

# R-Pi

Team Emertxe



# IoT: Protocols

## AMQP

# AMQP

## Introduction

- It's an open standard designed to allow development of applications which are tailored to work as middleware in brokering of messages between different processes, applications, or even unrelated systems that need to talk to each other and pass on messages.

- The Advanced Message Queuing Protocol (AMQP) creates interoperability between clients and brokers (i.e. messaging middleware).
- Its goal of creation was to enable a wide range of different applications and systems to be able to work together, regardless of their internal designs, standardizing enterprise messaging on industrial scale.
- AMQP includes the definitions for both the way networking takes place and the way message broker applications work. This means the specifications for:
  - Operations of routing and storage of messages with the message brokers and set of rules to define how components involved work
  - And a wire protocol to implement how communications between the clients and the brokers performing the above operations work



- Before AMQP, there used to be different message brokering and transferring applications created and set in use by different vendors.
- However, they had one big problem and it was their lack of interoperability.
- There was simply not a way for one to work with another.
- The only method that could be used to get different systems using different protocols to work was by introducing an additional layer for converting messages called messaging bridge.
- These systems, using individual adapters to be able to receive messages like regular clients, would be used to connect multiple and different messaging systems (e.g. WebSphere MQ and another).

•

•

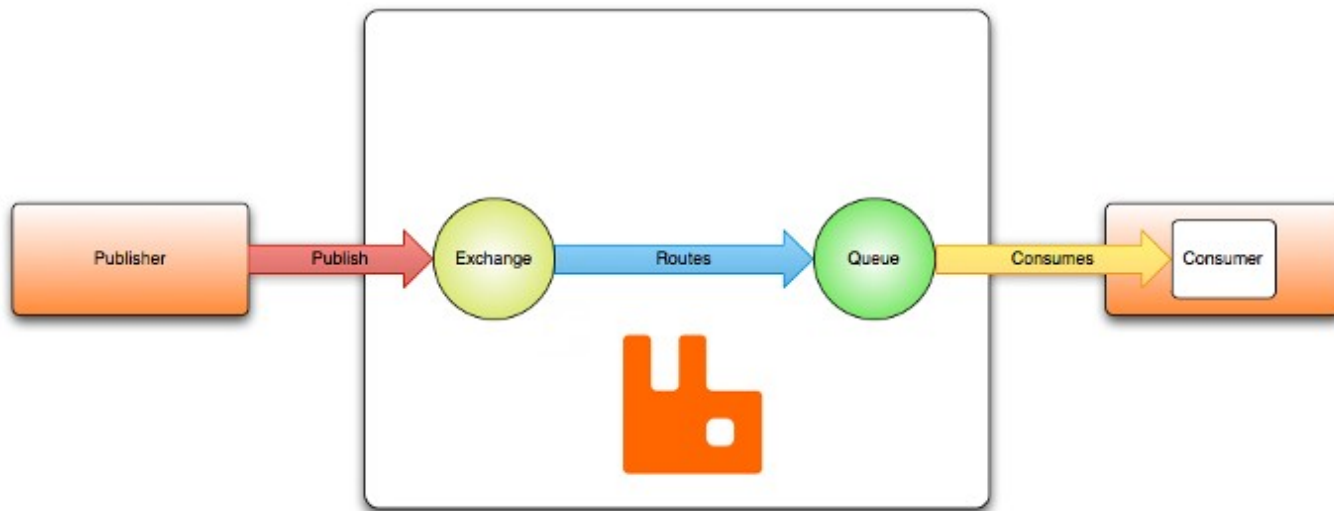
•

# AMQP

## Reasons: Creation and Use

- AMQP, by offering the clearly defined rules and instructions as we explained above, creates a common ground which can be used for all message queuing and brokering applications to work and interoperate.

## "Hello, world" example routing





- The AMQP has the following view of the world: messages are published to exchanges, which are often compared to post offices or mailboxes.
- Exchanges then distribute message copies to queues using rules called bindings.
- Then the broker either deliver messages to consumers subscribed to queues, or consumers fetch/pull messages from queues on demand.
- When publishing a message, publishers may specify various message attributes (message meta-data).
- Some of this meta-data may be used by the broker, however, the rest of it is completely opaque to the broker and is only used by applications that receive the message.





- Networks are unreliable and applications may fail to process messages therefore the AMQP model has a notion of message acknowledgements:
- when a message is delivered to a consumer the consumer notifies the broker, either automatically or as soon as the application developer chooses to do so.
- When message acknowledgements are in use, a broker will only completely remove a message from a queue when it receives a notification for that message (or group of messages).
- In certain situations, for example, when a message cannot be routed, messages may be returned to publishers, dropped, or, if the broker implements an extension, placed into a so-called "dead letter queue".
- Queues, exchanges and bindings are collectively referred to as **AMQP entities**.

# AMQP

## Use Cases

- Whenever there is a need for high-quality and safe delivery of messages between applications and processes, AMQP implementing message brokering solutions can be considered for use.
- AMQP ensures
  - Reliability of message deliveries
  - Rapid and ensured delivery of messages
  - Message acknowledgements

- capabilities make it ideal for
  - Monitoring and globally sharing updates
  - Connecting different systems to talks to each other
  - Allowing servers to respond to immediate requests quickly and delegate time consuming tasks for later processing
  - Distributing a message to multiple recipients for consumption
  - Enabling offline clients to fetch data at a later time
  - Introducing fully asynchronous functionality for systems
  - Increasing reliability

# AMQP

## Assembly and Terminology



- Understanding and working with AMQP involves being familiar with quite a few different terms and terminology.
- In this section, we will go over these key parts:
  - **Broker (Server):** An application - implementing the AMQP model - that accepts connections from clients for message routing, queuing etc.
  - **Message:** Content of data transferred / routed including information such as payload and message attributes.
  - **Consumer:** An application which receives message(s) - put by a producer - from queues.
  - **Producer:** An application which put messages to a queue via an exchange.

# AMQP

## Main Components

- The AMQP Model defining how messages are received, routed, stored, queued and how application parts handling these tasks work rely on the clear set definitions of the below components:
  - Exchange: A part of the broker (i.e. server) which receives messages and routes them to queues
  - Queue (message queue): A named entity which messages are associated with and from where consumers receive them
  - Bindings: Rules for distributing messages from exchanges to queues

- "message brokers" translate to applications which receive the actual messages and route (i.e. transfer) them to relevant parties.

- APPLICATION                  EXCHANGE                  TASK LIST                  WORKER

- [DATA] -----> [DATA] ---> [D]+[D] [D] [D] ---> [DATA]

- Publisher                  EXCHANGE                  Queue                  Consumer

# AMQP

## How does Exchanges Work?

- After receiving messages from publishers (i.e. clients), the exchanges process them and route them to one or more queues.
- The type of routing performed depend on the type of the exchange and there are currently four of them.
  - Direct Exchange
  - Fanout Exchange
  - Topic Exchange
  - Headers Exchange

# AMQP

## Direct Exchange

- Direct exchange type involves the delivery of messages to queues based on routing keys.
- Routing keys can be considered as additional data defined to set where a message will go.



# AMQP

## Fanout Exchange



- Fanout exchange completely ignores the routing key and sends any message to all the queues bound to it.
- Use cases for fanout exchanges usually involve distribution of a message to multiple clients for purposes similar to notifications:
  - ➔ Sharing of messages (e.g. chat servers) and updates (e.g. news)
  - ➔ Application states (e.g. configurations)

# AMQP

## Topic Exchange

- Topic exchange is mainly used for pub/sub (publish-subscribe) patterns.
- Using this type of transferring, a routing key alongside binding of queues to exchanges are used to match and send messages.

# AMQP

## Headers Exchange

- Headers exchange constitutes of using additional headers (i.e. message attributes) coupled with messages instead of depending on routing keys for routing to queues.

THANK YOU