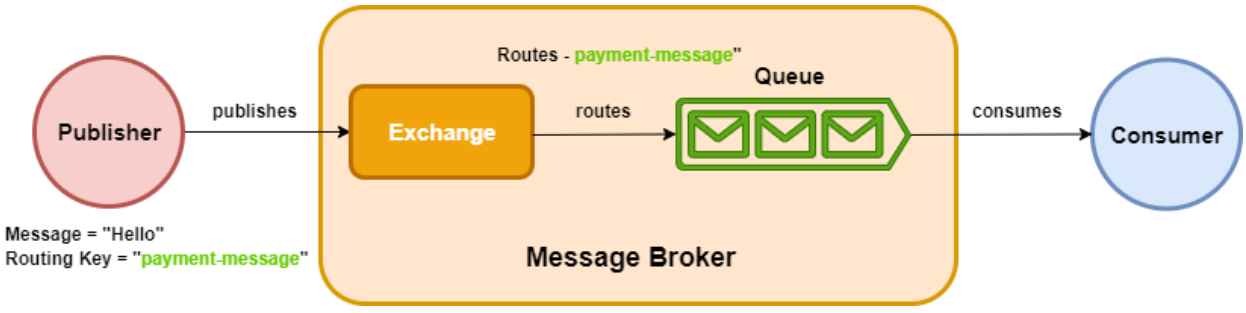
Chapter 2: Exchanges, Queues, Bindings and Consumers

In this chapter we will take a look at what exchanges, queues, bindings and consumers are and what are their roles and responsabilities in the RabbitMQ broker.

Exchanges

Exchanges are simply put routers in a RabbitMQ broker. They are used to rout messages to the specified (bounded) queues. They route the messages based on a specific routing key. A queue cannot be created without binding it to a certain exchange. Even the simplest queue creation (where you do not declaratively specify an exchange), the queue must be bound to an exchange that has routing key as the name of the queue itself.



An Exchange is defined by:

Key	Definition
Name	The name of the exchange
Durability	Persisting the messages to disk

Key	Definition
Auto-Delete	Delete message when not needed (when all queues are finished using it)
Arguments	These are message broker-dependent

There are four types of exchanges:

1. Direct Exchanges
2. Fanout Exchanges
3. Topic Exchanges
4. Header Exchanges

We will look at each one separately with code examples in the next chapter.

Queues

Queues are storage for messages and work on a first in first out (FIFO) principle.

They are defined by:

Key	Definition
Name	The name of the queue

Key	Definition
Durable	Persisting the queue disk so it can survive broker or server restart. **This only means that the queue is durable and not the messages inside the queue and that the queue will be redeclared if the server or broker restarts. If you want to make the messages durable, you need to make the messages persistent (passing arguments)???**
Exclusive	Delete queue when not needed. When a queue is used by one connection and it will be removed when that connection closes.
Auto Delete	Queue delete when consumer unsubscribes. Good for temporary queues.

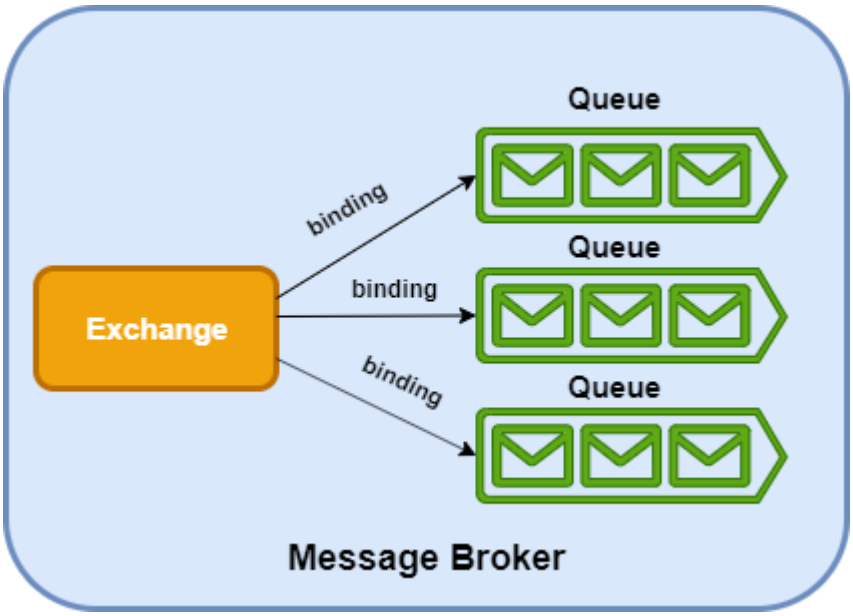
The name of the queue can be maximum 255 characters long. Their name can be declared from the developers in their client application or can be given a random name by the RabbitMQ broker.

In the client application you need to declare the queue and exchange (optional). Declaration of the queue or exchange is idempotent, meaning that if the queue/exchange already exist in the RabbitMQ broker, it is not overwritten. For example if the publisher already created the queue, consumer will just attach it self to that queue and not overwrite the queue declaration.

Bindings

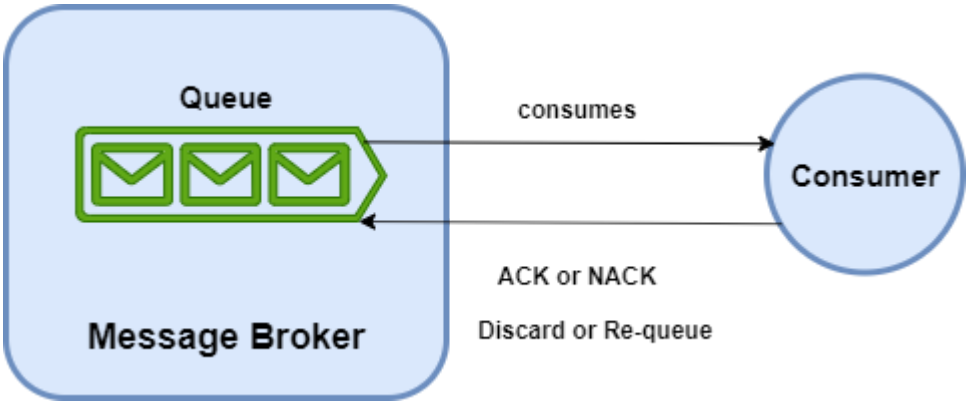
As we said earlier, every queue needs to be bound to a specific exchange. There can be multiple queues bound to one exchange. The

exchange can route messages only to the queues that are bound to it (using the routing key).



Consumer

Consumer is an application that consumes (reads) the messages from the queue(s). The developer is responsible to declare the exchange and the queue that it wants to listen (consumes) messages from. When the consumer reads the message, the message needs to be acknowledge so it can be removed from the queue. Acknowledgement (ACK) is a way for consumer to tell the RabbitMQ broker that the message has been read successfully and that it can be removed from the queue. The ACK can be performed automatically once the consumer gets the message from the queue (this is configured in the declaration of the queue), or the consumer can do it manually by invoking ACK from the client library. In addition to the ACK, there is NACK (NOT Acknowledge) to tell the message broker that there was some problem with the message. With the NACK you can specify if you want the message to be discarded or re-queued.



There can be multiple consumers reading messages from one queue. This is called **worker queue** and there will be code example later in this book for this kind of consumer pattern.