



---

# Advanced Message Queuing Protocol (AMQP) JMS Mapping Version 1.0

## Working Draft 2

16 December 2013

### Specification URIs

**This version:**

<http://docs.oasis-open.org/amqp-bindmap/jms/v1.0/wd02/amqp-bindmap-jms-v1.0-wd02.xml> (Authoritative)

<http://docs.oasis-open.org/amqp-bindmap/jms/v1.0/wd02/amqp-bindmap-jms-v1.0-wd02.html>

<http://docs.oasis-open.org/amqp-bindmap/jms/v1.0/wd02/amqp-bindmap-jms-v1.0-wd02.pdf>

**Previous version:**

N/A

**Latest version:**

<http://docs.oasis-open.org/amqp-bindmap/jms/v1.0/amqp-bindmap-jms-v1.0.html>

<http://docs.oasis-open.org/amqp-bindmap/jms/v1.0/amqp-bindmap-jms-v1.0.pdf>

<http://docs.oasis-open.org/amqp-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](mailto:shuston@riverace.com)), Individual

**Editors:**

Robert Gemmell ([robert.gemmell@jpmchase.com](mailto:robert.gemmell@jpmchase.com)), JPMorgan Chase & Co.

Robert Godfrey ([robert.godfrey@jpmorgan.com](mailto: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]**

*Advanced Message Queuing Protocol (AMQP) JMS Mapping Version 1.0*. 16 December 2013. OASIS Working Draft 2.

<http://docs.oasis-open.org/amqp-bindmap/jms/v1.0/wd02/amqp-bindmap-jms-v1.0-wd02.pdf>.

---

## Notices:

Copyright © OASIS Open 2013. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

---

# Contents

<b>1</b>	<b>References</b>	<b>4</b>
<b>2</b>	<b>Mapping JMS Types to AMQP</b>	<b>5</b>
<b>3</b>	<b>Mapping AMQP Types to Java</b>	<b>6</b>
<b>4</b>	<b>Messages</b>	<b>7</b>
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 . . . . .	12
4.2.3	JMS Properties . . . . .	13
4.2.4	Message Body Types . . . . .	13
4.3	Mapping AMQP Messages To Java . . . . .	16
4.3.1	Header Section . . . . .	16
4.3.2	Properties Section . . . . .	17
4.3.3	Application Properties Section . . . . .	20
4.3.4	Delivery Annotations Section . . . . .	20
4.3.5	Message Annotations Section . . . . .	20
4.3.6	Footer Section . . . . .	20
4.3.7	Body Sections . . . . .	20
<b>5</b>	<b>JMS Vendor Properties</b>	<b>22</b>
<b>6</b>	<b>Destinations</b>	<b>23</b>
<b>7</b>	<b>Message Producers</b>	<b>24</b>
7.1	Sending Messages . . . . .	24
<b>8</b>	<b>Delivery Count Handling</b>	<b>25</b>
<b>9</b>	<b>Supplementary Definitions</b>	<b>26</b>

---

# 1 References

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

## **[AMQPMSGFORMAT]**

*AMQP Message Format* <http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-messaging-v1.0-os.html#section-message-format>

## **[ISO88591]**

*ISO-8859-1* [http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=28245](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=28245)

---

## 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].

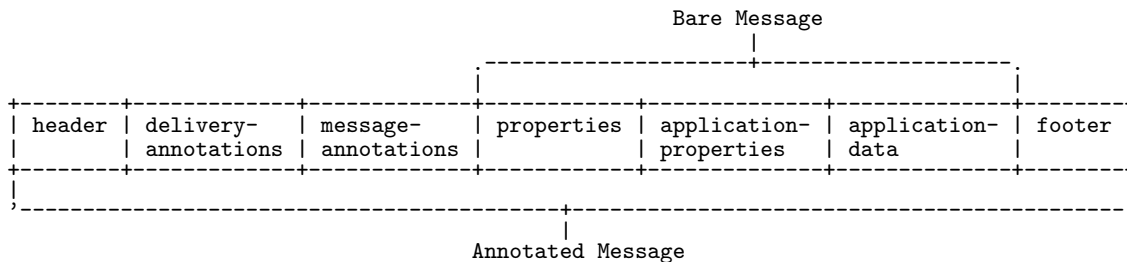


Figure 4.1: AMQP Message Structure

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:

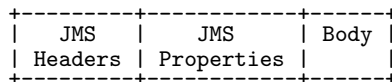


Figure 4.2: JMS Message

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.

Header Name	Description
JMSTimestamp	<p>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 the <code>creation-time</code> field of <code>properties</code> for the same purpose.</p>
JMSMessageID	<p>The JMSMessageID is defined as a Java String identifier for the Message which is set by the implementation during publication. AMQP uses the <code>message-id</code> field of <code>properties</code> for the same purpose, which is defined as being of type providing <i>message-id</i>, such as <code>message-id-ulong</code>, <code>message-id-uuid</code>, <code>message-id-binary</code> Or <code>message-id-string</code>.</p> <p>JMSMessageID values are required to have a prefix of <i>"ID:"</i>, however this prefix MUST NOT be part of the value of the <code>message-id</code> field of <code>properties</code> sent by producing JMS clients, and MUST NOT be required receiving JMS clients (i.e. this prefix should be synthesized by the client library).</p> <div data-bbox="657 835 1312 913" style="border: 1px solid black; background-color: #f8d7da; padding: 5px; border-radius: 5px; margin-top: 10px;">TODO (intent): recommend one AMQP type (e.g string)?</div>

JMSCorrelationID	<p>The JMSCorrelationID header is defined as a Java String or byte [] used to link one message with another.</p> <p>AMQP uses the correlation-id field of properties for the same purpose, which is defined as being of type providing <i>message-id</i>, such as message-id-ulong, message-id-uuid, message-id-binary or message-id-string.</p> <p>JMSMessageID String values are required to have a prefix of "ID:", however when using these as the value for JMSCorrelationID, the "ID:" prefix MUST NOT be included in the correlation-id field of properties sent by the producing JMS client and MUST NOT be required by receiving JMS clients (i.e. this prefix must be synthesized by the client library).</p> <p>When setting a String value for JMSCorrelationID with prefix of "ID:", this SHOULD be a valid JMSMessageID value from the same provider implementation as message id values from alternative providers may not be accepted.</p> <p>To signal to receiving JMS clients in an interoperable way that the the correlation-id field of properties represents an application-specific String or byte [], 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 <i>false</i>. When setting JMSCorrelationID to an application-specific String or a byte [] value, the annotation MUST be set to <i>true</i>.</p>
JMSReplyTo	<p>The JMSReplyTo header is equivalent to the reply-to field of properties.</p> <p>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 should be achieved.</p>

<p>JMSDestination</p>	<p>The JMSDestination header is equivalent to the <code>to</code> field of <code>properties</code>.</p> <p>Note that producers <b>MUST</b> set the <code>to</code> field of <code>properties</code> explicitly (intermediaries cannot derive it from address of the target of the link on which the message was sent).</p> <p>JMSDestination is defined as being of the JMS Destination type, while the <code>reply-to</code> field of <code>properties</code> requires an <code>address-string</code>. See 6. Destinations for REQUIRED detail as to how conversion between these types should be achieved.</p>
<p>JMSDeliveryMode</p>	<p>The JMSDeliveryMode header is defined as a Java <code>int</code> with two possible values: <code>NON_PERSISTENT</code> and <code>PERSISTENT</code>.</p> <p>The JMSDeliveryMode header relates to two different aspects of sending a JMS Message as an AMQP message. Firstly, its value is equivalent to the <code> durable </code> field of header. For <code>PERSISTENT</code> messages, the <code> durable </code> field of header should be set to <code>true</code>. For <code>NON_PERSISTENT</code> messages, the <code> durable </code> field of header may be set to <code>false</code> (the implicit default for AMQP).</p> <p>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 <code>PERSISTENT</code> 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 <code>NON_PERSISTENT</code> messages on a non-transacted session an implementer <b>MAY</b> choose to send messages considering them settled as soon as they are sent (i.e. with the <code>settled</code> flag set to <code>true</code> on their original transfer).</p>
<p>JMSRedelivered</p>	<p>This header is set by the client provider on receipt of the message, based on handling of the <code>delivery-count</code> field of header.</p> <p>See 8. Delivery Count Handling for more details on handling of the <code>delivery-count</code> value.</p>

JMSType	<p>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.</p> <p>In order to carry the JMSType value on a message in an interoperable way, a message annotation with symbol key of “<i>x-opt-jms-type</i>” should be used, containing a string representing the JMSType value.</p>
JMSExpiration	<p>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.</p> <p>If a non-zero <i>time-to-live</i> value is specified when sending the message, JMSExpiration contains the computed expiry time. If no <i>time-to-live</i> value (or a value of zero) is supplied when sending the message, then JMSExpiration has the value zero.</p> <p>AMQP uses the <i>absolute-expiry-time</i> field of <i>properties</i> for the purpose of setting an expiration time. When a non-zero value <i>time-to-live</i> is supplied, the computed expiration time should be set in the <i>absolute-expiry-time</i> field of <i>properties</i>. When no <i>time-to-live</i> value (or a value of zero) is supplied and JMSExpiration thus has the value zero, the <i>absolute-expiry-time</i> field of <i>properties</i> MUST be omitted rather than set to zero.</p> <p>See 7.1 Sending Messages for additional REQUIRED detail relating to message expiration.</p>
JMSPriority	<p>The JMSPriority is equivalent to the <i>priority</i> 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.</p>

<p>JMSDeliveryTime</p> <p>*New in JMS 2.0</p>	<p>The JMSDeliveryTime header has no equivalent in AMQP. It is defined as a Java long representing the earliest time at which the message may 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 <i>delivery delay</i> value to the time at which the message is sent.</p> <p>In order to carry the JMSDeliveryTime value on a message in an interoperable way, a message annotation with symbol key of “<i>x-opt-delivery-time</i>” and type timestamp MUST be used if a non-zero <i>delivery delay</i> 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.</p> <p>TODO (): define filter to enforce this on the broker?</p> <p>TODO (): define connection-capability for feature?</p>
---	--

## 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.

Property Name	Description
<p>JMSXUserId</p>	<p>The JMSXUserId property is equivalent to the <i>user-id</i> field of <i>properties</i>. The JMSXUserId is specified as String, while the <i>user-id</i> field of <i>properties</i> is specified as type binary.</p> <p>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.</p>
<p>JMSXAppID</p>	<p>The JMSXAppID property is defined as a Java String representing the identity of the application sending the message. If supported, this property should be stored in the <i>application-properties</i> section of the AMQP message.</p>
<p>JMSXDeliveryCount</p> <p>*Mandatory since JMS 2.0</p>	<p>This property is set by the client provider on receipt of the message, based on handling of the <i>delivery-count</i> field of header.</p> <p>See 8. Delivery Count Handling for more details on handling of the <i>delivery-count</i> value.</p>
<p>JMSXGroupID</p>	<p>The JMSXGroupID property is equivalent to the <i>group-id</i> field of <i>properties</i>.</p>

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 must be 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

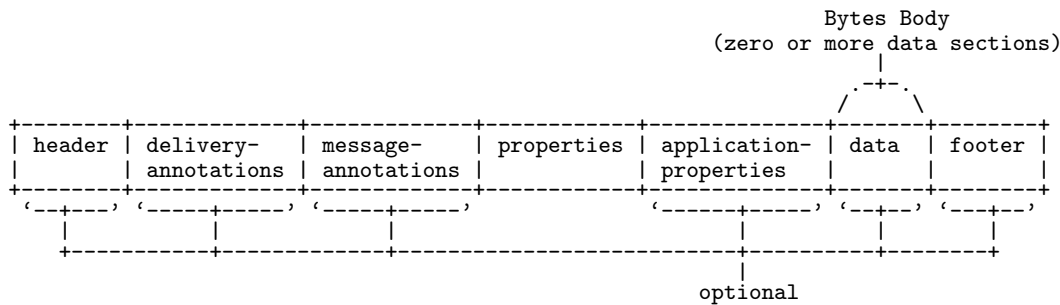
JMS defines a number of standard Message body types. These different forms of body each need to be encoded in a defined manner such that Messages which are communicated from one provider to another may be reassembled into the correct message type with full fidelity. Moreover this definition allows for 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` should return the combined length of the `data` sections, or 0 if none are present.



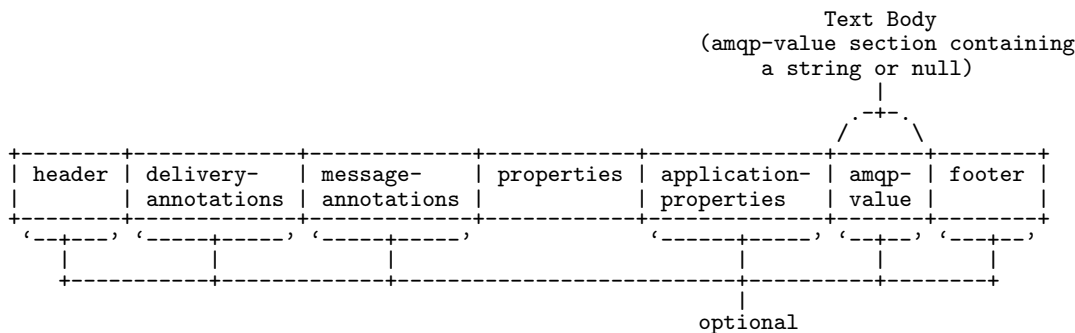
- content-type field of properties section SHOULD be "application/octet-stream" if data included.
- content-type field of properties section MUST be "application/octet-stream" if data is omitted.

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"`.



- 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  $2^{32} - 1$  octets of encoded map content. Attempting to send a `MapMessage` which exceeds these limits should result in an appropriate `JMSEException` being thrown.

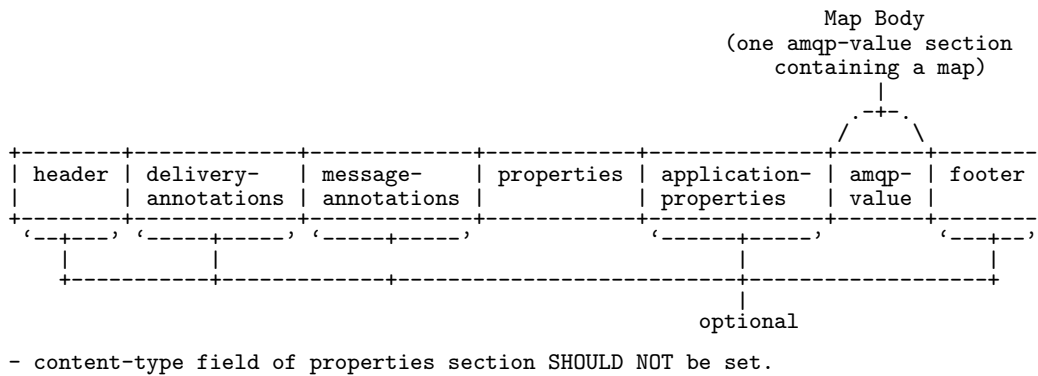


Figure 4.5: AMQP Message Structure of a MapMessage

The JMS Specification defines a number of restrictions on the allowable keys and values for MapMessage entries. A key must be of type String and the values may 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 should result in an appropriate JMSEException being thrown.

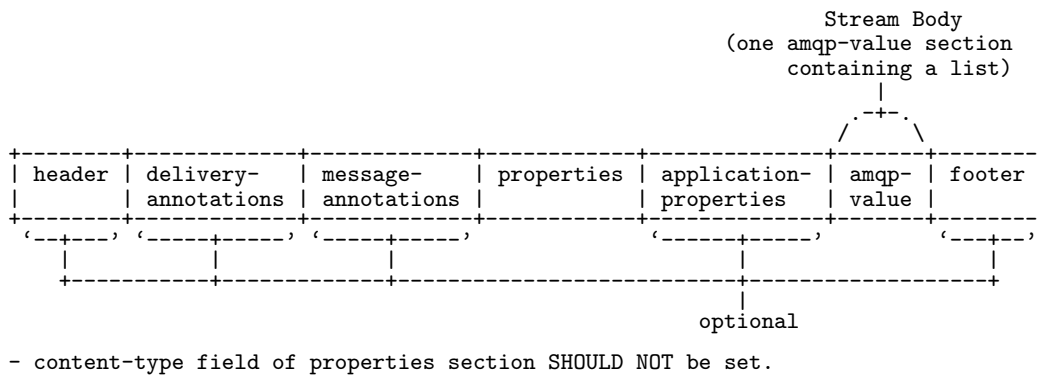


Figure 4.6: AMQP Message Structure of a StreamMessage

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".

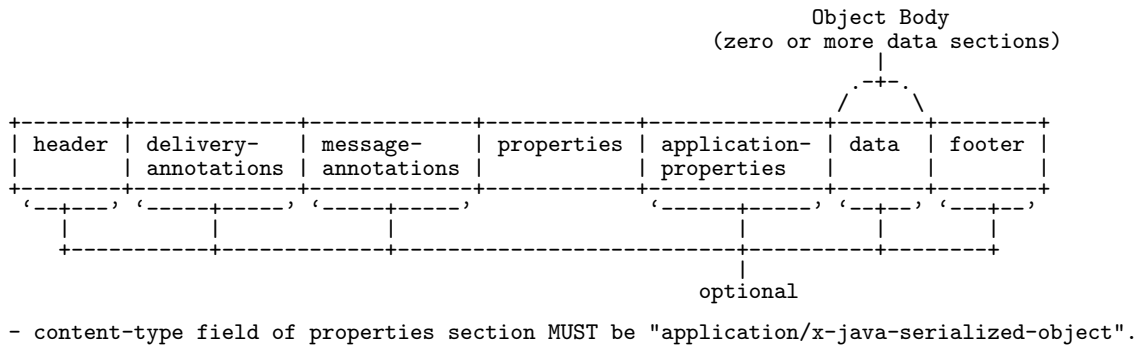


Figure 4.7: AMQP Message Structure of an ObjectMessage

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

TODO (intent): ability to represent objects (e.g maps) using the AMQP type encodings?

### 4.3 Mapping AMQP Messages To Java

The previous section defined how a Message as defined by the JMS specification should be 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 be defined.

#### 4.3.1 Header Section

Field Name	Description
durable	When receiving a message, the durable field of header can be taken to be equivalent to the JMSDeliveryMode header of the Message. If the the durable field of header is set to <i>false</i> or unset then the JMSDeliveryMode should be taken to be NON_PERSISTENT. When the durable field of header is set to <i>true</i> the JMSDeliveryMode of the Message should be taken to be PERSISTENT.

priority	<p>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> should be set to 9. If the <code>priority</code> field of header is unset the the <code>JMSPriority</code> should be taken to be <code>DEFAULT_PRIORITY</code> (i.e. the value 4).</p>
ttl	<p>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.</p> <p>If and only if the <code>absolute-expiry-time</code> field of <code>properties</code> is not set, <code>JMSExpiration</code> should be based on the <code>ttl</code> field of header, by summing it with the current time in milliseconds since the Unix Epoch.</p>
first-acquirer	<p>This field does not have a direct equivalent within the JMS specification, although <code>JMSRedelivered</code> is related, and so vendor property <code>JMS_AMQP_FIRST_ACQUIRER</code> should be used. For further details, see 5. JMS Vendor Properties .</p>
delivery-count	<p>AMQP uses the <code>delivery-count</code> 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.</p> <p><code>JMSXDeliveryCount</code> is defined as a Java <code>int</code> 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.</p> <p>The value of <code>JMSXDeliveryCount</code> property is thus equal to <i>delivery-count + 1</i>.</p> <p>The <code>JMSRedelivered</code> header should be considered to be true if and only if the <code>delivery-count</code> field of header has a value greater than 0.</p> <p>See 8. Delivery Count Handling for more details on handling of this field.</p>

### 4.3.2 Properties Section

Field Name	Description
------------	-------------

message-id	<p>This field is equivalent to the <code>JMSMessageID</code> header of the Message.</p> <p>The <code>JMSMessageID</code> value is a Java String whereas the <code>message-id</code> field of properties is defined as being of type providing <i>message-id</i>, such as <code>message-id-ulong</code>, <code>message-id-uuid</code>, <code>message-id-binary</code> or <code>message-id-string</code>. In order to preserve the type of the <code>message-id</code> field of properties in situations where the <code>JMSMessageID</code> value is used to set a <code>JMSCorrelationID</code> value, the returned <code>JMSMessageID</code> value SHOULD encode the type information.</p> <p><code>JMSMessageID</code> values MUST have a prefix of “ID:”, however it is expected that the received <code>message-id-string</code> does not include this. The receiving JMS client MUST synthesize this prefix if necessary.</p>
user-id	<p>This field is equivalent to the <code>JMSXUserId</code> header.</p> <p><code>JMSXUserId</code> is specified as being of type <code>String</code>, while the <code>user-id</code> field of properties field is specified as type <code>binary</code>. To maintain end-to-end fidelity for this property implementations SHOULD convert between AMQP <code>binary</code> and Java <code>String</code> by using the ISO-8859-1 [ISO88591] character set.</p>
to	<p>This field is equivalent to the <code>JMSDestination</code> header.</p> <p><code>JMSDestination</code> is defined as being of the JMS <code>Destination</code> type, while the <code>to</code> field of properties requires an <code>address-string</code>. See 6. Destinations for REQUIRED detail regarding how conversion between these types should be achieved if the <code>to</code> field of properties was set.</p> <p>If the <code>to</code> field of properties was not set on a received message, the <code>JMSDestination</code> header value SHOULD be derived from the <code>Destination</code> to which the receiving consumer was established.</p>
subject	<p>This field does not have an equivalent within the JMS specification, and so the vendor property <code>JMS_AMQP_SUBJECT</code> should be used. For further details, see 5. JMS Vendor Properties .</p>
reply-to	<p>This field is equivalent to the <code>JMSReplyTo</code> header.</p> <p><code>JMSReplyTo</code> is defined as being of the JMS <code>Destination</code> type, while the <code>reply-to</code> field of properties requires an <code>address-string</code>. See 6. Destinations for REQUIRED detail regarding how conversion between these types should be achieved if the <code>reply-to</code> field of properties was set.</p>

correlation-id	<p>This field is equivalent to the <code>JMSCorrelationID</code> header of the Message.</p> <p>The <code>JMSCorrelationID</code> value is a Java String whereas the <code>correlation-id</code> field of properties is defined as being of type providing <i>message-id</i>, such as <code>message-id-ulong</code>, <code>message-id-uuid</code>, <code>message-id-binary</code> or <code>message-id-string</code>. In order to preserve the type of the <code>correlation-id</code> field of properties for later use in sending a new AMQP message, the <code>JMSCorrelationID</code> value returned SHOULD encode the type information.</p> <p>Where the boolean message annotation with symbol key of "<i>x-opt-app-correlation-id</i>" is either not set on the received message or is false, the <code>correlation-id</code> field of properties value MUST be taken to be the id of a message and be formatted as if it were a <code>JMSMessageID</code>, that is the client library MUST ensure the returned <code>JMSCorrelationID</code> value has prefix "<i>ID:</i>" by synthesizing it if necessary.</p>
content-type	<p>This field does not have an equivalent within the JMS specification, and so the vendor property <code>JMS_AMQP_CONTENT_TYPE</code> should be used. For further details, see 5. JMS Vendor Properties .</p>
content-encoding	<p>This field does not have an equivalent within the JMS specification, and so the vendor property <code>JMS_AMQP_CONTENT_ENCODING</code> should be used. For further details, see 5. JMS Vendor Properties .</p>
absolute-expiry-time	<p>This field is equivalent to the <code>JMSExpiration</code> message header.</p> <p>If the <code>absolute-expiry-time</code> field of properties is set, then <code>JMSExpiration</code> should have the equivalent Java long value, representing the time at which the message expires, in milliseconds since the Unix Epoch.</p> <p>If the <code>absolute-expiry-time</code> field of properties is not set then <code>JMSExpiration</code> should be based on the <code>ttl</code> field of header, see 4.3.1 Header Section for more details.</p>
creation-time	<p>This field is equivalent to the <code>JMSTimestamp</code> message header.</p> <p>If the <code>creation-time</code> field of properties is not set, then <code>JMSTimestamp</code> should have the value zero. If the <code>creation-time</code> field of properties field is set, then <code>JMSTimestamp</code> should have the equivalent Java long value, representing the time at which the message was sent/created, in milliseconds since the Unix Epoch.</p>

group-id	This field is equivalent to the JMS-defined <code>JMSXGroupSeq</code> message property.
group-sequence	This field is equivalent to the JMS-defined <code>JMSXGroupID</code> message property.
reply-to-group-id	This field does not have an equivalent within the JMS specification, and so the vendor property <code>JMS_AMQP_REPLY_TO_GROUP_ID</code> should be used. For further details, see 5. JMS Vendor Properties .

### 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`.

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.

Property Name	Set By	Description
JMS_AMQP_TTL	Application	<p>Optionally used for controlling the value of the <code>ttl</code> field of <code>header</code> for the outgoing AMQP message independently from the value normally used due to the JMS <i>Time To Live</i> value applied when sending the message. If set, it MUST be a <code>long</code> property with a value in the range zero to <math>2^{32} - 1</math>. If the property value is zero then the <code>ttl</code> field of <code>header</code> MUST be omitted rather than set to zero.</p> <p>When setting the <code>ttl</code> field of <code>header</code> 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.</p>
JMS_AMQP_FIRST_ACQUIRER	Provider on Receive	Optionally used for accessing the <code>first-acquirer</code> field of <code>header</code> . If set, it MUST be of type <code>boolean</code> .
JMS_AMQP_SUBJECT	Application/ Provider on Receive	Optionally used for setting and/or accessing the <code>subject</code> field of <code>properties</code> . If set, it MUST be of type <code>String</code> .
JMS_AMQP_CONTENT_TYPE	Application/ Provider on Receive	Optionally used for setting and/or accessing the <code>content-type</code> field of <code>properties</code> to distinguish the content type within the message body where necessary. If set, it MUST be of type <code>String</code> .
JMS_AMQP_CONTENT_ENCODING	Application/ Provider on Receive	Optionally used for setting and/or accessing the <code>content-encoding</code> field of <code>properties</code> to distinguish the content encoding within the message body where necessary. If set, it MUST be of type <code>String</code> .
JMS_AMQP_REPLY_TO_GROUP_ID	Application/ Provider on Receive	Optionally used for setting and/or accessing the <code>reply-to-group-id</code> field of <code>properties</code> . If set, it MUST be of type <code>String</code> .

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 must also 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:

Destination Type	Annotation value
Queue	“queue”
TemporaryQueue	“queue,temporary”
Topic	“topic”
TemporaryTopic	“topic,temporary”

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 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 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 less than  $2^{32}-1$  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

Annotation Name	Reference
x-opt-app-correlation-id	For further details, see 4.2.1 JMS Headers
x-opt-jms-type	For further details, see 4.2.1 JMS Headers
x-opt-delivery-time	For further details, see 4.2.1 JMS Headers
x-opt-to-type	For further details, see 6. Destinations
x-opt-reply-type	For further details, see 6. Destinations

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

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