vFabric Hyperic Configuration

v.5.7

June 2013

EN-000957-01

vmware

Legal Notice

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Contents

Configure and Run the Hyperic Agent

Understand Agent Environment and Operation
For background information about agent configuration, communications, startup, and installation environment, see *Key Facts for Hyperic Installers* in *Getting Started with vFabric Hyperic*.

Start, Stop, and Other Agent Operations

Starting an Agent for the First Time
The first time you start a vFabric Hyperic Agent, you must supply setup information (unless it is available in `agent.properties`). For information about starting an agent for the first time, and installing the agent as a service in Windows environments, see Set Up Agent Interactively.

Run the Agent Launcher from the Command Line

You initiate the agent launcher and agent lifecycle commands with the `hq-agent.sh`, or `hq-agent.bat` script, in the `AgentHome/bin` directory.

Running hq-agent.sh

1. Open a command shell or terminal window.
2. Enter a command of this form:

```
sh hq-agent.sh command
```

   Where command is one of the following:
   - `start` — Starts the agent as a daemon process.
   - `stop` — Stops the agent's JVM process.
   - `restart` — Stops and then starts the agent's JVM process
   - `status` — Queries the status of the agent's JVM process.
   - `dump` — Runs a thread dump for the agent process, and writes the results to the `agent.log` file in `AgentHome/log`.
   - `ping` — Pings the agent process.
   - `setup` — Causes the Hyperic Agent to prompts you for the agent-server connection properties, allowing you to change the values that were provided at first agent startup.

Running hq-agent.bat

1. Open a terminal window.
2. Enter a command of this form:

```
hq-agent.bat command
```

   Where command is one of the following:
   - `start` - starts the agent as an NT service
   - `stop` - stops the agent as an NT service
   - `restart` - stops and then starts the agent's JVM process
   - `install` - installs the agent NT service
   - `remove` - removes the agent's service from the NT service table
   - `query` - queries the current status of the agent NT service (status
   - `ping` - pings the agent process for availability
   - `setup` - prompts for setup configuration for the agent process

Run the Agent Launcher from the Hyperic User Interface

In vFabric Hyperic, you can issue selected commands to an running Hyperic Agent.

Agent control commands are available on the **Views** tab for an Hyperic Agent or a group of agents in inventory.

Restart an Agent from the Hyperic User Interface

The **restart** command invokes the agent's Java Service Wrapper's restart command. The restart command shuts down the JVM process in which the agent runs, waits for the process to terminate cleanly, and spawns a new JVM process for the agent. During the restart process, the agent's metric collection and resource control functionality will be interrupted.

The restart command happens asynchronously. To verify that the restart succeeded you can go to the page for the agent in the Hyperic user interface and check its availability. Alternatively, you could configure an alert for the agent that fires when the agent's availability changes.

To restart an agent:

1. Navigate to the **Views** tab for an Hyperic Agent or a group of agents.
2. Select **restart** from the pull-down list
3. Click **Execute**.

Ping an Agent from the Hyperic User Interface

1. Navigate to the **Views** tab for an Hyperic Agent or a group of agents.
2. Select **ping** from the pull-down.
3. Click **Execute**.

Upgrade an Agent from the Hyperic User Interface

1. Navigate to the **Views** tab for an Hyperic Agent or a group of agents.
2. Select the **upgrade** command from the pull-down.
3. On the pull-down list of agent bundles that appears, select the desired bundle
4. Click **Execute**.
    - The selected agent bundle is transferred from the Hyperic Server to the target agent(s).
    - The agent expands the bundle locally.
    - The agent updates the local bundle property.
    - The server restarts the agent.

The configuration settings in the agent's `/conf/agent.properties` file are preserved.

Push a Resource Plugin to the Agent from the Hyperic User Interface

This **push plugin** command sends new and changed resource plugins to the target agent(s). Pushing plugins to an agent results in an agent restart.

1. Navigate to the **Views** tab for an Hyperic Agent or a group of agents.
2. Select the **push plugin** command from the pull-down.
3. On the pull-down list of plugins that appears, select the desired plugin.
4. Click **Execute**.
    - The selected plugin is transferred from the Hyperic Server to the target agent(s)
    - The server restarts the agent(s).

Run the Agent Without the Java Service Wrapper

If you run an Hyperic Agent on a system that does not support the Java Service Wrapper, or for other reasons prefer not to use the wrapper, you can start the agent without the wrapper.

The `hq-agent-nowrapper.sh` agent start script in `AgentHome/bundles/agent-x.y.z/bin`

Because `hq-agent-nowrapper.sh` does not fork itself into the background, run it using `nohup`:

```
nohup AgentHome/bundles/agent-x.y.z/bin/hq-agent-nowrapper.sh &
```

Configure Auto-Discovery Scanning and Reporting

This page lists agent properties that control auto-inventory scanning and reporting. For more information, see *Configure Auto-Discovery Frequency* in *vFabric Hyperic Administration*.

autoinventory.defaultScan.interval.millis
Description
Specifies how frequently the agent performs a default autoinventory scan.
The default scan detects servers and platform services, typically using the process table or the Windows registry. Default scans are less resource-intensive than runtime scans.
Default
Commented out, set to 86,400,000 milliseconds, or 1 day.
Note however, that by default, the agent performs the default scan at startup and every 15 minutes thereafter.

autoinventory.runtimeScan.interval.millis
Description
Specifies how frequently the agent performs a runtime scan.
A runtime scan may use more resource-intensive methods to detect services than a default scan. For instance, a runtime scan may involve issuing an SQL query or looking up an MBean.
Default
86,400,000 milliseconds, or 1 day.

Configure Agent Logging

Agent Log Files

Agent log files are stored in the `AgentHome/log` directory. They include:

- `agent.log`
- `agent.startup.log`
- `wrapper.log` - The agent JSW-based launcher writes messages to this log.

Configure Agent Log Name or Location

Use these properties to change the name or location of the agent log file:

agent.logDir
Description
You can add this property to the `agent.properties` file to specify the directory where the Hyperic Agent will write its log file. Unless you specify a fully qualified path, `agent.logDir` is evaluated relative to the agent installation directory.
This property does not exist in `agent.properties` unless you explicitly add it. To change the location for the agent log file, add `agent.logDir` to `agent.properties` and enter a path relative to the agent installation directory, or if desired, a fully qualified path.
Note that the name of the agent log file is configured with the `agent.logFile` property.
Default
`agent.logDir` does not exist in `agent.properties` unless you explicitly add it. The default behavior is equivalent to this setting:
`agent.logDir=log`
resulting in the agent log file being written to the `AgentHome/log` directory.

agent.logFile
Description
Specifies the path and name of the agent log file.
Default
Note that in `agent.properties`, the default setting for `agent.LogFile` is made up of a variable and a string.

```
agent.logFile=${agent.logDir}\agent.log
```

- `agent.logDir` is a variable - it supplies the value of an identically named agent property. By default, the value of `agent.logDir` is `log`, interpreted relative to the agent installation directory.
- `agent.log` is the name for the agent log file.

So, by default, the agent log file is named `agent.log`, and is written to the `AgentHome/log` directory.
If you want the agent to the log to a different directory, you must explicitly add the agent.logDir property to `agent.properties`.

Configure Agent Logging Level

Use this property to control what severity of messages that agent writes to the agent log file:

agent.logLevel
Description
Specifies the level of detail of the messages the Agent writes to the log file. Allowable values are: INFO and DEBUG.
Default
INFO

Setting the `agent.logLevel` to to "DEBUG" level is not advised. This level of logging across all subsystems will impose overhead, and can also cause the log file to roll over so frequently that log messages of interest are lost. It is preferable to configure debug level logging only at the subsystem level, as described in Configure Debug Log Level for an Agent Subsystem.

Redirect System Messages to the Agent Log

Use these properties to redirect system-generated messages to the agent log file:

agent.logLevel.SystemErr
Description
Redirects System.err to agent.log. Commenting out this setting will cause System.err to be directed to agent.log.startup.
Default
ERROR

agent.logLevel.SystemOut
Description
Redirects System.out to agent.log. Commenting out this setting will cause System.out to be directed to agent.log.startup.
Default
INFO

Configure Debug Log Level for an Agent Subsystem

For troubleshooting purposes you can increase logging level for an individual agent subsystem. To do so, uncomment the appropriate line in the section of `agent.properties` labelled "Agent Subsystems: Uncomment individual subsystems to see debug messages", shown in the excerpt below.

Agent log4j Properties

The log4j properties in `agent.properties` are excerpted below.

```
log4j.rootLogger=${agent.logLevel}, R

log4j.appender.R.File=${agent.logFile}
log4j.appender.R.MaxBackupIndex=1
log4j.appender.R.MaxFileSize=5000KB
log4j.appender.R.layout.ConversionPattern=%d{dd-MM-yyyy HH:mm:ss,SSS z} %-5p [%t]
[%c{1}@%L] %m%n
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R=org.apache.log4j.RollingFileAppender

##
## Disable overly verbose logging
##
log4j.logger.org.apache.http=ERROR
log4j.logger.org.springframework.web.client.RestTemplate=ERROR
log4j.logger.org.hyperic.hq.measurement.agent.server.SenderThread=INFO
log4j.logger.org.hyperic.hq.agent.server.AgentDListProvider=INFO
log4j.logger.org.hyperic.hq.agent.server.MeasurementSchedule=INFO
log4j.logger.org.hyperic.util.units=INFO
log4j.logger.org.hyperic.hq.product.pluginxml=INFO

# Only log errors from naming context
log4j.category.org.jnp.interfaces.NamingContext=ERROR
log4j.category.org.apache.axis=ERROR

#Agent Subsystems: Uncomment individual subsystems to see debug messages.
#-------------------------------------------------------------------------
#log4j.logger.org.hyperic.hq.autoinventory=DEBUG
#log4j.logger.org.hyperic.hq.livedata=DEBUG
#log4j.logger.org.hyperic.hq.measurement=DEBUG
#log4j.logger.org.hyperic.hq.control=DEBUG

#Agent Plugin Implementations
#log4j.logger.org.hyperic.hq.product=DEBUG

#Server Communication
#log4j.logger.org.hyperic.hq.bizapp.client.AgentCallbackClient=DEBUG

#Server Realtime commands dispatcher
#log4j.logger.org.hyperic.hq.agent.server.CommandDispatcher=DEBUG
```

```
#Agent Configuration parser
#log4j.logger.org.hyperic.hq.agent.AgentConfig=DEBUG

#Agent plugins loader
#log4j.logger.org.hyperic.util.PluginLoader=DEBUG

#Agent Metrics Scheduler (Scheduling tasks definitions & executions)
#log4j.logger.org.hyperic.hq.agent.server.session.AgentSynchronizer.SchedulerThrea
d=DEBUG

#Agent Plugin Managers
#log4j.logger.org.hyperic.hq.product.MeasurementPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.AutoinventoryPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.ConfigTrackPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.LogTrackPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.LiveDataPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.ControlPluginManager=DEBUG
```

Configure Plugin Loading

By default, at startup, a Hyperic Agent loads all of the plugins in its plugin directory - `AgentHome/bundles/agent-x.y.z-nnnn/pdk/plugins`.

You can reduce the agent's memory footprint by configuring it to load only the plugins you use. You can either specify a list of plugins to exclude, or configure a list of the plugins to load.

plugins.exclude
Description
Use this property to specify plugins that you do not wish the Hyperic Agent to load at startup. This is useful for reducing the agent's memory footprint.
Usage
Supply a comma-separated list of plugins to exclude, for example:

```
plugins.exclude=jboss,apache,mysql
```

plugins.include
Description
Use this property to specify plugins that you do wish the Hyperic Agent to load at startup. This is useful for reducing the agent's memory footprint.
Usage
Supply a comma-separated list of plugins to include, for example:

```
plugins.include=weblogic,apache
```

Tweak the Agent to Enable a Resource Plugin

These topics describe how to configure the agent to enable a particular plugin to perform one or more of its management functions:

Configure Agent Account Privileges under Solaris 10

To auto-discover certain products under Solaris 10, the Hyperic Agent must run as root or you must explicitly grant additional permissions to the account where the agent runs. For background information, see Solving Auto-Discovery Problems.

Under Solaris 10's Least Privilege Model (LPM), default privileges are minimal. The Hyperic Agent must be able to read `/proc/$pid/` files on the platform. Problems with auto-discovery on Solaris 10 may be the result of insufficient privileges. Depending on your account privilege implementation you may need to grant the `proc_zone` privilege to the agent account.

For example, you could add the following line to `/etc/user_attr`, to grant the `proc_owner` privilege to the hq user and deny the `proc_session` privilege:

```
hq::::type=normal;defaultpriv=basic,proc_owner,!proc_session
```

Note: After changing account privileges, the user needs to re-login.

Your approach for enabling agent access to `/proc/$pid/` files will depend on your company's LPM implementation and best practices.

Configure Agent HTTP Request Header

If you monitor a remote HTTP server, it is useful to configure the HTTP request header for agent HTTP requests.

http.useragent
Description
The `http.useragent` property defines the value for the User-Agent request header in HTTP requests issued by the Hyperic Agent. By default, the User-Agent in agent requests includes the Hyperic Agent version - and so change upon agent upgrade. If a target HTTP server is configured to block requests with an unknown User-Agent, agent requests will fail after agent upgrade.
You can use `http.useragent` to define a User-Agent value that will be consistent across upgrades.
Note: `agent.properties` does not contain this property by default. You must explicitly add it to the file.
Default
`Hyperic-HQ-Agent/Version`
For example:
`Hyperic-HQ-Agent/4.1.2-EE`

Configure Agent to Monitor JBoss

jboss.installpath
Description
To enable the agent to monitor JBoss, specify the location of the JBoss root directory.
Default
/usr/local/jboss-4.0.0

Configure the Data to Log for Windows Events

This page describes the use of the `platform.log_track.eventfmt` agent property to customize the content of events that the Hyperic Agent logs for Windows Events, when log tracking is enabled for a Windows resource.

platform.log_track.eventfmt
Description
Specifies the content and format of the Windows event attributes that a Hyperic Agent includes when logging a Windows event as an event in Hyperic. `agent.properties` does not contain the `platform.log_track.eventfmt` property, you must explicitly add it if you want to tailor the data logged for Windows events.
Default Behavior When Windows log tracking is enabled, an entry of this form is logged for

events that match the criteria you specified on the resource's **Configuration Properties** page:
[Timestamp] Log Message (EventLogName):EventLogName:EventAttributes
where:

- ***Timestamp*** - is when the event occurred
- **Log Message** - is an text string
- ***EventLogName*** - is the Windows event log type, "System", "Security", or "Application".
- ***EventAttributes*** - a colon delimited string made of the Windows event **Source** and **Message** attributes.

For example, this log entry:
04/19/2010 06:06 AM Log Message (SYSTEM): SYSTEM: Print: Printer HP LaserJet 6P was paused.
is for an Windows event written to the Windows System event log at 6:06 AM on 04/19/2010. The Windows event **Source** and **Message** attributes, are "Print" and "Printer HP LaserJet 6P was paused.", respectively.

Configuration

You can use the parameters below to configure the Windows event attributes that the agent writes for a Windows event. Each parameter maps to Windows event attribute of the same name.

- `%user%` — The name of the user on whose behalf the event occurred.
- `%computer%` — The name of the computer on which the event occurred.
- `%source%` — The software that logged the Windows event.
- `%event%` — A number identifying the particular event type.
- `%message%` — The event message.
- `%category%` — An application-specific value used for grouping events.

For example, with this property setting:

```
platform.log_track.eventfmt=%user%@%computer%
%source%:%event%:%message%
```

the Hyperic Agent will write the following data when logging Windows event:

```
04/19/2010 06:06 AM Log Message (SYSTEM): SYSTEM: HP_Admistrator@Office
Print:7:Printer HP LaserJet 6P was paused.
```

This entry is for as for an Windows event written to the Windows System event log at 6:06 AM on 04/19/2010. The software associated with the event was running as "HP_Administrator" on the host "Office". The Windows event's **Source, Event,** and **Message** attributes, are "Print", "7", and "Printer HP LaserJet 6P was paused.", respectively.

Manage the Hyperic Agent

These topics have information about monitoring and tuning the Hyperic Agent.

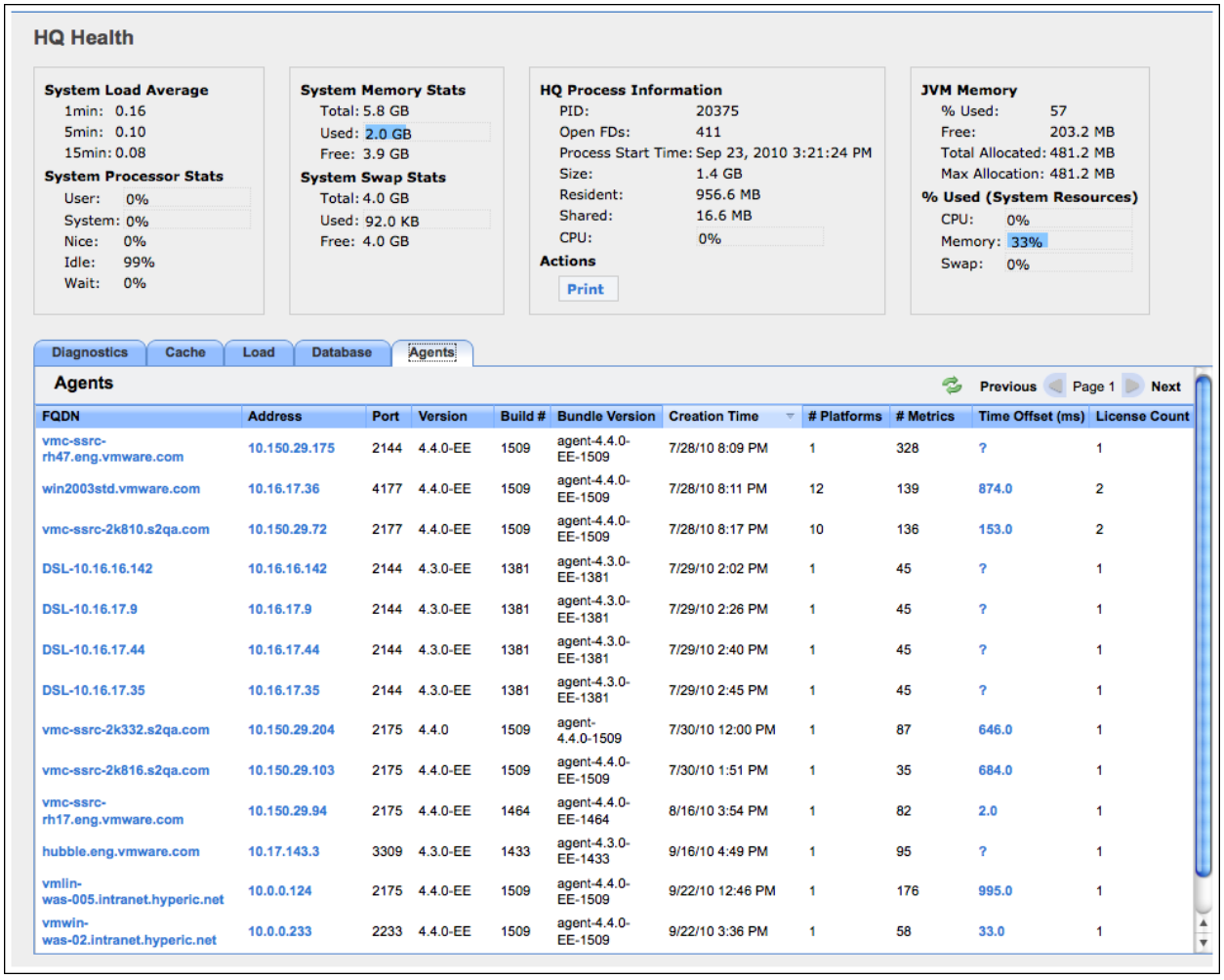
View Status of all Hyperic Agents

The page has information about the **Agents** tab of the **HQ Health** page — a screenshot, and definitions of the health data for an agent.

Agents Tab in HQ Health

The **Agents** tab of the **HQ Health** page — shown in the screenshot below — shows the status of all of the Hyperic Agents that are registered with the Hyperic Server.

To navigate to **HQ Health**, click the link for it in the **Plugins** section of the **Administration** page.



FQDN	Address	Port	Version	Build #	Bundle Version	Creation Time	# Platforms	# Metrics	Time Offset (ms)	License Count
vmc-ssrc-rh47.eng.vmware.com	10.150.29.175	2144	4.4.0-EE	1509	agent-4.4.0-EE-1509	7/28/10 8:09 PM	1	328	?	1
win2003std.vmware.com	10.16.17.36	4177	4.4.0-EE	1509	agent-4.4.0-EE-1509	7/28/10 8:11 PM	12	139	874.0	2
vmc-ssrc-2k810.s2qa.com	10.150.29.72	2177	4.4.0-EE	1509	agent-4.4.0-EE-1509	7/28/10 8:17 PM	10	136	153.0	2
DSL-10.16.16.142	10.16.16.142	2144	4.3.0-EE	1381	agent-4.3.0-EE-1381	7/29/10 2:02 PM	1	45	?	1
DSL-10.16.17.9	10.16.17.9	2144	4.3.0-EE	1381	agent-4.3.0-EE-1381	7/29/10 2:26 PM	1	45	?	1
DSL-10.16.17.44	10.16.17.44	2144	4.3.0-EE	1381	agent-4.3.0-EE-1381	7/29/10 2:40 PM	1	45	?	1
DSL-10.16.17.35	10.16.17.35	2144	4.3.0-EE	1381	agent-4.3.0-EE-1381	7/29/10 2:45 PM	1	45	?	1
vmc-ssrc-2k332.s2qa.com	10.150.29.204	2175	4.4.0	1509	agent-4.4.0-1509	7/30/10 12:00 PM	1	87	646.0	1
vmc-ssrc-2k816.s2qa.com	10.150.29.103	2175	4.4.0-EE	1509	agent-4.4.0-EE-1509	7/30/10 1:51 PM	1	35	684.0	1
vmc-ssrc-rh17.eng.vmware.com	10.150.29.94	2175	4.4.0-EE	1464	agent-4.4.0-EE-1464	8/16/10 3:54 PM	1	82	2.0	1
hubble.eng.vmware.com	10.17.143.3	3309	4.3.0-EE	1433	agent-4.3.0-EE-1433	9/16/10 4:49 PM	1	95	?	1
vmlin-was-005.intranet.hyperic.net	10.0.0.124	2175	4.4.0-EE	1509	agent-4.4.0-EE-1509	9/22/10 12:46 PM	1	176	995.0	1
vmwin-was-02.intranet.hyperic.net	10.0.0.233	2233	4.4.0-EE	1509	agent-4.4.0-EE-1509	9/22/10 3:36 PM	1	58	33.0	1

Health Data for an Agent

The table below defines the information on the **Agents** tab of **HQ Health**,

Field	Description	Notes
FQDN	Fully-qualified domain name of the platform where the agent runs.	
Address	IP address of the platform where the agent runs.	
Port	Port where the agent listens for communication with the Hyperic Server.	If you configure unidirectional agent - server communications, the agent initiates all communications with the Hyperic Server.
Version	Agent version number.	Although an agent might work successfully with an Hyperic Server of a later version, it is strongly recommended that you run the same version of the agent and server.
Build #	Agent build number	
Bundle Version	Agent bundle version	
Creation Time	The date/time that the Hyperic Agent was first started up.	
# Platforms	Number of platforms the agent manages.	Typically, an agent manages one platform - the platform where it runs. Exceptions include: • If the agent manages a vSphere vCenter instance, the number of platforms shown is the number of VMs the vCenter server manages. • If the agent manages remote network devices or network host platform types.
# Metrics	The number of resource metrics the agent collects. This is the total number of metrics that are configured for collection across all resources the agent monitors.	If one agent bears an inordinate metric load, you may be able to distribute it more evenly. See the "Overloaded Agent" section of *Troubleshoot Agent and Server Problems* in the *Getting Started with vFabric Hyperic Guide*.

Time Offset (ms)	The difference in system clock time between the agent and the Hyperic Server.	A time offset can cause incorrect availability reporting. See the "Out-of-Sync Agent and Server Clocks" section of *Troubleshoot Agent and Server Problems* in the *Getting Started with vFabric Hyperic Guide*. A question mark in this column indicates that the Hyperic Server is unable to contact the agent.
License Count	The number of platform licenses consumed by the agent.	Typically, a single agent consumes a single license. If an agent manages a vSphere vCenter instance, it consumes a license for the platform that hosts vCenter, a license for each vSphere vHost administered by the vCenter instance, and — if an agent is installed in each VM --- a license for each vSphere VM on each vHost.

View Hyperic Agent Metrics

The Hyperic Agent monitors itself. You can tailor the metric collection settings for an Hyperic Agent, use agent metrics to troubleshoot problems, and base alerts on agent metrics or events. This page describes the metrics and monitoring views for an Hyperic Agent.

For information about metrics that indicate Hyperic Agent problems, see *Troubleshoot Agent and Server Problems* in the *Getting Started with vFabric Hyperic Guide*.

Agent Monitoring Defaults

The metrics that the agent reports for itself by default are:

- Availability
- JVM Free Memory
- JVM Total Memory
- Number of Metrics Collected Per Minute
- Number of Metrics Sent to the Server Per Minute
- Server Offset
- Total Time Spend Fetching Metrics per Minute

See the Hyperic Agent Metrics section for a list of all supported Hyperic Agent metrics.

View Agent Indicators Charts

The **Indicators** page for an Hyperic Agent charts the agent's indicator metrics, by default:

- JVM Free Memory
- JVM Total Memory
- Number of Metrics Collected Per Minute

To view the **Indicators** page for an Hyperic Agent:

1. Click **Resources > Browse**.
2. Click **Servers**.
3. Select **HQ Agent** from **the Server Type** pull-down.

View Agent Metric Data

The Metric Data page for an Hyperic Agent displays all of the metrics collected for the agent in tabular form.

To view the Metric Data page for an Hyperic Agent:

1. Click **Resources > Browse**.
2. Click **Servers**.
3. Select **HQ Agent** from the **Server Type** pull-down.

Hyperic Agent Metrics

The table lists all of the metrics that can be collected for an Hyperic Agent. For information about using agent metrics to troubleshoot problems, see *Troubleshoot Agent and Server Problems* in the *Getting Started with vFabric Hyperic Guide*.

Category	Metric	Notes
Availability		
	Availability	Collected by default.
	Start Time	
	Up Time	
Throughput		
	Number of Active Threads	
	Number of Metrics Collected	
	Number of Metrics Collected per Minute	By default, this is an indicator metric.
	Number of Metrics which Failed to be Collecte	
	Number of Metrics which Failed to be Collected per Minute	
	Number of Requests Served	

	Number of Requests Served per Minute	
	Number of Scheduled Metrics	
Performance		
	Maximum Time Spent Fetching a Metric	
	Maximum Time Spent Processing a Request	
	Minimum Time Spent Fetching a Metric	
	Minimum Time Spent Processing a Request	
	Number of Connection Failures	
	Number of Connection Failures per Minute	
	Number of Metric Batches Sent to Server	
	Number of Metric Batches Sent to Server per Minute	
	Number of Metrics Sent to Server	
	Number of Metrics Sent to Server per Minute	Collected by default.
	Server Offset	Collected by default.
	Total Time Spent Fetching Metrics	
	Total Time Spent Fetching Metrics per Minute	Collected by default. High value can indicate overloaded agent or problem with scheduling thread.
	Total Time Spent Processing Requests	
	Total Time Spent Processing Requests per Minute	
	Total Time Spent Sending Metrics to Server	
	Total Time Spent Sending Metrics to Server per Minute	
Utilization		
	Cpu Total Time	
	Cpu Total Time per Minute	
	JVM Free Memory	By default, this is an indicator metric.
	JVM Total Memory	By default, this is an indicator metric.
	Open File Descriptors	

	Resident Memory Used	"Resident Memory" is the amount of memory the Hyperic Agent occupies in memory.
	Time Spent in System Mode	
	Time Spent in System Mode per Minute	
	Time Spent in User Mode	
	Time Spent in User Mode per Minute	
	Total Memory Used	

Reduce Agent Memory Footprint

This page describes options for reducing the amount of memory the Hyperic Agent uses.

Limit Plugin Loading

The best way to reduce an agent's footprint is to configure it to load only the plugins for the resource types you want to monitor. See Configure Plugin Loading for instructions.

Reduce Java Heap

To reduce the Java heap size that an Hyperic Agent allocates for itself on startup, add the `agent.javaOpts` property to the agent's `agent.properties` file. This property does not exist in `agent.properties` - the default behavior is equivalent to the setting shown below. You can reduce the heap from 128m to 64m.

agent.javaOpts

Couldn't find a page to include called: agent.javaOpts

Delete Javadocs Folder

In an environment where every MB is critical, you can delete the agent's javadocs folder, `agent-4.x.x/bundles/agent-4.x.x-yyyy/pdk/javadoc`; note however that this reduces the agent footprint by only (approximately) 70 MB.

Agent Properties

The properties supported in the `agent.properties` file are defined below. Note that not all supported properties appear in the default `agent.properties` file. To use a property that is not defined in `agent.properties`, you must add it explicitly.

For more information, see About Agent Configuration.

Looking for a property that is not listed here?
If your `agent.properties` file contains a property not listed here, please add a comment to this page. One explanation, although uncommon, is the use of special agent properties to override the descriptor-defined value for a `<property>` for resource instances on a particular platform. Such properties have names that reflect a resource type attribute and value. See *Overriding the Value of a Resource property at Platform Level* in *vFabric Hyperic Product Plug-in Development*.

agent.eventReportBatchSize

Description

The maximum number of events that the Hyperic Agent will send per contact with the server.

Default

As installed, agent.properties does not contain a line that defines the value of this property. The default behavior of the agent is to send a maximum of 100 events per contact with the server.

agent.keystore.alias

Description

For agents set up for unidirectional communication with the Hyperic Server, the `agent.keystore.alias` property configures the name of the user-managed keystore for the agent. By default, the agent looks for keystore named "hq". For unidirectional agents with user-managed keystores, you must define the keystore name with `agent.keystore.alias`.

For example, given this user-managed keystore for a unidirectional agent:

```
(hq self-signed cert), Jul 27, 2011, trustedCertEntry,
Certificate fingerprint (MD5): 98:FF:B8:3D:25:74:23:68:6A:CB:0B:9C:20:88:74:CE
hq-agent, Jul 27, 2011, PrivateKeyEntry,
Certificate fingerprint (MD5): 03:09:C4:BC:20:9E:9A:32:DC:B2:E8:29:C0:3C:FE:38
```

Define the name of the keystore like this:
`agent.keystore.alias=hq-agent`

If the value of `agent.keystore.alias` does not match the keystore name, agent-server communication will fail.

About unidirectional communication
If you configure an agent for unidirectional communication, all communication with the server is initiated by the agent. You can configure unidirectional communication at first agent startup, or with the `agent.setup.unidirectional` property in `agent.properties`.
Related topics:

- About Agent - Server Communication
- Hyperic Security Features and Recommendations

Default
hq

agent.keystore.password

Description

This property configures the password for a Hyperic Agent's SSL keystore. Define the location of the keystore using the `agent.keystore.path` property.

These values of `agent.keystore.path` and `agent.keystore.password` can only be supplied by defining them in `agent.properties`.

Starting in Hyper 4.6.6, the first time you start the Hyperic Agent after installation, if `agent.keystore.password` is uncommented and has a plain text value, the agent will automatically encrypt the property value. If you prefer, you can encrypt these (and other, if desired) property values yourself prior to starting the agent. For more information, see Encrypt Agent Property Value.

Password Requirement for Hyperic Keystores
The Hyperic Server's keystore password and private key password **must** be the same — otherwise, the Hyperic Server's internal Tomcat-based server will be unable to start. For information about why, see http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html. Follow the same convention for a Hyperic Agent keystore — set the password for the agent keystore be the same as the agent private key,

Best Practices for Hyperic Keystores
Please see:

- *Hyperic Security Features and Recommendations*
- *Configure SSL Options*

in *Getting Started with vFabric Hyperic*

Default
none

agent.keystore.path

Description

This property configures the location of a Hyperic Agent's (version 4.6 or later) SSL keystore. Specify the full path to the keystore. Define the password for the keystore using the `agent.keystore.password` property.

The values of `agent.keystore.path` and `agent.keystore.password` can only be supplied by defining them in `agent.properties`.

Specifying keystore path on Windows
On Windows platforms, specify the path to the keystore in this form:

```
C:/Documents and Settings/Desktop/keystore
```

Best Practices for Hyperic Agent Keystore
Please see:

- *Hyperic Security Features and Recommendations*
- *Configure SSL Options*

in *Getting Started with vFabric Hyperic*

Default
`AgentHome/data/keystore`

agent.listenIp

Description

The IP address to which the agent binds at startup. The default value allows the agent to listen on all IP addresses on the the agent host.

Default

As installed, agent.properties does not contain a line that defines the value of this property. The default behavior of the agent is to listen on all IP addresses on its host. This behavior is equivalent to to setting this property to an asterisk, like this:
*

agent.listenPort

Description

The port on the agent's listen address to which the agent binds at startup.

Default

As installed, agent.properties does not contain a line that defines the value of this property. The default behavior of the agent is to listen on port 2144.

agent.logDir

Description

You can add this property to the `agent.properties` file to specify the directory where the Hyperic Agent will write its log file. Unless you specify a fully qualified path, `agent.logDir` is evaluated relative to the agent installation directory.

This property does not exist in `agent.properties` unless you explicitly add it. To change the location for the agent log file, add `agent.logDir` to `agent.properties` and enter a path relative to the agent installation directory, or if desired, a fully qualified path.

Note that the name of the agent log file is configured with the `agent.logFile` property.

Default

`agent.logDir` does not exist in `agent.properties` unless you explicitly add it. The default behavior is equivalent to this setting:

```
agent.logDir=log
```

resulting in the agent log file being written to the `AgentHome/log` directory.

agent.logFile

Description

Specifies the path and name of the agent log file.

Default

Note that in `agent.properties`, the default setting for `agent.LogFile` is made up of a variable and a string.

```
agent.logFile=${agent.logDir}\agent.log
```

- `agent.logDir` is a variable - it supplies the value of an identically named agent property. By default, the value of `agent.logDir` is `log`, interpreted relative to the agent installation directory.
- `agent.log` is the name for the agent log file.

So, by default, the agent log file is named `agent.log`, and is written to the `AgentHome/log` directory.

If you want the agent to the log to a different directory, you must explicitly add the agent.logDir property to `agent.properties`.

agent.logLevel

Description

Specifies the level of detail of the messages the Agent writes to the log file. Allowable values are: INFO and DEBUG.

Default

INFO

agent.logLevel.SystemErr

Description

Redirects System.err to agent.log. Commenting out this setting will cause System.err to be directed to agent.log.startup.

Default

ERROR

agent.logLevel.SystemOut

Description

Redirects System.out to agent.log. Commenting out this setting will cause System.out to be directed to agent.log.startup.

Default

INFO

agent.maxBatchSize

Description

The maximum number of metrics that the agent will send per contact with the server.

Default

As installed, agent.properties does not contain a line that defines the value of this property. The default behavior of the agent is to send a maximum of 500 per contact with the server.

agent.proxyHost

Description

The host name or IP address of the proxy server that the Hyperic Agent must connect to first when establishing a connection to the Hyperic Server. Supported in vFabric Hyperic only, for agents configured for unidirectional communication.

Use in conjunction with `agent.proxyPort` and `agent.setup.unidirectional`.

Default

none

agent.proxyPort

Description

The port number of the proxy server that the Hyperic Agent must connect to first when establishing a connection to the Hyperic Server. Supported in vFabric Hyperic only, for agents configured for unidirectional communication.

Use in conjunction with `agent.proxyHost` and `agent.setup.unidirectional`.

Default

none

agent.storageProvider.info

Description

Configuration for data storage on the Agent side

Default

As installed, agent.properties does not contain a line that defines the value of this property. The default setting of the agent is

```
agent.storageProvider.info=$\{agent.dataDir\}|m|100|20|50
```

Which means, use the data directory to store the disklists, max size 100MB. The 20 and 50 numbers are used for purging data; check to see if the list can be shortened when the size is greater than 20MB and the list is 50% empty

agent.setup.acceptUnverifiedCertificate

Description

This property controls whether or not a Hyperic Agent (version 4.6 or later) issues a warning when the Hyperic Server presents an SSL certificate that is not in the agent's keystore and is either self-signed or signed by a different CA than the one that signed the the agent's SSL certificate.

Under these circumstances, if `agent.setup.acceptUnverifiedCertificate=false`, as it is by default, the agent issues this warning:

```
The authenticity of host 'localhost' can't be established.
Are you sure you want to continue connecting? [default=no]:
```

If you respond "yes", the agent imports the server's certificate, and will trust it henceforth.

Note that if `agent.setup.acceptUnverifiedCertificate` is "true", the agent automatically accepts and imports the certificate presented by the Hyperic Server, and does not issue a warning that the certificate is not trusted.

For more information, see *Hyperic Security Features and Recommendations* in *Getting Started with vFabric Hyperic*.

Default

```
agent.setup.acceptUnverifiedCertificate=false
```

agent.setup.camIP

Description

You can use this property to define for the agent the IP address of the Hyperic Server. The Hyperic Agent reads this value only in the event that it cannot find connection configuration in its `data` directory. Specifying this and other `agent.setup.*` properties is a way to reduce the user interaction required to configure an agent to communicate with the server.

The value can be provided as an IP address or a fully qualified domain name. To identify an server on the same host as the server, set the value to 127.0.0.1.

If there is a firewall between the agent and server, specify the address of the firewall, and configure the firewall to forward traffic on port 7080, or 7443 if you use the SSL port, to the Hyperic Server.

Default

Commented out, localhost.

agent.setup.camLogin

Description

You can use this property to define for the Hyperic Agent, at first startup after installation, the Hyperic username to use when registering itself with the server. The permission required on the server for this initialization is Create, for Platforms.

A login from the agent to the server is only required during the initial configuration of the agent.

The agent reads this value only in the event that it cannot find connection configuration in its `data` directory. Specifying this and other `agent.setup.*` properties is a way to reduce the user interaction required to configure an agent to communicate with the server.

Default

Commented out, hqadmin.

agent.setup.camPort

Description

You can use this property to define for a Hyperic Agent, at first startup after installation, what server port to use for non-secure communications with the server. The agent reads this value only in the event that it cannot find connection configuration in its `data` directory. Specifying this and other `agent.setup.*` properties is a way to reduce the user interaction required to configure an agent to communicate with the server.

Default

Commented out, 7080.

agent.setup.camPword

Description

You can use this property to define the password that the Hyperic Agent will use when connecting to the Hyperic Server, so that the agent will not prompt for the user to supply the password interactively at first startup. (This is the password for the user specified by `agent.setup.camLogin`.

The agent reads this value only in the event that it cannot find connection configuration in its `/data` directory. Specifying this and other `agent.setup.*` properties is a way to reduce the user interaction required to configure an agent to communicate with the server.

Starting in Hyper 4.6.6, the first time you start the Hyperic Agent after installation, if `agent.keystore.password` is uncommented and has a plain text value, the agent will automatically encrypt the property value. If you prefer, you can encrypt these (and other, if desired) property values yourself prior to starting the agent. For more information, see Encrypt Agent Property Value.

Default

Commented out, `hqadmin`

agent.setup.camSecure

Description

You can use this property to define for the agent, at first startup after installation, whether to communicate with the server over SSL. If you set this property to yes, all agent-server communications will be use the SSL secure port.

If acceptable in your environment, non-SSL communication offers improved performance for agent-server communications.

The agent reads this value only in the event that it cannot find connection configuration in its data directory. Specifying this and other agent.setup.* properties is a way to reduce the user interaction required to configure an agent to communicate with the server.

Default

Commented out, value of yes.

agent.setup.camSSLPort

Description

You can use this property to define for the Hyperic Agent, at first startup after installation, what server port to use for SSL communications with the Hyperic Server. The agent reads this value only in the event that it cannot find connection configuration in its `data` directory. Specifying this and other `agent.setup.*` properties is a way to reduce the user interaction required to configure an agent to communicate with the server.

Default

Commented out, 7443.

agent.setup.agentIP

Description

This specifies the IP address that the Hyperic Server will use to contact the Hyperic Agent. If the agent is on the same host as the server, value of 127.0.0.1 is valid.

If there is a firewall between the server and agent, specify the IP address of the firewall, and configure the firewall to forward traffic intended for the agent to the agent's listen address, which can be configured with `agent.listenIP`.

The agent reads this value only in the event that it cannot find connection configuration in its `data` directory. Specifying this and other `agent.setup.*` properties is a way to reduce the user interaction required to configure an agent to communicate with the server.

Default

As installed, `agent.properties` contains a commented out statement that sets the value to **default**. If you use the `agent.setup.*` properties to supply an agent's configuration at first startup, and uncomment this property and leave the value `default`, the Hyperic Server will contact the agent using the IP address that SIGAR detects on the agent host.

agent.setup.agentPort

Description

This specifies the port (on the IP address configured with `agent.setup.agentIP`) on the Hyperic Agent on which the Hyperic Server will communicate with the agent.

If there is a firewall between the agent and the server, set `agent.setup.agentPort` to the appropriate port on the firewall, and configure the firewall to forward traffic intended for the agent to the agent listen port, which can be configured with.

The agent reads this value only in the event that it cannot find its connection configuration in its `data` directory. Specifying this and other `agent.setup.*` properties is a way to reduce the user interaction required to configure an agent to communicate with the server.

Default

As installed, `agent.properties` contains a commented out statement that sets the value to *default*. If you use the `agent.setup.*` properties to supply an agent's configuration at first startup, and uncomment this property and leave the value *default*, the Hyperic Server will contact the agent on port 2144, unless SIGAR detects it is not available, in which case another default is selected.

agent.setup.resetupToken

Description

You can use this property to configure a Hyperic Agent to create a new token to use to authenticate with the server at startup. The agent reads this value only in the event that it cannot find connection configuration in its data directory. Regenerating a token is useful if the Agent cannot connect to the server because the token has been deleted or corrupted.

Regardless of the value of this property, an agent will generate a token the first time it is started after installation.

Default

As installed, `agent.properties` contains a commented out statement that sets the value to "no".

agent.setup.unidirectional

Description

Enables the unidirectional communications between the Hyperic Agent and Hyperic Server in vFabric Hyperic. For more information, see Configure Unidirectional Agent - Server Communication.

Note that a for a unidirectional agent with a user-managed keystore, you must configure the keystore name in `agent.properties`. See `agent.keystore.alias`.

About unidirectional communication
If you configure an agent for unidirectional communication, all communication with the server is initiated by the agent. You can configure unidirectional communication at first agent startup, or with the `agent.setup.unidirectional` property in `agent.properties`.
Related topics:

- *About Agent - Server Communication*
- *Hyperic Security Features and Recommendations*

in *Getting Started with vFabric Hyperic*.

Default

Commented out, defaults to no.

agent.startupTimeOut

Description

The number of seconds that the agent startup script will wait before determining that the agent did not startup successfully. If the agent is not determined to be listening for requests within this period of time, an error is logged, and the startup script times out.

Default

As installed, agent.properties does not contain a line that defines the value of this property. The default behavior of the agent is to timeout after 300 seconds.

autoinventory.defaultScan.interval.millis

Description

Specifies how frequently the agent performs a default autoinventory scan.

The default scan detects servers and platform services, typically using the process table or the Windows registry. Default scans are less resource-intensive than runtime scans.

Default

Commented out, set to 86,400,000 milliseconds, or 1 day.

Note however, that by default, the agent performs the default scan at startup and every 15 minutes thereafter.

autoinventory.runtimeScan.interval.millis

Description

Specifies how frequently the agent performs a runtime scan.

A runtime scan may use more resource-intensive methods to detect services than a default scan. For instance, a runtime scan may involve issuing an SQL query or looking up an MBean.

Default

86,400,000 milliseconds, or 1 day.

http.useragent

Description

The `http.useragent` property defines the value for the User-Agent request header in HTTP requests issued by the Hyperic Agent. By default, the User-Agent in agent requests includes the Hyperic Agent version - and so change upon agent upgrade. If a target HTTP server is configured to block requests with an unknown User-Agent, agent requests will fail after agent upgrade.

You can use `http.useragent` to define a User-Agent value that will be consistent across upgrades.

Note: `agent.properties` does not contain this property by default. You must explicitly add it to the file.

Default

`Hyperic-HQ-Agent/Version`

For example:

`Hyperic-HQ-Agent/4.1.2-EE`

log4j Properties

```
log4j.rootLogger=${agent.logLevel}, R

log4j.appender.R.File=${agent.logFile}
log4j.appender.R.MaxBackupIndex=1
log4j.appender.R.MaxFileSize=5000KB
log4j.appender.R.layout.ConversionPattern=%d{dd-MM-yyyy HH:mm:ss,SSS z} %-5p [%t]
[%c{1}@%L] %m%n
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R=org.apache.log4j.RollingFileAppender

##
## Disable overly verbose logging
##
log4j.logger.org.apache.http=ERROR
log4j.logger.org.springframework.web.client.RestTemplate=ERROR
log4j.logger.org.hyperic.hq.measurement.agent.server.SenderThread=INFO
log4j.logger.org.hyperic.hq.agent.server.AgentDListProvider=INFO
log4j.logger.org.hyperic.hq.agent.server.MeasurementSchedule=INFO
log4j.logger.org.hyperic.util.units=INFO
log4j.logger.org.hyperic.hq.product.pluginxml=INFO

# Only log errors from naming context
log4j.category.org.jnp.interfaces.NamingContext=ERROR
log4j.category.org.apache.axis=ERROR

#Agent Subsystems: Uncomment individual subsystems to see debug messages.
#-------------------------------------------------------------------------
#log4j.logger.org.hyperic.hq.autoinventory=DEBUG
#log4j.logger.org.hyperic.hq.livedata=DEBUG
#log4j.logger.org.hyperic.hq.measurement=DEBUG
#log4j.logger.org.hyperic.hq.control=DEBUG

#Agent Plugin Implementations
#log4j.logger.org.hyperic.hq.product=DEBUG

#Server Communication
```

```
#log4j.logger.org.hyperic.hq.bizapp.client.AgentCallbackClient=DEBUG

#Server Realtime commands dispatcher
#log4j.logger.org.hyperic.hq.agent.server.CommandDispatcher=DEBUG

#Agent Configuration parser
#log4j.logger.org.hyperic.hq.agent.AgentConfig=DEBUG

#Agent plugins loader
#log4j.logger.org.hyperic.util.PluginLoader=DEBUG

#Agent Metrics Scheduler (Scheduling tasks definitions & executions)
#log4j.logger.org.hyperic.hq.agent.server.session.AgentSynchronizer.SchedulerThread
=DEBUG

#Agent Plugin Managers
#log4j.logger.org.hyperic.hq.product.MeasurementPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.AutoinventoryPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.ConfigTrackPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.LogTrackPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.LiveDataPluginManager=DEBUG
#log4j.logger.org.hyperic.hq.product.ControlPluginManager=DEBUG
```

jboss.installpath

Description

To enable the agent to monitor JBoss, specify the location of the JBoss root directory.

Default

/usr/local/jboss-4.0.0

platform.log_track.eventfmt

Description

Specifies the content and format of the Windows event attributes that a Hyperic Agent includes when logging a Windows event as an event in Hyperic. `agent.properties` does not contain the `platform.log_track.eventfmt` property, you must explicitly add it if you want to tailor the data logged for Windows events.

Default Behavior

When Windows log tracking is enabled, an entry of this form is logged for events that match the criteria you specified on the resource's **Configuration Properties** page:

[Timestamp] Log Message (EventLogName):EventLogName:EventAttributes

where:

- ***Timestamp*** - is when the event occurred
- **Log Message** - is an text string
- ***EventLogName*** - is the Windows event log type, "System", "Security", or "Application".
- ***EventAttributes*** - a colon delimited string made of the Windows event **Source** and **Message** attributes.

For example, this log entry:

04/19/2010 06:06 AM Log Message (SYSTEM): SYSTEM: Print: Printer HP LaserJet 6P was paused.

is for an Windows event written to the Windows System event log at 6:06 AM on 04/19/2010. The Windows event **Source** and **Message** attributes, are "Print" and "Printer HP LaserJet 6P was paused.", respectively.

Configuration

You can use the parameters below to configure the Windows event attributes that the agent writes for a Windows event. Each parameter maps to Windows event attribute of the same name.

- `%user%` — The name of the user on whose behalf the event occurred.
- `%computer%` — The name of the computer on which the event occurred.
- `%source%` — The software that logged the Windows event.
- `%event%` — A number identifying the particular event type.
- `%message%` — The event message.
- `%category%` — An application-specific value used for grouping events.

For example, with this property setting:

```
platform.log_track.eventfmt=%user%@%computer%
%source%:%event%:%message%
```

the Hyperic Agent will write the following data when logging Windows event:

```
04/19/2010 06:06 AM Log Message (SYSTEM): SYSTEM: HP_Admistrator@Office
Print:7:Printer HP LaserJet 6P was paused.
```

This entry is for as for an Windows event written to the Windows System event log at 6:06 AM on 04/19/2010. The software associated with the event was running as "HP_Administrator" on the host "Office". The Windows event's **Source, Event,** and **Message** attributes, are "Print", "7", and "Printer HP LaserJet 6P was paused.", respectively.

plugins.exclude

Description

Use this property to specify plugins that you do not wish the Hyperic Agent to load at startup. This is useful for reducing the agent's memory footprint.

Usage

Supply a comma-separated list of plugins to exclude, for example:

```
plugins.exclude=jboss,apache,mysql
```

plugins.include

Description

Use this property to specify plugins that you do wish the Hyperic Agent to load at startup. This is useful for reducing the agent's memory footprint.

Usage

Supply a comma-separated list of plugins to include, for example:

```
plugins.include=weblogic,apache
```

postgresql.database.name.format

Description

This property specifies the format of the name that the PostgreSQL plugin (in Hyperic 5.0 and later) assigns to auto-discovered databases of types `PostgreSQL Database` and `vPostgreSQL Database`.

By default, the name of a PostgreSQL or vPostegreSQL database is:

`Database` *`DatabaseName`*

where
`DatabaseName` is the auto-discovered name of the database.

If you want to use a different naming convention, define `postgresql.database.name.format` Note that variable data you wish to use must be available from the PostgreSQL plugin. For information about the resource properties and options that the PostreSQL plugin defines, see PostgreSQL, in *vFabric Resource Configuration and Metrics*.

This property does not exist in `agent.properties` by default. You must add `postgresql.database.name.format` to the properties file if you wish to define it.

Default
This is the syntax you would use to specify the default table name assigned by the plugin.

```
Database ${db}
```

where:
`postgresql.db` — is the auto-discovered name of the `PostgreSQL` or `vPostgreSQL` database.

postgresql.index.name.format

Description
This property specifies the format of the name that the PostgreSQL plugin (in Hyperic 5.0 and later) assigns to auto-discovered database indexes of types `PostgreSQL Index` and `vPostgreSQL Index`.

By default, the name of a PostgreSQL or vPostegreSQL index is:

`Index` *`DatabaseName.Schema.Index`*

where:

- *`DatabaseName`* is the auto-discovered name of the database.
- *`Schema`* is the auto-discovered schema for the database.
- *`Index`* is the auto-discovered name of the table.

If you want to use a different naming convention, define `postgresql.index.name.format`. Note that variable data you wish to use must be available from the PostgreSQL plugin. For information about the resource properties and options that the PostreSQL plugin defines, see *PostgreSQL*, in *vFabric Resource Configuration and Metrics*.

This property does not exist in `agent.properties` by default. You must add `postgresql.table.name.format` to the properties file if you wish to define it.

Default

This is the syntax you would use to specify the default index name assigned by the plugin.

```
Index ${db}.${schema}.${index}
```

where:

- `db` — identifies the platform that hosts the `PostgreSQL` or `vPostgreSQL` server.
- `schema` — identifies the schema associates with the table.
- `table` — the table name in PostgreSQL.

postgresql.server.name.format

Description

This property specifies the format of the name that the PostgreSQL plugin (in Hyperic 5.0 and later) assigns to auto-discovered servers of types `PostgreSQL` and `vPostgreSQL`.

By default, the name of a PostgreSQL or vPostegreSQL server is:

`Host:Port`

where:

- *`Host`* is the FQDN of the platform that hosts the server.
- *`Port`* is the PostgreSQL listen port.

If you want to use a different naming convention for those PostgreSQL or vPostegreSQL servers, define `postgresql.server.name.format`. Note that variable data you wish to use must be available from the PostgreSQL plugin. For information about the resource properties and options that the PostreSQL plugin defines, see *PostgreSQL*, in *vFabric Resource Configuration and Metrics*.

This property does not exist in `agent.properties` by default. You must add `postgresql.server.name.format` to the properties file if you wish to define it.

Default

This is the syntax you would use to specify the default server name assigned by the plugin.

```
${postgresql.host}:${postgresql.port}
```

where:

- `postgresql.host` — is the FQDN of the hosting platform.
- `postgresql.port` — is the database listen port.

postgresql.table.name.format

Description

This property specifies the format of the name that the PostgreSQL plugin (in Hyperic 5.0 and later) assigns to auto-discovered tables of types `PostgreSQL Table` and `vPostgreSQL Table`.

By default, the name of a PostgreSQL or vPostegreSQL table is:

`Table` *`DatabaseName.Schema.Table`*

where:

- *`DatabaseName`* is the auto-discovered name of the database.
- *`Schema`* is the auto-discovered schema for the database.
- *`Table`* is the auto-discovered name of the table.

If you want to use a different naming convention, define `postgresql.table.name.format`. Note that variable data you wish to use must be available from the PostgreSQL plugin. For information about the resource properties and options that the PostreSQL plugin defines, see *PostgreSQL*, in *vFabric Resource Configuration and Metrics*.

This property does not exist in `agent.properties` by default. You must add `postgresql.table.name.format` to the properties file if you wish to define it.

Default

This is the syntax you would use to specify the default table name assigned by the plugin.

```
Table ${db}.${schema}.${table}
```

where:

- `db` — identifies the platform that hosts the `PostgreSQL` or `vPostgreSQL` server.
- `schema` — identifies the schema associates with the table.
- `table` — the table name in PostgreSQL.

scheduleThread.cancelTimeout

Description

The maximum time, in milliseconds, the `ScheduleThread` will allow a metric collection process to to run before attempting to interrupt it. When the timeout is exceeded, collection of the metric will be interrupted, if it is an interruptible state, that is, in a `wait()`, `sleep()` or non-blocking `read()` state.

Default is 5000.

Usage

```
scheduleThread.cancelTimeout=5000
```

scheduleThread.fetchLogTimeout

Description

The property controls when a warning message is issued for a long-running metric collection process. If a metric collection process exceeds the value of this property, measured in milliseconds, the agent writes a warning message to the `agent.log` file.

Default is 2000.

Usage

```
scheduleThread.fetchLogTimeout=2000
```

scheduleThread.poolsize

Description

This property allows a plugin to use multiple threads for metric collection. This property may increase metric throughput for plugins known to be thread safe.

Default is 1.

Usage

Specify the plugin by name and the number of threads to allocate for metric collection:

```
scheduleThread.poolsize.PluginName=2
```

where
`PluginName` is the name of the plugin to which you are allocating threads.

for example:

```
scheduleThread.poolsize.vsphere=2
```

scheduleThread.queuesize

Description

This property can be used to limit the metric collection queue size (the number of metrics) for a plugin.

Default is 10000.

Usage

Specify the plugin by name and the number of maximum metric queue length:

```
scheduleThread.queuesize.PluginName=15000
```

where
`PluginName` is the name of the plugin upon which you are imposing a metric limit.

for example:

```
scheduleThread.queuesize.vsphere=15000
```

sigar.mirror.procnet

Description

mirror /proc/net/tcp on linux

Default

true

snmpTrapReceiver.listenAddress

Description
Use this property to specify the port on which the Hyperic Agent listens for SNMP traps. By default, the agent is not configured to listen for SNMP traps. You must add this property to `agent.properties` to enable the agent to receive traps.

Typically SNMP uses the UDP port 162 for trap messages. This port is in the privileged range, so an agent listening for trap messages on it must run as root (or as an Administrative user on Windows).

If you prefer to run the the agent under the context of a non-administrative user, you can configure it to listen for trap messages on an unprivileged port.

Usage
Specify an IP address (or 0.0.0.0 to specify all interfaces on the platform) and the port for UDP communications in this format:

```
snmpTrapReceiver.listenAddress=udp:IP_address/port
```

To enable the Hyperic Agent to receive SNMP traps on an unprivileged port, specify port 1024 or above. The following setting allows the agent to receive traps on any interface on the platform, on UDP port 1620.

```
snmpTrapReceiver.listenAddress=udp:0.0.0.0/1620
```

weblogic.auth.method

Description

`weblogic.auth.method` is one of four properties you define to enable an Hyperic Agent to communicate with a WebLogic Administration Server using Two-Way-SSL.

Add the following line to the `agent.properties` file to specify that the agent will use Two-Way-SSL for communications with the Administration Server.

```
weblogic.auth.method=ssl2ways
```

Default

None.

weblogic.discovery.new

Description

This property controls how WebLogic Server Administration Servers and Managed Servers. Use of this property is not typically required. Define this property only if recommended by Hyperic Support.

Default

By default, this property does not exist in `agent.properties`.

weblogic.installpath

Description

To enable the agent to monitor WebLogic 8.1, specify the location server/lib/weblogic.jar

Default

/usr/local/bea/weblogic-8.1

weblogic.ssl2ways.cert

Description

`weblogic.ssl2ways.cert` is one of four properties you define to enable an Hyperic Agent to communicate with a WebLogic Administration Server using Two-Way-SSL.

Add `weblogic.ssl2ways.cert` to the `agent.properties` file and set its value to the location of the client certificate that the Hyperic Agent will present to the Administration Server:

```
weblogic.ssl2ways.cert=Client2Cert.pem
```

where *Client2Cert.pem* is the **path to the client certificate** the Hyperic Agent presents to the Administration Server it manages.

Default

None.

weblogic.ssl2ways.key

Description

`weblogic.ssl2ways.key` is one of four properties you define to enable a Hyperic Agent to communicate with a WebLogic Administration Server using Two-Way-SSL.

`weblogic.ssl2ways.key` to the `agent.properties` file and set its value to the location of client's private key:

`weblogic.ssl2ways.key=clientKey.pem`

where:

clientkey.pem is the **path to the private key** the Hyperic Agent presents to the Administration Server that the agent manages.

Default

None.

weblogic.ssl2ways.key.pass

Description

`weblogic.ssl2ways.key.pass` is one of four properties you define to enable an Hyperic Agent to communicate with a WebLogic Administration Server using Two-Way-SSL.

Add `weblogic.ssl2ways.key.pass` to the `agent.properties` file and set its value to the passphrase for the client private key:

weblogic.ssl2ways.key.pass=*ClientKey*

where *ClientKey* is the passphrase for the client private key.

Default

None.

websphere.installpath

Description

To enable the agent to monitor WebSphere, specify the location of the WebSphere jars.

Default

/opt/WebSphere/AppServer

websphere.useext

Description

This property is required to enable management of WebSphere 6.0 and 6.1.

Do **not** define the `websphere.useext` property to monitor WebSphere 7.

Usage

Add the following property definition to the `agent.properties file` for a Hyperic Agent that will manage WebSphere 6.0 or 6.1.

```
websphere.useext=true
```

Agent Java Service Wrapper Configuration

This section has information about the configuration file for the Hyperic Agent's Java Service Wrapper. The file is `AgentHome/bundles/BundleHome/conf/wrapper.conf`.

Listing of the Agent Wrapper Configuration File

```
#********************************************************************
# Java Service Wrapper Properties for Hyperic HQ Agent
#********************************************************************

# default JAVA_HOME in case it is not already set
set.default.HQ_JAVA_HOME=../../jre

# Java Application
wrapper.java.command=%HQ_JAVA_HOME%/bin/java

# Java Main class.  This class must implement the WrapperListener interface
#  or guarantee that the WrapperManager class is initialized.  Helper
#  classes are provided to do this for you.  See the Integration section
#  of the documentation for details.
wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperStartStopApp

# Java Classpath (include wrapper.jar)  Add class path elements as
#  needed starting from 1
wrapper.java.classpath.1=../../bundles/%HQ_AGENT_BUNDLE%/lib/*.jar
wrapper.java.classpath.2=../../bundles/%HQ_AGENT_BUNDLE%/pdk/lib/*.jar
wrapper.java.classpath.3=../../wrapper/lib/*.jar
wrapper.java.classpath.4=../../bundles/%HQ_AGENT_BUNDLE%/lib
wrapper.java.classpath.5=../../bundles/%HQ_AGENT_BUNDLE%/pdk/lib/jdbc/*.jar
wrapper.java.classpath.6=../../bundles/%HQ_AGENT_BUNDLE%/pdk/lib/mx4j/*.jar

# Java Library Path (location of Wrapper.DLL or libwrapper.so)
wrapper.java.library.path.1=%LD_LIBRARY_PATH%
wrapper.java.library.path.2=../../wrapper/lib
```

```
# Java Additional Parameters
wrapper.java.additional.1=-
Djava.security.auth.login.config=../../bundles/%HQ_AGENT_BUNDLE%/jaas.config
wrapper.java.additional.2=-Xmx128m
wrapper.java.additional.3=-Djava.net.preferIPv4Stack=true
wrapper.java.additional.4=-Dagent.install.home=../..
wrapper.java.additional.5=-Dagent.bundle.home=../../bundles/%HQ_AGENT_BUNDLE%
wrapper.java.additional.6=-Dsun.net.inetaddr.ttl=60

# Initial Java Heap Size (in MB)
#wrapper.java.initmemory=3

# Maximum Java Heap Size (in MB)
#wrapper.java.maxmemory=64

# Application parameters.  Add parameters as needed starting from 1
#wrapper.app.parameter.1=
wrapper.app.parameter.1=org.hyperic.hq.bizapp.agent.client.AgentClient
wrapper.app.parameter.2=1
wrapper.app.parameter.3=start

# The start parameters are followed by the name of the class whose main
# method is to be called to stop the application.  The stop class name
# is followed by a flag which controls whether or not the Wrapper should
# wait for all non daemon threads to complete before exiting the JVM.
# The flag is followed by the number of parameters to be passed to the
# stop class's main method.  Finally comes the actual parameters.
wrapper.app.parameter.4=org.hyperic.hq.bizapp.agent.client.AgentClient
wrapper.app.parameter.5=true
wrapper.app.parameter.6=2
wrapper.app.parameter.7=die
wrapper.app.parameter.8=30

#*********************************************************************
# Wrapper Logging Properties
#*********************************************************************
# Format of output for the console.  (See docs for formats)
wrapper.console.format=PM

# Log Level for console output.  (See docs for log levels)
wrapper.console.loglevel=INFO

# Log file to use for wrapper output logging.
wrapper.logfile=../../log/wrapper.log

# Format of output for the log file.  (See docs for formats)
wrapper.logfile.format=LPTM

# Log Level for log file output.  (See docs for log levels)
wrapper.logfile.loglevel=INFO

# Maximum size that the log file will be allowed to grow to before
```

```
#  the log is rolled. Size is specified in bytes.  The default value
#  of 0, disables log rolling.  May abbreviate with the 'k' (kb) or
#  'm' (mb) suffix.  For example: 10m = 10 megabytes.
wrapper.logfile.maxsize=0

# Maximum number of rolled log files which will be allowed before old
#  files are deleted.  The default value of 0 implies no limit.
wrapper.logfile.maxfiles=0

# Log Level for sys/event log output.  (See docs for log levels)
wrapper.syslog.loglevel=NONE

#*****************************************************************
# Wrapper Windows Properties
#*****************************************************************
# Title to use when running as a console
wrapper.console.title=Hyperic HQ Agent

#*****************************************************************
# Wrapper Windows NT/2000/XP Service Properties
#*****************************************************************
# WARNING - Do not modify any of these properties when an application
#  using this configuration file has been installed as a service.
#  Please uninstall the service before modifying this section.  The
#  service can then be reinstalled.

# Name of the service
wrapper.ntservice.name=Hyperic HQ Agent

# Display name of the service
wrapper.ntservice.displayname=Hyperic HQ Agent

# Description of the service
wrapper.ntservice.description=Agent for Hyperic HQ

# Service dependencies.  Add dependencies as needed starting from 1
wrapper.ntservice.dependency.1=

# Mode in which the service is installed.  AUTO_START or DEMAND_START
wrapper.ntservice.starttype=AUTO_START

# Allow the service to interact with the desktop.
wrapper.ntservice.interactive=false

# restart the JVM for all exit codes except the exit code 0
wrapper.on_exit.default=RESTART
wrapper.on_exit.0=SHUTDOWN

# limit the number of JVM restarts
wrapper.max_failed_invocations=5
# if running for over 60 sec assume it was successfully started
wrapper.successful_invocation_time=60
```

Define Java Options

Java options supplied to the Hyperic Agent at startup are configured in the "Java Additional Parameters" section of the `wrapper.conf`. You can edit the lines in this section to define desired Java options. For example, to set heap size 5o 256M, you would modify the `wrapper.java.additional.2` line, as shown below.

```
Java Additional Parameters
wrapper.java.additional.1=-
Djava.security.auth.login.config=../../bundles/%HQ_AGENT_BUNDLE%/jaas.config
wrapper.java.additional.2=-Xmx256m
wrapper.java.additional.3=-Djava.net.preferIPv4Stack=true
wrapper.java.additional.4=-Dagent.install.home=../..
wrapper.java.additional.5=-Dagent.bundle.home=../../bundles/%HQ_AGENT_BUNDLE%
wrapper.java.additional.6=-Dsun.net.inetaddr.ttl=60
```

Configure and Run the Hyperic Server

Start and Stop Hyperic Server

These topics have information about starting and stopping Hyperic Server:

- Starting the Server on Unix-Based Platforms
- Starting the Server on Windows To Run as a Service
- Configure Hyperic Server Java Options

Starting the Server on Unix-Based Platforms

If you installed Hyperic Server from an RPM package, see Starting Hyperic Server After Installing from RPM. Otherwise, start the server with this command:

```
ServerHome/bin/hq-server.sh start
```

The script will display some startup information on stdout, then it will detach and run in the background.

Detailed startup information is written to the `server.log` and `bootstrap.log` files in the `ServerHome/logs` director

Starting Hyperic Server After Installing from RPM

If you installed the Hyperic Server from the VMware yum repository to an RHEL VM, the Hyperic Server is configured to start automatically each time the VM starts up.

If you installed the Hyperic Server from a downloaded RPM, follow these steps to start the Hyperic Server as a daemon:

1. Log in to the Hyperic Server host as `root`.
2. Open a terminal window and run the `/etc/init.d/hyperic-hq-server` script with the `start` parameter:

   ```
   /etc/init.d/hyperic-hq-server start
   ```

   If you install Hyperic Server from the VMware yum repository, the VM where you install the server is configured to automatically start the Hyperic Server each time the VM starts up.

Starting the Server on Windows To Run as a Service

The first time you start up the server after installation, use this command to start it as a Windows Service:

```
<Server Installation directory>\bin\hq-server.bat install
```

Henceforth, use the Windows Service control panel to start and stop the server.

Configure Hyperic Server Java Options

Hyperic Server's Java options are configured with the `server.java.opts` property in `ServerHome/conf/hq-server.conf`. For more information see server.java.opts.

Configure Metric Baselining and Alert Processing Behavior

These topics have instructions for configuring Hyperic Server baselining and alert processing behaviors:

- Configure Global Alert Properties
- Configure Alert Notification Throttling
- Configure Alert Notification Email Properties
- Configure Metric Baselining Properties

Configure Global Alert Properties

- The settings in the **Global Alert Properties** section of the **Administration > HQ Server Settings** page enable immediate and global control of alert processing.
- **Alerts** - Disable or enable all alert definitions for all resources immediately. Disabling stops any alerts from firing; notifications defined in escalations that are currently in progress will be completed.
- **Alert Notifications** - Disable or enable alert notifications for all resources immediately. Disabling stops all notifications, include those for alerts with escalations currently in progress.
- **Hierarchical Alerting** - This setting controls whether alerts are evaluated using the hierarchical alerting method. When hierarchical alerting is enabled, before firing an alert for a resource, HQ considers the availability and alert status of the resource's parent. The purpose of hierarchical alerting is to avoid firing alerts for every resource affected by a single root cause. For more information, see *Manage Alert and Notification Volume*.

Note: You can extend the effect of hierarchical alerting by configuring the relationship between a network device or virtual host and the platforms that depend on it using the **Network and Host Dependency Manager** available in the "Plugins" section of the **Administration** tab. For more information see *Manage Alert and Notification Volume*.

Configure Alert Notification Throttling

You can use notification throttling to limit the number of alert email actions (notifications sent by email for a fired alert) that HQ will issue in a 15 second interval. When the threshold you specify is reached, HQ stops sending email alert notifications and instead sends a summary of alert activity every ten minutes to the recipients you specify.

After starting to throttle, HQ re-evaluates notification volume for fired alerts every 10 minutes; when it determines that the per interval volume of individual notifications that fired alerts would generate is less than the configured threshold, HQ resumes sending individual notifications.

In the **Notification Throttling Configuration Properties** section of the **Administration > HQ Server Settings** page:

1. Click the Notification Throttling ON control.
2. In the "Threshold" field, enter the maximum number of notifications you want sent in a 15 second interval.
3. Enter one or more email addresses in the "Notification Email(s) field".
4. Restart the vFabric Hyperic server.

For related information, see *Manage Alert and Notification Volume*.

Configure Alert Notification Email Properties

The settings in the **Email Configuration Properties** section of the **Administration > HQ Server Settings** are used to form notifications that Hyperic sends for a fired alert.

Property	Description

Base URL	The address:port where the Hyperic Server listens for web application requests. The initial value of **Base URL** is the web application listen port configured when the Hyperic Server was installed, for example:
	`http://Ms-MacBook-Pro-15.local:7080`
	Base URL forms the prefix of the URL to which Hyperic appends the remainder of the URL, which points to the **Alert Detail** page for the fired alert. For example:
	`http://Ms-MacBook-Pro-15.local:7080/alerts/Alerts.do?mode=viewAlert&eid=5:10611&a=16431`

From Email Address	The email address listed as the sender of the alert emails. For example: `hq@demo2.vmware.com`

Configure Metric Baselining Properties

In vFabric Hyperic, the properties in the **Automatic Baseline Configuration Properties** section of the **Administration > HQ Server Settings** page control the Hyperic baselining process and the accuracy of the baseline.

Server Setting	Description	Default
Baseline Frequency	The frequency with which HQ calculates a baseline for each metric.	3 days
Baseline Dataset	The time range of metric data used in calculating the baseline.	7 days
Baseline Minimum Data Points	The minimum number of data points used in calculating a baseline.	40
Track Out-of-Bounds Metrics	Controls whether or not HQ tracks OOB metrics].	off

Scaling and Tuning Hyperic Performance

About this section...
This section has information about tuning Hyperic Server for large deployments, including recommended values for server properties based on the number of platforms you will manage.

Sizing Profiles in vFabric Hyperic
In vFabric Hyperic the settings described below are implemented by the Hyperic installer — server property values are set based on the *sizing profile* you select when installing the Hyperic Server. For information about sizing profiles, see *About Sizing Profiles in vFabric Hyperic*. Note that you can run the Hyperic installer to change the current sizing profile for the vFabric Hyperic Server, as described in *Change vFabric Hyperic Server Sizing Profile*.

Sizing Considerations

The number of platforms the Hyperic Server can manage depends on the hardware it runs on, the number of Hyperic Agents reporting to the server, the volume of metrics that are collected, and the size of the Hyperic database.

See *vFabric Hyperic Supported Configurations and System Requirements* for Hyperic Server system requirements. Typically, a minimal system configuration will support 25 Hyperic Agents or more. On a high performance platform, a properly configured Hyperic Server can support up to 2,000 agents. There are a variety of Hyperic Server properties that govern the system resources available to the server — settings whose values should be set based on the number of platforms under management.

Server Configuration Settings for Scaling

The table lists server properties that relate to Hyperic Server scaling. The values shown in the "Small", "Medium", and "Large" columns correspond the values set for the corresponding sizing profiles in vFabric Hyperic 4.6.5. In Hyperic HQ, these properties default to the values shown in the "Small" column.

Property	Small (less than 50 platforms)	Medium (50-250 platforms)	Large (more than 250 platforms)
`server.jms.highmemory`	350	1400	2400
`server.jms.maxmemory`	400	1600	3600
`server.database-minpoolsize`	5	20	50
`server.database-maxpoolsize`	100	200	400
`server.java.opts`	`-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx512m -Xms512m -XX:+HeapDumpOnOutOfMemoryError -XX:+UseConcMarkSweepGC`	`-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx4g -Xms4g -XX:+HeapDumpOnOutOfMemoryError -XX:+UseConcMarkSweepGC`	`-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmn4g -Xmx8g -Xms8g -XX:+HeapDumpOnOutOfMemoryError -XX:+UseConcMarkSweepGC -XX:SurvivorRatio=12 -XX:+UseCompressedOops`
`tomcat.maxthreads` (new in vFabric Hyperic 4.6.5)	500	2000	4000
`tomcat.minspareth`	50	100	200

`reads` (new in vFabric Hyperic 4.6.5)			
`org.hyperic.lather.maxConns` (in `ServerHome\hq-engine\hq-server\webapps\ROOT\WEB-INF\web.xml`)	475	1900	3800

About Java Heap and Garbage Collection

Heap size startup options are set in the `server.java.opts` property. Note that how much you can increase the heap size depends on the amount of RAM on the Hyperic Server host. Given sufficient RAM, you could use these settings:

```
server.java.opts=-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx4096m -Xms4096m
-XX:+UseConcMarkSweepGC -XX:+UseCompressedOops
```

Note: If you are running Hyperic Server on a 64-bit system with 4GB (4096 MB) or less memory, Hyperic recommends you use 32-bit JVM. If you use a 64-bit JVM, be sure to set the `-XX:+UseCompressedOops` in `server.java.opts` property. with the oops option set.

About Hyperic Server Caches

Hyperic Server uses Ehcache for in-memory caching. Effective cache management is necessary for server stability and performance. Caching policies that define the cache size (maximum number of objects to cache) for each type are defined in `server-n.n.n-EE\hq-engine\hq-server\webapps\ROOT\WEB-INF\classes\ehcache.xml`. The cache size for a type depends on the on how dynamic that type is: how often it likely to be is updated. Given a fixed amount of memory, cache sizing in Hyperic tries to allocate cache according to these guidelines:

- Relatively static types — Caches for types that are not frequently updated — for instance, `Resource`, `Platform`,`Server`, and `Measurement` — are sized to keep objects in memory for the lifetime of the HQ Server. An extremely low miss rate desired. The default cache sizes (the maximum number elements in cache) configured in `ehcache.xml` for inventory types are:
  - Platforms — 2,000
  - Servers ---- 50,000
  - Services — 100,000

  This sizing should be adequate for medium to large deployments.

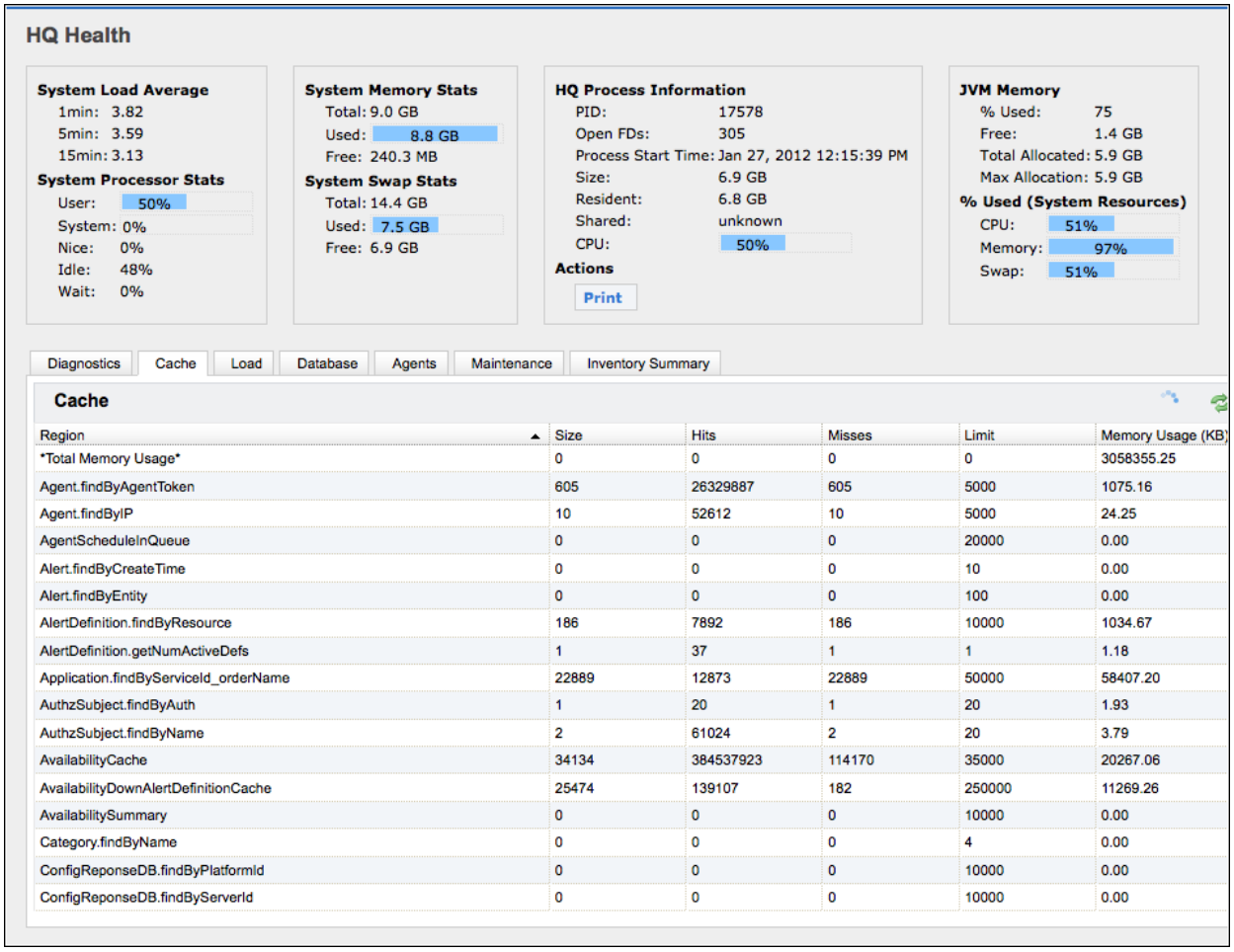
- Dynamic types — Caches for types that are frequently updated, (for instance `Alert` and `Galert`) and hence get stale sooner, are configured such that objects age out more quickly. A high hit/miss ratio of is optimal for dynamic types, in larger environments, on the order of 2:1 or 4:1.

Monitoring Hyperic Caches

You can monitor Hyperic caches on the **HQ Health** page, on the **Cache** tab, shown in the screenshot below. The following information is shown for each cache:

- Size — The number of objects currently in the cache.
- Hits — How many times a requested object was available in cache since last Hyperic Server restart.
- Misses — How many times a requested object was *not* available in cache since last Hyperic Server restart.
- Limit — The maximum number of objects the cache can contain.
- Total Memory Usage — The amount or memory (in KB) currently consumed by all objects in the cache.

Note: You can view size, hits and misses, but *not* cache limit by running the "ehCache Diagnostics" query on the **Diagnostics** tab. This data is also written periodically written to `server.log`.



Region	Size	Hits	Misses	Limit	Memory Usage (KB)
Total Memory Usage	0	0	0	0	3058355.25
Agent.findByAgentToken	605	26329887	605	5000	1075.16
Agent.findByIP	10	52612	10	5000	24.25
AgentScheduleInQueue	0	0	0	20000	0.00
Alert.findByCreateTime	0	0	0	10	0.00
Alert.findByEntity	0	0	0	100	0.00
AlertDefinition.findByResource	186	7892	186	10000	1034.67
AlertDefinition.getNumActiveDefs	1	37	1	1	1.18
Application.findByServiceId_orderName	22889	12873	22889	50000	58407.20
AuthzSubject.findByAuth	1	20	1	20	1.93
AuthzSubject.findByName	2	61024	2	20	3.79
AvailabilityCache	34134	384537923	114170	35000	20267.06
AvailabilityDownAlertDefinitionCache	25474	139107	182	250000	11269.26
AvailabilitySummary	0	0	0	10000	0.00
Category.findByName	0	0	0	4	0.00
ConfigReponseDB.findByPlatformId	0	0	0	10000	0.00
ConfigReponseDB.findByServerId	0	0	0	10000	0.00

Interpreting Cache Statistics

The values that indicate a well-tuned cache vary by the nature of the caches, and a host of deployment-specific factors. Key things to check for include:

- Has the cache limit been reached?
- What is the hits:misses ratio?

The table below lists statistics for several Hyperic caches and, in the "Comments" column, a possible interpretation of the data.

Cache	Size	Hits	Misses	Limit	Comments
`Agent.findByAgentToken`	605	26010977	605	5000	This cache looks healthy. It contains relatively static objects. The cache has not filled up, and the number of misses is equal to the number of hits, so misses occurred only upon first request of each object.

Cache	Size	Hits	Misses	Limit	Comments
org.hyperic.hq.events.server.session.Alert	70526	48049	71274	100000	This cache looks healthy. It contains a type that is likely to become stale relatively quickly, so aging out is appropriate. Although there are more misses than hits, the low number of objects in memory, compared to the cache limit, indicates a low level of server activity since last restart.
org.hyperic.hq.events.server.session.AlertDefinition	66287	44385	66340	100000	This cache looks healthy. It contains a relatively static type, so it is appropriate that the objects do not age out. The cache is not filled up, and the number of misses is very close to the number of hits, indicating most misses occurred upon first request of the object.

Cache	Size	Hits	Misses	Limit	Comments
`Measurement.findByTemplateForInstance`	10000	6766	25772	10000	This cache looks less healthy. It has reached its maximum size, and the hit ratio is around 20-25%. Ideally, the number of misses should peak at about the maximum size of the cache. Increasing the cache limit would probably improve Hyperic performance. (Note that the rule-of-thumb that misses should peak around the limit of the cache does not apply to the `UpdateTimestampsCache` and the `PermissionCache` caches, which contain types that are invalidated frequently.

About Java Heap and Garbage Collection

Heap size startup options are set in the `server.java.opts` property. Note that how much you can increase the heap size depends on the amount of RAM on the Hyperic Server host. Given sufficient RAM, you could use these settings:

```
server.java.opts=-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx4096m -Xms4096m
-XX:+UseConcMarkSweepGC -XX:+UseCompressedOops
```

Note: If you are running Hyperic Server on a 64-bit system with 4GB (4096 MB) or less memory, Hyperic recommends you use 32-bit JVM. If you use a 64-bit JVM, be sure to set the `-XX:+UseCompressedOops` in `server.java.opts` property. with the oops option set.

About Hyperic Server Caches

Hyperic Server uses Ehcache for in-memory caching. Effective cache management is necessary for server stability and performance. Caching policies that define the cache size (maximum number of objects to cache) for each type are defined in `server-n.n.n-EE\hq-engine\hq-server\webapps\ROOT\WEB-INF\classes\ehcache.xml`. The cache size for a type depends on the on how dynamic that type is: how often it likely to be is updated. Given a fixed amount of memory, cache sizing in Hyperic tries to allocate cache according to these guidelines:

- Relatively static types — Caches for types that are not frequently updated — for instance, `Resource`, `Platform`,`Server`, and `Measurement` — are sized to keep objects in memory for the lifetime of the HQ Server. An extremely low miss rate desired. The default cache sizes (the maximum number elements in cache) configured in `ehcache.xml` for inventory types are:
    - Platforms — 2,000
    - Servers ---- 50,000
    - Services — 100,000

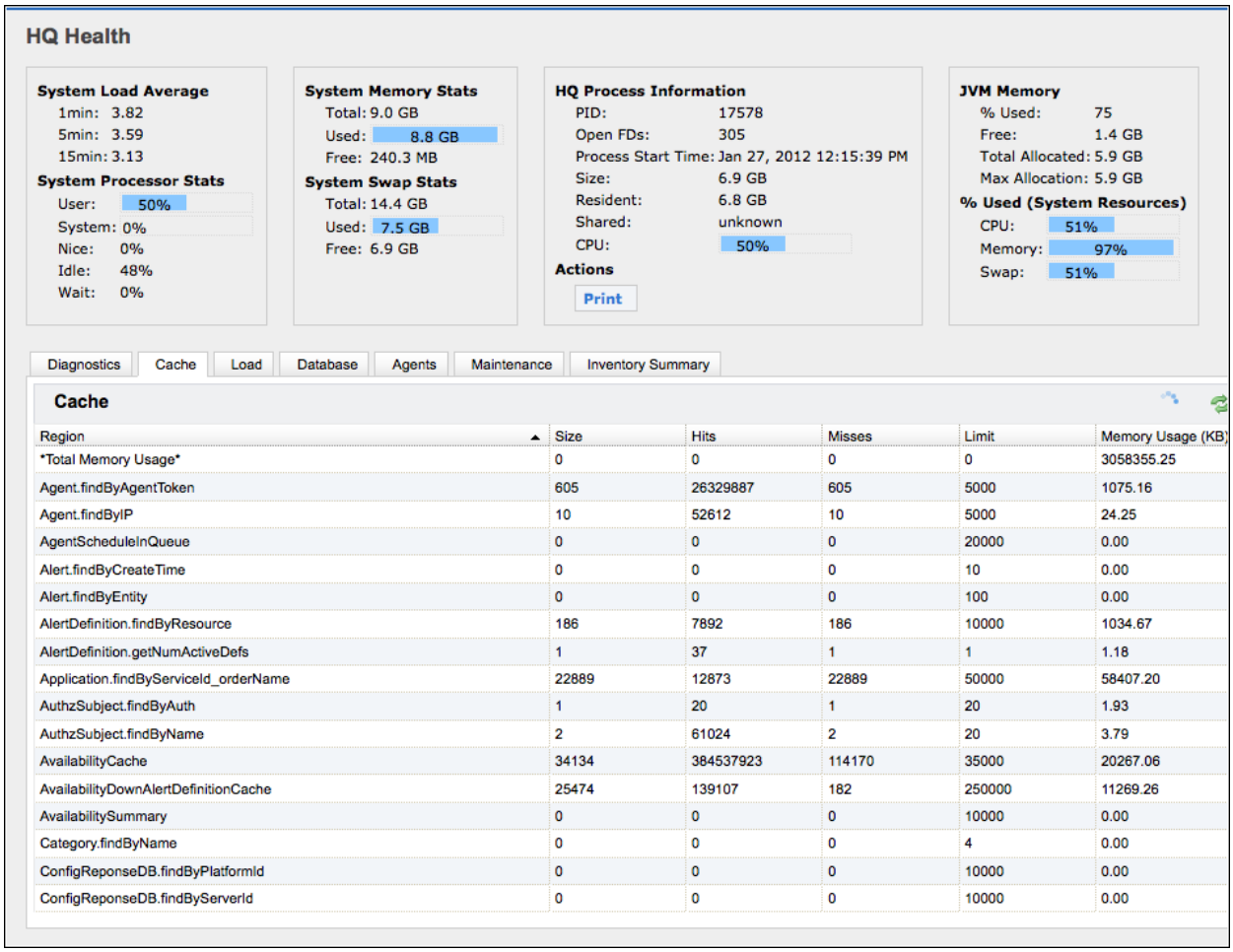
    This sizing should be adequate for medium to large deployments.
- Dynamic types — Caches for types that are frequently updated, (for instance `Alert` and `Galert`) and hence get stale sooner, are configured such that objects age out more quickly. A high hit/miss ratio of is optimal for dynamic types, in larger environments, on the order of 2:1 or 4:1.

Monitoring Hyperic Caches

You can monitor Hyperic caches on the **HQ Health** page, on the **Cache** tab, shown in the screenshot below. The following information is shown for each cache:

- Size — The number of objects currently in the cache.
- Hits — How many times a requested object was available in cache since last Hyperic Server restart.
- Misses — How many times a requested object was *not* available in cache since last Hyperic Server restart.
- Limit — The maximum number of objects the cache can contain.
- Total Memory Usage — The amount or memory (in KB) currently consumed by all objects in the cache.

Note: You can view size, hits and misses, but *not* cache limit by running the "ehCache Diagnostics" query on the **Diagnostics** tab. This data is also written periodically written to `server.log`.



Region	Size	Hits	Misses	Limit	Memory Usage (KB)
Total Memory Usage	0	0	0	0	3058355.25
Agent.findByAgentToken	605	26329887	605	5000	1075.16
Agent.findByIP	10	52612	10	5000	24.25
AgentScheduleInQueue	0	0	0	20000	0.00
Alert.findByCreateTime	0	0	0	10	0.00
Alert.findByEntity	0	0	0	100	0.00
AlertDefinition.findByResource	186	7892	186	10000	1034.67
AlertDefinition.getNumActiveDefs	1	37	1	1	1.18
Application.findByServiceId_orderName	22889	12873	22889	50000	58407.20
AuthzSubject.findByAuth	1	20	1	20	1.93
AuthzSubject.findByName	2	61024	2	20	3.79
AvailabilityCache	34134	384537923	114170	35000	20267.06
AvailabilityDownAlertDefinitionCache	25474	139107	182	250000	11269.26
AvailabilitySummary	0	0	0	10000	0.00
Category.findByName	0	0	0	4	0.00
ConfigReponseDB.findByPlatformId	0	0	0	10000	0.00
ConfigReponseDB.findByServerId	0	0	0	10000	0.00

Interpreting Cache Statistics

The values that indicate a well-tuned cache vary by the nature of the caches, and a host of deployment-specific factors. Key things to check for include:

- Has the cache limit been reached?
- What is the hits:misses ratio?

The table below lists statistics for several Hyperic caches and, in the "Comments" column, a possible interpretation of the data.

Cache	Size	Hits	Misses	Limit	Comments
`Agent.findByAgentToken`	605	26010977	605	5000	This cache looks healthy. It contains relatively static objects. The cache has not filled up, and the number of misses is equal to the number of hits, so misses occurred only upon first request of each object.

Cache	Size	Hits	Misses	Limit	Comments
org.hyperic.hq.events.server.session.Alert	70526	48049	71274	100000	This cache looks healthy. It contains a type that is likely to become stale relatively quickly, so aging out is appropriate. Although there are more misses than hits, the low number of objects in memory, compared to the cache limit, indicates a low level of server activity since last restart.
org.hyperic.hq.events.server.session.AlertDefinition	66287	44385	66340	100000	This cache looks healthy. It contains a relatively static type, so it is appropriate that the objects do not age out. The cache is not filled up, and the number of misses is very close to the number of hits, indicating most misses occurred upon first request of the object.

Cache	Size	Hits	Misses	Limit	Comments
`Measurement.findByTemplateForInstance`	10000	6766	25772	10000	This cache looks less healthy. It has reached its maximum size, and the hit ratio is around 20-25%. Ideally, the number of misses should peak at about the maximum size of the cache. Increasing the cache limit would probably improve Hyperic performance. (Note that the rule-of-thumb that misses should peak around the limit of the cache does not apply to the `UpdateTimestampsCache` and the `PermissionCache` caches, which contain types that are invalidated frequently.

Configuring Caches

Caches you cannot change
There are two caches that you cannot reconfigure:

- `org.hibernate.cache.UpdateTimestampsCache` is managed by Hibernate.
- `AvailabilityCache` is managed by Hyperic Server.

To modify the size of a Hyperic cache, you edit the associate element in `server-n.n.n-EE\hq-engine\hq-server\webapps\ROOT\WEB-INF\classes\ehcache.xml`. (In general, only cache sizes should need to be changed.) Each cache is defined with an entry like:

```
<cache name="DerivedMeasurement.findByTemplateForInstance"
    maxElementsInMemory="10000"
    eternal="true"
    timeToIdleSeconds="0"
    timeToLiveSeconds="0"
    memoryStoreEvictionPolicy="LRU"/>
```

You may need to iterate on the cache size to find the optimal setting.

Configure Hyperic Version and Security Announcements

Hyperic sends email announcements to Hyperic administrators when a key release is upcoming, or to distribute important product information. You can configure the level of messages you wish to receive or disable receipt of Hyperic notifications with the **HQ Version and Security Announcements** property, in the **Announcement Properties** section of the **Administration > HQ Server Settings** page. You can choose:

- **All**
- **Major** — default value
- **None**

Integrate Hyperic Server with Other Systems

These topics have instructions for enabling Hyperic Server to communication with other enterprise systems:

- Configure Kerberos Properties
- Configure LDAP Properties
- Configuring Hyperic Server for SMTP Server
- Enable vFabric Hyperic to Send SNMP Traps

Configure Kerberos Properties

In vFabric Hyperic, these properties on the **Admin > HQ Server Settings** page configure Hyperic Server to use Kerberos authentication.

- **Realm** — Identifies the Kerberos realm.
- **KDC** — Identifies the Kerberos kdc
- **Debug** — Enables debug logging.

Configure LDAP Properties

Configure LDAP Authentication

To configure Hyperic Server to use LDAP authentication for new users and to assign user roles based on LDAP group membership:

1. Click HQ Server Settings on the Administration tab.
2. Scroll down to the "LDAP Configuration Properties* section of the page
3. In the enter the properties described below:

Property	Description
Use LDAP Authentication	Checkmark this option to enable LDAP authentication.

Property	Description
URL	Enter the location of your LDAP or Active Directory server. If other than the standard LDAP port is used, specify it the URL. Add the port to the end of the URL, after a colon (:) character. For example: `ldap://YourLDAPHost:44389` If your LDAP directory requires SSL, specify the SSL port in the URL.
SSL	Place a checkmark in the box if your LDAP directory requires SSL connections.
Username	Supply an LDAP username with sufficient privileges to view the sections of the directory that contain the information for LDAP users who will access Hyperic. (Not necessary if the LDAP directory allows anonymous searching, rare insecure environments.
Password	Supply the password for the LDAP user specified in "Username" above.
Search Base	(Required) The "Search Base" property, sometimes referred to as the suffix, defines the location in the LDAP directory from which the LDAP user search begins. Supply the full path to the branch for example: `ou=people,dc=example,dc=com` Consult your LDAP administrator if necessary.
Search Filter	If desired, enter a filter to limits the LDAP user search to a subset of the object identified by the "Search Base" property. For example, `(!(location=SFO*))`
Login Property	(Required) The LDAP property (for an LDAP user) that Hyperic will use as the username for the user's Hyperic account. The default value is "cn". Depending on your LDAP environment, a different property, for instance, "uid", may be appropriate.

Property	Description
Group Search Base	Analogous to "Search Base", this property defines the location in the LDAP directory from which the LDAP group search begins. If you want Hyperic to automatically assign Hyperic roles to new users, supply a value for this property.
Search Subtree	If you have configured the "Group Search Base", described above, you can checkmark this box, to enable search of the entire subtree of the object identified by "Group Search Base"
Group Search Filter	If you have configured the "Group Search Base", described above, you can enter a filter to limit the LDAP group search to a subset of the objects found in the group search. The default value "Member={0}", results in filtering by the full distinguished name of a user. To filter by user login name, set "Member={1}" to filter on the login name.

4. Click **OK**.

Configuring Hyperic Server for SMTP Server

Hyperic sends emails using the SMTP server specified during Hyperic Server installation. On many Unix and Linux machines, the default — `localhost` is satisfactory. In this case, no additional configuration is required. To use a remote SMTP server, you configure the Hyperic Server with the remote host connection information, and set up authentication in `hq-server.conf`.

1. Define SMTP properties in the "Email Settings section" of the `<HQ Server directory>/conf/hq-server.conf` file. As installed, `hq-server.conf` does not contain the mail properties - you must add the properties to override Hyperic's default settings. The properties you define depend on whether you wish to use plain text or SSL communication. Note that changes to `hq-server.conf` take effect only after restart Hyperic Server restart.
    - To configure plain text communication, add the mail properties shown below to `hq-server.conf`. Hyperic's default behavior is equivalent to the values shown below - replace the values as appropriate for your environment.

```
# Change to the SMTP gateway server
# maps to mail.smtp.host,
  server.mail.host=localhost
# Change to SMTP port
  mail.smtp.port=25
# SMTP properties
  mail.smtp.auth=false
  mail.smtp.socketFactory.class=javax.net.SocketFactory
  mail.smtp.socketFactory.fallback=false
```

```
mail.smtp.socketFactory.port=25
mail.smtp.starttls.enable=false
```

- To configure SSL communication, define the following properties:

```
server.mail.host=SmtpServerHost
mail.user=SmtpUser
mail.password=SmtpPassword
mail.smtp.port=587
mail.smtp.auth=true
mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory
mail.smtp.socketFactory.fallback=false
mail.smtp.socketFactory.port=465
mail.smtp.starttls.enable=true
```

2. If you are using SSL/TLS, add the SMTP Server's TLS certificate to the JRE keystore:
   a. Obtain a copy of the public certificate for the SMTP server's TLS configuration (not the private key) on the Hyperic Server.
   b. As the user that owns the Hyperic installation, execute the following in the server installation directory:

```
jre/bin/keytool -keystore jre/lib/security/cacerts -import -storepass
changeit -file /path/to/smtp_server_tls.cert
```

   c. When asked if you want to trust the certificate, answer "yes."

Note: The certificate import example above assumes the use of a JRE that is bundled with the Hyperic Server. When using a non-bundled JRE, use that JRE's keytool and cacerts file. For more information, see *Configure SSL Options*.

Enable vFabric Hyperic to Send SNMP Traps

This section has information about enabling vFabric Hyperic to send SNMP traps to an SNMP management system.

Note: For information about enabling Hyperic to *receive traps*, see Configuring Hyperic as an SNMP Trap Receiver.

Configure HQ Server to Send SNMP Traps

1. Click **HQ Server Settings** on the Administration page.
2. At the bottom of the page, in the "SNMP Server Configuration Properties" section, define the properties for your version of SNMP. See the appropriate section below.

Configure vFabric Hyperic Server for SNMP v1

Select "v1" from the **SNMP Protocol Version** pulldown and supply values for the properties defined in the table below.

The table below defines the properties for configuring Hyperic Server for SNMP V1 communications with an NMS.

Configuration Option	Description	Allowable Values
SNMP Trap OID	The OID of the notification to be sent. Supplies the value of `snmpTrapOID.0` - the second varbind in a trap or inform that Hyperic Server generates. (The first varbind is `SysUpTime.0`.)	
Default Notification Mechanism	Your selection governs the notification type that will appear as the default notification type option in the "Notification Mechanism" pulldown list that is presented in configuration dialogs when user configures an SNMP notification as an alert action, or as a step in an escalation.	For v1 of the SNMP protocol, choose V1 Trap. This is the only trap type you can generate for SNMP v1.
Enterprise OID	Enterprise OID.	
Community	The community name to be sent with the trap.	
Generic ID	Single digit identifier of the trap type.	0 - coldStart 1 - warmStart 2 - linkDown 3 - linkUp 4 - authenticationFailure 5 - egpNeighborLoss 6 - enterpriseSpecific
Specific ID	The specific trap code for an enterprise-specific trap (when **Generic ID** is set to to 6).	
Agent Address	Address of the managed object that generates the trap.	

Configure vFabric Hyperic Server for SNMP v2c

Configuration Option	Description	Allowable Values
SNMP Trap OID	The OID of the notification to be sent. Supplies the value of `snmpTrapOID.0` - the second varbind in a trap or inform that Hyperic Server generates. (The first varbind is `SysUpTime.0`.)	
Default Notification Mechanism	Specifies the default notification type that will appear in configuration dialogs when an authorized user configures an SNMP notification as an alert action, or as a step in an escalation. This choice simply defines the default option - the user configuring an alert action or escalation can choose a different message type.	• V1 Trap • V2c Trap • Inform
Community	The community name to be sent with the trap.	

Configure vFabric Hyperic Server for SNMP v3

This section lists the properties for enabling vFabric Hyperic to send SNMP notifications to an NMS. When Hyperic is so enabled, you can use SNMP notifications in alert definitions - as alert actions and escalation steps.

Configuration Option	Description	Allowable Values
SNMP Trap OID	The OID of the notification to be sent. Supplies the value of `snmpTrapOID.0` - the second varbind in a trap or inform that Hyperic Server generates. (The first varbind is `SysUpTime.0`.)	
Default Notification Mechanism	Specifies the default notification type that will appear in configuration dialogs when an authorized user configures an SNMP notification as an alert action, or as a step in an escalation. This choice simply defines the default option - the user configuring an alert action or escalation can choose a different message type.	• V1 Trap • V2c Trap • Inform
Security Name	The username Hyperic's SNMP agent should use when sending notifications to the NMS.	Required.

Configuration Option	Description	Allowable Values
Local Engine ID	ID of Hyperic's SNMP agent; this value appears automatically, and is not user-configurable.	
Auth Protocol	The SNMP authentication protocol Hyperic Server should use for communications with the NMS.	• none • MD5 • SHA
Auth Passphrase	The SNMP authorization passphrase configured for use when communication with the NMS.	
Privacy Protocol	The SNMP Privacy Protocol Hyperic Server should use for communication with the NMS.	• none • DES • 3DES • AES-128, • AES-192 • AES-256
Privacy Passphrase	The SNMP privacy passphrase configured for use when communication with the NMS.	
Context Engine ID	The EngineID of the NMS. This, along with Context Name, identifies the SNMP context for accessing management data.	Required for v1 and v2c traps. Do not supply for Inform.
Context Name	The name of the SNMP context that provides access to management information on the NMS. A context is identified by the Context Name and Context Engine ID.	

Using SNMP Traps in Alert Definitions

After the configuration above is complete, the "SNMP Trap" notification tab is available when you define or edit an alert definition.

Managing the HQ Database

This section has topics related to the Hyperic database. It includes information about database maintenance and optional configurations.

Building a Metric Data Warehouse

Hyperic's retention strategy for measurement data is it to store the minimum amount of data that enables it to pinpoint when change in performance or availability occur. Detailed measurement data is stored for a limited period of time - two days, by default - after which the data is compressed and archived as hourly averages with highs and lows. You can configure Hyperic to keep detailed measurement data for longer, up to a maximum of 7 days.

To support requirements for trend analysis over a longer time frame, vFabric Hyperic provides the MetricDataReplicator class, which you can use to replicate uncompressed measurement data in a secondary database.

Metric Replication Strategy Overview

Detailed steps for creating and populating a secondary database for detailed metrics are provided in the sections that follow. This is a summary of the approach:

- A secondary database instance is configured to store detailed measurement data replicated from the primary Hyperic database. The secondary database contains one table, EAM_MEASUREMENT_DATA. This guide currently only covers MySQL, adjustments will need to be made for Oracle and PostgreSQL.
- The secondary database has a database link to the primary Hyperic database, and five views that point to the primary Hyperic database for resource inventory data. The resource inventory data does not physically reside on the secondary database. The database link to the main database allows views on the secondary database to access inventory data in the primary Hyperic database. These are the views that are required on the secondary database:
    - EAM_PLATFORM
    - EAM_SERVER
    - EAM_SERVICE
    - EAM_RESOURCE
    - EAM_MEASUREMENT_TEMPL
    - EAM_MEASUREMENT

For documentation on these database tables, see Hyperic Database Table Schemas.

Instructions for Establishing Secondary MySQL Database for Metrics

These sections below have instructions for configuring a secondary Hyperic database for metrics on MySQL.

Note: MySQL's query optimizer has limitations that have negative impact on the the performance of Hyperic's metric replicator class. This performance degradation can have a ripple effect on the performance of your primary Hyperic database during metric replication as such the replicator class should be ran against a slave mysql db for optimum performance. This document is written around the use of the slave DB.

Set Up the Secondary Database

Perform the steps below to create the secondary database and configure access to your primary Hyperic database. All of the steps in this section apply to the secondary database.

Another option would be to use MySQL federated tables

1. Install your secondary MySQL Database Server and setup replication.
2. Create user and database for warehouse

```
CREATE DATABASE hqdata;
GRANT ALL ON hqdata.* to hqdata identified by 'password';
FLUSH PRIVILEGES;
USE hqdata;
```

3. Create table for measurements

```
create table EAM_MEASUREMENT_DATA
(
            TIMESTAMP bigint,
            MEASUREMENT_ID int,
            VALUE numeric(24, 5),
            primary key (TIMESTAMP, MEASUREMENT_ID)
);
```

4. Create view measurements, replace hqdb with your Hyperic Db name

```
create view EAM_MEASUREMENT
 as select ID,
            VERSION_COL,
            INSTANCE_ID,
            TEMPLATE_ID,
            MTIME,
            ENABLED,
            COLL_INTERVAL,
            DSN
            from hqdb.EAM_MEASUREMENT;
```

5. Create view for platforms, replace hqdb with your Hyperc DB name

```
create view EAM_PLATFORM
 as select ID,
           VERSION_COL,
           FQDN,
           CERTDN,
           DESCRIPTION,
           CTIME,
           MTIME,
           MODIFIED_BY,
           LOCATION,
           COMMENT_TEXT,
           CPU_COUNT,
           PLATFORM_TYPE_ID,
           CONFIG_RESPONSE_ID,
           AGENT_ID,
           RESOURCE_ID
           from hqdb.EAM_PLATFORM;
```

6. Create view for servers, replace hqdb with your Hypierc DB name

```
create view EAM_SERVER as
  select ID,
           VERSION_COL,
           DESCRIPTION,
           CTIME,
           MTIME,
           MODIFIED_BY,
           LOCATION,
           PLATFORM_ID,
           AUTOINVENTORYIDENTIFIER,
           RUNTIMEAUTODISCOVERY,
           WASAUTODISCOVERED,
           SERVICESAUTOMANAGED,
           AUTODISCOVERY_ZOMBIE,
           INSTALLPATH,
           SERVER_TYPE_ID,
           CONFIG_RESPONSE_ID,
           RESOURCE_ID
           from hqdb.EAM_SERVER;
```

7. Create view for services, replace hqdb with your Hyperic DB name

```
create view EAM_SERVICE as
  select ID,
           VERSION_COL,
           DESCRIPTION,
           CTIME,
           MTIME,
           MODIFIED_BY,
           LOCATION,
           AUTODISCOVERY_ZOMBIE,
```

```
           SERVICE_RT,
           ENDUSER_RT,
           PARENT_SERVICE_ID,
           SERVER_ID,
           SERVICE_TYPE_ID,
           CONFIG_RESPONSE_ID,
           RESOURCE_ID
           from hqdb.EAM_SERVICE;
```

8. Create view for resources, replace hqdb with your Hyperic Db name

```
create view EAM_RESOURCE
 as select ID,
           VERSION_COL,
           RESOURCE_TYPE_ID,
           INSTANCE_ID,
           SUBJECT_ID,
           PROTO_ID,
           NAME,
           SORT_NAME,
           FSYSTEM,
           MTIME
           from hqdb.EAM_RESOURCE;
```

9. Create view for measurement templates, replace hqdb with your Hyperic DB name

```
create view EAM_MEASUREMENT_TEMPL as
  select ID,
           VERSION_COL,
           NAME,
           ALIAS,
           UNITS,
           COLLECTION_TYPE,
           DEFAULT_ON,
           DEFAULT_INTERVAL,
           DESIGNATE,
           TEMPLATE,
           PLUGIN,
           CTIME,
           MTIME,
           MONITORABLE_TYPE_ID,
           CATEGORY_ID
           from hqdb.EAM_MEASUREMENT_TEMPL;
```

10. OPTIONAL: Create view for Availability data, replace hqdb with your Hyperic DB name

```
create view HQ_AVAIL_DATA_RLE
 as select MEASUREMENT_ID,
            STARTIME,
            ENDTIME,
            AVAILVAL
            from hqdb.HQ_AVAIL_DATA_RLE;
```

Availability isn't replicated as it is stored differently and doesn't have compression. This view is provided as an example if you want to query a single db for availability data.

Test the New Views

To verify the views you created work, you can run a query that lists all servers in the database. Enter the following query at the mysql prompt of the secondary database and to list all of the servers. This query runs against the hqdb database.

```
SELECT * FROM EAM_SERVER;
```

Set up the Metric Data Replicator

Once your secondary database is ready to store and view the Hypierc data, you must set up the metric_replicator.properties file with the appropriate parameters. Create a directory where the replicator files will be stored, for instance:
/usr/hyperic/replicator

Property Setting	Description
pri_user=hqadmin	Primary database username
pri_pass=hqadmin	Primary database password
pri_url=jdbc:mysql://<ipaddress>:<port>/hqdb	Connection string for primary server, including IP Address and port
sec_user=hqdata	Secondary database username
sec_pass=password	Secondary database password
sec_url=jdbc:mysql://<ipaddress>:<port>/hqdata?	Connection string for secondary server, including IP Address and port.
interval	Time interval in minutes. Use to specify the interval the script runs at. Default: 90
time_chunk	Amount of minutes to do during this run. Default: 90
batch_size	Number of metrics do to in a batch. Default: 2000

Create log4j Properties File

Create a file called log4j.properties in the replicator directory you created in the previous step and paste this text into the file:

```
log4j.rootLogger=DEBUG, R
log4j.appender.R=org.apache.log4j.ConsoleAppender
log4j.appender.R.layout=org.apache.log4j.PatternLayout
log4j.appender.R.layout.ConversionPattern=%d \[PRIVATE:%t\] %-5p %c - %m%n
```

Create Script to Run the Replication Process

Create a file called `run.sh`, which will be the script that runs the replication process. Copy the commands shown below, changing the values of `JAVA_HOME` and `SERVER_HOME` to point to your Java and Hyperic Server installations respectively.

```
#!/bin/bash

#update these params
SERVER_HOME="/opt/vmware/vfabric/hyperic/server/active/hq-engine/hq-server"

# props file which configures the replicator
PROPS="metric_replicator"

HQ_ROOT="$SERVER_HOME/webapps/ROOT"

# path to the prop files
# update to reflect appropriate database
PROP_FILES="."
DB_PKGS="$HQ_ROOT/WEB-INF/lib/mysql-connector-java-commercial-5.1.10.jar"
HQ_PKGS="$HQ_ROOT/WEB-INF/lib/hq-common-4.5.jar"
LOG_PKGS="$HQ_ROOT/WEB-INF/lib/commons-logging-1.0.4.jar:$HQ_ROOT/WEB-
INF/lib/log4j-1.2.14.jar"
HQEE_PKGS="$HQ_ROOT/WEB-INF/lib/hqee-server-4.5.jar"
HQUTIL_PKGS="$HQ_ROOT/WEB-INF/lib/hq-util-4.5.jar"
PKGS="$PROP_FILES:$DB_PKGS:$HQ_PKGS:$HQEE_PKGS:$LOG_PKGS:$HQUTIL_PKGS"
ARGS="$LOG_ARGS -Dreplprops=$PROPS -Djdbc.drivers=com.mysql.jdbc.Driver -cp $PKGS"
JAVA="$JAVA_HOME/bin/java"

set -x

$JAVA $ARGS com.hyperic.hq.measurement.shared.MetricDataReplicator
```

Run the Replication Process

To run the replication process, open a terminal window and enter:

```
run.sh
```

Verify the Replication Process Results

Run the queries in the following sections to verify that the replication process succeeded.

Query 1 - Show all the disk stats

Run the following query to show all metrics whose name contains the string "disk", replacing the your.platform.nam below with a valid platform name in your resource inventory.

```
SELECT p.fqdn,
r.name,
t.name,
d.value,
d.timestamp
from EAM_MEASUREMENT_TEMPL t,
EAM_MEASUREMENT m,
EAM_MEASUREMENT_DATA d,
EAM_SERVER s,
EAM_PLATFORM p,
EAM_RESOURCE r
where t.id = m.template_id AND
m.id = d.measurement_id AND
p.id = s.platform_id AND
r.instance_id = m.instance_id AND
lower(p.fqdn) = 'your.platform.name' AND
lower(r.name) LIKE '%Mount%'
ORDER BY d.timestamp DESC;
```

Query 2 - Component Service Usage Information for Servers and Services

Once the appropriate join has been made to access the server or service layer, filtering by server or service name or metric template is the easiest way to select specific metrics of interest per server or service. (Meaning "server" and "service", as defined in the HQ inventory model.) For example, JBoss is a server while the individual web applications running within the container are services. Metrics specific to the JBoss container are available from the server layer while the internal web applications are available via the service layer.

```
SELECT platform.fqdn,
resource.name,
template.name,
data.value,
data.timestamp
FROM EAM_MEASUREMENT_TEMPL template,
EAM_MEASUREMENT measurement,
EAM_MEASUREMENT_DATA data,
EAM_SERVICE service,
EAM_SERVER server,
EAM_PLATFORM platform,
EAM_RESOURCE resource
WHERE 1=1 AND
template.id = measurement.template_id AND
measurement.id = data.measurement_id AND
platform.id = server.platform_id AND
server.id = service.server_id AND
```

```
service.id = measurement.instance_id AND
lower(platform.fqdn) = 'your.platform.name' AND
lower(resource.name) like '%jboss%'
AND lower(template.name) like '%transaction count%'
ORDER BY data.timestamp desc;
```

Hyperic Database Backup and Recovery

The Hyperic database contains most of the data necessary to recreate your Hyperic Server environment after a failure, or to move the database to a different host. In addition to historical metrics, the database contains configuration settings, such as Hyperic Agent connection information, collection intervals, portlet configurations, groups, roles, and users. Some server configuration data, such as database connection information, the mail server for alerts, and Java arguments used at server startup, is stored in external files.

Like any other database, your Hyperic database should be backed up on a regular basis, so that you can restore the data in the event of a failure that corrupts or destroys the database. It is also good practice to backup the database prior to upgrading Hyperic, your database server, or other software that resides on the server machine.

You should define Hyperic backup procedures and incorporate them into your overall backup processes. Your local requirements and practices will dictate backup frequency, timing, naming conventions, and retention policies. A daily backup is sufficient for most environments.

Shut down Hyperic Server if backup makes database unavailable
If your database backup process makes the Hyperic database unavailable, shut down the Hyperic Server before running the backup.

Backing up Built-In PostgreSQL Database

If you use Hyperic's built-in PostgreSQL database, back it up with the PostreSQL pg_dump command:

```
pg_dump hqdb | gzip > hqdb-MM.DD.YY.dump.gz
```

Copy the dump file to your backup location.

Always use this method to back up the built-in database; do not simply copy the contents of the database's data directory.

Hyperic Files to Backup

In addition to the Hyperic database, you may want to create a backup of the following directories and files in your server directory:

```
conf/
bin/hq-server.sh
hqdb/data/postgresql.conf
```

You can back up these files while Hyperic Server is running.

The contents of these files are stable. Changes are infrequent once your Hyperic Server is installed at configured. Back them up at that time and after making changes to the sever configuration.

Configure Hyperic Server Data Compression and Purge Behavior

Hyperic Server Data Management Processes

HQ Server stores monitoring results using a tiered model to minimize the volume of data stored, while still providing sufficient data granularity. Periodically, the HQ Server removes detailed metric data from the database and archives it. Alerts and events older than a specified age are removed from the database, and not archived.

The server performs the following periodic data management functions:

- Compress and archive measurement data — Hyperic Server stores detailed metric data (all data points reported) in the Hyperic database for a configurable period (up to 7 days) of time, after which the metrics are eligible for compression and archival. On a (configurable) periodic basis, the server removes the aged individual metric data points from the database, and archives the metric data in compressed form: hourly metric averages, highs, and lows. Hyperic Server retains the archived metric data for 2 years.
- Purge alert data — Hyperic Server retains fired alert data for a configurable period, after which the alerts are deleted.
- Purge event data — Hyperic Server retains event data for a configurable period, after which the events are deleted.
- Rebuild metric table indexes — During normal Hyperic operation, the metric data tables in the Hyperic database contain a lot of frequently changing data. The Hyperic Server rebuilds the metric table indexes on a (configurable) periodic basis to avoid performance problems that heavily fragmented indexes can cause.

Configure Hyperic Data Management Settings

You can configure how Hyperic condenses and purges the contents of the Hyperic database on the **Administration** > **Server Settings** page. Retaining fewer days of detailed metric data and deleting alerts and other events on a timely basis can improve Hyperic performance.

Option	Description	Default	Notes
Run Database Maintenance Every	Controls how frequently Hyperic compresses and archives detailed metric data that is older than the age specified by the following property.	Hourly	
Delete Detailed Metric Data Older Than	Controls how many days of detailed metric data Hyperic retains before compressing it into hourly averages with highs and lows and archiving those values.	2 days	You cannot enter a value greater than 7.
Reindex Metric Data Tables Nightly	Controls whether Hyperic reindexes metric data tables every night. If configured to re-index nightly, Hyperic re-indexes the tables around midnight.	Nightly	
Delete Alerts Older Than	Controls how long Hyperic stores alert event data.	31 days	
Delete Events and Logs Older Than	Controls how long Hyperic stores other Hyperic event and log data.	31 days	

Warning: Data Management Changes Require Server Restart
Any changes made in this section require the Hyperic Server to be restarted before they take effect.

Monitoring the Hyperic Database

Hyperic administrators can view real-time Hyperic Server and database health and load data by clicking **HQ Health** on the Administration page in the HQ user interface.

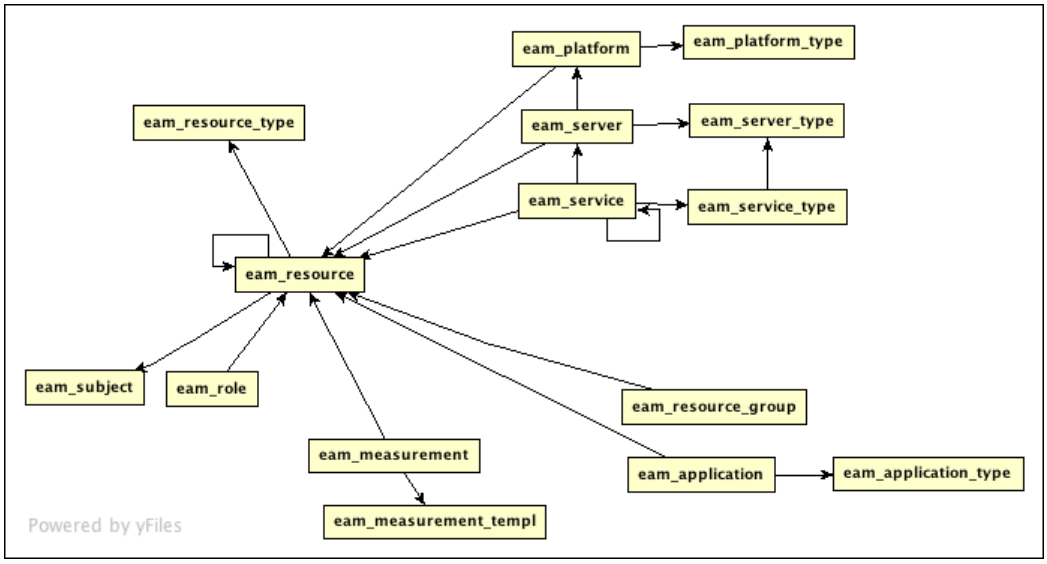
The information on the HQ Health page is useful to Hyperic internals experts; Hyperic support engineers may use the **HQ Health** data and diagnostics to diagnose and troubleshoot Hyperic Server and database problems.

For database configuration options configuration options that may be useful in large Hyperic deployments, see the topics in Scaling and Tuning Hyperic Performance.

Hyperic Database Table Schemas

Key Resource and Measurement Tables

This section has information about key Hyperic database tables that contain information about resources, metric collection, and measurements.



EAM_RESOURCE Table - All Resource Types and Instances

The EAM_RESOURCE table contains information about the types in the Hyperic inventory model (described in Resources, Resource Types and Inventory Types and instances of those types in the database. This table has a row for *every* managed resource in the Hyperic database, including

- Operating system platforms, and the servers and services that run on them.
- Virtual or network host platforms, and the servers and services that run on them.
- Groups and applications
- Roles and users
- Escalations

Tables for Inventory Resources

The following tables have information about resource instances of a particular inventory type:

- EAM_PLATFORM - Contains a row for each platform in inventory.
- EAM_SERVER - Contains a row for each server in inventory.
- EAM_SERVICE - Contains a row for each service in inventory.
- EAM_RESOURCE_GROUP - Contains a row for each group in inventory.
- EAM_APPLICATION - Contains a row for each application in inventory.

Tables for Platform, Server, and Service Types

The following tables have information about resource types for an inventory type.

- EAM_PLATFORM_TYPE — Contains a row for every platform type that Hyperic can manage.
- EAM_SERVER_TYPE — Contains a row for every server type that Hyperic can manage.
- EAM_SERVICE_TYPE — Contains a row for every service type that Hyperic can manage.

Tables for Measurement Information

The following tables have information about the measurements that Hyperic can collect.

- EAM_ MEASUREMENT_TEMPL - Contains a row for a every metric available for every inventory resource type with its metric template and default metric collection settings.
- EAM_MEASUREMENT - Contains a row for every metric available for every resource in inventory, with metric collection configuration information: whether collection is enabled and the collection interval for enabled metrics.

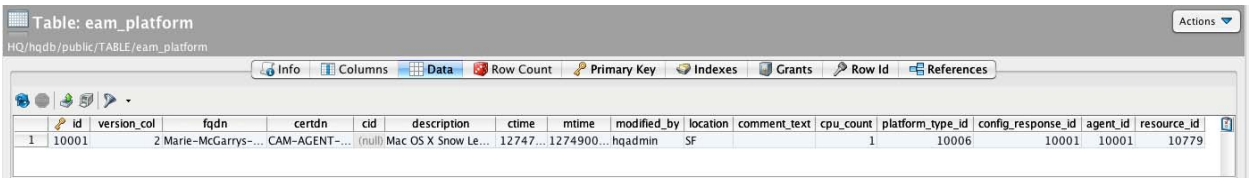
Note: These tables do not store metric values. Metric data is stored in the EAM_MEASUREMENT_DATA_1H, EAM_MEASUREMENTt_DATA_6H, and EAM_MEASUREMENT_DATA_1D tables.

Table Documentation

The sections below document the structure for key resource and measurement tables in the Hyperic database.

EAM_PLATFORM

Contains a row for each platform in inventory. Click the thumbnail to see example column data.



Field	Type	Description
ID	int4	An ID for the platform, unique among platforms.
VERSION_COL	int8	Version of the row. Increments when the row is modified. increments with any change to configuration of this row
FQDN	varchar(200)	Fully qualified domain name of the platform.

Field	Type	Description
CERTDN	varchar(200)	SSL Certificate for the agent which is monitoring this platform.
CID	int4	not used
DESCRIPTION	varchar(256)	Description of platform.
CTIME	int8	Creation time of platform.
MTIME	int8	Last modification time of the platform.
MODIFIED_BY	varchar(100)	Last modification user.
LOCATION	varchar(100)	String entered by user, optionally.
COMMENT_TEXT	varchar(256)	String entered by user, optionally.
CPU_COUNT	int4	Number of CPUs on this platform.
PLATFORM_TYPE_ID	int4	ID for the platform type. Points to EAM_PLATFORM_TYPE table.
CONFIG_RESPONSE_ID	int4	Link to configuration string in plugin xml file.
AGENT_ID	Int4	A unique identifier to the agent which is monitoring this platform.
RESOURCE_ID	int4	Uniquely identifies the resource, unique across platforms, servers, services. Points to the EAM_RESOURCE table.

EAM_PLATFORM_TYPE

Contains a row table for each Hyperic-supported platform type. Click the thumbnail to see column names and example column data.

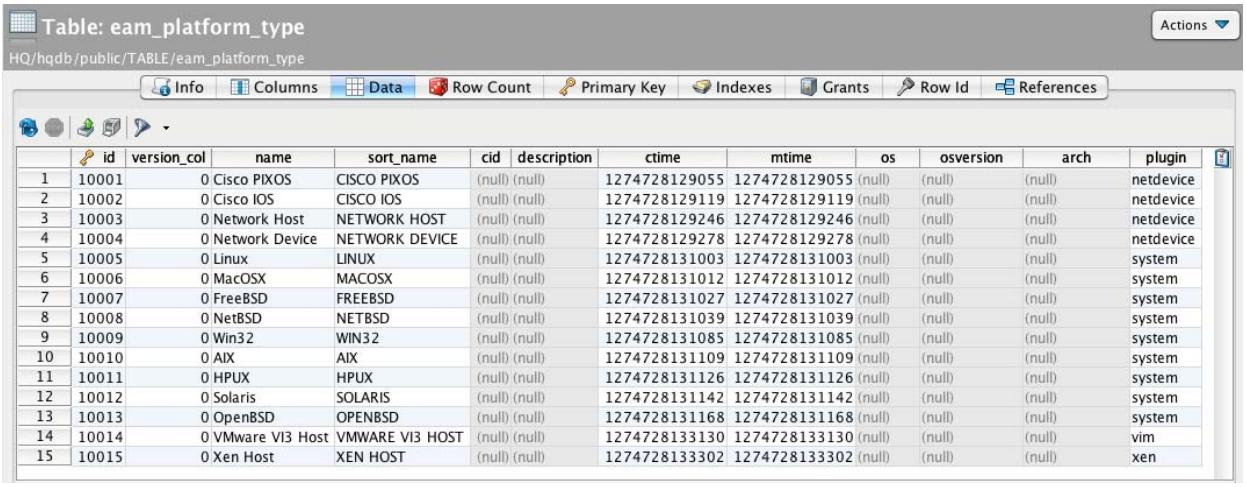


Table: eam_platform_type

HQ/hqdb/public/TABLE/eam_platform_type

Actions

Info | Columns | Data | Row Count | Primary Key | Indexes | Grants | Row Id | References

	id	version_col	name	sort_name	cid	description	ctime	mtime	os	osversion	arch	plugin
1	10001	0	Cisco PIXOS	CISCO PIXOS	(null)	(null)	1274728129055	1274728129055	(null)	(null)	(null)	netdevice
2	10002	0	Cisco IOS	CISCO IOS	(null)	(null)	1274728129119	1274728129119	(null)	(null)	(null)	netdevice
3	10003	0	Network Host	NETWORK HOST	(null)	(null)	1274728129246	1274728129246	(null)	(null)	(null)	netdevice
4	10004	0	Network Device	NETWORK DEVICE	(null)	(null)	1274728129278	1274728129278	(null)	(null)	(null)	netdevice
5	10005	0	Linux	LINUX	(null)	(null)	1274728131003	1274728131003	(null)	(null)	(null)	system
6	10006	0	MacOSX	MACOSX	(null)	(null)	1274728131012	1274728131012	(null)	(null)	(null)	system
7	10007	0	FreeBSD	FREEBSD	(null)	(null)	1274728131027	1274728131027	(null)	(null)	(null)	system
8	10008	0	NetBSD	NETBSD	(null)	(null)	1274728131039	1274728131039	(null)	(null)	(null)	system
9	10009	0	Win32	WIN32	(null)	(null)	1274728131085	1274728131085	(null)	(null)	(null)	system
10	10010	0	AIX	AIX	(null)	(null)	1274728131109	1274728131109	(null)	(null)	(null)	system
11	10011	0	HPUX	HPUX	(null)	(null)	1274728131126	1274728131126	(null)	(null)	(null)	system
12	10012	0	Solaris	SOLARIS	(null)	(null)	1274728131142	1274728131142	(null)	(null)	(null)	system
13	10013	0	OpenBSD	OPENBSD	(null)	(null)	1274728131168	1274728131168	(null)	(null)	(null)	system
14	10014	0	VMware VI3 Host	VMWARE VI3 HOST	(null)	(null)	1274728133130	1274728133130	(null)	(null)	(null)	vim
15	10015	0	Xen Host	XEN HOST	(null)	(null)	1274728133302	1274728133302	(null)	(null)	(null)	xen

EAM_SERVER

Contains a row for each server in Hyperic inventory.

Field	Type	Description
ID	int4	A unique identifier of the server.
VERSION_COL	int8	A column which increments with any change to configuration of this row.
CID	int4	
DESCRIPTION	varchar(300)	Description of server.
CTIME	int8	Creation time of server.
MTIME	int8	Last modification time of the server.
MODIFIED_BY	varchar(100)	Last modification user
LOCATION	varchar(100)	
PLATFORM_ID	int4	The Unique ID of the platform on which this server is installed
AUTOINVENTORYIDENTIFIER	varchar(250)	A unique ID describing this server via the plugin XML.
RUNTIMEAUTODISCOVERY	bool	Is runtime autodiscovery enabled on this server?
WASAUTODISCOVERED	bool	Was this server autodiscovered?
SERVICESAUTOMANAGED	bool	Not used.
AUTODISCOVERY_ZOMBIE	bool	Were there deletions on the client side for this server?
INSTALLPATH	varchar(200)	Install path of this server on the platform.
SERVER_TYPE_ID	int4	Unique ID of the server type that describes this server.
CONFIG_RESPONSE_ID	int4	Link to configuration string in plugin xml file.
RESOURCE_ID	int4	Uniquely identifies the resource, unique across platforms, servers, services. Points to the EAM_ RESOURCE table.

EAM_SERVICE

Contains a row for each service in Hyperic inventory

Field	Type	Description
ID	int4	An ID for the service, unique among services.
VERSION_COL	int8	A column which increments with any change to configuration of this row
CID	int4	

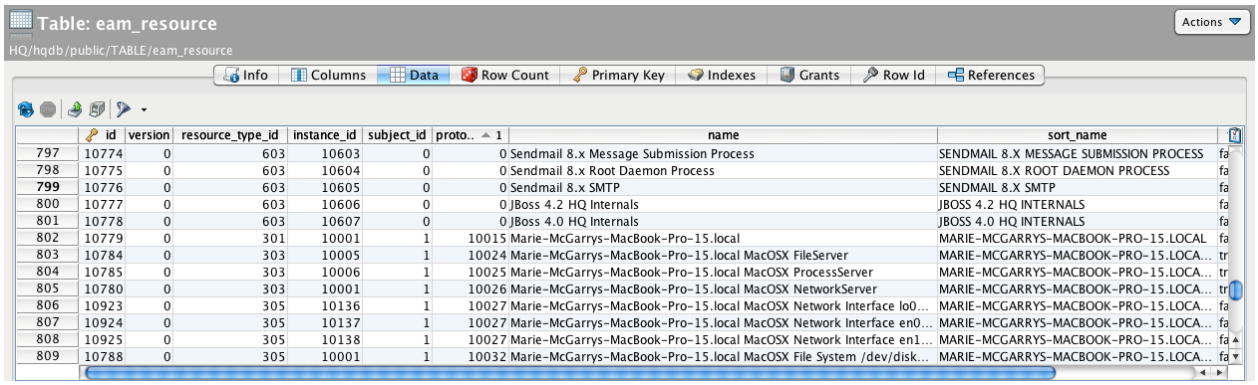
Field	Type	Description
DESCRIPTION	varchar(200)	Description of service.
CTIME	int8	Creation time of service.
MTIME	int8	Last modification time of the service.
MODIFIED_BY	varchar(100)	Last modification user.
LOCATION	varchar(100)	Not used.
AUTODISCOVERY_ZOMBIE	bool	Were there deletions on the client side for this service?
SERVICE_RT	bool	Is response time enabled for this service?
ENDUSER_RT	bool	Is end user response time enabled for this service?
PARENT_SERVICE_ID	int4	Unique ID into the parent service for this service.
SERVER_ID	int4	Were there deletions on the client side for this server?
AUTOINVENTORYIDENTIFIER	varchar(500)	A unique ID describing this server via the plugin XML.
SERVICE_TYPE_ID	Int4	Unique ID of service type for this service.
CONFIG_RESPONSE_ID	int4	Link to configuration string in plugin xml file.
RESOURCE_ID	int4	Uniquely identifies the resource, unique across platforms, servers, services. Points to the EAM_ RESOURCE table.

EAM_RESOURCE

This table contains a row for each type in the Hyperic inventory model, and a row for each instance of each type in the Hyperic database, including:

- basic inventory types - Platforms, Servers, and Services
- configurable inventory types - Groups and Applications
- Users and Roles
- Escalations

Click the thumbnail to see example column data.



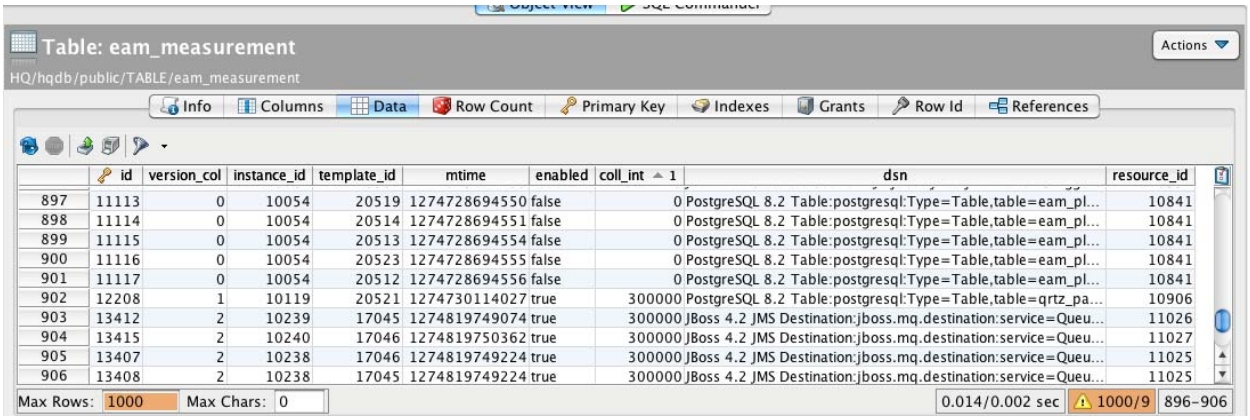
id	version	resource_type_id	instance_id	subject_id	proto.. ▲ 1	name	sort_name
797	10774	0	603	10603	0	0 Sendmail 8.x Message Submission Process	SENDMAIL 8.X MESSAGE SUBMISSION PROCESS
798	10775	0	603	10604	0	0 Sendmail 8.x Root Daemon Process	SENDMAIL 8.X ROOT DAEMON PROCESS
799	10776	0	603	10605	0	0 Sendmail 8.x SMTP	SENDMAIL 8.X SMTP
800	10777	0	603	10606	0	0 JBoss 4.2 HQ Internals	JBOSS 4.2 HQ INTERNALS
801	10778	0	603	10607	0	0 JBoss 4.0 HQ Internals	JBOSS 4.0 HQ INTERNALS
802	10779	0	301	10001	1	10015 Marie-McGarrys-MacBook-Pro-15.local	MARIE-MCGARRYS-MACBOOK-PRO-15.LOCAL
803	10784	0	303	10005	1	10024 Marie-McGarrys-MacBook-Pro-15.local MacOSX FileServer	MARIE-MCGARRYS-MACBOOK-PRO-15.LOCA...
804	10785	0	303	10006	1	10025 Marie-McGarrys-MacBook-Pro-15.local MacOSX ProcessServer	MARIE-MCGARRYS-MACBOOK-PRO-15.LOCA...
805	10780	0	303	10001	1	10026 Marie-McGarrys-MacBook-Pro-15.local MacOSX NetworkServer	MARIE-MCGARRYS-MACBOOK-PRO-15.LOCA...
806	10923	0	305	10136	1	10027 Marie-McGarrys-MacBook-Pro-15.local MacOSX Network Interface lo0...	MARIE-MCGARRYS-MACBOOK-PRO-15.LOCA...
807	10924	0	305	10137	1	10027 Marie-McGarrys-MacBook-Pro-15.local MacOSX Network Interface en0...	MARIE-MCGARRYS-MACBOOK-PRO-15.LOCA...
808	10925	0	305	10138	1	10027 Marie-McGarrys-MacBook-Pro-15.local MacOSX Network Interface en1...	MARIE-MCGARRYS-MACBOOK-PRO-15.LOCA...
809	10788	0	305	10001	1	10032 Marie-McGarrys-MacBook-Pro-15.local MacOSX File System /dev/disk...	MARIE-MCGARRYS-MACBOOK-PRO-15.LOCA...

Field	Type	Description
ID	int4	Uniquely identifies a type or an instance of a type.
VERSION_COL	int8	Increments with any change to configuration of this row.
RESOURCE_TYPE_ID	int4	Identifies a type in the Hyperic inventory model. The values this attribute identify a resource as one of the following:
INSTANCE_ID	int4	Uniquely identifies a type or an instance of a particular type in the inventory model. For a type, corresponds to the ID column in one of the following tables: EAM_PLATFORM_TYPE, EAM_SERVER_TYPE&, EAM_SERVICE_TYPE, EAM_APPLICATION_TYPE, or EAM_RESOURCE_TYPE. For an instance of a type, corresponds to the ID column in one of the following tables: EAM_PLATFORM, EAM_SERVER, EAM_SERVICE, EAM_RESOURCE_GROUP, EAM_APPLICATION, EAM_ROLE, EAM_SUBJECT. EAM_ESCALATION
SUBJECT_ID	int4	Identifies the Hyperic user owns the resource.
PROTO_ID	int4	For a type, value is zero. For an instance of a type, contains the value of the ID column for the type in this table.
NAME	varchar(500)	Display name for a resource, for example, "My-Office-MacBook-Pro-15.local JBoss 4.2 default ServiceManager Stateless Session EJB".

Field	Type	Description
SORT_NAME	varchar(500)	Same as the NAME column but all in upper case, for example, "MY-OFFICE-MACBOOK-PRO-15.LOCAL JBOSS 4.2 DEFAULT SERVICEMANAGER STATELESS SESSION EJB".
FSYSTEM	boolean	
MTIME	int8	Last modification time of the resource.

EAM_MEASUREMENT

Each row contains information about a measurement for a resource under management. Click the thumbnail to see example column data.



	id	version_col	instance_id	template_id	mtime	enabled	coll_int ▲ 1	dsn	resource_id
897	11113	0	10054	20519	1274728694550	false	0	PostgreSQL 8.2 Table:postgresql:Type=Table,table=eam_pl...	10841
898	11114	0	10054	20514	1274728694551	false	0	PostgreSQL 8.2 Table:postgresql:Type=Table,table=eam_pl...	10841
899	11115	0	10054	20513	1274728694554	false	0	PostgreSQL 8.2 Table:postgresql:Type=Table,table=eam_pl...	10841
900	11116	0	10054	20523	1274728694555	false	0	PostgreSQL 8.2 Table:postgresql:Type=Table,table=eam_pl...	10841
901	11117	0	10054	20512	1274728694556	false	0	PostgreSQL 8.2 Table:postgresql:Type=Table,table=eam_pl...	10841
902	12208	1	10119	20521	1274730114027	true	300000	PostgreSQL 8.2 Table:postgresql:Type=Table,table=qrtz_pa...	10906
903	13412	2	10239	17045	1274819749074	true	300000	JBoss 4.2 JMS Destination:jboss.mq.destination:service=Queu...	11026
904	13415	2	10240	17046	1274819750362	true	300000	JBoss 4.2 JMS Destination:jboss.mq.destination:service=Queu...	11027
905	13407	2	10238	17046	1274819749224	true	300000	JBoss 4.2 JMS Destination:jboss.mq.destination:service=Queu...	11025
906	13408	2	10238	17045	1274819749224	true	300000	JBoss 4.2 JMS Destination:jboss.mq.destination:service=Queu...	11025

EAM_MEASUREMENT_TEMPL

Contains a row for a every measurement that Hyperic can collect, for every resource type it can manage, with information about the default metric collection settings.

Field	Type	Description
ID	int4	A unique identifier of a measurement template for a metric for a resource.
VERSION_COL	int8	A column which increments with any change to configuration of this row
NAME	varchar(100)	Name of this measurement template.
ALIAS	varchar(100)	String that describes the alias portion of XML file.
UNITS	varchar(50)	Units of this measurement.
COLLECTION_TYPE	int4	Static/Dynamic data.
DEFAULT_ON	bool	Does this measurement collect by default?
DEFAULT_INTERVAL	int8	The default collection interval of this metric.

Field	Type	Description
DESIGNATE	bool	Is this metric on the indicator page by default?
TEMPLATE	varchar(2048)	Template string from plugin XML.
PLUGIN	varchar(250)	Name of the plugin which houses this measurement template.
CTIME	int8	Creation time of server.
MTIME	int8	Last modification time of the server.
MONITORABLE_TYPE_ID	int4	Key into the monitorable type data.
CATEGORY_ID	int4	Key into the category ID table.

MySQL Maintenance Examples

Here are examples of regular maintenance for mysql

- Simple MySQL Backup Script

Field	Type	Description
ID	int4	Unique ID for a metric that can be collected for a resource. Points to actually measurements in EAM_MEASUREMENT_DATA_* tables.
VERSION_COL	int8	Indicates version of the row, increments upon each change to the row.
INSTANCE_ID	int4	The resource type the measurement is for. Uniquely identifies a resource type of a given inventory level - platform, server, service. For example, the ID 10001 uniquely identifies the platform type "MacOSX".
TEMPLATE_ID	int4	ID of a template points to the EAM_MEASUREMENT_TEMPL table.
MTIME	int8	Time modified.
ENABLED	boolean	Is this metric enabled?
COLL_INTERVAL	int8	How often this metric is collected.

Field	Type	Description
DSN	varchar(2048)	A string which describes the measurement from the plugin-xml text.
RESOURCE_ID	int4	Uniquely identifies the resource for which the metric is associated, unique across platforms, servers, services. Points to the EAM_ RESOURCE table.

```
#!/bin/sh

START=`date '+%A %Y/%m/%d %H:%M:%S'`
DAY=`date +%A`
MYSQLADMIN="/usr/bin/mysqladmin"
MYSQLDUMP="/usr/bin/mysqldump"
USER="root"
PASSWORD="mysql"
DBNAME="hqdb"
DEST="/home/mysql/dumps/$DBNAME-$DAY.sql.gz"
flushCmd="$MYSQLADMIN -u $USER -p$PASSWORD flush-logs"
dumpCmd="$MYSQLDUMP -u $USER -p$PASSWORD --quick --single-transaction
$DBNAME"
gzip="gzip"
echo "Starting backup: $START"
echo "$flushCmd && $dumpCmd | $gzip > $DEST"
$flushCmd && $dumpCmd | $gzip > $DEST
END=`date '+%A %Y/%m/%d %H:%M:%S'`
echo "Backup completed: $END"
```

- Simple Log Rollover Scheme. This may be done with error files, log files, etc.

```
cp /path/to/mysql/log/mysqld.err /path/to/mysql/log/mysqld-`date '+%w'`.err
;
cp /dev/null /path/to/mysql/log/mysqld.err
```

- Sample Unix Cron Entries (empty lines will fail in cron, beware)

```
#
#       Field 1: (0-59) minute
#       Field 2: (0-23) hour
#       Field 3: (1-31) day  of  the month
#       Field 4: (1-12) month of the year
#       Field 5: (0-6)  day of the week - 1=Monday
# ------------------------------------------------------------------------
--
#
0 2 * * * backup.sh
0 1 * * * cp /path/to/mysql/log/mysqld.err /path/to/mysql/log/mysqld-`date
'+%w'`.err ;
cp /dev/null /path/to/mysql/log/mysqld.err
```

Clustering Hyperic Servers for Failover

Overview

To avoid interruption of Hyperic Server operation in the case of failure, you can configure a cluster of Hyperic Servers. The failover configuration uses:

- EHCache's distributed caching for replicating changes throughout the cluster.
- The `nodeStatus.hqu` plugin for monitoring the availability of nodes.
- A hardware load balancer for managing failover when an node becomes unavailable. The load balancer checks the status of each node every 10 seconds, by issuing an HTTP request to the node's `nodeStatus.hqu` plugin. The check returns a response of `master=true` with a return code of `200` for the primary node. The check returns `master=false` with a return code of `404` inside the body of the response for other nodes in the cluster.

A Hyperic Server cluster contains multiple nodes; two are generally sufficient. One Hyperic Server, automatically selected by Hyperic, serves as the primary node. The other node or nodes serve as hot backups---they do not share the workload with the primary node.

A failover configuration is transparent to users and Hyperic administrators; it is not apparent that the active Hyperic server instance is clustered, or which node is currently active.

Requirements for an Failover Deployment

- A hardware-based load balancer.
- Only one Hyperic Server in an Hyperic Server cluster should receive agent communications at a time. The load balancer should should not direct agent connections to an Hyperic server instance that serves as the secondary node.

- Database Considerations---All nodes in the Hyperic cluster must share the same database. You cannot use Hyperic's internal PostgreSQL database in a failover configuration. You must use an external Postgres database.
- The database password, and the encryption key used to encrypt the database password on each Hyperic Server instance must be identical. Supply the same database password and encryption key when installing each of the server instances to be clustered.

Configuring a Server Cluster

These instructions assume that you do not already have an Hyperic Server installation.

Step 1 - Install the First Hyperic Server Instance

Run the full installer, following the instructions at Run Hyperic Installer and select the external Postgres database option. Clustering requires the use of an external Hyperic database. The installer will create the Hyperic database schema.

Note the encryption key you supply during the installation process.

Step 2 - Install Additional Hyperic Server Nodes

For each additional node:

- Run the full installer and select the external Postgres database option.
- When the installer installer prompts for the location of the Hyperic database, specify the location of the database created for the first server instance.
- When the installer asks if you want to upgrade, overwrite, or exit the process, select the choice for "upgrade".
- When prompted to supply the database password and the encryption key to be used to encrypt the database password, enter the same password and encryption key supplied when installing the first server instance.

Step 3 - Configure Cluster Name and Communications Properties

Configure the cluster-related properties on each of the Hyperic Servers in the cluster, in the "Cluster Settings" section of its `conf/hq-server.conf` file.

Default hq-server.conf File

```
# Cluster Settings
##############################################################################
#
# Property: ha.partition
#  
# This property defines the name of the HQ cluster. Each HQ server
with the
# same ha.partition name will join the same cluster. This property is
required
```

```
# for proper cluster initialization.
#
#ha.partition=

#
# Property: ha.node.address
#
# This property defines the IP address or hostname to bind the
multicast listener
# to. This property is required for proper cluster initialization.
#
#ha.node.address=

#
# Property: ha.node.mcast_addr
#
# This property defines the multicast address to use. This property is
not required
# and defaults to 238.1.2.3.
#
#ha.node.mcast_addr=238.1.2.3

#
# Property ha.node.mcast_port
#
# This property defines the multicast port to use. This property is
not required
# and defaults to 45566.
#
#ha.node.mcast_port=45566

#
# Property ha.node.cacheListener.port
#
# This property defines the multicast port that is used to discover
cache peers. This
# property is not required and defaults to 45567
#ha.node.cacheListener.port=45567

#
# Property ha.node.cacheProvider.port
#
# This property defines the multicast port that is used to synchronize
caches throughout
```

```
# the HQ cluster. This property is not required and defaults to 45568.
#ha.node.cacheProvider.port=45568
```

Required Cluster Properties

For each Hyperic Server in the cluster you must specify:

ha.partition	Name of the cluster---this value is identical for each node in the cluster
ha.node.address	Multicast listen address---specifies IP address or hostname upon which the node listens for multicast traffic; this value is unique to each node in the cluster.

Note: If you are upgrading from a pre-v3.0 failover configuration, the each server's .conf file will contain obsolete cluster properties, including server.cluster.mode and server.ha.bind_addr properties. Delete these properties and replace with the current failover properties described below.

Optional Cluster Properties

If desired, you can control these communication behaviors for the nodes in the cluster:

ha.node.mcast_addr and ha.node.mcast_port	Address and port for sending multicast messages to other nodes. Note: ha.node.mcast_addr must be the same on each node.
ha.node.cacheListener.port and ha.node.cacheProvider.port	Ports used for discovering and synchronizing with cache peers.

Step 4 - Configure the Load Balancer

Configure the load balancer, according to the vendor or supplier instructions. Procedures vary, but at a minimum you will identify the Hyperic Server nodes in the cluster and the failover behavior.

1. Identify the Hyperic Server nodes in the cluster.
2. Configure the load balancer to check the nodeStatus.hqu URL every 10 seconds. For example, in a 2-node cluster, if the the IP addresses of the nodes are 10.0.0.1 and 10.0.0.2, configure the load balancer to check these URLs every 10 seconds:

   ```
   http://hqadmin:hqadmin@10.0.0.1:7080/hqu/health/status/nodeStatus.hqu
   ```

   ```
   http://hqadmin:hqadmin@10.0.0.2:7080/hqu/health/status/nodeStatus.hqu
   ```

3. Configure the load balancer to direct all traffic to the node whose status is `master=true`.

Step 5 - Configure Agents to Communicate with Hyperic Server Cluster

The Hyperic Agents in your environment communicate with the Hyperic Server cluster through the load balancer. When you startup a newly installed agent, either supply the load balancer listen address and port interactively, or specify the connection information in agent.properties.

For existing agents, you can run `hq-agent.sh setup`, to force the setup dialog.

Step 6 - Start the Nodes

Start the Hyperic Servers.

Troubleshooting a Failover Configuration

This section describes the most common sources of problems the failover configuration.

- Multicast blocking ---T he cluster detection and cache peer detection relies on multicast. Make sure your router isn't blocking multicast packets; otherwise the Hyperic cluster will fail to initialize properly. It's also common for virtualization technologies like VMware and Xen to not enable multicast by default.
- Don't register agents using the loopback address — If you install an Hyperic Agent on the same machine as a Hyperic Server node, when you specify the IP address the server should use to contact the agent, don't specify loopback address (127.0.0.1).
- Alerts that were currently firing or in escalation were "lost" — A failover to another cluster node occurred in the middle of the alerts being fired or escalated. The alert state could be lost.

Hyperic Server Properties

Configuration Settings in hq-server.conf

`hq-server.conf` contains the configuration settings that Hyperic Server requires to start up and get ready for work. For instance, `hq-server.conf` has properties that tell the server how to connect to the database and where to listen for agent and web application communications.

When you install Hyperic Server, the selections you can make - port selections, use of plaintext or SSL communications, and so on - correspond to properties in `hq-server.conf`. The configuration settings you supply during installation are persisted in `ServerHome/conf/hq-server.conf`.

In addition to the properties that reflect installation choices, `hq-server.conf` contains properties with default values that you can modify, after installation, based on the your environment and the size of your Hyperic deployment. For example, there are properties in `hq-server.conf` that set defaults for database and JMS configuration options.

Each time Hyperic Server starts up, it reads the values of the properties in `hq-server.conf`.

Note: Hyperic Server supports some properties that do not appear in `hq-server.conf` unless you add them explicitly.

After you change the values of properties in `hq-server.conf` or add new properties to the file, you must restart the server for the new settings to take effect.

Configuration Settings in the Database

Some of the configuration data that governs Hyperic Server behavior is stored in the Hyperic Server database. For example, the data Hyperic Server needs to contact an Hyperic Agent is stored in in the Hyperic Database. For information about how Hyperic Server obtains Hyperic Agent address information, see About the Agent Launcher and Agent Startup.

Server Property Definitions

- accept.unverified.certificates
- server.caf.brokerAddress
- server.caf.clientId
- server.database-driver
- server.java.opts
- server.quartzDelegate
- server.database-url
- server.connection-validation-sql
- server.database-password
- server.database-user
- server.encryption-key
- server.webapp.port
- server.webapp.secure.port
- server.custom.plugin.dir
- server.keystore.password
- server.keystore.path
- server.pluginsync.enable
- server.hibernate.dialect
- server.jms.jmxport
- server.jms.usejmx
- server.database-blockingtimeout
- server.database-minpoolsize
- server.database
- tomcat.minsparethreads
- tomcat.maxthreads
- server.jms.maxmemory
- server.jms.highmemory
- server.database-maxpoolsize
- server.mail.host
- vcops.license.key
- vfabric.licenseServer.url
- vcloud.license.key

Clustering Properties in vFabric Hyperic

For information about properties for configuring a Hyperic Server cluster, see Clustering Hyperic Servers for Failover.

accept.unverified.certificates

Internal Use Only
This property definition is visible to Confluence users in the "Hyperic Members" group — not to other site visitors.

Description

This property controls whether or a warning is issued when a Hyperic Agent presents an SSL certificate that is not in the server's keystore and is either self-signed or signed by a different CA than the one that signed the the server's SSL certificate.

Under these circumstances, if `accept.unverified.certificates=false`, as it is by default, this warning is issued:

```
The server to agent communication channel is using a self-signed certificate and
could not be verified
Are you sure you want to continue connecting? [default=no]: yes
```

If you respond "yes", the server imports the agents's certificate, and will trust it henceforth.

Note that if `accept.unverified.certificates` is "true", the server automatically accepts and imports the certificate presented by a Hyperic Agent, and does not issue a warning if an agent presents a certificate that the server does not trust.
Do NOT set `accept.unverified.certificates=true` unless ALL agents reporting to the Hyperic Server have been upgraded to Hyperic 4.6.

For more information, see *Hyperic Security Features and Recommendations* in *Getting Started with vFabric Hyperic*.

Default

`agent.setup.acceptUnverifiedCertificate=false`

server.caf.brokerAddress

Description

The address upon which the Hyperic Server's internal RabbitMQ node (an Advanced Message Queuing Protocol (AMQP) Broker) listens for TCP/IP requests from Common Agent Framework (CAF) agents.

Default

`localhost`

server.caf.clientId

Description

Common Agent Framework (CAF) UUID.

Default

server.database-driver

Description

The JDBC driver to use. This property is rarely modified.

Default

None. The value is set as a result of the database selected during Hyperic Server installation.

server.java.opts

Description

Options to pass to Java at Hyperic Server startup.

For information about the effect of this and other server properties on Hyperic performance, see Scaling and Tuning Hyperic Performance.

Default
In Hyperic HQ the default value is:

```
server.java.opts=-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx512m -Xms512m -
XX:+HeapDumpOnOutOfMemoryError -XX:+UseConcMarkSweepGC
```

In vFabric Hyperic, the value of the property can vary by the installation profile for the Hyperic Server.

- small — `server.java.opts=Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx512m -Xms512m -XX:+HeapDumpOnOutOfMemoryError -XX:+UseConcMarkSweepGC`
- medium — `-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx4g -Xms4g -XX:+HeapDumpOnOutOfMemoryError -XX:+UseConcMarkSweepGC`
- large — `-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx8g -Xms8g -XX:+HeapDumpOnOutOfMemoryError -XX:SurvivorRatio=12 -XX:+UseConcMarkSweepGC -XX:+UseCompressedOops -Xmn4g`

For information about installation profiles and how to select one, see *About Sizing Profiles in vFabric Hyperic*.

Setting Hyperic Server Timezone
You can set the time zone for the JVM in which Hyperic Server runs by adding `-Duser.timezone=Area/Location` to `server.java.opts`, where:

- `Area` — Is a continent or ocean, for example `America`}
- `Location` — Is a city, with underbar (_) for embedded spaces, for example `New_York`.

server.quartzDelegate

Description

The PostgreSQL drive class used by Hyperic Server's scheduler service.

Default

`org.quartz.impl.jdbcjobstore.PostgreSQLDelegate`

server.database-url

Description

The JDBC URL to connect to.

Default

None. The value is set as a result of the database selected during HQ Server installation.

If you select...	the default database URL is...
HQ Built-in Database PostgreSQL	`postgresql://127.0.0.1:9432/hqdb?protocolVersion=2`
PostgreSQL (external)	`jdbc:postgresql://localhost:5432/HQ?protocolVersion=2`

server.connection-validation-sql

Description

The SQL query to run in order to validate a connection from the pool.

Default

`server.connection-validation-sql=select 1`

server.database-password

Description

The database user's password.

Default

none

server.database-user

Description

The database user to connect as.

Default

hqadmin

server.encryption-key

Description

The key for decrypting the Hyperic database user password. The key must be at least 8 characters long, and can contain letters and numbers.

Default

None. The Hyperic installer prompts for `server.encryption-key` during Hyperic Server installation.

server.webapp.port

Description

The HTTP listen port for the Hyperic web-based GUI.

Default

7080

server.webapp.secure.port

Description

The HTTPS listen port for the Hyperic web-based GUI.

Default

7443

server.custom.plugin.dir

Description

The `server.custom.plugin.dir` property can be used to specify the location of the Hyperic's Server's custom plugin directory.

Restart the Hyperic Server after uncommenting `server.custom.plugin.dir` and setting the property value:

```
server.plugin.custom.dir=FullDirectoryPath
```

For more information about plugin deployment, see Plugin Deployment and Management.

Default

By default, the Hyperic Server stores custom plugins in a subdirectory of the current working directory of the Hyperic Server process — `user.dir/hq-plugins`. You must add this property to the configuration file to specify a directory.

server.keystore.password

Description

This property configures the password for Hyperic Servers's SSL keystore. The location of the keystore is defined by the `server.keystore.path` property.

The Hyperic installer (in `-full` mode), prompts for the values of `server.keystore.path` and `server.keystore.password` and persists the responses in `hq-server.conf`.

Password Requirement for Hyperic Keystores
The Hyperic Server's keystore password and private key password **must** be the same — otherwise, the Hyperic Server's internal Tomcat-based server will be unable to start. Follow the same convention for a Hyperic Agent keystore — set the password for the agent keystore be the same as the agent private key,

Best Practices for Hyperic Keystores
Please see:

- *Hyperic Security Features and Recommendations*
- *Configure SSL Options*

in *Getting Started with vFabric Hyperic*

Default

The initial value of `server.keystore.password` is set, based on the response to the installation dialog when you run the Hyperic installer in `-full` mode, depending on how you respond to the following prompt:

```
Would you like us to use a user managed java keystore?
```

- If the response is "yes", the installer prompts for the path and password for your keystore, and saves the values supplied in `server.keystore.path` and `server.keystore.password` (this property), respectively.
- If the response is "no", the sets the value of `server.keystore.path` to `server.keystore.path=ServerHome/conf/hyperic.keystore` — which is the default location for the self-signed certificate that the Hyperic Server generates at first startup, and sets `server.keystore.password` to the default password — `hyperic`.

server.keystore.path

Description

This property configures the location of the Hyperic Servers's SSL keystore. Supply the full path to the keystore. The password for the keystore is defined by the `server.keystore.password` property.

The Hyperic installer (in `-full` mode), prompts for the values of `server.keystore.path` and `server.keystore.password` and persists the responses in `hq-server.conf`.

Specifying keystore path on Windows
On Windows platforms, specify the path to the keystore with Unix-style syntax. To use specify a full Windows path:

- replace back slashes with forward slashes
- put a forward slash at the beginning of the path (before the drive letter)
- if the path contains spaces, put a backslash before each space in the path
  For example, to specify this Windows path using Unix syntax:

```
C:\Documents and Settings\Desktop\keystore
```

  and change it to:

```
/C:/Documents\ and\ Settings/Desktop/keystore
```

Best Practices for Hyperic Keystores
Please see:

- *Hyperic Security Features and Recommendations*
- *Configure SSL Options*

in *Getting Started with vFabric Hyperic*

Default

The initial value of `server.keystore.path` is set, based on the response to the installation dialog when you run the Hyperic installer in `-full` mode, depending on how you respond to the following prompt:

```
Would you like us to use a user managed java keystore?
```

- If the response is "yes", the installer prompts for the path and password for your keystore, and saves the values supplied in `server.keystore.path` (this property) and `server.keystore.password`, respectively.
- If the response is "no", the sets the value of `server.keystore.path` to `server.keystore.path=ServerHome/conf/hyperic.keystore` — which is the default location for the self-signed certificate that the Hyperic Server generates at first startup, and sets `server.keystore.password` to the default password.

server.pluginsync.enable

Description
The `server.pluginsync.enable` property enables or disables Hyperic's Server-Agent Plugin Synchronization (SAPS) feature, described in Plugin Deployment and Management.

Default

`true`

server.hibernate.dialect

Description

The database-specific dialect class used by Hibernate in HQ

Default

`org.hyperic.hibernate.dialect.PostgreSQLDialect`

server.jms.jmxport

Description

This property, new in Hyperic 4.6.5, specifies the port upon which the Hyperic Server JVM listens for JMX requests, if JMX is enabled. By default, JMX is disabled. The property that controls whether or not JMX is enabled is `server.jms.usejmx`. The primary reason to enable JMX is to enable monitoring of the Hyperic Server's internal ActiveMQ Server. With the port closed, the a Hyperic Agent on the same platform as the Hyperic Server will discover ActiveMQ, but cannot obtain ActiveMQ metrics.

Default
1099

server.jms.usejmx

Description

This property, new in Hyperic 4.6.5, controls whether the JMX port on the Hyperic Server JVM is open or closed. The primary reason to open the port is to enable monitoring of the Hyperic Server's internal ActiveMQ Server. With the port closed, the a Hyperic Agent on the same platform as the Hyperic Server will discover ActiveMQ, but cannot obtain ActiveMQ metrics.

When JMX is enabled, the property that defines the JMX port is `server.jms.jmxport`.

Default
The default value is `false`.

server.database-blockingtimeout

Description

Maximum time in milliseconds to wait for a connection from the pool.

Default

10000

server.database-minpoolsize

Description

The minimum number of database connections to keep in the pool.

For information about the effect of this and other server properties on Hyperic performance, see Scaling and Tuning Hyperic Performance.

Default

In Hyperic HQ the default value is 5.

In vFabric Hyperic the default value of the property depends on the installation sizing profile for the server. A sizing profile is selected during the installation process, and may be updated by running the Hyperic Server installer with the `-updateScale` option.

- small — 5
- medium — 20
- large — 50

For information about installation profiles and how to select one, see *About Sizing Profiles in vFabric Hyperic*.

server.database

Description

The kind of database the Hyperic Server will use.

Valid values are:

- PostgreSQL

Default

PostgreSQL

tomcat.minsparethreads

Description

This property was added in vFabric Hyperic 4.6.5.

The mininum number of unused request processing threads that must be available in vFabric Hyperic Server's internal tc Server's thread pool.

For information about the effect of other server properties on Hyperic performance, see Scaling and Tuning Hyperic Performance.

Default

The value of the property varies by the installation profile for the Hyperic Server.

- small — 50
- medium — 100
- large — 200

For information about installation profiles and how to select one, see *About Sizing Profiles in vFabric Hyperic*.

tomcat.maxthreads

Description

This property was added in vFabric Hyperic 4.6.5.

The maximum size of vFabric Hyperic Server's internal tc Server's thread pool.

For information about the effect ofother server properties on Hyperic performance, see Scaling and Tuning Hyperic Performance.

Default

The value of the property varies by the installation profile for the Hyperic Server.

- small — 500
- medium — 2000
- large — 4000

For information about installation profiles and how to select one, see *About Sizing Profiles in vFabric Hyperic*.

server.jms.maxmemory

Description

Configures the JMS broker memory limit.

If the broker memory limit is reached, the broker will block the `send()` call until some messages are consumed and space becomes available on the broker.

The recommended setting for `server.jms.maxmemory` is 90% of the Java heap size. Erratic alert behavior or missed alerts may indicate the settings are too low.

For information about the effect of this and other server properties on Hyperic performance, see Scaling and Tuning Hyperic Performance.

Default

In Hyperic HQ the default value is 400.

In vFabric Hyperic the default value of the property depends on the installation sizing profile for the server. A sizing profile is selected during the installation process, and may be updated by running the Hyperic Server installer with the `-updateScale` option.

- small — 400
- medium — 1600
- large — 3600

For information about installation profiles and how to select one, see *About Sizing Profiles in vFabric Hyperic*.

server.jms.highmemory

Description

The high memory mark for the JMS queue.

For information about the effect of this and other server properties on Hyperic performance, see Scaling and Tuning Hyperic Performance.

Default

In Hyperic HQ the default value is 350.

In vFabric Hyperic the default value of the property depends on the installation sizing profile for the server. A sizing profile is selected during the installation process, and may be updated by running the Hyperic Server installer with the `-updateScale` option.

- small — 350
- medium — 1400
- large — 2400

For information about installation profiles and how to select one, see *About Sizing Profiles in vFabric Hyperic*.

server.database-maxpoolsize

Description

The maximum number of database connections to keep in the pool. This must be set lower than the total number of connections allowed to the backend database.

For information about the effect of this and other server properties on Hyperic performance, see Scaling and Tuning Hyperic Performance.

Default

In Hyperic HQ, the default value is 100.

In vFabric Hyperic the default value of the property depends on the installation sizing profile for the server. A sizing profile is selected during the installation process, and may be updated by running the Hyperic Server installer with the `-updateScale` option.

- small — 100
- medium — 200
- large — 400

For information about installation profiles and how to select one, see *About Sizing Profiles in vFabric Hyperic*.

server.mail.host

Description

The IP or hostname of the SMTP server that the Hyperic server will use for sending alerts and other Hyperic-related emails. Most UNIX platforms have a local SMTP server, in which case localhost or 127.0.0.1 can be used here.

Default

127.0.0.1

vcops.license.key

If you have vCenter Management Operations Suite license, use this property to specify the key in `ServerHome/conf/hq-server.conf`.

For more information about vFabric Hyperic licensing, see *Install or Configure Hyperic License*.

vfabric.licenseServer.url

This property only applies to vFabric Hyperic servers acquired as a part of vFabric Suite; it is ignored if you obtained vFabric Hyperic as a stand-alone component.

Use `vfabric.licenseServer.url` to specify the URL of the VMware Licensing Server that administers your vFabric Suite license. Note that by default, `ServerHome/conf` does not contain `vfabric.licenseServer.url`. If you wish to define the location of the VMware License Server, you must add `vfabric.licenseServer.url` to `ServerHome/conf/hq-server.conf`.

For more information about vFabric Hyperic licensing, see *Install or Configure Hyperic License*.

vcloud.license.key

If you have a vCloud license, use this property to specify the key in `ServerHome/conf/hq-server.conf`.

For more information about vFabric Hyperic licensing, see *Install or Configure Hyperic License*.

vsphere.license.key

“If you have a vSphere license, use this property to specify the key in `ServerHome/conf/hq-server.conf`.

For more information about vFabric Hyperic licensing, see *Install or Configure Hyperic License*.

Tune Hyperic vApp

This section has information about tuning the Hyperic vApp for large deployments. A large environment is defined as one in which the Hyperic Server manages more than 250 platforms.

Operating System Settings

On the Hyperic Server platform and on the Hyperic database platform, add these parameters to `/etc/security/limits.conf`:

```
hyperic          soft    nofile       8192
hyperic          hard    nofile       16384
```

Restart the vApps after saving the chances to `/etc/security/limits.conf`

On the Hyperic Server platform and on the Hyperic database platform, add these parameters to `/etc/sysctl.conf`:

```
net.ipv4.neigh.default.gc_thresh1 = 1024
net.ipv4.neigh.default.gc_thresh2 = 4096
net.ipv4.neigh.default.gc_thresh3 = 8192

net.core.rmem_max=33554432
net.core.wmem_max=33554432
net.ipv4.tcp_rmem=4096 87380 16777216
net.ipv4.tcp_wmem=4096 65536 16777216
net.core.netdev_max_backlog=50000
```

After saving the changes to `/etc/sysctl.conf`, reload the file with this command:

```
root@localhost# sysctl -p
```

Hyperic Server Settings

Increase VM memory for to 10GB.

Add the following to `hq-server.conf`:

```
server.java.opts=-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx8g -Xms8g -
XX:+HeapDumpOnOutOfMemoryError -XX:SurvivorRatio=12 -XX:+UseConcMarkSweepGC -
XX:+UseCompressedOops -Xmn4g
tomcat.maxthreads=3000
server.database-maxpoolsize=400
```

Hyperic Database Settings

Increase VM memory to 11GB

Edit `/opt/vmware/vpostgres/9.1/data/postgresql.conf` to set:

```
shared_buffers = 8GB
effective_cache_size = 2GB
max_connections = 410
```