A message queue has various properties: private or shared, durable or temporary, client-named or server-named, etc. By selecting the desired properties we can use a message queue to implement conventional middleware entities such as:

- A shared **store-and-forward queue**, which holds messages and distributes these between consumers on a round-robin basis. Store and forward queues are typically durable and shared between multiple consumers.
- A **private reply queue**, which holds messages and forwards these to a single consumer. Reply queues are typically temporary, server-named, and private to one consumer.
- A **private subscription queue**, which holds messages collected from various "subscribed" sources, and forwards these to a single consumer.

Subscription queues are typically temporary, server-named, and private to one consumer. These categories are not formally defined in AMQP: they are examples of how message queues can be used. It is trivial to create new entities such as durable, shared subscription queues.

2.1.1.2 The Exchange

An exchange accepts messages from a producer application and routes these to message queues according to pre-arranged criteria. These criteria are called "bindings". Exchanges are matching and routing engines. That is, they inspect messages and using their binding tables, decide how to forward these messages to message queues or other exchanges. Exchanges never store messages.

The term "exchange" is used to mean both a class of algorithm, and the instances of such an algorithm. More properly, we speak of the "exchange type" and the "exchange instance".

AMQP defines a number of standard exchange types, which cover the fundamental types of routing needed to do common message delivery. AMQP servers will provide default instances of these exchanges. Applications that use AMQP can additionally create their own exchange instances. Exchange types are named so that applications which create their own exchanges can tell the server what exchange type to use. Exchange instances are also named so that applications can specify how to bind queues and publish messages.

Exchanges can do more than route messages. They can act as intelligent agents that work from within the server, accepting messages and producing messages as needed. The exchange concept is intended to define a model for adding extensibility to AMQP servers in a reasonably standard way, since extensibility has some impact on interoperability.

2.1.1.3 The Routing Key

In the general case an exchange examines a message's properties, its header fields, and its body content, and using this and possibly data from other sources, decides how to route the message.

In the majority of simple cases the exchange examines a single key field, which we call the "routing key". The routing key is a virtual address that the exchange may use to decide how to route the message.

For point-to-point routing, the routing key is usually the name of a message queue.

For topic pub-sub routing, the routing key is usually the topic hierarchy value.

In more complex cases the routing key may be combined with routing on message header fields and/or its content.

2.1.1.4 Analogy to Email

If we make an analogy with an email system we see that the AMQP concepts are not radical:

- an AMQP message is analogous to an email message;
- a message queue is like a mailbox;
- a consumer is like a mail client that fetches and deletes email;