PRO22</td>
<td>MGMTWalths</td>
<td>WindowsPowerShellHost</td>
<td>C:\scripts\UpdateCheck.ps1</td>
<td>Invoke-Expression</td>
<td>ParameterBinding(Invoke-Expression); name=&quot;Command&quot;; value=&quot;Invoke-Expression&quot;</td>
</tr>
</tbody>
</table>

Figure 46

-X-