Server Side:
AMQ model

- The **"exchange"** receives messages from publisher applications and routes these to "message queues", based on arbitrary criteria, usually message properties or content.
- The "**message queue**" stores messages until they can be safely processed by a consuming client application (or multiple applications).
- The "**binding**" defines the relationship between a message queue and an exchange and provides the message routing criteria.

classic message-oriented middleware concepts of store-and-forward queues and topic subscriptions

AMQP command architecture: An encoded wire-level protocol command which executes actions on the state of the AMQ model Architecture.

AMQ model architecture: A logical framework representing the key entities and semantics which
must be made available by an AMQP compliant server implementation, such that the server state can be manipulated by a client in order to achieve the semantics defined in this specification.

Connection: A network connection, e.g. a TCP/IP socket connection.

Channel: A bi-directional stream of communications between two AMQP peers. Channels are multiplexed so that a single network connection can carry multiple channels.

Client: The initiator of an AMQP connection or channel. AMQP is not symmetrical. Clients produce and consume messages while servers queue and route messages.
Server: The process that accepts client connections and implements the AMQP message queueing and routing functions. Also known as "broker".
Peer: Either party in an AMQP connection. An AMQP connection involves exactly two peers (one is the client, one is the server).
Frame: A formally-defined package of connection data. Frames are always written and read contiguously - as a single unit - on the connection.
Protocol class: A collection of AMQP commands (also known as Methods) that deal with a specific type of functionality.
Method: A specific type of AMQP command frame that passes instructions from one peer to the other.
Content: Application data passed from client to server and from server to client. The term is synonymous with "message".
Content header: A specific type of frame that describes a content's properties.

Content body: A specific type of frame that contains raw application data. Content body frames are entirely opaque - the server does not examine or modify these in any way.

Message: Synonymous with "content".
Exchange: The entity within the server which receives messages from producer applications and optionally routes these to message queues within the server.
Exchange type: The algorithm and implementation of a particular model of exchange. In contrast to the "exchange instance", which is the entity that receives and routes messages within the server.
Message queue: A named entity that holds messages and forwards them to consumer applications.
Binding: An entity that creates a relationship between a message queue and an exchange.
Routing key: A virtual address that an exchange may use to decide how to route a specific message.
Durable: A server resource that survives a server restart.
Transient: A server resource or message that is wiped or reset after a server restart.
Persistent: A message that the server holds on reliable disk storage and MUST NOT lose after a server restart.
Consumer: A client application that requests messages from a message queue.
Producer: A client application that publishes messages to an exchange.
Virtual host: A collection of exchanges, message queues and associated objects. Virtual hosts are independent server domains that share a common authentication and encryption environment.
Assertion: A condition that must be true for processing to continue.
Exception: A failed assertion, handled by closing either the Channel or the Connection

Types of Exchange
The direct exchange type, which routes on a routing key. The default exchange is a direct exchange.
the topic exchange type, which routes on a routing pattern.

Message Queue Properties
name - if left unspecified, the server chooses a name and provides this to the client. Generally, when applications share a message queue they agree on a message queue name beforehand, and when an application needs a message queue for its own purposes, it lets the server provide a name.
exclusive - if set, the queue belongs to the current connection only, and is deleted when the connection closes.
durable - if set, the message queue remains present and active when the server restarts. It may lose transient messages if the server restarts.

The Connection Class

The client opens a TCP/IP connection to the server and sends a protocol header. This is the only data the client sends that is not formatted as a method.
The server responds with its protocol version and other properties, including a list of the security mechanisms that it supports (the Start method).
The client selects a security mechanism (Start-Ok).
The server starts the authentication process, which uses the SASL challenge-response model. It sends the client a challenge (Secure).
The client sends an authentication response (Secure-Ok). For example using the "plain" mechanism,the response consist of a login name and password.
The server repeats the challenge (Secure) or moves to negotiation, sending a set of parameters such as
maximum frame size (Tune).
The client accepts or lowers these parameters (Tune-Ok).
The client formally opens the connection and selects a virtual host (Open).
The server confirms that the virtual host is a valid choice (Open-Ok).
The client now uses the connection as desired.
One peer (client or server) ends the connection (Close).
The other peer hand-shakes the connection end (Close-Ok).
The server and the client close their socket connection.

The Channel Class
The client opens a new channel (Open).
The server confirms that the new channel is ready (Open-Ok).
The client and server use the channel as desired.
One peer (client or server) closes the channel (Close).
The other peer hand-shakes the channel close (Close-Ok).

The Exchange Class
1. The client asks the server to make sure the exchange exists (Declare). The client can refine this into,
"create the exchange if it does not exist", or "warn me but do not create it, if it does not exist".
2. The client publishes messages to the exchange.
3. The client may choose to delete the exchange (Delete).

The Queue Class
The life-cycle for a durable message queue is fairly simple:
1. The client asserts that the message queue exists (Declare, with the "passive" argument).
2. The server confirms that the message queue exists (Declare-Ok).
3. The client reads messages off the message queue.

The life-cycle for a subscription involves an extra bind stage:
1. The client creates the message queue (Declare), and the server confirms (Declare-Ok).

2. The client binds the message queue to a topic exchange (Bind) and the server confirms (Bind-Ok).
3. The client uses the message queue as in the previous examples.

The Basic Class
Sending messages from client to server, which happens asynchronously (Publish)
Starting and stopping consumers (Consume, Cancel)
Sending messages from server to client, which happens asynchronously (Deliver, Return)
Acknowledging messages (Ack, Reject)
Taking messages off the message queue synchronously (Get).

The Transaction Class

AMQP is a binary protocol. Information is organised into "frames", of various types. Frames carry
protocol methods and other information. All frames have the same general format: frame header, payload,
and frame end. The frame payload format depends on the frame type.
We assume a reliable stream-oriented network transport layer (TCP/IP or equivalent).
Within a single socket connection, there can be multiple independent threads of control, called "channels".
Each frame is numbered with a channel number. By interleaving their frames, different channels share the
connection. For any given channel, frames run in a strict sequence that can be used to drive a protocol
parser (typically a state machine).

Frame Details
All frames consist of a header (7 octets), a payload of arbitrary size, and a 'frame-end' octet that detects
malformed frames
Check PDF
Type
Channel
Size
Payload
Frame-end

Method Frames

Class-id
Method-id
Arguments

Content Frames

Content is the application data we carry from client-to-client via the AMQP server. Content is, roughly
speaking, a set of properties plus a binary data part. The set of allowed properties are defined by the Basic
class, and these form the "content header frame". The data can be any size, and MAY be broken into
several (or many) chunks, each forming a "content body frame".

Heartbeat Frames

AMQP Client Architecture

The basic unit of data in AMQP is a *frame*. There are nine AMQP frame bodies defined that are used to initiate, control and tear down the transfer of messages between two peers. These are:

- open (the *connection*)
- begin (the *session*)
- attach (the *link*)
- transfer
- flow
- disposition
- detach (the *link*)
- end (the *session*)
- close (the *connection*)

The *link protocol* is at the heart of AMQP.