Advanced Message Queuing Protocol (AMQP)
JMS Mapping Version 1.0

Working Draft 3
11 February 2014

Specification URIs

This version:
  http://docs.oasis-open.org/amq-bindmap/jms/v1.0/wd03/amqp-bindmap-jms-v1.0-wd03.xml (Authoritative)
  http://docs.oasis-open.org/amq-bindmap/jms/v1.0/wd03/amqp-bindmap-jms-v1.0-wd03.html
  http://docs.oasis-open.org/amq-bindmap/jms/v1.0/wd03/amqp-bindmap-jms-v1.0-wd03.pdf

Previous version:
  N/A

Latest version:
  http://docs.oasis-open.org/amq-bindmap/jms/v1.0/amqp-bindmap-jms-v1.0.html
  http://docs.oasis-open.org/amq-bindmap/jms/v1.0/amqp-bindmap-jms-v1.0.pdf
  http://docs.oasis-open.org/amq-bindmap/jms/v1.0/amqp-bindmap-jms-v1.0.xml (Authoritative)

Technical Committee:
  OASIS Advanced Message Queuing Protocol (AMQP) Bindings and Mappings (AMQP-BINDMAP) TC

Chairs:
  Steve Huston (shuston@riverace.com), Individual

Editors:
  Robert Gemmell (robert.gemmell@jpmchase.com), JPMorgan Chase & Co.
  Robert Godfrey (robert.godfrey@jpmorgan.com), JPMorgan Chase & Co.
Abstract:
TODO

Status:
This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee's web page at http://www.oasis-open.org/committees/amqp-bindmap/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/amqp-bindmap/ipr.php).

Citation format:
When referencing this specification the following citation format should be used:

[amqp-bindmap-jms-v1.0]
http://docs.oasis-open.org/amqp-bindmap/jms/v1.0/wd03/amqp-bindmap-jms-v1.0-wd03.pdf.
Contents

1 References 4

2 Mapping JMS Types to AMQP 5

3 Mapping AMQP Types to Java 6

4 Messages 7
  4.1 Message Structure ................................................................. 7
  4.2 Mapping JMS Messages To AMQP ............................................... 7
    4.2.1 JMS Headers .................................................................. 8
    4.2.2 JMS-defined 'JMSX' Properties ......................................... 13
    4.2.3 JMS Properties ................................................................ 14
    4.2.4 Message Body Types .......................................................... 14
  4.3 Mapping AMQP Messages To Java .............................................. 18
    4.3.1 Header Section ................................................................ 18
    4.3.2 Properties Section .............................................................. 19
    4.3.3 Application Properties Section ............................................ 21
    4.3.4 Delivery Annotations Section .............................................. 21
    4.3.5 Message Annotations Section .............................................. 22
    4.3.6 Footer Section .................................................................. 22
    4.3.7 Body Sections ................................................................... 22

5 JMS Vendor Properties 24

6 Destinations 25

7 Message Producers 26
  7.1 Sending Messages ................................................................. 26

8 Delivery Count Handling 27

9 Supplementary Definitions 28
1 References

TODO (presentation): move refs to wherever they are meant to go, ensure they are structured correctly, etc.

[AMQPMGSFORMAT]
AMQP Message Format http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html#section-message-format

[ISO88591]
2 Mapping JMS Types to AMQP

TODO (content): define mapping from JMS types to AMQP types.
3 Mapping AMQP Types to Java

TODO (content): define mapping from AMQP types to Java types.
4 Messages

4.1 Message Structure

Both JMS and AMQP define Message structure in terms of “Header”, “Properties” and the message “Body”. Unfortunately the definitions of these terms are not consistent. For JMS the Headers refer to a defined set of attributes which are a mix of “immutable” and “mutable” (i.e. some which are invariant over the lifetime of the message, and some which are updated as the message travels from sender to eventual receiver). In contrast JMS Properties are (mostly) application defined message attributes set by the sender and invariant over the message lifetime from sender to receiver. A number of JMS-defined ‘JMSX’ Properties also exist which live in the same namespace as the application properties.

The AMQP Message is defined as a sequence of “Sections” [AMQPMSGFORMAT].

![AMQP Message Structure Diagram]

Figure 4.1: AMQP Message Structure

The AMQP header section defines a set of attributes which apply to the message (or rather this particular transfer of the message). These attributes are “mutable” throughout the passage of the message through the AMQP network. The properties section defines “immutable” properties of the message.

4.2 Mapping JMS Messages To AMQP

In overview we can say that a JMS Message has the following logical layout:

```
<table>
<thead>
<tr>
<th>JMS</th>
<th>JMS</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>Headers</td>
<td>Properties</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 4.2: JMS Message

In overview we can say that a JMS Message maps to an AMQP message as follows: The JMS Headers and some JMS-defined ‘JMSX’ Properties will be stored within the header and properties sections, with occasional aid of additional message-annotations. JMS Properties set by applications will be stored in the application-properties section, including some JMS-defined ‘JMSX’ Properties. If no such properties are set, the application-properties section MAY be omitted. The Message body will be stored in application-data section(s) with type dependent on the particular JMS Message type in use.

TODO (content): do we enable setting (and thus describe here) delivery-annotations or footer details?
4.2.1 JMS Headers

The following section describes how each of the defined JMS Headers can be mapped to an AMQP Message.

<table>
<thead>
<tr>
<th>Header Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMSMessageID</td>
<td>The JMSMessageID is defined as a Java String identifier for the Message which is set by the implementation during publication. AMQP uses the message-id field of properties for the same purpose, which is defined as being of type providing message-id, that is message-id-ulong, message-id-uuid, message-id-binary or message-id-string. Sending JMS clients SHOULD use the message-id-string type for the message-id field of properties by default. JMSMessageID String values are REQUIRED to have a prefix of “ID:”. A single instance of this prefix MUST be removed from the value prior to setting the message-id field of properties on producing JMS clients, and MUST be added by receiving JMS clients (i.e. this prefix MUST be synthesized by the client library). See 4.2.1.1 JMSMessageID And JMSCorrelationID Handling for REQUIRED detail relating to supporting usage of the various AMQP types possible for the message-id field of properties.</td>
</tr>
<tr>
<td>JMSTimestamp</td>
<td>The JMSTimestamp header is defined as a Java long representing the time at which the message was handed off to the provider to send, in milliseconds since the Unix Epoch. That is, the value is set at the originating client and not changed thereafter. AMQP uses the creation-time field of properties for the same purpose.</td>
</tr>
</tbody>
</table>
| **JMSCorrelationID** | The JMSCorrelationID header is defined as a Java String or byte[] used to link one message with another.

AMQP uses the correlation-id field of properties for the same purpose, which is defined as being of type providing message-id, that is message-id-ulong, message-id-uuid, message-id-binary or message-id-string.

To signal to receiving JMS clients in an interoperable way that the correlation-id field of properties represents an application-specific String and not a JMSMessageID String, a boolean message annotation with symbol key of "x-opt-app-correlation-id" is used. When setting JMSCorrelationID to a JMSMessageID (i.e. a String with prefix "ID:"), the annotation MUST be omitted or set to false. When setting JMSCorrelationID to an application-specific String value, the annotation MUST be set to true.

JMSMessageID String values are required to have a prefix of "ID:". When using these as the value for JMSCorrelationID, a single instance of this prefix MUST be removed before using the remainder to set the correlation-id field of properties in the producing JMS client and MUST be added by receiving JMS clients (i.e. this prefix MUST be synthesized by the client library).

See 4.2.1.1 JMSMessageID And JMSCorrelationID Handling for REQUIRED detail relating to supporting usage of the various AMQP types possible for the correlation-id field of properties. Application-specific JMSCorrelationID values MUST be sent using the message-id-string type. |
| **JMSReplyTo** | The JMSReplyTo header is equivalent to the reply-to field of properties.

JMSReplyTo is defined as being of the JMS Destination type, while the reply-to field of properties requires an address-string. See 6. Destinations for REQUIRED detail as to how conversion between these types is achieved. |
<table>
<thead>
<tr>
<th>Header Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMSDestination</td>
<td>The JMSDestination header is equivalent to the to field of properties. Note that producers MUST set the to field of properties explicitly (intermediaries cannot derive it from address of the target of the link on which the message was sent). JMSDestination is defined as being of the JMS Destination type, while the to field of properties requires an address-string. See 6. Destinations for REQUIRED detail as to how conversion between these types is be achieved.</td>
</tr>
<tr>
<td>JMSDeliveryMode</td>
<td>The JMSDeliveryMode header is defined as a Java int with two possible values: NON_PERSISTENT and PERSISTENT. The JMSDeliveryMode header relates to two different aspects of sending a JMS Message as an AMQP message. Firstly, its value is equivalent to the durable field of header. For PERSISTENT messages, the durable field of header MUST be set to true. For NON_PERSISTENT messages, the durable field of header MUST be either set to false or omitted. Additionally, the JMSDeliveryMode value relates to the reliability guarantees of the AMQP message transfer, specifically the point at which sent messages are considered settled. For PERSISTENT messages the sender MUST NOT consider the message settled until the point that the sender has received notification of the disposition at the receiver. For NON_PERSISTENT messages on a non-transacted session an implementer MAY choose to send messages considering them settled as soon as they are sent (i.e. with the settled flag set to true on their original transfer).</td>
</tr>
<tr>
<td>JMSRedelivered</td>
<td>This header is set by the client provider on receipt of the message, based on handling of the delivery-count field of header. See 8. Delivery Count Handling for more details on handling of the delivery-count value.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JMSType</td>
<td>The JMSType field has no equivalent in AMQP. It is a Java String identifier defined with respect to a notional message definition repository in which message type definitions are contained. This definition would perhaps map closest to the descriptor used on a message whose body consisted of a single instance of an AMQP described type, however as such AMQP types carry their own descriptor it does not need to appear in the message headers. In order to carry the JMSType value on a message in an interoperable way, a message annotation with symbol key of “x-opt-jms-type” MUST be used, containing a string representing the JMSType value. If JMSType is not set, the annotation MUST be omitted.</td>
</tr>
<tr>
<td>JMSExpiration</td>
<td>The JMSExpiration header is defined as a Java long representing the time at which the message expires, in milliseconds since the Unix Epoch. A value for JMSExpiration is set by the provider when sending the message. That is, the value is set at the originating client and not changed thereafter. If a non-zero time-to-live value is specified when sending the message, JMSExpiration contains the computed expiry time. If no time-to-live value (or a value of zero) is supplied when sending the message, then JMSExpiration has the value zero. AMQP uses the absolute-expiry-time field of properties for the purpose of setting an expiration time. When a non-zero value time-to-live is supplied, the computed expiration time MUST be set in the absolute-expiry-time field of properties. When no time-to-live value (or a value of zero) is supplied and JMSExpiration thus has the value zero, the absolute-expiry-time field of properties MUST be omitted rather than set to zero. See 7.1 Sending Messages for additional REQUIRED detail relating to message expiration.</td>
</tr>
<tr>
<td>JMSPriority</td>
<td>The JMSPriority is equivalent to the priority field of header. JMSPriority is specified as being a Java int despite the valid values only being 0-9. AMQP allows the priority to be any valid ubyte value. When messages are being sent with a priority of DEFAULT_PRIORITY, the priority field of header SHOULD be omitted.</td>
</tr>
</tbody>
</table>
The `JMSDeliveryTime` header has no equivalent in AMQP. It is defined as a Java `long` representing the earliest time at which the message it to be made available for delivery to a consumer, in milliseconds since the Unix Epoch. The value is set at the producing client by adding any provided delivery delay value to the time at which the message is sent.

In order to carry the `JMSDeliveryTime` value on a message in an interoperable way, a message annotation with symbol key of "x-opt-delivery-time" and type `timestamp` MUST be used if a non-zero delivery delay is specified. If no delivery-delay is specified then the annotation SHOULD be omitted, and receiving JMS clients MUST then synthesize the value via use of the `JMSTimestamp` header instead.

TODO (): define filter to enforce this on the broker?

TODO (): define connection-capability for feature?

### 4.2.1.1 JMSMessageID And JMSCorrelationID Handling

As indicated in 4.2.1 JMS Headers and 4.3.2 Properties Section, AMQP allows for messages to use a variety of types for the message-id and correlation-id fields of the properties section, specifically message-id-ulong, message-id-uuid, message-id-binary or message-id-string. JMS utilises the String type for both JMSMessageID and JMSCorrelationID. Due to this difference in possible types, it is necessary to support controlling and preserving the type information for the underlying fields of the AMQP message when setting and accessing the JMSMessageID and JMSCorrelationID headers of the JMS Message, both for basic interoperability and because it is common to retrieve a message id and then use it as a correlation id value.

To that end, implementing JMS clients MUST support the behaviour described herein for encoding/decoding all JMSMessageID values, and any JMSCorrelationID values which represent a JMSMessageID rather than an application-specific value, in order to influence values sent and received using the underlying message-id and correlation-id fields of the AMQP properties section.

After the client removes, or before it adds, the synthesized “ID:” prefix of such values, the following mapping is defined between the associated AMQP types and a String representation thereof:

- message-id-binary: “AMQP_BINARY:<hex representation of bytes>”
- message-id-uuid: “AMQP_UUID:<string representation of uuid>”
- message-id-ulong: “AMQP_ULONG:<string representation of ulong>”
- message-id-string: “<original string>”
- message-id-string: “AMQP_STRING:<original string>”

The AMQP_STRING encoding prefix exists only to escape message-id-string values that represent a JMSMessageID (for either JMSMessageID or JMSCorrelationID) and would happen to continue with one of the encoding prefixes above (including AMQP_STRING itself). It MUST NOT be used otherwise by the client library.

For the AMQP_BINARY encoding prefix, the client MUST return upper-case hex characters when the `getJMSMessageID` and `getJMSCorrelationID` methods of `Message` are used, but MUST accept both upper-case and lower-case
values via the `setJMSMessageID(String id)` and `setJMSCorrelationID(String id)` methods.

When `JMSCorrelationID` is set using the `setJMSCorrelationID(String id)` method, any value that begins with the "ID:" prefix of a `JMSMessageID` and attempts to identify itself as an encoded `message-id-binary`, `message-id-uuid`, or `message-id-ulong` but which can’t be converted into the indicated format MUST cause an appropriate JMSException to be thrown. For example, "ID:AMQP_ULONG:foo" can’t be converted to a `message-id-ulong` and so MUST cause an exception.

If implemented, the `getJMSCorrelationIDAsBytes()` method of the `Message` MUST throw an exception if the type of the `correlation-id` field of `properties` is not `message-id-binary`.

The following table provides examples of various attempts to set `JMSCorrelationID`, and the associated output including whether the "x-opt-app-correlation-id" is set true to indicate an application-specific value. The mapping for the `getJMSCorrelationID` method is essentially the reverse of this process.

<table>
<thead>
<tr>
<th>Method</th>
<th>AMQP type</th>
<th>AMQP value</th>
<th>Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setJMSCorrelationID(&quot;ID:foo&quot;)</code></td>
<td><code>message-id-string</code></td>
<td>&quot;foo&quot;</td>
<td>X</td>
</tr>
<tr>
<td><code>setJMSCorrelationID(&quot;ID:42&quot;)</code></td>
<td><code>message-id-string</code></td>
<td>&quot;42&quot;</td>
<td>X</td>
</tr>
<tr>
<td><code>setJMSCorrelationID( &quot;ID:AMQP_STRING:AMQP_ULONG:42&quot;)</code></td>
<td><code>message-id-string</code></td>
<td>&quot;AMQP_ULONG:42&quot;</td>
<td>X</td>
</tr>
<tr>
<td><code>setJMSCorrelationID( &quot;ID:AMQP_STRING:AMQP_STRING:foo&quot;)</code></td>
<td><code>message-id-string</code></td>
<td>&quot;AMQP_STRING:foo&quot;</td>
<td>X</td>
</tr>
<tr>
<td><code>setJMSCorrelationID(&quot;app-spec&quot;)</code></td>
<td><code>message-id-string</code></td>
<td>&quot;app-spec&quot;</td>
<td>true</td>
</tr>
<tr>
<td><code>setJMSCorrelationID(&quot;AMQP_ULONG:42&quot;)</code></td>
<td><code>message-id-string</code></td>
<td>&quot;AMQP_ULONG:42&quot;</td>
<td>true</td>
</tr>
<tr>
<td><code>setJMSCorrelationID(&quot;AMQP_ULONG:foo&quot;)</code></td>
<td><code>message-id-string</code></td>
<td>&quot;AMQP_ULONG:foo&quot;</td>
<td>true</td>
</tr>
<tr>
<td><code>setJMSCorrelationID(&quot;ID:AMQP_ULONG:42&quot;)</code></td>
<td><code>message-id-ulong</code></td>
<td>42</td>
<td>X</td>
</tr>
<tr>
<td><code>setJMSCorrelationID( &quot;ID:AMQP_UUID:&quot; + &lt;uuid&gt; )</code></td>
<td><code>message-id-uuid</code></td>
<td>&lt;uuid&gt;</td>
<td>X</td>
</tr>
<tr>
<td><code>setJMSCorrelationID( &quot;ID:AMQP_BINARY:0123ABCD&quot;)</code></td>
<td><code>message-id-binary</code></td>
<td>0x0123ABCD</td>
<td>X</td>
</tr>
<tr>
<td><code>setJMSCorrelationIDAsBytes( 0x0123ABCD)</code></td>
<td><code>message-id-binary</code></td>
<td>0x0123ABCD</td>
<td>X</td>
</tr>
</tbody>
</table>

The behaviour for `setJMSCorrelationID` and `getJMSCorrelationID` methods is related, with the key distinction that all `JMSMessageID` are prefixed “ID:” and there is no corresponding ‘AsBytes’ alternative methods.

### 4.2.2 JMS-defined 'JMSX' Properties

The following section describes how each of the JMS-defined 'JMSX' Properties can be mapped to an AMQP Message.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMSXUserId</td>
<td>The JMSXUserId property is equivalent to the user-id field of <code>properties</code>. The JMSXUserId is specified as String, while the user-id field of <code>properties</code> is specified as type <code>binary</code>. To maintain end-to-end fidelity for this property, implementations SHOULD convert between AMQP binary and Java String by using the ISO-8859-1 [ISO88591] character set.</td>
</tr>
</tbody>
</table>
### JMSXAppID
The JMSXAppID property is defined as a Java String representing the identity of the application sending the message. If this property is supported by the client library, it MUST be stored in the application-properties section of the AMQP message.

### JMSXDeliveryCount
*Mandatory since JMS 2.0*
This property is set by the client provider on receipt of the message, based on handling of the delivery-count field of header.

See 8. Delivery Count Handling for more details on handling of the delivery-count value.

### JMSXGroupID
The JMSXGroupID property is equivalent to the group-id field of properties.

### JMSXGroupSeq
The JMSXGroupSeq property is equivalent to the group-sequence field of properties.

### JMSXProducerTXID
No standard mapping is provided for JMSXProducerTXID nor is a relation of its semantics to AMQP provided.

### JMSXConsumerTXID
No standard mapping is provided for JMSXConsumerTXID nor is a relation of its semantics to AMQP provided. Should the semantics of this property be defined with respect to AMQP it would not affect the on-the-wire encoding as this property is defined to be set by the JMS on receipt of the message at the client.

### JMSXRcvTimestamp
This value is (if supported) set by the client provider on receipt of the message, it is not transported on the wire and therefore does not need to be mapped to AMQP.

### JMSXState
There is no direct mapping of the JMSXState property to AMQP. It is advised that implementers do not attempt to provide any sort of implementation of this property.

### 4.2.3 JMS Properties

JMS Properties set by applications will be stored in the application-properties section, including some JMS-defined 'JMSX' Properties. If no such properties are set, the application-properties section MAY be omitted.

The JMS Specification defines a number of restrictions on the allowable keys and values for JMS Properties. A JMS property key is of type String and, in addition to other naming restrictions, are forbidden to be null or the empty String. Keys in the application-properties section MUST be of type string, thus precluding null values, but impose no other restriction. The value of a JMS property may only be of the types given in 2. Mapping JMS Types to AMQP (excluding char, which is not allowed in this context). There are no such restrictions on the values within the application-properties section.

### 4.2.4 Message Body Types

JMS defines a number of standard Message body types. These different forms of body each need to be encoded in a defined manner such that Messages which are communicated from one provider to another may be reassembled.
into the correct message type with full fidelity. Moreover this definition allows for non-JMS producers to create messages of a form where their interpretation by a JMS client can be predicted.

Different Message body formats can be expressed through the use of different types of application-data sections within the encoded AMQP message, different values within those sections, and by using fields in the message properties section to indicate the nature of the body content.

4.2.4.1 BytesMessage

A BytesMessage is encoded using zero or more body sections of type data. When data sections are included, the content-type field of properties SHOULD contain the symbol value "application/octet-stream". The data section MAY be omitted when the content is zero-length, in which case the content-type field of properties MUST contain the symbol value "application/octet-stream".

The getBodyLength() method on BytesMessage MUST return the combined length of the data sections, or 0 if none are present.

Figure 4.3: AMQP Message Structure of a BytesMessage

TODO (intent): confirm we can omit the data section entirely?

4.2.4.2 TextMessage

A TextMessage is encoded as an amqp-value section containing a single encoded string or null. The amqp-value section MAY be omitted when the TextMessage body is null. If the amqp-value section is included, the content-type field of properties SHOULD NOT be set. If the amqp-value section is omitted, the content-type field of properties MUST contain the symbol value "text/plain".
Text Body

(amqp-value section containing a string or null)

+--------+-------------+-------------+------------+--------------+-------+--------+
| header | delivery- | message- | properties | application- | amqp- | footer |
| | annotations | annotations | | properties | value | |
+--------+-------------+-------------+------------+--------------+-------+--------+

optional

- amqp-value section SHOULD be present and contain a single encoded string or null.
- content-type field of properties section SHOULD NOT be set if amqp-value is included.
- content-type field of properties section MUST contain "text/plain" if amqp-value is omitted.

Figure 4.4: AMQP Message Structure of a TextMessage

TODO (intent): confirm we can omit the data section entirely?

TODO (intent): charset in content-type?

4.2.4.3 MapMessage

A MapMessage body is encoded as a single amqp-value section containing a single map value. As a result, the content-type field of properties SHOULD NOT be set.

Note that this restricts the MapMessage to having at most $2^{31} - 1$ entries, and at most $2^{32} - 1$ octects of encoded map content. Attempting to send a MapMessage which exceeds these limits MUST result in an appropriate JMSException being thrown.

Map Body

(one amqp-value section containing a map)

+--------+-------------+-------------+------------+--------------+-------+--------+
| header | delivery- | message- | properties | application- | amqp- | footer |
| | annotations | annotations | | properties | value | |
+--------+-------------+-------------+------------+--------------+-------+--------+

optional

- content-type field of properties section SHOULD NOT be set.

Figure 4.5: AMQP Message Structure of a MapMessage

The JMS Specification defines a number of restrictions on the allowable keys and values for MapMessage entries. A key is of type String and the values can be only of the types given in 2. Mapping JMS Types to AMQP. There are no such restrictions on the keys and values in an AMQP map value.

TODO (intent): sending non-JMS types?
4.2.4.4 StreamMessage

A StreamMessage body is encoded as a single amqp-value section containing a single list. As a result, the content-type field of properties SHOULD NOT be set.

Note that this restricts the StreamMessage to having at most $2^{32} - 1$ elements, and at most $2^{32} - 1$ octets of encoded list content. Attempting to send a StreamMessage which exceeds these limits MUST result in an appropriate JMSException being thrown.

The JMS Specification restricts the allowable entry values for a StreamMessage entries to be only of the types given in 2. Mapping JMS Types to AMQP. There are no such restrictions on the entries in an AMQP list value.

TODO (intent): sending non-JMS types?

4.2.4.5 ObjectMessage

An ObjectMessage is encoded using zero or more body sections of type data, where the content is either (i) empty, or (ii) a single encoded AMQP binary value containing serialised object data. If multiple data sections are used, e.g. because the serialised object data exceeds the limits of a single binary value, each subsequent data section MUST contain a binary value holding a continuation of the serialised object content in the previous section. The data sections MAY be omitted when the ObjectMessage payload is empty or null. In all cases, the content-type field of properties MUST contain the symbol value "application/x-java-serialized-object".

TODO (intent): confirm we can omit the data section entirely?
4.3 Mapping AMQP Messages To Java

The previous section defined how a Message as defined by the JMS specification is mapped into AMQP in order to achieve interoperability. In this section the mapping of both these and other arbitrary messages from an AMQP to JMS will defined.

4.3.1 Header Section

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>durable</td>
<td>When receiving a message, the <code>durable</code> field of header MUST be taken to be equivalent to the <code>JMSDeliveryMode</code> header of the Message. If the <code>durable</code> field of header is set to <code>false</code> or isn’t set then the <code>JMSDeliveryMode</code> MUST be taken to be <code>NON_PERSISTENT</code>. When the <code>durable</code> field of header is set to <code>true</code> the <code>JMSDeliveryMode</code> of the Message MUST be taken to be <code>PERSISTENT</code>.</td>
</tr>
<tr>
<td>priority</td>
<td>This field is equivalent to the <code>JMSPriority</code> header of the Message. <code>JMSPriority</code> is specified as being of type <code>int</code> despite the valid values only being 0-9. AMQP allows for the <code>priority</code> field of header to be any valid ubyte value. When receiving a message with the <code>priority</code> field of header greater than 9, the <code>JMSPriority</code> MUST be set to 9. If the <code>priority</code> field of header is unset then the <code>JMSPriority</code> MUST be taken to be <code>DEFAULT_PRIORITY</code> (i.e. the value 4).</td>
</tr>
<tr>
<td>ttl</td>
<td>This field defines the number of milliseconds for which a given message is considered “live”. There is no direct equivalent for the <code>ttl</code> field of header in the JMS specification. If and only if the <code>absolute-expiry-time</code> field of properties is not set, <code>JMSExpiration</code> SHOULD be based on the <code>ttl</code> field of header if set, by summing it with the current time in milliseconds since the Unix Epoch.</td>
</tr>
<tr>
<td>first-acquirer</td>
<td>This field does not have a direct equivalent within the JMS specification, although JMSRedelivered is related, and so vendor property <code>JMS_AMQP_FIRST_ACQUIRER</code> SHOULD be used. For further details, see 5. JMS Vendor Properties.</td>
</tr>
</tbody>
</table>
AMQP uses the delivery-count field of header to track previously failed delivery attempts for a message, with the first delivery attempt having a value of zero, and so on.

JMSXDeliveryCount is defined as a Java int count of delivery attempts, set by the provider on receive, where the first delivery attempt has value 1, the second has value 2 and so on.

The value of JMSXDeliveryCount property is thus equal to delivery-count + 1.

The JMSRedelivered header MUST be considered to be true if and only if the delivery-count field of header has a value greater than 0.

See 8. Delivery Count Handling for more details on handling of this field.

### 4.3.2 Properties Section

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>message-id</td>
<td>This field is equivalent to the JMSMessageID header of the Message. The JMSMessageID value is a Java String whereas the message-id field of properties is defined as being of type providing message-id, that is message-id-ulong, message-id-uuid, message-id-binary or message-id-string. The JMS client library MUST prefix “ID:” to the value of the message-id field of properties before returning it as the JMSMessageID value. See 4.2.1.1 JMSMessageID And JMSCorrelationID Handling for REQUIRED detail relating to supporting usage of the various AMQP types possible for the message-id field of properties.</td>
</tr>
<tr>
<td>user-id</td>
<td>This field is equivalent to the JMSXUserId header. JMSXUserId is specified as being of type String, while the user-id field of properties field is specified as type binary. To maintain end-to-end fidelity for this property implementations SHOULD convert between AMQP binary and Java String by using the ISO-8859-1 [ISO88591] character set.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>to</td>
<td>This field is equivalent to the JMSDestination header. JMSDestination is defined as being of the JMS Destination type, while the to field of properties requires an address-string. See 6. Destinations for REQUIRED detail regarding how conversion between these types is achieved if the to field of properties was set. If the to field of properties was not set on a received message, the JMSDestination header value SHOULD be derived from the Destination to which the receiving consumer was established.</td>
</tr>
<tr>
<td>subject</td>
<td>This field does not have an equivalent within the JMS specification, and so the vendor property JMS_AMQP_SUBJECT SHOULD be used. For further details, see 5. JMS Vendor Properties.</td>
</tr>
<tr>
<td>reply-to</td>
<td>This field is equivalent to the JMSReplyTo header. JMSReplyTo is defined as being of the JMS Destination type, while the reply-to field of properties requires an address-string. See 6. Destinations for REQUIRED detail regarding how conversion between these types is achieved if the reply-to field of properties was set.</td>
</tr>
<tr>
<td>correlation-id</td>
<td>This field is equivalent to the JMSCorrelationID header of the Message. The JMSCorrelationID value is a Java String whereas the correlation-id field of properties is defined as being of type providing message-id, that is message-id-ulong, message-id-uuid, message-id-binary or message-id-string. Where the correlation-id field of properties for the received message is of type message-id-string and the boolean message annotation with symbol key of “x-opt-app-correlation-id” is either not set or is false, then the correlation-id field of properties MUST be formatted as a JMSMessageID, that is the client library MUST prefix “ID:” to the value before returning it as the JMSCorrelationID value. See 4.2.1.1 JMSMessageID And JMSCorrelationID Handling for REQUIRED detail relating to supporting usage of the various AMQP types possible for the correlation-id field of properties.</td>
</tr>
<tr>
<td>content-type</td>
<td>This field does not have an equivalent within the JMS specification, and so the vendor property JMS_AMQP_CONTENT_TYPE SHOULD be used. For further details, see 5. JMS Vendor Properties.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>content-encoding</td>
<td>This field does not have an equivalent within the JMS specification, and so the vendor property JMS_AMQP_CONTENT_ENCODING SHOULD be used. For further details, see 5. JMS Vendor Properties.</td>
</tr>
<tr>
<td>absolute-expiry-time</td>
<td>This field is equivalent to the JMSExpiration message header.</td>
</tr>
<tr>
<td></td>
<td>If the absolute-expiry-time field of properties is set, then JMSExpiration MUST have the equivalent Java long value, representing the time at which the message expires, in milliseconds since the Unix Epoch.</td>
</tr>
<tr>
<td></td>
<td>If the absolute-expiry-time field of properties is not set then JMSExpiration SHOULD be based on the ttl field of header if set, see 4.3.1 Header Section for more details.</td>
</tr>
<tr>
<td>creation-time</td>
<td>This field is equivalent to the JMSTimestamp message header.</td>
</tr>
<tr>
<td></td>
<td>If the creation-time field of properties is not set, then JMSTimestamp MUST have the value zero. If the creation-time field of properties field is set, then JMSTimestamp MUST have the equivalent Java long value, representing the time at which the message was sent/created, in milliseconds since the Unix Epoch.</td>
</tr>
<tr>
<td>group-id</td>
<td>This field is equivalent to the JMS-defined JMSXGroupID message property.</td>
</tr>
<tr>
<td>group-sequence</td>
<td>This field is equivalent to the JMS-defined JMSXGroupSeq message property.</td>
</tr>
<tr>
<td>reply-to-group-id</td>
<td>This field does not have an equivalent within the JMS specification, and so the vendor property JMS_AMQP_REPLY_TO_GROUP_ID MUST be used. For further details, see 5. JMS Vendor Properties.</td>
</tr>
</tbody>
</table>

4.3.3 Application Properties Section

The application-properties section contents are equivalent to the JMS Message Properties.

TODO (intent): how to handle receiving (and sending?) the following:

- String property names which do not conform with the JMS restrictions on naming
- property values with types not defined in the JMS specification

4.3.4 Delivery Annotations Section

TODO (content):
4.3.5 Message Annotations Section

TODO (content):

4.3.6 Footer Section

TODO (content):

4.3.7 Body Sections

The type and content of the message body received will influence the particular JMS Message type used to represent the AMQP message.

4.3.7.1 No Body

Where no body sections are received and the content-type field of properties is either not set, or set to the symbol value "application/octet-stream" the message SHOULD be interpreted as a BytesMessage with zero-length content.

Where no body sections are received and the content-type field of properties is set to the symbol value "application/x-java-serialized-object" the message SHOULD be interpreted as an ObjectMessage with null content.

Where no body sections are received and the content-type field of properties is set to the symbol value "text/plain" the message SHOULD be interpreted as a TextMessage with null content.

TODO (intent): charset in content-type?

TODO (intent): confirm we can omit the body section entirely?

4.3.7.2 Data

Where one or more data sections are received and the content-type field of properties is either not set, or set to the symbol value "application/octet-stream" the message SHOULD be interpreted as a BytesMessage.

Where one or more data sections are received and the content-type field of properties is set to the symbol value "application/x-java-serialized-object" the message SHOULD be interpreted as an ObjectMessage.

Where one data section is received and the content-type field of properties is set to the symbol value "text/plain", the message SHOULD be interpreted as a TextMessage. Where the data section is empty, then the return value from the getText() method MUST be a Java String of length 0.

TODO (intent): charset in content-type?

4.3.7.3 Amqp-value

Where an amqp-value body section is received that contains a string value, the message SHOULD be interpreted as a TextMessage.

Where an amqp-value body section is received that contains a null value, the message SHOULD be interpreted as a TextMessage with null content.
Where an amqp-value body section is received that contains a map value, the message SHOULD be interpreted as a MapMessage.

Where an amqp-value body section is received that contains a list value, the message SHOULD be interpreted as a StreamMessage.

Where an amqp-value body section is received that contains a binary value, the message SHOULD be interpreted as a BytesMessage.

4.3.7.4 Todo

TODO (intent): how to handle receiving the following:

- Any other typical alternative representations of the JMS message types
- multiple body sections which were not described previously (i.e not data)
- bodies containing non-JMS types

TODO (content): discuss scope for receiving AMQP encoded types as a particular variety of Message or body type (e.g receive a map as a java.util.Map via an ObjectMessage, rather than receiving a MapMessage). The new JMS 2.0 getBody() method both eases and complicates this.
## 5 JMS Vendor Properties

This document defines the following JMS Vendor Properties.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Set By</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JMS_AMQP_TTL</td>
<td>Application</td>
<td>Optionally used for controlling the value of the ttl field of header for the outgoing AMQP message independently from the value normally used due to the JMS Time To Live value applied when sending the message. If set, it MUST be a long property with a value in the range zero to 2^32 - 1. If the property value is zero then the ttl field of header MUST be omitted rather than set to zero. When setting the ttl field of header by using the JMS_AMQP_TTL property, an entry with this key MUST NOT be included in the application-properties section of the transmitted AMQP message.</td>
</tr>
<tr>
<td>JMS_AMQP_FIRST_ACQUIRER</td>
<td>Provider on Receive</td>
<td>Optionally used for accessing the first-acquirer field of header. If set, it MUST be of type boolean.</td>
</tr>
<tr>
<td>JMS_AMQP_SUBJECT</td>
<td>Application/ Provider on Receive</td>
<td>Optionally used for setting and/or accessing the subject field of properties. If set, it MUST be of type String.</td>
</tr>
<tr>
<td>JMS_AMQP_CONTENT_TYPE</td>
<td>Application/ Provider on Receive</td>
<td>Optionally used for setting and/or accessing the content-type field of properties to distinguish the content type within the message body where necessary. If set, it MUST be of type String.</td>
</tr>
<tr>
<td>JMS_AMQP_CONTENT_ENCODING</td>
<td>Application/ Provider on Receive</td>
<td>Optionally used for setting and/or accessing the content-encoding field of properties to distinguish the content encoding within the message body where necessary. If set, it MUST be of type String.</td>
</tr>
<tr>
<td>JMS_AMQP_REPLY_TO_GROUP_ID</td>
<td>Application/ Provider on Receive</td>
<td>Optionally used for setting and/or accessing the reply-to-group-id field of properties. If set, it MUST be of type String.</td>
</tr>
</tbody>
</table>

Each implementation MAY, in addition, define its own extension properties but these MUST NOT use AMQP as the "vendor" name, i.e. the additional extension property names MUST NOT begin with "JMS_AMQP".

**TODO (presentation):** Decide where this goes, it isn’t necessarily a section.
6 Destinations

In order to faithfully re-construct the Destination objects used in the JMSDestination and JMSReplyTo headers of a Message following its transmission via AMQP, information regarding the particular type of Destination object also has to be transmitted in an interoperable fashion.

This type information is transferred via message annotations with symbol keys of “x-opt-to-type” and “x-opt-reply-type”. The value of these annotation is of type string containing a comma-separated set of destination type attributes. Possible attributes include “queue”, “topic”, and “temporary”. These are used to define the following possible annotation values:

<table>
<thead>
<tr>
<th>Destination Type</th>
<th>Annotation value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue</td>
<td>“queue”</td>
</tr>
<tr>
<td>TemporaryQueue</td>
<td>“queue,temporary”</td>
</tr>
<tr>
<td>Topic</td>
<td>“topic”</td>
</tr>
<tr>
<td>TemporaryTopic</td>
<td>“topic,temporary”</td>
</tr>
</tbody>
</table>

Producing JMS clients SHOULD set the “x-opt-to-type” message annotation on each message sent. Producing JMS clients SHOULD set the “x-opt-reply-type” message annotation on each message sent that has a JMSReplyTo header value.

When receiving an AMQP message the lacks the message annotations outlined above, additional steps are necessary order to ensure that the JMSDestination and/or JMSReplyTo headers can be populated appropriately. If the “x-opt-to-type” and/or “x-opt-reply-type” message annotations are not present, the JMSDestination and/or JMSReplyTo values respectively SHOULD be constructed using the same Destination type derivative as that used to create the consumer which received the message.

When receiving an AMQP message the lacks the to field of properties, receiving JMS clients SHOULD synthesize this by returning the Destination value supplied when creating the consumer which received the message.

TODO (intent): Change the annotations to use a ubyte for efficiency?

TODO (intent): Define that x-opt-reply-type = x-opt-to-type unless both are set, so allowing that only one be set?

TODO (presentation): Decide where this goes, it isn’t necessarily a section.
7 Message Producers

7.1 Sending Messages

JMS producers (e.g. MessageProducer) are required to set various headers on a message during the sending operation.

For the JMSExpiration header, specific handling was discussed in 4.2.1 JMS Headers. However, beyond setting the JMSExpiration header with the computed expiration, producing JMS clients need additionally ensure an appropriate value for the ttl field of header on outgoing messages.

If the JMS_AMQP_TTL vendor property outlined in 5. JMS Vendor Properties has been set on the Message, its value SHOULD be used to populate the ttl field of header.

If the JMS_AMQP_TTL vendor property has not been set and a Time To Live value of 0 is applicable when sending a Message, then producing JMS clients MUST NOT set the ttl field of header, that is it MUST be omitted rather than set to zero.

If the JMS_AMQP_TTL vendor property has not been set, and a non-zero Time To Live value of $2^{32} - 1$ or less is applicable when sending a Message, the ttl field of header MUST be set accordingly by the provider on the AMQP message. If the applicable Time To Live value exceeds $2^{32} - 1$ then the ttl field of header MUST be omitted instead rather than populated with a value less than specified by the application.
8 Delivery Count Handling

TODO (intent): define handling for delivery-count and its relationship to JMSXDeliveryCount and JMSRedelivered. That is, when to update it based on rollback, recover etc (and how this further depends on the way those methods are actually implemented, i.e locally or by pushing them back to the source). Decide where this goes, it isn’t necessarily a section.
9 Supplementary Definitions

<table>
<thead>
<tr>
<th>Annotation Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-opt-app-correlation-id</td>
<td>For further details, see 4.2.1 JMS Headers and 4.3.2 Properties Section</td>
</tr>
<tr>
<td>x-opt.jms-type</td>
<td>For further details, see 4.2.1 JMS Headers</td>
</tr>
<tr>
<td>x-opt-delivery-time</td>
<td>For further details, see 4.2.1 JMS Headers</td>
</tr>
<tr>
<td>x-opt-to-type</td>
<td>For further details, see 6. Destinations</td>
</tr>
<tr>
<td>x-opt-reply-type</td>
<td>For further details, see 6. Destinations</td>
</tr>
</tbody>
</table>

TODO (content): add annotations to registry, back-reference these definitions.

TODO (presentation): Decide where this goes, it isn’t necessarily a section.