Abstract

This guide describes how to install and configure the library, run hands-on examples, and use your client with other AMQ components.
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
CHAPTER 1. OVERVIEW

AMQ Spring Boot Starter is an adapter for creating Spring-based applications that use AMQ messaging. It provides a Spring Boot starter module that enables you to build standalone Spring applications. The starter uses the AMQ JMS client to communicate using the AMQP 1.0 protocol.

AMQ Spring Boot Starter is part of AMQ Clients, a suite of messaging libraries supporting multiple languages and platforms. For an overview of the clients, see AMQ Clients Overview. For information about this release, see AMQ Clients 2.10 Release Notes.

AMQ Spring Boot Starter is based on the AMQP 1.0 JMS Spring Boot project.

1.1. KEY FEATURES

- Quickly build standalone Spring applications with built-in messaging
- Automatic configuration of JMS resources
- Configurable pooling of JMS connections and sessions

1.2. SUPPORTED STANDARDS AND PROTOCOLS

- Version 2.2 of the Spring Boot API
- Version 2.0 of the Java Message Service API
- Version 1.0 of the Advanced Message Queueing Protocol (AMQP)

1.3. SUPPORTED CONFIGURATIONS

Refer to Red Hat AMQ 7 Supported Configurations on the Red Hat Customer Portal for current information regarding AMQ Spring Boot Starter supported configurations.

1.4. DOCUMENT CONVENTIONS

The sudo command
In this document, sudo is used for any command that requires root privileges. Exercise caution when using sudo because any changes can affect the entire system. For more information about sudo, see Using the sudo command.

File paths
In this document, all file paths are valid for Linux, UNIX, and similar operating systems (for example, /home/andrea). On Microsoft Windows, you must use the equivalent Windows paths (for example, C:\Users\andrea).

Variable text
This document contains code blocks with variables that you must replace with values specific to your environment. Variable text is enclosed in arrow braces and styled as italic monospace. For example, in the following command, replace <project-dir> with the value for your environment:

```bash
$ cd <project-dir>
```
CHAPTER 2. INSTALLATION

This chapter guides you through the steps to install AMQ Spring Boot Starter in your environment.

2.1. PREREQUISITES

- You must have a subscription to access AMQ release files and repositories.
- To build programs with AMQ Spring Boot Starter, you must install Apache Maven.
- To use AMQ Spring Boot Starter, you must install Java.

2.2. USING THE RED HAT MAVEN REPOSITORY

Configure your Maven environment to download the client library from the Red Hat Maven repository.

Procedure

1. Add the Red Hat repository to your Maven settings or POM file. For example configuration files, see Section B.1, “Using the online repository”.

```xml
<repository>
  <id>red-hat-ga</id>
  <url>https://maven.repository.redhat.com/ga</url>
</repository>
```

2. Add the library dependency to your POM file.

```xml
<dependency>
  <groupId>org.amqphub.spring</groupId>
  <artifactId>amqp-10-jms-spring-boot-starter</artifactId>
  <version>2.5.0.redhat-00001</version>
</dependency>
```

The client is now available in your Maven project.

2.3. INSTALLING A LOCAL MAVEN REPOSITORY

As an alternative to the online repository, AMQ Spring Boot Starter can be installed to your local filesystem as a file-based Maven repository.

Procedure

1. Use your subscription to download the AMQ Clients 2.10.0 Spring Boot Starter Maven repository .zip file.

2. Extract the file contents into a directory of your choosing. On Linux or UNIX, use the `unzip` command to extract the file contents.

   ```bash
   $ unzip amq-clients-2.10.0-spring-boot-starter-maven-repository.zip
   ```

   On Windows, right-click the .zip file and select Extract All.
3. Configure Maven to use the repository in the maven-repository directory inside the extracted install directory. For more information, see Section B.2, “Using a local repository”.
CHAPTER 3. GETTING STARTED

This chapter guides you through the steps to set up your environment and run a simple messaging program.

3.1. PREREQUISITES

- To build the example, Maven must be configured to use the Red Hat repository or a local repository.
- You must have a message broker listening for connections on localhost. It must have anonymous access enabled. For more information, see Starting the broker.
- You must have a queue named example. For more information, see Creating a queue.

3.2. RUNNING HELLO WORLD

The Hello World example creates a connection to the broker, sends a message containing a greeting to the example queue, and receives it back. On success, it prints the received message to the console.

Example: Sending and receiving "Hello World!" - HelloWorld.java

```java
package net.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.jms.annotation.EnableJms;
import org.springframework.jms.annotation.JmsListener;
import org.springframework.jms.core.JmsTemplate;

@SpringBootApplication
@EnableJms
public class HelloWorld implements CommandLineRunner {
    @Autowired
    private JmsTemplate jmsTemplate;
    
    public static void main(String[] args) {
        SpringApplication.run(HelloWorld.class, args);
    }

    @Override
    public void run(String... strings) throws Exception {
        sendMessage("Hello World!");
    }

    public void sendMessage(String text) {
        System.out.println(String.format("Sending '%s'", text));
        this.jmsTemplate.convertAndSend("example", text);
    }

    @JmsListener(destination = "example")
    public void receiveMessage(String text) {
```
Running the example
To compile and run the example program, use the following procedure:

Procedure

1. Create a new project directory. This is referred to as `<project-dir>` in the steps that follow.

2. Copy the example listing to the following location:

   `<project-dir>/src/main/java/net/example/HelloWorld.java`

3. Use a text editor to create a new `<project-dir>/pom.xml` file. Add the following XML to it:

   ```xml
   <project>
     <modelVersion>4.0.0</modelVersion>

     <groupId>net.example</groupId>
     <artifactId>example</artifactId>
     <version>1.0.0-SNAPSHOT</version>

     <parent>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-parent</artifactId>
       <version>2.5.0</version>
     </parent>

     <dependencies>
       <dependency>
         <groupId>org.amqphub.spring</groupId>
         <artifactId>amqp-10.jms.spring-boot-starter</artifactId>
         <version>2.5.0.redhat-00001</version>
       </dependency>
     </dependencies>

     <build>
       <plugins>
         <plugin>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-maven-plugin</artifactId>
         </plugin>
       </plugins>
     </build>
   </project>
   
4. Change to the project directory and use the `mvn` command to compile the program.

   ```
   $ mvn clean package
   ```

5. Use the `java` command to run the program.
Running the Hello World example results in the following console output:

$ java -jar target/example-1.0.0-SNAPSHOT.jar

Send 'Hello World!' to the exchange
Received 'Hello World!' from the queue

3.3. ADDITIONAL EXAMPLES

More example programs are available from the AMQP 1.0 JMS Spring Boot project and the Spring project.
CHAPTER 4. CONFIGURATION

The following options can be used in an application-properties file to configure your Spring Boot application.

4.1. CONNECTION OPTIONS

These options determine how AMQ Spring Boot Starter establishes connections to remote AMQP peers. The starter uses AMQ JMS to communicate over the network. For more information, see Using the AMQ JMS Client.

**amqphub.amqp10jms.remoteUrl**

The connection URI that the AMQ JMS client uses to establish new connections.

Connection URI format

```
amqp[s]://host:port[?option=value[&option2=value...]]
```

For more information, see Connection URIs in Using the AMQ JMS Client.

**amqphub.amqp10jms.username**

The username used to authenticate the connection.

**amqphub.amqp10jms.password**

The password used to authenticate the connection.

**amqphub.amqp10jms.clientId**

The client ID applied to the connection.

**amqphub.amqp10jms.receiveLocalOnly**

If enabled, calls to receive with a timeout argument check the consumer’s local message buffer only. Otherwise, the remote peer is checked as well to ensure there are no messages available. It is disabled by default.

**amqphub.amqp10jms.receiveNoWaitLocalOnly**

If enabled, calls to receiveNoWait check the consumer’s local message buffer only. Otherwise, the remote peer is checked as well to ensure there are no messages available. It is disabled by default.

4.2. POOLING OPTIONS

These options determine how AMQ Spring Boot Starter caches JMS connections and sessions. The starter uses AMQ JMS Pool for its pooling. For more information, see Using the AMQ JMS Pool Library.

**amqphub.amqp10jms.pool.enabled**

Controls whether pooling is enabled. It is disabled by default.

**amqphub.amqp10jms.pool.maxConnections**

The maximum number of connections for a single pool. The default is 1.

**amqphub.amqp10jms.pool.maxSessionsPerConnection**

The maximum number of sessions for each connection. The default is 500. A negative value removes any limit.

If the limit is exceeded, createSession() either blocks or throws an exception, depending on configuration.
amqphub.amqp10jms.pool.blockIfSessionPoolIsFull

If enabled, calls to `createSession()` block until a session becomes available in the pool. It is enabled by default.
If disabled, calls to `createSession()` throw an `IllegalStateException` if no session is available.

amqphub.amqp10jms.pool.blockIfSessionPoolIsFullTimeout

The time in milliseconds before a blocked call to `createSession()` throws an `IllegalStateException`. The default is -1, meaning the call blocks forever.

amqphub.amqp10jms.pool.connectionIdleTimeout

The time in milliseconds before a connection not currently on loan can be evicted from the pool. The default is 30 seconds. A value of 0 disables the timeout.

amqphub.amqp10jms.pool.connectionCheckInterval

The time in milliseconds between periodic checks for expired connections. The default is 0, meaning the check is disabled.

amqphub.amqp10jms.pool.useAnonymousProducers

If enabled, use a single anonymous JMS `MessageProducer` for all calls to `createProducer()`. It is enabled by default.
In rare cases, this behavior is undesirable. If disabled, every call to `createProducer()` results in a new `MessageProducer` instance.

amqphub.amqp10jms.pool.explicitProducerCacheSize

When not using anonymous producers, the JMS `Session` can be configured to cache a certain number of `MessageProducer` objects with explicit destinations. As new producers are created that do not match the cached producers, the oldest entry in the cache is evicted.

amqphub.amqp10jms.pool.useProviderJMSContext

If enabled, use the JMS `Context` classes of the underlying JMS provider. It is disabled by default.
In normal operation, the pool uses its own generic JMS `Context` implementation to wrap connections from the pool instead of using the provider implementation. The generic implementation might have limitations the provider implementation does not. However, when enabled, connections from the JMS `Context` API are not managed by the pool.
APPENDIX A. USING YOUR SUBSCRIPTION

AMQ is provided through a software subscription. To manage your subscriptions, access your account at the Red Hat Customer Portal.

A.1. ACCESSING YOUR ACCOUNT

Procedure
1. Go to access.redhat.com.
2. If you do not already have an account, create one.
3. Log in to your account.

A.2. ACTIVATING A SUBSCRIPTION

Procedure
1. Go to access.redhat.com.
2. Navigate to My Subscriptions.
3. Navigate to Activate a subscription and enter your 16-digit activation number.

A.3. DOWNLOADING RELEASE FILES

To access .zip, .tar.gz, and other release files, use the customer portal to find the relevant files for download. If you are using RPM packages or the Red Hat Maven repository, this step is not required.

Procedure
1. Open a browser and log in to the Red Hat Customer Portal Product Downloads page at access.redhat.com/downloads.
2. Locate the Red Hat AMQ entries in the INTEGRATION AND AUTOMATION category.
3. Select the desired AMQ product. The Software Downloads page opens.
4. Click the Download link for your component.

A.4. REGISTERING YOUR SYSTEM FOR PACKAGES

To install RPM packages for this product on Red Hat Enterprise Linux, your system must be registered. If you are using downloaded release files, this step is not required.

Procedure
1. Go to access.redhat.com.
2. Navigate to Registration Assistant.
3. Select your OS version and continue to the next page.
4. Use the listed command in your system terminal to complete the registration.

For more information about registering your system, see one of the following resources:

- Red Hat Enterprise Linux 7 - Registering the system and managing subscriptions
- Red Hat Enterprise Linux 8 - Registering the system and managing subscriptions
APPENDIX B. USING RED HAT MAVEN REPOSITORIES

This section describes how to use Red Hat-provided Maven repositories in your software.

B.1. USING THE ONLINE REPOSITORY

Red Hat maintains a central Maven repository for use with your Maven-based projects. For more information, see the repository welcome page.

There are two ways to configure Maven to use the Red Hat repository:

- Add the repository to your Maven settings
- Add the repository to your POM file

Adding the repository to your Maven settings

This method of configuration applies to all Maven projects owned by your user, as long as your POM file does not override the repository configuration and the included profile is enabled.

Procedure

1. Locate the Maven settings.xml file. It is usually inside the .m2 directory in the user home directory. If the file does not exist, use a text editor to create it.
   On Linux or UNIX:
   ```
   /home/<username>/.m2/settings.xml
   ```
   On Windows:
   ```
   C:\Users\<username>\m2\settings.xml
   ```

2. Add a new profile containing the Red Hat repository to the profiles element of the settings.xml file, as in the following example:

   **Example: A Maven settings.xml file containing the Red Hat repository**

   ```xml
   <settings>
   <profiles>
     <profile>
       <id>red-hat</id>
       <repositories>
         <repository>
           <id>red-hat-ga</id>
           <url>https://maven.repository.redhat.com/ga</url>
         </repository>
         <repositories>
         </pluginRepositories>
       </repositories>
     </profile>
   </profiles>
   <snaptshots>
   </snaptshots>
   ```
For more information about Maven configuration, see the [Maven settings reference](#).

**Adding the repository to your POM file**
To configure a repository directly in your project, add a new entry to the `repositories` element of your POM file, as in the following example:

**Example: A Maven pom.xml file containing the Red Hat repository**

```xml
<project>
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>example-app</artifactId>
  <version>1.0.0</version>

  <repositories>
    <repository>
      <id>red-hat-ga</id>
      <url>https://maven.repository.redhat.com/ga</url>
    </repository>
  </repositories>
</project>
```

For more information about POM file configuration, see the [Maven POM reference](#).

### B.2. USING A LOCAL REPOSITORY

Red Hat provides file-based Maven repositories for some of its components. These are delivered as downloadable archives that you can extract to your local filesystem.

To configure Maven to use a locally extracted repository, apply the following XML in your Maven settings or POM file:

```xml
<repository>
  <id>red-hat-local</id>
  <url>${repository-url}</url>
</repository>
```

`${repository-url}` must be a file URL containing the local filesystem path of the extracted repository.

**Table B.1. Example URLs for local Maven repositories**

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Local</td>
<td>${repository-url}</td>
</tr>
<tr>
<td>2</td>
<td>Remote</td>
<td><a href="https://maven.repository.redhat.com/ga">https://maven.repository.redhat.com/ga</a></td>
</tr>
<tr>
<td>Operating system</td>
<td>Filesystem path</td>
<td>URL</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Linux or UNIX</td>
<td>/home/alice/maven-repository</td>
<td>file:/home/alice/maven-repository</td>
</tr>
<tr>
<td>Windows</td>
<td>C:\repos\red-hat</td>
<td>file:C:\repos\red-hat</td>
</tr>
</tbody>
</table>
APPENDIX C. USING AMQ BROKER WITH THE EXAMPLES

The AMQ Spring Boot Starter examples require a running message broker with a queue named `example`. Use the procedures below to install and start the broker and define the queue.

C.1. INSTALLING THE BROKER

Follow the instructions in *Getting Started with AMQ Broker* to install the broker and create a broker instance. Enable anonymous access.

The following procedures refer to the location of the broker instance as `<broker-instance-dir>`.

C.2. STARTING THE BROKER

Procedure

1. Use the `artemis run` command to start the broker.

   $ <broker-instance-dir>/bin/artemis run

2. Check the console output for any critical errors logged during startup. The broker logs *Server is now live* when it is ready.

   $ example-broker/bin/artemis run

   Red Hat AMQ <version>

   ...
   2020-06-03 12:12:12,336 INFO [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
   ...

C.3. CREATING A QUEUE

In a new terminal, use the `artemis queue` command to create a queue named `example`.

   $ <broker-instance-dir>/bin/artemis queue create --name example --address example --auto-create-address --anycast

You are prompted to answer a series of yes or no questions. Answer N for no to all of them.

Once the queue is created, the broker is ready for use with the example programs.

C.4. STOPPING THE BROKER

$ <broker-instance-dir>/bin/artemis run
When you are done running the examples, use the `artemis stop` command to stop the broker.

```
$ <broker-instance-dir>/bin/artemis stop
```

Revised on 2021-08-24 14:29:42 UTC