

Red Hat Enterprise MRG 1.1

Messaging Installation Guide

Installation information for the Messaging
component of Red Hat Enterprise MRG



Lana Brindley

Red Hat Enterprise MRG 1.1 Messaging Installation Guide

Installation information for the Messaging component of Red Hat Enterprise MRG

Edition 2

Author

Lana Brindley

lbrindle@redhat.com

Copyright © 2008 Red Hat, Inc

Copyright © 2008 Red Hat, Inc. This material may only be distributed subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version of the OPL is presently available at <http://www.opencontent.org/openpub/>).

Red Hat and the Red Hat "Shadow Man" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

1801 Varsity Drive
Raleigh, NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588 Research Triangle Park, NC 27709 USA

This book will show you how to download and install the MRG Messaging component of the Red Hat Enterprise MRG distributed computing platform. To learn how to program MRG Messaging applications, see the MRG Messaging Tutorial.

Preface	v
1. Document Conventions	v
1.1. Typographic Conventions	v
1.2. Pull-quote Conventions	vii
1.3. Notes and Warnings	viii
2. We Need Feedback!	viii
1. Installing MRG Messaging	1
1.1. Installing MRG Messaging on Red Hat Enterprise Linux 5	1
1.2. Installing MRG Messaging on Red Hat Enterprise Linux 4	2
1.3. Available Packages — RPM	2
2. Starting the Broker	5
3. Options for Running the C++ Broker	7
3.1. Using Modules with the Broker	8
3.2. Logging Broker Errors	9
4. Persistence	11
5. Command line utilities	13
6. Clustering and federation	15
7. Authentication and Authorization	19
8. More Information	21
A. Revision History	23

Preface

Red Hat Enterprise MRG

This book contains basic installation information for the MRG Messaging component of Red Hat Enterprise MRG. Red Hat Enterprise MRG is a high performance distributed computing platform consisting of three components:

1. *Messaging* — Cross platform, high performance, reliable messaging using the Advanced Message Queuing Protocol (AMQP) standard.
2. *Realtime* — Consistent low-latency and predictable response times for applications that require microsecond latency.
3. *Grid* — Distributed High Throughput (HTC) and High Performance Computing (HPC).

All three components of Red Hat Enterprise MRG are designed to be used as part of the platform, but can also be used separately.

MRG Messaging

MRG Messaging is an open source, high performance, reliable messaging distribution that implements the Advanced Message Queuing Protocol (AMQP) standard. MRG Messaging is based on [Apache Qpid](#)¹, but includes persistence options, additional components, Linux kernel optimizations, and operating system services not found in the Qpid implementation. We have worked closely with companies that rely heavily on high performance messaging, and created a system to meet their real-world needs.

This guide shows you how to install MRG Messaging and start the broker, and explains the basic options available. If you want to write your own applications for use with MRG Messaging, you should also look at the *MRG Messaging Tutorial*.

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the [Liberation Fonts](#)² set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

Mono-spaced Bold

¹ <http://cwiki.apache.org/qpid/>

² <https://fedorahosted.org/liberation-fonts/>

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

To see the contents of the file **my_next_bestselling_novel** in your current working directory, enter the **cat my_next_bestselling_novel** command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

Press **Enter** to execute the command.

Press **Ctrl-Alt-F1** to switch to the first virtual terminal. Press **Ctrl-Alt-F7** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **Mono-spaced Bold**. For example:

File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System > Preferences > Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications > Accessories > Character Map** from the main menu bar. Next, choose **Search > Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit > Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the **>** shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **System** menu of the main menu bar' approach.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username@domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules (MPMs)*. Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in Mono-spaced Roman and presented thus:

```
books      Desktop  documentation  drafts  mss    photos  stuff  svn
books_tests Desktop1  downloads      images  notes  scripts svgs
```

Source-code listings are also set in Mono-spaced Roman but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
```

```
Object      ref      = iniCtx.lookup("EchoBean");
EchoHome    home     = (EchoHome) ref;
Echo        echo     = home.create();

System.out.println("Created Echo");

System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
}
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

A Note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.



Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: <http://bugzilla.redhat.com/bugzilla/> against the product **Red Hat Enterprise MRG**.

When submitting a bug report, be sure to mention the manual's identifier:

Messaging_Installation_Guide

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Installing MRG Messaging

In order to install MRG Messaging you will need to have registered your system with [Red Hat Network](#)¹. This table lists the Red Hat Enterprise MRG channels available on Red Hat Network for MRG Messaging.

Channel Name	Operating System	Architecture
Red Hat MRG Messaging	RHEL-4 AS	32bit, 64bit
Red Hat MRG Messaging	RHEL-4 ES	32bit, 64bit
Red Hat MRG Messaging	RHEL-5 Server	32bit, 64bit
Red Hat MRG Messaging	Non-Linux	32bit
Red Hat MRG Messaging Base	RHEL-4 AS	32bit, 64bit
Red Hat MRG Messaging Base	RHEL-4 ES	32bit, 64bit
Red Hat MRG Messaging Base	RHEL-5 Server	32bit, 64bit

Table 1.1. Red Hat Enterprise MRG Channels Available on Red Hat Network



Important

Before you install Red Hat Enterprise MRG check that your hardware and platform is supported. A complete list is available on the [Red Hat Enterprise MRG Supported Hardware Page](#)².

1.1. Installing MRG Messaging on Red Hat Enterprise Linux 5

1. Install the MRG Messaging group using the **yum** command.

```
# yum groupinstall "MRG Messaging"
```

2. You can check the installation location and that the components have been installed successfully by using the **rpm -ql** command with the name of the package you installed. For example:

```
# rpm -ql rhm
/usr/lib/qpiddlibbdbstore.so.0
/usr/lib/qpidd/libbdbstore.so.0.1.0
/usr/share/doc/rhm-0.2
/usr/share/doc/rhm-0.2/COPYING
/usr/share/doc/rhm-0.2/README
/var/rhm
```

¹ <https://rhn.redhat.com/help/about.pxt>



Note

If you find that yum is not installing all the dependencies you require, make sure that you have registered your system with [Red Hat Network](#)³.

1.2. Installing MRG Messaging on Red Hat Enterprise Linux 4

1. Install the MRG Messaging components using the **up2date** command.

```
# up2date python-qpid qpid-java-client qpidc-devel rhm rhm-docs
```

2. You can check the installation location and that the components have been installed successfully by using the **rpm -ql** command with the name of the package you installed. For example:

```
# rpm -ql rhm
/usr/lib/qpid/libbdbstore.so.0
/usr/lib/qpid/libbdbstore.so.0.1.0
/usr/share/doc/rhm-0.2
/usr/share/doc/rhm-0.2/COPYING
/usr/share/doc/rhm-0.2/README
/var/rhm
```



Note

If you find that **up2date** is not installing all the dependencies you require, make sure that you have registered your system with [Red Hat Network](#)⁴.

1.3. Available Packages — RPM

This section lists the RPM packages available for MRG Messaging..

RPM Package Name	Description	Language
qpid	MRG Messaging broker (Apache Qpid binaries for i386).	C++
rhm	MRG Messaging libraries, providing guaranteed message delivery.	C++
qpidc	MRG Messaging client libraries. Required to run the broker.	C++
qpidc-devel	C++ client libraries, including header files, developer	C++

RPM Package Name	Description	Language
	documentation, and symbolic links to shared libraries.	

Table 1.2. MRG Messaging Packages - x86 (AMD or Intel 32bit)

RPM Package Name	Description	Language
qpidd	MRG Messaging broker (Apache Qpid binaries for x86).	C++
rhm	MRG Messaging libraries, providing guaranteed message delivery.	C++
qpiddc	MRG Messaging client libraries. Required to run the broker.	C++
qpiddc-devel	C++ client libraries, including header files, developer documentation, and symbolic links to shared libraries.	C++

Table 1.3. MRG Messaging Packages - x86 (AMD64 or Intel 64)

RPM Package Name	Description	Language	Architecture
qpidd-java-client	Java client library including JMS implementation	Java	Architecture Independent
qpidd-java-common	Java client and broker (to be released) common library.	Java	Architecture Independent

Table 1.4. MRG Messaging Packages - Java

RPM Package Name	Description	Language	Architecture
amqp	AMQP specification (used internally by the Python client).	Python	Architecture Independent
python-qpidd	Python client libraries and command line utilities.	Python	Architecture Independent

Table 1.5. MRG Messaging Packages - Python

RPM Package Name	Description	Language	Architecture
rhm-docs	Installation Guide, Tutorial and source code examples.	US English	Architecture Independent

Table 1.6. MRG Messaging Packages - Documentation

Starting the Broker

Starting the C++ Broker

1. By default, the broker is installed in **/usr/sbin/**. If this is not on your path, you will need to type the whole path to start the broker:

```
$ /usr/sbin/qpidd -t
[date] [time] info Loaded Module: libbdbstore.so.0
[date] [time] info Locked data directory: /var/lib/qpidd
[date] [time] info Management enabled
[date] [time] info Listening on port 5672
```

The **-t** or **--trace** option enables debug tracing, printing messages to the terminal.



Important

When starting the broker, it will inform you that it has locked a data directory. This data directory is used for persistence, which is enabled by default in MRG Messaging. For more information about persistence see [Chapter 4, Persistence](#)

2. To stop the broker, type **CTRL+C** at the shell prompt

```
[date] [time] notice Shutting down.
[date] [time] info Unlocked data directory: /var/lib/qpidd
```

3. For production use, MRG Messaging is usually run as a service. To start the broker as a service, run the following command as the root user:

```
# service qpidd start
Starting qpidd daemon:      [ OK ]
```

4. You can check on the status of the service using the **service status** command and stop the broker with **service stop**.

```
# service qpidd status
qpidd (pid PID) is running...

# service qpidd stop
Stopping qpidd daemon:      [ OK ]
```

Running multiple brokers on a single machine

To run more than one broker on a single machine, they must run on different ports and use different directories for the journals.

1. Identify an available port by using the **-p** with a value of 0 (zero):

```
$ qpid -p 0  
40882
```

```
$ qpid -p 0  
45672
```

2. Start each new broker, using the **--data-dir** command to specify a new data directory for each:

```
$ qpid -p 40882 --data-dir /tmp/qpid/store/1
```

```
$ qpid -p 45672 --data-dir /tmp/qpid/store/2
```

Options for Running the C++ Broker

The broker can be run with a number of options. An overview of the most common options are given here.

Setting Options Using the Configuration File

1. For options that persist across sessions, you can put the options in the configuration file. Open the **/etc/qpidd.conf** file in your preferred text editor to make the necessary changes.
2. This example uses the configuration file to enable debug tracing. Changes will take effect from the next time the broker is started and will be used in every subsequent session.

```
# Configuration file for qpidd
trace=1
```

3. If you are running the broker as a service, you will need to restart the service once you have made the changes.

```
# service qpidd restart
Stopping qpidd daemon:          [ OK ]
Starting qpidd daemon:          [ OK ]
```

4. If you are *not* running the broker as a service, you can now start the broker as normal.

```
# /usr/sbin/qpidd -t
[date] [time] info Locked data directory: /var/lib/qpidd
[date] [time] info Management enabled
[date] [time] info Listening on port 5672
```

Setting Options Using the Command Line

To set options for a single instance, add the option to the command line when you start the broker. You will need to be the root user.

1. This example uses command line options to start the broker with debug tracing. These options will need to be explicitly stated every time the broker is run.

```
# /usr/sbin/qpidd -t
```

Common Options

For more options, type **man qpidd** or **/usr/sbin/qpidd --help** at the shell prompt.

General options for running the broker	
-t	This option enables debug tracing, with output printed to the screen.
-p <Port_Number>	Instructs the broker to use the specified port. Defaults to port 5672. It is possible to run multiple brokers simultaneously by using different port numbers.
-v	Displays the installed version.
-h	Displays the help message.

Table 3.1. General Broker Options

Options for running the broker as a service	
-d	This option instructs rhmd to run in the background as a daemon. Messages retrieved using a consumer are displayed, but any output usually displayed by the broker is suppressed.
-q	When the broker is running as a daemon this command shuts the broker down politely; that is, by closing the child processes, followed by the parent processes.
-c	This command checks if the daemon is already running. If it is running, it returns the process ID number.
-d --wait=<seconds>	This sets the maximum wait time (in seconds) for the daemon to initialize. If the daemon has not successfully completed initialization within this time, an error is returned. This option must be used in conjunction with the -d option, or it will be ignored.

Table 3.2. Options for running the broker as a service (daemon)

3.1. Using Modules with the Broker

MRG Messaging installs several modules by default, which are automatically loaded when the broker is started. The module directory for the client is located at **/usr/local/lib/qpidd/client** and for the daemon at **/usr/local/lib/qpidd/daemon**. The default modules are:

- SSL
- Authorization (ACL enforcement)
- RDMA (Infiniband)
- XML exchange type
- Persistence
- Clustering

All these modules are server side only, with the exception of the SSL and RDMA modules, which have both client and server side plugins. More information on working with these modules can be found in the *MRG Messaging User Guide*.

Options for using modules with the broker	
--load-module <i>MODULENAME</i>	Instructs the broker to use the specified module as a plug-in.
--module-dir <i><DIRECTORY></i>	Causes the broker to use a different module directory.
--no-module-dir	Causes the broker to ignore module directories.

Table 3.3. Options for using modules with the broker

Getting Help with Modules

To see the help text for any module, you will need to load the module in order to use the **--help** command:

```
# /usr/sbin/qpidd --load-module MODULENAME --help
```

3.2. Logging Broker Errors

Logging is enabled in the broker by default for all errors. Output is sent to **stderr** - the standard output error stream (usually this is the shell prompt). You can choose to log using **syslog**, which allows you to store logs to either a local file or to a remote logging server.

Options for logging with syslog	
--log-to-file <i>FILE</i>	Send log output to the specified filename. <i>FILE</i> can also be one of the special values: <ul style="list-style-type: none"> • stderr Standard output error stream • stdout Standard output • syslog Use syslog for log output The default is stderr .
--syslog-name <i>NAME</i>	Specify the name to use in syslog messages. The default is qpidd .
--syslog-facility <i>LOG_XXX</i>	Specify the facility to use in syslog messages. The default is LOG_DAEMON .

Table 3.4. Logging Options

Persistence

A persistence library allows MRG Messaging to store messages and queue configuration, ready to be reloaded in the event of machine or network failure. When the persistent store module is loaded, it allows messages and other persistent state information to be recovered when a broker is restarted.

In order for messages to be stored the persistence store must be loaded. The **--store-dir** command specifies the directory used for the persistence store and any configuration information. The default directory is **/var/lib/qpidd**. See [Table 4.1, “Persistence Options”](#) for options on how to change this behaviour.

In addition to loading the persistence store, queues and messages also need to be identified as *durable*. This can be done in the client application or by using the **qpidd-config** command line tool. See the *MRG Messaging Tutorial* for more information about creating client applications.



Important

If the persistence module is not loaded, messages and the broker state will not be stored to disk, even if the queue and messages sent to it are marked persistent.

Persistence Options	
--data-dir <i>DIRECTORY</i>	Specifies the directory for data storage and log files generated by the broker. The default is /var/lib/qpidd .
--no-data-dir <i>DIRECTORY</i>	Disables storage of configuration information and other data. If the default directory at /var/lib/qpidd exists, it will be ignored.
--num-jfiles <i>NUMBER</i>	Set the number of files for each instance of the persistence journal. The default is 8.
--jfile-size-pgs <i>NUMBER</i>	Set the size of each journal file in multiples of 64KB. The default is 24.
--wcache-page-size <i>NUMBER</i>	The size (in KB) of the pages in the write page cache. Allowable values must be powers of 2 (1, 2, 4, ... 128). Lower values will decrease latency but also decrease throughput. The default is 32.

Table 4.1. Persistence Options



Note

Persistence is dealt with in more depth in the *MRG Messaging User Guide*

Command line utilities

MRG Messaging contains a number of command line utilities for monitoring and configuring messaging brokers. A graphical interface is also available for download from the Red Hat Enterprise MRG yum repository, see the *MRG Management Console Installation Guide* for more information on this tool.

qpidd-config

Display and configure exchanges, queues, and bindings in the broker

qpidd-route

Display and configure broker federation, including routing and links between brokers

qpidd-tool

Access configuration, statistics, and control within the broker

qpidd-queue-stats

Monitor the size and enqueue/dequeue rates of queues in a broker

The command line utilities are included in the python client library package. Follow the installation instructions in [Chapter 1, Installing MRG Messaging](#) and install the **python-qpidd** and **amqp** packages.



Note

For more information on the command line utilities see the *MRG Messaging User Guide*

Clustering and federation

Messaging Clusters

A *Messaging Cluster* is a group of brokers that act as a single broker. Changes on any broker are replicated to all other brokers in the same Messaging Cluster, so if one broker fails, its clients can fail-over to another broker without loss of state. The brokers in a Messaging Cluster may run on the same host or on different hosts. Two brokers are in the same cluster if

1. They use the same OpenAIS **mcastaddr**, **mcastport**, and **bindnetaddr**, and
2. They use the same cluster name.

Messaging Clusters are implemented using using OpenAIS, which provides a reliable multicast protocol, tools, and infrastructure for implementing replicated services. You must install and configure OpenAIS to use MRG broker groups. Once you have installed OpenAIS, configure MRG Messaging to run in a cluster as follows.

1. Set the binding address for openais in `/etc/ais/openais.conf`. Use **ifconfig** to find the inet addr and the netmask for the interface you want:

```
# ifconfig
eth0  Link encap:Ethernet  HWaddr 00:E0:81:76:B6:C6
      inet addr:10.16.44.222  Bcast:10.16.47.255  Mask:255.255.248.0
      inet6 addr: fe80::2e0:81ff:fe76:b6c6/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:35914541 errors:6 dropped:0 overruns:0 frame:6
      TX packets:6529841 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:20294124383 (18.9 GiB)  TX bytes:12925473031 (12.0 GiB)
      Interrupt:98 Base address:0x8000
```

The binding address in `/etc/ais/openais.conf` should be the network address for the interface, which you can find by doing a bitwise **AND** of the inet addr (in this case, 10.16.44.222) and the network mask (in this case, 255.255.248.0). The result is 10.16.40.0. As a sanity check, you can use **route** and make sure the address you computed is associated with the interface:

```
$ /sbin/route
Kernel IP routing table
Destination  Gateway      Genmask      Flags Metric Ref  Use  Iface
20.0.10.0    *            255.255.255.0  U      0      0    0  eth1
192.168.122.0 *           255.255.255.0  U      0      0    0  virbr0
10.16.40.0    *            255.255.248.0  U      0      0    0  eth0
169.254.0.0  *            255.255.0.0   U      0      0    0  eth1
default      10.16.47.254 0.0.0.0       UG     0      0    0  eth0
```

To use **eth0** as the interface for the cluster, find the setting for **bindnetaddr** in `/etc/ais/openais.conf`, and set it to 10.16.40.0:

```
bindnetaddr: 10.16.40.0
```

2. Make sure that the primary group for the user running `qpidd` is `ais`. For instance, if you are running `qpidd` as a daemon, the user is named **qpidd**. You can make **ais** the primary group for **qpidd** as follows:

```
# usermod -g ais qpidd
```

3. Set the name of the cluster in `qpidd.conf`.

```
cluster-name="Mick"
```

4. Use **qpidd-tool** to see the cluster.

```
[root@mrg15 ~]# qpidd-tool localhost:5672
```

The cluster is one of the objects shown by the **list** command.

```
qpidd: list
Management Object Types:
ObjectType                                Active Deleted
=====
com.redhat.rhm.store:journal              1         0
com.redhat.rhm.store:store                 1         0
org.apache.qpid.broker:binding             5         0
org.apache.qpid.broker:broker              1         0
org.apache.qpid.broker:connection          1         0
org.apache.qpid.broker:exchange            7         0
org.apache.qpid.broker:queue               2         0
org.apache.qpid.broker:session             1         0
org.apache.qpid.broker:system              1         0
org.apache.qpid.broker:vhost               1         0
org.apache.qpid.cluster:cluster            1         0
```

To see the properties of the cluster, use **show cluster**:

```
qpidd: show cluster
Object of type org.apache.qpid.cluster:cluster: (last sample time:
13:56:40)
Type      Element      112
=====
property  brokerRef    102
property  clusterName  foo
property  clusterID    da821ff9-2a88-4002-b976-f18680556290
property  publishedURL
amqp:tcp:10.16.44.222:52265,tcp:20.0.10.15:52265,tcp:192.168.122.1:52265
```

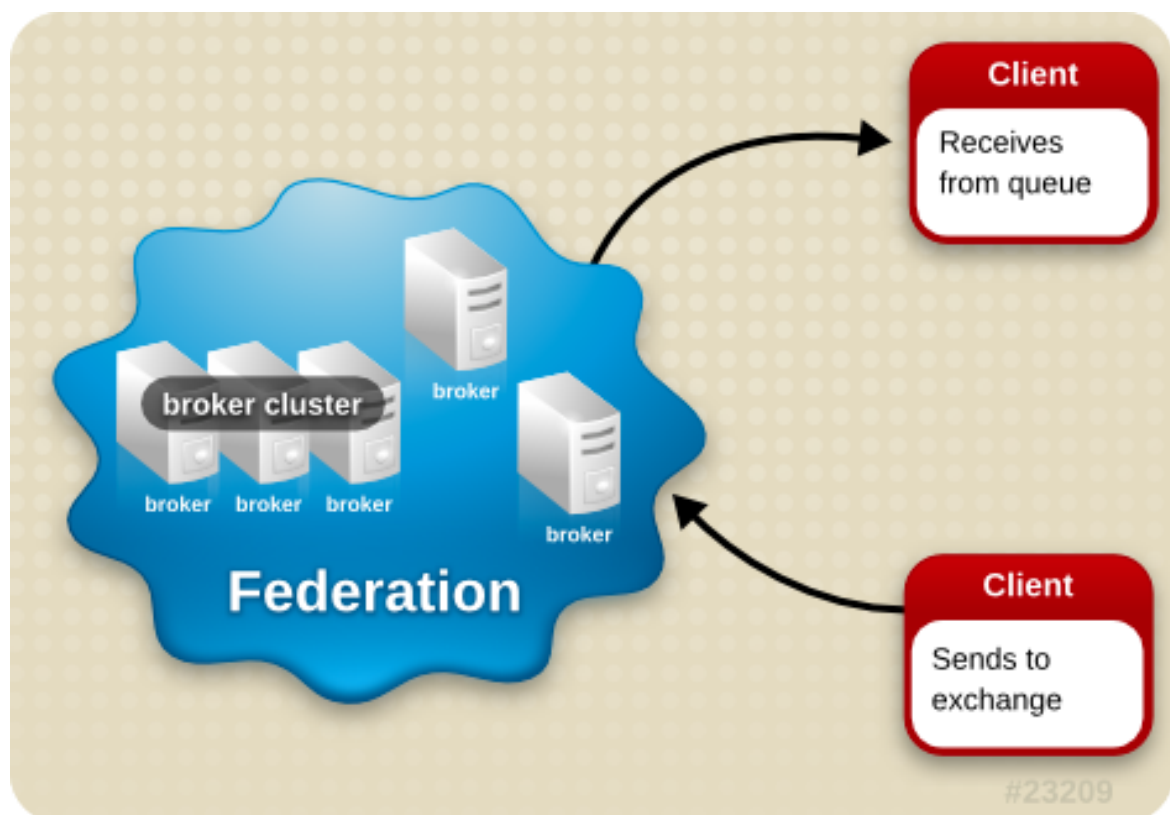


```
property clusterSize 1
property status ACTIVE
property members
  amqp:tcp:10.16.44.222:52265, tcp:20.0.10.15:52265, tcp:192.168.122.1:52265
```

Messaging Clusters can be used together with Red Hat Clustering Services (RHCS) by starting brokers with the **--cluster-cman** option.

Federation

Federation is used to provide geographical distribution of brokers. A number of individual brokers, or clusters of brokers, can be federated together. This allows client machines to see and interact with the federation as though it were a single broker. Federation can also be used where client machines need to remain on a local network, even though their messages have to be routed out.



Federation is used primarily for connecting disparate locations across a wide area network. Full connectivity across an enterprise can be achieved while keeping local message traffic isolated to a single location. Departmental brokers can be specified with individual policies that control inter-departmental message traffic flow.

Some applications can benefit from having a broker co-resident with the client. This is good for situations where the client produces data that must be delivered reliably but connectivity can not be guaranteed. In this case, a co-resident broker provides queueing and durability that is not available in the client on its own.

Federation bridges disjointed IP networks. Message brokers can be configured to allow message connectivity between networks where there is no IP connectivity. For example, an isolated, private IP network can have messaging connectivity to brokers in other outside IP networks.

Links and routes

Federation is configured through a series of links and routes.

A *link* is a connection between two brokers that allows messages to be passed between them. A link is a transport level connection (using a protocol such as TCP, RDMA, or SSL) that is initiated by a broker and accepted by another broker. The broker that initiates the link is considered the client in the connection. The broker that receives that connection will not treat it any differently from any other client connection, other than annotating it as being for federation.

Routes are the paths that messages take from one broker to another, and can run along one or more links to the final destination. A route is associated with an AMQP session established over the link connection. A route controls the flow of messages across the link between brokers, and multiple routes can share the same link. Messages will flow over a single route in only one direction. For bi-directional connectivity a pair of routes must be created, one for each direction of message flow. Routes always consist of a session and a subscription for consuming messages. Depending on the configuration, a route can have a private queue on the source broker with a binding to an exchange on that broker.



Note

Clustering and federation are dealt with in more depth in the *MRG Messaging User Guide*

Authentication and Authorization

MRG Messaging uses Simple Authentication and Security Layer (SASL) for identifying and authorizing incoming connections to the broker, as mandated in the AMQP specification. SASL provides a variety of authentication methods. While MRG Messaging clients primarily implement the **PLAIN** method, the broker uses the [Cyrus SASL library](http://cyrusimap.web.cmu.edu/)¹ to allow for a full SASL implementation.



Important

The **PLAIN** authentication method sends passwords in cleartext. For complete security, it is advised that Mozilla's Network Security Services Library (SSL) is also used. See the *MRG Messaging User Guide* for information on setting SSL on client machines.

Enabling and Using SASL Plain Authentication

To use the default SASL PLAIN authentication mechanism implemented by the MRG Messaging client libraries, either use the default username and password of *guest*, which are included in the database at `/var/lib/qpidd/qpidd.sasldb` on installation, or add your own accounts.

1. Add new users to the database by using the **saslpasswd2** command. The User ID for authentication and ACL authorization uses the form **user-id@domain..**

Ensure that the correct realm has been set for the broker. This can be done by editing the configuration file or using the **-u** option. The default realm for the broker is *QPID*.

```
# saslpasswd2 -f /var/lib/qpidd/qpidd.sasldb -u QPIDnew_user_name
```

2. Existing user accounts can be listed by using the **-f** option:

```
# sasldblistusers2 -f /var/lib/qpidd/qpidd.sasldb
```



Note

The user database at `/var/lib/qpidd/qpidd.sasldb` is readable only by the *qpidd* user. If you start the broker from a user other than the *qpidd* user, you will need to either modify the configuration file, or turn authentication off.

3. To switch authentication on or off, use the **auth yes|no** option when you start the broker:

```
# /usr/sbin/qpidd --auth yes
# /usr/sbin/qpidd --auth no
```

You can also set authentication to be on or off by adding the appropriate line to to the `/etc/qpidd.conf` configuration file:

¹ <http://cyrusimap.web.cmu.edu/>

```
auth=no  
  
auth=yes
```

4. The SASL configuration file is in `/etc/sasl2/qpidd.conf` for Red Hat Enterprise Linux 5 and `/usr/lib/sasl2/qpidd.conf` for Red Hat Enterprise Linux 4.

For information on using a different configuration, use your web browser to view the Cyrus SASL documentation at `/usr/share/doc/cyrus-sasl-lib-2.1.22/index.html` for Red Hat Enterprise Linux 5 or `/usr/share/doc/cyrus-sasl-2.1.19/index.html` for Red Hat Enterprise Linux 4.

Using ACL

1. The ACL module is loaded by default. You can check that it is loaded by running the `qpidd --help` command and checking the output for ACL options:

```
$ qpidd --help  
...[output truncated]...  
ACL Options:  
--acl-file FILE (policy.acl) The policy file to load from, loaded from  
data dir
```

2. To start using the ACL, you will need to specify the file to use. This is done by using the `--acl-file` command with a path and filename. The filename should have a `.acl` extension:

```
$ qpidd --acl-file ./aclfilename.acl
```

You can now view the file with the `cat` command and edit it in your preferred text editor. If the path and filename is not found, `qpidd` will fail to start.



Note

For more information on authentication and authorization see the *MRG Messaging User Guide*

More Information

Reporting a Bug

If you have found a bug in MRG Messaging, follow these instructions to enter a bug report:

1. You will need a [Bugzilla](#)¹ account. You can create one at [Create Bugzilla Account](#)².
2. Once you have a Bugzilla account, log in and click on [Enter A New Bug Report](#)³.
3. When submitting a bug report, you will need to identify the product (Red Hat Enterprise MRG), the version (1.1), and whether the bug occurs in the software (component = messaging) or in the documentation (component = Messaging_Installation_Guide).

Further Reading

- Red Hat Enterprise MRG and MRG Messaging Product Information
 - <http://www.redhat.com/mrg>
- Red Hat Enterprise MRG and MRG Messaging Documentation
 - http://redhat.com/docs/en-US/Red_Hat_Enterprise_MRG
 - <http://www.redhat.com/mrg/resources/>
- MRG Messaging Users Mailing List
 - Subscribe by sending an email to rhemrg-users-list@redhat.com with the word *Subscribe* in the subject line.

Appendix A. Revision History

Revision 1.8	Mon Jan 19 2009	Lana Brindley lbrindle@redhat.com
Added links to product page		
Revision 1.6	Tue Nov 25 2008	Lana Brindley lbrindle@redhat.com
Updates from QE review		
Revision 1.5	Thu Nov 20 2008	Lana Brindley lbrindle@redhat.com
Minor updates prior to releasing document to Quality Engineering		
Revision 1.4	Thu Nov 13 2008	Lana Brindley lbrindle@redhat.com
Updated Security Information - BZ #470378		
Minor updates to Concepts chapter arising from technical review		
Revision 1.3	Wed Oct 29 2008	Lana Brindley lbrindle@redhat.com
Fixed typos (authorisation/authorization)		
Revision 1.2	Wed Oct 29 2008	Lana Brindley lbrindle@redhat.com
Updated Authentication chapter		
Updated Starting the Broker chapter - Bugzilla #459118		
Revision 1.1	Thu Oct 23 2008	Lana Brindley lbrindle@redhat.com
Updated Persistence chapter		
Updated Federation chapter		
Revision 1.0	Thu Jun 5 2008	Lana Brindley lbrindle@redhat.com
Completed Revision for 1.0 Release		
Revision 0.4	Fri May 9 2008	Lana Brindley lbrindle@redhat.com
Updated Broker Options and Configuration file information		
Revision 0.3	Mon Feb 4 2008	Lana Brindley lbrindle@redhat.com
Updated Installation Instructions		
Revision 0.2	Tue Oct 16 2007	Jonathan Robie jonathan.robie@redhat.com
Major Edit		

Appendix A. Revision History

Revision 0.1 Fri Oct 12 2007
Preliminary Text, Untested

Lana Brindley lbrindle@redhat.com