End To End Monitoring

Hyperic-HQ® Integration With OpenNMS®

David Hustace
The OpenNMS Group, Inc.

Executive Summary

There are two Java based open source network management software projects available today from SourceForge: Hyperic HQ and OpenNMS. Both applications are truly developed for scalability and integration in the enterprise and each are written in Java and are professionally developed and commercially supported.

Even though at first glance it would appear each of these projects serve the same purpose, as it turns out, these projects are actually more complementary than competitive with only a fractional amount of overlap. Hyperic HQ is used by many organizations to support the monitoring of applications and their platforms leveraging it's strong agent based technology. OpenNMS is used by organizations monitoring network infrastructure. The overlap is where Hyperic's agents can be used to test local network infrastructure and OpenNMS can be used to communicate with agents and monitor platforms. OpenNMS is designed to be is strongly "agent-less" and Hyperic HQ's architecture is strongly "agent-based."

Often the argument for or against agent vs. agent-less monitoring is supposed by someone, somewhere, on the "Net". I suppose a hybrid "End-to-End" solution using both agent-less and agent-based monitoring systems can best serve the IT operations staff challenged with monitoring today's complex and diverse web-based applications. To support this supposition, the OpenNMS project has written plugins for Hyperic-HQ that will natively interact with OpenNMS. This allows OpenNMS distributable synthetic transaction monitors coupled with Hyperic's expert application and platform monitoring technologies to provide a comprehensive view from each vantage point of an application.

Hopefully, this integration will help is only the first step in a an ongoing collaborative integration project. The OpenNMS project will soon be releasing other integration software for other management and trouble ticketing systems. We hope this encourages and fosters more collaboration in this space. As always, since the OpenNMS project in an engineer's project, we speak with our feet. What follows are the technical details for integrating Hyperic-HQ with OpenNMS. Enjoy!

Concept

Integration with another monitoring application begins with sharing alert information. However, in order to make sense of (correlate) these shared alerts from other monitoring applications, inventory must be also be shared or related (for a complete design, see NGOSS). The OpenNMS configurations and Hyperic-HQ (HQ) software plugins included in this integration, accomplish these requirements from the perspective of OpenNMS. OpenNMS provides an API for synchronizing monitoring elements from one or more other monitoring systems, such as Hyperic, and provisioning databases. It also provides flexible means for correctly associating alerts from external systems with the correct entities imported into OpenNMS via an Event Bus adaptor called the Event Translator.

The HQ software is used to forward notifications from HQ to OpenNMS as an HQ escalation action and a servlet is provided for HQ (using HQ's cool Groovy plugin design) to export the platform inventory as an OpenNMS Model Import. Alerts are sent to OpenNMS via the OpenNMS TCP EventProxy and are then associated with an OpenNMS node entity that was imported from HQ using the translator.

Platforms imported as OpenNMS nodes retain their identity (platform ID) as an attribute on the OpenNMS node called foreign ID. When an alert is received from HQ, the HQ alert event is persisted for historical purposes and then translated into an event that is then assigned to the OpenNMS node by matching the alert event's platform ID parameter with the node's foreign ID. Confused yet? Don't worry, it's all been configured for you.

Preparing OpenNMS for Integration

There are two main components of this integration:

- On the OpenNMS Side:
  - The Hyperic Agent service definition and monitor
  - Hyperic HQ service definition and monitor
  - Definition of the new Hyperic Events and Alarms
  - Hyperic Alert Event Translator configuration
- On the Hyperic Side:
  - Hyperic Groovy based Servlet plugin
  - Hyperic SMS Email groovy script replacement
  - Configuration of escalation actions to send events to OpenNMS

The Completed OpenNMS Configuration

The default OpenNMS install comes with everything configured for the integration. The following is an itemized that defines all the configurations added to the default installation based on the OpenNMS hyperic-integration examples directory:

1. Verify "HypericAgent" service is defined in capsd-configuration.xml (see example)
2. Verify "HypericAgent" service is defined in a polling package in poller-configuration.xml (see example)
3. Verify the HQ event translation is configured in translator-configuration.xml (see example)
4. Verify the HQ event definition is defined in $OPENNMS_HOME/etc/events/Hyperic.events.xml (see example)
5. Verify that there is a reference to it and that a definition for the translated event are found in $OPENNMS_HOME/etc/eventconf.xml (see example)

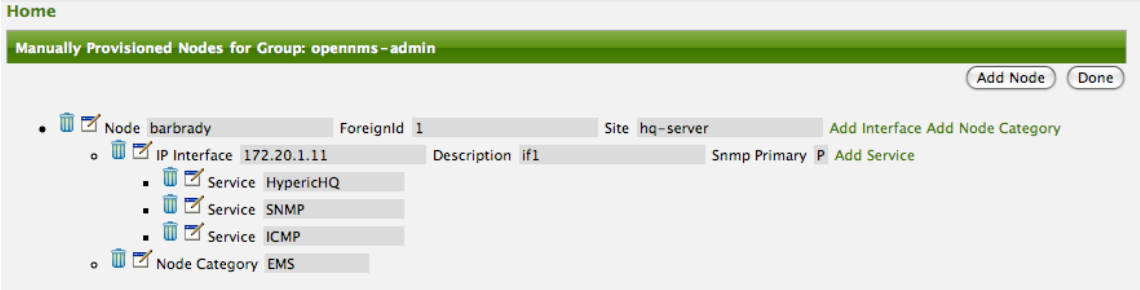
More Cool OpenNMS Preparation Stuff:

Additionally, another service monitor has been added to the OpenNMS default installation. This monitor supports the monitoring of the HQ server itself and is called: HypericHQ. This is a synthetic transaction monitor that connects to the HQ server and verifies a successful login and logout. Other pages may be added for further testing and verification of the server. Responsiveness is recorded so that thresholding can be used to alert for possible monitoring outages of the HQ application and server itself.

The default "hqadmin" login and password are used so change these to meet your requirements. You may also have to change the scheme to "https".

Security Note: *Scanning for this service has been disabled to prevent randomly sending your HQ admin login credentials to any host configured with the default HQ 7080 port.*

It is suggested that you create a provisioning group for your HQ servers to be monitored since the scanning for this service is disabled by default. Define the server with the HypericHQ service and import. A sample definition can be found in the OpenNMS hyperic-integration contrib directory called: imports-opennms-admin.xml.



Provisioning HQ Servers in OpenNMS

Best Practice Note: *This is a "best practice"... monitoring of entities that support your network management environment: SMS gateways, e-mail, ticketing systems, etc.*

Preparing Hyperic-HQ for Integration

Integration from HQ requires that 2 new facilities are added to HQ: an OpenNMS export servlet plugin and a replacement for the SMS groovy escalation action.

Installing Groovy Servlet

Copy all the files from the hyperic integration contrib directory to your HQ server:

```
scp -rp $OPENNMS_HOME/contrib/hyperic-integration/opennms \
$HQ_SERVER_HOME/hq-engine/server/default/deploy/hq.ear/hq.war/hqu
```

HQ will automatically recognize that the plugin has been installed and will server up the XML when requested.

> ***Note**: If you navigate to this URL in your desktop's browser, you will need to use the "view source" feature of your browser to see the XML.*

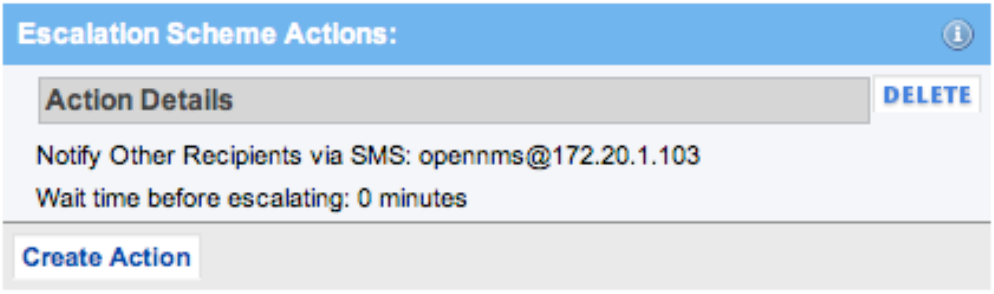
Installing OpenNMS HQ Escalation Action

HQ allows you to replace the SMS escalation action with your own Groovy plugin. This integration includes a plugin that you will use for this purpose. This plugin receives alerts from HQ and sends it as an XML encoded OpenNMS event to your OpenNMS instance's TCP Event Proxy on port 5817. Copy the replacement plugin as follows:

```
scp -rp $OPENNMS_HOME/contrib/hyperic-integration/alertTemplates/sms_email.gsp \
$HQ_SERVER_HOME/hq-engine/server/default/deploy/hq.ear/alertTemplates/sms_email.gsp
```

Configuring Escalation Action

The new SMS plugin reads the recipient details to determine if the alert should be sent to the OpenNMS Event Proxy. When the recipient is configured with an ID in the form of: "opennms@<host>", the new SMS action will forward the alert and all the alert details to OpenNMS to the specified "@host".



Hyperic HQ OpenNMS Alert Definition

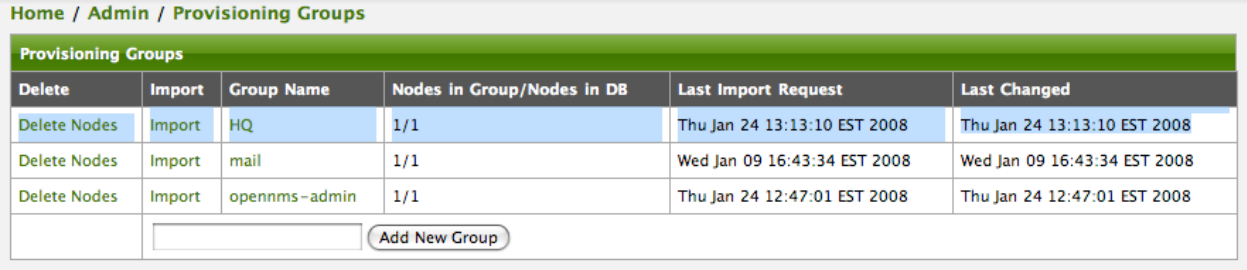
The Integration in Action

Importing HQ Platforms to OpenNMS Nodes

A Groovy based servlet has been developed for Hyperic called ExporterController.groovy and can be found in your OpenNMS install's contrib directory (versions 1.3.10+). This servlet will stream the model importer's XML and will synchronize Hyperic's platforms as OpenNMS nodes. You can add the URL to this servlet in the Model Importer service's properties file ($OPENNMS_HOME/etc/model-importer.properties) for cron like scheduled imports or you can use a tool such as wget for use with the OpenNMS Provision Group's WebUI interface. There is a sample script (imports-hq.sh) provided in the OpenNMS contrib directory.

Import using Provisioning Groups Interface

Edit the import-hq.sh located in the hyperic-integration contrib directory an run it to create the import-HQ.xml file in etc/ of $OPENNMS_HOME. Then, using the Provisioning Groups Interface, import the nodes to OpenNMS.



The Group Name will be, and must be, "**HQ**" to match with the "foreign-source" attribute inside the import file from the servlet and the name of the itself "imports-**HQ**.xml"

Configure model-importer.properties

Adjust the cron schedule for imports to your liking and add the proper URL:

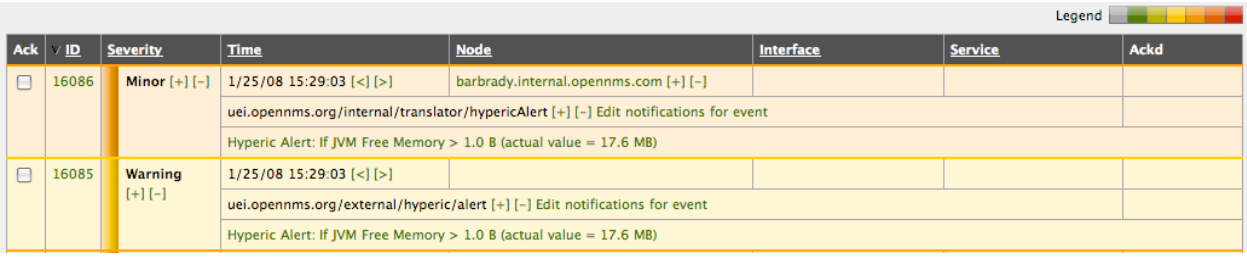
```
importer.importURL=http://myhqserver.mydomain.com:7080/hqu/opennms/exporter/index.hqu
importer.importSchedule=1 0 0 * * ?
```

Making these changes will schedule an import from HQ into opennms daily at one second after midnight.

Screen Shots

After OpenNMS 1.3.10 in installed and the contribution code is installed on Hyperic-HQ, OpenNMS will begin processing alerts from Hyperic and creating alarms. Just as with all OpenNMS alarms, the Hyperic alarms will be integrated into the automation system and notification systems.

The following is a screen shot of the event received from HQ and the translated event being associated with the node that was imported from HQ. The OpenNMS nodes are synchronized with HQ platforms during each import.

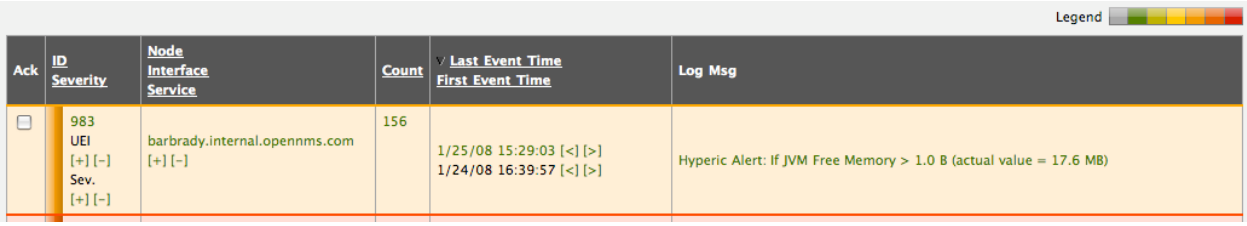


Legend

Ack	ID	Severity	Time	Node	Interface	Service	Ackd
☐	16086	Minor [+] [-]	1/25/08 15:29:03 [<] [>]	barbrady.internal.opennms.com [+] [-]			
			uei.opennms.org/internal/translator/hypericAlert [+] [-] Edit notifications for event				
			Hyperic Alert: If JVM Free Memory > 1.0 B (actual value = 17.6 MB)				
☐	16085	Warning [+] [-]	1/25/08 15:29:03 [<] [>]				
			uei.opennms.org/external/hyperic/alert [+] [-] Edit notifications for event				
			Hyperic Alert: If JVM Free Memory > 1.0 B (actual value = 17.6 MB)				

The HQ alert event and the translated event matching the imported from HQ node.

The next screen shot is from the OpenNMS Alarms interface showing that the translated event has been defined as an alarm in OpenNMS. Notice that the alarm has been raised 156 times (this is the event reduction feature of OpenNMS... we hope that your staff fixes this problem before it is seen this many times!)

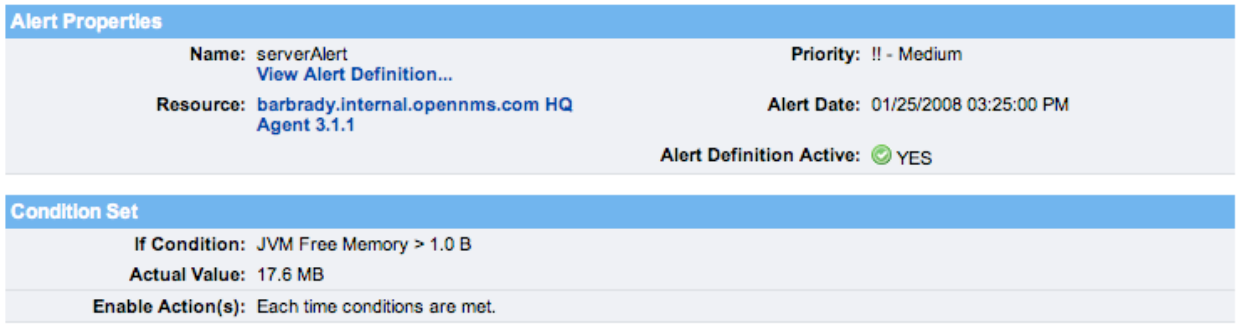


Legend

Ack	ID Severity	Node Interface Service	Count	Last Event Time First Event Time	Log Msg
☐	983 UEI [+] [-] Sev. [+] [-]	barbrady.internal.opennms.com [+] [-]	156	1/25/08 15:29:03 [<] [>] 1/24/08 16:39:57 [<] [>]	Hyperic Alert: If JVM Free Memory > 1.0 B (actual value = 17.6 MB)

The HQ alert now an OpenNMS alarm

Notice (if you can) that the Log Msg is formatted as a hyper-link. Clicking this link will take you directly to the alert in HQ console. Each time the event is reduced to the single alarm, the count will increase and the URL will update to be the most recent occurrence of the alert in HQ.



Alert Properties

Name: serverAlert
View Alert Definition...

Priority: !! - Medium

Resource: barbrady.internal.opennms.com HQ Agent 3.1.1

Alert Date: 01/25/2008 03:25:00 PM

Alert Definition Active: YES

Condition Set

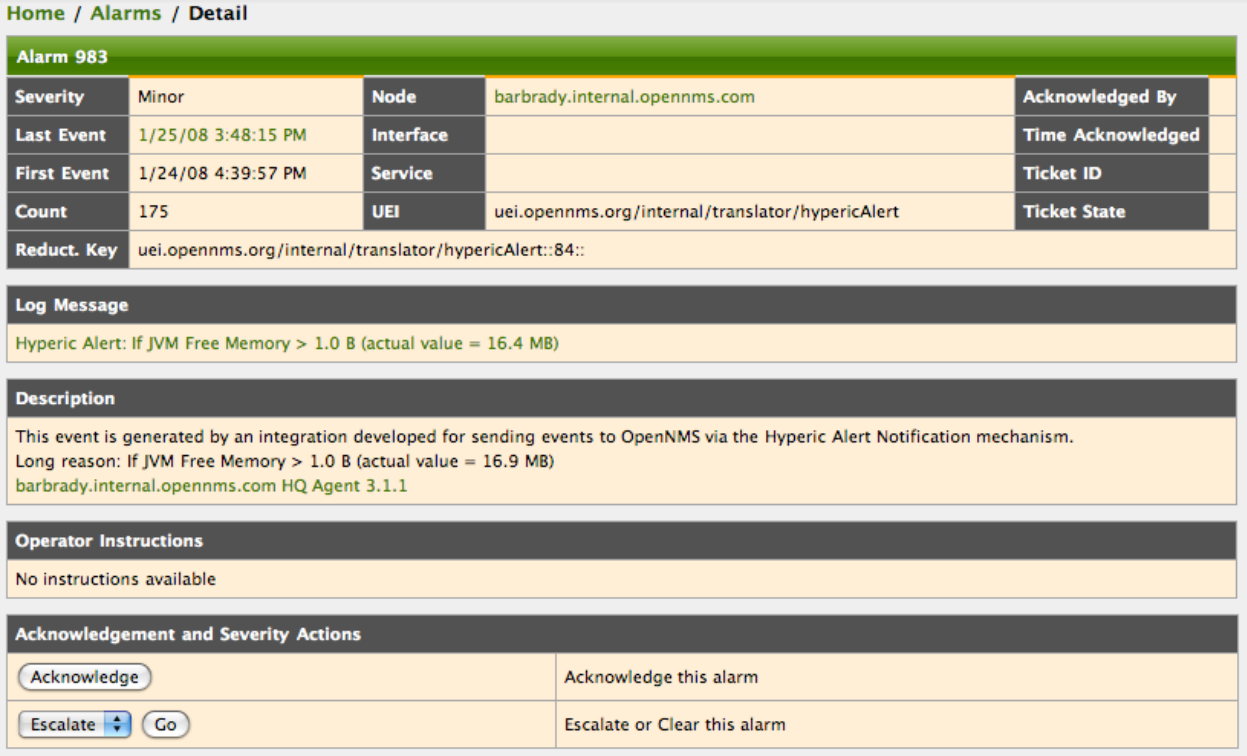
If Condition: JVM Free Memory > 1.0 B

Actual Value: 17.6 MB

Enable Action(s): Each time conditions are met.

The HQ alert viewed from clicking the URL in the OpenNMS alarm

Viewing the details of the alarm will present standard OpenNMS workflow: acknowledgments, escalations, clears, and trouble ticket system integration (not shown here).



HQ alert shown in OpenNMS alarm details page

Notice in the Description text another hyper-link. This hyper-link will direct the user to the page for this resource in the HQ console.

Default OpenNMS Configuration Examples

Define the services in capsd-configuration.xml

```
  <protocol-plugin protocol="HypericAgent" class-name="org.opennms.netmgt.capsd.plugins.TcpPlugin" scan="on"
user-defined="false">
    <property key="port" value="2144" />
    <property key="timeout" value="2000" />
    <property key="retry" value="1" />
  </protocol-plugin>
  <protocol-plugin protocol="HypericHQ" class-name="org.opennms.netmgt.capsd.plugins.HttpPlugin" scan="off"
user-defined="false">
    <property key="port" value="7080" />
    <property key="timeout" value="2000" />
    <property key="retry" value="1" />
  </protocol-plugin>
```

Define the monitoring of Hyperic Services in poller-configuration.xml

```
    <service name="HypericAgent" interval="300000" user-defined="false" status="on">
      <parameter key="retry" value="1" />
      <parameter key="timeout" value="2200" />
      <parameter key="port" value="2144" />
    </service>
    <service name="HypericHQ" interval="300000" user-defined="false" status="on">
      <parameter key="retry" value="1" />
      <parameter key="timeout" value="3000" />
      <parameter key="rrd-repository" value="/opt/opennms/share/rrd/response" />
      <parameter key="rrd-base-name" value="hyperic-hq" />
      <parameter key="ds-name" value="hyperic-hq" />
      <parameter key="page-sequence">
        <page-sequence>
          <page path="/Login.do" port="7080" successMatch="HQ Login" />
          <page path="/j_security_check.do" port="7080" method="POST"
            failureMatch="(?s)The username or password provided does not match our records" failureMessage="HQ
Login in Failed"
            successMatch="HQ Dashboard">
            <parameter key="j_username" value="hqadmin" />
            <parameter key="j_password" value="hqadmin" />
          </page>
          <page path="/Logout.do" port="7080" successMatch="HQ Login" />
        </page-sequence>
      </parameter>
    </service>
```

Define the event translations for Hyperic Alerts in translator-configuration.xml

```
    <event-translation-spec uei="uei.opennms.org/external/hyperic/alert">
      <mappings>
        <mapping>
          <assignment name="uei" type="field">
            <value type="constant" result="uei.opennms.org/internal/translator/hypericAlert" />
          </assignment>
          <assignment name="nodeid" type="field">
            <value type="sql" result="SELECT n.nodeid FROM node n WHERE n.foreignid = ? AND n.foreignsource
= ?">
              <value type="parameter" name="platform.id" matches=".*" result="${0}" />
              <value type="constant" result="HQ" />
            </value>
```

```
          </assignment>
        </mapping>
      </mappings>
    </event-translation-spec>
```

Define the Hyperic Alert Event in Hyperic.events.xml

```
<events>
<!-- Start of Hyperic Events -->
  <event>
    <uei>uei.opennms.org/external/hyperic/alert</uei>
    <event-label>OpenNMS defined event: An event has been received from Hyperic HQ</event-label>
    <descr>
      This event is generated by an integration developed for sending events to OpenNMS via
      the Hyperic Alert Notification mechanism.&lt;p&gt;

      &lt;p&gt;Long reason: %parm[action.longReason]% &lt;/p&gt;
      &lt;p&gt;&lt;a href=&quot;%parm[resource.url]%" &gt; %parm[resource.name]% &lt;/a&gt;&lt;/p&gt;
    </descr>
    <logmsg dest='logndisplay'>
      &lt;p&gt;&lt;a href=&quot;%parm[alert.url]%" &gt; Hyperic Alert: %parm[action.longReason]% &lt;/
a&gt;&lt;/p&gt;
    </logmsg>
    <severity>Warning</severity>
<!--    <alarm-data reduction-key="%uei%:%dpname%" alarm-type="1" auto-clean="false" /> -->
  </event>

</events>
```

Define the Hyperic Translated Event in eventconf.xml

```
  <event>
    <uei>uei.opennms.org/internal/translator/hypericAlert</uei>
    <event-label>OpenNMS defined event: An event received from Hyperic has been translated</event-label>
    <descr>
      This is a translated Hyperic Alert to associate with OpenNMS entity..&lt;p&gt;

      &lt;p&gt;Alert reason: %parm[action.shortReason]% &lt;/p&gt; &lt;p&gt;Alert reason:
%parm[action.longReason]% &lt;/p&gt;
      &lt;p&gt;&lt;a href=&quot;%parm[resource.url]%" &gt; %parm[resource.name]% &lt;/a&gt;&lt;/p&gt;
    </descr>
    <logmsg dest='logndisplay'>
      &lt;p&gt;&lt;a href=&quot;%parm[alert.url]%" &gt; Hyperic Alert: %parm[action.longReason]% &lt;/
a&gt;&lt;/p&gt;
    </logmsg>
    <severity>Minor</severity>
    <alarm-data reduction-key="%uei%:%dpname%:%nodeid%:%interface%:%service%" alarm-type="1" auto-
clean="false" />
  </event>

  <event-file>events/Hyperic.events.xml</event-file>
```