vmware

Configure and Run the HQ Server

Version:	8
Date:	11/19/10
Creator:	vFabric Documentation Team

Table of Contents

1. Start and Stop HQ Server 3
    1.1. Starting the Server on Unix-Based Platforms 3
    1.2. Starting the Server on Windows To Run as a Service 3
    1.3. Troubleshooting HQ Server Startup Problems 3
2. Configure HQ Server Communications and Connections 4
    2.1. Updating SSL Certs for HQ Server 5
    2.2. Configure User Interface Help Location 6
    2.3. Configure the Maximum Number of Agent - Server Connections 7
    2.4. Configure Size of HQ Server Database Connection Pool 8
    2.5. Configure HQ Version and Security Announcements 9
    2.6. Configure HQ Server Alert Notification Email Properties 10
3. Configure HQ Server Memory, Garbage Collection, and Caches 11
    3.1. Configure HQ Server Caches for Improved Performance 12
    3.2. Configure HQ Server Heap and Memory Settings 14
4. Configure Metric Baselining and Alert Processing Behavior 15
    4.1. Configure Global Alert Properties 16
    4.2. Configure Alert Notification Throttling 17
    4.3. Configure Metric Baselining Properties 18
5. Managing the HQ Database 19
    5.1. HQ Database Backup and Recovery 20
    5.2. Building a Metric Data Warehouse 21
    5.3. Configure HQ Server Data Compression and Purge Behavior 30
    5.4. Monitoring the HQ Database 32
6. Configuring HQ for Large Environments and Improved Performance 33
    6.1. Increasing Java Heap and Changing GC Settings 33
    6.2. Increase JMS Memory 33
    6.3. Increase Max Lather Connections 33
    6.4. Increase the Number Connections to Database 34
    6.5. Configure HQ Cache Settings for Improved Performance 34
7. Integrate HQ Server with Other Systems 36
    7.1. Configure LDAP Properties 37
    7.2. Configure Kerberos Properties 38
    7.3. Configuring HQ Server for SMTP Server 39
    7.4. Enable Hyperic Enterprise to Send SNMP Traps 41
    7.5. Configure HQ Server for Hyperic IQ 44
8. Clustering HQ Servers for Failover 45
    8.1. Overview 45
    8.2. Requirements for an Failover Deployment 45
    8.3. Configuring a Server Cluster 45
9. Server Properties 49
    9.1. About HQ Server Configuration 50
    9.2. Server Property Definitions 51
10. HQ Database Table Schemas 55
    10.1. Key Resource and Measurement Tables 56
    10.2. Table Documentation 57

1. Start and Stop HQ Server

These topics have information about starting and stopping HQ Server:

1.1. Starting the Server on Unix-Based Platforms

Start the server with this command:

```
<Server Installation Directory>/bin/hq-server.sh start
```

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

The server.log file and and bootstrap.log files in the logs directory under the server installation directory will have detailed startup information.

1.2. 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.

1.3. Troubleshooting HQ Server Startup Problems

In HQ Enterprise, the HQ Server will not start if it does not find a valid `license.xml` file in its `/conf` directory.

2. Configure HQ Server Communications and Connections

These topics have instructions for configuring HQ Server communications with the HQ database and HQ Agents:

2.1. Updating SSL Certs for HQ Server

When HQ is configured to use SSL for agent-server communications, it uses the built-in certs in `hq-engine/hq-server/conf/hyperic.keystore`. You can replace the built-in certs with your own.

If your certificate is not PKCS12 format, you can use the **openssl** tool to convert it. For more information, see http://community.jboss.org/wiki/sslsetup.

After ensuring your certificate is in the correct format, use **java-keytool** to install it. For more information, see http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html.

2.2. Configure User Interface Help Location

- **Context-Sensitive Help** - This property, in the **Announcement Properties** section of the **Administration > HQ Server Settings** page, defines the location from which help pages will be served when an HQ user clicks **Help**. When set to "Remote", help pages will be served from the Hyperic documentation wiki. When set to "Internal", the help pages that are embedded in the HQ Server will be served.

2.3. Configure the Maximum Number of Agent - Server Connections

Lather is HQ's connection protocol for communication between the agent and the server. By default, the maximum number of connections to agents the server will allow open is 3000. If you have a large number of agents, you can increase the number of connections. Set the number of connections to 1.2 times the number of agents, in this file:

`"ServerHome/hq-engine/hq-server/webapps/ROOT/WEB-INF/web.xml`

in this stanza:

```
<init-param>
  <param-name>org.hyperic.lather.maxConns</param-name>
  <param-value>3000</param-value>
</init-param>
```

2.4. Configure Size of HQ Server Database Connection Pool

The size of the connection pool for communication between HQ Server and the database is configured with `server.database-minpoolsize` and `server.database-maxpoolsize` in `Server-Home/conf/hq-server.conf`, by default, 5 and 100, respectively.

In large environments, it may be appropriate to increase the size of the connection pool. If you are managing between 800 and 1500 platforms, set both `server.database-minpoolsize` and `server.database-maxpoolsize` to 300.

Given properly configured caches, it should not be necessary to increase connection pool size to more than 700. For more information, see Section 3.1, “Configure HQ Server Caches for Improved Performance”

Note: If you increase `server.database-minpoolsize` to 100 or more, set `server.database-maxpoolsize` equal to that value.

2.5. Configure HQ Version and Security Announcements

- **HQ Version and Security Announcements** - This property, in the **Announcement Properties** section of the **Administration > HQ Server Settings** page controls what security and version announcements will be sent to HQ Administrators:
  - All
  - Major
  - None

2.6. Configure HQ Server Alert Notification Email Properties

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

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

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

3. Configure HQ Server Memory, Garbage Collection, and Caches

These topics have information about configuring HQ internals for improved performance in large environments:

3.1. Configure HQ Server Caches for Improved Performance

The default cache settings in HQ are suitable for most deployments. However, depending on your deployment and use of HQ, especially in large environments, changing the cache settings might improve performance.

3.1.1. Where Are Caches Kept

All internal caches are kept in EHCache and are accessible via the "HQ Health" screen. EHCache provides:

- Automatic integration with Hibernate
- Optional distributed caching when running in a cluster.
- A variety of proven, well-tested eviction algorithms
- Better cache management and configuration

3.1.2. Default Cache Settings

By default, the cache settings in HQ are set for a medium to large deployment:

- 100 Platforms
- 500 Servers
- 5000 Services

These settings should suffice for most deployments.

3.1.3. Monitoring Caches to Identify Areas for Improvement

You can monitor HQ's internal caches through the server.log or via the "HQ Health" screen. The log contains diagnostic information, which is written to the log periodically. The cache information should look like:

```
Cache Size Hits Misses
===== ==== ==== =====
Agent.findByAgentToken 2 4184 2
AgentScheduleInQueue 0 0 0
Alert.findByCreateTime 10 318 694
Alert.findByEntity 0 0 0
....
```

This listing shows each cache, its size, and its hit counts. At the end of the listing there is a summary for the total size of all HQ caches.

If a cache has an unusually high miss rate or becomes full, you can manually configure it to improve performance. For example, consider this cache:

```
Measurement.findByTemplateForInstance 10000 6766 25772
```

The cache has 10,000 elements. With only 6766 hits and 25,772 misses, the hit ratio is around 20-25%. Ideally the number of misses should peak at about the maximum size of the cache. (There are some exceptions to this:

the UpdateTimestampsCache and the PermissionCache have items invalidated from the cache frequently, so these rules do not apply.) So, increasing this cache's size would probably improve HQ performance.

3.1.4. Changing Caches Settings

After you identify a cache whose configuration should be changed, you can change its configuration in the file <Server installation directory>/hq-engine/hq-server/webapps/ROOT/WEB-INF/classes/ehcache.xml. In general, only the cache sizes should need to be changed.

To continue with the example cache from above, the entry for the cache in this file looks like this:

```
<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.

The format of this file is described in EHCache's documentation.

3.1.5. Configuring Permission Cache for Improved Performance *

The PermissionCache can be changed to improve performance. This cache caches recently evaluated permission checks:

```
<cache name="PermissionCache"
maxElementsInMemory="1000"
timeToIdleSeconds="0"
timeToLiveSeconds="60"
memoryStoreEvictionPolicy="LRU"/>
```

This cache is set to expire the checks after 60 seconds. In an environment where the user and role definitions are not updated frequently, you can increase the timeToLiveSeconds value to effect a significant performance increase.

While the right value for timeToLiveSeconds depends on the environment, the rule of thumb is that it should be set no higher than the acceptable time for a permission-related operation to take effect. Perhaps this is even zero, but that can be costly because the Server needs to go to the database to check permission on every resource the user views.

3.2. Configure HQ Server Heap and Memory Settings

3.2.1. Increase Java Heap and Change GC Settings

Heap size startup options are set in the `ServerHome/conf/hq-server.conf` file using the `server.java.opts` property. Default settings are described in server.java.opts.

How much you can increase the default heap options depends on the amount of RAM on the HQ Server host. Given sufficient RAM, you could use these settings:

```
server.java.opts=-XX:MaxPermSize=192m -Xmx4096m -Xms4096m -XX:+UseConcMarkSweepGC
```

On Windows, restart the HQ Server Windows service after modifying server.java.opts.

If you increase heap size, you may need to change JMS memory settings as well. For more information, see the Increase JMS Memory Settings below.

Note: If you are running HQ Server on a 64-bit system with 4GB (4096 MB) or less memory, Hyperic recommends you use 32-bit JVM. A 64-bit JVM is not recommended unless you have more memory. You might need twice as much heap on a 64-bit system as on a a 32-bit system to achieve the same performance.

3.2.2. Increase JMS Memory Settings

The JMS memory settings are configured using `server.HighMemory` and `server.MaxMemory` in `ServerHome/conf/hq-server.conf`, by default, 350MB and 400MB, respectively.

The values for `server.HighMemory` and `server.MaxMemory` should be 80% and 90% of the Java heap size, respectively. Erratic alert behavior or missed alerts may indicate the settings are too low.

4. Configure Metric Baselining and Alert Processing Behavior

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

4.1. 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 Understanding Hierarchical Alerting.

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 Configure Network Host Dependencies for Hierarchical Alerting.

4.2. 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 Controlling Alert and Notification Volume.

4.3. Configure Metric Baselining Properties

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

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

5. Managing the HQ Database

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

5.1. HQ Database Backup and Recovery

The HQ database contains most of the data necessary to recreate your HQ 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 HQ 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 HQ, your database server, or other software that resides on the server machine.

You should define HQ 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.

5.1.1. Backing up Built-In PostgreSQL Database

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

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

Copy the dump file to your backup location.

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

5.1.2. Backup and Recovery of MySQL and Oracle Databases

If you use MySQL or Oracle for your HQ database, extend your existing database backup and recovery procedures to the HQ database. Perform a full database backup on a scheduled basis.

5.1.3. HQ Files to Backup

In addition to the HQ 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 HQ Server is running.

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

5.2. Building a Metric Data Warehouse

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

To support requirements for trend analysis over a longer timeframe, HQ versions 3.2 and later provide the MetricDataReplicator class, which you can use to replicate uncompressed measurement data in a secondary database.

5.2.1. 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 HQ database. The secondary database contains one table, EAM_MEASUREMENT_DATA.
- The secondary database has a database link to the primary HQ database, and five views that point to the primary HQ 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 HQ database. These are the views that are required on the secondary database:
  - EAM_PLATFORM
  - EAM_SERVER
  - EAM_SERVICE
  - EAM_MEASUREMENT_TEMPL
  - EAM_MEASUREMENT

For documentation on these database tables, see Section 10, “HQ Database Table Schemas”.

5.2.2. Instructions for Establishing Secondary PostgreSQL Database for Metrics

These sections below have instructions for configuring a secondary HQ database for metrics on PostgreSQL.

MySQL and Oracle have similar, but not identical, methods for creating and using database links. If you use one of these database servers, and are familiar with its database linking facilities, you can use the instructions below as guide, but the details of how you perform some of the steps will depend on the database you use.

Note: MySQL's query optimizer has limitations that have negative impact on the the performance of HQ's metric replicator class. This performance degradation can have a ripple effect on the performance of your primary HQ database during metric replication.

5.2.3. Set Up the Secondary Database

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

1. Install PostgreSQL on a different machine than your primary database.

2. Install the PostgreSQL dblink component, which is required to create a link from the secondary database to the primary HQ database.

```
$ cd postgresql-8.x.x/contrib/dblink/
$ make
$ make install
```

> **Finding dblink on Debian**
>
> If you have trouble finding dblink on a Debian-based Linux system, try:
>
> `$ apt-cache search postgresql`
>
> You should find the `postgresql-contrib-x.y` package (where x.y is the PostgreSQL version), which contains dblink. To install the package:
>
> `$ apt-get install postgresql-contrib`

3. Create the secondary database. These commands create a database named "hqdata", the "hqadmin" user, and make that user the owner of the secondary database:

```
$ psql -d postgres
postgres =# create user hqadmin with encrypted password '<passwd>';
postgres =# create database hqdata owner = hqadmin;
postgres =# q
```

4. Read the `dblink.sql` file into PostgreSQL to enable the secondary database to access data in the primary HQ database.

```
$ psql hqdata < $PGHOME/contrib/dblink.sql
```

Where `$PGHOME` is the PostgreSQL installation directory, for example, `/usr/share/pgsql`.

5. Configure PostgreSQL network access to allow remote queries.

```
$ vi $PGDATA/pg_hba.conf
```

The `$PGDATA` variable represents the path where the databases are being stored. By default this is `/var/lib/pgsql/data`.

6. Edit the `pg_hba.conf` file to include an entries for the primary database and the secondary database.

```
# TYPE DATABASE USER CIDR-ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all trust
# IPv4 local connections:
host all all 127.0.0.1/32 trust
# ADD THE FOLLOWING TWO HOST ENTRIES}}
# main hq database ip}}
host all all 10.2.0.206/32 trust
# replicated hq database ip (even though loop back is already specified)
host all all 10.2.0.100/32 trust
```

Note: In the entries for the "main hq database ip" and "replicated hq database" replace the IP addresses (10.2.0.206/32 and 10.2.0.100/32) to those of your primary and secondary databases respectively.

7. Create database tables.

a) Create table for raw measurements:

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

b) Create view for measurements instances

```
create view EAM_MEASUREMENT as
select * from dblink(
'host=10.2.0.206 dbname=hqdb user=hqadmin password=hqadmin',
'select id,
VERSION_COL,
INSTANCE_ID,
TEMPLATE_ID,
MTIME,
ENABLED,
COLL_INTERVAL,
DSN
from eam_measurement')
as remote_eam_measurement
(
ID int,
MEASUREMENT_CLASS char(1),
VERSION_COL bigint,
INSTANCE_ID int,
TEMPLATE_ID int,
MTIME bigint,
ENABLED boolean,
COLL_INTERVAL bigint,
DSN varchar(2048)
);
```

c) Create view for platforms.

```
create view EAM_PLATFORM as
select * from dblink(
'host=10.2.0.206 dbname=hqdb user=hqadmin password=hqadmin',
'select id,
VERSION_COL,
FQDN,
CERTDN,
DESCRIPTION,
CTIME,
MTIME,
MODIFIED_BY,
LOCATION,
COMMENT_TEXT,
CPU_COUNT,
PLATFORM_TYPE_ID,
CONFIG_RESPONSE_ID,
AGENT_ID
from EAM_PLATFORM')
as remote_eam_platform
(
ID int4,
VERSION_COL int8,
FQDN varchar(200),
```

```
CERTDN varchar(200),
DESCRIPTION varchar(256),
CTIME int8,
MTIME int8,
MODIFIED_BY varchar(100),
LOCATION varchar(100),
COMMENT_TEXT varchar(256),
CPU_COUNT int4,
PLATFORM_TYPE_ID int4,
CONFIG_RESPONSE_ID int4,
AGENT_ID int4
);
```

d) Create view for servers.

```
create view EAM_SERVER as
select * from dblink(
'host=10.2.0.206 dbname=hqdb user=hqadmin password=hqadmin',
'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
from EAM_SERVER')
as remote_eam_server
(
ID int4,
VERSION_COL int8,
DESCRIPTION varchar(300),
CTIME int8,
MTIME int8,
MODIFIED_BY varchar(100),
LOCATION varchar(100),
PLATFORM_ID int4,
AUTOINVENTORYIDENTIFIER varchar(250),
RUNTIMEAUTODISCOVERY bool,
WASAUTODISCOVERED bool,
SERVICESAUTOMANAGED bool,
AUTODISCOVERY_ZOMBIE bool,
INSTALLPATH varchar(200),
SERVER_TYPE_ID int4,CONFIG_RESPONSE_ID int4
);
```

e) Create view for services.

```
create view EAM_SERVICE as
select * from dblink(
'host=10.2.0.206 dbname=hqdb user=hqadmin password=hqadmin',
'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
from EAM_SERVICE')
as remote_eam_service
(
ID int4,
VERSION_COL int8,
DESCRIPTION varchar(200),
CTIME int8,
MTIME int8,
MODIFIED_BY varchar(100),
LOCATION varchar(100),
AUTODISCOVERY_ZOMBIE bool,
SERVICE_RT bool,
ENDUSER_RT bool,
PARENT_SERVICE_ID int4,
SERVER_ID int4,
SERVICE_TYPE_ID int4,
CONFIG_RESPONSE_ID int4
);
```

f) Create view for measurement templates.

```
create view EAM_MEASUREMENT_TEMPL as
select * from dblink(
'host=10.2.0.206 dbname=hqdb user=hqadmin password=hqadmin',
'select id,
VERSION_COL,
NAME,
ALIAS,
UNITS,
COLLECTION_TYPE,
DEFAULT_ON,
DEFAULT_INTERVAL,
DESIGNATE,
TEMPLATE,
PLUGIN,
CTIME,
MTIME,
MONITORABLE_TYPE_ID,
CATEGORY_ID
from EAM_MEASUREMENT_TEMPL')
as remote_eam_measurement_templ
(
ID int4,
VERSION_COL int8,
NAME varchar(200),
ALIAS varchar(50),
UNITS varchar(50),
COLLECTION_TYPE int4,
DEFAULT_ON bool,
DEFAULT_INTERVAL int8,
DESIGNATE bool,
TEMPLATE varchar(2048),
PLUGIN varchar(250),
CTIME int8,
MTIME int8,
MONITORABLE_TYPE_ID int4,
CATEGORY_ID int4
);
```

5.2.4. 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 PSQL prompt of the secondary database and to list all of the servers. This query runs against the primary database.

```
SELECT *
FROM EAM_SERVER
```

5.2.5. Set up the Metric Data Replicator

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

Property Setting	**Description**
pri_user=hqadmin	Primary database username
pri_pass=hqadmin	Primary database password
pri_url=jdbc:postgresql://<ipaddress>:<port>/hqdb?protocolVersion=2	Connection string for primary server, including IP Address and port
sec_user=hyperic	Secondary database username
sec_pass=hyperic	Secondary database password
sec_url=jdbc:postgresql://<ipaddress>:<port>/hqdata?protocolVersion=2	Connection string for secondary server, including IP Address and port.

5.2.6. Create log4j Properties File

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

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

5.2.7. 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 HQ Server installations respectively.

```
#!/bin/bash

#update these params
JAVA_HOME="/usr/jdk/java-1.4"
SERVER_HOME="/usr/local/hyperic/ServerHomeDir/hq-engine/hq-server"

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

```
HQ_WAR="${SERVER_HOME}/webapps/ROOT/WEB-INF/lib"

# path to the prop files
# update to reflect appropriate database
PROP_FILES="."
DB_PKGS="${HQ_WAR}/postgresql-8.2-505.jdbc3.jar"
HQ_PKGS="${HQ_WAR}/hqee-server-4.5.0.BUILD-SNAPSHOT.jar"
LOG_PKGS="${HQ_WAR}/commons-logging-1.0.4.jar:${HQ_WAR}/log4j-1.2.14.jar"
PKGS="${PROP_FILES}:${DB_PKGS}:${HQ_PKGS}:${LOG_PKGS}"
ARGS="$LOG_ARGS -Dreplprops=$PROPS -Djdbc.drivers=org.postgresql.Driver -cp ${PKGS}"
JAVA="${JAVA_HOME}/bin/java"

set -x

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

5.2.8. Run the Replication Process

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

```
run.sh
```

5.2.9. 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.name, shown in bold below with a valid platform name in your resource inventory.

```
Select p.fqdn,
svc.name,
s.name,
t.name,
d.value,
d.timestamp
from EAM_MEASUREMENT_TEMPL t,
EAM_MEASUREMENT m,
EAM_MEASUREMENT_DATA d,
EAM_SERVICE svc,
EAM_SERVER s,
EAM_PLATFORM p
where t.id = m.template_id
and m.id = d.measurement_id
and p.id = s.platform_id
and s.id = svc.server_id
and svc.id = m.instance_id
and lower(p.fqdn) = '*your.platform.name*'
and lower(t.name) like '%disk%' \* lower(t.alias) works here as well */
order by d.timestamp desc;
```

Query 2: Availability/CPU/Memory/Network Information Per Platform

The following query returns the Availability metrics for a named Platform (EAM_PLATFORM.FQDN). The relationship between Platform and detailed metric (EAM_MEASUREMENT_DATA) is made via the measurement header table (EAM_MEASUREMENT). The INSTANCE_ID column of EAM_MEASUREMENT

matches multiple object types (e.g. Platform, Server, Service), so joining on a given objects ID will allow any detailed metric data available for that object to be selectable. The metric template table (EAM_MEASUREMENT_TEMPL) acts as a filter for specific metric types. This table contains all the metric types in the system, so full or partial matching on the name of the template will filter results for that type of metric. There is no specific "metric type" column that allows for true categorical filtering, so matching on name is currently the best method of filtering.

```
SELECT platform.fqdn as platform
,template.name as template
,data.value
,data.timestamp
FROM EAM_MEASUREMENT_TEMPL template
,EAM_MEASUREMENT measurement
,EAM_MEASUREMENT_DATA data
,EAM_PLATFORM platform
WHERE 1=1
AND template.id = measurement.template_id
AND measurement.id = data.measurement_id
AND platform.id = measurement.instance_id
AND lower(platform.fqdn) = 'your.platform.name'
AND lower(template.name) = 'availability'
ORDER BY data.timestamp desc;
```

Query 3: Disk usage per FileServer Service

In the following query, the import join is Platform -> Server -> Service (highlighted). This relationship allows you to "go up the stack" to select metrics from Servers or Services of interest. In most cases, Platform metrics are very specific to the hardware attributes of a given system running the HQ agent (e.g. CPU, Memory, Network) but disk metrics are a bit different. Disks are seen as Services of the FileServer Server type. Therefore disk metrics are accessible from the Services layer of a given Platform. As you can see, the join from Service to metric is the same as Platform to metric (on object ID and measurement INSTANCE_ID).

```
SELECT platform.fqdn as platform
,service.name as service
,server.name as server
,template.name as template
,data.value
,data.timestamp
FROM EAM_MEASUREMENT_TEMPL template
,EAM_MEASUREMENT measurement
,EAM_MEASUREMENT_DATA data
,EAM_SERVICE service
,EAM_SERVER server
,EAM_PLATFORM platform
WHERE 1=1
AND template.id = measurement.template_id
AND measurement.id = data.measurement_id
AND platform.id = server.platform_id
AND server.id = service.server_id
AND service.id = measurement.instance_id
AND lower(platform.fqdn) = 'your.platform.name'
AND lower(template.name) like '%disk%'
ORDER BY data.timestamp desc;
```

Query 4: Component Service Usage Information for Servers and Services

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

applications running within the container are services. Metrics specific to the JBoss container are available from the server layer while the internal web applications are available via the service layer.

```
SELECT platform.fqdn as platform
,service.name as service
,server.name as server
,template.name as template
,data.value
,data.timestamp
FROM EAM_MEASUREMENT_TEMPL template
,EAM_MEASUREMENT measurement
,EAM_MEASUREMENT_DATA data
,EAM_SERVICE service
,EAM_SERVER server
,EAM_PLATFORM platform
WHERE 1=1
AND template.id = measurement.template_id
AND measurement.id = data.measurement_id
AND platform.id = server.platform_id
AND server.id = service.server_id
AND service.id = measurement.instance_id
AND lower(platform.fqdn) = 'your.platform.name'
AND lower(server.name) like '%jboss%'
AND lower(template.name) like '%transaction count%'
ORDER BY data.timestamp desc;
```

5.3. Configure HQ Server Data Compression and Purge Behavior

5.3.1. HQ 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 — HQ Server stores detailed metric data (all data points reported) in the HQ 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. HQ Server retains the archived metric data for 2 years.
- Purge alert data — HQ Server retains fired alert data for a configurable period, after which the alerts are deleted.
- Purge event data — HQ Server retains event data for a configurable period, after which the events are deleted.
- Rebuild metric table indexes — During normal HQ operation, the metric data tables in the HQ database contain a lot of frequently changing data. The HQ Server rebuilds the metric table indexes on a (configurable) periodic basis to avoid performance problems that heavily fragmented indexes can cause.

5.3.2. Configure HQ Data Management Settings

You can configure how HQ condenses and purges the contents of the HQ 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 HQ performance.

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

	ly, HQ re-indexes the tables around midnight.		
Delete Alerts Older Than	Controls how long HQ stores alert event data.	31 days	
Delete Events and Logs Older Than	Controls how long HQ stores other HQ event and log data.	31 days	

Warning: Data Management Changes Require Server Restart

Any changes made in this section require the HQ Server to be restarted before they take effect.

5.4. Monitoring the HQ Database

HQ administrators can view real-time HQ 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 HQ internals experts; Hyperic support engineers may use the HQ Health data and diagnostics to diagnose and troubleshoot HQ Server and database problems. For more information, see the associated help page.

For database configuration options configuration options that may be useful in large HQ deployments, see the topics in Section 6, “Configuring HQ for Large Environments and Improved Performance”.

6. Configuring HQ for Large Environments and Improved Performance

The size of the environment a single HQ Server can manage is determined by the hardware the server is installed on. As a general rule, assume a minimal system configuration can support up to 25 Agents, and the ideal configuration can support 100 Agents or more, depending on the size of the database used.

6.1. Increasing Java Heap and Changing GC Settings

Heap size startup options are set in the `ServerHome/conf/hq-server.conf` file using the `server.java.opts` property. Default settings are described in server.java.opts.

How much you can increase the default heap options depends on the amount of RAM on the HQ Server host. Given sufficient RAM, you could use these settings:

```
server.java.opts=-XX:MaxPermSize=192m -Xmx4096m -Xms4096m -XX:+UseConcMarkSweepGC
```

If you increase heap size, you may need to change JMS memory settings as well. For more information, see the Increase JMS Memory Settings below.

Note: If you are running HQ Server on a 64-bit system with 4GB (4096 MB) or less memory, Hyperic recommends you use 32-bit JVM. A 64-bit JVM is not recommended unless you have more memory. You might need twice as much heap on a 64-bit system as on a a 32-bit system to achieve the same performance.

6.2. Increase JMS Memory

JMS memory is configured using `server.jmx.MaxMemory` in `ServerHome/conf/hq-server.conf`, by default 400MB.

The value of `server.MaxMemory` should be 90% of the Java heap size. Erratic alert behavior or missed alerts may indicate the settings are too low.

6.3. Increase Max Lather Connections

Lather is HQ's connection protocol for communication between the agent and the server. By default, the maximum number of connections to agents the server will allow open is 3000. If you have a large number of agents, you can increase the number of connections. Set the number of connections to 1.2 times the number of agents, in this file:

`<Server installation directory>/hq-engine/hq-server/webapps/ROOT/WEB-INF/classes/ehcache.xml`

in this stanza:

```
<init-param>
       <param-name>org.hyperic.lather.maxConns</param-name>
       <param-value>3000</param-value>
</init-param>
```

6.4. Increase the Number Connections to Database

The size of the connection pool for communication between HQ Server and the database is configured with `server.database-minpoolsize` and `server.database-maxpoolsize` in `Server-Home/conf/hq-server.conf`, by default, 5 and 100, respectively.

In large environments, it may be appropriate to increase the size of the connection pool. If you are managing between 800 and 1500 platforms, set both `server.database-minpoolsize` and `server.database-maxpoolsize` to 300.

Given properly configured caches, it should not be necessary to increase connection pool size to more than 700.

Note: If you increase `server.database-minpoolsize` to 100 or more, set `server.database-maxpoolsize` equal to that value.

6.5. Configure HQ Cache Settings for Improved Performance

The default cache settings in HQ are suitable for most deployments. However, depending on your deployment and use of HQ, especially in large environments, changing the cache settings might improve performance.

6.5.1. Where Are Caches Kept

All internal caches are kept in EHCache and are accessible via the "HQ Health" screen. EHCache provides:

- Automatic integration with Hibernate
- Optional distributed caching when running in a cluster.
- A variety of proven, well-tested eviction algorithms
- Better cache management and configuration

6.5.2. Default Cache Settings

By default, the cache settings in HQ are set for a medium to large deployment:

- 100 Platforms
- 500 Servers
- 5000 Services

These settings should suffice for most deployments.

6.5.3. Monitoring Caches to Identify Areas for Improvement

You can monitor HQ's internal caches through the server.log or via the "HQ Health" screen. The log contains diagnostic information, which is written to the log periodically. The cache information should look like:

```
Cache                                                Size   Hits   Misses
=====                                                ====   ====    =====
Agent.findByAgentToken                               2      4184       2
```

```
AgentScheduleInQueue                                    0      0          0
Alert.findByCreateTime                                  10     318        694
Alert.findByEntity                                      0      0          0
....
```

This listing shows each cache, its size, and its hit counts. At the end of the listing there is a summary for the total size of all HQ caches.

If a cache has an unusually high miss rate or becomes full, you can manually configure it to improve performance. For example, consider this cache:

```
Measurement.findByTemplateForInstance            10000   6766    25772
```

The cache has 10,000 elements. With only 6766 hits and 25,772 misses, the hit ratio is around 20-25%. Ideally the number of misses should peak at about the maximum size of the cache. (There are some exceptions to this: the UpdateTimestampsCache and the PermissionCache have items invalidated from the cache frequently, so these rules do not apply.) So, increasing this cache's size would probably improve HQ performance.

6.5.4. Changing Caches Settings

After you identify a cache whose configuration should be changed, you can change its configuration in the file `server-n.n.n-EE\hq-engine\hq-server\webapps\ROOT\WEB-INF\ehcache.xml`. In general, only the cache sizes should need to be changed.

To continue with the example cache from above, the entry for the cache in this file looks like this:

```
<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.

The format of this file is described in EHCache's documentation.

6.5.5. Configuring Permission Cache for Improved Performance *

The PermissionCache can be changed to improve performance. This cache caches recently evaluated permission checks:

```
<cache name="PermissionCache"
    maxElementsInMemory="1000"
    timeToIdleSeconds="0"
    timeToLiveSeconds="60"
    memoryStoreEvictionPolicy="LRU"/>
```

This cache is set to expire the checks after 60 seconds. In an environment where the user and role definitions are not updated frequently, you can increase the timeToLiveSeconds value to effect a significant performance increase.

While the right value for timeToLiveSeconds depends on the environment, the rule of thumb is that it should be set no higher than the acceptable time for a permission-related operation to take effect. Perhaps this is even zero, but that can be costly because the Server needs to go to the database to check permission on every resource the user views.

7. Integrate HQ Server with Other Systems

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

7.1. Configure LDAP Properties

These properties, on the **Administration > HQ Server Settings** page, configure HQ Server for your LDAP server.

Property	Description
URL	The LDAP URL the HQ Server will use to access the LDAP directory.
SSL	Indicates whether the LDAP directory requires SSL connections.
Username and Password	Used if the LDAP directory does not allow anonymous searching, as is common in secure environments. The username must be an LDAP user with sufficient privileges to view at least the sections of the directory containing the information for HQ users.
Search Base	Also known as the suffix. Required for an LDAP connection. If unsure of this setting, check with the LDAP administrator.
Search Filter	Limits the users in LDAP to a subset of the entire directory.
Login Property	The LDAP property that HQ will use as the username. Very important. Examples of common login properties are "cn" and "uid".

7.2. Configure Kerberos Properties

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

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

7.3. Configuring HQ Server for SMTP Server

HQ sends emails using the SMTP server specified during HQ 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 HQ 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 HQ'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 HQ Server restart.

   - To configure plain text communication, add the mail properties shown below to `hq-server.conf`. HQ's default behavior is equivalent to the values shown below - replace the values as appropriate for your environment.

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

   - To configure SSL communication, define the following properties:

```
server.mail.hostserver.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
mail.user=EAM Application
mail.password=password
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. 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 HQ Server.

   b. As the user that owns the HQ 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 HQ Server. When using a non-bundled JRE, use that JRE's keytool and cacerts file.

7.4. Enable Hyperic Enterprise to Send SNMP Traps

Available only in vFabric Hyperic

This page has information about enabling HQ Enterprise to send SNMP traps to an SNMP management system.

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

7.4.1. 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 HQ Server Enterprise 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 HQ Server for SNMP V1 communications with an NMS.

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

Configuration Option	Description	Allowable Values
Agent Address	Address of the managed object that generates the trap.	

Configure HQ Server Enterprise for SNMP v2c

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

Configure HQ Server Enterprise for SNMP v3

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

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

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

7.4.2. 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.

7.5. Configure HQ Server for Hyperic IQ

The **Hyperic IQ Server URL** property, on the **Administration** > **Server Settings** page identifies the URL of a Hyperic IQ installation. You must specify the IQ server to enable users to view IQ reports in the Hyperic IQ Reports dashboard portlet. The URL is:

```
http://IQ_host:9080/arc
```

where *IQ_host* is the address of the IQ server.

8. Clustering HQ Servers for Failover

Available only in vFabric Hyperic

8.1. Overview

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

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

A HQ Server cluster contains multiple nodes; two are generally sufficient. One HQ Server, automatically selected by HQ, 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 HQ administrators; it is not apparent that the active HQ server instance is clustered, or which node is currently active.

8.2. Requirements for an Failover Deployment

- A hardware-based load balancer.
- Only one HQ Server in an HQ Server cluster should receive agent communications at a time. The load balancer should should not direct agent connections to an HQ server instance that serves as the secondary node.
- Database Considerations---All nodes in the HQ cluster must share the same database. You cannot use HQ's internal PostgreSQL database in a failover configuration. You must use an external database; MySQL, Oracle, and PostgreSQL are supported.

8.3. Configuring a Server Cluster

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

8.3.1. Step 1: Install the First HQ Server Instance

Run the full installer in the appropriate mode for the type of database server you will use (-mysql, -postgresql, or -oracle). You **must** choose one of these options, because clustering requires the use of an external HQ database. The installer will create the HQ database schema.

8.3.2. Step 2: Install Additional HQ Server Nodes

For each additional node:

- Run the full installer in the appropriate mode for the type of database created during installation of the first server instance (-mysql, -postgresql, or -oracle).
- When the installer installer prompts for the location of the HQ 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".

8.3.3. Step 3: Configure Cluster Name and Communications Properties

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

Default hq-server.conf File

```
# Cluster Settings
################################################################################
#
# Property: ha.partition
# &amp;nbsp;
# 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 HQ Server in the cluster you must specify:

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

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

Optional Cluster Properties

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

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

8.3.4. 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 HQ Server nodes in the cluster and the failover behavior.

1. Identify the Hyperic Server nodes in the cluster.
2. Configure the load balancer to check the nodeStatus.hqu URL every 10 seconds. For example, in a 2-node cluster, if the the IP addresses of the nodes are 10.0.0.1 and 10.0.0.2, configure the load balancer to check these URLs every 10 seconds:
   - `http://hqadmin:hqadmin@10.0.0.1:7080/hqu/health/status/nodeStatus.hqu`
   - `http://hqadmin:hqadmin@10.0.0.2:7080/hqu/health/status/nodeStatus.hqu`
3. Configure the load balancer to direct all traffic to the node whose status is `master=true`.

8.3.5. Step 5: Configure Agents to Communicate with HQ Server Cluster

The HQ Agents in your environment communicate with the HQ 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.

8.3.6. Step 6: Start the Nodes

Start the HQ Servers.

8.3.7. 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 HQ 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 install an HQ Agent on the same machine as an HQ 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.

9. Server Properties

9.1. About HQ Server Configuration

9.1.1. Configuration Settings in hq-server.conf

`hq-server.conf` contains the configuration settings that HQ 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 HQ 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 HQ deployment. For example, there are properties in `hq-server.conf` that set defaults for database and JMS configuration options.

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

Note: HQ 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.

9.1.2. Configuration Settings in the Database

Some of the configuration data that governs HQ Server behavior is stored in the HQ Server database. For example, the data HQ Server needs to contact an HQ Agent is stored in in the HQ Database. For information about how HQ Server obtains HQ Agent address information, see Agent Server Communications Diagram.

9.2. Server Property Definitions

9.2.1. 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
```

9.2.2. server.database

Description

The kind of database the HQ server will use. The HQ server adjusts its interactions with the database according to the value of this property.

Valid values are:

- PostgreSQL
- Oracle8
- Oracle9i

Default

PostgreSQL

9.2.3. server.database-blockingtimeout

Description

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

Default

10000

9.2.4. server.database-driver

Description

The JDBC driver to use. You shouldn't change this unless you really know what you're doing.

Default

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

9.2.5. 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.

Default

100

9.2.6. server.database-minpoolsize

Description

The minimum number of database connections to keep in the pool

Default

5

9.2.7. server.database-password

Description

The database user's password.

Default

hqadmin

9.2.8. 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.

9.2.9. server.database-user

Description

The database user to connect as.

Default

hqadmin

9.2.10. server.encryption-key

Description

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

Default

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

9.2.11. server.hibernate.dialect

Description

The database-specific dialect class used by Hibernate in HQ

Default

`org.hyperic.hibernate.dialect.PostgreSQLDialect`

9.2.12. server.java.opts

Description

Additional options to pass to Java.

Default

```
-Djava.awt.headless=true -XX:MaxPermSize=192m -Xmx512m -Xms512m -XX:+Heap-
DumpOnOutOfMemoryError
```

For information about using different Java options when starting HQ Server, see Increasing Java Heap and Changing GC Settings.

For information about how to change Java heap settings on Windows, see Changing Heap Size on Windows.

9.2.13. server.jms.highmemory

Description

The high memory mark for the JMS queue.

Default

350

9.2.14. 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.

Default

400

9.2.15. server.mail.host

Description

The IP or hostname of the SMTP server that the HQ server will use for sending alerts and other HQ-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

9.2.16. server.quartzDelegate

Description

The database-specific plugin class used by HQ's internal scheduler service.

If you use Oracle as your HQ database, specify:

org.quartz.impl.jdbcjobstore.oracle.OracleDelegate

If you use the either HQ's internal PostgreSQL database or an external PostgreSQL database as your HQ database, specify:

org.quartz.impl.jdbcjobstore.PostgreSQLDelegate

Default

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

9.2.17. server.webapp.port

Description

The HTTP listen port. This is for the HQ web-based GUI and also HQ agents that communicate with the HQ server in non-secure mode.

Default

7080

9.2.18. server.webapp.secure.port

Description

The HTTPS listen port. This is for the HQ web-based GUI and also HQ agents that communicate with the HQ server in secure mode.

Default

7443

9.2.19. Clustering Properties in HQ Enterprise

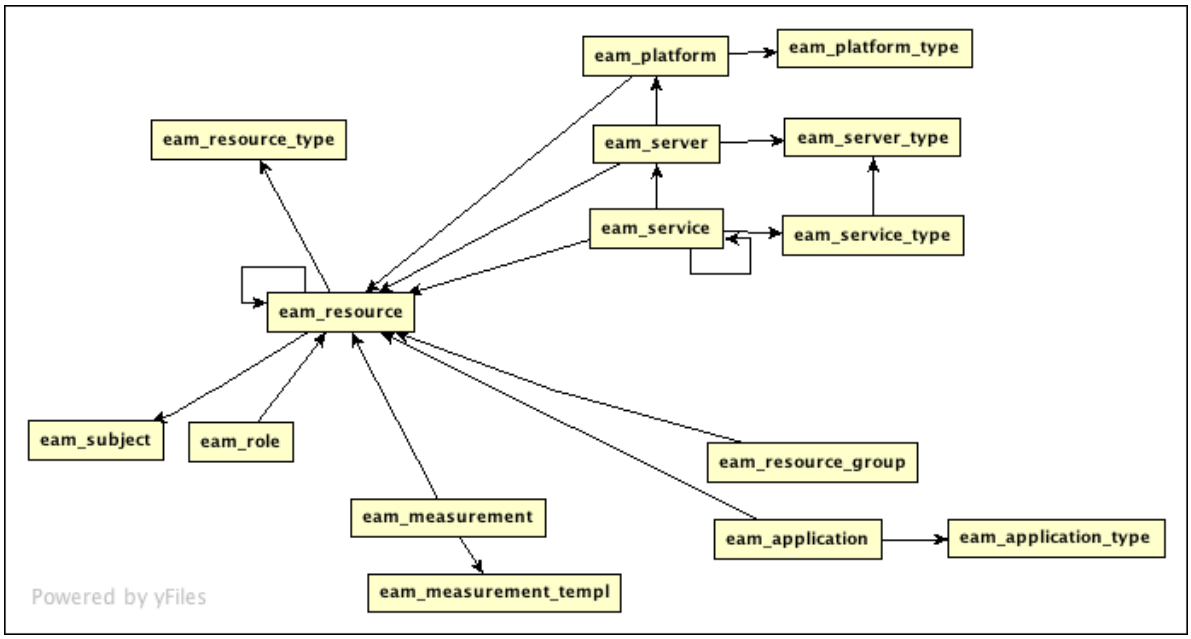
For information about properties for configuring an HQ Server cluster, see Section 8, “Clustering HQ Servers for Failover”.

10. HQ Database Table Schemas

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

10.1. Key Resource and Measurement Tables

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



10.1.1. EAM_RESOURCE Table: All Resource Types and Instances

The EAM_RESOURCE table contains information about the types in the HQ Inventory Model and instances of those types in the database. This table has a row for *every* managed resource in the HQ 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

10.1.2. 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.

10.1.3. 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 HQ can manage.
- EAM_SERVER_TYPE - Contains a row for every server type that HQ can manage.
- EAM_SERVICE_TYPE - Contains a row for every service type that HQ can manage.

10.1.4. Tables for Measurement Information

The following tables have information about the measurements that HQ can collect.
EAM_ MEASUREMENT_TEMPL - Contains a row for a every metric available for every inventory resource type with its metric template and default metric collection settings.

EAM_MEASUREMENT - Contains a row for every metric available for every resource in inventory, with metric collection configuration information: whether collection is enabled and the collection interval for enabled metrics.

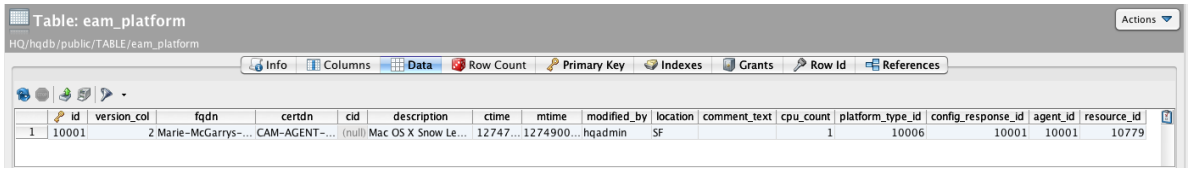
Note: These tables do not store metric values. Metric data is stored in the EAM_MEASUREMENT_DATA_1H, EAM_MEASUREMENTt_DATA_6H, and EAM_MEASUREMENT_DATA_1D tables.

10.2. Table Documentation

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

10.2.1. EAM_PLATFORM

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



Field	Type	Description
ID	int4	An ID for the platform, unique among platforms.
VERSION_COL	int8	Version of the row. Increments when the row is modified. increments with any change to configuration of this row
FQDN	varchar(200)	Fully qualified domain name of the platform.
CERTDN	varchar(200)	SSL Certificate for the agent which is monitoring this platform.
CID	int4	not used
DESCRIPTION	varchar(256)	Description of platform.
CTIME	int8	Creation time of platform.

MTIME	int8	Last modification time of the platform.
MODIFIED_BY	varchar(100)	Last modification user.
LOCATION	varchar(100)	String entered by user, optionally.
COMMENT_TEXT	varchar(256)	String entered by user, optionally.
CPU_COUNT	int4	Number of CPUs on this platform.
PLATFORM_TYPE_ID	int4	ID for the platform type. Points to EAM_PLATFORM_TYPE table.
CONFIG_RESPONSE_ID	int4	Link to configuration string in plugin xml file.
AGENT_ID	Int4	A unique identifier to the agent which is monitoring this platform.
RESOURCE_ID	int4	Uniquely identifies the resource, unique across platforms, servers, services. Points to the EAM_RESOURCE table.

10.2.2. EAM_PLATFORM_TYPE

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

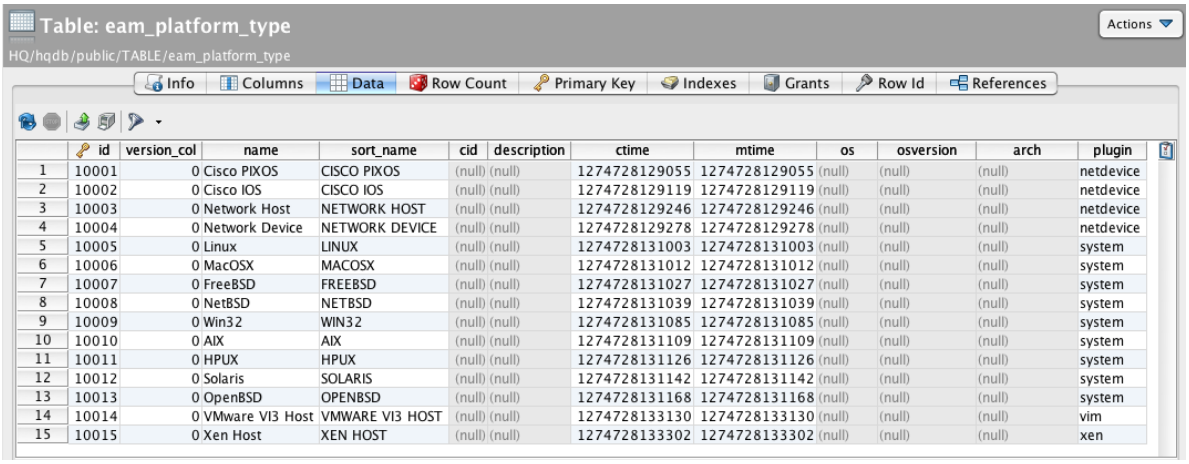


Table: eam_platform_type

Actions

HQ/hqdb/public/TABLE/eam_platform_type

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

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

10.2.3. EAM_SERVER

Contains a row for each server in HQ inventory.

Field	Type	Description
ID	int4	A unique identifier of the server.
VERSION_COL	int8	A column which increments with any change to configuration of this row.
CID	int4	
DESCRIPTION	varchar(300)	Description of server.

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

10.2.4. EAM_SERVICE

Contains a row for each service in HQ inventory

Field	Type	Description
ID	int4	An ID for the service, unique among services.
VERSION_COL	int8	A column which increments with any change to configuration of this row
CID	int4	
DESCRIPTION	varchar(200)	Description of service.
CTIME	int8	Creation time of service.
MTIME	int8	Last modification time of the service.
MODIFIED_BY	varchar(100)	Last modification user.
LOCATION	varchar(100)	Not used.
AUTODISCOVERY_ZOMBIE	bool	Were there deletions on the client side for this service?

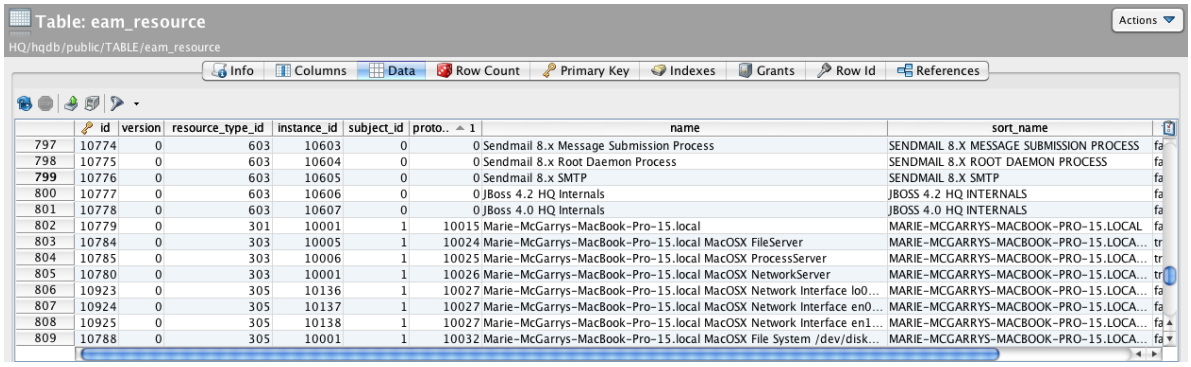
SERVICE_RT	bool	Is response time enabled for this service?
ENDUSER_RT	bool	Is end user response time enabled for this service?
PARENT_SERVICE_ID	int4	Unique ID into the parent service for this service.
SERVER_ID	int4	Were there deletions on the client side for this server?
AUTOINVENTORYIDENTIFIER	varchar(500)	A unique ID describing this server via the plugin XML.
SERVICE_TYPE_ID	Int4	Unique ID of service type for this service.
CONFIG_RESPONSE_ID	int4	Link to configuration string in plugin xml file.
RESOURCE_ID	int4	Uniquely identifies the resource, unique across platforms, servers, services. Points to the EAM_ RESOURCE table.

10.2.5. EAM_RESOURCE

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

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

Click the thumbnail to see example column data.



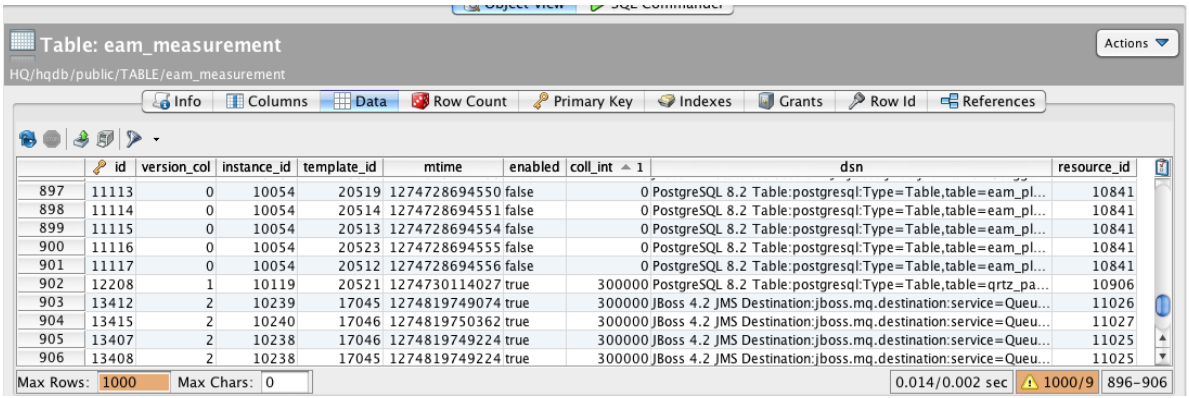
id	version	resource_type_id	instance_id	subject_id	proto.. 1	name	sort_name	
797	10774	0	603	10603	0	0 Sendmail 8.x Message Submission Process	SENDMAIL 8.X MESSAGE SUBMISSION PROCESS	fa
798	10775	0	603	10604	0	0 Sendmail 8.x Root Daemon Process	SENDMAIL 8.X ROOT DAEMON PROCESS	fa
799	10776	0	603	10605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

Field	Type	Description
ID	int4	Uniquely identifies a type or an instance of a type.
VERSION_COL	int8	Increments with any change to configuration of this row.

RESOURCE_TYPE_ID	int4	Identifies a type in the HQ inventory and access control model. The values this attribute identify a resource as one of the following:
INSTANCE_ID	int4	Uniquely identifies a type or an instance of a particular type in the inventory and access control model. For a type, corresponds to the ID column in one of the following tables: EAM_PLATFORM_TYPE, EAM_SERVER_TYPE&, EAM_SERVICE_TYPE, EAM_APPLICATION_TYPE, or EAM_RESOURCE_TYPE. For an instance of a type, corresponds to the ID column in one of the following tables: EAM_PLATFORM, EAM_SERVER, EAM_SERVICE, EAM_RESOURCE_GROUP, EAM_APPLICATION, EAM_ROLE, EAM_SUBJECT. EAM_ESCALATION
SUBJECT_ID	int4	Identifies the HQ user owns the resource.
PROTO_ID	int4	For a type, value is zero. For an instance of a type, contains the value of the ID column for the type in this table.
NAME	varchar(500)	Display name for a resource, for example, "My-Office-Mac-Book-Pro-15.local JBoss 4.2 default ServiceManager Stateless Session EJB".
SORT_NAME	varchar(500)	Same as the NAME column but all in upper case, for example, "MY-OFFICE-MAC-BOOK-PRO-15.LOCAL JBOSS 4.2 DEFAULT SERVICEMANAGER STATELESS SESSION EJB".
FSYSTEM	boolean	
MTIME	int8	Last modification time of the resource.

10.2.6. EAM_MEASUREMENT

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



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

Field	Type	Description
ID	int4	Unique ID for a metric that can be collected for a resource. Points to actually measurements in EAM_MEASUREMENT_DATA_* tables.
VERSION_COL	int8	Indicates version of the row, increments upon each change to the row.
INSTANCE_ID	int4	The resource type the measurement is for. Uniquely identifies a resource type of a given inventory level - platform, server, service. For example, the ID 10001 uniquely identifies the platform type "MacOSX".
TEMPLATE_ID	int4	ID of a template points to the EAM_MEASUREMENT_TEMPL table.
MTIME	int8	Time modified.
ENABLED	boolean	Is this metric enabled?
COLL_INTERVAL	int8	How often this metric is collected.
DSN	varchar(2048)	A string which describes the measurement from the plugin-xml text.
RESOURCE_ID	int4	Uniquely identifies the resource for which the metric is associated, unique across platforms, servers, services. Points to the EAM_ RESOURCE table.

10.2.7. EAM_MEASUREMENT_TEMPL

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

Field	Type	Description

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