

Sample MP

Event Based Alerting Rules

Steve Wilson

3/26/2007

This guide details the contents of the sample event based alert rule management pack (AuthorMPs.Demo.EventAlertingRules).

Contents

Introduction	3
Discovery	3
Views	3
Alert Module	4
String Resources	4
Alert Context and Events	4
Rules	5
AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent401.SuppressOnWorkflow	5
AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent402.SuppressOnParam1	5
AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent403.NoSuppression	6
AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent404.StoreEvents	6
AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent405.UseTargetProperties	6
AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnMultipleEvents.UseDataProperties	7
AuthorMPs.Demo.EventAlertingRules.ApplicationXComponent.AlertOnEvent420	8

Introduction

This management pack shows you how to create alerts from events in the Windows event log. This can be applied to any event based data source. For the configuration of other data sources you should look at the AuthorMPs.Demo.EventCollectionRules sample MP.

I cover the following concepts in this sample:

- Alert suppression
- Using String Resources
- Substituting \$Target properties into an alert description
- Substituting \$Data properties into an alert description

Discovery

The MP defines two classes:

- AuthorMPs Demo - Application X
- AuthorMPs Demo - Application X Component

To discover instances of these classes, create a directory called C:\AuthorMPs (this will create an instance of Application X). Now create some files in that directory (this will create one instance of Application X Component for each file).

Views

There are four views in this management pack:

- Application X State - a state view showing all instances of Application X
- Application X Alerts - an alert view showing all alerts for all instances of Application X
- Application X Component State - a state view showing all instances of Application X Component
- Application X Component Alerts- an alert view showing all alerts for all instances of Application X Component

You should note that the alert view for Application X will also show alerts for Application X components due to the hosting relationship that is defined.

As well as using the views I have provided, you can pivot from a state view to an alert view. For example, to see alerts for a specific Application X, find the instance you want in the state view then right click and select Open, Alert View. This will show you alerts just for this specific instance.

If you want to check which rule was responsible for an alert you can pivot to the rule from the alert details view.

Alert Module

The module that is used to generate an alert is the System.Health.GenerateAlert module in the System.Health.Library management pack. You will see all the rules in this management pack use this module. This module takes configuration to specify:

- Alert Priority
- Alert Severity
- Alert Message String Resource
- Alert Parameters
- Suppression

The use of different configuration is discussed in each rule later in this guide.

String Resources

String resources are used to provide a localized name and description for the alert. When you use the alert module you specify an alert message ID e.g.

```
<AlertMessageId>$MPElement [Name="AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent401.SuppressOnWorkflow.AlertMessage"] $</AlertMessageId>
```

This points to a string resource later in the MP (in the presentation section). This string resource is a simple XML element e.g.

```
<StringResource  
ID="AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent401.SuppressOn  
Workflow.AlertMessage"/>
```

This string resource then has a display name and description assigned in the language pack. This means that different language strings can be set for the alert name and description if required. It is also possible to substitute data properties and target properties into the alert description using alert parameters and substitution placeholders such as {0} in the alert description. I cover this in the specify rule details below.

Alert Context and Events

By default, the event that triggered the alert is not stored in the database. You have to explicitly add the modules to do this if you want. I do this in one rule below. However, you should not that although the raw event is not stored by default, the context of an alert is always stored. Therefore when an event triggers an alert if you look in the alert context you will see the event that first triggered the alert. Therefore you should think about whether you need to collect the raw events as well or whether alert context will suit you. You can see the alert context from the alert property page.

Rules

Here is a summary of the rules in the management pack and how you can see them working:

[AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent401.SuppressOnWorkflow](#)

Target: Application X

This rule will create an alert from a simple Windows event that has a publisher of EventCreate and an event ID of 401. To generate the alert use the EventCreate utility that is part of Windows Server 2003 or later. At a command prompt on a server that you have a discovered instance of Application X type:

```
EventCreate /ID 401 /T information /D Test
```

You will see an alert created against your Application X instance. This rule is configured to suppress alerts based on workflow. That means you will never have more than one alert active for a given instance of Application X. You can see this by logging more 401 events. You will not see additional alerts. If you bring up the alert properties you will see the repeat count increment.

I suppress on workflow by specifying the following suppression settings:

```
<Suppression>  
    <SuppressionValue/>  
</Suppression>
```

You should also see that the alert name and description you see in the Operations Console are taken from the string resource I specified in the alert module for this rule.

[AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent402.SuppressOnParam1](#)

Target: Application X

This rule is very similar to the 401 alert rule. The only difference is the alert suppression settings. This time I suppress on the value in parameter 1 of the event. I will get one alert for each different value of parameter 1. Type the following at a command prompt:

```
EventCreate /ID 402 /T information /D Test1
```

```
EventCreate /ID 402 /T information /D Test2
```

Now generate a few more of each event (some with Test1 in the description and some with Test2 in). You should see two alerts. The alerts are suppressed based on parameter 1. I suppress like this using the following settings:

```
<Suppression>  
    <SuppressionValue>$Data/Params/Param[1] $</SuppressionValue>  
</Suppression>
```

I could specify multiple suppression values if I needed to e.g.

```
<Suppression>  
  <SuppressionValue>$Data/Params/Param[1] $</SuppressionValue>  
  <SuppressionValue>$Data/Params/Param[2] $</SuppressionValue>  
</Suppression>
```

AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent403.NoSuppression

Target: Application X

This rule is again a slight variation on alert suppression settings. This time I have no alert suppression. This means that I will get a separate alert for every event I match on. No alerts are suppressed. You can see this by running the following:

```
EventCreate /ID 403 /T information /D Test
```

Now run the same command a few times – you will see an alert for each event. I specify no suppression by using the following configuration:

```
<Suppression/>
```

AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent404.StoreEvents

Target: Application X

So far, all the rules have just created an alert. The event is available as alert context but the raw events are not stored in the database. You can verify this by selecting an instance of Application X and pivoting to an event view.

This rule is almost the same as the 401 alert rule. However, I have added two additional write actions to write the event to the Operations Database and the Data Warehouse. You can generate the alert with:

```
EventCreate /ID 404 /T information /D Test
```

You should see the alert as before. Now you can also pivot to an event view from your Application X instance in the state view and you should see an event has been stored. Generate a few more event 404. No more alerts are created but each event is stored.

AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent405.UseTargetProperties

Target: Application X

So far all the alerts we have seen have used a static name and description. This rule shows how to use target properties in the alert description while still maintaining a localizable description. You can generate the event with:

```
EventCreate /ID 405 /T information /D Test
```

You will see that the description contains the computer name as well as the path we discovered for Application X. You will get an alert description like this:

“Alert from Application X on computer stwilson01.redmond.corp.microsoft.com. Application X is installed in directory: C:\AuthorMPs”

I do this by adding alert parameters to the alert module configuration:

```
<AlertParameters>
  <AlertParameter1>$Target/Property[Type="AuthorMPs.Demo.EventAlertingRules.ApplicationX"]/Path$</AlertParameter1>
  <AlertParameter2>$Target/Host/Property[Type="Windows!Microsoft.Windows.Computer"]/NetworkName$</AlertParameter2>
</AlertParameters>
```

Notice I specify two properties, one from my Application X instance and one from the computer that hosts this instance.

I then substitute in these parameters in my description for the string resource:

```
<DisplayString
ElementID="AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnEvent405.UseTargetProperties.AlertMessage">
  <Name>Alert on 405</Name>
  <Description>Alert from Application X on computer {1}. Application X is installed in directory: {0}</Description>
</DisplayString>
```

I use {0} to refer to Parameter1, {1} to refer to Parameter2 etc.

AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnMultipleEvents.UseDataProperties

Target: Application X

The previous rule showed you how to substitute target properties into an alert description. This rule shows how to substitute data properties. I use a regex in my expression to look for events 410, 411 or 412. I then want to include this event ID in the description. I specify the alert parameters as:

```
<AlertParameters>
  <AlertParameter1>$Target/Host/Property[Type="Windows!Microsoft.Windows.Computer"]/NetworkName$</AlertParameter1>
  <AlertParameter2>$Data/EventDisplayNumber$</AlertParameter2>
</AlertParameters>
```

Parameter 2 contains a pointer to the Event ID. In my string resource display string I specify:

```
<DisplayString
ElementID="AuthorMPs.Demo.EventAlertingRules.ApplicationX.AlertOnMultipleEvents.UseDataProperties.AlertMessage">
  <Name>Alert on 410, 411 or 412</Name>
  <Description>Alert from Application X on computer {0}. The event that triggered the alert was ID: {1}</Description>
</DisplayString>
```

This means that the alert description will come through as something like this:

“Alert from Application X on computer stwilson01.redmond.corp.microsoft.com. The event that triggered the alert was ID: 410”

To generate the alert type:

```
EventCreate /ID 410 /T information /D Test
```

You can replace the 410 with 411 or 412. I suppress on the event ID so you will only see one alert for each event ID no matter how many times you log it on one computer. Notice that the correct ID is substituted into the alert description.

AuthorMPs.Demo.EventAlertingRules.ApplicationXComponent.AlertOnEvent420

Target: Application X Component

This rule shows an alert rule for a multi instance class just to show you what suppress on workflow means. In the 401 alert rule (first rule in this MP we talked about) I set it to suppress on workflow. Since my management pack is written so that there is only ever one instance of Application X on one computer you would only ever get one active alert per computer at a time for this event.

I can have multiple instances of Application X Component. My rule is targeted at the Application X Component class. I will have one workflow for every instance of this class. If I have 4 instances I will have 4 workflows.

I have written the rule to suppress use event parameter 1 as the instance identifier:

```
<Expression>
  <SimpleExpression>
    <ValueExpression>
      <XPathQuery Type="String">Params/Param[1]</XPathQuery>
    </ValueExpression>
    <Operator>Equal</Operator>
    <ValueExpression>
      <Value
Type="String">$Target/Property [Type="AuthorMPs.Demo.EventAlertingRules.Applic
ationXComponent"]/ID$</Value>
      </ValueExpression>
    </SimpleExpression>
  </Expression>
```

I have also specified the alert should be suppressed on workflow

```
<Suppression>
  <SuppressionValue/>
</Suppression>
```

To show this working, suppose I have 2 instances of Application X Component with ID of “Comp1” and “Comp2”. Type the following two commands at a command prompt:

```
EventCreate /ID 420 /T information /D Comp1
```

```
EventCreate /ID 420 /T information /D Comp2
```

You will see an alert created for each component. If you ran these command more times, no more alerts would be generated. The alert is suppressed on the workflow.