RabbitMQ for Windows: Exchange Types

28 March, 2012. It was a Wednesday.

Posts In This Series

- RabbitMQ for Windows: Introduction
- RabbitMQ for Windows: Building Your First Application
- RabbitMQ for Windows: Hello World Review
- RabbitMQ for Windows: Exchange Types
- RabbitMQ for Windows: Direct Exchanges
- RabbitMQ for Windows: Fanout Exchanges
- RabbitMQ for Windows: Topic Exchanges
- RabbitMQ for Windows: Headers Exchanges

This is the fourth installment to the series: RabbitMQ for Windows. In the last installment, we reviewed our Hello World example and introduced the concept of Exchanges. In this installment, we'll discuss the four basic types of RabbitMQ exchanges.

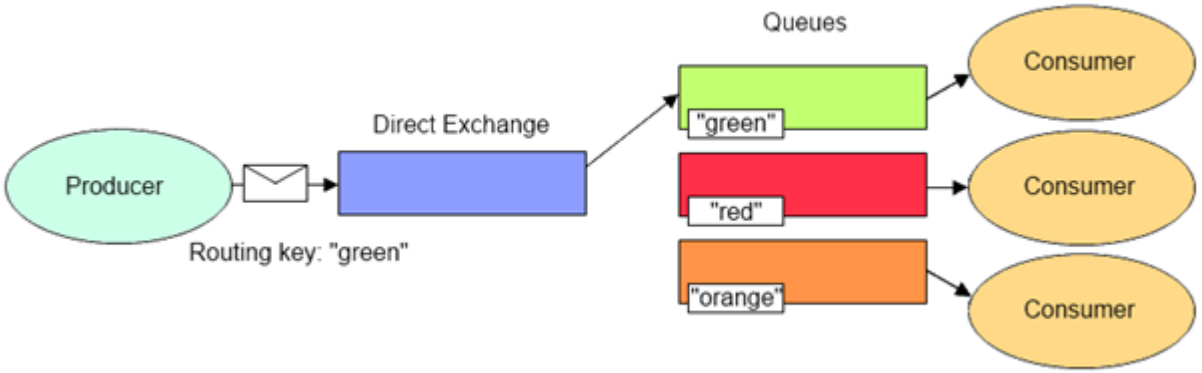
Exchange Types

Exchanges control the routing of messages to queues. Each exchange type defines a specific routing algorithm which the server uses to determine which bound queues a published message should be routed to.

RabbitMQ provides four types of exchanges: Direct, Fanout, Topic, and Headers.

Direct Exchanges

The Direct exchange type routes messages with a routing key equal to the routing key declared by the binding queue. The following illustrates how the direct exchange type works:

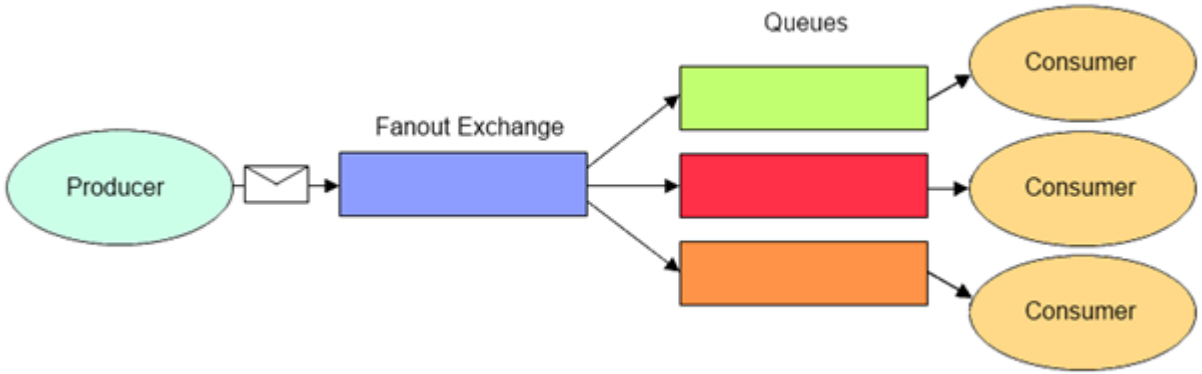


The Direct exchange type is useful when you would like to distinguish messages published to the same exchange using a simple string identifier. This is the type of exchange that was used in our Hello World example. As discussed in part 3 of our series, every queue is automatically bound to a default exchange using a routing key equal to the queue name. This default exchange is declared as a Direct exchange. In our example, the queue named "hello-world-queue" was bound to

the default exchange with a routing key of “hello-world-queue”, so publishing a message to the default exchange (identified with an empty string) routed the message to the queue named “hello-world-queue”.

Fanout Exchanges

The Fanout exchange type routes messages to all bound queues indiscriminately. If a routing key is provided, it will simply be ignored. The following illustrates how the fanout exchange type works:

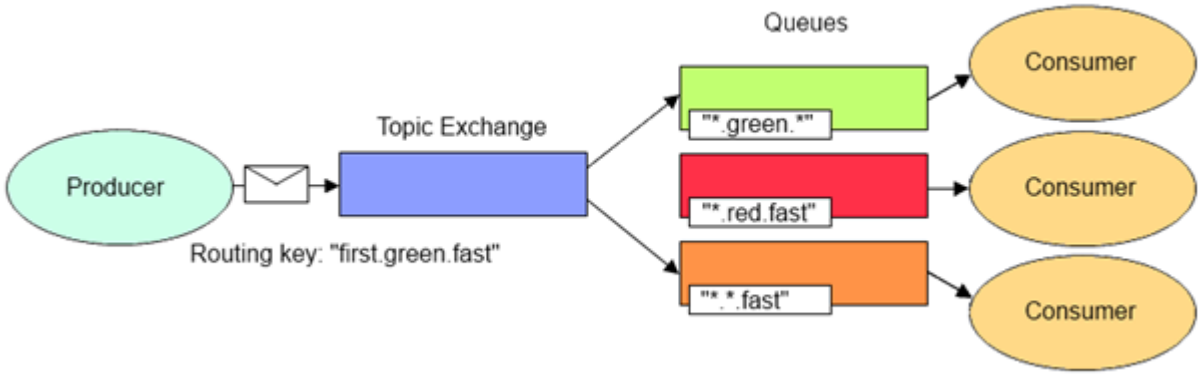


The Fanout exchange type is useful for facilitating the publish-subscribe pattern. When using the fanout exchange type, different queues can be declared to handle messages in different ways. For instance, a message indicating a customer order has been placed might be received by one queue whose consumers fulfill the order, another whose consumers update a read-only history of orders, and yet another whose consumers record the order for reporting purposes.

Topic Exchanges

The Topic exchange type routes messages to queues whose routing key matches all, or a portion of a routing key. With topic exchanges, messages are published with routing keys containing a series of words separated by a dot (e.g. “word1.word2.word3”). Queues binding to a topic exchange supply a matching pattern for the server to use when routing

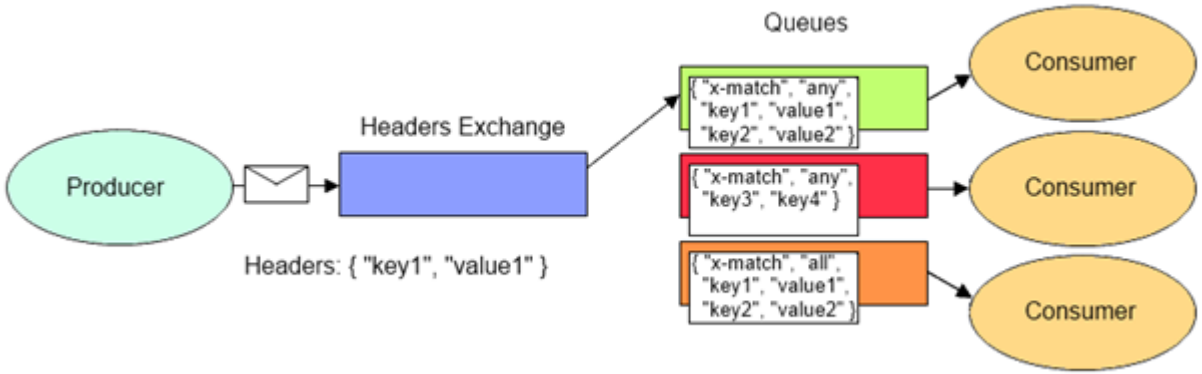
the message. Patterns may contain an asterisk (“*”) to match a word in a specific position of the routing key, or a hash (“#”) to match zero or more words. For example, a message published with a routing key of “honda.civic.navy” would match queues bound with “honda.civic.navy”, “*.civic.*”, “honda.#”, or “#”, but would not match “honda.accord.navy”, “honda.accord.silver”, “*.accord.*”, or “ford.#”. The following illustrates how the fanout exchange type works:



The Topic exchange type is useful for directing messages based on multiple categories (e.g. product type and shipping preference), or for routing messages originating from multiple sources (e.g. logs containing an application name and severity level).

Headers Exchanges

The Headers exchange type routes messages based upon a matching of message headers to the expected headers specified by the binding queue. The headers exchange type is similar to the topic exchange type in that more than one criteria can be specified as a filter, but the headers exchange differs in that its criteria is expressed in the message headers as opposed to the routing key, may occur in any order, and may be specified as matching any or all of the specified headers. The following illustrates how the headers exchange type works:



The Headers exchange type is useful for directing messages which may contain a subset of known criteria where the order is not established and provides a more convenient way of matching based upon the use of complex types as the matching criteria (i.e. a serialized object).

Conclusion

That wraps up our introduction to each of the exchange types. Next time, we'll walk through an example which demonstrates declaring a direct exchange explicitly and take a look at the the push API.

← RabbitMQ for Windows: Hello World Review

RabbitMQ for Windows: Direct Exchanges →

DavidS • 8 years ago

There a bit of a typo within the "Topic Exchange" section. You've got:

The following illustrates how the fanout exchange type works:

when it should be

The following illustrates how the topic exchange type works:

Recent Author Posts

- Hello, React! - A Beginner's Setup Tutorial
- Exploring TypeScript
- Git on Windows: Whence Cometh Configuration
- Separation of Concerns: Application Builds & Continuous Integration
- Survey of Entity Framework Unit of Work Patterns
- Introducing NUnit.Specifications
- Being Agile
- Expected Objects Custom Comparisons
- RabbitBus: An Example
- Adventures in Debugging: The NHibernate 'don't flush the Session' Error

Recent Site Posts

- Domain-Driven Refactoring: Defactoring and Pushing Behavior Down
- Domain-Driven Refactoring: Extracting Domain Services
- Domain-Driven Refactoring: Long Methods
- Domain-Driven Refactoring: Procedural Beginnings
- Domain-Driven Refactoring: Intro
- Local Development with Azure Service Bus
- Taming the WSL 2 Resource Monster
- Crossing the Generics Divide
- OpenTelemetry 1.0 Extensions Released
- Choosing a ServiceLifetime

Authors

- Andrew Siemer
- Chad Myers
- Chris Missal
- Chris Patterson
- Derek Greer
- Derik Whittaker
- Eric Anderson
- Eric Hexter
- Gabriel Schenker
- Gregory Long
- Hugo Bonacci
- James Gregory
- Jimmy Bogard

- John Teague
- Josh Arnold
- Joshua Flanagan
- Joshua Lockwood
- Keith Dahlby
- Matt Hinze
- Patrick Lioi
- Rod Paddock
- Ryan Rauh
- Ryan Svihla
- Scott Densmore
- Sean Biefeld
- Sean Chambers
- Sharon Cichelli
- Steve Donie