Trap Configuration How-To

In addition to internally generated events, OpenNMS can also receive SNMP traps via the trapd process. These are used to match SNMP traps in `eventconf.xml` by using the `<mask>` tag.

Contents

Creating event definitions from trap definitions in SNMP MIBs

OpenNMS 1.12 and later include a GUI for parsing SNMP MIB definitions and creating OpenNMS event definitions from any *TRAP-TYPE* and/or *NOTIFICATION-TYPE* macros they contain. This interface is accessible from the web UI under **Admin** / **SNMP MIB Compiler**.

Two command-line utilities, *mib2events* and *mib2opennms*, are also available to do the heavy lifting of making event definitions from MIB definitions. The GUI editor does at least as good a job as *mib2events*. Only masochists should be using *mib2opennms* nowadays; it tends to produce more bus errors than event definitions.

SNMP trap event definitions

Event configurations for traps use the `<mask>` tag to match incoming SNMP trap values.

```
<event>
  <mask>
    <maskelement>
      <mename>id</mename>
      <mevalue>.1.3.6.1.4.1.9.9.70.2</mevalue>
    </maskelement>
    <maskelement>
      <mename>generic</mename>
      <mevalue>6</mevalue>
    </maskelement>
    <maskelement>
      <mename>specific</mename>
      <mevalue>17</mevalue>
    </maskelement>
  </mask>
  <uei>http://uei.opennms.org/vendor/Cisco/traps/ciscoC3800SysAggregateStatusChange</uei>
  <event-label>CISCO-C3800-MIB defined trap event: ciscoC3800SysAggregateStatusChange</event-label>
  <descr>&#38lt;p&#38gt;Notification that the aggregate status of a node
         has changed.&#38lt;/p&#38gt;&#38lt;table&#38gt;
         &#38lt;tr&#38gt;&#38lt;td&#38gt;&#38lt;b&#38gt;
         c3800SysNextTrapSeqNum&#38lt;/b&#38gt;</td&#38gt;&#38lt;td&#38gt;%parm[#1]%
```

```
        &#38lt;/td&#38gt;&#38lt;td&#38gt;&#38lt;p;&#38gt;</p&#38gt;&#38lt;/td;&#38gt;&#38lt;
/tr&#38gt;&#38lt;tr&#38gt;&#38lt;td&#38gt;&#38lt;b&#38gt;
        sysName&#38lt;/b&#38gt;&#38lt;/td&#38gt;&#38lt;td&#38gt;%parm[#2]%
        &#38lt;/td&#38gt;&#38lt;td&#38gt;&#38lt;p;&#38gt;&#38lt;/p&#38gt;&#38lt;/td;&#38gt;&#38lt;
/tr&#38gt;&#38lt;tr&#38gt;&#38lt;td&#38gt;&#38lt;b&#38gt;
        c3800SysTrapSeverity&#38lt;/b&#38gt;&#38lt;/td&#38gt;&#38lt;td&#38gt;%parm[#3]%
        &#38lt;/td&#38gt;&#38lt;td&#38gt;&#38lt;p;&#38gt;
        clear(1) minor(2) major(3)&#38lt;/p&#38gt;
        &#38lt;/td;&#38gt;&#38lt;/tr&#38gt;&#38lt;tr&#38gt;&#38lt;td&#38gt;&#38lt;b&#38gt;
        c3800SysAggregateStatus&#38lt;/b&#38gt;&#38lt;/td&#38gt;&#38lt;td&#38gt;%parm[#4]%
        &#38lt;/td&#38gt;&#38lt;td&#38gt;&#38lt;p;&#38gt;
        clear(1) minor(2) major(3)</p>
        &#38lt;/td;&#38gt;&#38lt;/tr&#38gt;&#38lt;/table&#38gt;
  </descr>
  <logmsg dest='logndisplay'><p>Cisco Event: C3900: Node Status has changed.</p></logmsg>
  <severity>Indeterminate</severity>
</event>
```

This is a Cisco Systems event for their C3800 device. Parts of it look similar to the internally generated events, with the main difference being the `<mask>` block. This block consists of `<maskelement>` tags, and the event will only match if all the defined tags are met.

This particular event will match an SNMP trap whose enterprise OID (id) is equal to ".1.3.6.1.4.1.9.9.70.2", its generic trap value is enterprise specific (6) and its specific trap value is 17.

The possible `<mename>` values are:

- uei
- source
- host
- snmphost
- nodeid
- interface
- service
- id
- specific
- generic
- community

It is possible to use the "%" symbol to indicate a wildcard in the mask values. For example, to match all Cisco events, I could use:

```
<mask>
  <maskelement>
    <mename>id</mename>
    <mevalue>.1.3.6.1.4.1.9.%</mevalue>
  </maskelement>
</mask>
```

Note: The order in which events are listed in the `eventconf.xml` file is extremely important. The search will stop with the first event definition that matches the given event. Thus if the above code with the wildcard was listed before the more specific `ciscoC3800SysAggregateStatusChange` event, the latter event would never be generated. Also note that the wildcard is simply a substring match. If an event was generated from a Cisco device with the Enterprise OID of ".1.3.6.1.4.1.9" it would *not* match this event, as there is no trailing ".". If the trailing "." is left off, care must be taken so that a trap with an OID of ".1.3.6.1.4.1.99" is listed before the ".1.3.6.1.4.1.9%" event or else it will match the more generic event.

Using the parm Replacement Token with trap events

Some events, especially SNMP traps, have additional information sent with them called "variable bindings" or "varbinds" for short. In the ciscoC3800SysAggregateStatusChange event listed above, there are four of them, and they can be accessed using the parm replacement token. Each parameter consists of a name and a value.

%parm[all]%
: Will return a space-separated list of all parameter values in the form parmName1="parmValue1" parmName2="parmValue2" etc.

%parm[values-all]%
: Will return a space-separated list of all parameter values associated with the event.

%parm[names-all]%
: Will return a space-separated list of all parameter names associated with the event.

%parm[<name>]%
: Will return the value of the parameter named <name> if it exists.

%parm[##]%
: Will return the total number of parameters.

%parm[#<num>]%
: Will return the value of parameter number <num>.

%parm[name-#<num>]%
: Will return the name of parameter number <num>.

For example, the ciscoC3800SysAggregateStatusChange event description lists out each of the parameters. Thus the second paramater, the sysName is printed out using %parm[#2]%.

Filtering on varbinds (OpenNMS 1.10 and beyond)

Let's take a look at the example ciscoC3800SysAggregateStatusChange event once more. What should its severity be? Since the event is generated whenever the status changes, you don't know if the change is "bad" (from operational to non-operational) or "good" (the non-operational status is cleared). That information is contained within the parameters that are passed with the event, particularly parameter #3, the trap severity.

In version 1.1.0, the ability to filter on variable bindings was added. This is done in the <mask> block. To re-write the above event:

```
<mask>
  <maskelement>
    <mename>id</mename>
    <mevalue>.1.3.6.1.4.1.9.9.70.2</mevalue>
  </maskelement>
  <maskelement>
    <mename>generic</mename>
    <mevalue>6</mevalue>
  </maskelement>
  <maskelement>
    <mename>specific</mename>
    <mevalue>17</mevalue>
  </maskelement>
  <varbind>
    <vbnumber>3</vbnumber>
    <vbvalue>3</vbvalue>
  </varbind>
</mask>
```

With a "status change" event, you will likely want to create separate events for each status value. To do this, copy the

event definition once for each status value, add the `<varbind>` mask, and then change the:

- uei
- description
- severity
- logmsg

to be appropriate for the varbind value. In the Cisco example, adding a `<mask>` with a `<varbind>` tag will match on the same id, generic and specific values, but also will require that the third parameter is equal to "3" (indicating a Cisco determined trap severity of "major"). Thus you could change the description and/or severity to match the event.

It is also possible to match more than one varbind, and more than one value per varbind:

```
<varbind>
  <vbnumber>3</vbnumber>
  <vbvalue>2</vbvalue>
  <vbvalue>3</vbvalue>
</varbind>
<varbind>
  <vbnumber>4</vbnumber>
  <vbvalue>2</vbvalue>
  <vbvalue>3</vbvalue>
</varbind>
```

The above code snippet will match if the third parameter has a value of "2" or "3" *and* the fourth parameter has a value of "2" or "3".

This feature was updated before the 1.6.0 release to allow a regular expression match on the varbind value. Just specify the expression prefixed with a with a "~".

```
<varbind>
  <vbnumber>1</vbnumber>
  <vbvalue>~[Dd]own</vbvalue>
</varbind>
```

This will match a varbind 1 containing the word "Down" or "down" anywhere within its value. You can also do quick prefix matches with the '%' in a varbind value:

```
<varbind>
  <vbnumber>1</vbnumber>
  <vbvalue>Error:%</vbvalue>
</varbind>
```

This will match varbind 1 with any string beginning with "Error:". **Again, note that the order in which events are listed is very important. Put the most specific events first.**

Decoding varbinds (OpenNMS 1.7.0 and beyond)

A lot of MIBs define specific variables to code the value of some OID. As an example the snmp agent returns a numerical value for the ifAdminStatus and ifOperStatus: 1 means Up and 2 means Down.

Because of the fact that OpenNMS does not have a MibParser, we usually put this map (between numerical encoded value and their meaning) into the event Description.

Configuring the Event properly now are able to decode the numerical value sent into trap varbinds to the corresponding string value into the <logmsg>.

Let consider a Cisco HSRP status changes trap (OID .1.3.6.1.4.1.9.9.106.2 generic 6 and specific 1), this trap correspond to uei.opennms.org/vendor/Cisco/traps/cHsrpStateChange event.

The trap contains the following varbind: cHsrpGrpStandbyState whose possible values are from 1 to 6 and whose meaning is:

```
initial(1) learn(2) listen(3) speak(4) standby(5) active(6).
```

We want to display the literal meaning of the HSRP status inside the logmsg. Here is the original event definition:

```
<event>
 <mask>
  <maskelement>
   <mename>id</mename>
   <mevalue>.1.3.6.1.4.1.9.9.106.2</mevalue>
  </maskelement>
  <maskelement>
   <mename>generic</mename>
   <mevalue>6</mevalue>
  </maskelement>
  <maskelement>
   <mename>specific</mename>
   <mevalue>1</mevalue>
  </maskelement>
 </mask>
 <uei>uei.opennms.org/vendor/Cisco/traps/cHsrpStateChange</uei>
 <event-label>CISCO-HSRP-MIB defined trap event: cHsrpStateChange</event-label>
 <descr><p>A cHsrpStateChange notification is sent when a
 cHsrpGrpStandbyState transitions to either active or
 standby state, or leaves active or standby state. There
 will be only one notification issued when the state change
 is from standby to active and vice versa.</p><table>
 <tr><td><b>
 cHsrpGrpStandbyState</b></td><td>%parm[#1]%
 </td><td><p;>
 initial(1) learn(2) listen(3) speak(4) standby(5) active(6)</p>
 </td;></tr></table>
 </descr>
 <logmsg dest='logndisplay'><p>Cisco Event: HSRP State Change.</p></logmsg>
 <severity>Minor</severity>
 </event>
```

This is how we would change the event definition so that the status is decoded inside the logmsg:

```
<event>
 <mask>
  <maskelement>
   <mename>id</mename>
   <mevalue>.1.3.6.1.4.1.9.9.106.2</mevalue>
  </maskelement>
  <maskelement>
   <mename>generic</mename>
   <mevalue>6</mevalue>
  </maskelement>
  <maskelement>
   <mename>specific</mename>
   <mevalue>1</mevalue>
  </maskelement>
 </mask>
 <uei>uei.opennms.org/vendor/Cisco/traps/cHsrpStateChange</uei>
 <event-label>CISCO-HSRP-MIB defined trap event: cHsrpStateChange</event-label>
 <descr><p>A cHsrpStateChange notification is sent when a
```

```
cHsrpGrpStandbyState transitions to either active or
standby state, or leaves active or standby state. There
will be only one notification issued when the state change
is from standby to active and vice versa.</p><table>
<tr><td><b>
cHsrpGrpStandbyState</b></td><td>%parm[#1]%
</td><td><p;>
initial(1) learn(2) listen(3) speak(4) standby(5) active(6)</p>
</td;></tr></table>
</descr>
<logmsg dest='logndisplay'><p>Cisco Event: HSRP State Change to %parm[#1]%.</p></logmsg>
<severity>Minor</severity>
<varbindsdecode>
<parmid>parm[#1]</parmid>
<decode varbindvalue="1" varbinddecodedstring="initial"/>
<decode varbindvalue="2" varbinddecodedstring="learn"/>
<decode varbindvalue="3" varbinddecodedstring="listen"/>
<decode varbindvalue="4" varbinddecodedstring="speak"/>
<decode varbindvalue="5" varbinddecodedstring="standby"/>
<decode varbindvalue="6" varbinddecodedstring="active"/>
</varbindsdecode>
</event>
```

Here the parm[#1] (So the first varbind into the trap is translated using the decode map. If the value of the first OID in this trap is 6 the the log message will be:

```
<p>Cisco Event: HSRP State Change to active.</p>
```

1. REDIRECT Target page name
