Red Hat AMQ Broker 7.11

管理 AMQ Broker

用于 AMQ Broker 7.11

Last Updated: 2024-06-11

Red Hat AMQ Broker 7.11 管理 AMQ Broker

用于 AMQ Broker 7.11

法律通告

摘要

本指南介绍了如何监控、管理和升级 AMQ Broker。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 概述

AMQ Broker 是一个基于 ActiveMQ Artemis 的高性能消息传递的实施。它具有快速、基于日志的消息持久性，并支持多种语言、协议和平台。

AMQ Broker 提供多个接口来管理和与代理实例交互，如管理控制台、管理 API 和命令行界面。另外，您可以通过收集运行时指标来监控代理性能，将代理配置为主动监控死锁条件等问题，并以交互方式检查代理和队列的健康状态。

本指南提供有关典型代理管理任务的详细信息，例如：

- 升级代理实例
- 使用命令行界面和管理 API
- 检查代理和队列的健康状况
- 收集代理运行时指标
- 主动监控关键代理操作

1.1. 支持的配置

有关 AMQ Broker 支持的配置的当前信息，请参阅红帽客户门户网站中的"Red Hat AMQ 7 支持的配置 "文章。

1.2. 文档惯例

本文档对 **sudo** 命令、文件路径和可替换值使用以下惯例：

sudo 命令
在本文档中，**sudo** 用于任何需要 root 特权的命令。使用 **sudo** 时，您应始终谨慎操作，因为任何更改都可能会影响整个系统。

有关使用 **sudo** 的更多信息，请参阅管理 sudo 访问。

关于在此文档中使用文件路径
在这个文档中，所有文件路径都对 Linux、UNIX 和类似操作系统（例如 **/home/...**）有效。如果您使用的是 Microsoft Windows，则应使用等效的 Microsoft Windows 路径（例如，**C:\Users\...**）。

可替换值
本文档有时会使用可替换值，您必须将这些值替换为特定于环境的值。可替换的值为小写，以尖括号 (**<>**)括起，样式则使用斜体和 **monospace** 字体。用下划线(_)分隔多个词语。

例如，在以下命令中，将 ***<install_dir>*** 替换为您自己的目录名称。

```
$ <install_dir>/bin/artemis create mybroker
```

第 2 章 升级代理

2.1. 关于升级

红帽向 客户门户网站 发布新版本的 AMQ Broker。将代理更新至最新版本，以确保您有最新的改进和修复。通常，红帽通过三种方法之一发布 AMQ Broker 的新版本：

主发行版本

当应用程序从一个主版本转换到下一个主版本时，需要一个主要升级或迁移，例如从 AMQ Broker 6 转换到 AMQ Broker 7。本指南中没有解决这种类型的升级。

次发行版本

AMQ Broker 定期提供次发行版本，它们是包括新功能的更新，以及程序错误和安全修复。如果您计划从一个 AMQ Broker 次版本升级到另一个 AMQ Broker 次版本，例如从 AMQ Broker 7.0 升级到 AMQ Broker 7.1，则不应对不使用私有、不受支持或技术预览组件的应用程序进行更改。

微版本

AMQ Broker 定期提供包含小改进和修复的微版本。微版本将次版本递增到最后一个数字，例如从 7.0.1 增加到 7.0.2。微版本应该不需要更改代码，但有些版本可能需要配置更改。

2.2. 升级旧的 7.X 版本

2.2.1. 将代理实例从 7.0.x 升级到 7.0.y

将 AMQ Broker 从 7.0 的一个版本升级到另一个版本的步骤与安装类似：您可以从客户门户网站下载存档，然后提取它。

下面的子章节描述了如何为不同的操作系统升级 7.0.x 代理。

- 在 Linux 上从 7.0.x 升级到 7.0.y
- 在 Windows 上从 7.0.x 升级到 7.0.y

2.2.1.1. 在 Linux 上从 7.0.x 升级到 7.0.y

您下载的存档名称可能与以下示例中使用的内容不同。

先决条件

- 在升级 AMQ Broker 前，请查看目标发行版本的发行注记。
  本发行注记描述了目标版本中行为的重要改进、已知问题和更改。

  如需更多信息，请参阅 AMQ Broker 7.0 发行注记 。

流程

1. 按照 下载 AMQ Broker 归档中的说明，从红帽客户门户网站下载所需的归档。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。

```
sudo chown amq-broker:amq-broker jboss-amq-7.x.x.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。在以下示例中，使用了目录 **/opt/redhat**。

```
sudo mv jboss-amq-7.x.x.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。归档采用压缩格式。在以下示例中，用户 **amq-broker** 使用 unzip 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip jboss-amq-7.x.x.redhat-1.zip
```

5. 如果代理正在运行，则停止代理。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，日志文件末尾会显示类似如下的行，该文件可在 < ***broker_instance_dir>*** **/log/artemis.log** 中找到。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.0.0.amq-700005-redhat-1 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

8. 编辑 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 配置文件，将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME='/opt/redhat/jboss-amq-7.x.x-redhat-1'
```

9. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

10. （可选）确认代理正在运行并且版本已更改。启动代理后，打开日志文件 < ***broker_instance_dir>*** **/log/artemis.log**，并找到类似如下的两行。请注意代理实时后日志中显示的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.1.0.amq-700005-redhat-1 [0.0.0.0, nodeID=4782d50d-47a2-11e7-a160-9801a793ea45]
```

2.2.1.2. 在 Windows 上从 7.0.x 升级到 7.0.y

先决条件

- 在升级 AMQ Broker 前，请查看目标发行版本的发行注记。
本发行注记描述了目标版本中行为的重要改进、已知问题和更改。

如需更多信息，请参阅 AMQ Broker 7.0 发行注记。

流程

1. 按照 下载 AMQ Broker 归档中的说明，从红帽客户门户网站下载所需的归档。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 输入以下命令，如果代理正在运行，则停止代理。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。
   b. 右键点击同一窗口并选择 **粘贴**。
6. （可选）请注意代理的当前版本。代理停止后，日志文件末尾会显示类似如下的行，该文件可在 < ***broker_instance_dir>*** **\log\artemis.log** 中找到。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.0.0.amq-700005-redhat-1 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 配置文件，将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

9. （可选）确认代理正在运行并且版本已更改。启动代理后，打开日志文件 < ***broker_instance_dir>*** **\log\artemis.log**，找到类似如下的两行。请注意代理实时后日志中显示的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.1.0.amq-700005-redhat-1 [0.0.0.0, nodeID=4782d50d-47a2-11e7-a160-9801a793ea45]
```

2.2.2. 将代理实例从 7.0.x 升级到 7.1.0

AMQ Broker 7.1.0 包括之前版本中没有包括的配置文件和设置。将代理实例从 7.0.x 升级到 7.1.0 需要将这些新文件和设置添加到现有 7.0.x 代理实例中。下面的子部分描述了如何为不同的操作系统将 7.0.x 代理实例升级到 7.1.0。

重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.0.x 升级到 7.1.0
- 在 Windows 上从 7.0.x 升级到 7.1.0

2.2.2.1. 在 Linux 上从 7.0.x 升级到 7.1.0

在升级 7.0.x 代理前，您需要安装 Red Hat AMQ Broker 7.1.0 并创建一个临时代理实例。这将生成升级 7.0.x 代理所需的 7.1.0 配置文件。

先决条件

- 在升级 AMQ Broker 前，请查看目标发行版本的发行注记。
  本发行注记描述了目标版本中行为的重要改进、已知问题和更改。

  如需更多信息，请参阅 AMQ Broker 7.1 发行注记。
- 在升级 7.0.x 代理前，您必须首先安装 7.1 版本。
  有关在 Linux 上安装 7.1 的步骤，请参阅安装 AMQ Broker。

流程

1. 如果正在运行，请停止您要升级的 7.0.x 代理：

   ```
   $ <broker_instance_dir>/bin/artemis stop
   ```

2. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

   ```
   cp -r <broker_instance_dir> ~/
   ```

3. 在 7.0.x 代理的 < ***broker_instance_dir>*** **/etc/** 目录中打开 **artemis.profile** 文件。

   a. 更新 **ARTEMIS_HOME** 属性，以便其值引用 AMQ Broker 7.1.0 的安装目录：

      ```
      ARTEMIS_HOME="<7.1.0_install_dir>"
      ```

   b. 在您更新的行中，添加属性 **ARTEMIS_INSTANCE_URI**，并为它分配一个指向 7.0.x 代理实例目录的值：

      ```
      ARTEMIS_INSTANCE_URI="file://<7.0.x_broker_instance_dir>"
      ```

   c. 通过添加 **jolokia.policyLocation** 参数并为其分配以下值来更新 **JAVA_ARGS** 属性：

      ```
      -Djolokia.policyLocation=${ARTEMIS_INSTANCE_URI}/etc/jolokia-access.xml
      ```

4. 创建 7.1.0 代理实例。创建过程会生成从 7.0.x 升级到 7.1.0 所需的配置文件。在以下示例中，请注意实例在目录 **upgrade_tmp** 中创建：

```
$ <7.1.0_install_dir>/bin/artemis create --allow-anonymous --user admin --password admin
upgrade_tmp
```

5. 将临时 7.1.0 实例的 **etc** 目录中的配置文件复制到 7.0.x 代理的 & ***lt;broker_instance_dir>*** **/etc/** 目录中。

   a. 复制 **management.xml** 文件：

   ```
   $ cp <temporary_7.1.0_broker_instance_dir>/etc/management.xml
   <7.0_broker_instance_dir>/etc/
   ```

   b. 复制 **jolokia-access.xml** 文件：

   ```
   $ cp <temporary_7.1.0_broker_instance_dir>/etc/jolokia-access.xml
   <7.0_broker_instance_dir>/etc/
   ```

6. 在 7.0.x 代理的 < ***broker_instance_dir>*** **/etc/** 目录中打开 **bootstrap.xml** 文件。

   a. 注释掉或删除以下两行：

   ```
   <app url="jolokia" war="jolokia.war"/>
   <app url="hawtio" war="hawtio-no-slf4j.war"/>
   ```

   b. 添加以下内容来替换上一步中删除的两行：

   ```
   <app url="console" war="console.war"/>
   ```

7. 启动您升级的代理：

   ```
   $ <broker_instance_dir>/bin/artemis run
   ```

其它资源

有关创建代理实例的更多信息，请参阅创建代理实例。

2.2.2.2. 在 Windows 上从 7.0.x 升级到 7.1.0

在升级 7.0.x 代理前，您需要安装 Red Hat AMQ Broker 7.1.0 并创建一个临时代理实例。这将生成升级 7.0.x 代理所需的 7.1.0 配置文件。

先决条件

- 在升级 AMQ Broker 前，请查看目标发行版本的发行注记。
  本发行注记描述了目标版本中行为的重要改进、已知问题和更改。

  如需更多信息，请参阅 AMQ Broker 7.1 发行注记。

- 在升级 7.0.x 代理前，您必须首先安装 7.1 版本。
  有关在 Windows 上安装 7.1 的步骤，请参阅安装 AMQ Broker。

流程

1. 如果正在运行，请停止您要升级的 7.0.x 代理：

```
> <broker_instance_dir>\bin\artemis-service.exe stop
```

2. 使用文件管理器备份代理的实例目录。

   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。

   b. 右键点击同一窗口并选择 **粘贴**。

3. 在 7.0.x 代理的 < ***broker_instance_dir>*** **/etc/** 目录中打开 **artemis.profile** 文件。

   a. 更新 **ARTEMIS_HOME** 属性，以便其值引用 AMQ Broker 7.1.0 的安装目录：

   ```
   ARTEMIS_HOME="<7.1.0_install_dir>"
   ```

   b. 在您更新的行中，添加属性 **ARTEMIS_INSTANCE_URI**，并为它分配一个指向 7.0.x 代理实例目录的值：

   ```
   ARTEMIS_INSTANCE_URI="file://<7.0.x_broker_instance_dir>"
   ```

   c. 通过添加 **jolokia.policyLocation** 参数并为其分配以下值来更新 **JAVA_ARGS** 属性：

   ```
   -Djolokia.policyLocation=${ARTEMIS_INSTANCE_URI}/etc/jolokia-access.xml
   ```

4. 创建 7.1.0 代理实例。创建过程会生成从 7.0.x 升级到 7.1.0 所需的配置文件。在以下示例中，请注意实例在目录 **upgrade_tmp** 中创建：

```
> <7.1.0_install_dir>/bin/artemis create --allow-anonymous --user admin --password admin
upgrade_tmp
```

5. 将临时 7.1.0 实例的 **etc** 目录中的配置文件复制到 7.0.x 代理的 & ***lt;broker_instance_dir>*** **/etc/** 目录中。

   a. 复制 **management.xml** 文件：

   ```
   > cp <temporary_7.1.0_broker_instance_dir>/etc/management.xml
   <7.0_broker_instance_dir>/etc/
   ```

   b. 复制 **jolokia-access.xml** 文件：

   ```
   > cp <temporary_7.1.0_broker_instance_dir>/etc/jolokia-access.xml
   <7.0_broker_instance_dir>/etc/
   ```

6. 在 7.0.x 代理的 < ***broker_instance_dir>*** **/etc/** 目录中打开 **bootstrap.xml** 文件。

   a. 注释掉或删除以下两行：

   ```
   <app url="jolokia" war="jolokia.war"/>
   <app url="hawtio" war="hawtio-no-slf4j.war"/>
   ```

   b. 添加以下内容来替换上一步中删除的两行：

   ```
   <app url="console" war="console.war"/>
   ```

7. 启动您升级的代理：

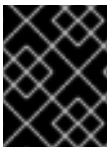
```
> <broker_instance_dir>\bin\artemis-service.exe start
```

其它资源

有关创建代理实例的更多信息，请参阅创建代理实例。

2.2.3. 将代理实例从 7.1.x 升级到 7.2.0

AMQ Broker 7.2.0 包括没有包括在 7.0.x 版本中的配置文件和设置。如果您正在运行 7.0.x 实例，您必须首先将这些代理实例从 7.0.x 升级到 7.1.0， 然后才能升级到 7.2.0。下面的小节描述了如何为不同的操作系统将 7.1.x 代理实例升级到 7.2.0。



重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.1.x 升级到 7.2.0
- 在 Windows 上从 7.1.x 升级到 7.2.0

2.2.3.1. 在 Linux 上从 7.1.x 升级到 7.2.0

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 按照 下载 AMQ Broker 归档中的说明，从红帽客户门户网站下载所需的归档。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。

```
sudo chown amq-broker:amq-broker amq-7.x.x.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。在以下示例中，使用了目录 **/opt/redhat**。

```
sudo mv amq-7.x.x.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 unzip 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip jboss-amq-7.x.x.redhat-1.zip
```

5. 如果代理正在运行，则停止代理。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，日志文件末尾会显示类似如下的行，该文件可在 < ***broker_instance_dir>*** **/log/artemis.log** 中找到。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.5.0.amq-720001-redhat-1 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 配置文件，将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME='/opt/redhat/amq-7.x.x-redhat-1'
```

9. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

10. （可选）确认代理正在运行并且版本已更改。启动代理后，打开日志文件 < ***broker_instance_dir>*** **/log/artemis.log**，并找到类似如下的两行。请注意代理实时后日志中显示的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.5.0.amq-720001-redhat-1 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.2.3.2. 在 Windows 上从 7.1.x 升级到 7.2.0

流程

1. 按照 下载 AMQ Broker 归档中的说明，从红帽客户门户网站下载所需的归档。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 输入以下命令，如果代理正在运行，则停止代理。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。

   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。

   b. 右键点击同一窗口并选择 **粘贴**。

6. （可选）请注意代理的当前版本。代理停止后，日志文件末尾会显示类似如下的行，该文件可在 < ***broker_instance_dir>*** **\log\artemis.log** 中找到。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.0.0.amq-700005-redhat-1 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>*** **\etc\artemis.profile.cmd** 和 < ***broker_instance_dir>*** **\bin\artemis-service.xml** 配置文件，将 **ARTEMIS_HOME** 属性设置为在提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

9. （可选）确认代理正在运行并且版本已更改。启动代理后，打开日志文件 < ***broker_instance_dir>*** **\log\artemis.log**，找到类似如下的两行。请注意代理实时后日志中显示的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.5.0.amq-720001-redhat-1 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.2.4. 将代理实例从 7.2.x 升级到 7.3.0

下面的子部分描述了如何为不同的操作系统将 7.2.x 代理实例升级到 7.3.0。

2.2.4.1. 解决已弃用的分配控制台时出现异常

从版本 7.1.0 开始，AMQ Broker 不再附带 Hawtio 分配控制台插件 **dispatch-hawtio-console.war**。在以前的版本中，分配控制台用于管理 AMQ Interconnect。但是，AMQ Interconnect 现在使用自己的独立 Web 控制台。这个更改会影响后续部分中的升级步骤。

如果您在将代理实例升级到 7.3.0 之前没有进一步的操作，升级过程会生成类似如下的异常：

```
2019-04-11 18:00:41,334 WARN  [org.eclipse.jetty.webapp.WebAppContext] Failed startup of context o.e.j.w.WebAppContext@1ef3efa8{/dispatch-hawtio-console,null,null}{/opt/amqbroker/amq-broker-7.3.0/web/dispatch-hawtio-console.war}: java.io.FileNotFoundException: /opt/amqbroker/amq-broker-7.3.0/web/dispatch-hawtio-console.war.
```

您可以在不影响升级成功的情况下安全地忽略前面的异常。

但是，如果您不希望在升级过程中看到这个异常，您必须首先在现有代理实例的 **bootstrap.xml** 文件中删除对 Hawtio 分配控制台插件的引用。**bootstrap.xml** 文件位于代理实例的 **{instance_directory}/etc/** 目录中。以下示例显示了 AMQ Broker 7.2.4 实例的 **bootstrap.xml** 文件的内容：

```
<broker xmlns="http://activemq.org/schema">
....
  <!-- The web server is only bound to localhost by default -->
  <web bind="http://localhost:8161" path="web">
     <app url="redhat-branding" war="redhat-branding.war"/>
     <app url="artemis-plugin" war="artemis-plugin.war"/>
     <app url="dispatch-hawtio-console" war="dispatch-hawtio-console.war"/>
     <app url="console" war="console.war"/>
  </web>
</broker>
```

为了避免在将 AMQ Broker 升级到 7.3.0 时出现异常，请删除 < **app url="dispatch-hawtio-console" war="dispatch-hawtio-console.war"/&gt**; 行，如上例所示。然后，保存修改后的 bootstrap 文件并启动升级过程，如后续章节所述。

重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.2.x 升级到 7.3.0
- 在 Windows 上从 7.2.x 升级到 7.3.0

2.2.4.2. 在 Linux 上从 7.2.x 升级到 7.3.0

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 按照 下载 AMQ Broker 归档中的说明，从红帽客户门户网站下载所需的归档。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。

```
sudo chown amq-broker:amq-broker amq-7.x.x.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。在以下示例中，使用了目录 **/opt/redhat**。

```
sudo mv amq-7.x.x.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 unzip 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip jboss-amq-7.x.x.redhat-1.zip
```

5. 如果代理正在运行，则停止代理。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，日志文件末尾会显示类似如下的行，该文件可在 < ***broker_instance_dir>*** **/log/artemis.log** 中找到。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.6.3.amq-720001-redhat-1 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 配置文件，将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME='/opt/redhat/amq-7.x.x-redhat-1'
```

9. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

10. （可选）确认代理正在运行并且版本已更改。启动代理后，打开日志文件 < ***broker_instance_dir>*** **/log/artemis.log**，并找到类似如下的两行。请注意代理实时后日志中显示的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.7.0.redhat-00054 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.2.4.3. 在 Windows 上从 7.2.x 升级到 7.3.0

流程

1. 按照 下载 AMQ Broker 归档中的说明，从红帽客户门户网站下载所需的归档。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 输入以下命令，如果代理正在运行，则停止代理。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。
   b. 右键点击同一窗口并选择 **粘贴**。
6. （可选）请注意代理的当前版本。代理停止后，日志文件末尾会显示类似如下的行，该文件可在 < ***broker_instance_dir>*** **\log\artemis.log** 中找到。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.6.3.amq-720001-redhat-1 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>*** **\etc\artemis.profile.cmd** 和 < ***broker_instance_dir>*** **\bin\artemis-service.xml** 配置文件，将 **ARTEMIS_HOME** 属性设置为在提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 编辑 & ***lt;broker_instance_dir>*** **\etc\artemis.profile.cmd** 配置文件，以设置 JAVA_ARGS 环境变量来引用正确的日志管理器版本。

```
JAVA_ARGS=<install_dir>\lib\jboss-logmanager-2.0.3.Final-redhat-1.jar
```

9. 编辑 & ***lt;broker_instance_dir>*** **\bin\artemis-service.xml** 配置文件，以设置 bootstrap 类路径启动参数来引用正确的日志管理器版本。

```
<startargument>Xbootclasspath/a:%ARTEMIS_HOME%\lib\jboss-logmanager-2.0.3.Final-redhat-1.jar</startargument>
```

10. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

11. （可选）确认代理正在运行并且版本已更改。启动代理后，打开日志文件 < ***broker_instance_dir>*** **\log\artemis.log**，找到类似如下的两行。请注意代理实时后日志中显示的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis
```

```
Message Broker version 2.7.0.redhat-00054 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.2.5. 将代理实例从 7.3.0 升级到 7.4.0

下面的子部分描述了如何为不同的操作系统将 7.3.0 代理实例升级到 7.4.0。

重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.3.0 升级到 7.4.0
- 在 Windows 上从 7.3.0 升级到 7.4.0

2.2.5.1. 在 Linux 上从 7.3.0 升级到 7.4.0

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。以下示例显示了名为 **amq-broker** 的用户。

```
sudo chown amq-broker:amq-broker amq-broker-7.x.x.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.x.x.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.x.x.redhat-1.zip
```

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.7.0.redhat-00054 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir&*gt; /etc/artemis.profile** 配置文件。

   a. 将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME='/opt/redhat/amq-broker-7.x.x-redhat-1'
```

   b. 编辑 **JAVA_ARGS** 属性。添加 bootstrap 类路径参数，该参数引用日志管理器的依赖文件。

```
-Xbootclasspath/a:$ARTEMIS_HOME/lib/wildfly-common-1.5.1.Final-redhat-00001.jar
```

9. 编辑 & ***lt;broker_instance_dir&*gt; /etc/bootstrap.xml** 配置文件。在 & **lt;web** > 配置元素中，为 AMQ Broker 添加对 metrics 插件文件的引用。

```
<app url="metrics" war="metrics.war"/>
```

10. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

11. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*/log/artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>* /etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.2.5.2. 在 Windows 上从 7.3.0 升级到 7.4.0

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 如果代理正在运行，请停止它。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。
   b. 右键点击同一窗口并选择 **粘贴**。
6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*\log\artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.7.0.redhat-00054 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 和 ***<broker_instance_dir>* \bin\artemis-service.xml** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 配置文件。设置 JAVA_ARGS 环境变量来引用正确的日志管理器版本和依赖文件。

```
JAVA_ARGS=-Xbootclasspath/%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.1.Final-redhat-00001.jar
```

9. 编辑 & ***lt;broker_instance_dir>* \bin\artemis-service.xml** 配置文件。设置 bootstrap 类路径启动参数来引用正确的日志管理器版本和依赖文件。

```
<startargument>-Xbootclasspath/a:%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.1.Final-redhat-00001.jar</startargument>
```

10. 编辑 & ***lt;broker_instance_dir>* \etc\bootstrap.xml** 配置文件。在 & **lt;web** > 配置元素中，为 AMQ Broker 添加对 metrics 插件文件的引用。

```
<app url="metrics" war="metrics.war"/>
```

11. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

12. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 <***broker_instance_dir*>\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

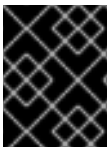
其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir*> \etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.3. 将代理实例从 7.4.0 升级到 7.4.X

重要

AMQ Broker 7.4 被指定为 Long Term Support (LTS)发行版本。在一系列微发行本 (7.4.1、7.4.1.2 等)中，AMQ Broker 7.4 将提供程序错误修正和安全公告（至少 12 个月）。这意味着，您可以在不需要升级到新的次版本的情况下为 AMQ Broker 获取最新的程序错误修复和安全公告。如需更多信息，请参阅 AMQ Broker 的长期支持。



重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

以下小节介绍了如何为不同的操作系统将 7.4.0 代理实例升级到 7.4.x。

- 在 Linux 上从 7.4.0 升级到 7.4.x
- 在 Windows 上从 7.4.0 升级到 7.4.x

2.3.1. 在 Linux 上从 7.4.0 升级到 7.4.x

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。以下示例显示了名为 **amq-broker** 的用户。

```
sudo chown amq-broker:amq-broker amq-broker-7.4.x.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.4.x.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.4.x.redhat-1.zip
```

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.7.0.redhat-00054 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir>* /etc/artemis.profile** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME='/opt/redhat/amq-broker-7.4.x-redhat-1'
```

9. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

10. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*/log/artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>* /etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件

和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.3.2. 在 Windows 上从 7.4.0 升级到 7.4.x

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 如果代理正在运行，请停止它。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。
   b. 右键点击同一窗口并选择 **粘贴**。
6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*\log\artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.7.0.redhat-00054 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>*** **\etc\artemis.profile.cmd** 和 ***<broker_instance_dir>*** **\bin\artemis-service.xml** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

9. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的

位置。在 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.4. 将代理实例从 7.4.X 升级到 7.5.0

以下小节介绍了如何为不同的操作系统将 7.4.x 代理实例升级到 7.5.0。

重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.4.x 升级到 7.5.0
- 在 Windows 上从 7.4.x 升级到 7.5.0

2.4.1. 在 Linux 上从 7.4.x 升级到 7.5.0

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。以下示例显示了名为 **amq-broker** 的用户。

```
sudo chown amq-broker:amq-broker amq-broker-7.5.0.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.5.0.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.5.0.redhat-1.zip
```

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.7.0.redhat-00054 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir>* /etc/artemis.profile** 配置文件。

   a. 将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME='/opt/redhat/amq-broker-7.5.0-redhat-1'
```

   b. 编辑 **JAVA_ARGS** 属性。添加 bootstrap 类路径参数，该参数引用日志管理器的依赖文件。

```
-Xbootclasspath/a:$ARTEMIS_HOME/lib/wildfly-common-1.5.2.Final-redhat-00001.jar
```

9. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

10. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*/log/artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>* /etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.4.2. 在 Windows 上从 7.4.x 升级到 7.5.0

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 如果代理正在运行，请停止它。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。

   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。

   b. 右键点击同一窗口并选择 **粘贴**。

6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*\log\artemis.log** 文件末尾看到类似如下的行。

   ```
   INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.7.0.redhat-00054 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
   ```

7. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 和 ***<broker_instance_dir>* \bin\artemis-service.xml** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

   ```
   ARTEMIS_HOME=<install_dir>
   ```

8. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 配置文件。设置 **JAVA_ARGS** 环境变量来引用正确的日志管理器版本和依赖文件。

   ```
   JAVA_ARGS=-Xbootclasspath/%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00001.jar
   ```

9. 编辑 & ***lt;broker_instance_dir>* \bin\artemis-service.xml** 配置文件。设置 bootstrap 类路径启动参数来引用正确的日志管理器版本和依赖文件。

   ```
   <startargument>-Xbootclasspath/a:%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00001.jar</startargument>
   ```

10. 启动升级的代理。

    ```
    <broker_instance_dir>\bin\artemis-service.exe start
    ```

11. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

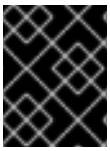
    ```
    INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
    ...
    INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
    ```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>* \etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.5. 将代理实例从 7.5.0 升级到 7.6.0

下面的子部分描述了如何为不同的操作系统将 7.5.0 代理实例升级到 7.6.0。



重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.5.0 升级到 7.6.0
- 在 Windows 上从 7.5.0 升级到 7.6.0

2.5.1. 在 Linux 上从 7.5.0 升级到 7.6.0

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。以下示例显示了名为 **amq-broker** 的用户。

```
sudo chown amq-broker:amq-broker amq-broker-7.6.0.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.6.0.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.6.0.redhat-1.zip
```

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00054 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 配置文件。

   a. 将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

   ```
   ARTEMIS_HOME='/opt/redhat/amq-broker-7.6.0-redhat-1'
   ```

   b. 编辑 **JAVA_ARGS** 属性。添加 bootstrap 类路径参数，该参数引用日志管理器的依赖文件。

   ```
   -Xbootclasspath/a:$ARTEMIS_HOME/lib/wildfly-common-1.5.2.Final-redhat-00002.jar
   ```

9. 启动升级的代理。

   ```
   <broker_instance_dir>/bin/artemis run
   ```

10. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 <***broker_instance_dir>*** **/log/artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

    ```
    INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
    ...
    INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.11.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
    ```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.5.2. 在 Windows 上从 7.5.0 升级到 7.6.0

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 如果代理正在运行，请停止它。

   ```
   <broker_instance_dir>\bin\artemis-service.exe stop
   ```

5. 使用文件管理器备份代理。

a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 Copy。

b. 右键点击同一窗口并选择 **粘贴**。

6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*****\log\artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00054 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>*** **\etc\artemis.profile.cmd** 和 ***<broker_instance_dir>*** **\bin\artemis-service.xml** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 编辑 & ***lt;broker_instance_dir>*** **\etc\artemis.profile.cmd** 配置文件。设置 **JAVA_ARGS** 环境变量来引用正确的日志管理器版本和依赖文件。

```
JAVA_ARGS=-Xbootclasspath/%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar
```

9. 编辑 & ***lt;broker_instance_dir>*** **\bin\artemis-service.xml** 配置文件。设置 bootstrap 类路径启动参数来引用正确的日志管理器版本和依赖文件。

```
<startargument>-Xbootclasspath/a:%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar</startargument>
```

10. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

11. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*****\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.11.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.6. 将代理实例从 7.6.0 升级到 7.7.0

下面的子部分描述了如何为不同的操作系统将 7.6.0 代理实例升级到 7.7.0。

重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.6.0 升级到 7.7.0
- 在 Windows 上从 7.6.0 升级到 7.7.0

2.6.1. 在 Linux 上从 7.6.0 升级到 7.7.0

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。以下示例显示了名为 **amq-broker** 的用户。

```
sudo chown amq-broker:amq-broker amq-broker-7.7.0.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.7.0.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.7.0.redhat-1.zip
```

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis
Message Broker version 2.11.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-
54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir&*gt; /etc/artemis.profile** 配置文件。

   a. 将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。例如：

   ```
   ARTEMIS_HOME='/opt/redhat/amq-broker-7.7.0-redhat-1'
   ```

   b. 找到 **JAVA_ARGS** 属性。确保 bootstrap 类路径参数引用日志管理器所需的依赖文件版本，如下所示。

   ```
   -Xbootclasspath/a:$ARTEMIS_HOME/lib/wildfly-common-1.5.2.Final-redhat-00002.jar
   ```

9. 编辑 & ***lt;broker_instance_dir&*gt; /etc/logging.properties** 配置文件。

   a. 在要配置的额外日志记录器列表中，包括在 AMQ Broker 7.7.0 中添加的 **org.apache.activemq.audit.resource** 资源日志记录器。

   ```
   loggers=org.eclipse.jetty,org.jboss.logging,org.apache.activemq.artemis.core.server,org.ap
   ache.activemq.artemis.utils,org.apache.activemq.artemis.journal,org.apache.activemq.arte
   mis.jms.server,org.apache.activemq.artemis.integration.bootstrap,org.apache.activemq.aud
   it.base,org.apache.activemq.audit.message,org.apache.activemq.audit.resource
   ```

   b. 在 **Console handler 配置部分** 之前，为资源日志记录器添加默认配置。

   ```
   ..

   logger.org.apache.activemq.audit.resource.level=ERROR
   logger.org.apache.activemq.audit.resource.handlers=AUDIT_FILE
   logger.org.apache.activemq.audit.resource.useParentHandlers=false

   # Console handler configuration
   ..
   ```

10. 启动升级的代理。

    ```
    <broker_instance_dir>/bin/artemis run
    ```

11. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*/log/artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

    ```
    INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
    ...
    INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis
    Mesq.audit.resource.handlers=AUDIT_FILE
    logger.org.apache.activemq.audit.resource.useParentHandlers=false
    sage Broker version 2.13.0.redhat-00003 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-
    54ee759954c4]
    ```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 &*lt;broker_instance_dir*> **/etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.6.2. 在 Windows 上从 7.6.0 升级到 7.7.0

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 如果代理正在运行，请停止它。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。
   b. 右键点击同一窗口并选择 **粘贴**。
6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*\log\artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.11.0.redhat-00001 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 &*lt;broker_instance_dir>* **\etc\artemis.profile.cmd** 和 ***<broker_instance_dir>*\bin\artemis-service.xml** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 编辑 &*lt;broker_instance_dir>* **\etc\artemis.profile.cmd** 配置文件。确保 **JAVA_ARGS** 环境变量引用日志管理器和依赖文件的正确版本，如下所示。

```
JAVA_ARGS=-Xbootclasspath/%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar
```

9. 编辑 &*lt;broker_instance_dir>* **\bin\artemis-service.xml** 配置文件。确保 bootstrap 类路径启动参数引用日志管理器和依赖文件的正确版本，如下所示。

```
<startargument>-Xbootclasspath/a:%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar</startargument>
```

10. 编辑 &*lt;broker_instance_dir>* **\etc\logging.properties** 配置文件。

a. 在要配置的额外日志记录器列表中，包括在 AMQ Broker 7.7.0 中添加的 **org.apache.activemq.audit.resource** 资源日志记录器。

```
loggers=org.eclipse.jetty,org.jboss.logging,org.apache.activemq.artemis.core.server,org.apache.activemq.artemis.utils,org.apache.activemq.artemis.journal,org.apache.activemq.artemis.jms.server,org.apache.activemq.artemis.integration.bootstrap,org.apache.activemq.audit.base,org.apache.activemq.audit.message,org.apache.activemq.audit.resource
```

b. 在 **Console handler 配置部分** 之前，为资源日志记录器添加默认配置。

```
..

logger.org.apache.activemq.audit.resource.level=ERROR
logger.org.apache.activemq.audit.resource.handlers=AUDIT_FILE
logger.org.apache.activemq.audit.resource.useParentHandlers=false

# Console handler configuration
..
```

11. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

12. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 <*broker_instance_dir*>**\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

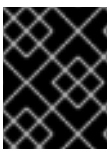
```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.13.0.redhat-00003 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.7. 将代理实例从 7.7.0 升级到 7.8.0

下面的小节描述了如何为不同的操作系统将 7.7.0 代理实例升级到 7.8.0。



重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.7.0 升级到 7.8.0

- 在 Windows 上从 7.7.0 升级到 7.8.0

2.7.1. 在 Linux 上从 7.7.0 升级到 7.8.0

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。以下示例显示了名为 **amq-broker** 的用户。

```
sudo chown amq-broker:amq-broker amq-broker-7.8.0.redhat-1.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.8.0.redhat-1.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.8.0.redhat-1.zip
```

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.13.0.redhat-00003 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 配置文件。
   a. 将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。例如：

```
ARTEMIS_HOME='/opt/redhat/amq-broker-7.8.0-redhat-1'
```

   b. 找到 **JAVA_ARGS** 属性。确保 bootstrap 类路径参数引用日志管理器所需的依赖文件版本，如下所示。

```
-Xbootclasspath/a:$ARTEMIS_HOME/lib/wildfly-common-1.5.2.Final-redhat-00002.jar
```

9. 编辑 & ***lt;broker_instance_dir>*** **/etc/bootstrap.xml** 配置文件。在 **web** 元素中，更新 7.8 中 AMQ 管理控制台所需的 **.war** 文件的名称。

```
<web bind="http://localhost:8161" path="web">
    ...
    <app url="console" war="hawtio.war"/>
    ...
</web>
```

10. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

11. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*** **/log/artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis
Mesq.audit.resource.handlers=AUDIT_FILE
logger.org.apache.activemq.audit.resource.useParentHandlers=false
sage Broker version 2.16.0.redhat-00007 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-
54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.7.2. 在 Windows 上从 7.7.0 升级到 7.8.0

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 如果代理正在运行，请停止它。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。

b. 右键点击同一窗口并选择 **粘贴**。

6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir*>\log\artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.13.0.redhat-00003 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 和 ***<broker_instance_dir>* \bin\artemis-service.xml** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 配置文件。确保 **JAVA_ARGS** 环境变量引用日志管理器和依赖文件的正确版本，如下所示。

```
JAVA_ARGS=-Xbootclasspath/%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar
```

9. 编辑 & ***lt;broker_instance_dir>* \bin\artemis-service.xml** 配置文件。确保 bootstrap 类路径启动参数引用日志管理器和依赖文件的正确版本，如下所示。

```
<startargument>-Xbootclasspath/a:%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar</startargument>
```

10. 编辑 & ***lt;broker_instance_dir>* \etc\bootstrap.xml** 配置文件。在 **web** 元素中，更新 7.8 中 AMQ 管理控制台所需的 **.war** 文件的名称。

```
<web bind="http://localhost:8161" path="web">
    ...
    <app url="console" war="hawtio.war"/>
    ...
</web>
```

11. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

12. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir*>\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.16.0.redhat-00007 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.8. 将代理实例从 7.8.X 升级到 7.9.X

以下小节描述了如何为不同的操作系统将 7.8.x 代理实例升级到 7.9.x。

重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.8.x 升级到 7.9.x
- 在 Windows 上从 7.8.x 升级到 7.9.x

2.8.1. 在 Linux 上从 7.8.x 升级到 7.9.x

注意

您下载的存档名称可能与以下示例中使用的内容不同。

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。以下示例显示了名为 **amq-broker** 的用户。

```
sudo chown amq-broker:amq-broker amq-broker-7.x.x-bin.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.x.x-bin.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.x.x-bin.zip
```

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.13.0.redhat-00003 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir>* /etc/artemis.profile** 配置文件。

   a. 将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。例如：

   ```
   ARTEMIS_HOME='/opt/redhat/amq-broker-7.x.x-bin'
   ```

   b. 找到 **JAVA_ARGS** 属性。确保 bootstrap 类路径参数引用日志管理器所需的依赖文件版本，如下所示。

   ```
   -Xbootclasspath/a:$ARTEMIS_HOME/lib/wildfly-common-1.5.2.Final-redhat-00002.jar
   ```

9. 编辑 & ***lt;broker_instance_dir>* /etc/bootstrap.xml** 配置文件。在 **web** 元素中，更新 AMQ 管理控制台在 7.9 中所需的 **.war** 文件的名称。

```
<web bind="http://localhost:8161" path="web">
    ...
    <app url="console" war="hawtio.war"/>
    ...
</web>
```

10. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

11. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir>*/log/artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Mes
INFO [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
sage Broker version 2.18.0.redhat-00010 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>* /etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.8.2. 在 Windows 上从 7.8.x 升级到 7.9.x

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 如果代理正在运行，请停止它。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，选择 **Copy**。
   b. 右键点击同一窗口并选择 **粘贴**。
6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*\log\artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.13.0.redhat-00003 [4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 和 ***<broker_instance_dir>*\bin\artemis-service.xml** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 配置文件。确保 **JAVA_ARGS** 环境变量引用日志管理器和依赖文件的正确版本，如下所示。

```
JAVA_ARGS=-Xbootclasspath/%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar
```

9. 编辑 & ***lt;broker_instance_dir>* \bin\artemis-service.xml** 配置文件。确保 bootstrap 类路径启动参数引用日志管理器和依赖文件的正确版本，如下所示。

```
<startargument>-Xbootclasspath/a:%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar</startargument>
```

10. 编辑 & ***lt;broker_instance_dir>* \etc\bootstrap.xml** 配置文件。在 **web** 元素中，更新 AMQ 管理控制台在 7.9 中所需的 **.war** 文件的名称。

```
<web bind="http://localhost:8161" path="web">
    ...
    <app url="console" war="hawtio.war"/>
    ...
</web>
```

11. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

12. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 <*broker_instance_dir*>**\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.18.0.redhat-00010 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.9. 将代理实例从 7.9.X 升级到 7.10.X

下面的小节描述了如何为不同的操作系统将 7.9.x 代理实例升级到 7.10.x。

重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.9.x 升级到 7.10.x
- 在 Windows 上从 7.9.x 升级到 7.10.x

2.9.1. 在 Linux 上从 7.9.x 升级到 7.10.x

注意

您下载的存档名称可能与以下示例中使用的内容不同。

先决条件

- 至少，AMQ Broker 7.11 需要 Java 版本 11 运行。确保每个 AMQ Broker 主机都在运行 Java 版本 11 或更高版本。有关支持的配置的更多信息，请参阅 Red Hat AMQ Broker 7 支持的配置 。
- 如果将 AMQ Broker 7.9 配置为保留数据库中的消息数据，HOLDER_EXPIRATION_TIME 栏的数据类型是节点管理器数据库表中 **的时间戳**。在 AMQ Broker 7.11 中，列的数据类型更改为 **number**。在升级到 AMQ Broker 7.11 之前，您必须丢弃节点管理器表，将其从数据库中删除。丢弃表后，重启升级的代理时，会使用新的模式重新创建它。在共享存储高可用性(HA)配置中，节点管理器表在代理间共享。因此，您必须确保在丢弃表前停止共享表的所有代理。以下示例丢弃名为 **NODE_MANAGER_TABLE** 的节点管理器表：

```
DROP TABLE NODE_MANAGER_TABLE
```

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将存档的所有者改为拥有 AMQ Broker 安装的同一用户，以升级。以下示例显示了名为 **amq-broker** 的用户。

```
sudo chown amq-broker:amq-broker amq-broker-7.x.x-bin.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.x.x-bin.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.x.x-bin.zip
```

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

7. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.18.0.redhat-00010 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

8. 编辑 & ***lt;broker_instance_dir>* /etc/artemis.profile** 配置文件。
   a. 将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。例如：

```
ARTEMIS_HOME='/opt/redhat/amq-broker-7.x.x-bin'
```

9. 编辑 & ***lt;broker_instance_dir>* /etc/bootstrap.xml** 配置文件。
   在 **web** 元素中，更新 AMQ 管理控制台在 7.10 中所需的 **.war** 文件的名称。

```
<web path="web">
   <binding uri="https://localhost:8161"
   ...
   <app url="console" war="hawtio.war"/>
   ...
</web>
```

+ 在 **代理 xmlns** 元素中，将 schema 值从 **"http://activemq.org/schema"** 更改为 **"http://activemq.apache.org/schema** "。

+

```
<broker xmlns="http://activemq.apache.org/schema">
```

1. 编辑 & ***lt;broker_instance_dir>*** **/etc/management.xml** 文件。
   在 **management-context xmlns** 元素中，将 schema 值从 **"http://activemq.org/schema"** 更改为 **"http://activemq.apache.org/schema** "。

   ```
   <management-context xmlns="http://activemq.apache.org/schema">
   ```

2. 启动升级的代理。

   ```
   <broker_instance_dir>/bin/artemis run
   ```

3. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 <***broker_instance_dir>*** **/log/artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

   ```
   INFO [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis
   Mes
   INFO [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
   ...
   sage Broker version 2.21.0.redhat-00025 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-
   54ee759954c4]
   ```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

2.9.2. 在 Windows 上从 7.9.x 升级到 7.10.x

先决条件

- 至少，AMQ Broker 7.11 需要 Java 版本 11 运行。确保每个 AMQ Broker 主机都在运行 Java 版本 11 或更高版本。有关支持的配置的更多信息，请参阅 Red Hat AMQ Broker 7 支持的配置 。
- 如果将 AMQ Broker 7.9 配置为保留数据库中的消息数据，HOLDER_EXPIRATION_TIME 栏的数据类型是节点管理器数据库表中 **的时间戳**。在 AMQ Broker 7.11 中，列的数据类型更改为 **number**。在升级到 AMQ Broker 7.11 之前，您必须丢弃节点管理器表，将其从数据库中删除。丢弃表后，重启升级的代理时，会使用新的模式重新创建它。在共享存储高可用性(HA)配置中，节点管理器表在代理间共享。因此，您必须确保在丢弃表前停止共享表的所有代理。以下示例丢弃名为 **NODE_MANAGER_TABLE** 的节点管理器表：

```
DROP TABLE NODE_MANAGER_TABLE
```

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。
4. 如果代理正在运行，请停止它。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。
   b. 右键点击同一窗口并选择 **粘贴**。
6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*\log\artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.18.0.redhat-00010[4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

7. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 和 ***<broker_instance_dir>*\bin\artemis-service.xml** 配置文件。将 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。

```
ARTEMIS_HOME=<install_dir>
```

8. 编辑 & ***lt;broker_instance_dir>* \etc\artemis.profile.cmd** 配置文件。确保 **JAVA_ARGS** 环境变量引用日志管理器和依赖文件的正确版本，如下所示。

```
JAVA_ARGS=-Xbootclasspath/%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar
```

9. 编辑 & ***lt;broker_instance_dir>* \bin\artemis-service.xml** 配置文件。确保 bootstrap 类路径启动参数引用日志管理器和依赖文件的正确版本，如下所示。

```
<startargument>-Xbootclasspath/a:%ARTEMIS_HOME%\lib\jboss-logmanager-2.1.10.Final-redhat-00001.jar;%ARTEMIS_HOME%\lib\wildfly-common-1.5.2.Final-redhat-00002.jar</startargument>
```

10. 编辑 & ***lt;broker_instance_dir>* \etc\bootstrap.xml** 配置文件。
在 **web** 元素中，更新 AMQ 管理控制台在 7.10 中所需的 **.war** 文件的名称。

```
<web path="web">
    <binding uri="https://localhost:8161"
    ...
    <app url="console" war="hawtio.war"/>
    ...
</web>
```

在 **broker xmlns** 元素中，将 schema 值从 **"http://activemq.org/schema"** 更改为 **"http://activemq.apache.org/schema** "。

```
<broker xmlns="http://activemq.apache.org/schema">
```

11. 编辑 & ***lt;broker_instance_dir>*** **/etc/management.xml** 文件。
在 **management-context xmlns** 元素中，将 schema 值从 **"http://activemq.org/schema"** 更改为 **"http://activemq.apache.org/schema** "。

```
<management-context xmlns="http://activemq.apache.org/schema">
```

12. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

13. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 <***broker_instance_dir***>**\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

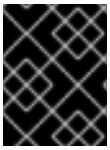
```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.21.0.redhat-00025 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir>*** **\etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

2.10. 将代理实例从 7.10.X 升级到 7.11.X

以下小节描述了如何为不同的操作系统将 7.10.x 代理实例升级到 7.11.x。



重要

从 AMQ Broker 7.1.0 开始，默认只能从本地主机访问 AMQ 管理控制台。要了解如何配置对控制台的远程访问，请参阅配置对 AMQ 管理控制台的本地和远程访问。

- 在 Linux 上从 7.10.x 升级到 7.11.x
- 在 Windows 上从 7.10.x 升级到 7.11.x

2.10.1. 在 Linux 上从 7.10.x 升级到 7.11.x

注意

您下载的存档名称可能与以下示例中使用的内容不同。

先决条件

- 至少，AMQ Broker 7.11 需要 Java 版本 11 运行。确保每个 AMQ Broker 主机都在运行 Java 版本 11 或更高版本。有关支持的配置的更多信息，请参阅 Red Hat AMQ Broker 7 支持的配置 。

流程

1. 从红帽客户门户下载所需的存档。按照 下载 AMQ Broker 归档 中的说明进行操作。
2. 将您下载的存档所有者改为拥有 AMQ Broker 安装的同一用户。以下示例显示了名为 **amq-broker** 的用户。

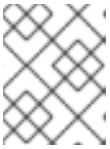
```
sudo chown amq-broker:amq-broker amq-broker-7.x.x-bin.zip
```

3. 将存档移到 AMQ Broker 原始安装过程中创建的目录中。以下示例使用 **/opt/redhat**。

```
sudo mv amq-broker-7.x.x-bin.zip /opt/redhat
```

4. 作为目录所有者，提取压缩存档的内容。在以下示例中，用户 **amq-broker** 使用 **unzip** 命令提取存档。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.x.x-bin.zip
```



注意

存档的内容提取到当前目录中名为 **apache-artemis-2.28.0.redhat-00019** 的目录。

5. 如果代理正在运行，请停止它。

```
<broker_instance_dir>/bin/artemis stop
```

6. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*/log/artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.18.0.redhat-00010 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

7. 通过将代理复制到当前用户的主目录，备份代理的实例目录。

```
cp -r <broker_instance_dir> ~/
```

8. 切换到您提取压缩存档内容的目录。

```
cd /opt/redhat/apache-artemis-2.28.0.redhat-00019/bin
```

9. 运行 **artemis upgrade** 命令，以升级现有的代理。以下示例升级 **/var/opt/amq-broker/mybroker** 目录中的代理实例。

```
./artemis upgrade /var/opt/amq-broker/mybroker
```

artemis upgrade 命令完成以下步骤来升级代理。

- 在您要升级的代理的 broker 实例目录的 **old-config-bkp.*<n>*** 子目录中备份它修改的每个文件。
- 将 < ***broker_instance_dir*> /etc/artemis.profile** 文件中的 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。
- 更新 & ***lt;broker_instance_dir*> bin/artemis** 脚本以使用 Apache Log4j 2 日志记录工具，该实用程序与 AMQ Broker 7.11 捆绑，而不是之前版本中使用的 JBoss Logging 框架。
- 删除 JBoss 使用的现有 ***<broker_instance_dir>* /etc/logging.properties** 文件，并为 Apache Log4j 2 日志记录实用程序创建一个新的 ***<broker_instance_dir>* /etc/log4j2.properties** 文件。

10. 如果在 7.10.x 中启用了 AMQ Broker 中包含的 Prometheus 指标插件，请将插件的类名称从 **org.apache.activemq.artemis.core.server.metrics.plugins.plugins.ArtemisPrometheusMetricsPlugin** 改为 **com.redhat.amq.broker.server.metrics.plugins.ArtemisPrometheusMetricsPlugin**，这是 AMQ Broker 7.11 中插件的新类名称。
    a. 打开 & ***lt;broker_instance_dir&*gt; /etc/broker.xml** 配置文件。
    b. 在 & **lt; metrics> 元素的 <plugin&** gt; 子元素中，将插件类名称更新为 **com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin**。

```
<metrics>
   <plugin class-
name="com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugi
n"/>
</metrics>
```

    c. 保存 **broker.xml** 配置文件。

11. 启动升级的代理。

```
<broker_instance_dir>/bin/artemis run
```

12. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir*>/log/artemis.log** 文件。查找与以下类似的行。请注意代理启动后日志中显示的新版本号。

```
2023-02-08 20:53:50,128 INFO  [org.apache.activemq.artemis.integration.bootstrap]
AMQ101000: Starting ActiveMQ Artemis Server version {upstreamversion}.redhat-{build}
2023-02-08 20:53:51,077 INFO  [org.apache.activemq.artemis.core.server] AMQ221001:
Apache ActiveMQ Artemis Message Broker version {upstreamversion}.redhat-{build} [0.0.0.0,
nodeID=be02a2b2-3e42-11ec-9b8a-4c796e887ecb]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的

位置。在 & ***lt;broker_instance_dir>*** **/etc/artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **etc/** 和 **data/** 目录中。

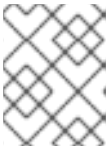
2.10.2. 在 Windows 上从 7.10.x 升级到 7.11.x

先决条件

- 至少，AMQ Broker 7.11 需要 Java 版本 11 运行。确保每个 AMQ Broker 主机都在运行 Java 版本 11 或更高版本。有关支持的配置的更多信息，请参阅 Red Hat AMQ Broker 7 支持的配置 。

流程

1. 按照 下载 AMQ Broker 归档中 的说明下载 AMQ Broker 归档。
2. 使用文件管理器将存档移到您在最后一次安装 AMQ Broker 时创建的文件夹。
3. 提取存档的内容。右键点击 .zip 文件并选择 **Extract All**。



注意

存档的内容提取到当前文件夹中名为 **apache-artemis-2.28.0.redhat-00019** 的文件夹。

4. 如果代理正在运行，请停止它。

```
<broker_instance_dir>\bin\artemis-service.exe stop
```

5. （可选）请注意代理的当前版本。代理停止后，您会在 < ***broker_instance_dir>*****\log\artemis.log** 文件末尾看到类似如下的行。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.18.0.redhat-00010[4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

6. 使用文件管理器备份代理。
   a. 右键点击 < ***broker_instance_dir>*** 文件夹，然后选择 **Copy**。
   b. 右键点击同一窗口并选择 **粘贴**。
7. 切换到您提取压缩存档内容的目录。例如：

```
cd \redhat\amq-broker\apache-artemis-2.28.0.redhat-00019\bin
```

8. 运行 **artemis upgrade** 命令，以升级现有的代理。以下示例升级 **C:\redhat\amq-broker\mybroker** 目录中的代理实例。

```
artemis upgrade C:\redhat\amq-broker\mybroker
```

artemis upgrade 命令完成以下步骤来升级代理。

- 在您要升级的代理的 broker 实例目录的 **old-config-bkp.*<n>*** 子目录中备份它修改的每个文件。
- 将 < ***broker_instance_dir*> \etc\artemis.cmd.profile** 文件中的 **ARTEMIS_HOME** 属性设置为提取存档时创建的新目录。
- 更新 & ***lt;broker_instance_dir*> \bin\artemis.cmd** 脚本以使用 Apache Log4j 2 日志记录工具，该实用程序与 AMQ Broker 7.11 捆绑，而不是之前版本中使用的 JBoss Logging 框架。
- 删除 JBoss 使用的现有 ***<broker_instance_dir>* \etc\logging.properties** 文件，并为 Apache Log4j 2 日志记录实用程序创建一个新的 ***<broker_instance_dir>* \etc\log4j2.properties** 文件。

9. 如果在 7.10.x 中启用了 AMQ Broker 中包含的 Prometheus 指标插件，请将插件的类名称从 **org.apache.activemq.artemis.core.server.metrics.plugins.plugins.ArtemisPrometheusMetricsPlugin** 改为 **com.redhat.amq.broker.server.metrics.plugins.ArtemisPrometheusMetricsPlugin**，这是 7.11 中插件的新类名称。
   a. 打开 & ***lt;broker_instance_dir&*gt; \etc\broker.xml** 配置文件。
   b. 在 & **lt; metrics> 元素的 <plugin&** gt; 子元素中，将插件类名称更新为 **com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin**。

```
<metrics>
   <plugin class-
name="com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugi
n"/>
</metrics>
```

   c. 保存 **broker.xml** 配置文件。

10. 启动升级的代理。

```
<broker_instance_dir>\bin\artemis-service.exe start
```

11. （可选）确认代理正在运行并且版本已更改。启动代理后，打开 < ***broker_instance_dir*>\log\artemis.log** 文件。找到与下面相似的两行。请注意，当代理可用时，日志中出现的新版本号。

```
2023-02-08 20:53:50,128 INFO  [org.apache.activemq.artemis.integration.bootstrap]
AMQ101000: Starting ActiveMQ Artemis Server version {upstreamversion}.redhat-{build}
2023-02-08 20:53:51,077 INFO  [org.apache.activemq.artemis.core.server] AMQ221001:
Apache ActiveMQ Artemis Message Broker version {upstreamversion}.redhat-{build} [0.0.0.0,
nodeID=be02a2b2-3e42-11ec-9b8a-4c796e887ecb]
```

其它资源

- 有关创建代理实例的更多信息，请参阅创建代理实例。
- 现在，您可以将代理实例的配置文件和数据存储在任何自定义目录中，包括代理实例目录之外的位置。在 & ***lt;broker_instance_dir*> \etc\artemis.profile** 文件中，通过在创建代理实例后指定自定义目录的位置来更新 **ARTEMIS_INSTANCE_ETC_URI** 属性。在以前的版本中，这些配置文件和数据只能存储在代理实例目录中的 **\etc** 和 **\data** 目录中。

第 3 章 使用命令行界面

命令行界面(CLI)允许使用交互式终端与消息代理交互。使用 CLI 管理代理操作、配置消息并输入有用的命令。

命令行界面(CLI)允许使用交互式进程将用户和组添加到文件中。

3.1. 启动代理实例

代理实例是一个包含所有配置和运行时数据的目录，如日志和数据文件。运行时数据与唯一代理进程关联。

您可以使用 **artemis** 脚本、Linux 服务或 Windows 服务在前台启动代理。

3.1.1. 启动代理实例

创建代理实例后，您将使用 **artemis run** 命令启动它。

流程

1. 切换到安装期间创建的用户帐户。

```
$ su - amq-broker
```

2. 使用 **artemis run** 命令启动代理实例。

```
$ /var/opt/amq-broker/mybroker/bin/artemis run


        __  __  ____    ____          _
      /\  |  \/  |/ __ \ |  _ \        | |
     /  \ | \  / | |  | | |_) |_ __ ___ | | _____ _ __
    / /\ \| |\/| | |  | | |  _ <| '__/ _ \| |/ / _ \ '__|
   / ____ \| |  | | |__| | | |_) | | | (_) |   <  __/ |
  /_/    \_\_|  |_|\____\_\ |____/|_|  \___/|_|\_\___|_|

Red Hat JBoss AMQ 7.2.1.GA

10:53:43,959 INFO  [org.apache.activemq.artemis.integration.bootstrap] AMQ101000:
Starting ActiveMQ Artemis Server
10:53:44,076 INFO  [org.apache.activemq.artemis.core.server] AMQ221000: live Message
Broker is starting with configuration Broker Configuration
(clustered=false,journalDirectory=./data/journal,bindingsDirectory=./data/bindings,largeMessage
sDirectory=./data/large-messages,pagingDirectory=./data/paging)
10:53:44,099 INFO  [org.apache.activemq.artemis.core.server] AMQ221012: Using AIO
Journal
...
```

代理使用以下信息启动并显示日志输出：

- 事务日志和集群配置的位置。
- 用于消息持久性的日志类型（本例中为 AIO）。
- 可接受客户端连接的 URI。

默认情况下，端口 61616 可以接受来自任何支持的协议（CORE、MQTT、AMQP、STOMP、HORNETQ 和 OPENWIRE）的连接。每个协议也都有单独的端口。

- Web 控制台位于 http://localhost:8161。
- Jolokia 服务(JMX over REST)位于 http://localhost:8161/jolokia。

3.1.2. 将代理作为 Linux 服务启动

如果在 Linux 上安装代理，您可以将其作为服务运行。

流程

1. 在 **/etc/systemd/system/** 目录中创建一个新的 **amq-broker.service** 文件。
2. 将以下文本复制到文件中。
   根据代理实例创建过程中提供的信息修改路径和用户字段。在以下示例中，用户 **amq-broker** 启动安装在 **/var/opt/amq-broker/mybroker/** 目录下的代理服务。

```
[Unit]
Description=AMQ Broker
After=syslog.target network.target

[Service]
ExecStart=/var/opt/amq-broker/mybroker/bin/artemis run
Restart=on-failure
User=amq-broker
Group=amq-broker

# A workaround for Java signal handling
SuccessExitStatus=143

[Install]
WantedBy=multi-user.target
```

3. 打开终端。
4. 使用以下命令启用代理服务：

```
sudo systemctl enable amq-broker
```

5. 使用以下命令运行代理服务：

```
sudo systemctl start amq-broker
```

3.1.3. 将代理作为 Windows 服务启动

如果在 Windows 上安装代理，您可以将其作为服务运行。

流程

1. 打开命令提示符以输入命令
2. 使用以下命令将代理作为服务安装：

```
<broker_instance_dir>\bin\artemis-service.exe install
```

3. 使用以下命令启动该服务：

```
<broker_instance_dir>\bin\artemis-service.exe start
```

4. （可选）卸载该服务：

```
<broker_instance_dir>\bin\artemis-service.exe uninstall
```

3.2. 停止代理实例

手动停止代理实例，或者将代理配置为安全关闭。

3.2.1. 停止 broker 实例

创建独立代理并生成和使用测试消息后，您可以停止代理实例。

此流程手动停止代理，代理强制关闭所有客户端连接。在生产环境中，您应该将代理配置为正常停止，以便正确关闭客户端连接。

流程

- 使用 **artemis stop** 命令停止代理实例：

```
$ /var/opt/amq-broker/mybroker/bin/artemis stop
2018-12-03 14:37:30,630 INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.6.1.amq-720004-redhat-1 [b6c244ef-f1cb-11e8-a2d7-0800271b03bd] stopped, uptime 35 minutes
Server stopped!
```

3.2.2. 正常停止代理实例

在输入 **stop** 命令后，手动关闭会强制断开所有客户端。另外，使用 **启用了 secureful-shutdown 的配置元素将代理配置为安全关闭**。

当将 **secureful-shutdown-enabled** 设置为 **true** 时，在输入 **stop** 命令后不允许新的客户端连接。但是，允许现有连接在关闭过程启动前在客户端上关闭。**graceful-shutdown-enabled** 的默认值为 **false**。

使用 **secureful-shutdown-timeout** 配置元素设置时长（以毫秒为单位），以便客户端在从代理侧强制关闭连接前断开连接。关闭所有连接后，关闭过程将启动。使用 secureful **-shutdown-timeout** 的一个优点是，它可防止客户端连接延迟关闭。**graceful-shutdown-timeout** 的默认值为 **-1，** 这意味着代理无限期等待客户端断开连接。

以下流程演示了如何配置使用超时的安全关闭。

流程

1. 打开配置文件 < ***broker_instance_dir>*** **\etc\broker.xml**。
2. 添加 secure **ful-shutdown-enabled** 配置元素，并将值设为 **true**。

```
<configuration>
```

```
    <core>
        ...
        <graceful-shutdown-enabled>
            true
        </graceful-shutdown-enabled>
        ...
    </core>
</configuration>
```

3. 添加 secure **ful-shutdown-timeout** 配置元素，并以毫秒为单位为超时设置值。在以下示例中，在发出 **stop** 命令后，客户端连接会强制关闭 30 秒(**30000** 毫秒)。

```
<configuration>
    <core>
        ...
        <graceful-shutdown-enabled>
            true
        </graceful-shutdown-enabled>
        <graceful-shutdown-timeout>
            30000
        </graceful-shutdown-timeout>
        ...
    </core>
</configuration>
```

3.3. 通过截获数据包来审核消息

截获进入或退出代理的数据包，以审核数据包或过滤消息。拦截器更改它们拦截器的数据包。这使得拦截器强大，但也有可能存在危险。

开发拦截器以满足您的业务需求。拦截器特定于协议，必须实施适当的接口。

拦截器必须实施 **intercept （）** 方法，它返回一个布尔值。如果值为 **true**，则消息 packet 继续。如果为 **false**，则进程中止，则不会调用其他拦截器，并且消息数据包不会被进一步处理。

3.3.1. 创建拦截器

拦截器可以更改它们拦截器的数据包。您可以创建自己的传入和传出拦截器。所有拦截器都是特定于协议的，分别针对输入或退出服务器的任何数据包调用。这样，您可以创建拦截器来满足审计数据包等业务需求。

拦截器及其依赖项必须放在代理的 Java 类路径中。您可以使用 &lt ***;broker_instance_dir>*** **/lib** 目录，因为它默认为 classpath 的一部分。

以下示例演示了如何创建一个拦截器来检查传递给它的每个数据包的大小。

注意

示例为每个协议实施特定的接口。

流程

1. 实施适当的接口并覆盖其拦截器 **（）** 方法。

    a. 如果使用 AMQP 协议，请实施

org.apache.activemq.artemis.protocol.amqp.broker.AmqpInterceptor 接口。

```
package com.example;

import org.apache.activemq.artemis.protocol.amqp.broker.AMQPMessage;
import org.apache.activemq.artemis.protocol.amqp.broker.AmqpInterceptor;
import org.apache.activemq.artemis.spi.core.protocol.RemotingConnection;

public class MyInterceptor implements AmqpInterceptor
{
  private final int ACCEPTABLE_SIZE = 1024;

  @Override
  public boolean intercept(final AMQPMessage message, RemotingConnection
connection)
  {
    int size = message.getEncodeSize();
    if (size <= ACCEPTABLE_SIZE) {
      System.out.println("This AMQPMessage has an acceptable size.");
      return true;
    }
    return false;
  }
}
```

b. 如果使用核心协议，您的拦截器必须实施 **org.apache.artemis.activemq.api.core.Interceptor** 接口。

```
package com.example;

import org.apache.artemis.activemq.api.core.Interceptor;
import org.apache.activemq.artemis.core.protocol.core.Packet;
import org.apache.activemq.artemis.spi.core.protocol.RemotingConnection;

public class MyInterceptor implements Interceptor
{
  private final int ACCEPTABLE_SIZE = 1024;

  @Override
  boolean intercept(Packet packet, RemotingConnection connection)
  throws ActiveMQException
  {
    int size = packet.getPacketSize();
    if (size <= ACCEPTABLE_SIZE) {
      System.out.println("This Packet has an acceptable size.");
      return true;
    }
    return false;
  }
}
```

c. 如果您使用 mq 协议，请实施 **org.apache.activemq.artemis.core.protocol.mqtt.mqInterceptor** 接口。

```
package com.example;
```

```
import org.apache.activemq.artemis.core.protocol.mqtt.MQTTInterceptor;
import io.netty.handler.codec.mqtt.MqttMessage;
import org.apache.activemq.artemis.spi.core.protocol.RemotingConnection;

public class MyInterceptor implements Interceptor
{
  private final int ACCEPTABLE_SIZE = 1024;

  @Override
  boolean intercept(MqttMessage mqttMessage, RemotingConnection connection)
  throws ActiveMQException
  {
    byte[] msg = (mqttMessage.toString()).getBytes();
    int size = msg.length;
    if (size <= ACCEPTABLE_SIZE) {
      System.out.println("This MqttMessage has an acceptable size.");
      return true;
    }
    return false;
  }
}
```

d. 如果您使用 STOMP 协议，请实施 **org.apache.activemq.artemis.core.protocol.stomp.StompFrameInterceptor** 接口。

```
package com.example;

import org.apache.activemq.artemis.core.protocol.stomp.StompFrameInterceptor;
import org.apache.activemq.artemis.core.protocol.stomp.StompFrame;
import org.apache.activemq.artemis.spi.core.protocol.RemotingConnection;

public class MyInterceptor implements Interceptor
{
  private final int ACCEPTABLE_SIZE = 1024;

  @Override
  boolean intercept(StompFrame stompFrame, RemotingConnection connection)
  throws ActiveMQException
  {
    int size = stompFrame.getEncodedSize();
    if (size <= ACCEPTABLE_SIZE) {
      System.out.println("This StompFrame has an acceptable size.");
      return true;
    }
    return false;
  }
}
```

3.3.2. 将代理配置为使用拦截器

先决条件

- 创建一个拦截器类，并将其（及其依赖项）添加到代理的 Java 类路径中。您可以使用 <*broker_instance_dir*>**/lib** 目录，因为它默认为 classpath 的一部分。

流程

1. Open ***<broker_instance_dir>*/etc/broker.xml**
2. 通过在 <broker ***_instance_dir> /etc/broker*.xml**中添加配置，将代理配置为使用拦截器
   a. 如果拦截器旨在用于传入消息，请将其 **class-name** 添加到 **remoting-incoming-interceptors** 列表中。

```
<configuration>
 <core>
  ...
  <remoting-incoming-interceptors>
    <class-name>org.example.MyIncomingInterceptor</class-name>
  </remoting-incoming-interceptors>
  ...
 </core>
</configuration>
```

   b. 如果拦截器旨在用于传出消息，请将其 **class-name** 添加到 **remoting-outgoing-interceptors** 列表中。

```
<configuration>
 <core>
  ...
  <remoting-outgoing-interceptors>
    <class-name>org.example.MyOutgoingInterceptor</class-name>
  </remoting-outgoing-interceptors>
 </core>
</configuration>
```

3.3.3. 客户端上的拦截器

客户端可以使用拦截器将客户端发送到服务器或者服务器发送到客户端的数据包。如果代理拦截器返回 **false** 值，则不会调用其他拦截器，且客户端不会进一步处理数据包。除非以 **阻塞** 方式发送传出数据包，否则这个过程发生透明。在本例中，一个 **ActiveMQException** 被抛出到调用者。**ActiveMQException** thrown 包含返回 **false** 值的拦截器的名称。

在服务器上，客户端拦截器类及其依赖项必须添加到客户端的 Java 类路径中，才能正确实例化和调用。

3.4. 检查代理和队列的健康状况

AMQ Broker 包含一个命令行工具，可让您在代理拓扑中的代理和队列上执行各种健康检查。

以下示例演示了如何使用 实用程序来运行健康检查。

流程

1. 请参阅代理拓扑中为特定代理（即 *节点*）运行的检查列表。

```
$ <broker_instance_dir>/bin/artemis help check node
```

   您会看到描述可与 **artemis check node** 命令一起使用的一组选项的输出。

```
NAME
        artemis check node - Check a node

SYNOPSIS
        artemis check node [--backup] [--clientID <clientID>]
                [--diskUsage <diskUsage>] [--fail-at-end] [--live]
                [--memoryUsage <memoryUsage>] [--name <name>] [--password <password>]
                [--peers <peers>] [--protocol <protocol>] [--silent]
                [--timeout <timeout>] [--up] [--url <brokerURL>] [--user <user>]
                [--verbose]

OPTIONS
        --backup
            Check that the node has a backup

        --clientID <clientID>
            ClientID to be associated with connection

        --diskUsage <diskUsage>
            Disk usage percentage to check or -1 to use the max-disk-usage

        --fail-at-end
            If a particular module check fails, continue the rest of the checks

        --live
            Check that the node has a live

        --memoryUsage <memoryUsage>
            Memory usage percentage to check

        --name <name>
            Name of the target to check

        --password <password>
            Password used to connect

        --peers <peers>
            Number of peers to check

        --protocol <protocol>
            Protocol used. Valid values are amqp or core. Default=core.

        --silent
            It will disable all the inputs, and it would make a best guess for any required input

        --timeout <timeout>
            Time to wait for the check execution, in milliseconds

        --up
            Check that the node is started, it is executed by default if there are no other checks

        --url <brokerURL>
            URL towards the broker. (default: tcp://localhost:61616)

        --user <user>
            User used to connect
```

```
    --verbose
        Adds more information on the execution
```

2. 例如，检查本地代理的磁盘用量是否低于为代理配置的最大磁盘用量。

```
$ <broker_instance_dir>/bin/artemis check node --url tcp://localhost:61616 --diskUsage -1

Connection brokerURL = tcp://localhost:61616
Running NodeCheck
Checking that the disk usage is less then the max-disk-usage ... success
Checks run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.065 sec - NodeCheck
```

在上例中，为 **--diskUsage** 选项指定 **-1** 值意味着，实用程序会根据为代理配置 **的最大** 磁盘用量检查磁盘使用情况。代理的最大磁盘用量是使用 **broker.xml** 配置文件中的 **max-disk-usage** 参数进行配置。为 **max-disk-usage** 指定的值代表代理允许使用的可用物理磁盘空间的百分比。

3. 请参阅您可以在代理拓扑中为特定队列运行的检查列表。

```
$ <broker_instance_dir>/bin/artemis help check queue
```

您会看到描述可与 **artemis check queue** 命令一起使用的一组选项的输出。

```
NAME
    artemis check queue - Check a queue

SYNOPSIS
    artemis check queue [--browse <browse>] [--clientID <clientID>]
        [--consume <consume>] [--fail-at-end] [--name <name>]
        [--password <password>] [--produce <produce>] [--protocol <protocol>]
        [--silent] [--timeout <timeout>] [--up] [--url <brokerURL>]
        [--user <user>] [--verbose]

OPTIONS
    --browse <browse>
        Number of the messages to browse or -1 to check that the queue is
        browsable

    --clientID <clientID>
        ClientID to be associated with connection

    --consume <consume>
        Number of the messages to consume or -1 to check that the queue is consumable

    --fail-at-end
        If a particular module check fails, continue the rest of the checks

    --name <name>
        Name of the target to check

    --password <password>
        Password used to connect

    --produce <produce>
        Number of the messages to produce
```

```
        --protocol <protocol>
           Protocol used. Valid values are amqp or core. Default=core.

        --silent
           It will disable all the inputs, and it would make a best guess for any required input

        --timeout <timeout>
           Time to wait for the check execution, in milliseconds

        --up
           Check that the queue exists and is not paused, it is executed by default if there are no
other checks

        --url <brokerURL>
           URL towards the broker. (default: tcp://localhost:61616)

        --user <user>
           User used to connect

        --verbose
           Adds more information on the execution
```

4. 该工具可以使用单个命令执行多个选项。例如，要检查本地代理上默认 **helloworld** 队列上的 1000 个消息，请使用以下命令：

```
$ <broker_instance_dir>/bin/artemis check queue --name helloworld --produce 1000 --
browse 1000 --consume 1000

Connection brokerURL = tcp://localhost:61616
Running QueueCheck
Checking that a producer can send 1000 messages to the queue helloworld ... success
Checking that a consumer can browse 1000 messages from the queue helloworld ... success
Checking that a consumer can consume 1000 messages from the queue helloworld ...
success
Checks run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.882 sec - QueueCheck
```

在上例中，观察在运行队列检查时没有指定代理 URL。如果您没有明确指定 URL，则工具会使用默认值 **tcp://localhost:61616**。

3.5. 命令行工具

AMQ Broker 包括一组命令行界面(CLI)工具，因此您可以管理消息传递日志。下表列出了每个工具的名称及其对应的描述。

工具	描述
address	Address 工具组(create/delete/update/show) （示例 **./artemis address create**）。
浏览器	浏览实例上的消息。
consumer	消耗实例上的消息。

工具	描述
data	输出有关日志记录的报告，并压缩数据。
decode	从编码中导入内部日志格式。
编码	显示编码为 String 的日志的内部格式。
exp	使用特殊和独立的 XML 格式导出消息数据。
帮助	显示帮助信息。
imp	使用 **exp** 提供的输出，将日志导入到正在运行的代理中。
kill	终止通过 --allow-kill 启动的代理实例。
mask	屏蔽密码并将其打印出。
perf-journal	计算您应该与当前数据文件夹一起使用的 journal-buffer 超时。
queue	队列工具组(create/delete/update/stat) （示例 **./artemis queue create**）。
run	运行代理实例。
stop	停止代理实例。
user	基于文件的默认用户 managament (add/rm/list/reset) (example **./artemis user list**)

如需每个工具的可用命令的完整列表，请使用 **help** 参数，后跟工具的名称。例如，在下面的示例中，CLI 输出列出了用户在用户输入命令 **./artemis help data** 后可供 data 工具使用的所有命令。

```
$ ./artemis help data

NAME
        artemis data - data tools group
        (print|imp|exp|encode|decode|compact) (example ./artemis data print)

SYNOPSIS
        artemis data
        artemis data compact [--broker <brokerConfig>] [--verbose]
                [--paging <paging>] [--journal <journal>]
                [--large-messages <largeMessges>] [--bindings <binding>]
        artemis data decode [--broker <brokerConfig>] [--suffix <suffix>]
                [--verbose] [--paging <paging>] [--prefix <prefix>] [--file-size <size>]
                [--directory <directory>] --input <input> [--journal <journal>]
                [--large-messages <largeMessges>] [--bindings <binding>]
        artemis data encode [--directory <directory>] [--broker <brokerConfig>]
                [--suffix <suffix>] [--verbose] [--paging <paging>] [--prefix <prefix>]
                [--file-size <size>] [--journal <journal>]
                [--large-messages <largeMessges>] [--bindings <binding>]
```

```
        artemis data exp [--broker <brokerConfig>] [--verbose]
              [--paging <paging>] [--journal <journal>]
              [--large-messages <largeMessges>] [--bindings <binding>]
        artemis data imp [--host <host>] [--verbose] [--port <port>]
              [--password <password>] [--transaction] --input <input> [--user <user>]
        artemis data print [--broker <brokerConfig>] [--verbose]
              [--paging <paging>] [--journal <journal>]
              [--large-messages <largeMessges>] [--bindings <binding>]

COMMANDS
        With no arguments, Display help information

        print
           Print data records information (WARNING: don't use while a
           production server is running)

        ...
```

您可以使用 **help** 参数来了解有关如何执行每个命令的更多信息。例如，CLI 在用户输入 **./artemis help data print** 后列出了 **data print** 命令的更多信息。

```
$ ./artemis help data print

NAME
        artemis data print - Print data records information (WARNING: don't use
        while a production server is running)

SYNOPSIS
        artemis data print [--bindings <binding>] [--journal <journal>]
              [--paging <paging>]

OPTIONS
        --bindings <binding>
           The folder used for bindings (default ../data/bindings)

        --journal <journal>
           The folder used for messages journal (default ../data/journal)

        --paging <paging>
           The folder used for paging (default ../data/paging)
```

第 4 章 使用 AMQ 管理控制台

AMQ 管理控制台是 AMQ Broker 安装中包含的 Web 控制台，可让您使用 Web 浏览器管理 AMQ Broker。

AMQ 管理控制台基于 hawtio。

4.1. 概述

AMQ Broker 是一个功能齐全的、面向消息的中间件代理。它提供专用队列行为、消息持久性和站。它支持多种协议和客户端语言，供您使用许多应用程序资产。

AMQ Broker 的主要功能允许您：

- 监控 AMQ 代理和客户端
  - 查看拓扑
  - 查看网络健康状况
- 使用以下方法管理 AMQ 代理：
  - AMQ 管理控制台
  - 命令行界面(CLI)
  - 管理 API

AMQ 管理控制台支持的 Web 浏览器是 Firefox 和 Chrome。有关支持的浏览器版本的更多信息，请参阅 AMQ 7 支持的配置。

4.2. 配置 AMQ 管理控制台的本地和远程访问权限

本节中的步骤演示了如何配置 AMQ 管理控制台的本地和远程访问。

远程访问控制台可以采用两种形式之一：

- 在本地代理的控制台会话中，您可以使用 **Connect** 选项卡连接到另一个远程代理
- 从远程主机中，您可以使用本地代理的外部 IP 地址连接到本地代理的控制台

先决条件

- 您必须至少升级到 AMQ Broker 7.1.0。作为此升级的一部分，名为 **jolokia-access.xml** 的 access-management 配置文件被添加到代理实例中。有关升级的更多信息，请参阅将 代理实例从 7.0.x 升级到 7.1.0。

流程

1. 打开 & ***lt;broker_instance_dir>*** **/etc/bootstrap.xml** 文件。
2. 在 **web** 元素中，观察 web 端口默认仅绑定到 **localhost**。

```
<web path="web">
  <binding uri="http://localhost:8161">
```

```
    <app url="redhat-branding" war="redhat-branding.war"/>
    <app url="artemis-plugin" war="artemis-plugin.war"/>
    <app url="dispatch-hawtio-console" war="dispatch-hawtio-console.war"/>
    <app url="console" war="console.war"/>
  </binding>
</web>
```

3. 要从远程主机为本地代理启用到控制台的连接，请将 Web 端口绑定改为可网络可访问的接口。例如：

```
<web path="web">
  <binding uri="http://0.0.0.0:8161">
```

在上例中，通过指定 **0.0.0.0**，您可以将 Web 端口绑定到本地代理 **中的所有** 接口。

4. 保存 **bootstrap.xml** 文件。

5. 打开 & ***lt;broker_instance_dir&*gt; /etc/jolokia-access.xml** 文件。

6. 在 & **lt;cors** > （即 *Cross-Origin Resource Sharing*）元素中，为您要允许访问控制台的每个 HTTP 原始请求标头添加一个 **allow-origin** 条目。例如：

```
<cors>
  <allow-origin>*://localhost*</allow-origin>
  <allow-origin>*://192.168.0.49*</allow-origin>
  <allow-origin>*://192.168.0.51*</allow-origin>
  <!-- Check for the proper origin on the server side, too -->
  <strict-checking/>
</cors>
```

在前面的配置中，您可以指定允许以下连接：

- 从本地主机（即本地代理实例的主机机器）连接到控制台。
  - 根据您为安全连接配置了控制台，**在连接** 请求中允许指定第一个星号(DSL)通配符字符。
  - **第二个星号通配符字符允许主机上任何端口用于连接。**

- **使用本地代理的外部可访问的 IP 地址，从远程主机连接到本地代理的控制台。在这种情况下，本地代理的外部可访问的 IP 地址是 192.168.0.49。**

- **从在另一个远程代理上打开的控制台会话中连接到本地代理的连接。在本例中，远程代理的 IP 地址为 192.168.0.51。**

7. **保存 jolokia-access.xml 文件。**

8. 打开 & *lt;broker_instance_dir&gt*; /etc/artemis.profile 文件。

9. 要在控制台中启用 Connect 选项卡，请将 Dhawtio.disableProxy 参数的值设置为 false。

```
-Dhawtio.disableProxy=false
```

重要

建议您从控制台启用远程连接（即，当控制台公开给安全网络时，将 Dhawtio.disableProxy 参数的值设置为 false）。

10. 将新参数 Dhawtio.proxyWhitelist 添加至 Java 系统参数的 JAVA_ARGS 列表中。作为以逗号分隔的列表，为您要从本地代理（即，使用本地代理上运行的控制台会话中的 Connect 选项卡）连接的远程代理指定 IP 地址。例如：

```
-Dhawtio.proxyWhitelist=192.168.0.51
```

根据上述配置，您可以使用本地代理控制台会话中的 Connect 选项卡连接到另一个，IP 地址为 192.168.0.51 的远程代理。

11. 保存 aretmis.profile 文件。

其他资源

- 要了解如何访问控制台，请参阅 第 4.3 节 “访问 AMQ 管理控制台”。
- 有关以下内容的更多信息：
  - 跨 Origin Resource Sharing，请参阅 W3C Recommendations。
  - Jolokia 安全性，请参阅 Jolokia 协议。
  - 保护到控制台的连接，请参阅 第 4.4.3 节 “保护对 AMQ 管理控制台的网络访问”。

4.3. 访问 AMQ 管理控制台

本节中的步骤演示了如何：

- 从本地代理打开 AMQ 管理控制台
- 从本地代理的控制台会话中连接到其他代理
- 使用本地代理的外部 IP 地址从远程主机打开本地代理的控制台实例

先决条件

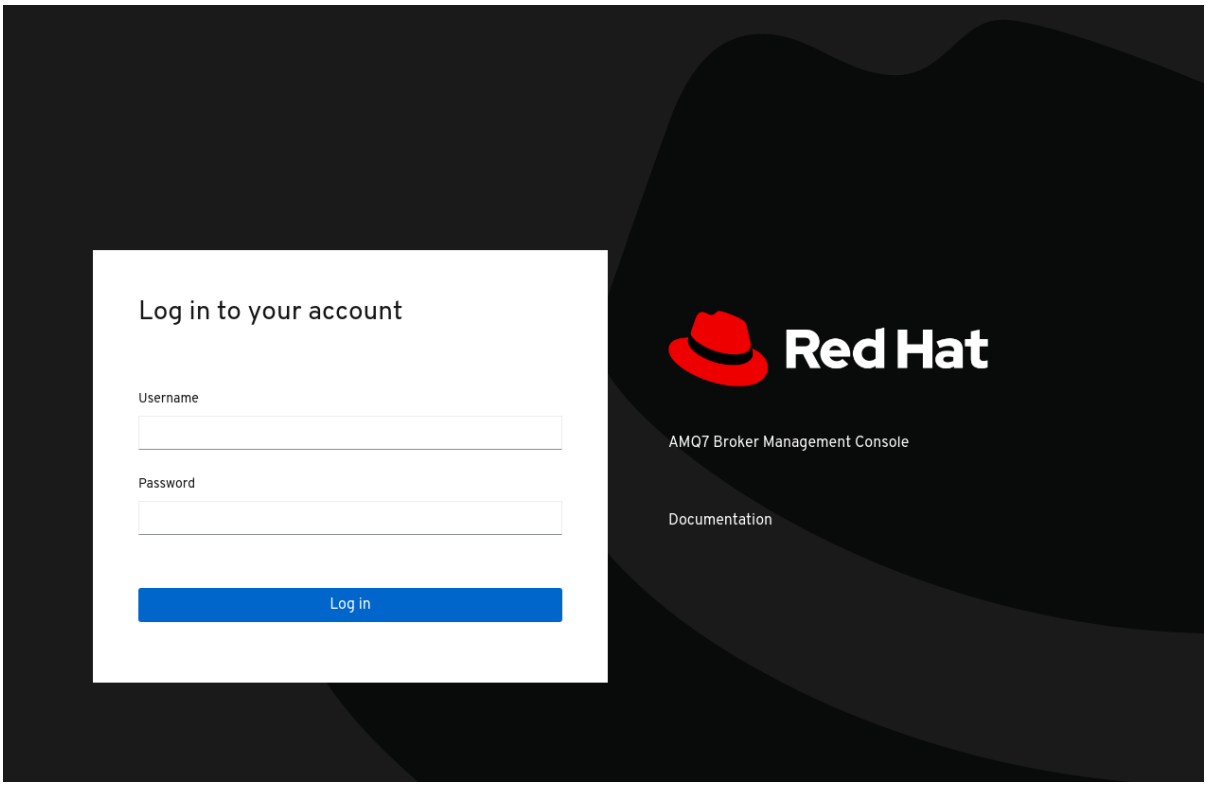
- 您必须已配置了本地和远程访问控制台。更多信息请参阅 第 4.2 节 “配置 AMQ 管理控制台的本地和远程访问权限”。

流程

1. 在 Web 浏览器中，导航到本地代理的控制台地址。

   控制台地址为 http:// *<host:port>* /console/login。如果您使用默认地址，请导航到 http://localhost:8161/console/login。否则，请使用为 *< broker_instance_dir>* /etc/bootstrap.xml 配置文件中的 web 元素的 bind 属性定义的主机和端口值。

图 4.1. 控制台登录页面



2. 使用您在创建代理时创建的默认用户名和密码登录到 AMQ 管理控制台。

3. 要连接到另一个代理，请从本地代理的控制台会话连接到远程代理：

   a. 在左侧菜单中，点 Connect 选项卡。

   b. 在主窗格中，在 Remote 选项卡中点 Add connection 按钮。

   c. 在 Add Connection 对话框中指定以下详情：

   名称

   远程连接的名称，如 my_other_broker。

   Scheme

   用于远程连接的协议。为非安全连接选择 http，或者为安全连接选择 https。

   主机

远程代理的 IP 地址。您必须已为这个远程代理配置了控制台访问权限。

端口

用于远程连接的本地代理上的端口。指定为 *< broker_instance_dir>* /etc/bootstrap.xml 配置文件中的 web 元素的 bind 属性定义的端口值。默认值为 8161。

路径

用于控制台访问的路径。指定 console/jolokia。

d. 要测试连接，请点击 Test Connection 按钮。

如果连接测试成功，点 Add 按钮。如果连接测试失败，请根据需要检查并修改连接详情。再次测试连接。

e. 在 Remote 页面中，对于您添加的连接，点 Connect 按钮。

为远程代理上的 console 实例打开一个新的 Web 浏览器标签页。

f. 在 Log In 对话框中，输入远程代理的用户名和密码。点 Log In。

远程代理的控制台实例将打开。

4. 要从远程主机连接到本地代理的控制台，请在 Web 浏览器中为本地代理指定 Jolokia 端点。此端点包含您在配置远程控制台访问时为本地代理指定的外部可访问的 IP 地址。例如：

```
http://192.168.0.49/console/jolokia
```

4.4. 配置 AMQ 管理控制台

配置用户访问和请求对代理上资源的访问权限。

4.4.1. 使用红帽单点登录保护 AMQ 管理控制台

先决条件

- **Red Hat Single Sign-On 7.4**

流程

1. **配置 Red Hat Single Sign-On：**

   a. **导航到您要用来保护 AMQ 管理控制台的 Red Hat Single Sign-On 中的域。Red Hat Single Sign-On 中的每个域都包括一个名为 Broker 的客户端。此客户端与 AMQ 无关。**

   b. **在 Red Hat Single Sign-On 中创建一个新客户端，如 artemis-console。**

   c. **进入客户端设置页面并设置：**

      - **有效的重定向 URI 到 AMQ 管理控制台 URL，后跟 192.168.1.0/24，例如：**

        ```
        https://broker.example.com:8161/console/*
        ```

      - **Web Origins 与 Valid Redirect URI 的值相同。Red Hat Single Sign-On 允许您输入 +，这表示允许的 CORS 源包含 Valid Redirect URI 的值。**

   d. **为客户端创建一个角色，如 guest。**

   e. **确保需要访问 AMQ 管理控制台的所有用户都被分配了上述角色，例如使用 Red Hat Single Sign-On 组。**

2. **配置 AMQ Broker 实例：**

   a. **将以下内容添加到 < *broker-instance-dir&gt*; /instances/broker0/etc/login.config 文件中，将 AMQ 管理控制台配置为使用 Red Hat Single Sign-On：**

      ```
      console {
        org.keycloak.adapters.jaas.BearerTokenLoginModule required
          keycloak-config-file="${artemis.instance}/etc/keycloak-bearer-token.json"
          role-principal-
      ```

```
    class=org.apache.activemq.artemis.spi.core.security.jaas.RolePrincipal
        ;
    };
```

添加此配置设置 JAAS 主体，以及来自红帽单点登录的 bearer 令牌的要求。与 Red Hat Single Sign-On 的连接在 keycloak-bearer-token.json 文件中定义，如下一步所述。

b. 创建包含以下内容的 &lt ;*broker-instance-dir*> /etc/keycloak-bearer-token.json 文件，以指定与用于 bearer 令牌交换的 Red Hat Single Sign-On 的连接：

```
{
 "realm": "<realm-name>",
 "resource": "<client-name>",
 "auth-server-url": "<RHSSO-URL>/auth",
 "principal-attribute": "preferred_username",
 "use-resource-role-mappings": true,
 "ssl-required": "external",
 "confidential-port": 0
}
```

<realm-name>

Red Hat Single Sign-On 中的域名

<client-name>

Red Hat Single Sign-On 中的客户端名称

<RHSSO-URL>

Red Hat Single Sign-On 的 URL

c. 创建包含以下内容的 &lt ;*broker-instance-dir*> /etc/keycloak-js-token.json 文件，以指定 Red Hat Single Sign-On 身份验证端点：

```
{
 "realm": "<realm-name>",
 "clientId": "<client-name>",
 "url": "<RHSSO-URL>/auth"
}
```

d. 通过编辑 < *broker-instance-dir*> /etc/broker.xml 文件来配置 安全设置。

例如，要允许具有 amq 角色的用户使用消息并允许具有 guest 角色的用户发送消息，请添加以下内容：

```
<security-setting match="Info">
  <permission roles="amq" type="createDurableQueue"/>
  <permission roles="amq" type="deleteDurableQueue"/>
  <permission roles="amq" type="createNonDurableQueue"/>
  <permission roles="amq" type="deleteNonDurableQueue"/>
  <permission roles="guest" type="send"/>
  <permission roles="amq" type="consume"/>
</security-setting>
```

3. 运行 **AMQ Broker** 实例并验证 **AMQ** 管理控制台配置。

4.4.2. 设置用户对 AMQ 管理控制台的访问权限

您可以使用代理登录凭证访问 AMQ 管理控制台。下表提供了有关添加额外代理用户访问 AMQ 管理控制台的不同方法的信息：

身份验证方法	描述
客户端验证	启用匿名访问。在此配置中，任何在没有凭证或带有错误凭证的情况下连接的用户都会被自动进行身份验证，并为特定的用户和角色进行身份验证。 如需更多信息， *请参阅配置 AMQ Broker 中的 配置客户机访问。*
基本用户和密码身份验证	对于每个用户，您必须定义一个用户名和密码，并分配安全角色。用户只能使用这些凭证登录 AMQ 管理控制台。 如需更多信息， *请参阅配置 AMQ Broker 中的 配置基本用户和密码身份验证。*
LDAP 身份验证	用户通过针对存储在中央 X.500 目录服务器中的用户数据检查凭据来进行身份验证和授权。 如需更多信息， *请参阅配置 AMQ Broker 中的配置 LDAP 以验证客户端。*

4.4.3. 保护对 AMQ 管理控制台的网络访问

要在通过 WAN 或互联网访问控制台时保护 AMQ 管理控制台，请使用 SSL 指定网络访问使用 https 而不是 http。

先决条件

以下命令位于 < *broker_instance_dir*>/etc/ 目录中：

- Java 密钥存储
- Java 信任存储（只有在您需要客户端身份验证时才需要）

流程

1. 打开 & *lt;broker_instance_dir>* /etc/bootstrap.xml 文件。
2. 在 & lt;web > 元素中，添加以下属性：

```
<web path="web">
    <binding uri="https://0.0.0.0:8161" keyStorePath="<path_to_keystore>"
keyStorePassword="<password>"
    clientAuth="<true/false>" trustStorePath="<path_to_truststore>" trustStorePassword="
<password>">
    </binding>
</web>
```

bind

对于到控制台的安全连接，请将 URI 方案更改为 https。

keyStorePath

密钥存储文件的路径。例如：

```
keyStorePath="<broker_instance_dir>/etc/keystore.jks"
```

keyStorePassword

密钥存储密码。此密码可以被加密。

clientAuth

指定是否需要客户端身份验证。默认值为 false。

trustStorePath

信任存储文件的路径。只有在 clientAuth 设为 true 时，才需要定义此属性。

trustStorePassword

信任存储密码。此密码可以被加密。

其他资源

- 有关在代理配置文件中加密密码的更多信息，包括 bootstrap.xml，请参阅 配置文件中的加密密码。

4.4.4. 配置 AMQ 管理控制台以使用基于证书的身份验证

您可以配置 AMQ 管理控制台，以使用证书而不是密码来验证用户。

流程

1. 从可信证书颁发机构获取代理和客户端的证书，或生成自签名证书。如果要生成自签名证书，请完成以下步骤：

   a. 为代理生成自签名证书。

   ```
   $ keytool -storetype pkcs12 -keystore broker-keystore.p12 -storepass securepass
   -keypass securepass -alias client -genkey -keyalg "RSA" -keysize 2048 -dname
   "CN=ActiveMQ Broker, OU=Artemis, O=ActiveMQ, L=AMQ, S=AMQ, C=AMQ" -ext
   bc=ca:false -ext eku=cA
   ```

   b. 从代理密钥存储导出证书，使其可以与客户端共享。

   ```
   $ keytool -storetype pkcs12 -keystore broker-keystore.p12 -storepass securepass
   -alias client -exportcert -rfc > broker.crt
   ```

   c. 在客户端上，将代理证书导入到客户端信任存储中。

   ```
   $ keytool -storetype pkcs12 -keystore client-truststore.p12 -storepass securepass
   -keypass securepass -importcert -alias client-ca -file broker.crt -noprompt
   ```

   d. 在客户端上，为客户端生成自签名证书。

   ```
   $ keytool -storetype pkcs12 -keystore client-keystore.p12 -storepass securepass -
   keypass securepass -alias client -genkey -keyalg "RSA" -keysize 2048 -dname
   "CN=ActiveMQ Client, OU=Artemis, O=ActiveMQ, L=AMQ, S=AMQ, C=AMQ" -ext
   ```

```
bc=ca:false -ext eku=cA
```

e. 将客户端证书从客户端密钥存储导出到文件，以便可以将其添加到代理信任存储中。

```
$ keytool -storetype pkcs12 -keystore client-keystore.p12 -storepass securepass -
alias client -exportcert -rfc > client.crt
```

f. 将客户端证书导入到代理信任存储中。

```
$ keytool -storetype pkcs12 -keystore client-truststore.p12 -storepass securepass
-keypass securepass -importcert -alias client-ca -file client.crt -noprompt
```

注意

在代理机器上，确保密钥存储和信任存储文件位于代理可访问的位置。

2. 在 & lt;broker_instance_dir>/etc/bootstrap.xml 文件中，更新 Web 配置来为代理控制台启用 HTTPS 协议和客户端身份验证。例如：

```
...
<web path="web">
   <binding uri="https://localhost:8161" keyStorePath="${artemis.instance}/etc/server-
keystore.p12" keyStorePassword="password"
   clientAuth="true" trustStorePath="${artemis.instance}/etc/client-truststore.p12"
trustStorePassword="password">
   ...
   </binding>
</web>
...
```

binding uri

指定 https 协议来启用 SSL 并添加主机名和端口。

keystorePath

安装代理证书的密钥存储的路径。

keystorePassword

安装代理证书的密钥存储的密码。

clientAuth

设置为 **true**，将代理配置为要求每个客户端在客户端尝试连接到代理控制台时提供证书。

trustStorePath

如果客户端使用自签名证书，请指定安装客户端证书的信任存储的路径。

trustStorePassword

如果客户端使用自签名证书，请指定安装客户端证书的信任存储的密码。

备注。只有在客户端使用自签名证书时，才需要配置 **trustStorePath** 和 **trustStorePassword** 属性。

3. 从每个客户端证书获取 Subject *Distinguished Names* (DN)，以便您可以在每个客户端证书和代理用户之间创建映射。

a. 将客户端密钥存储文件中的每个客户端证书导出到临时文件中。例如：

```
keytool -export -file <file_name> -alias broker-localhost -keystore broker.ks -storepass
<password>
```

b. 显示导出的证书的内容：

```
keytool -printcert -file <file_name>
```

输出结果类似如下：

```
Owner: CN=AMQ Client, OU=Artemis, O=AMQ, L=AMQ, ST=AMQ, C=AMQ
Issuer: CN=AMQ Client, OU=Artemis, O=AMQ, L=AMQ, ST=AMQ, C=AMQ
Serial number: 51461f5d
Valid from: Sun Apr 17 12:20:14 IST 2022 until: Sat Jul 16 12:20:14 IST 2022
Certificate fingerprints:
  SHA1: EC:94:13:16:04:93:57:4F:FD:CA:AD:D8:32:68:A4:13:CC:EA:7A:67
  SHA256:
85:7F:D5:4A:69:80:3B:5B:86:27:99:A7:97:B8:E4:E8:7D:6F:D1:53:08:D8:7A:BA:A7:0A:7A:
96:F3:6B:98:81
```

Owner 条目是 **Subject DN**。用于输入 **Subject DN** 的格式取决于您的平台。以上字符串也可以表示：

```
Owner: `CN=localhost,\ OU=broker,\ O=Unknown,\ L=Unknown,\ ST=Unknown,\
C=Unknown`
```

4. 为代理控制台启用基于证书的验证。

a. 打开 & *lt;broker_instance_dir&*gt; /etc/login.config 配置文件。添加证书登录模块并引用用户和角色属性文件。例如：

```
activemq {

org.apache.activemq.artemis.spi.core.security.jaas.TextFileCertificateLoginModule
      debug=true
      org.apache.activemq.jaas.textfiledn.user="artemis-users.properties"
      org.apache.activemq.jaas.textfiledn.role="artemis-roles.properties";
};
```

org.apache.activemq.artemis.spi.core.security.jaas.TextFileCertificateLoginModule

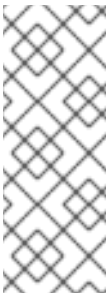
实施类。

org.apache.activemq.jaas.textfiledn.user

指定相对于包含登录配置文件的目录的用户属性文件的位置。

org.apache.activemq.jaas.textfiledn.role

指定将用户映射到为登录模块实施定义的角色的属性文件。



注意

如果您更改了 < *broker_instance_dir*>/etc/login.config 文件中的证书登录模块配置的默认名称，您必须更新 < broker_instance_dir>/etc/artemis.profile 文件中的 -dhawtio. realm 参数的值，以匹配新名称。默认名称为 activemq。

b. 打开 & lt;broker_instance_dir>/etc/artemis-users.properties 文件。通过将从每个客户端证书获取的 Subject DNS 添加到代理用户，创建客户端证书和代理用户之间的映射。例如：

```
user1=CN=user1,O=Progress,C=US
user2=CN=user2,O=Progress,C=US
```

在本例中，user1 代理用户映射到具有 Subject Distinguished Name 为 CN=user1,O=Progress,C=US Subject DN 的客户端证书。在客户端证书和代理用户之间创建映射后，代理可以使用证书来验证用户。

c.

打开 & lt;broker_instance_dir>/etc/artemis-roles.properties 文件。通过将用户权限添加到为 < broker_instance_dir>/etc/artemis.profile 文件中 HAWTIO_ROLE 变量指定的角色中，授予用户登录控制台的权限。HAWTIO_ROLE 变量的默认值为 amq。例如：

```
amq=user1, user2
```

5.

为 HTTPS 协议配置以下推荐的安全属性。

a.

打开 & lt;broker_instance_dir>/etc/artemis.profile 文件。

b.

设置 hawtio.http.strictTransportSecurity 属性，仅允许 HTTPS 请求到 AMQ 管理控制台，并将任何 HTTP 请求转换为 HTTPS。例如：

```
hawtio.http.strictTransportSecurity = max-age=31536000; includeSubDomains;
preload
```

c.

设置 hawtio.http.publicKeyPins 属性，以指示 Web 浏览器将特定的加密密钥与 AMQ 管理控制台关联，以减少使用伪证书的"man-in-the-middle"攻击的风险。例如：

```
hawtio.http.publicKeyPins = pin-sha256="..."; max-age=5184000;
includeSubDomains
```

4.4.5. 配置 AMQ 管理控制台以处理 X 转发的标头

如果对 AMQ 管理控制台的请求通过代理服务器路由，您可以配置 AMQ Broker 嵌入式 Web 服务器（托管 AMQ 管理控制台）来处理 X-Forwarded 标头。通过处理 X-Forwarded 标头，AMQ 管理控制台可以接收当涉及请求路径时更改或丢失的标头信息。例如，代理可以使用 HTTPS 公开 AMQ 管理控制台，而使用 HTTP 的 AMQ 管理控制台可以从 X-Forwarded 标头中识别浏览器和代理之间的连接使用 HTTPS 并切换到 HTTPS 来提供浏览器请求。

流程

1.

打开 & *lt;broker_instance_dir>* /etc/bootstrap.xml 文件。

2.

在 & lt;web > 元素中，使用 org.eclipse.jetty.server.ForwardedRequestCustomizer 的值添加 customizer 属性。例如：

```
<web path="web" customizer="org.eclipse.jetty.server.ForwardedRequestCustomizer">
..
</web>
```

3. 保存 bootstrap.xml 文件。

4. 输入以下命令启动或重启代理：

- On Linux: *<broker_instance_dir>*/bin/artemis run
- 在 Windows 上： *<broker_instance_dir>* \bin\artemis-service.exe start

4.5. 使用 AMQ 管理控制台管理代理

您可以使用 AMQ 管理控制台查看有关正在运行的代理的信息，并管理以下资源：

- 入站网络连接（接受者）
- addresses
- 队列

4.5.1. 查看有关代理的详情

要查看代理配置方式，请在左侧菜单中点击 Artemis。在文件夹树中，默认选择本地代理。

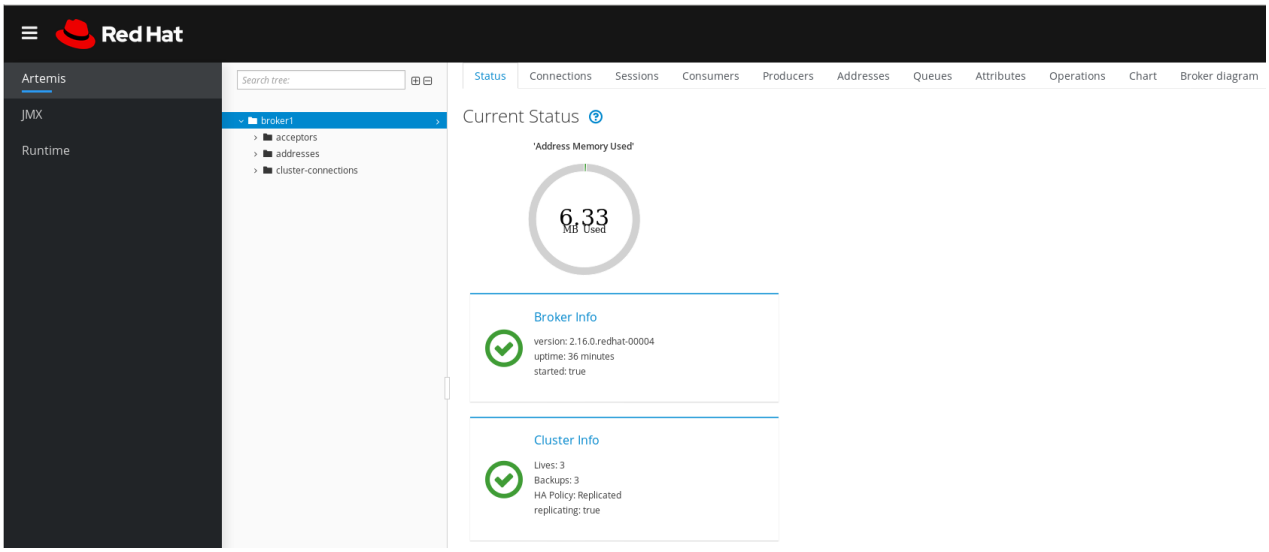
在主窗格中，有以下标签页：

状态

显示代理当前状态的信息，如运行时间和集群信息。另外还显示代理当前使用的地址内存量。图

中显示这个值作为 global-max-size 配置参数的比例。

图 4.2. Status 标签页



连接

显示有关代理连接的信息，包括客户端、集群和桥接连接。

会话

显示有关当前在代理上打开的所有会话的信息。

消费者

显示有关当前在代理上打开的所有消费者的信息。

producers

显示有关当前在代理上打开的制作者的信息。

addresses

显示代理上地址的信息。这包括内部地址，如 storage-and-forward 地址。

队列

显示代理上队列的信息。这包括内部队列，如 store-and-forward 队列。

属性

显示代理上配置的属性的详细信息。

操作

显示您可以从控制台在代理上执行的 JMX 操作。当您点操作时，会打开一个对话框，供您为操作指定参数值。

Chart

显示代理上配置的属性的实时数据。您可以编辑 chart 以指定 chart 中包含的属性。

代理图

显示集群拓扑图。这包括集群中的所有代理，以及本地代理中的任何地址和队列。

4.5.2. 查看代理图

您可以查看拓扑中的所有 AMQ Broker 资源图，包括代理（活跃和备份代理）、生产者和消费者、地址和队列。

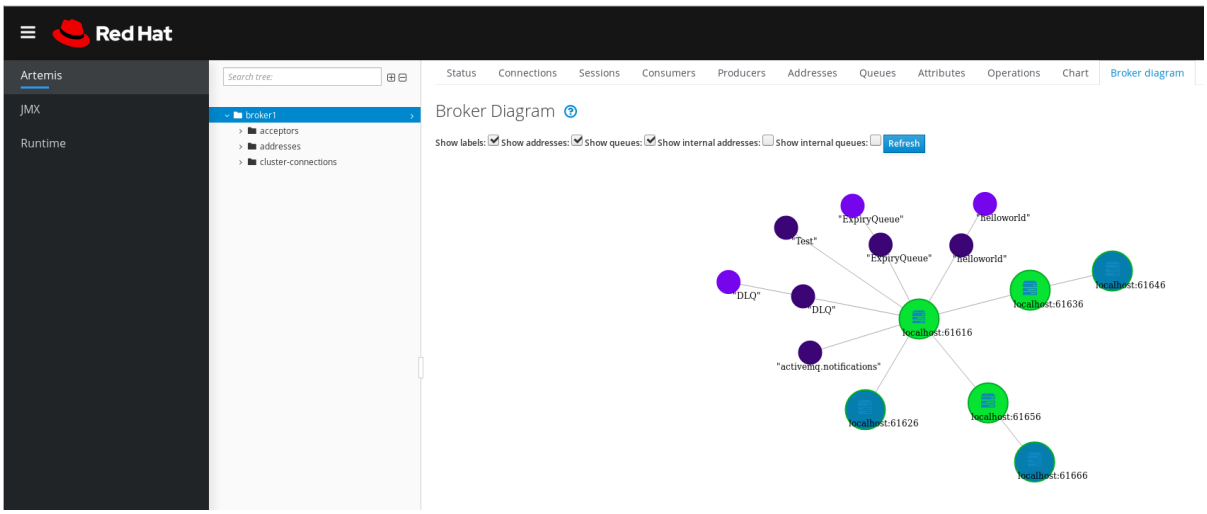
流程

1. 在左侧菜单中，单击 **Artemis**。

2. 在主窗格中，点 **Broker** 图 选项卡。

   控制台显示集群拓扑图。这包括集群中的所有代理以及本地代理中的任何地址和队列，如图所示。

   图 **4.3. 代理图** 标签页

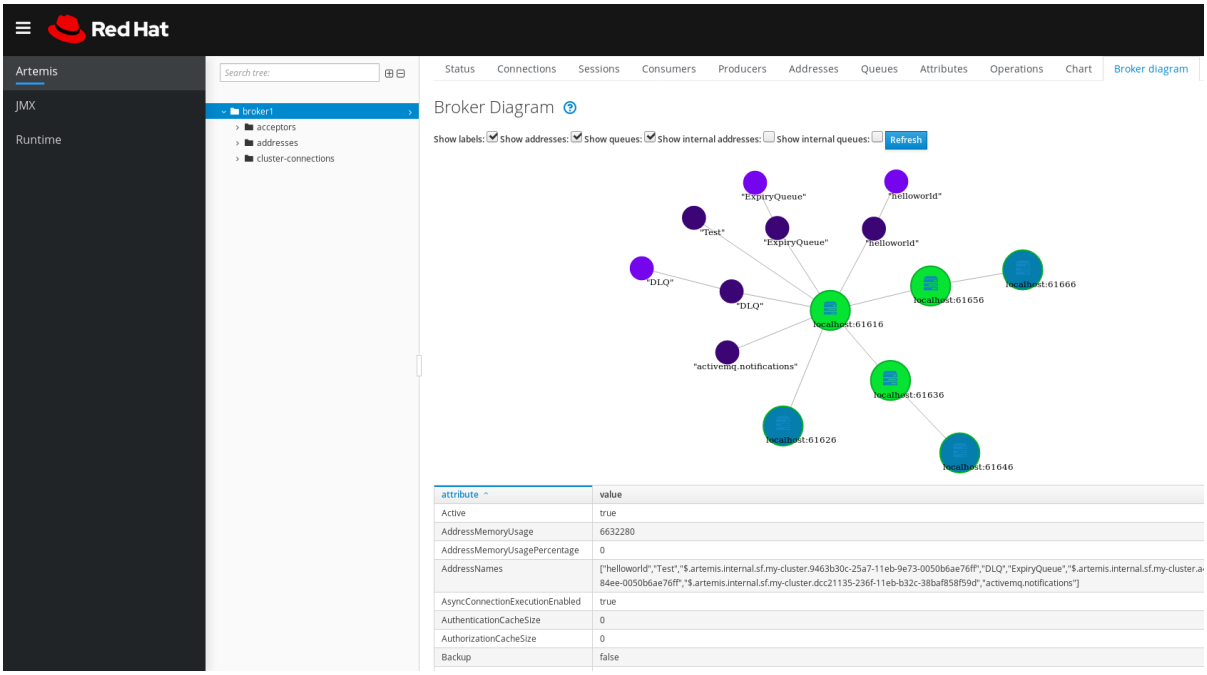


3. 要更改图中显示哪些项目，请使用图顶部的复选框。单击 **Refresh**。

4.

要显示本地代理或连接的地址或队列的属性，请点击图中的节点。例如，下图显示包含本地代理属性的示意图。

图 4.4. 代理图 标签页，包括属性



4.5.3. 查看接受者

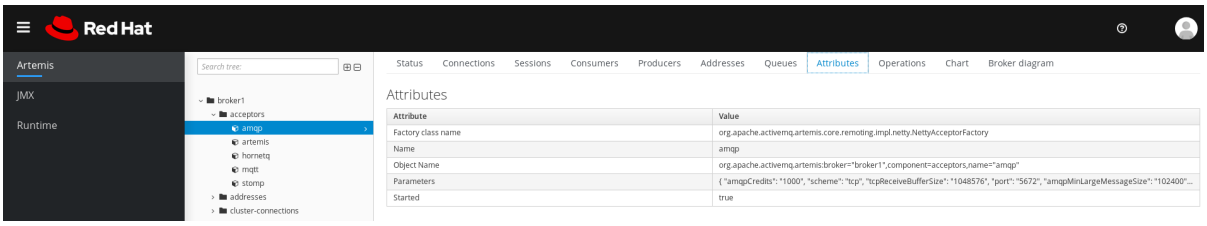
您可以查看为代理配置的 acceptors 的详情。

流程

1. 在左侧菜单中，单击 **Artemis**。
2. 在文件夹树中，单击 **acceptors**。
3. 要查看有关如何配置接受者的详细信息，请点 **acceptor**。

控制台显示在 **Attributes** 选项卡上的对应属性，如图所示。

图 4.5. AMQP acceptor 属性



4. 要查看属性的完整详情，请点属性。此时会打开一个额外的窗口来显示详情。

4.5.4. 管理地址和队列

一个地址代表消息传递端点。在配置中，会指定一个唯一名称的典型地址。

队列与地址关联。每个地址可以有多个队列。传入的消息与地址匹配后，根据配置的路由类型，消息将发送到一个或多个队列。队列可以被配置为自动创建和删除。

4.5.4.1. 创建地址

典型的地址被授予唯一的名称、零个或者多个队列，以及路由类型。

路由类型决定了如何将消息发送到与地址关联的队列。地址可以通过两种不同的路由类型进行配置。

如果您希望消息路由到...	使用此路由类型...
匹配地址中的单个队列，以点对点的方式。	anycast
匹配地址中的每个队列都以发布方式发布。	多播

您可以创建和配置地址和队列，然后在不再使用时删除它们。

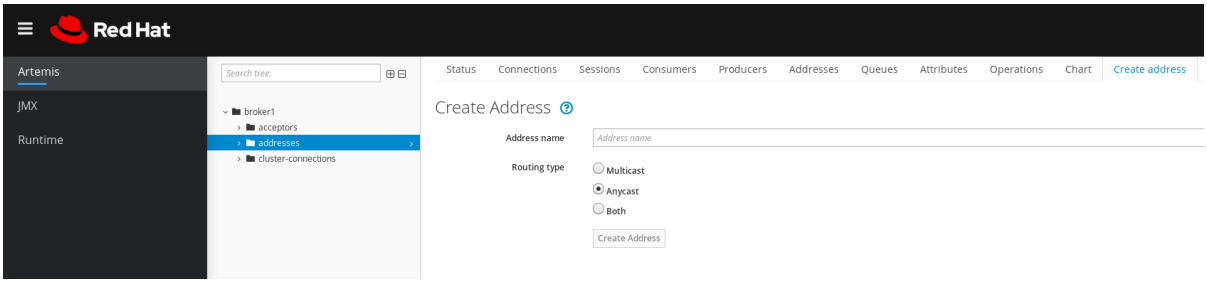
流程

1. 在左侧菜单中，单击 **Artemis**。
2. 在文件夹树中，单击 **地址**。

3. 在主窗格中，点 **Create address** 选项卡。

   此时会出现一个页面，供您创建地址，如图所示。

   图 **4.6.** 创建地址页面



4. 完成以下字段：

   地址名称

   地址的路由名称。

   路由类型

   选择以下选项之一：

   - **多播** ：发送到地址的消息将以发布订阅的方式分发到所有订阅者。
   - **anycast** ：发送到此地址的消息将仅以点到点的方式分发到一个订阅者。
   - **两者** ：允许您为每个地址定义多个路由类型。这通常会产生反模式，我们不推荐使用。

   注意

   如果一个地址同时使用两个路由类型，且客户端没有显示首选项，代理会默认使用 **anycast** 路由类型。一个例外是客户端使用 **MQTT** 协议时。在这种情况下，默认路由类型是 **multicast**。

5. 单击 **Create Address**。

4.5.4.2. 发送消息到地址

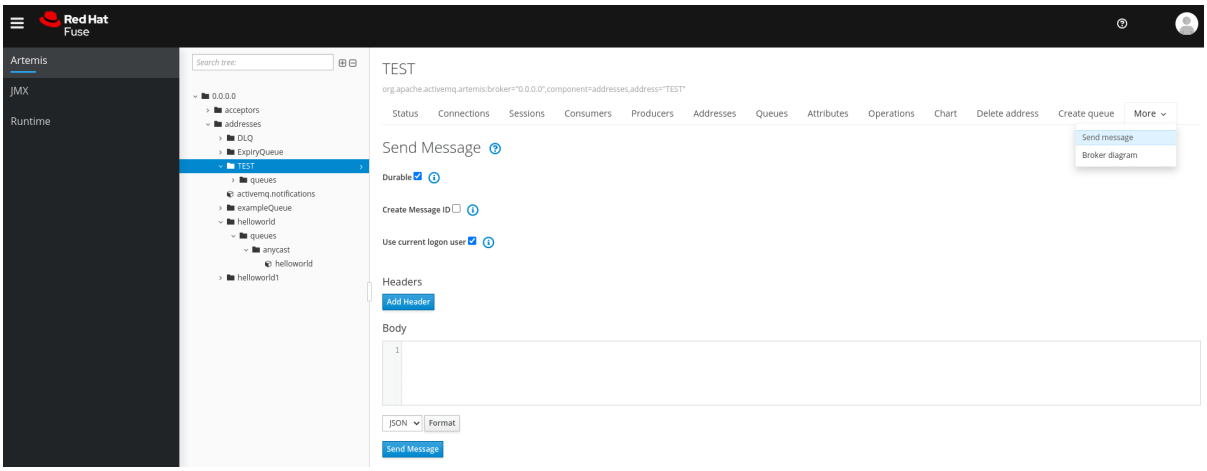
以下流程演示了如何使用控制台向地址发送消息。

流程

1. 在左侧菜单中，单击 **Artemis**。
2. 在文件夹树中，选择一个地址。
3. 在主窗格的导航栏中，点 **More** → **Send message**。

   此时会出现一个页面，供您创建信息，如图所示。

图 4.7. 发送消息页面



4. 默认情况下，使用您用来登录到 AMQ 管理控制台的凭证发送消息。如果要使用不同的凭证，请清除 **Use current logon user** 复选框，并在 **Username** 和 **Password** 字段中指定值，它们会在清除复选框后显示。
5. 如有必要，点 **Add Header** 按钮来添加消息标头信息。
6. 输入消息正文。
7. 在 **Format** 下拉菜单中选择消息正文格式的选项，然后单击 **Format**。消息正文使用您选择的格式的人类可读样式格式化。

8. 单击 **Send message**。

   发送消息。

9. 若要发送附加消息，可更改您输入的任何信息，然后单击 **Send message**。

4.5.4.3. 创建队列

队列在生成者和消费者之间提供频道。

先决条件

- 要绑定队列的地址必须存在。要了解如何使用控制台创建地址，请参阅 第 4.5.4.1 节 “创建地址”。

流程

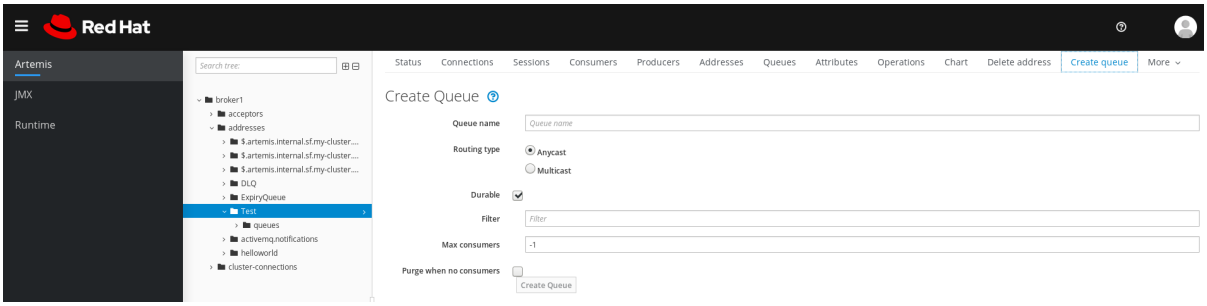
1. 在左侧菜单中，单击 **Artemis**。

2. 在文件夹树中，选择您要将队列绑定到的地址。

3. 在主窗格中，单击 **Create queue** 选项卡。

   此时会出现一个页面，供您创建队列，如图所示。

图 4.8. 创建 Queue 页面



4. 完成以下字段：

队列名称

队列的唯一名称。

路由类型

选择以下选项之一：

- 多播 ：发送到父地址的消息将分发到绑定到地址的所有队列。
- anycast ：只有一个绑定到父地址的队列将会收到消息的副本。消息将在绑定到地址的所有队列中平均分配。

durable

如果您选择这个选项，则队列及其消息将持久。

Filter

连接到代理时使用的用户名。

最大消费者

在给定时间可以访问队列的最大消费者数。

没有消费者时清除

如果选中，则没有用户连接时将清除队列。

5. 单击 **Create Queue**。

4.5.4.4. 检查队列的状态

chart 提供代理中队列状态的实时视图。

流程

1. 在左侧菜单中，单击 **Artemis**。

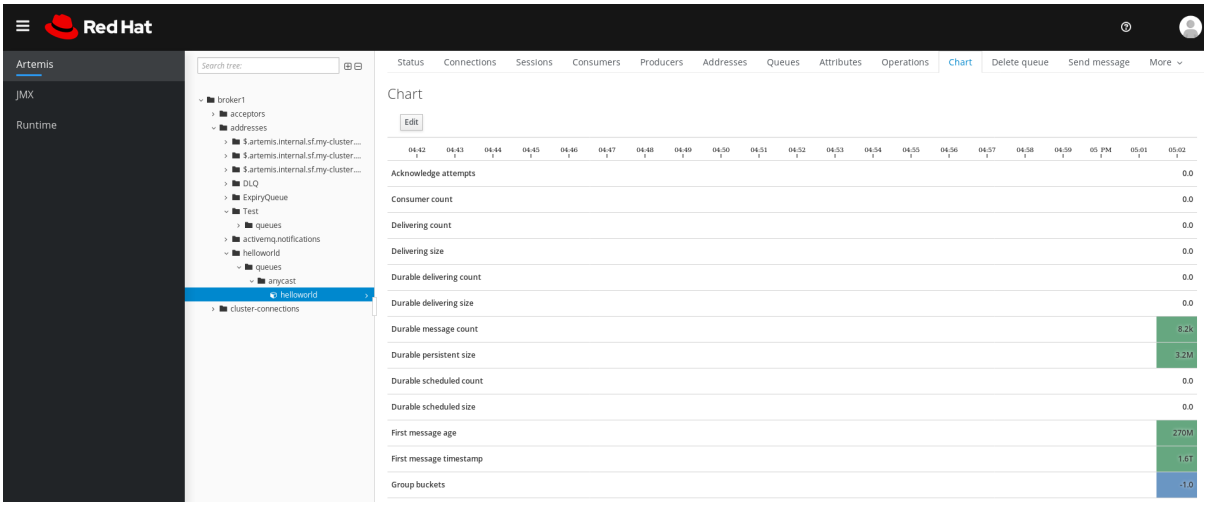
2.

2. 在文件夹树中，导航到队列。

3. 在主窗格中，点 Chart 选项卡。

   控制台显示一个图表，显示所有队列属性的实时数据。

图 4.9. 队列的 Chart 标签页



注意

要查看地址上多个队列的图表，请选择包含队列的 anycast 或 multicast 文件夹。

4. 如有必要，为 chart 选择不同的条件：

   a. 在主窗格中，单击 Edit。

   b. 在 Attributes 列表中，选择您要包含在图表中的一个或多个属性。要选择多个属性，请按并存放 Ctrl 键并选择每个属性。

   c. 点 View Chart 按钮。Chart 根据您选择的属性更新。

4.5.4.5. 浏览队列

浏览队列会显示队列中的所有消息。您还可以过滤并排序列表以查找特定消息。

流程

1. 在左侧菜单中，单击 **Artemis**。

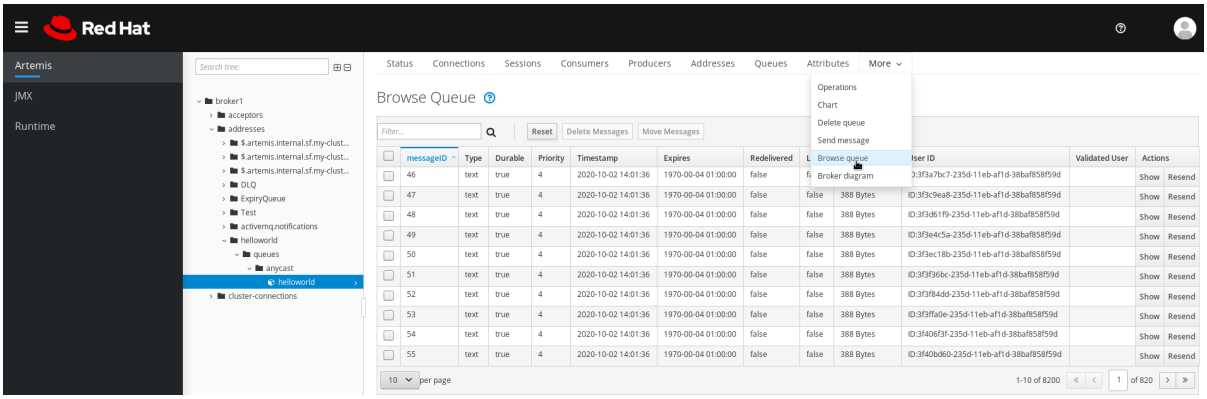
2. 在文件夹树中，导航到队列。

   队列位于它们绑定到的地址中。

3. 在主窗格的导航栏中，点 **More** → **Browse queue**。

   此时会显示队列中的消息。默认情况下会显示前 200 个消息。

图 4.10. 浏览队列页面



4. 要浏览特定消息或一组信息，请执行以下操作之一：

to...	这些以下操作
过滤消息列表	在 **Filter...** 文本字段中输入过滤器条件。点搜索（即，放大）图标。
排序消息列表	在消息列表中，点列标头。要降序对消息进行排序，请单击标头第二次。

5. 要查看消息的内容，请单击 **Show** 按钮。

您可以查看消息标头、属性和正文。

4.5.4.6. 发送消息到队列

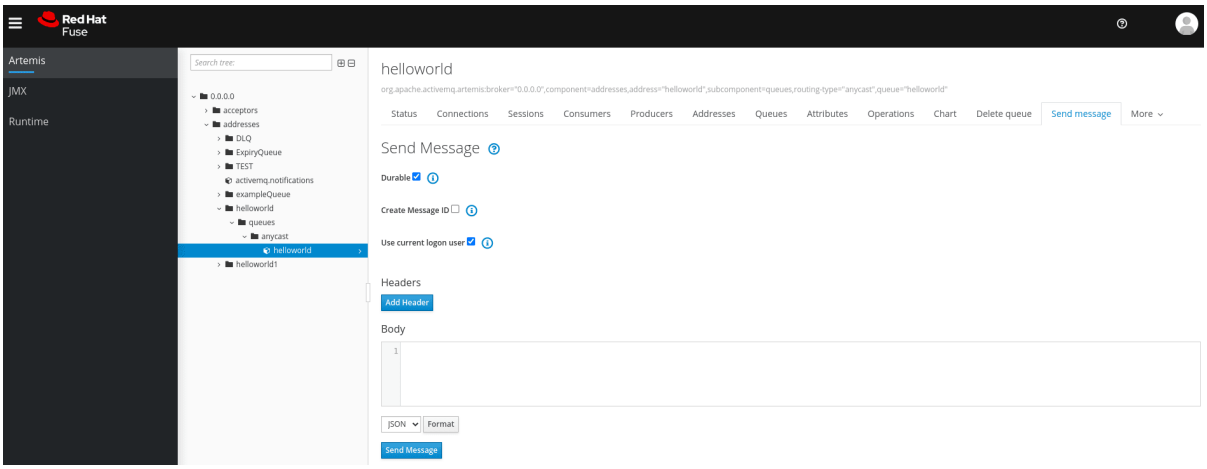
创建队列后，您可以向其发送一条消息。以下流程概述了向现有队列发送消息所需的步骤。

流程

1. 在左侧菜单中，单击 **Artemis**。
2. 在文件夹树中，导航到队列。
3. 在主窗格中，单击 **Send message** 选项卡。

   此时会出现用于编写消息的页面。

图 4.11. 为队列发送消息页面



4. 默认情况下，使用您用来登录到 AMQ 管理控制台的凭证发送消息。如果要使用不同的凭证，请清除 **Use current logon user** 复选框，并在 **Username** 和 **Password** 字段中指定值，它们会在清除复选框后显示。
5. 如有必要，点 **Add Header** 按钮来添加消息标头信息。
6. 输入消息正文。

7. 在 Format 下拉菜单中选择消息正文格式的选项，然后单击 Format。消息正文使用您选择的格式的人类可读样式格式化。

8. 单击 Send message。发送消息。

9. 若要发送附加消息，可更改您输入的任何信息，然后单击 Send message。

4.5.4.7. 将消息重新发送到队列

您可以重新发送之前发送的消息。

流程

1. 浏览您要重新发送的消息。

2. 单击您要重新发送的消息旁边的复选框。

3. 点 Resend 按钮。此时会显示消息。

4. 根据需要更新 消息标头和正文，然后单击 Send message。

4.5.4.8. 将消息移动到不同的队列

您可以将队列中的一个或多个消息移到不同的队列中。

流程

1. 浏览您要移动的消息。

2. 点击您要移动的每个消息旁边的复选框。

3. 在导航栏中，点 Move Messages。

此时会出现确认对话框。

4. 从下拉菜单中，选择要将消息移到的队列的名称。点 Move。

4.5.4.9. 删除消息或队列

您可以删除队列或从队列中清除所有消息。

流程

1. 浏览您要删除的队列，或清除。

2. 执行以下操作之一：

to...	这些以下操作
从队列中删除消息	1. 点击您要删除的每个消息旁边的复选框。 2. 点击 Delete 按钮。
从队列中清除所有消息	1. 在主窗格中的导航栏中，单击 Delete queue。 2. 点 Purge Queue 按钮。
删除队列	1. 在主窗格中的导航栏中，单击 Delete queue。 2. 单击 Delete Queue 按钮。

第 5 章 监控代理运行时指标

安装 AMQ Broker 时，安装中包含 Prometheus 指标插件。Prometheus 是一个用于监控大型、可扩展的系统的软件，并在延长期间存储历史运行时数据。您必须修改代理配置以启用插件。启用后，插件会为代理收集运行时指标，并将其导出为 Prometheus 格式。然后，您可以使用 Prometheus 查看指标。您还可以使用图形化工具（如 Grafana ）来配置更多数据的高级视觉化。

注意

Prometheus metrics 插件允许您以 Prometheus 格式 收集和导出代理指标。但是，红帽 不提供 安装或配置 Prometheus 本身的支持，也不提供视觉化工具，如 Grafana。如果您需要支持安装、配置或运行 Prometheus 或 Grafana，请访问产品网站以获取社区支持和文档等资源。

除了 Prometheus 插件收集的代理指标外，您可以修改代理配置来捕获与代理的主机 Java 虚拟机 (JVM)相关的标准指标集合。特别是，您可以捕获 Garbage Collection (GC)、内存和线程的 JVM 指标。

后续描述的部分：

- Prometheus 插件导出的指标
- 如何启用 Prometheus 插件
- 如何配置代理来收集 JVM 指标

5.1. 指标概述

要监控代理实例的健康和性能，您可以使用 AMQ Broker 的 Prometheus 插件来监控和存储代理运行时指标。AMQ Broker Prometheus 插件将代理运行时指标导出到 Prometheus 格式，可让您使用 Prometheus 本身来视觉化并在数据上运行查询。

您还可以使用图形化工具（如 Grafana ）为 Prometheus 插件收集的指标配置更高级的视觉化和仪表板。

插件导出到 Prometheus 格式的指标如下所述。

代理指标

artemis_address_memory_usage

此代理上所有地址使用的字节数用于内存消息。

artemis_address_memory_usage_percentage

此代理上的所有地址使用的内存作为 global-max-size 参数的百分比。

artemis_connection_count

连接到此代理的客户端数。

artemis_total_connection_count

自启动此代理后连接到此代理的客户端数量。

地址指标

artemis_routed_message_count

路由到一个或多个队列绑定的消息数。

artemis_unrouted_message_count

*没有*路由到任何队列绑定的消息数。

队列指标

artemis_consumer_count

消耗来自给定队列的消息的客户端数量。

artemis_delivering_durable_message_count

给定队列当前提供给消费者的持久消息数。

artemis_delivering_durable_persistent_size

给定队列当前提供给消费者的持久持久性消息大小。

artemis_delivering_message_count

指定队列当前提供给消费者的消息数。

artemis_delivering_persistent_size

给定队列当前提供给消费者的持久消息大小。

artemis_durable_message_count

当前给定队列中的持久消息数。这包括已调度、分页和发送消息。

artemis_durable_persistent_size

当前给定队列中的持久性消息大小。这包括已调度、分页和发送消息。

artemis_messages_acknowledged

从队列创建以来从给定队列确认的消息数。

artemis_messages_added

创建队列以来添加到给定队列的消息数。

artemis_message_count

当前给定队列中的消息数。这包括已调度、分页和发送消息。

artemis_messages_killed

从队列创建以来从给定队列中删除的消息数。当消息超过配置的最大发送尝试次数时，代理会终止消息。

artemis_messages_expired

创建队列后从给定队列已过期的消息数。

artemis_persistent_size

当前在给定队列中的所有消息(durable 和 non-durable)的持久性大小。这包括已调度、分页和发送消息。

artemis_scheduled_durable_message_count

给定队列中可调度的、可调度的消息数。

artemis_scheduled_durable_persistent_size

给定队列中的持久化、可调度的消息的持久性大小。

artemis_scheduled_message_count

给定队列中调度的消息数量。

artemis_scheduled_persistent_size

给定队列中调度消息的持久性大小。

对于以上未列出的高级别代理指标，您可以通过聚合较低级别的指标来计算这些指标。例如，若要计算总消息数，您可以从代理部署中的所有队列聚合 artemis_message_count 指标。

对于 AMQ Broker 的内部部署，托管代理的 Java 虚拟机(JVM)的指标也会导出到 Prometheus 格式。这不适用于在 OpenShift Container Platform 上部署 AMQ Broker。

5.2. 为 AMQ BROKER 启用 PROMETHEUS 指标插件

安装 AMQ Broker 时，安装中包含 Prometheus 指标插件。虽然插件已经配置为使用，但在代理配置中启用插件。启用后，插件会为代理收集运行时指标，并将其导出为 Prometheus 格式。

以下流程演示了如何为 AMQ Broker 启用 Prometheus 插件。

流程

1. 打开 & *lt;broker_instance_dir&*gt; /etc/broker.xml 配置文件。

2. 在代理配置中启用 Prometheus 插件。使用 & lt; plugin> 子元素添加 <metrics > 元素，如下所示。

```
<metrics>
   <plugin class-
name="com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin"
/>
</metrics>
```

3. 保存 broker.xml 配置文件。metrics 插件开始以 Prometheus 格式收集代理运行时指标。

5.3. 配置代理以收集 JVM 指标

以下流程演示了如何配置代理来为 Garbage Collection (GC)、内存和线程收集 Java 虚拟机(JVM)指标。

先决条件

- 您之前已在代理配置中启用 Prometheus metrics 插件。更多信息请参阅 第 5.2 节 “为 AMQ Broker 启用 Prometheus 指标插件”。

流程

1. 打开 & *lt;broker_instance_dir*> /etc/broker.xml 配置文件。

2. 在启用 Prometheus 指标插件时 添加到配置的 <metrics> 元素中，指定代理是否为 Garbage Collection (GC)、内存和线程收集 JVM 指标。例如：

```
<metrics>
   <jvm-gc>true</jvm-gc>
   <jvm-memory>true</jvm-memory>
   <jvm-threads>true</jvm-threads>
   <plugin class-
name="com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin"
/>
</metrics>
```

注意

如果您没有在配置中显式添加 jvm-memory 参数并指定值，则代理会使用默认值 true。这意味着代理默认导出 JVM 内存指标。jvm-gc 和 jvm-threads 参数的默认值为 false。

3. 保存 broker.xml 配置文件。代理开始收集您启用的 JVM 指标。这些指标也导出到 Prometheus 格式。

5.4. 为特定地址禁用指标集合

当您为 AMQ Broker 配置指标插件（如 Prometheus 指标插件）时，指标集合会被默认启用。但是，在特定地址 *或一组* 地址的 address-setting 配置元素中，您可以明确禁用指标集合。

以下流程演示了如何为特定地址 *或一组* 地址禁用指标集合。

流程

1. 打开 & *lt;broker_instance_dir&*gt; /etc/broker.xml 配置文件。

2. 在匹配地址 *或一组* 地址的 address-setting 项中，添加 enable-metrics 参数，并将参数的值设置为 false。例如，以下配置为名为 orders 的地址禁用指标集合。

```
<configuration>
 <core>
  ...
  <address-settings>
   <address-setting match="orders">
    ...
    <enable-metrics>false</enable-metrics>
    ...
   </address-setting>
  </address-settings>
  ...
 </core>
</configuration>
```

5.5. 使用 PROMETHEUS 访问代理运行时数据

先决条件

- 要查询和视觉化 Prometheus 插件收集的代理运行时数据，您需要安装 Prometheus。如需更多信息，请参阅 Prometheus 文档中的安装 Prometheus。

流程

1. 在 Prometheus 安装目录中，打开 prometheus.yml 配置文件。

2. 在配置文件的 static_configs 部分中，将 targets 元素改为 localhost:8161。这个位置是代理运行其 Web 服务器的位置。默认情况下，/metrics 附加到此主机名中，形成存储在代理 web 服务器上的指标的完整路径。

3. 要查看 Prometheus 插件收集的代理运行时指标，请在 Web 浏览器中打开 localhost:8161/metrics。

在生成的网页中，您会看到插件收集的指标的当前值，具体取决于您在代理中配置的队列和地址。如果您在 JVM 中有多个正在运行的代理实例，您会看到每个代理的指标。

4. 从 Prometheus 安装目录中，运行 Prometheus。

```
$ ./prometheus
```

当 Prometheus 启动时，shell 输出包括以下行：

```
component=web, msg="Start listening for connections" address=0.0.0.0:9090
```

上面的行表示 Prometheus 正在侦听端口 9090 上的 HTTP 流量。

5. 要访问 Prometheus Web 控制台，请在 Web 浏览器中打开 127.0.0.1:9090。

6. 在 Prometheus web 控制台中，您可以使用 Expression 字段创建代理数据的查询。您创建的查询基于 Prometheus 查询语言 PromQL。可在查询中插入的代理指标位于 Insert 指标 下拉列表中。

例如，假设您要随着时间的推移查询 DLQ 队列上的消息数。在本例中，从指标下拉列表中选择 artemis_message_count。通过指定 DLQ 队列名称和地址来完成查询。这个示例查询如下所示。

```
artemis_message_count{address="DLQ", queue="DLQ"}
```

对于更高级的视觉化，您可以使用正则表达式来创建覆盖多个指标的复杂查询，例如：或者，您可以对多个指标执行数学操作，如聚合它们。有关创建 Prometheus 查询的更多信息，请参阅 Prometheus 文档中的查询 Prometheus。

第 6 章 使用管理 API

AMQ Broker 有一个广泛的管理 API，可用于修改代理的配置、创建新资源（如地址和队列），检查这些资源（例如，当前在队列中保存的消息）并与它们交互（例如，从队列中删除消息）。

另外，客户端可以使用管理 API 管理代理并订阅管理通知。

6.1. 使用管理 API 管理 AMQ BROKER 的方法

使用管理 API 管理代理的方法有两种：

- 使用 JMX the- theJMX 是管理 Java 应用程序的标准方法
- 使用 JMS API\":\"- themanagement 操作使用 JMS 消息和 AMQ JMS 客户端发送到代理

虽然管理代理的方法有两种，但每个 API 支持相同的功能。如果可以使用 JMX 管理资源，也可以使用 JMS 消息和 AMQ JMS 客户端来实现相同的结果。

此选择取决于您的特定要求、应用程序设置和环境。无论您调用管理操作的方式是什么，管理 API 都相同。

对于每个受管资源，有一个 Java 接口，用于描述可以针对这类资源调用的 Java 接口。代理在 org.apache.activemq.artemis.api.core.management 软件包中公开其受管资源。调用管理操作的方式取决于 JMX 消息或 JMS 消息以及 AMQ JMS 客户端是否使用。

注意

有些管理操作需要 filter 参数来选择受操作影响的消息。传递 null 或空字符串意味着将对 *所有消息执行管理操作*。

6.2. 使用 JMX 管理 AMQ BROKER

您可以使用 Java 管理扩展(JMX)来管理代理。管理 API 由代理使用 MBeans 接口公开。代理将其资源注册到域 org.apache.activemq。

例如，管理名为 exampleQueue 的队列的 ObjectName 是：

```
org.apache.activemq.artemis:broker="__BROKER_NAME__",component=addresses,address=
"exampleQueue",subcomponent=queues,routingtype="anycast",queue="exampleQueue"
```

MBean 是：

```
org.apache.activemq.artemis.api.management.QueueControl
```

MBean 的 ObjectName 使用 helper 类 org.apache.activemq.artemis.api.core.management.ObjectNameBuilder 构建。您还可以使用 jconsole 查找您要管理的 MBeans 的 ObjectName。

使用 JMX 管理代理与使用 JMX 管理任何 Java 应用程序相同。它可以通过反映或创建 MBeans 的代理来完成。

6.2.1. 配置 JMX 管理

默认情况下，JMX 被启用来管理代理。您可以通过在 broker.xml 配置文件中设置 jmx-management-enabled 属性来启用或禁用 JMX 管理。

流程

1. 打开 & *lt;broker_instance_dir*> /etc/broker.xml 配置文件。
2. 设置 & lt;jmx-management-enabled>。

```
<jmx-management-enabled>true</jmx-management-enabled>
```

如果启用了 JMX，可以使用 jconsole 在本地管理代理。

注意

为安全起见，默认不启用到 JMX 的远程连接。

3. 如果要从同一 MBeanServer 管理多个代理，请为每个代理配置 JMX 域。

   默认情况下，代理使用 JMX 域 org.apache.activemq.artemis。

```
<jmx-domain>my.org.apache.activemq</jmx-domain>
```

注意

如果您在 Windows 系统上使用 AMQ Broker，则必须在 artemis 或 artemis.cmd 中设置系统属性。shell 脚本位于 < *install_dir*> /bin 下。

其他资源

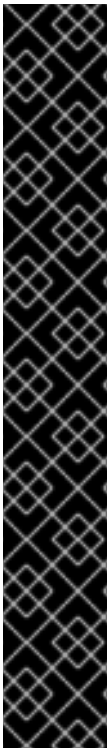
- 有关为远程管理配置代理的更多信息，请参阅 Oracle 的 Java 管理指南。

6.2.2. 配置 JMX 管理访问

默认情况下，因为安全原因，禁用对代理的远程 JMX 访问。但是，AMQ Broker 有一个 JMX 代理，它允许远程访问 JMX MBeans。您可以通过在代理 management.xml 配置文件中配置 connector 元素来启用 JMX 访问。

注意

虽然也可以使用 'com.sun.management.jmxremote ' JVM 系统属性启用 JMX 访问，但该方法不被支持且不受保护。修改 JVM 系统属性可以绕过代理上的 RBAC。要最大程度降低安全风险，请考虑对 localhost 进行有限的访问。



重要

为远程管理公开代理的 JMX 代理具有安全隐患。

要保护您的配置，如此流程所述：

- 将 SSL 用于所有连接。
- 明确定义连接器主机，即公开代理的主机和端口。
- 明确定义 RMI （远程方法调用）registry 绑定的端口。

先决条件

- 正常工作的代理实例
- Java jconsole 工具

流程

1. 打开 & *lt;broker-instance-dir*> /etc/management.xml 配置文件。
2. 为 JMX 代理定义连接器。connector-port 设置建立 RMI registry，如 jconsole 查询 JMX 连接器服务器。例如，允许在端口 1099 上进行远程访问：

```
<connector connector-port="1099"/>
```

3. 使用 jconsole 验证与 JMX 代理的连接：

```
service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi
```

4. 在连接器中定义附加属性，如下所述。

connector-host

用于公开代理的代理服务器主机。要防止远程访问，请将 connector-host 设置为 127.0.0.1 (localhost)。

rmi-registry-port

JMX RMI 连接器服务器绑定的端口。如果没有设置，则端口始终为随机。设置此属性以避免通过防火墙隧道的远程 JMX 连接问题。

jmx-realm

用于身份验证的 JMX 域。默认值为 activemq，以匹配 JAAS 配置。

object-name

用于公开远程连接器的对象名称。默认值为 connector:name=rmi。

secured

指定连接器是否使用 SSL 进行保护。默认值为 false。将值设为 true 以确保安全通信。

key-store-path

密钥存储的位置。如果您设置了 secure="true"，则需要此项。

key-store-password

密钥存储密码。如果您设置了 secure="true"，则需要此项。可以加密密码。

key-store-provider

密钥存储提供程序。如果您设置了 secure="true"，则需要此项。默认值为 JKS。

trust-store-path

信任存储的位置。如果您设置了 secure="true"，则需要此项。

trust-store-password

truststore 密码。如果您设置了 secure="true"，则需要此项。可以加密密码。

trust-store-provider

truststore 供应商。如果您设置了 secure="true"，则需要此项。默认值为 JKS

password-codec

要使用的密码 codec 的完全限定类名称。有关此操作的详情，请参见以下链接的密码屏

蔽文档。

5. 使用 jdk.serialFilter 为端点序列化设置适当的值，如 Java Platform 文档所述。

其他资源

- 有关配置文件中加密密码的更多信息，请参阅 在配置文件中加密密码。

6.2.3. MBeanServer 配置

代理以独立模式运行时，它使用 Java 虚拟机的平台 MBeanServer 注册其 MBeans。默认情况下，还部署了 Jolokia，以允许使用 REST 访问 MBean 服务器。

6.2.4. 如何使用 Jolokia 公开 JMX

默认情况下，AMQ Broker 附带作为 Web 应用程序部署的 Jolokia HTTP 代理。Jolokia 是一个通过 HTTP 网桥的远程 JMX，公开 MBeans。

注意

要使用 Jolokia，用户必须属于 < *broker_instance_dir*> /etc/artemis.profile 配置文件中的 hawtio. role 系统属性定义的角色。默认情况下，此角色是 amq。

例 6.1. 使用 Jolokia 查询代理版本

本例使用 Jolokia REST URL 来查找代理的版本。Origin 标志应指定代理服务器的域名或 DNS 主机名。另外，您为 Origin 指定的值必须与 Jolokia Cross- Origin Resource Sharing (CORS)规格中的 <allow-origin > 条目对应。

```
$ curl
http://admin:admin@localhost:8161/console/jolokia/read/org.apache.activemq.artemis:broker=\"0.0.0.0\"/Version -H "Origin: mydomain.com"
{"request":
{"mbean":"org.apache.activemq.artemis:broker=\"0.0.0.0\"","attribute":"Version","type":"read"},"value":"2.4.0.amq-710002-redhat-1","timestamp":1527105236,"status":200}
```

其他资源

- 有关使用 JMX-HTTP 网桥的更多信息，请参阅 Jolokia 文档。

- 有关为用户分配用户的更多信息，请参阅 添加用户。

- 有关指定 Jolokia Cross-Origin Resource Sharing (CORS)的更多信息，请参阅 安全 4.1.5 部分。

6.2.5. 订阅 JMX 管理通知

如果环境中启用了 JMX，您可以订阅管理通知。

流程

- 订阅 ObjectName org.apache.activemq.artemis:broker=" <*broker-name*>".

其他资源

- 有关管理通知的详情，请参考 第 6.5 节 “管理通知”。

6.3. 使用 JMS API 管理 AMQ BROKER

Java 消息服务(JMS) API 允许您创建、发送、接收和读取消息。您可以使用 JMS 和 AMQ JMS 客户端来管理代理。

6.3.1. 使用 JMS 消息和 AMQ JMS 客户端配置代理管理

要使用 JMS 管理代理，您必须首先使用管理权限配置代理的管理地址。

流程

1. 打开 & *lt;broker_instance_dir*> /etc/broker.xml 配置文件。

2. 添加 & lt;management-address > 元素并指定管理地址。

默认情况下，管理地址为 activemq.management。您只需要在您不想使用默认地址时指定不同的地址。

```
<management-address>my.management.address</management-address>
```

3. 为管理地址提供 管理用户权限 类型。

此权限类型可让管理地址接收和处理管理消息。

```
<security-setting-match="activemq.management">
   <permission-type="manage" roles="admin"/>
</security-setting>
```

6.3.2. 使用 JMS API 和 AMQ JMS 客户端管理代理

要使用 JMS 消息调用管理操作，AMQ JMS 客户端必须实例化特殊的管理队列。

流程

1. 创建一个 QueueRequestor，将消息发送到管理地址并接收回复。
2. 创建 消息。
3. 使用 helper 类 org.apache.activemq.artemis.api.jms.management.JMSManagementHelper 来填充管理属性的消息。
4. 使用 QueueRequestor 发送消息。
5. 使用 helper 类 org.apache.activemq.artemis.api.jms.management.JMSManagementHelper，从管理回复中检索操作结果。

例 6.2. 查看队列中的消息数

本例演示了如何使用 JMS API 查看 JMS 队列 示例Queue 中的消息数：

```
Queue managementQueue = ActiveMQJMSClient.createQueue("activemq.management");

QueueSession session = ...
QueueRequestor requestor = new QueueRequestor(session, managementQueue);
connection.start();
Message message = session.createMessage();
JMSManagementHelper.putAttribute(message, "queue.exampleQueue", "messageCount");
Message reply = requestor.request(message);
int count = (Integer)JMSManagementHelper.getResult(reply);
System.out.println("There are " + count + " messages in exampleQueue");
```

6.4. 管理操作

无论您使用 JMX 或 JMS 消息来管理 AMQ Broker，都可以使用相同的 API 管理操作。使用管理 API，您可以管理代理、地址和队列。

6.4.1. 代理管理操作

您可以使用管理 API 管理代理。

列出、创建、部署和销毁队列

可以使用 getQueueNames （） 方法来检索已部署队列的列表。

可以在 ActiveMQServerControl 上使用管理操作 createQueue （）、deployQueue （） 或 destroyQueue （） （带有 ObjectName org.apache.activemq.artemis:broker="*BROKER_NAME*" 或资源 名称服务器）来创建或销毁队列。

如果 deployQueue 已存在队列，则 createQueue 将失败。

暂停和恢复队列

QueueControl 可以暂停并恢复底层队列。当队列暂停时，它将接收消息，但不会发送消息。恢复后，它将开始发送已排队的消息（若有）。

列出和关闭远程连接

使用 listRemoteAddresses （） 检索客户端的远程地址。也可以使用 closeConnectionsForAddress （） 方法关闭与远程地址关联的连接。

或者，使用 listConnectionIDs （） 列出连接 ID，并使用 listSessions （） 列出给定连接 ID

的所有会话。

管理事务

如果代理崩溃，当代理重启时，一些事务可能需要手动干预。使用以下方法帮助您解决遇到的问题。

使用 listPreparedTransactions （） 方法列表，列出处于准备状态（事务以 opaque Base64 Strings）状态的事务。

使用 commitPreparedTransaction （） 或 rollbackPreparedTransaction （） 提交或回滚给定的准备事务，以解决 heuristic 事务。

使用 listHeuristic CommittedTransactions （）和 listHeur isticRolledBackTransactions 方法列出 heuristicly 完成事务。

启用和重置消息计数器

使用 enableMessageCounters （） 或 disableMessageCounters （） 方法启用和禁用消息计数器。

使用 resetAllMessageCounters （） 和 resetAllMessageCounterHistories （） 方法重置消息计数器。

检索代理配置和属性

ActiveMQServerControl 通过其所有属性（如 getVersion ）方法公开代理的配置，以检索代理的版本等。

列出、创建和销毁 Core Bridge 和 diverts

使用 getBridgeNames （） 和 getDivertNames （） 方法列出部署的 Core Bridge 和 diverts。

使用网桥创建或销毁，并使用 createBridge （） 和 destroyBridge （） 或在 ActiveMQServerControl 上的 createDivert （） 和 destroyDivert （） （带有 ObjectName org.apache.activemq.artemis:broker="*BROKER_NAME*" 或资源 名称服务器）创建或销毁。

停止代理并强制在任何当前附加的客户端中进行故障转移

在 ActiveMQServerControl 上使用 forceFailover （） （带有 ObjectName org.apache.activemq.artemis:broker="*BROKER_NAME*" 或资源 名称服务器）

注意

由于此方法实际上停止代理，所以您可能会收到错误。确切的错误取决于您用于调用方法的管理服务。

6.4.2. 地址管理操作

您可以使用管理 API 管理地址。

使用带有 ObjectName org.apache.activemq.artemis:broker=" <broker-*name*> ", *component=addresses,address=*" <*address-name*> " 或资源名称地址的 AddressControl 类来管理地址。<*address-name*>。

使用 addRole （） 或 removeRole （） 方法修改地址的角色和权限。您可以使用 getRoles （） 方法列出与队列关联的所有角色。

6.4.3. 队列管理操作

您可以使用管理 API 管理队列。

核心管理 API 处理队列。QueueControl 类定义了队列管理操作（带有 ObjectName,org.apache.activemq.artemis:broker=" <*broker-name*> ",component=addresses,address=" <*bound-address*> ",subcomponent=queues,routing-type=" <routing-type> ",queue=" <*queue-name*> " 或资源名称 queue . <*queue-name*>）。

队列上的大多数管理操作都会使用一个消息 ID （例如，删除单个消息）或过滤器（例如，将所有消息与给定属性过期）。

过期、发送到死信地址并移动消息

使用 expireMessages （） 方法使来自队列的消息过期。如果定义了到期地址，则会将消息发送到此地址，否则将丢弃它们。您可以在 broker.xml 配置文件的 address-settings 元素中为地址 或一组 地址（以及绑定到这些地址的队列）定义到期地址。例如，请参阅 了解默认代理配置 中的"默认消息地址设置"部分。

使用 sendMessagesToDeadLetterAddress （） 方法向死信地址发送消息。此方法返回发送到

死信地址的消息数。如果定义了死信地址，则会将消息发送到此地址，否则将从队列中删除它们并丢弃。您可以在 broker.xml 配置文件的 address 或 一组 地址（以及绑定到那些地址的队列）定义死信地址。例如，请参阅 了解默认代理配置 中的"默认消息地址设置"部分。

使用 moveMessages （） 方法将消息从一个队列移动到另一个队列。

列出和删除消息

使用 listMessages （） 方法列出来自队列的消息。它将返回一个 Map 数组，每个消息一个 映射。

使用 removeMessages （） 方法从队列中删除消息，它返回单个消息 ID 变体的布尔值或过滤器变体删除的消息数量。此方法采用 过滤器参数 来只删除过滤的消息。将过滤器设置为空字符串将删除所有消息。

计数消息

队列中的消息数量通过 getMessageCount （） 方法返回。或者，countMessages （） 将返回队列中与给定过滤器匹配的消息数。

更改消息优先级

可以使用 changeMessagesPriority （） 方法更改消息优先级，该方法 返回单个消息 ID 变体的布尔值或过滤器变体更新的消息数量。

消息计数器

可以使用 listMessageCounter （） 和 listMessageCounterHistory （） 方法（请参阅 第 6.6 节 “使用消息计数器”）为队列列出消息计数器（请参阅 ）。也可以使用 resetMessageCounter （） 方法为单个队列重置消息计数器。

检索队列属性

QueueControl 通过其属性（例如 getFilter （） ）公开队列设置，以检索队列的过滤器（如果使用一个队列，即 isDurable （） ），以了解队列是否存活，以此类推。

暂停和恢复队列

QueueControl 可以暂停并恢复底层队列。当队列暂停时，它将接收消息，但不会发送消息。恢复后，它将开始发送已排队的消息（若有）。

6.4.4. 远程资源管理操作

您可以使用管理 API 启动和停止代理的远程资源（接受者、解除者、桥接等），以便在不需要完全停

止的情况下使代理离线。

acceptors

使用 start （） 或.AcceptorControl 类的 stop （） 方法（带有 ObjectName org.apache.activemq.artemis:broker=" <broker-name> ",component=acceptors,name=" <acceptor-name>" 或资源名称 acceptor. <address-name>）。acceptor 参数可以使用 AcceptorControl 属性来检索。有关 接受器的更多信息，请参阅 网络连接：接受器和连接器。

diverts

在 DivertControl 类中使用 start （） 或 stop （） 方法（带有 ObjectName org.apache.activemq.artemis:broker=" <broker-name> ",component=diverts,name=" <divert-name> " 或 resource name divert. < divert-name> ）来启动或停止一个 divert。可以使用 DivertControl 属性来检索 divert 参数。

网桥

使用 start （） (Resp)启动或停止网桥。BridgeControl 类上的 stop （）方法（带有 ObjectName org.apache.activemq.artemis:broker=" <broker-name> ",component=bridge,name=" <bridge-name>" 或资源名称 bridge. <bridge-name>）。网桥参数可以使用 BridgeControl 属性来检索。

广播组

使用 BroadcastGroupControl 类的 start （） 或 stop （） 方法启动或停止广播组（带有 ObjectName org.apache.activemq.artemis:broker=" <broker-name> ",component=broadcast-group,name=" <broadcast-group-name> " 或资源名称 broadcastgroup. <broadcast-group-name>）。可以使用 BroadcastGroupControl 属性来检索广播组参数。如需更多信息，请参阅 代理发现方法。

发现组

在 DiscoveryGroupControl 类中使用 start （） 或 stop （） 方法（带有 ObjectName org.apache.activemq.artemis:broker=" <broker-name> ",component=discovery-group,name=" <discovery-group-name> " 或 resource name discovery . <discovery-group-name& gt; ）来启动或停止发现组。可以使用 DiscoveryGroupControl 属性来检索发现组参数。如需更多信息，请参阅 代理发现方法。

集群连接

使用 ClusterConnectionControl 类的 start （） 或 stop （） 方法（带有 ObjectName org.apache.activemq.artemis:broker=" <broker-name> ",component=cluster-connection,name=" <cluster-connection-name> " 或资源名称 clusterconnection . <cluster-connection-name> ）来启动或停止集群连接。集群连接参数可以使用 ClusterConnectionControl 属性来检索。如需更多信息，请参阅 创建代理集群。

6.5. 管理通知

*以下是所有不同类型的通知列表，以及哪些标头位于消息中。每个通知都有一个 **_AMQ_NotifType***
*（在括号中记录的值）和 **_AMQ_NotifTimestamp** 标头。时间戳是调用*
java.lang.System.currentTimeMillis （） 的未格式化结果。

通知类型	Headers
绑定_ADDED (0)	**_AMQ_Binding_Type** **_AMQ_Address** **_AMQ_ClusterName** **_AMQ_RoutingName** **_AMQ_Binding_ID** **_AMQ_Distance** **_AMQ_FilterString**
BINDING_REMOVED (1)	**_AMQ_Address** **_AMQ_ClusterName** **_AMQ_RoutingName** **_AMQ_Binding_ID** **_AMQ_Distance** **_AMQ_FilterString**
CONSUMER_CREATED (2)	**_AMQ_Address** **_AMQ_ClusterName** **_AMQ_RoutingName** **_AMQ_Distance** **_AMQ_ConsumerCount** **_AMQ_User** **_AMQ_RemoteAddress** **_AMQ_SessionName** **_AMQ_FilterString**

通知类型	Headers
CONSUMER_CLOSED (3)	**_AMQ_Address** **_AMQ_ClusterName** **_AMQ_RoutingName** **_AMQ_Distance** **_AMQ_ConsumerCount** **_AMQ_User** **_AMQ_RemoteAddress** **_AMQ_SessionName** **_AMQ_FilterString**
SECURITY_AUTHENTICATION_VIOLATION (6)	**_AMQ_User**
SECURITY_PERMISSION_VIOLATION (7)	**_AMQ_Address** **_AMQ_CheckType** **_AMQ_User**
DISCOVERY_GROUP_STARTED (8)	**name**
DISCOVERY_GROUP_STOPPED (9)	**name**
BROADCAST_GROUP_STARTED (10)	**name**
BROADCAST_GROUP_STOPPED (11)	**name**
BRIDGE_STARTED (12)	**name**
BRIDGE_STOPPED (13)	**name**
CLUSTER_CONNECTION_STARTED (14)	**name**
CLUSTER_CONNECTION_STOPPED (15)	**name**
ACCEPTOR_STARTED (16)	**factory** **id**
ACCEPTOR_STOPPED (17)	**factory** **id**

通知类型	Headers
PROPOSAL (18)	**_JBM_ProposalGroupId** **_JBM_ProposalValue** **_AMQ_Binding_Type** **_AMQ_Address** **_AMQ_Distance**
PROPOSAL_RESPONSE (19)	**_JBM_ProposalGroupId** **_JBM_ProposalValue** **_JBM_ProposalAltValue** **_AMQ_Binding_Type** **_AMQ_Address** **_AMQ_Distance**
CONSUMER_SLOW (21)	**_AMQ_Address** **_AMQ_ConsumerCount** **_AMQ_RemoteAddress** **_AMQ_ConnectionName** **_AMQ_ConsumerName** **_AMQ_SessionName**

6.6. 使用消息计数器

您可以使用消息计数器获取一段时间内队列的信息。这有助于您识别很难看到的趋势。

例如，您可以使用消息计数器来确定如何随着时间的推移使用特定队列。您还可以使用管理 API 定期查询队列中的消息数量，但这不会显示队列实际使用的使用方式。队列中的消息数量可能会保持恒定，因为没有客户端在其上发送或接收消息，或者因为发送到队列的消息数量等于来自它的消息数量。在这两种情况下，队列中的消息数量都保持不变，即使它以非常不同的方式使用。

6.6.1. 消息计数器的类型

消息计数器提供有关代理队列的附加信息。

数量

从代理启动后添加到队列的消息总数。

countDelta

从上次消息计数器更新以来添加到队列的消息数量。

lastAckTimestamp

最后一次确认来自队列中消息的时间戳。

lastAddTimestamp

最后一次向队列添加消息的时间戳。

messageCount

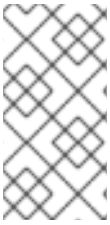
队列中的当前消息数量。

messageCountDelta

从上次消息计数器更新以来从队列中添加/删除的消息总数。例如，如果 messageCountDelta 是 -10，则整个消息已从队列中删除 10 个消息。

udpateTimestamp

最后消息计数器更新的时间戳。



注意

您还可以组合消息计数器来确定其他有意义的数据。例如，若要知道从上一次更新以来从队列中消耗了多少个消息，您可以从 countDelta 中减去 messageCountDelta。

6.6.2. 启用消息计数器

消息计数器可能会对代理内存有小的影响，因此默认禁用它们。要使用消息计数器，您必须首先启用它们。

流程

1. *打开 & lt;broker_instance_dir> /etc/broker.xml 配置文件。*

2. 启用消息计数器。

```
<message-counter-enabled>true</message-counter-enabled>
```

3. 设置消息计数器历史记录和抽样周期。

```
<message-counter-max-day-history>7</message-counter-max-day-history>
<message-counter-sample-period>60000</message-counter-sample-period>
```

message-counter-max-day-history

代理应存储队列指标的天数。默认值为 10 天。

message-counter-sample-period

代理应对其进行队列抽样（以毫秒为单位）来收集指标的频率。默认值为 10000 毫秒。

6.6.3. 检索消息计数器

您可以使用管理 API 来检索消息计数器。

先决条件

- 代理上必须启用消息计数器。

  更多信息请参阅 第 6.6.2 节 “启用消息计数器”。

流程

- 使用管理 API 来检索消息计数器。

```
// Retrieve a connection to the broker's MBeanServer.
MBeanServerConnection mbsc = ...
JMSQueueControlMBean queueControl =
(JMSQueueControl)MBeanServerInvocationHandler.newProxyInstance(mbsc,
   on,
   JMSQueueControl.class,
   false);

// Message counters are retrieved as a JSON string.
String counters = queueControl.listMessageCounter();
```

```
// Use the MessageCounterInfo helper class to manipulate message counters more
easily.
MessageCounterInfo messageCounter = MessageCounterInfo.fromJSON(counters);
System.out.format("%s message(s) in the queue (since last sample: %s)\n",
messageCounter.getMessageCount(),
messageCounter.getMessageCountDelta());
```

其他资源

- 有关消息计数器的详情请参考 第 6.4.3 节 “队列管理操作”。

第 7 章 监控代理问题

AMQ Broker 包括一个称为 Critical Analyzer 的内部工具，它主动监控正在运行的代理，以了解死锁条件等问题。在生产环境中，一个问题（如死锁条件）可能是 IO 错误、缺陷磁盘、内存短缺或因其他进程导致的 CPU 使用量造成的。

Critical 分析会定期测量关键操作的响应时间，如队列交付（即，将消息添加到代理队列）和日志操作。如果检查操作的响应时间超过可配置的超时值，则代理被视为不稳定。在这种情况下，您可以将 Critical Analyzer 配置为仅记录消息或采取措施来保护代理，如关闭代理或停止运行代理的虚拟机(VM)。

7.1. 配置关键分析器

以下流程演示了如何配置 Critical 分析器来监控代理是否有问题。

流程

1. 打开 & lt;broker_instance_dir> /etc/broker.xml 配置文件。

   Critical 分析器的默认配置如下所示。

   ```
   <critical-analyzer>true</critical-analyzer>
   <critical-analyzer-timeout>120000</critical-analyzer-timeout>
   <critical-analyzer-check-period>60000</critical-analyzer-check-period>
   <critical-analyzer-policy>HALT</critical-analyzer-policy>
   ```

2. 指定参数值，如下所述。

   critical-analyzer

   指定是否启用或禁用 Critical 分析器工具。默认值为 **true**，这意味着启用了该工具。

   critical-analyzer-timeout

   由 Critical 分析器运行的检查的超时时间（毫秒）。如果检查的操作花费的时间超过这个值，则代理被视为不稳定。

   critical-analyzer-check-period

   在关键分析器连续检查每个操作的时间段（以毫秒为单位）。

   critical-analyzer-policy

如果代理失败检查并被视为不稳定，此参数指定代理是否记录一条消息(LOG)、停止托管代理的虚拟机(VM)或关闭代理(SHUTDOWN)。

根据您配置的策略选项，如果关键操作的响应时间超过配置的超时值，您会看到类似如下的输出：

critical-analyzer-policy=LOG

> *[Artemis Critical Analyzer] 18:11:52,145 WARN [org.apache.activemq.artemis.core.server] AMQ224081: The component org.apache.activemq.artemis.tests.integration.critical.CriticalSimpleTest$2@5af97850 is not responsive*

critical-analyzer-policy=HALT

> *[Artemis Critical Analyzer] 18:10:00,831 ERROR [org.apache.activemq.artemis.core.server] AMQ224079: The process for the virtual machine will be killed, as component org.apache.activemq.artemis.tests.integration.critical.CriticalSimpleTest$2@5af97850 is not responsive*

critical-analyzer-policy=SHUTDOWN

> *[Artemis Critical Analyzer] 18:07:53,475 ERROR [org.apache.activemq.artemis.core.server] AMQ224080: The server process will now be stopped, as component org.apache.activemq.artemis.tests.integration.critical.CriticalSimpleTest$2@5af97850 is not responsive*

您还可以在代理中看到类似如下的线程转储：

> *[Artemis Critical Analyzer] 18:10:00,836 WARN [org.apache.activemq.artemis.core.server]*

AMQ222199: Thread dump: AMQ119001: Generating thread dump
*
==========
AMQ119002: Thread Thread[Thread-1 (ActiveMQ-scheduled-threads),5,main] name = Thread-1 (ActiveMQ-scheduled-threads) id = 19 group = java.lang.ThreadGroup[name=main,maxpri=10]
sun.misc.Unsafe.park(Native Method)
java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1088)
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:809) java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1067)
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1127)
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
java.lang.Thread.run(Thread.java:745)
==========
....
==========
*AMQ119003: End Thread dump **

更新于 2024-06-11