Java Broker Design - Operational Logging

Operational Logging

The current logging configuration in the Java broker is focused for developers. Logging is performed on a class basis and as a result it is not easy to enable logging to get an operational view of the broker. Some work has been done to create configuration files that set the levels on various classes to provide the operational view of the broker (see Debug using Log4j). While this provides some good detail it does not provide the full picture.

The page will document the logging information that would be useful to provide, a suggested approach that should be followed and guidelines to developers for adding operational logging messages.

Overview

Currently logging is performed added in an ad-hoc manner by the devloper, ususlly to assist in the developemnt of the code base.

Log messages should be aimed at helping to provide users and/or support staff with information on the health of the broker; and - in the case where there has been some issue - help them diagnose the cause of that issue.

As such these messages should be readable without knowledge of the Qpid code base, they should not be so frequenet as to impact the performance of the broker but should be frequent enough such that diagnosis of issues is possible.

Log messages should occur whenever a significant event occurs, for instance the creation or destruction of a connection to the broker. The log message should contain enough information to be able to correlate the message with a business process event. In the case of a connection open the remote address, the login name, the virtual host, and the application id should be included in the log message.

Log messages at **INFO** level and above are expected to be turned on in a production environment.

Log Streams

There are number of data streams that this work should aim to address either directly as part of the standard opperation or through a dynamically configurable change.

- Major status changes reflecting broker state.
- Receive periodic statistical data on broker performance/processing.
- Ability to full track a single message through the broker.

Logging Content

There are two types of logging that are valuable to the operation of a Qpid broker:

1. State updates
2. Statistical updates

The main focus of the logging update is to improve the log file output however it should be remembered that the values should also be easily queried via the management console.

The existing logging within the broker should also be taken into consideration when performing this work. The analysis of this logging can be found here.

Status Updates

The following model objects should log updates on state changes, creation / descruction events, this will usually mean a single log instruction as the event occurs.

- Broker
- VirtualHost
- MessageStore
- Connection
- Channel
- Queue
- Exchange
- Binding

Statistics updates

In addition to status events being logged, we should periodically log statistics. Each model object may have a set of statistics that they wish to report but it is expected that the statistics will be based on the period between logs. There may also be desire to log at more than one interval. i.e once per minute, hour, day. The models objects may record state that can then be reported with the periodic statistics update.

Rate Statistics

The rate values can be reported at each of the levels as highlighed bellow. However, for Subscription only the outgoing rate makes sense.

Broker
VirtualHost
MessasgeStore
Connection
Channel
Queue
Exchange
Incoming message rate (count / bytes)
Outgoing message rate (count / bytes)
Min / Max / Average message size

Subscription
Outgoing message rate (count / bytes)
Min / Max / Average message size

Total Statistics

The total values can be reported for a number of objects as listed here, as with the rate values the totals for subscription only make sence for outgoing messages.

Broker
VirtualHost
MessageStore
Connection
Channel
Last Activity / Idle Notification
Total Incoming message (count / bytes)
Total Outgoing message (count / bytes)
Peak Incoming message (count / bytes)
Peak Outgoing message (count / bytes)

Broker
VirtualHost
MessageStore
Number of currrent connections
Total number of connections made

Subscription
Last Activity / Idle Notification
Total Outgoing messages (count / bytes)
Peak Outgoing message (count / bytes)

In addition there are a number of statistics that can be reported by the various model objects. These values can be included in any periodic report.

Broker
Heap current usage
Heap peak usage

MessasgeStore
Disk space used
Memory used
Entities Stored (Queue, Exchange, Binding, Message)

Connection
Number of active sessions
Number of current sessions
Number of consumers

Channel
Number of consumers

Queue
Current subcription count
Active subcription count
Binding Count
Current Queue size (count / bytes)
Messages sent (count / bytes)

Exchange
Min / Max / Average message size
Last Activity / Idle Notification
Bound queues
Total Messages sent (count / bytes)

Subscription
Min / Max / Average message size
Last Activity / Idle Notification
Total Outgoing messages (count / bytes)
Peak Outgoing message (count / bytes)
Unacknowledged size (count / bytes)

As will be mentioned in the guidelines section, any collection and reporting must be mindful of the performance overhead in doing so.

Logging Format

In order to keep the amount of data logged to the essential the message should be short and unambiguous - but easily recognisable. So the following formats are recommend for each model. The UID listed bellow will allow the disabmbigiuation between multiple entries. This value must be human readable so is expected to be an integer value.

Broker
b-
VirtualHost
vh(<name>)
Conection
con:<uid>(<username>, <ip>, <vhost-name>)
Channel
ch:<uid>
Subscription
sub:<uid>(<queue-name>)

Example:

```
[conn:1(guest, 127.0.0.1, /)]/[ch:2]/[sub:1(myqueue)]
```

Logging Hierarchy

When looking at augmenting the logging of the broker it makes sence to take a step back and provide an operational based logging hierarchy *qpid.* **in addition to the developer focused *org.apache.qpid.***.

The hierarchy of loggers should be structured such that it is easy to enable start or stop monitoring at runtime. The suggested hierarchy is as follows. **NOTE** The hierarchy is more of a graph and as a result it is possible to have more than one path to a logger. When implementing care must be taken so that each event is only logged once. i.e. Enabling *Exchange* and *Queue* logging should **not** result in the *Binding* operation being logged twice, once for the 'queue to exchange' and once for the 'exchange to queue'. In both cases the log statement will be the same however ony one instance should actually be logged.

```
qpid  Broker

                     [<username>]
       Connection  [<id>]  Channel  [<id>]  Subscription  [<id>]

       VirtualHost  <name>                             Binding

                                   Exchange  [<name>]

                                   Queue  [<name>]

                                   MessageStore
```

This would allow the quick enabling or disabling of the various logging events.

Guidelines for logging changes

To date there has been no discussion around what, who or even when to log and each developer has been left provide logging that they see fit. As mentioned earlier Log messages should occur whenever a significant event occurs, The log message should contain enough information to be able to correlate the message with a business process event. In the case of a connection open the remote address, the login name, the virtual host, and the application id should be included in the log message.

The log statements should be short and the reader should not need to refer to previous log statements in order to fully understand the situation. i.e. A new consumer log statement should include the connnection and virtualhost details rather than just the connection details and so requiring the user to read back to find the details around the connection creation.

All log statements that perform any computation before the log call must be wrapped with the *is<LEVEL>Enabled* calls to remove the computation should the log statement not be required.

When adding a new log statement a comment should be placed before hand highlighting if this is on the critical message routing path to allow reviewers to better gauge the potential impact. In addition the performance suit should be run with and without the log statement to get an actual measure of the impact.

Further Design Details

- Existing Logging Analysis
- Logging Format Design
- Status Update Design