vFabric Hyperic Configuration

VMware vFabric Hyperic 5.0

This document supports the version of each product listed and supports all subsequent versions until the document is replaced by a new edition. To check for more recent editions of this document, see http://www.vmware.com/support/pubs.

EN-000957-00
You can find the most up-to-date technical documentation on the VMware Web site at:
http://www.vmware.com/support/

The VMware Web site also provides the latest product updates.

If you have comments about this documentation, submit your feedback to:
docfeedback@vmware.com
Table of Contents

1 Configure and Run the Hyperic Agent 5
   Start, Stop, and Other Agent Operations 6
   Configure Auto-Discovery Scanning and Reporting 9
   Configure Agent Logging 10
   Configure Plugin Loading 14
   Tweak the Agent to Enable a Resource Plugin 15
   Manage the Hyperic Agent 21
   Agent Properties 30
   Agent Java Service Wrapper Configuration 51

2 Configure and Run the Hyperic Server 54
   Start and Stop Hyperic Server 55
   Configure Metric Baselining and Alert Processing Behavior 55
   Scaling and Tuning Hyperic Performance 59
   Configure Hyperic Version and Security Announcements 64
   Integrate Hyperic Server with Other Systems 65
   Managing the HQ Database 73
   Clustering Hyperic Servers for Failover 94
   Hyperic Server Properties 99
   Tune Hyperic vApp 110
- Configure and Run the Hyperic Agent (see page 5)
- Configure and Run the Hyperic Server (see page 54)
Configure and Run the Hyperic Agent

These topics provide instructions for configuring the Hyperic® Agent.

1. Understand Agent Environment and Operation

   For background information about agent configuration, communications, startup, and installation environment, see Key Facts for Hyperic Installers.

- Start, Stop, and Other Agent Operations (see page 6)
- Configure Auto-Discovery Scanning and Reporting (see page 9)
- Configure Agent Logging (see page 10)
- Configure Plugin Loading (see page 14)
- Tweak the Agent to Enable a Resource Plugin (see page 15)
- Manage the Hyperic Agent (see page 21)
- Agent Properties (see page 30)
- Agent Java Service Wrapper Configuration (see page 51)
Start, Stop, and Other Agent Operations

Topics marked with * relate to features available only in vFabric Hyperic.

- Run the Agent Launcher from the Command Line (see page 6)
  - Running hq-agent.sh (see page 6)
  - Running hq-agent.bat (see page 6)
- Run the Agent Launcher from the Hyperic User Interface (see page 7)
  - Restart an Agent from the Hyperic User Interface (see page 7)
  - Ping an Agent from the Hyperic User Interface (see page 7)
  - Upgrade an Agent from the Hyperic User Interface (see page 7)
  - Push a Resource Plugin to the Agent from the Hyperic User Interface (see page 8)
- Run the Agent Without the Java Service Wrapper (see page 8)

Starting an Agent for the First Time

The first time you start a 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 a running Hyperic Agent.

**Available only in vFabric Hyperic**

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** (see page 7)
- **Ping an Agent from the Hyperic User Interface** (see page 7)
- **Upgrade an Agent from the Hyperic User Interface** (see page 7)
- **Push a Resource Plugin to the Agent from the Hyperic User Interface** (see page 8)

**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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Default</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoinventory.defaultScan.interval.millis</td>
<td>Specifies how frequently the agent performs a default autoinventory scan.</td>
<td>Commented out, set to 86,400,000 milliseconds, or 1 day.</td>
<td>Note however, that by default, the agent performs the default scan at startup and every 15 minutes thereafter.</td>
</tr>
<tr>
<td>autoinventory.runtimeScan.interval.millis</td>
<td>Specifies how frequently the agent performs a runtime scan.</td>
<td>86,400,000 milliseconds, or 1 day.</td>
<td></td>
</tr>
</tbody>
</table>
Configure Agent Logging

Topics marked with * relate to features available only in vFabric Hyperic.

- Agent Log Files (see page 10)
- Configure Agent Log Name or Location (see page 10)
- Configure Agent Logging Level (see page 11)
- Redirect System Messages to the Agent Log (see page 11)
- Configure Debug Log Level for an Agent Subsystem (see page 12)

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent.logDir</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 (see page 34) 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 "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 (see page 12).

---

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.hq.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.hq.agent.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
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 (see page 16)
- Configure Agent HTTP Request Header (see page 17)
- Configure Agent to Monitor JBoss (see page 18)
- Configure the Data to Log for Windows Events (see page 19)
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

<table>
<thead>
<tr>
<th>jboss.installpath</th>
</tr>
</thead>
</table>

**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. If `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_Administrator@Office Print:7:Printer HP LaserJet 6P was paused.

This entry is for a 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 (see page 22)
- View Hyperic Agent Metrics (see page 25)
- Reduce Agent Memory Footprint (see page 29)
View Status of all Hyperic Agents

Topics marked with * relate to features available only in vFabric Hyperic.

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 (see page 22)
- Health Data for an Agent (see page 23)

For information about using HQ Health to troubleshoot problems, see Troubleshoot Agent and Server Problems.

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.

![Agents Tab in HQ Health](image-url)
### Health Data for an Agent

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>FQDN</td>
<td>Fully-qualified domain name of the platform where the agent runs.</td>
<td></td>
</tr>
<tr>
<td>Address</td>
<td>IP address of the platform where the agent runs.</td>
<td></td>
</tr>
<tr>
<td>Port</td>
<td>Port where the agent listens for communication with the Hyperic Server.</td>
<td>If you configure unidirectional agent - server communications, the agent initiates all communications with the Hyperic Server.</td>
</tr>
<tr>
<td>Version</td>
<td>Agent version number.</td>
<td>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.</td>
</tr>
<tr>
<td>Build #</td>
<td>Agent build number</td>
<td></td>
</tr>
<tr>
<td>Bundle</td>
<td>Agent bundle version</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creation</td>
<td>The date/time that the Hyperic Agent was first started up.</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platforms</td>
<td>Number of platforms the agent manages.</td>
<td>Typically, an agent manages one platform - the platform where it runs. Exceptions include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- If the agent manages a vSphere vCenter instance, the number of platforms shown is the number of VMs the vCenter server manages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- If the agent manages remote network devices or network host platform types.</td>
</tr>
<tr>
<td># Metrics</td>
<td>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.</td>
<td>If one agent bears an inordinate metric load, you may be able to distribute it more evenly. See the &quot;Overloaded Agent&quot; section of Troubleshoot Agent and Server Problems.</td>
</tr>
<tr>
<td>Time</td>
<td>The difference in system clock time between the agent and the Hyperic Server.</td>
<td>A time offset can cause incorrect availability reporting. See the &quot;Out-of-Sync Agent and Server Clocks&quot; section of Troubleshoot Agent and Server Problems.</td>
</tr>
<tr>
<td>Offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(ms)</td>
<td></td>
<td>A question mark in this column indicates that the Hyperic Server is unable to contact the agent.</td>
</tr>
</tbody>
</table>
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

*Topics marked with * relate to features available only in vFabric Hyperic.*

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.

- Agent Monitoring Defaults (see page 25)
- View Agent Indicators Charts (see page 25)
- View Agent Metric Data (see page 26)
- Hyperic Agent Metrics (see page 26)

For information about metrics that indicate Hyperic Agent problems, see Troubleshoot Agent and Server Problems.

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 (see page 26) 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.

The screenshot below is the **Indicators** page for an Hyperic Agent.
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.

<table>
<thead>
<tr>
<th>Category</th>
<th>Metric</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>Availability</td>
<td>Collected by default.</td>
</tr>
<tr>
<td></td>
<td>Start Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Up Time</td>
<td></td>
</tr>
<tr>
<td>Throughput</td>
<td>Number of Active Threads</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Metrics Collected</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Metrics Collected per Minute</td>
<td>By default, this is an indicator metric.</td>
</tr>
<tr>
<td></td>
<td>Number of Metrics which Failed to be</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Collecte</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Metrics which Failed to be</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Collected per Minute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Requests Served</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Requests Served per Minute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Scheduled Metrics</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>Maximum Time Spent Fetching a Metric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maximum Time Spent Processing a Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Minimum Time Spent Fetching a Metric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Minimum Time Spent Processing a Request</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Connection Failures</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Connection Failures per Minute</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of Metric Batches Sent to Server</td>
<td></td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>Number of Metric Batches Sent to Server per Minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Metrics Sent to Server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Metrics Sent to Server per Minute</td>
<td>Collected by default.</td>
<td></td>
</tr>
<tr>
<td>Server Offset</td>
<td>Collected by default.</td>
<td></td>
</tr>
<tr>
<td>Total Time Spent Fetching Metrics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time Spent Fetching Metrics per Minute</td>
<td>Collected by default. High value can indicate overloaded agent or problem with scheduling thread.</td>
<td></td>
</tr>
<tr>
<td>Total Time Spent Processing Requests</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time Spent Processing Requests per Minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time Spent Sending Metrics to Server</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Time Spent Sending Metrics to Server per Minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Utilization</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cpu Total Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cpu Total Time per Minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JVM Free Memory</td>
<td>By default, this is an indicator metric.</td>
<td></td>
</tr>
<tr>
<td>JVM Total Memory</td>
<td>By default, this is an indicator metric.</td>
<td></td>
</tr>
<tr>
<td>Open File Descriptors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resident Memory Used</td>
<td>&quot;Resident Memory&quot; is the amount of memory the Hyperic Agent occupies in memory.</td>
<td></td>
</tr>
<tr>
<td>Time Spent in System Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Spent in System Mode per Minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Spent in User Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Spent in User Mode per Minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Memory Used</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reduce Agent Memory Footprint

*Topics marked with * relate to features available only in vFabric Hyperic.*

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

- Limit Plugin Loading (see page 29)
- Reduce Java Heap (see page 29)
- Delete Javadocs Folder (see page 29)

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 (see page 14) 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
```

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` (see page 32)
- `agent.keystore.alias` (see page 32)
- `agent.keystore.password` (see page 32)
- `agent.keystore.path` (see page 33)
- `agent.listenIp` (see page 34)
- `agent.listenPort` (see page 34)
- `agent.logDir` (see page 34)
- `agent.logFile` (see page 35)
- `agent.logLevel` (see page 35)
- `agent.logLevel.SystemErr` (see page 35)
- `agent.logLevel.SystemOut` (see page 35)
- `agent.maxBatchSize` (see page 36)
- `agent.proxyHost` (see page 36)
- `agent.proxyPort` (see page 36)
- `agent.storageProvider.info` (see page 36)
- `agent.setup.acceptUnverifiedCertificate` (see page 36)
- `agent.setup.camIP` (see page 37)
- `agent.setup.camLogin` (see page 37)
- `agent.setup.camPort` (see page 38)
- `agent.setup.camPword` (see page 38)
- `agent.setup.camSecure` (see page 38)
- `agent.setup.camSSLPort` (see page 39)
- `agent.setup.agentIP` (see page 39)
- `agent.setup.agentPort` (see page 39)
- `agent.setup.resetupToken` (see page 40)
- `agent.setup.unidirectional` (see page 40)
- `agent.startupTimeOut` (see page 40)
- `autoinventory.defaultScan.interval.millis` (see page 41)
- `autoinventory.runtimeScan.interval.millis` (see page 41)
- `http.userAgent` (see page 41)
- `log4j Properties` (see page 41)
- `jboss.installpath` (see page 42)
- `platform.log_track.eventfmt` (see page 43)
- plugins.exclude (see page 44)
- plugins.include (see page 44)
- postgresql.database.name.format (see page 44)
- postgresql.index.name.format (see page 45)
- postgresql.server.name.format (see page 45)
- postgresql.table.name.format (see page 46)
- scheduleThread.cancelTimeout (see page 47)
- scheduleThread.fetchLogTimeout (see page 47)
- scheduleThread.poolsize (see page 47)
- scheduleThread.queuesize (see page 48)
- sigar.mirror.procnet (see page 48)
- snmpTrapReceiver.listenAddress (see page 48)
- weblogic.auth.method (see page 49)
- weblogic.discovery.new (see page 49)
- weblogic.installpath (see page 49)
- weblogic.ssl2ways.cert (see page 49)
- weblogic.ssl2ways.key (see page 50)
- weblogic.ssl2ways.key.pass (see page 50)
- websphere.installpath (see page 50)
- websphere.useext (see page 51)
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:

```plaintext
(hq self-signed cert), Jul 27, 2011, trustedCertEntry,
hq-agent, Jul 27, 2011, PrivateKeyEntry,
```

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

---

**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

---

**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 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 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:

```java
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.

```java
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` (see page 34) 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 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.

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**

Available only in vFabric Hyperic

**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 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.

---

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=${agent.logLevel}, R
log4j.appenders.R.File=${agent.logFile}
log4j.appenders.R.MaxBackupIndex=1
log4j.appenders.R.MaxFileSize=5000KB
log4j.appenders.R.layout.ConversionPattern=%d{dd-MM-yyyy HH:mm:ss,SSS z} %-5p [%t] [%c{1}@%L] %m%n
log4j.appenders.R.layout=org.apache.log4j.PatternLayout
log4j.appenders.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.hq.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:
This entry is 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 vPostgreSQL 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 vPostgreSQL 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 vPostgreSQL 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 vPostgreSQL 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 PostgreSQL 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 vPostgreSQL 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 PostgreSQL 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**

*This property was added in Hyperic 4.5.1.*

**Description**

The maximum time, in milliseconds, the ScheduleThread will allow a metric collection process 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**

*This property was added in Hyperic 4.5.1.*

**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**

*This property was added in Hyperic 4.5.1.*

**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**

This property was added in Hyperic 4.5.1.

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.procnent**

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 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:

```java
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.

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

```java
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:

```java
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

- Listing of the Agent Wrapper Configuration File (see page 51)
- Tailoring the Agent Wrapper Configuration (see page 53)
  - Define Java Options (see page 53)

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

For related information about agent startup, see About the Agent Launcher and Agent Startup.

For information about the features of the Java Services Wrapper, and how to configure its behavior, see "http://wrapper.tanukisoftware.org/doc/english/download.jsp".

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=-Djava.security.debug
wrapper.java.additional.3=-Djava.security.proxy
wrapper.java.additional.4=-Djava.security.providers
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.
wrapperlogfile=../../log/wrapper.log

# Format of output for the log file. (See docs for formats)
wrapper.logformat=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 log is rolled. Size is specified in bytes. The default value of 0, disnables 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

**Tailoring the Agent Wrapper Configuration**

**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 50256M, 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=-agentlib:ehIJVMInit=stack=256M
wrapper.java.additional.4=-Dsun.net.inetaddr.ttl=60

wrapper.java.additional.5=-Dagent.bundle.home=../../bundles/%HQ_AGENT_BUNDLE%
```
These topics have instructions for configuring the Hyperic® Server.

- Start and Stop Hyperic Server (see page 55)
- Configure Metric Baselining and Alert Processing Behavior (see page 55)
- Scaling and Tuning Hyperic Performance (see page 59)
- Configure Hyperic Server Announcement Behavior (see page 64)
- Integrate Hyperic Server with Other Systems (see page 65)
- Managing the HQ Database (see page 73)
- Clustering Hyperic Servers for Failover (see page 94)
- Hyperic Server Properties (see page 99)
- Tune Hyperic vApp (see page 110)
Start and Stop Hyperic Server

These topics have information about starting and stopping Hyperic Server:

- Starting the Server on Unix-Based Platforms (see page 55)
- Starting the Server on Windows To Run as a Service (see page 55)
- Configure Hyperic Server Java Options (see page 55)

Starting the Server on Unix-Based Platforms

If you installed Hyperic Server from an RPM package, see Starting Hyperic Server After Installing from RPM (see page 55). 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` directory.

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 VMare 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` (see page 101).
Configure Metric Baselining and Alert Processing Behavior

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

- Configure Global Alert Properties (see page 57)
- Configure Alert Notification Throttling (see page 57)
- Configure Alert Notification Email Properties (see page 57)
- Configure Metric Baselining Properties (see page 58)
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

Available only in vFabric Hyperic

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".

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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
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:

```
```

From Email Address

The email address listed as the sender of the alert emails. For example:

```
hq@demo2.vmware.com
```

---

**Configure Metric Baselining Properties**

Available only in **vFabric Hyperic**

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.

<table>
<thead>
<tr>
<th>Server Setting</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline Frequency</td>
<td>The frequency with which HQ calculates a baseline for each metric.</td>
<td>3 days</td>
</tr>
<tr>
<td>Baseline Dataset</td>
<td>The time range of metric data used in calculating the baseline.</td>
<td>7 days</td>
</tr>
<tr>
<td>Baseline Minimum Data Points</td>
<td>The minimum number of data points used in calculating a baseline.</td>
<td>40</td>
</tr>
<tr>
<td>Track Out-of-Bounds Metrics</td>
<td>Controls whether or not HQ tracks OOB metrics.</td>
<td>off</td>
</tr>
</tbody>
</table>
Scaling and Tuning Hyperic Performance

- Sizing Considerations (see page 59)
- Server Configuration Settings for Scaling (see page 59)
- About Java Heap and Garbage Collection (see page 60)
- About Hyperic Server Caches (see page 60)
  - Monitoring Hyperic Caches (see page 61)
  - Interpreting Cache Statistics (see page 62)
  - Configuring Caches (see page 63)

About this page...

This page 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.

Hyperic vApp Tuning

If you run the Hyperic vApp, see Tuning Hyperic vApp

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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Small (less than 50 platforms)</th>
<th>Medium (50-250 platforms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.jms.highmemory</td>
<td>350</td>
<td>1400</td>
</tr>
<tr>
<td>Property</td>
<td>Value 1</td>
<td>Value 2</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>server.jms.maxmemory</td>
<td>400</td>
<td>1600</td>
</tr>
<tr>
<td>server.database-minpoolsize</td>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>server.database-maxpoolsize</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>server.java.opts</td>
<td>-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx512m -Xms512m -XX:+HeapDumpOnOutOfMemory Error -XX:+UseConcMarkSweepGC</td>
<td>-Djava.awt. -XX:MaxPerm -Xmx4g -Xms4g -XX:+UseConcMarkSweepGC</td>
</tr>
<tr>
<td>tomcat.maxthreads</td>
<td>500</td>
<td>2000</td>
</tr>
<tr>
<td>tomcat.minsparethreads</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>org.hyperic.lather.maxConns</td>
<td>475</td>
<td>1900</td>
</tr>
</tbody>
</table>

### 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 of 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.
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.

<table>
<thead>
<tr>
<th>Cache</th>
<th>Size</th>
<th>Hits</th>
<th>Misses</th>
<th>Limit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Agent.findByAgentToken</code></td>
<td>605</td>
<td>26010977</td>
<td>605</td>
<td>5000</td>
<td>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.</td>
</tr>
<tr>
<td>Cache Name</td>
<td>Hits</td>
<td>Misses</td>
<td>Size Limit</td>
<td>Health Status</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>--------</td>
<td>------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td><code>org.hyperic.hq.events.server.session.Alert</code></td>
<td>70526</td>
<td>48049</td>
<td>71274</td>
<td>100000</td>
<td>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.</td>
</tr>
<tr>
<td><code>org.hyperic.hq.events.server.session.AlertDefinition</code></td>
<td>66287</td>
<td>44385</td>
<td>66340</td>
<td>100000</td>
<td>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.</td>
</tr>
<tr>
<td><code>Measurement.findByTemplate ForInstance</code></td>
<td>10000</td>
<td>6766</td>
<td>25772</td>
<td>10000</td>
<td>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 <code>UpdateTimestampsCache</code> and the <code>PermissionCache</code> caches, which contain types that are invalidated frequently.</td>
</tr>
</tbody>
</table>

## 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:

```xml
<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 communicate with other enterprise systems:

- Configure Kerberos Properties (see page 66)
- Configure LDAP Properties (see page 67)
- Configuring Hyperic Server for SMTP Server (see page 69)
- Enable vFabric Hyperic to Send SNMP Traps (see page 70)
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

Available only in vFabric Hyperic

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.

<table>
<thead>
<tr>
<th>LDAP Configuration Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use LDAP Authentication</td>
</tr>
<tr>
<td>Username:</td>
</tr>
<tr>
<td>Search Base: cn=Hypercit.US</td>
</tr>
<tr>
<td>Login Property: on</td>
</tr>
<tr>
<td>Group Search Base:</td>
</tr>
<tr>
<td>Group Search Filter: (member=*)(member=*0)</td>
</tr>
</tbody>
</table>

![Configuration Table Image](image-url)
3. In the enter the properties described below:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use LDAP Authentication</td>
<td>Checkmark this option to enable LDAP authentication.</td>
</tr>
<tr>
<td>URL</td>
<td>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: <code>ldap://YourLDAPHost:44389</code> If your LDAP directory requires SSL, specify the SSL port in the URL.</td>
</tr>
<tr>
<td>SSL</td>
<td>Place a checkmark in the box if your LDAP directory requires SSL connections.</td>
</tr>
<tr>
<td>Username</td>
<td>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.</td>
</tr>
<tr>
<td>Password</td>
<td>Supply the password for the LDAP user specified in &quot;Username&quot; above.</td>
</tr>
<tr>
<td>Search Base</td>
<td>(Required) The &quot;Search Base&quot; 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: <code>ou=people,dc=example,dc=com</code> Consult your LDAP administrator if necessary.</td>
</tr>
<tr>
<td>Search Filter</td>
<td>If desired, enter a filter to limits the LDAP user search to a subset of the object identified by the &quot;Search Base&quot; property. For example, <code>(!(location=SFO*))</code></td>
</tr>
<tr>
<td>Login Property</td>
<td>(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 &quot;cn&quot;. Depending on your LDAP environment, a different property, for instance, &quot;uid&quot;, may be appropriate.</td>
</tr>
<tr>
<td>Group Search Base</td>
<td>Analogous to &quot;Search Base&quot;, 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.</td>
</tr>
<tr>
<td>Search Subtree</td>
<td>If you have configured the &quot;Group Search Base&quot;, described above, you can checkmark this box, to enable search of the entire subtree of the object identified by &quot;Group Search Base&quot;</td>
</tr>
<tr>
<td>Group Search Filter</td>
<td>If you have configured the &quot;Group Search Base&quot;, 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 &quot;Member={0}&quot;, results in filtering by the full distinguished name of a user. To filter by user login name, set &quot;Member={1}&quot; to filter on the login name.</td>
</tr>
</tbody>
</table>

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
     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

Available only in vFabric Hyperic

- Configure HQ Server to Send SNMP Traps (see page 70)
  - Configure vFabric Hyperic Server for SNMP v1 (see page 70)
  - Configure vFabric Hyperic Server for SNMP v2c (see page 70)
  - Configure vFabric Hyperic Server for SNMP v3 (see page 71)
- Using SNMP Traps in Alert Definitions (see page 72)

This page 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.

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Description</th>
<th>Allowable Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP Trap OID</td>
<td>The OID of the notification to be sent. Supplies the value of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>snmpTrapOID.0 - the second varbind in a trap or inform that Hyperic</td>
<td>For v1 of the SNMP protocol, choose V1 Trap. This is the only trap type</td>
</tr>
<tr>
<td></td>
<td>Server generates. (The first varbind is SysUpTime.0.)</td>
<td>you can generate for SNMP v1.</td>
</tr>
<tr>
<td>Default Notification Mechanism</td>
<td>Your selection governs the notification type that will appear as the default notification type option in the &quot;Notification Mechanism&quot; 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.</td>
<td></td>
</tr>
<tr>
<td>Enterprise OID</td>
<td>Enterprise OID.</td>
<td></td>
</tr>
<tr>
<td>Community</td>
<td>The community name to be sent with the trap.</td>
<td>0 - coldStart</td>
</tr>
<tr>
<td>Generic ID</td>
<td>Single digit identifier of the trap type.</td>
<td>1 - warmStart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 - linkDown</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 - linkUp</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4 - authenticationFailure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 - egpNeighborLoss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6 - enterpriseSpecific</td>
</tr>
<tr>
<td>Specific ID</td>
<td>The specific trap code for an enterprise-specific trap (when Generic ID is</td>
<td>set to to 6).</td>
</tr>
<tr>
<td>Agent Address</td>
<td>Address of the managed object that generates the trap.</td>
<td></td>
</tr>
</tbody>
</table>
### Configure vFabric Hyperic Server for SNMP v2c

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Description</th>
<th>Allowable Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP Trap OID</td>
<td>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.)</td>
<td></td>
</tr>
<tr>
<td>Default Notification Mechanism</td>
<td>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.</td>
<td>V1 Trap, V2c Trap, Inform</td>
</tr>
</tbody>
</table>

| 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.

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Description</th>
<th>Allowable Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP Trap OID</td>
<td>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.)</td>
<td></td>
</tr>
<tr>
<td>Default Notification Mechanism</td>
<td>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.</td>
<td>V1 Trap, V2c Trap, Inform</td>
</tr>
</tbody>
</table>

| Security Name           | The username Hyperic's SNMP agent should use when sending notifications to the NMS. | Required. |
| 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 |

<p>| Auth Passphrase         | The SNMP authorization passphrase configured for use when communication with the NMS. |          |</p>
<table>
<thead>
<tr>
<th>Privacy Protocol</th>
<th>The SNMP Privacy Protocol Hyperic Server should use for communication with the NMS.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>▪ none ▪ DES ▪ 3DES ▪ AES-128, ▪ AES-192 ▪ AES-256</td>
</tr>
<tr>
<td>Privacy Passphrase</td>
<td>The SNMP privacy passphrase configured for use when communication with the NMS.</td>
</tr>
<tr>
<td>Context Engine ID</td>
<td>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.</td>
</tr>
<tr>
<td>Context Name</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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 (see page 74)
- Hyperic Database Backup and Recovery (see page 80)
- Configure Hyperic Server Data Compression and Purge Behavior (see page 82)
- Monitoring the Hyperic Database (see page 84)
- Hyperic Database Table Schemas (see page 85)
- MySQL Maintenance Examples (see page 92)
Building a Metric Data Warehouse

Hyperic's retention strategy for measurement data is 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 (see page 74)
- Instructions for Establishing Secondary MySQL Database for Metrics (see page 74)
- Set Up the Secondary Database (see page 74)
- Test the New Views (see page 78)
- Set up the Metric Data Replicator (see page 78)
- Create log4j Properties File (see page 78)
- Create Script to Run the Replication Process (see page 78)
- Run the Replication Process (see page 79)
- Verify the Replication Process Results (see page 79)

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 (see page 85).

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.

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 Hyperic DB name

```sql
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 Hyperic DB name

```sql
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

```sql
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

```sql
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

```sql
create view EAM_MEASUREMENT_TEMPL as
    select ID,
           VERSION_COL,
           NAME,
           ALIAS,
           UNITS,
           COLLECTION_TYPE,
           DEFAULT_ON,
           DEFAULT_INTERVAL,
           DESIGNATE,
           TEMPLATE,
           PLUGIN,
           CTIME,
           MTIME,
           MONITIORABLE_TYPE_ID,
           CATEGORY_ID
    from hqdb.EAM_MEASUREMENT_TEMPL;
```
10. OPTIONAL: Create view for Availability data, replace hqdb with your Hyperic DB name

```sql
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.

```sql
SELECT * FROM EAM_SERVER;
```

**Set up the Metric Data Replicator**

Once your secondary database is ready to store and view the Hyperic 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
```

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

**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:

```properties
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.

```bash
#!/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="$HQ_ROOT/WEB-INF/lib/mysql-connector-java-commercial-5.1.10.jar"  # Database JARs
HQ_PKGS="$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/commons-logging-1.0.4.jar:$HQ_ROOT/WEB-INF/lib/log4j-1.2.14.jar"

PKGS="$PROP_FILES:$DB_PKGS:$HQ_PKG:$LOG_PKG:$HQEE_PKG:$HQUTIL_PKG"
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.
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.
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.

- Backing up Built-In PostgreSQL Database (see page 81)
- Hyperic Files to Backup (see page 81)

⚠️ 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 PostgreSQL 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 server 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.

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

For database configuration options that may be useful in large Hyperic deployments, see the topics in Scaling and Tuning Hyperic Performance (see page 59).
Hyperic Database Table Schemas

Topics marked with * relate to features available only in vFabric Hyperic.

- Key Resource and Measurement Tables (see page 86)
  - EAMRESOURCE Table - All Resource Types and Instances (see page 86)
  - Tables for Inventory Resources (see page 86)
  - Tables for Platform, Server, and Service Types (see page 86)
  - Tables for Measurement Information (see page 86)
- Table Documentation (see page 87)
  - EAMPLATFORM (see page 87)
  - EAMPLATFORMTYPE (see page 87)
  - EAMSERVICE (see page 89)
  - EAMRESOURCE (see page 89)
  - EAMMEASUREMENT (see page 90)
  - EAMMEASUREMENTTEMPL (see page 91)
Key Resource and Measurement Tables

This page 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 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_MEASUREMENT_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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>int4</td>
<td>An ID for the platform, unique among platforms.</td>
</tr>
<tr>
<td>VERSION_COL</td>
<td>int8</td>
<td>Version of the row. Increments when the row is modified. increments with any change to configuration of this row</td>
</tr>
<tr>
<td>FQDN</td>
<td>varchar(200)</td>
<td>Fully qualified domain name of the platform.</td>
</tr>
<tr>
<td>CERTDN</td>
<td>varchar(200)</td>
<td>SSL Certificate for the agent which is monitoring this platform.</td>
</tr>
<tr>
<td>CID</td>
<td>int4</td>
<td>not used</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>varchar(256)</td>
<td>Description of platform.</td>
</tr>
<tr>
<td>CTIME</td>
<td>int8</td>
<td>Creation time of platform.</td>
</tr>
<tr>
<td>MTIME</td>
<td>int8</td>
<td>Last modification time of the platform.</td>
</tr>
<tr>
<td>MODIFIED_BY</td>
<td>varchar(100)</td>
<td>Last modification user.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>varchar(100)</td>
<td>String entered by user, optionally.</td>
</tr>
<tr>
<td>COMMENT_TEXT</td>
<td>varchar(256)</td>
<td>String entered by user, optionally.</td>
</tr>
<tr>
<td>CPU_COUNT</td>
<td>int4</td>
<td>Number of CPUs on this platform.</td>
</tr>
<tr>
<td>PLATFORM_TYPE_ID</td>
<td>int4</td>
<td>ID for the platform type. Points to EAM_PLATFORM_TYPE table.</td>
</tr>
<tr>
<td>CONFIG_RESPONSE_ID</td>
<td>int4</td>
<td>Link to configuration string in plugin xml file.</td>
</tr>
<tr>
<td>AGENT_ID</td>
<td>int4</td>
<td>A unique identifier to the agent which is monitoring this platform.</td>
</tr>
<tr>
<td>RESOURCE_ID</td>
<td>int4</td>
<td>Uniquely identifies the resource, unique across platforms, servers, services. Points to the EAMRESOURCE table.</td>
</tr>
</tbody>
</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.

EAM_SERVER
Contains a row for each server in Hyperic inventory.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>int4</td>
<td>A unique identifier of the server.</td>
</tr>
<tr>
<td>VERSION_COL</td>
<td>int8</td>
<td>A column which increments with any change to configuration of this row.</td>
</tr>
<tr>
<td>CID</td>
<td>int4</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>varchar(300)</td>
<td>Description of server.</td>
</tr>
<tr>
<td>CTIME</td>
<td>int8</td>
<td>Creation time of the server.</td>
</tr>
<tr>
<td>MTIME</td>
<td>int8</td>
<td>Last modification time of the server.</td>
</tr>
<tr>
<td>MODIFIED_BY</td>
<td>varchar(100)</td>
<td>Last modification user</td>
</tr>
<tr>
<td>LOCATION</td>
<td>varchar(100)</td>
<td></td>
</tr>
<tr>
<td>PLATFORM_ID</td>
<td>int4</td>
<td>The Unique ID of the platform on which this server is installed</td>
</tr>
<tr>
<td>AUTOINVENTORYIDENTIFIER</td>
<td>varchar(250)</td>
<td>A unique ID describing this server via the plugin XML.</td>
</tr>
<tr>
<td>RUNTIMEAUTODISCOVERY</td>
<td>bool</td>
<td>Is runtime autodiscovery enabled on this server?</td>
</tr>
<tr>
<td>WASAUTODISCOVERED</td>
<td>bool</td>
<td>Was this server autodiscovered?</td>
</tr>
<tr>
<td>SERVICESAUTOMANAGED</td>
<td>bool</td>
<td>Not used.</td>
</tr>
<tr>
<td>AUTODISCOVERY_ZOMBIE</td>
<td>bool</td>
<td>Were there deletions on the client side for this server?</td>
</tr>
<tr>
<td>INSTALLPATH</td>
<td>varchar(200)</td>
<td>Install path of this server on the platform.</td>
</tr>
<tr>
<td>SERVER_TYPE_ID</td>
<td>int4</td>
<td>Unique ID of the server type that describes this server.</td>
</tr>
<tr>
<td>CONFIG_RESPONSE_ID</td>
<td>int4</td>
<td>Link to configuration string in plugin xml file.</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RESOURCE_ID</td>
<td>int4</td>
<td>Uniquely identifies the resource, unique across platforms, servers, services. Points to the EAM_ RESOURCE table.</td>
</tr>
<tr>
<td>EAM_SERVICE</td>
<td></td>
<td>Contains a row for each service in Hyperic inventory</td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>ID</td>
<td>int4</td>
<td>An ID for the service, unique among services.</td>
</tr>
<tr>
<td>VERSION_COL</td>
<td>int8</td>
<td>A column which increments with any change to configuration of this row</td>
</tr>
<tr>
<td>CID</td>
<td>int4</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>varchar(200)</td>
<td>Description of service.</td>
</tr>
<tr>
<td>CTIME</td>
<td>int8</td>
<td>Creation time of service.</td>
</tr>
<tr>
<td>MTIME</td>
<td>int8</td>
<td>Last modification time of the service.</td>
</tr>
<tr>
<td>MODIFIED_BY</td>
<td>varchar(100)</td>
<td>Last modification user.</td>
</tr>
<tr>
<td>LOCATION</td>
<td>varchar(100)</td>
<td>Not used.</td>
</tr>
<tr>
<td>AUTODISCOVERY_ZOMBIE</td>
<td>bool</td>
<td>Were there deletions on the client side for this service?</td>
</tr>
<tr>
<td>SERVICE_RT</td>
<td>bool</td>
<td>Is response time enabled for this service?</td>
</tr>
<tr>
<td>ENDUSER_RT</td>
<td>bool</td>
<td>Is end user response time enabled for this service?</td>
</tr>
<tr>
<td>PARENT_SERVICE_ID</td>
<td>int4</td>
<td>Unique ID into the parent service for this service.</td>
</tr>
<tr>
<td>SERVER_ID</td>
<td>int4</td>
<td>Were there deletions on the client side for this server?</td>
</tr>
<tr>
<td>AUTOINVENTORYIDENTIFIER</td>
<td>varchar(500)</td>
<td>A unique ID describing this server via the plugin XML.</td>
</tr>
<tr>
<td>SERVICE_TYPE_ID</td>
<td>Int4</td>
<td>Unique ID of service type for this service.</td>
</tr>
<tr>
<td>CONFIG_RESPONSE_ID</td>
<td>int4</td>
<td>Link to configuration string in plugin xml file.</td>
</tr>
<tr>
<td>RESOURCE_ID</td>
<td>int4</td>
<td>Uniquely identifies the resource, unique across platforms, servers, services. Points to the EAM_ RESOURCE table.</td>
</tr>
<tr>
<td>EAM_RESOURCE</td>
<td></td>
<td>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:</td>
</tr>
<tr>
<td>- basic inventory types - Platforms, Servers, and Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- configurable inventory types - Groups and Applications</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Users and Roles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Escalations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Click the thumbnail to see example column data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ID</td>
<td>int4</td>
<td>Uniquely identifies a type or an instance of a type.</td>
</tr>
<tr>
<td>VERSION_COL</td>
<td>int8</td>
<td>Increments with any change to configuration of this row.</td>
</tr>
<tr>
<td>RESOURCE_TYPE_ID</td>
<td>int4</td>
<td>Identifies a type in the Hyperic inventory model. The values this attribute identify a resource as one of the following:</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>int4</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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_TYPE, EAM_RESOURCE_GROUP, EAM_APPLICATION, EAM_ROLE, EAM_SUBJECT, EAM_ESCALATION</td>
</tr>
<tr>
<td>SUBJECT_ID</td>
<td>int4</td>
<td>Identifies the Hyperic user owns the resource.</td>
</tr>
<tr>
<td>PROTO_ID</td>
<td>int4</td>
<td>For a type, value is zero.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For an instance of a type, contains the value of the ID column for the type in this table.</td>
</tr>
<tr>
<td>NAME</td>
<td>varchar(500)</td>
<td>Display name for a resource, for example, &quot;My-Office-MacBook-Pro-15.local JBoss 4.2 default ServiceManager Stateless Session EJB&quot;.</td>
</tr>
<tr>
<td>SORT_NAME</td>
<td>varchar(500)</td>
<td>Same as the NAME column but all in upper case, for example, &quot;MY-OFFICE-MACBOOK-PRO-15.LOCAL JBOSS 4.2 DEFAULT SERVICEMANAGER STATELESS SESSION EJB&quot;.</td>
</tr>
<tr>
<td>FSYSTEM</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>MTIME</td>
<td>int8</td>
<td>Last modification time of the resource.</td>
</tr>
</tbody>
</table>

**EAM_MEASUREMENT**

Each row contains information about a measurement for a resource under management. Click the thumbnail to see example column data.
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>int4</td>
<td>Unique ID for a metric that can be collected for a resource. Points to actually measurements in EAM_MEASUREMENT_DATA_* tables.</td>
</tr>
<tr>
<td>VERSION_COL</td>
<td>int8</td>
<td>Indicates version of the row, increments upon each change to the row.</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>int4</td>
<td>The resource type the measurement is for. Unique identifiers a resource type of a given inventory level - platform, server, service. For example, the ID 10001 uniquely identifies the platform type &quot;MacOSX&quot;.</td>
</tr>
<tr>
<td>TEMPLATE_ID</td>
<td>int4</td>
<td>ID of a template points to the EAM_MEASUREMENT_TEMPL table.</td>
</tr>
<tr>
<td>MTIME</td>
<td>int8</td>
<td>Time modified.</td>
</tr>
<tr>
<td>ENABLED</td>
<td>boolean</td>
<td>Is this metric enabled?</td>
</tr>
<tr>
<td>COLL_INTERVAL</td>
<td>int8</td>
<td>How often this metric is collected.</td>
</tr>
<tr>
<td>DSN</td>
<td>varchar(2048)</td>
<td>A string which describes the measurement from the plugin-xml text.</td>
</tr>
<tr>
<td>RESOURCE_ID</td>
<td>int4</td>
<td>Uniquely identifies the resource for which the metric is associated, unique across platforms, servers, services. Points to the EAM_RESOURCE table.</td>
</tr>
</tbody>
</table>

### EAM_MEASUREMENT_TEMPL

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>int4</td>
<td>A unique identifier of a measurement template for a metric for a resource.</td>
</tr>
<tr>
<td>VERSION_COL</td>
<td>int8</td>
<td>A column which increments with any change to configuration of this row</td>
</tr>
<tr>
<td>NAME</td>
<td>varchar(100)</td>
<td>Name of this measurement template.</td>
</tr>
<tr>
<td>ALIAS</td>
<td>varchar(100)</td>
<td>String that describes the alias portion of XML file.</td>
</tr>
<tr>
<td>UNITS</td>
<td>varchar(50)</td>
<td>Units of this measurement.</td>
</tr>
<tr>
<td>COLLECTION_TYPE</td>
<td>int4</td>
<td>Static/Dynamic data.</td>
</tr>
<tr>
<td>DEFAULT_ON</td>
<td>bool</td>
<td>Does this measurement collect by default?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>DEFAULT_INTERVAL</td>
<td>int8</td>
<td>The default collection interval of this metric.</td>
</tr>
<tr>
<td>DESIGNATE</td>
<td>bool</td>
<td>Is this metric on the indicator page by default?</td>
</tr>
<tr>
<td>TEMPLATE</td>
<td>varchar(2048)</td>
<td>Template string from plugin XML.</td>
</tr>
<tr>
<td>PLUGIN</td>
<td>varchar(250)</td>
<td>Name of the plugin which houses this measurement template.</td>
</tr>
<tr>
<td>CTIME</td>
<td>int8</td>
<td>Creation time of server.</td>
</tr>
<tr>
<td>MTIME</td>
<td>int8</td>
<td>Last modification time of the server.</td>
</tr>
<tr>
<td>MONITORABLE_TYPE_ID</td>
<td>int4</td>
<td>Key into the monitorable type data.</td>
</tr>
<tr>
<td>CATEGORY_ID</td>
<td>int4</td>
<td>Key into the category ID table.</td>
</tr>
</tbody>
</table>

**MySQL Maintenance Examples**

Here are examples of regular maintenance for mysql

1. Simple MySQL Backup Script

```
#!/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"
```

2. 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

```
```
3. 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

**Available only in vFabric Hyperic**

- Overview (see page 94)
- Requirements for an Failover Deployment (see page 94)
- Configuring a Server Cluster (see page 94)
  - Step 1 - Install the First Hyperic Server Instance (see page 95)
  - Step 2 - Install Additional Hyperic Server Nodes (see page 95)
  - Step 3 - Configure Cluster Name and Communications Properties (see page 95)
  - Step 4 - Configure the Load Balancer (see page 97)
  - Step 5 - Configure Agents to Communicate with Hyperic Server Cluster (see page 97)
  - Step 6 - Start the Nodes (see page 97)
- Troubleshooting a Failover Configuration (see page 97)

**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:

- EHCach'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 a 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 will return a response of `master=true` for the primary node, and a response of `master=false` 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 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

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ha.partition</td>
<td>Name of the cluster---this value is identical for each node in the cluster</td>
</tr>
<tr>
<td>ha.node.address</td>
<td>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.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ha.node.mcast_addr and ha.node.mcast_port</td>
<td>Address and port for sending multicast messages to other nodes. Note: ha.node.mcast_addr must be the same on each node.</td>
</tr>
<tr>
<td>ha.node.cacheListener.port and ha.node.cacheProvider.port</td>
<td>Ports used for discovering and synchronizing with cache peers.</td>
</tr>
</tbody>
</table>

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 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 --- The 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

- About Hyperic Server Configuration (see page 99)
  - Configuration Settings in hq-server.conf (see page 99)
  - Configuration Settings in the Database (see page 99)
- Server Property Definitions (see page 99)
- Clustering Properties in vFabric Hyperic (see page 100)

About Hyperic Server Configuration

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 (see page 100)
- server.caf.brokerAddress (see page 101)
- server.caf.clientId (see page 101)
- server.database-driver (see page 101)
- server.java.opts (see page 101)
- server.quartzDelegate (see page 102)
- server.database-url (see page 102)
- server.connection-validation-sql (see page 103)
- server.database-password (see page 103)
- server.database-user (see page 103)
- server.encryption-key (see page 103)
- server.webapp.port (see page 103)
Cluster Properties in vFabric Hyperic

For information about properties for configuring a Hyperic Server cluster, see Clustering Hyperic Servers for Failover (see page 94).

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 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 agent'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.

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* (see page 59).

**Default**

In Hyperic HQ the default value is:
server.java.opts=-Djava.awt.headless=true -XX:MaxPermSize=192m -Xms512m -Xmx512m
-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 -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.

<table>
<thead>
<tr>
<th>If you select...</th>
<th>the default database URL is...</th>
</tr>
</thead>
<tbody>
<tr>
<td>HQ Built-in Database PostgreSQL</td>
<td><code>postgresql://127.0.0.1:9432/hqdb?protocolVersion=2</code></td>
</tr>
</tbody>
</table>
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. 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
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 Server'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
```

change it to:

```
/C:/Documents\ and\ Settings/Desktop/keystore
```
Best Practices for Hyperic Keystores

Please see:
- Hyperic Security Features and Recommendations
- Configure SSL Options

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.
**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` (see page 59).

**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 minimum 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 (see page 59).

**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 of other server properties on Hyperic performance, see Scaling and Tuning Hyperic Performance (see page 59).

**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 (see page 59)`.

**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 (see page 59)`.

**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 (see page 59)`.
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 (VCOPS} 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.

Tune Hyperic vApp

- Operating System Settings (see page 111)
- Hyperic Server Settings (see page 111)
Hyperic Database Settings (see page 111)

This page 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:

<table>
<thead>
<tr>
<th></th>
<th>soft</th>
<th>nofile</th>
</tr>
</thead>
<tbody>
<tr>
<td>hyperic</td>
<td>8192</td>
<td>16384</td>
</tr>
</tbody>
</table>

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
```