Qpid Java Broker Transaction Timeouts

This page applies to Java Broker versions before 0.20. For the master copy of this documentation see the Producer Transaction Timeout section in the Java documentation

Transaction Timeouts User Guide

Introduction

Qpid 0.11 includes a new feature to allow deployers to configure transaction timeout thresholds, and the associated actions to perform on crossing of the specified threshold.

The transaction timeout mechanism is used to control broker resources when clients producing messages using transactional sessions hang or otherwise become unresponsive, or simply begin a transaction and keep using it without ever calling `Session#commit()`.

Users can choose to configure an *idleWarn* or *openWarn* threshold, after which the identified transaction should be logged as a **WARN** level alert as well as (more importantly) an *idleClose* or *openClose* threshold after which the transaction and the connection it applies to will be closed.

This feature is particularly useful in environments where the owner of the broker does not have full control over the implementation of clients, such as in a shared services deployment.

The following section provide more details on this feature and its use.

Purpose

This feature has been introduced to address the scenario where an open transaction on the broker holds an open transaction on the persistent store. This can have undesirable consequences if the store does not time out or close long-running transactions, such as with Oracle BDB. This can can result in a rapid increase in disk usage size, bounded only by available space, due to growth of the transaction log.

Scope

Note that only `MessageProducer` clients will be affected by a transaction timeout, since store transaction lifespan on a consumer only spans the execution of the call to `Session#commit()` and there is no scope for a long-lived transaction to arise.

It is also important to note that the transaction timeout mechanism is purely a *JMS* transaction timeout, and unrelated to any other timeouts in the Qpid client library and will have no impact on any RDBMS your application may utilise.

Effect

Full details of configuration options are provided in the sections that follow. This section gives a brief overview of what the Transaction Timeout feature can do.

Broker Logging and Connection Close

When the *openWarn* or *idleWarn* specified threshold is exceeded, the broker will log a **WARN** level alert with details of the connection and channel on which the threshold has been exceeded, along with the age of the transaction.

When the *openClose* or *idleClose* specified threshold value is exceeded, the broker will throw an exception back to the client connection via the `ExceptionListener`, log the action and then close the connection.

The example broker log output shown below is where the *idleWarn* threshold specified is lower than the *idleClose* threshold and the broker therefore logs the idle transaction 3 times before the close threshold is triggered and the connection closed out.

```
INFO [qpid.message] MESSAGE [Queue-housekeeping-test][con:1(guest@anonymous(29494388)/test)/ch:2] CHN-1008 :
Idle Transaction : 13,116 ms
WARN [apache.qpid.server.AMQChannel] IDLE TRANSACTION ALERT [con:1(guest@anonymous(29494388)/test)/ch:2]  13116
ms
INFO [qpid.message] MESSAGE [Queue-housekeeping-test][con:1(guest@anonymous(29494388)/test)/ch:2] CHN-1008 :
Idle Transaction : 14,116 ms
WARN [apache.qpid.server.AMQChannel] IDLE TRANSACTION ALERT [con:1(guest@anonymous(29494388)/test)/ch:2]  14116
ms
INFO [qpid.message] MESSAGE [Queue-housekeeping-test][con:1(guest@anonymous(29494388)/test)/ch:2] CHN-1008 :
Idle Transaction : 15,118 ms
WARN [apache.qpid.server.AMQChannel] IDLE TRANSACTION ALERT [con:1(guest@anonymous(29494388)/test)/ch:2]  15118
ms
INFO [qpid.server.protocol.AMQProtocolSession] Closing connection due to: org.apache.qpid.
AMQConnectionException:
 Idle transaction timed out [error code 506: resource error]
INFO [qpid.message] MESSAGE [Queue-housekeeping-test][con:1(guest@anonymous(29494388)/test)/ch:2] CHN-1003 :
Close
```

The second example broker log output shown below illustrates the same mechanism operating on an open transaction.

```
INFO [qpid.message] MESSAGE [Queue-housekeeping-test][con:2(guest@anonymous(27925944)/test)/ch:2] CHN-1007 :
Open Transaction : 12,406 ms
WARN [apache.qpid.server.AMQChannel] OPEN TRANSACTION ALERT [con:2(guest@anonymous(27925944)/test)/ch:2]  12406
ms
INFO [qpid.message] MESSAGE [Queue-housekeeping-test][con:2(guest@anonymous(27925944)/test)/ch:2] CHN-1007 :
Open Transaction : 13,406 ms
WARN [apache.qpid.server.AMQChannel] OPEN TRANSACTION ALERT [con:2(guest@anonymous(27925944)/test)/ch:2]  13406
ms
INFO [qpid.message] MESSAGE [Queue-housekeeping-test][con:2(guest@anonymous(27925944)/test)/ch:2] CHN-1007 :
Open Transaction : 14,406 ms
WARN [apache.qpid.server.AMQChannel] OPEN TRANSACTION ALERT [con:2(guest@anonymous(27925944)/test)/ch:2]  14406
ms
INFO [qpid.server.protocol.AMQProtocolSession] Closing connection due to: org.apache.qpid.
AMQConnectionException:
 Open transaction timed out [error code 506: resource error]
INFO [qpid.message] MESSAGE [Queue-housekeeping-test][con:2(guest@anonymous(27925944)/test)/ch:2] CHN-1003 :
Close
```

Client Side Effect

After a Close threshold has been exceeded, the trigger client will receive this exception on its exception listener, prior to being disconnected:

```
org.apache.qpid.AMQConnectionClosedException: Error: Idle transaction timed out [error code 506: resource
error] [error code 506: resource error]
```

Any later attempt to use the connection will result in this exception being thrown:

```
Producer: Caught an Exception: javax.jms.IllegalStateException: Object org.apache.qpid.client.
AMQSession_0_8@129b0e1 has been closed
javax.jms.IllegalStateException: Object org.apache.qpid.client.AMQSession_0_8@129b0e1 has been closed
        at org.apache.qpid.client.Closeable.checkNotClosed(Closeable.java:70)
        at org.apache.qpid.client.AMQSession.checkNotClosed(AMQSession.java:555)
        at org.apache.qpid.client.AMQSession.createBytesMessage(AMQSession.java:573)
        at org.apache.qpid.test.unit.transacted.TransactedProducer.runTest(TransactedProducer.java:138)
        at org.apache.qpid.test.unit.transacted.TransactedProducer.main(TransactedProducer.java:86)
```

Thus **clients must be able to handle this case successfully, reconnecting where required and registering an exception listener on all connections** . This is critical, and must be communicated to client applications by any broker owner switching on transaction timeouts.

Configuration

Overview

Transaction timeouts are configurable separately on each defined virtual host, using the `virtualhosts.xml` file.

We would recommend that only warnings are configured at first, which should allow broker administrators to obtain an idea of the distribution of transaction lengths on their systems, and configure production settings appropriately for both warning and closure. Ideally establishing thresholds should be achieved in a representative UAT environment, with clients and broker running, prior to any production deployment.

It is impossible to give suggested values, due to the large variation in usage depending on the applications using a broker. However, clearly transactions should not span the expected lifetime of any client application as this would indicate a hung client.

When configuring warning and closure timeouts, it should be noted that these only apply to message producers that are connected to the broker, but that a timeout will cause the connection to be closed - this disconnecting all producers and consumers created on that connection.

This should not be an issue for environments using Mule or Spring, where connection factories can be configured appropriately to manage a single `MessageProducer` object per JMS Session and Connection. Clients that use the JMS API directly should be aware that sessions managing both consumers and producers, or multiple producers, will be affected by a single producer hanging or leaving a transaction idle or open, and closed, and must take appropriate action to handle that scenario.

Virtualhosts.xml Entries

The JMS transaction timeouts are configured on each virtual host defined in the XML configuration files.

The default values for each of the parameters is *0*, indicating that that particular check is disabled.

Any or all of the parameters can be set, using the desired value in milliseconds, and will be checked each time the housekeeping process runs, usually set to run every 30 seconds in standard configuration. The meaning of each property is as follows:

- **openWarn** - the time a transaction can be open for (with activity occurring on it) after which a warning alert will be issued.
- **openClose** - the time a transaction can be open for before the connection it is on is closed.
- **idleWarn** - the time a transaction can be idle for (with no activity occurring on it) after which a warning alert will be issued.
- **idleClose** - the time a transaction can be idle for before the connection it is on is closed.

The virtualhosts configuration is shown below, and must occur inside the `//virtualhosts/virtualhost/name/` elements:

virtualhosts.xml

```
<transactionTimeout>
    <openWarn>10000</openWarn>
    <openClose>20000</openClose>
    <idleWarn>5000</idleWarn>
    <idleClose>15000</idleClose>
</transactionTimeout>
```