MuntsOS

Application Note #25: RabbitMQ Enterprise Message Broker Client Programs

Revision 0
29 December 2025

by Philip Munts
***dba* Munts Technologies**
http://tech.munts.com

Introduction

This application note describes some examples of and best practices for **MuntsOS Embedded Linux** (hereafter just **MuntsOS**) target programs that send messages to and/or receive messages from a RabbitMQ Enterprise Message Broker (hereafter just **RabbitMQ**), which is often used in backend systems to implement an information bus architecture.

An information bus architecture works well for IoT (Internet of Things) networks, with each IoT end node running software that pushes messages to and/or pulls messages from the information bus.

Hereafter, the term **broker** means an instance of **RabbitMQ** installed, configured, and running on a server computer accessible from a **MuntsOS** target computer. Installation and configuration of **RabbitMQ** are beyond the scope of this document.

Virtual Host

A RabbitMQ Virtual Host (hereafter, **vhost**) is analogous to a web server virtual host. Just as tech.munts.com and repo.munts.com are web servers with separate name spaces that run on the same piece of hardware, each vhost creates a distinct name space at the connection level.

Every broker has an anonymous default vhost sometimes referred to as "/".

Named vhosts can be managed (created and destroyed) using the `rabbitmqctl` command.

Each vhost its own URI of the form:

```
scheme://user:password@host[:port][/[vhost]]
```

where:

`scheme` can be either `amqp` (unencrypted) or `amqps` (encrypted).

`user` and `password` have default values `guest` and `guest`.

`host` is an IP address or domain name

`port` is a TCP port number. If omitted, one of the default port numbers is selected: 5672 (unencrypted) or 5671 (encrypted).

`vhost` is the name of a vhost. If omitted, with or without the trailing `/`, the anonymous default vhost is selected.

A **RabbitMQ** client program may connect to more than one vhost, each with a distinct URI and a separate connection object.

Exchange

A RabbitMQ Exchange (hereafter, **exchange**) is a named entity inside a vhost name space to which a **RabbitMQ** client program can write/publish/push/send messages, each of which includes a routing key string. The exchange then forwards each message to one or more output queues. There are many exchange types, each with different forwarding policies. The most common and useful exchange types are:

A **fanout** exchange writes every incoming message to every output queue bound to the exchange. The routing key is ignored.

A **direct** exchange writes incoming messages to every output queue bound to the exchange with a matching (exactly) routing key.

A **topic** exchange also writes incoming message to every queue bound to the exchange with a matching routing key. However, each output queue's routing key string may be a rudimentary regular expression consisting of tokens separated by periods, *e.g.* `House.Bedroom.Table`. Two special wildcard tokens are defined: `*` matches exactly one token and `#` matches zero or more tokens.

So, a topic exchange will write an incoming message with the routing key `House.Bedroom.Table` to output queues created with routing keys `House.Bedroom.Table`, `House.Bedroom.*`, `House.*.Table`, `House.*.*`, `*.Bedroom.*` , `*.*.Table` and `House.#`.

Every vhost will contain default exchanges named `amq.fanout`, `amq.direct`, and `amq.topic` among others.

Queue

Example Client *aka* End Node Programs

The Linux Simple I/O Library contains the following Ada and C# example **RabbitMQ** client *aka* end node programs, which can be cross-compiled to run on **MuntsOS** target computers:

- test_rabbitmq_consume.adb
- test_rabbitmq_produce.adb
- test_rabbitmq_consume
- test_rabbitmq_produce

Each of these example programs obtain their runtime configuration from the following environment variables, most of which have default values:

- **`RABBITMQ_SCHEME`** (default **`amqp`**)
- **`RABBITMQ_USER`** (default **`guest`**)
- **`RABBITMQ_PASS`** (default **`guest`**)
- **`RABBITMQ_SERVER`** (default **`localhost`**)
- **`RABBITMQ_PORT`** (default **`5672`**)
- **`RABBITMQ_VHOST`** (default **`/`**)
- **`RABBITMQ_EXCHANGE`** (default **`amq.topic`**)
- **`RABBITMQ_QUEUE`**
- **`RABBITMQ_ROUTING`** (default empty string "")

The minimum set of environment variables that you will need to define in the file **`/etc/environment`** on your **MuntsOS** target computer is:

- **`RABBITMQ_USER`**
- **`RABBITMQ_PASS`**
- **`RABBITMQ_SERVER`**
- **`RABBITMQ_VHOST`**

Additionally, you may need to define **`RABBITMQ_ROUTING`** to set the **RabbitMQ** routing key (*e.g.* topic) if the client program does not generate a custom routing key on the fly.

The LoRa radio network example producer program wioe5_ham1_rabbitmq generates a custom routing key for each incoming radio message that it passes to the broker, enabling consumer programs to select messages to and/or from particular radio nodes.