AMQP 0-9-1 Quick Reference

This page provides a guide to RabbitMQ's implementation of AMQP 0-9-1. In addition to the classes and methods defined in the **AMQP specification**, RabbitMQ supports several **protocol extensions**, which are also listed here. The original and extended specification downloads can be found on the **protocol page**.

For your convenience, links are provided from this guide to the relevant sections of the API guides for the RabbitMQ **Java** and **.NET** clients. Full details of each method and its parameters are available in our **complete AMQP 0-9-1 reference**.

Basic

```
basic.ack(delivery-tag delivery-tag, bit multiple)
```

Acknowledge one or more messages. *Support:* **full**

When sent by the client, this method acknowledges one or more messages delivered via the Deliver or Get-Ok methods. When sent by server, this method acknowledges one or more messages published with the Publish method on a channel in confirm mode. The acknowledgement can be for a single message or a set of messages up to and including a specific message.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
basic.cancel(consumer-tag consumer-tag, no-wait no-wait) → cancel-ok
```

End a queue consumer. *Support:* **full**

This method cancels a consumer. This does not affect already delivered messages, but it does mean the server will not send any more messages for that consumer. The client may receive an arbitrary number of messages in between sending the cancel method and receiving the cancel-ok reply. It may also be sent from the server to the client in the event of the consumer being unexpectedly cancelled (i.e. cancelled for any reason other than the server receiving the corresponding basic.cancel from the client). This allows clients to be notified of the loss of consumers due to events such as queue deletion. Note that as it is not a MUST for clients to accept this method from the client, it is advisable for the broker to be able to identify those clients that are capable of accepting the method, through some means of capability negotiation.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
basic.consume(short reserved-1, queue-name queue, consumer-tag consumer-tag,
  no-local no-local, no-ack no-ack, bit exclusive, no-wait no-wait,
  table arguments) → consume-ok
```

Start a queue consumer. *Support:* **partial**

This method asks the server to start a "consumer", which is a transient request for messages from a specific queue. Consumers last as long as the channel they were declared on, or until the client cancels them.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
basic.deliver(consumer-tag consumer-tag, delivery-tag delivery-tag,
  redelivered redelivered, exchange-name exchange, shortstr routing-key)
```

Notify the client of a consumer message. *Support:* **full**

This method delivers a message to the client, via a consumer. In the asynchronous message delivery model, the client starts a consumer using the Consume method, then the server responds with Deliver methods as and when messages arrive for that consumer.

[amqpdoc] **(back to top)**

```
basic.get(short reserved-1, queue-name queue, no-ack no-ack) → get-ok | get-empty
```

Direct access to a queue. *Support:* **full**

This method provides a direct access to the messages in a queue using a synchronous dialogue that is designed for specific types of application where synchronous functionality is more important than performance.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
basic.nack(delivery-tag delivery-tag, bit multiple, bit requeue)
```

THIS METHOD IS A RABBITMQ-SPECIFIC EXTENSION OF AMQP

Reject one or more incoming messages.

This method allows a client to reject one or more incoming messages. It can be used to interrupt and cancel large incoming messages, or return untreatable messages to their original queue. This method is also used by the server to inform publishers on channels in confirm mode of unhandled messages. If a publisher receives this method, it probably needs to republish the offending messages.

RabbitMQ Documentation

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

`basic.`**`publish`**`(`**`short`** *`reserved-1`*`, `**`exchange-name`** *`exchange`*`, `**`shortstr`** *`routing-key`*`,`
`bit` *`mandatory`*`, `**`bit`** *`immediate`*`)`

Publish a message. *Support:* **full**

This method publishes a message to a specific exchange. The message will be routed to queues as defined by the exchange configuration and distributed to any active consumers when the transaction, if any, is committed.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

`basic.`**`qos`**`(`**`long`** *`prefetch-size`*`, `**`short`** *`prefetch-count`*`, `**`bit`** *`global`*`)` ➔ **`qos-ok`**

Specify quality of service. *Support:* **partial**

This method requests a specific quality of service. The QoS can be specified for the current channel or for all channels on the connection. The particular properties and semantics of a qos method always depend on the content class semantics. Though the qos method could in principle apply to both peers, it is currently meaningful only for the server.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

`basic.`**`recover`**`(`**`bit`** *`requeue`*`)`

Redeliver unacknowledged messages. *Support:* **partial**

This method asks the server to redeliver all unacknowledged messages on a specified channel. Zero or more messages may be redelivered. This method replaces the asynchronous Recover.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

`basic.`**`recover-async`**`(`**`bit`** *`requeue`*`)`

Redeliver unacknowledged messages.

This method asks the server to redeliver all unacknowledged messages on a specified channel. Zero or more messages may be redelivered. This method is deprecated in favour of the synchronous Recover/Recover-Ok.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

`basic.`**`reject`**`(`**`delivery-tag`** *`delivery-tag`*`, `**`bit`** *`requeue`*`)`

Reject an incoming message. *Support:* **partial**

This method allows a client to reject a message. It can be used to interrupt and cancel large incoming messages, or return untreatable messages to their original queue.

RabbitMQ blog post

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

`basic.`**`return`**`(`**`reply-code`** *`reply-code`*`, `**`reply-text`** *`reply-text`*`, `**`exchange-name`** *`exchange`*`,`
`shortstr` *`routing-key`*`)`

Return a failed message. *Support:* **full**

This method returns an undeliverable message that was published with the "immediate" flag set, or an unroutable message published with the "mandatory" flag set. The reply code and text provide information about the reason that the message was undeliverable.

[amqpdoc] **(back to top)**

Channel

`channel.`**`close`**`(`**`reply-code`** *`reply-code`*`, `**`reply-text`** *`reply-text`*`, `**`class-id`** *`class-id`*`,`
`method-id` *`method-id`*`)` ➔ **`close-ok`**

Request a channel close. *Support:* **full**

This method indicates that the sender wants to close the channel. This may be due to internal conditions (e.g. a forced shut-down) or due to an error handling a specific method, i.e. an exception. When a close is due to an exception, the sender provides the class and method id of the method which caused the exception.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
channel.flow(bit active) ➔ flow-ok
```

Enable/disable flow from peer. *Support:* **partial**

This method asks the peer to pause or restart the flow of content data sent by a consumer. This is a simple flow-control mechanism that a peer can use to avoid overflowing its queues or otherwise finding itself receiving more messages than it can process. Note that this method is not intended for window control. It does not affect contents returned by Basic.Get-Ok methods.

[amqpdoc] **(back to top)**

```
channel.open(shortstr reserved-1) ➔ open-ok
```

Open a channel for use. *Support:* **full**

This method opens a channel to the server.

[amqpdoc] **(back to top)**

Confirm

THIS CLASS IS A RABBITMQ-SPECIFIC EXTENSION OF AMQP

```
confirm.select(bit nowait) ➔ select-ok
```

.

This method sets the channel to use publisher acknowledgements. The client can only use this method on a non-transactional channel.

RabbitMQ Documentation

[javadoc] [dotnetdoc] [amqpdoc]

(back to top)

Exchange

```
exchange.bind(short reserved-1, exchange-name destination, exchange-name source,
  shortstr routing-key, no-wait no-wait, table arguments) ➔ bind-ok
```

THIS METHOD IS A RABBITMQ-SPECIFIC EXTENSION OF AMQP

Bind exchange to an exchange.

This method binds an exchange to an exchange.

RabbitMQ Documentation

RabbitMQ blog post

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
exchange.declare(short reserved-1, exchange-name exchange, shortstr type, bit passive,
  bit durable, bit auto-delete*, bit internal*, no-wait no-wait,
  table arguments) ➔ declare-ok
```

** RABBITMQ-SPECIFIC EXTENSION OF AMQP*

Verify exchange exists, create if needed. *Support:* **full**

This method creates an exchange if it does not already exist, and if the exchange exists, verifies that it is of the correct and expected class.

RabbitMQ implements an extension to the AMQP specification that allows for unroutable messages to be delivered to an *Alternate Exchange* (AE). The AE feature helps to detect when clients are publishing messages that cannot be routed and can provide "or else" routing semantics where some messages are handled specifically and the remainder are processed by a generic handler.

AE documention

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
exchange.delete(short reserved-1, exchange-name exchange, bit if-unused,
  no-wait no-wait) ➔ delete-ok
```

Delete an exchange. *Support:* **partial**

This method deletes an exchange. When an exchange is deleted all queue bindings on the exchange are cancelled.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
exchange.unbind(short reserved-1, exchange-name destination, exchange-name source,
  shortstr routing-key, no-wait no-wait, table arguments) ➔ unbind-ok
```

THIS METHOD IS A RABBITMQ-SPECIFIC EXTENSION OF AMQP

Unbind an exchange from an exchange.

This method unbinds an exchange from an exchange.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

Queue

```
queue.bind(short reserved-1, queue-name queue, exchange-name exchange,
  shortstr routing-key, no-wait no-wait, table arguments) ➔ bind-ok
```

Bind queue to an exchange. *Support:* **full**

This method binds a queue to an exchange. Until a queue is bound it will not receive any messages. In a classic messaging model, store-and-forward queues are bound to a direct exchange and subscription queues are bound to a topic exchange.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
queue.declare(short reserved-1, queue-name queue, bit passive, bit durable,
  bit exclusive, bit auto-delete, no-wait no-wait, table arguments) ➔ declare-ok
```

Declare queue, create if needed. *Support:* **full**

This method creates or checks a queue. When creating a new queue the client can specify various properties that control the durability of the queue and its contents, and the level of sharing for the queue.

RabbitMQ implements extensions to the AMQP specification that permits the creator of a queue to control various aspects of its behaviour.

Per-Queue Message TTL

This extension determines for how long a message published to a queue can live before it is discarded by the server. The time-to-live is configured with the *x-message-ttl* argument to the arguments parameter of this method.

Queue Expiry

Queues can be declared with an optional lease time. The lease time determines how long a queue can remain unused before it is automatically deleted by the server. The lease time is provided as an *x-expires* argument in the arguments parameter to this method.

x-message-ttl documentation

x-expires documentation

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
queue.delete(short reserved-1, queue-name queue, bit if-unused, bit if-empty,
  no-wait no-wait) ➔ delete-ok
```

Delete a queue. *Support:* **partial**

This method deletes a queue. When a queue is deleted any pending messages are sent to a dead-letter queue if this is defined in the server configuration, and all consumers on the queue are cancelled.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
queue.purge(short reserved-1, queue-name queue, no-wait no-wait) ➔ purge-ok
```

Purge a queue. *Support:* **full**

This method removes all messages from a queue which are not awaiting acknowledgment.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
queue.unbind(short reserved-1, queue-name queue, exchange-name exchange,
  shortstr routing-key, table arguments) → unbind-ok
```

Unbind a queue from an exchange. *Support:* **partial**

This method unbinds a queue from an exchange.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

Tx

```
tx.commit() → commit-ok
```

Commit the current transaction. *Support:* **full**

This method commits all message publications and acknowledgments performed in the current transaction. A new transaction starts immediately after a commit.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
tx.rollback() → rollback-ok
```

Abandon the current transaction. *Support:* **full**

This method abandons all message publications and acknowledgments performed in the current transaction. A new transaction starts immediately after a rollback. Note that unacked messages will not be automatically redelivered by rollback; if that is required an explicit recover call should be issued.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**

```
tx.select() → select-ok
```

Select standard transaction mode. *Support:* **full**

This method sets the channel to use standard transactions. The client must use this method at least once on a channel before using the Commit or Rollback methods.

[javadoc] [dotnetdoc] [amqpdoc] **(back to top)**