# Qpid Design - Message Acknowledgement

## Message Acknowledgements and Delivery Modes

When implementing the JMS client it became apparent that the JMS specification offered a considerable degree of latitude for interpreting the precise semantics of acknowledgement modes and it also did not cover all acknowledgement modes that are of interest.

Here we describe the precise semantics of the JMS acknowledgement modes and the additional modes that the JMS client provides.

In this discussion, "the client" refers to the JMS client implementation and "the user" refers to code that is part of the client application (i.e. code written by the end-user developer).

### AUTO_ACKNOWLEDGE (JMS)

In this mode, the client acknowledges each message once it has been received by the user. In the case of an asynchronous message consumer, this means that an acknowledgement is sent once the onMessage method of a message listener has completed without throwing an exception of any sort. For a synchronous consumer, it means when the receive() method has returned the message to the user.

A single BasicAckBody is sent with the delivery tag of the message and the multiple flag set to false, acknowledging that message only.

### CLIENT_ACKNOWLEDGE (JMS)

In this mode, the user acknowledges messages manually by calling the acknowledge() method on either the session or the message itself. These both have the same effect.

The JMS does not say how many message a client is allowed to receive before acknowledging. However, it does talk in vague terms about implementations making sure clients don't go too long without acknowledging to avoid resource exhaustion. Qpid uses the prefetch value for this - the consumer must ack it's messages before it reaches this limit if it wants to recieve any more.

Calling either Session.acknowledge or Message.acknowledge() acknowledges the receipt of all messages up to and including the current one.

### DUPS_OK_ACKNOWLEDGE (JMS)

This mode is identical to AUTO_ACKNOWLEDGE from an implementation perspective, however the user application must be prepared to deal with duplicate messages.

### PRE_ACKNOWLEDGE (non-JMS)

A mode not covered by the JMS specification is one where the client acknowledges a message before calling the onMessage() or receive() methods. It sends a BasicAcknowledge for each message as the message is passed to onMessage or receive() retrives it. The semantics are therefore exactly the same as AUTO_ACKNOWLEDGE for receive(), but differ for onMessage() in that the message is acknowledged regardless of whether the method completes successfully or not.

The constant org.apache.qpid.jms.Session.PRE_ACKNOWLEDGE defines this mode.

### NO_ACKNOWLEDGE (non-JMS)

Certain data may be time sensitive in the sense that redelivery is pointless - if the client cannot process it at the instant it is sent there is no point in redelivering it.

In this case, acks are redundant. Since TCP means that the server can be sure the client received the message the only problem could be client error.

Setting NO_ACKNOWLEDGE means that the client never sends a BasicAcknowledge and the broker removes the message from the queue as soon as it is sent.

The constant org.apache.qpid.jms.Session.NO_ACKNOWLEDGE defines this mode.

## Delivery Modes

For message production, similar considerations apply. JMS defines two delivery modes, PERSISTENT and NON_PERSISTENT which allow the implementor considerable freedom of implementation.

Unfortunately the JMS specification addresses what are really two separate reliability concerns with a single delivery mode.

The default delivery mode can be set on a producer. This can be overridden on each message sent.

### PERSISTENT (JMS)

Persistent is the straightforward option. Messages are sent with the persistent flag set to true which means that they will be committed to stable storage.

### NON_PERSISTENT (JMS)

Non persistent gives maximum performance with least guarantees. The persistent flag is set to false in each message which means that if the broker suffers an error it is neither required nor expected to recover those messages.