

End To End Monitoring

Hyperic HQ[®] Integration With OpenNMS[®]

David Hustace, Jeff Gehlbach
The OpenNMS Group, Inc.

Revision History

Version	Date	Author	Comments
1.0	1-Jan-2008	David Hustace	Initial version
1.1	29-Apr-2009	Jeff Gehlbach	Updated for OpenNMS 1.6 and HQ 4.1

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 both are written in Java and are professionally developed and commercially supported.

Even though at first glance it would appear each of these projects serves 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 its 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 the vantage point of each application user.

Hopefully, this integration will be only the first step in a an ongoing collaborative integration project. The OpenNMS project is constantly releasing 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 is 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 also be shared or related (for a complete design, see NGOSS¹). 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 monitored elements from one or more other monitoring systems, such as HQ, 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 alert action and a servlet is provided for HQ (using HQ's cool HQU [Groovy](#) plugin framework) 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 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.

¹ <http://www.tmforum.org/SolutionFrameworks/1911/home.html>

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
 - Eventd TCP listener
- On the Hyperic HQ side:
 - Hyperic HQU platform-export plugin (already done for you)
 - Hyperic Groovy based OpenNMS alert action definition
 - Configuration of alerts to send events to OpenNMS

Version Note: *The following instructions pertain to OpenNMS release from 1.6.0 onward.*

The Completed OpenNMS Configuration

The default OpenNMS install comes with all but one security-related setting configured for the integration. The following is an itemized list that defines all the configurations included in the default installation based on the OpenNMS *hyperic-integration* examples directory:

1. Verify "HypericAgent" service is defined in `capsd-configuration.xml` (already done for you; [see details](#))
2. Verify "HypericAgent" service is defined in a polling package in `poller-configuration.xml` (already done for you; [see details](#))
3. Verify the HQ event translation is configured in `translator-configuration.xml` (already done for you; [see details](#))
4. Verify the HQ event definition is defined in `$(OPENNMS_HOME)/etc/events/Hyperic.events.xml` (already done for you; [see details](#))
5. Verify that there is a reference to it and that a definition for the translated event are found in `$(OPENNMS_HOME)/etc/eventconf.xml`
6. Configure the OpenNMS Eventd TCP listener to listen for connections on all interfaces (must be done once manually; [see details](#))

More Cool OpenNMS Preparation Stuff

For security reasons, the TCP listener used by OpenNMS' event daemon to receive XML-formatted events binds by default only to the software loopback interface at IP address 127.0.0.1. If HQ is on a separate server from OpenNMS, you will need to configure this listener to bind to all IP interfaces. Edit the file `$(OPENNMS_HOME)/etc/eventd-configuration.xml` and change the `TCPAddress` attribute of the `<EventdConfiguration>` element from "127.0.0.1" to "0.0.0.0". You must restart OpenNMS for this change to take effect.

Possibly Required Changes to HQ Service Monitoring in OpenNMS

A service monitor has been added to the OpenNMS default installation that 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. Response time 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 match the configuration of your HQ server. You may also have to change the *scheme* to “https” if your HQ server is configured not to offer its web interface via unsecured HTTP.

Security Note: *Scanning for the HypericHQ service has been disabled to avoid sending your HQ admin login credentials to any host configured to listen on the default HQ port (7080).*

It is suggested that you create a provisioning group for your HQ servers to be monitored since 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*.

The screenshot shows the OpenNMS Provisioning Groups interface. At the top, there is a breadcrumb trail: Home / Admin / Provisioning Groups / Edit. Below this is a green header bar that reads "Manually Provisioned Nodes for Group: opennms-admin". To the right of this bar are two buttons: "Add Node" and "Done". The main content area displays a tree view of nodes. The root node is "Node barbrady" with a trash icon and a checkmark. It has three sub-nodes: "IP Interface 172.20.1.11" (with a trash icon and checkmark), "Node Category EMS" (with a trash icon and checkmark), and "ForeignId 1" (with a trash icon and checkmark). The "IP Interface 172.20.1.11" node has three sub-services: "Service HypericHQ", "Service SNMP", and "Service ICMP", each with a trash icon and checkmark. To the right of the "IP Interface" node are fields for "Description if0" and "Snmp Primary P", and a button "Add Service". To the right of the "Node barbrady" node are fields for "Site opennms-admin" and buttons "Add Interface" and "Add Node Category".

Manually Provisioning HQ Servers in the OpenNMS Provisioning Groups interface

Best Practice Note: *Monitoring of entities that support your network management environment (SMS gateways, e-mail, ticketing systems, etc.) is considered a “best practice”*

Feature Availability Note: *The Event Translator, Alarms, Model Importer, and Provisioning Groups features are available in all stable OpenNMS releases starting with 1.6.0.*

Preparing Hyperic HQ for Integration

Integration from the HQ side requires that the OpenNMS alert mechanism (included with HQ but disabled by default) be enabled.

Version Note: *These instructions pertain to Hyperic HQ version 4.1.1.*

Enabling the HQ OpenNMS Alert Definition

HQ includes an alert mechanism for sending events to an OpenNMS server, but the default HQ configuration does not include a definition enabling this alert mechanism to be used. To enable this alert mechanism, edit the following file on your HQ server:

```
$HQ_SERVER_HOME/hq-engine/server/default/deploy/hq.ear/alertdef-prop-service.xml
```

Near the bottom of this file is an XML `<attribute name="Properties">` tag that contains an empty definition for the value of a property called `actions`. Simply add the name of the OpenNMS event alert action class as the value of this property:

```
actions = \
    org.hyperic.hq.bizapp.server.action.integrate.OpenNMSAction
```

The HQ server must be restarted in order for the new alert mechanism to be available for use.

Configuring Alerts

To configure HQ to sent alerts for certain conditions as OpenNMS events, simply configure an alert against some resource as you would normally do to notify an HQ user or other recipients. Once the OpenNMS alert action is enabled, you will see a new tab in the alert workflow labeled *OpenNMS*. Fill in the IP address of your OpenNMS server in the "Server:" text field. If you have configured OpenNMS' Eventd TCP listener to bind to a port other than the default of 5817, enter that port number in the "Port (default 5817):" text field.

Alert Properties	
Name: 5 min Load > 1	Priority: !! - Medium
Description: Five-minute load average above 1.0	Active: Yes
	Date Created: 04/16/2009 11:56 AM
	Date Modified: 04/17/2009 01:26 PM
EDIT...	
Condition Set	
If Condition: Load Average 5 Minutes > 1.0	
Enable Action(s): Each time conditions are met.	
EDIT...	
Escalation Notify HQ Users Notify Other Recipients OpenNMS	
Server: <input type="text" value="172.20.1.11"/>	Port (default 5817): <input type="text" value="5817"/>
SET REMOVE	

OpenNMS Alert Definition in HQ

Feature Availability Note: The open source community edition of Hyperic HQ requires that resource alerts be configured individually. Automatic application of alert definitions to groups of resources or globally across resources of a given type is available in HQ Enterprise Edition.

The Integration in Action

Importing HQ Platforms to OpenNMS Nodes

A servlet ships with Hyperic HQ that makes HQ's inventory of platforms available in an XML format that OpenNMS can use. The OpenNMS Model Importer, once configured to use the URL of this exporter servlet, will synchronize HQ's platforms as OpenNMS nodes. You can add the URL of this servlet in the Model Importer service's properties file (`$OPENNMS_HOME/etc/model-importer.properties`) for scheduled imports on a *cron*-like schedule, or you can use a tool such as *wget* or *curl* to fetch the servlet's output for manual use with the OpenNMS Provisioning Groups web interface. There is a sample script ([imports-hq.sh](#)) provided in the OpenNMS *contrib/hyperic-integration* directory that shows this latter usage.

Import Using Provisioning Groups Interface

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

Home / Admin / Provisioning Groups

Provisioning Groups					
Delete	Import	Group Name	Nodes in Group/Nodes in DB	Last Import Request	Last Changed
Delete Nodes	Import	HQ	2/1		Fri Apr 17 13:09:35 EDT 2009
Delete Group	Import	mail	3/0	Wed Apr 15 10:30:42 EDT 2009	Wed Apr 15 10:30:42 EDT 2009
Delete Nodes	Import	opennms-admin	1/1	Fri Apr 17 12:37:22 EDT 2009	Fri Apr 17 12:42:00 EDT 2009
		<input type="text"/>	<input type="button" value="Add New Group"/>		

The Group Name will be, and must be, "**HQ**" to match the *foreign-source* attribute inside the import file from the servlet and the name of the file 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 1.6.x is installed and the OpenNMS event alert action is enabled in HQ, OpenNMS will begin processing alerts from HQ and creating alarms. Just as with all OpenNMS alarms, the Hyperic HQ alarms will be integrated into the automation system and notification systems and can be used to create help desk tickets in an external system via OpenNMS' trouble ticket integration API.

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.

Ack	ID	Severity	Time	Node	Interface	Service	Ackd
<input type="checkbox"/>	5493	Minor [+] [-]	4/17/09 13:10:51 [<] [>]	barbrady.internal.opennms.com [+] [-]			
uei.opennms.org/internal/translator/hypericAlert [+] [-] Edit notifications for event							
Hyperic Alert: If Load Average 5 Minutes > 1.0 (actual value = 1.4)							
<input type="checkbox"/>	5492	Warning [+] [-]	4/17/09 13:10:51 [<] [>]				
uei.opennms.org/external/hyperic/alert [+] [-] Edit notifications for event							
Hyperic HQ Alert: If Load Average 5 Minutes > 1.0 (actual value = 1.4)							

The HQ alert event, and translated event matching the node imported from corresponding HQ platform, in the OpenNMS event browser

The next screen shot is from the OpenNMS Alarms interface showing that the translated event has spawned an alarm in OpenNMS. Note that the alarm has been raised 110 times (the event reduction feature of OpenNMS... we hope that your staff fixes this problem before it gets this bad!)

Ack	ID Severity	Node Interface Service	Count	Last Event Time First Event Time	Log Msg
<input type="checkbox"/>	24 UEI [+] [-] Sev. [+] [-]	barbrady.internal.opennms.com [+] [-]	110	4/17/09 13:05:50 [<] [>] 4/17/09 13:05:50 [<] [>]	Hyperic Alert: If Load Average 5 Minutes > 1.0 (actual value = 1.0)

The HQ alert, now represented as an alarm (note the Count column) in the OpenNMS alarm browser

Notice that the Log Message is formatted as a hyperlink. 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 reference the most recent occurrence of the alert in HQ.

Alert Properties	
Name: 5 min Load > 1	Priority: !! - Medium
Resource: barbrady.internal.opennms.com	Alert Date: 04/17/2009 12:05 PM
Description: Five-minute load average above 1.0	Alert Status: NOT FIXED
Condition Set	
If Condition: Load Average 5 Minutes > 1.0	
Actual Value: 1.0	
Enable Action(s): Each time conditions are met.	

The HQ alert viewed in HQ by clicking the drill-back link from the OpenNMS alarm details

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

Home / Alarms / Detail

Alarm 24				
Severity	Minor	Node	barbrady.internal.opennms.com	Acknowledged By
Last Event	4/17/09 1:05:50 PM	Interface		Time Acknowledged
First Event	4/17/09 1:05:50 PM	Service		Ticket ID
Count	1	UEI	uei.opennms.org/internal/translator/hypericAlert	Ticket State
Reduct. Key	uei.opennms.org/internal/translator/hypericAlert::6::barbrady.internal.opennms.com:5 min Load > 1			

Log Message

Hyperic Alert: If Load Average 5 Minutes > 1.0 (actual value = 1.0)

Description

This event is generated by an integration developed for sending events to OpenNMS via the Hyperic Alert Notification mechanism.
 Long reason: If Load Average 5 Minutes > 1.0 (actual value = 1.0)
[barbrady.internal.opennms.com](#)

Operator Instructions

No instructions available

Acknowledgement and Severity Actions

<input type="button" value="Acknowledge"/>	Acknowledge this alarm
<input type="button" value="Escalate"/> <input type="button" value="Go"/>	Escalate or Clear this alarm

Details for an alarm derived from an HQ alert, viewed in the OpenNMS alarm browser

Notice in the Description text another hyperlink whose text is just the name of the node to which the alarm pertains. This hyperlink will direct the user to the page for the corresponding platform in the HQ console.

Default OpenNMS Configuration Examples

Define the services in capsd-configuration.xml

These definitions are already present in the default configuration files.

```
<protocol-plugin protocol="HypericAgent" class-name="org.opennms.netmgt.capsd.plugins.TcpPlugin" scan="on">
  <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">
  <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

These definitions are already present in the default configuration files.

```
<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)|(Sign in to Hyperic HQ)" />
      <page path="/j_security_check.do" port="7080" method="POST"
        failureMatch="(?!s)(The username or password provided does not match our records)|(You are not
signed in)" failureMessage="HQ Login 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>
<service name="MySQL" interval="300000" user-defined="false" status="on">
```

Define the event translations for HQ alerts in translator-configuration.xml

These definitions are already present in the default configuration files.

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

```

    <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 HQ alert event in Hyperic.events.xml

These definitions are already present in the default configuration files.

```

<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>

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

```

Define the HQ translated event in eventconf.xml

These definitions are already present in the default configuration files.

```

<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>

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

```

Configure the Eventd TCP listener

This step represents a change to the default configuration files, and *must* be completed for the integration to work. Only the value of the *TCPAddress* attribute (highlighted distinctively below) needs to be changed.

```
<EventdConfiguration
  TCPAddress="0.0.0.0"
  TCPPort="5817"
  UDPAddress="127.0.0.1"
  UDPPort="5817"
  receivers="5"
  getNextEventID="SELECT nextval('eventsNxtId')"
  getNextAlarmID="SELECT nextval('alarmsNxtId')"
  socketSoTimeoutRequired="yes"
  socketSoTimeoutPeriod="3000">
</EventdConfiguration>
```