

ADAMS

Advanced **D**ata mining **A**nd **M**achine learning **S**ystem

Module: adams-rabbitmq



Peter Reutemann

January 28, 2020

©2019



THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato



Except where otherwise noted, this work is licensed under
<http://creativecommons.org/licenses/by-sa/4.0/>

Contents

1	Introduction	5
2	Installation	7
2.1	Linux	7
2.2	Windows	7
2.3	TLS Support	7
3	Flow	9
3.1	TLS Support	10
3.2	Large payloads	10
4	Data exchange server	11
	Bibliography	13

Chapter 1

Introduction

RabbitMQ[2] is an open source message broker. It supports:

- multiple messaging protocols
- message queuing
- delivery acknowledgement
- flexible routing to queues
- multiple exchange type

ADAMS makes use of it for publishing and consuming messages, as well as for remote procedure calls (RPC).

Chapter 2

Installation

Installation instructions for all available platforms can be found here:
<https://www.rabbitmq.com/download.html>

2.1 Linux

You can install the RabbitMQ server on Debian systems like this:

```
sudo apt-get install rabbitmq-server
```

For RPM-based distributions, please follow the instructions here:
<https://www.rabbitmq.com/install-rpm.html>

2.2 Windows

For installing the server on Windows, please follow the instructions here:
<https://www.rabbitmq.com/install-windows.html>

2.3 TLS Support

You can find information on how to set up RabbitMQ using TLS on the following URL:
<https://www.rabbitmq.com/ssl.html>

Chapter 3

Flow

The following standalones are available:

- *DataExchangeServerConnection* – defines a data exchange server connection.
- *RabbitMQConnection* – defines a RabbitMQ broker connection.
- *RabbitMQChannelAction* – performs the specified channel action.

The following sources are available:

- *RabbitMQConsume* – outputs the data it consumes from either a queue or an exchange.

The following transformers are available:

- *DataExchangeServerDownload* – downloads data from a data exchange server using the specified token.
- *DataExchangeServerRemove* – removes data from a data exchange using the received token.
- *DataExchangeServerUpload* – uploads a file or byte array to a data exchange server and forwards the received token.
- *RabbitMQRemoteProcedureCall* – for performing remote procedure calls via a RabbitMQ broker.

The following sinks are available:

- *RabbitMQMessageDeliveryAction* – for executing an action using the incoming delivery tag, e.g., acknowledging it.
- *RabbitMQPublish* – publishes the incoming data to a queue or an exchange.

The following control actors are available:

- *RabbitMQRemoteSubProcess* – executes the sub-actors remotely for processing the input, forwards the generated output.
- *RabbitMQRemoteTee* – executes the sub-actors remotely for processing the input, forwards the input.
- *RabbitMQRemoteTrigger* – executes the sub-actors remotely, forwards the input.

The following conversions are available:

- *RabbitMQEnvelopeToMap* – converts RabbitMQ envelope into a Java Map, e.g., to obtain a delivery tag to acknowledge consumption of a message.
- *RabbitMQPropertiesToMap* – converts RabbitMQ basic properties into a Java Map, e.g., to obtain the reply-to queue name.

The following JobRunners are available:

- *RabbitMQJobRunner* – distributes the jobs using the specified broker.

3.1 TLS Support

If you want to take advantage of TLS, i.e., encrypting your connections to the RabbitMQ server, then you need to configure your server for that (see 2.3) and encapsulate your actual connection factory used by *RabbitMQConnection* in one of the following meta-factories:

- *NonValidatingSSLConnectionFactory* – only to be used for testing/development
- *SSLConnectionFactory*

3.2 Large payloads

In theory, you can use RabbitMQ to send any amount of data. However, queues may be constrained in disk size and sending large amounts (e.g., processing large datasets) will also slow down the overall performance of the message broker.

As long as the sending and receiving ends have access to the same shared directory, it is possible to exchange the actual payload via this directory and only send the filename via the message broker. For this scenario, you can use the following converters:

- `adams.core.net.rabbitmq.send.FileBasedConverter`
- `adams.core.net.rabbitmq.receive.FileBasedConverter`

The receiving converter automatically deletes the payload file once read successfully.

In case a shared directory is not an option, see chapter 4 on how to use a *data exchange server* instead.

Chapter 4

Data exchange server

The *Data exchange server* (DEX) is a simple REST webservice plugin that enables one to store data temporarily on a server and access it. The underlying idea is to avoid clogging up message queues on RabbitMQ servers. Rather than sending the data as well in the message, only a token that is associated with the actual data on such an exchange server will get sent through RabbitMQ. The following *RESTProvider* that can be used for this service:

```
adams.flow.rest.dex.DataExchangeServer
```

The *RESTPlugin* itself (used internally by the above provider) is the following:

```
adams.flow.rest.dex.DataExchange
```

You can use a data exchange server for offloading payloads from RabbitMQ messages with the following converters:

- *adams.core.net.rabbitmq.receive.DataExchangeServerBasedConverter*
- *adams.core.net.rabbitmq.send.DataExchangeServerBasedConverter*

The following example flows demonstrate the data exchange functionality:

- *adams-rabbitmq-dex-server.flow* – starts an in-memory data exchange server.
- *adams-rabbitmq-dex-upload.flow* – uploads a file to the data exchange server.
- *adams-rabbitmq-dex-download.flow* – downloads data from the data exchange server using the entered token.
- *adams-rabbitmq-dex-queue-publish.flow* – publishes data through RabbitMQ via a data exchange server.
- *adams-rabbitmq-dex-queue-consume.flow* – consumes data through RabbitMQ via a data exchange server.

Bibliography

- [1] *ADAMS* – Advanced Data mining and Machine learning System
<https://adams.cms.waikato.ac.nz/>
- [2] *RabbitMQ* – and open source message broker.
<http://www.rabbitmq.com/>