Red Hat AMQ Broker 7.12

AMQ Broker の管理

AMQ Broker 7.11 で使用する場合

Last Updated: 2024-05-22

Red Hat AMQ Broker 7.12 AMQ Broker の管理

AMQ Broker 7.11 で使用する場合

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドでは、AMQ Broker の監視、管理、およびアップグレード方法について説明します。

目次

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 概要

AMQ Broker は、ActiveMQ Artemis をベースとした高パフォーマンスのメッセージング実装です。高速なジャーナルベースのメッセージ永続性を備え、複数の言語、プロトコル、およびプラットフォームをサポートします。

AMQ Broker は、管理コンソール、管理 API、コマンドラインインターフェイスなど、ブローカーインスタンスを管理および操作するための複数のインターフェイスを提供します。さらに、実行時メトリクスを収集してブローカのパフォーマンスを監視したり、デッドロック状態などの問題をプロアクティブにモニターするようにブローカを設定したり、ブローカやキューの健全性をインタラクティブにチェックしたりできます。

このガイドでは、次のような一般的なブローカー管理タスクに関する詳細情報を提供します。

- ブローカーインスタンスのアップグレード
- コマンドラインインターフェイスと管理 API の使用
- ブローカーおよびキューの健全性の確認
- ブローカーランタイムメトリックの収集
- 重要なブローカー操作をプロアクティブに監視

1.1. サポートされる構成

AMQ Broker のサポートされる設定に関する最新情報は、Red Hat カスタマーポータルの記事 Red Hat AMQ 7 でサポートされる構成 を参照してください。

1.2. このドキュメントの表記慣例

このドキュメントでは、**sudo** コマンド、ファイルパス、および置き換え可能な値について、以下の規則を使用します。

sudo コマンド
このドキュメントでは、root 権限を必要とするすべてのコマンドに対して **sudo** が使用されています。何らかの変更がシステム全体に影響を与える可能性があるため、**sudo** を使用する場合は、常に注意が必要です。

sudo の使用の詳細は、sudo アクセスの管理 を参照してください。

このドキュメントにおけるファイルパスの使用
このドキュメントでは、すべてのファイルパスは Linux、UNIX、および同様のオペレーティングシステムで有効です (例: **/home/...**)。Microsoft Windows を使用している場合は、同等の Microsoft Windows パスを使用する必要があります (例: **C:\Users\...**)。

交換可能な値
このドキュメントでは、お客様の環境に合わせた値に置き換える必要のある置換可能な値を使用している場合があります。置き換え可能な値は小文字で、角括弧 (**< >**) で囲まれ、イタリックおよび **monospace** フォントを使用してスタイルされます。単語が複数になる場合は、アンダースコア (_) で区切ります。

たとえば、**<install_dir>** を独自のディレクトリー名に置き換えます。

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

第2章 ブローカーのアップグレード

2.1. アップグレードについて

Red Hat は、AMQ Broker の新しいバージョンを カスタマーポータル にリリースします。ブローカーを最新バージョンに更新し、最新の機能強化および修正があることを確認します。通常、Red Hat は AMQ Broker の新バージョンを以下の 3 つの方法でリリースします。

メジャーリリース

AMQ Broker 6 から AMQ Broker 7 など、アプリケーションがあるメジャーリリースから次のメジャーリリースに移行する場合は、メジャーアップグレードまたは移行が必要です。この種のアップグレードについては、本書では扱いません。

マイナーリリース

AMQ Broker では、マイナーリリースを定期的に提供します。マイナーリリースは、新機能やバグ修正が含まれる更新です。AMQ Broker マイナーリリースを別のリリースにアップグレードする計画がある場合 (AMQ Broker 7.0 から AMQ Broker 7.1 など)、プライベート、サポートされていない、またはテクノロジープレビューコンポーネントを使用しないアプリケーションには、コードの変更は必要ありません。

マイクロリリース

AMQ Broker は、マイナーな機能強化および修正が含まれるマイクロリリースを定期的に提供します。マイクロリリースは、7.0.1 から 7.0.2 など、最後の数字のマイナーリリースバージョンを増分します。マイクロリリースはコードの変更を必要としませんが、一部のリリースには設定の変更が必要になる場合があります。

2.2. 古い 7.X バージョンのアップグレード

2.2.1. ブローカーインスタンスの 7.0.x から 7.0.y へのアップグレード

AMQ Broker を別のバージョン 7.0 にアップグレードする手順は、インストール用の手順と類似しています。カスタマーポータルからアーカイブをダウンロードしてデプロイメントします。

以下のサブセクションでは、異なるオペレーティングシステムの 7.0.x ブローカーをアップグレードする方法を説明します。

- Linux の 7.0.x から 7.0.y へのアップグレード
- Windows の 7.0.x から 7.0.y へのアップグレード

2.2.1.1. Linux の 7.0.x から 7.0.y へのアップグレード

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

前提条件

- AMQ Broker をアップグレードする前に、ターゲットリリースのリリースノートを確認してください。
  本リリースノートでは、ターゲットリリースにおける重要な機能拡張、既知の問題、および動作の変更を説明します。

  詳細は、AMQ Broker 7.0 リリースノート を参照してください。

手順

1. AMQ Broker アーカイブのダウンロード の手順に従って、Red Hat カスタマーポータルから必要なアーカイブを ダウンロードします。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** というディレクトリーを使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。アーカイブは圧縮形式で保持されます。以下の例では、ユーザー **amq-broker** は unzip コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行している場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** にあるログファイルの最後に、以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集して、アーカイブを抽出した際に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

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

9. アップグレードされたブローカーを起動します。

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

10. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーの起動後、ログファイル **<broker_instance_dir>/log/artemis.log** を開くと、以下のような 2 行があります。ブローカーの稼働後にログに表示される新しいバージョン番号に注意してください。

```
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. Windows の 7.0.x から 7.0.y へのアップグレード

前提条件

- AMQ Broker をアップグレードする前に、ターゲットリリースのリリースノートを確認してください。
  本リリースノートでは、ターゲットリリースにおける重要な機能拡張、既知の問題、および動作の変更を説明します。

  詳細は、AMQ Broker 7.0 リリースノート を参照してください。

手順

1. AMQ Broker アーカイブのダウンロード の手順に従って、Red Hat カスタマーポータルから必要なアーカイブを ダウンロードします。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。
4. 以下のコマンドを入力してブローカーが実行している場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。
   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。
   b. 同じウィンドウを右クリックし、**Paste** を選択します。
6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** にあるログファイルの最後に、以下のような行が表示されます。

   ```
   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. **<broker_instance_dir>\etc\artemis.profile** 設定ファイルを編集して、アーカイブを抽出した際に作成された新しいディレクトリーに、**ARTEMIS_HOME** プロパティーを設定してください。

   ```
   ARTEMIS_HOME=<install_dir>
   ```

8. アップグレードされたブローカーを起動します。

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

9. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、ログファイル **<broker_instance_dir>\log\artemis.log** を開くと、以下のような行があります。ブローカーの稼働後にログに表示される新しいバージョン番号に注意してください。

```
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. ブローカーインスタンスの 7.0.x から 7.1.0 へのアップグレード

AMQ Broker 7.1.0 には、以前のバージョンに含まれていない設定ファイルおよび設定が含まれています。ブローカーインスタンスを 7.0.x から 7.1.0 にアップグレードするには、これらの新しいファイルおよび設定を既存の 7.0.x ブローカーインスタンスに追加する必要があります。以下のサブセクションでは、異なるオペレーティングシステムの 7.0.x ブローカーインスタンスを 7.1.0 にアップグレードする方法を説明します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux での 7.0.x から 7.1.0 へのアップグレード
- Windows での 7.0.x から 7.1.0 へのアップグレード

2.2.2.1. Linux での 7.0.x から 7.1.0 へのアップグレード

7.0.x ブローカーをアップグレードする前に、Red Hat AMQ Broker 7.1.0 をインストールし、一時的なブローカーインスタンスを作成する必要があります。これにより、7.0.x ブローカーのアップグレードに必要な 7.1.0 設定ファイルが生成されます。

前提条件

- AMQ Broker をアップグレードする前に、ターゲットリリースのリリースノートを確認してください。
  本リリースノートでは、ターゲットリリースにおける重要な機能拡張、既知の問題、および動作の変更を説明します。

  詳細は、AMQ Broker 7.1 リリースノート を参照してください。
- 7.0.x ブローカーをアップグレードする前に、最初にバージョン 7.1 をインストールする必要があります。
  Linux に 7.1 をインストールする手順は、Installing AMQ Broker を参照してください。

手順

1. 実行中の場合は、アップグレードする 7.0.x ブローカーを停止します。

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

2. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

3. 7.0.x ブローカーの **<broker_instance_dir>/etc/** ディレクトリーにある **artemis.profile** ファイルを開きます。

   a. **ARTEMIS_HOME** プロパティーを更新し、その値が AMQ Broker 7.1.0 のインストールディレクトリーを参照するようにします。

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

   b. 更新した行の 1 つ下の行で、プロパティー **ARTEMIS_INSTANCE_URI** を追加して、7.0.x ブローカーインスタンスディレクトリーを参照する値を割り当てます。

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

   c. **JAVA_ARGS** プロパティーに **jolokia.policyLocation** パラメーターを追加し、以下の値を割り当てて更新します。

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

4. 7.1.0 ブローカーインスタンスを作成します。作成手順では、7.0.x から 7.1.0 へのアップグレードに必要な設定ファイルが生成されます。以下の例では、インスタンスが **upgrade_tmp** ディレクトリーに作成されることに注意してください。

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

5. 一時的な 7.1.0 インスタンスの **etc** ディレクトリーから、7.0.x ブローカーの **<broker_instance_dir>/etc/** ディレクトリーに設定ファイルをコピーします。

   a. **management.xml** ファイルをコピーします。

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

   b. **jolokia-access.xml** ファイルをコピーします。

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

6. 7.0.x ブローカーの **<broker_instance_dir>/etc/** ディレクトリーにある **bootstrap.xml** ファイルを開きます。

   a. 以下の 2 つの行をコメントアウトまたは削除します。

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

b. 以下の行を追加して、直前の手順で削除された 2 つの行を置き換えます。

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

7. アップグレードしたブローカーを起動します。

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

関連情報

ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。

2.2.2.2. Windows での 7.0.x から 7.1.0 へのアップグレード

7.0.x ブローカーをアップグレードする前に、Red Hat AMQ Broker 7.1.0 をインストールし、一時的なブローカーインスタンスを作成する必要があります。これにより、7.0.x ブローカーのアップグレードに必要な 7.1.0 設定ファイルが生成されます。

前提条件

- AMQ Broker をアップグレードする前に、ターゲットリリースのリリースノートを確認してください。
  本リリースノートでは、ターゲットリリースにおける重要な機能拡張、既知の問題、および動作の変更を説明します。

  詳細は、AMQ Broker 7.1 リリースノート を参照してください。

- 7.0.x ブローカーをアップグレードする前に、最初にバージョン 7.1 をインストールする必要があります。
  Windows に 7.1 をインストールする手順は、Installing AMQ Broker を参照してください。

手順

1. 実行中の場合は、アップグレードする 7.0.x ブローカーを停止します。

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

2. ファイルマネージャーを使用してブローカーのインスタンスディレクトリーをバックアップします。

   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。

   b. 同じウィンドウを右クリックし、**Paste** を選択します。

3. 7.0.x ブローカーの **<broker_instance_dir>/etc/** ディレクトリーにある **artemis.profile** ファイルを開きます。

   a. **ARTEMIS_HOME** プロパティーを更新し、その値が AMQ Broker 7.1.0 のインストールディレクトリーを参照するようにします。

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

   b. 更新した行の 1 つ下の行で、プロパティー **ARTEMIS_INSTANCE_URI** を追加して、7.0.x ブローカーインスタンスディレクトリーを参照する値を割り当てます。

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

c. **JAVA_ARGS** プロパティーに **jolokia.policyLocation** パラメーターを追加し、以下の値を割り当てて更新します。

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

4. 7.1.0 ブローカーインスタンスを作成します。作成手順では、7.0.x から 7.1.0 へのアップグレードに必要な設定ファイルが生成されます。以下の例では、インスタンスが **upgrade_tmp** ディレクトリーに作成されることに注意してください。

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

5. 一時的な 7.1.0 インスタンスの **etc** ディレクトリーから、7.0.x ブローカーの **<broker_instance_dir>/etc/** ディレクトリーに設定ファイルをコピーします。

a. **management.xml** ファイルをコピーします。

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

b. **jolokia-access.xml** ファイルをコピーします。

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

6. 7.0.x ブローカーの **<broker_instance_dir>/etc/** ディレクトリーにある **bootstrap.xml** ファイルを開きます。

a. 以下の 2 つの行をコメントアウトまたは削除します。

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

b. 以下の行を追加して、直前の手順で削除された 2 つの行を置き換えます。

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

7. アップグレードしたブローカーを起動します。

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

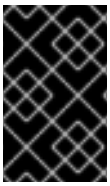
関連情報

ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。

2.2.3. ブローカーインスタンスの 7.1.x から 7.2.0 へのアップグレード

AMQ Broker 7.2.0 には、7.0.x バージョンに含まれていない設定ファイルおよび設定が含まれています。7.0.x インスタンスを実行している場合は、最初にこれらのブローカーインスタンスを 7.0.x から 7.1.0 にアップグレードしてから 7.2.5.0 にアップグレードする必要があります。以下のサブセクション

では、異なるオペレーティングシステムの 7.1.x ブローカーインスタンスを 7.2.0 にアップグレードする方法を説明します。



重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux での 7.1.x から 7.2.0 へのアップグレード
- Windows での 7.1.x から 7.2.0 へのアップグレード

2.2.3.1. Linux での 7.1.x から 7.2.0 へのアップグレード

注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. AMQ Broker アーカイブのダウンロード の手順に従って、Red Hat カスタマーポータルから必要なアーカイブを ダウンロードします。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** というディレクトリーを使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は unzip コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行している場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** にあるログファイルの最後に、以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集して、アーカイブを抽出した際に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

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

9. アップグレードされたブローカーを起動します。

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

10. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーの起動後、ログファイル **<broker_instance_dir>/log/artemis.log** を開くと、以下のような 2 行があります。ブローカーの稼働後にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.2.3.2. Windows での 7.1.x から 7.2.0 へのアップグレード

手順

1. AMQ Broker アーカイブのダウンロード の手順に従って、Red Hat カスタマーポータルから必要なアーカイブを ダウンロードします。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。

4. 以下のコマンドを入力してブローカーが実行している場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。

   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。

   b. 同じウィンドウを右クリックし、**Paste** を選択します。

6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** にあるログファイルの最後に、以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集して、アーカイブを抽出した際に作成された新しいディレクトリーに、**ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. アップグレードされたブローカーを起動します。

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

9. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、ログファイル **<broker_instance_dir>\log\artemis.log** を開くと、以下のような行があります。ブローカーの稼働後にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.2.4. ブローカーインスタンスの 7.2.x から 7.3.0 へのアップグレード

以下のサブセクションでは、異なるオペレーティングシステムの 7.2.x ブローカーインスタンスを 7.3.0 にアップグレードする方法を説明します。

2.2.4.1. 非推奨のディスパッチコンソールによる例外の解決

7.3.0 以降、AMQ Broker には Hawtio ディスパッチコンソールプラグインである **dispatch-hawtio-console.war** は同梱されなくなりました。以前のバージョンでは、AMQ Interconnect の管理にディスパッチコンソールを使用していました。ただし、AMQ Interconnect は独自のスタンドアロン Web コンソールを使用するようになりました。この変更は、以降のセクションのアップグレード手順に影響します。

ブローカーインスタンスを 7.3.0 にアップグレードする前に追加のアクションを実行すると、アップグレードプロセスにより以下のような例外が生成されます。

```
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.
```

アップグレードの成功に影響を及ぼすことなく、前述の例外を無視しても問題ありません。

ただし、アップグレード中にこの例外が表示されないようにする場合は、最初に既存のブローカーインスタンスの **bootstrap.xml** ファイルで Hawtio ディスパッチコンソールプラグインへの参照を削除する必要があります。**bootstrap.xml** ファイルは、ブローカーインスタンスの **{instance_directory}/etc/** ディレクトリーにあります。以下の例は、AMQ Broker 7.2.4 インスタンスの **bootstrap.xml** ファイルの内容の一部を示しています。

```
<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>
```

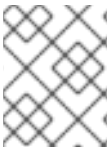
AMQ Broker をバージョン 7.3.0 にアップグレードする際に例外を回避するには、前述の例のように **<app url="dispatch-hawtio-console" war="dispatch-hawtio-console.war"/>** の行を削除します。次に、後続のセクションで説明されているように、変更したブートストラップファイルを保存し、アップグレードプロセスを開始します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux での 7.2.x から 7.3.0 へのアップグレード
- Windows での 7.2.x から 7.3.0 へのアップグレード

2.2.4.2. Linux での 7.2.x から 7.3.0 へのアップグレード



注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. AMQ Broker アーカイブのダウンロード の手順に従って、Red Hat カスタマーポータルから必要なアーカイブを ダウンロードします。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** というディレクトリーを使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は unzip コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行している場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** にあるログファイルの最後に、以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集して、アーカイブを抽出した際に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

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

9. アップグレードされたブローカーを起動します。

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

10. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーの起動後、ログファイル **<broker_instance_dir>/log/artemis.log** を開くと、以下のような 2 行があります。ブローカーの稼働後にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.2.4.3. Windows での 7.2.x から 7.3.0 へのアップグレード

手順

1. AMQ Broker アーカイブのダウンロード の手順に従って、Red Hat カスタマーポータルから必要なアーカイブを ダウンロードします。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。
4. 以下のコマンドを入力してブローカーが実行している場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。
   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。
   b. 同じウィンドウを右クリックし、**Paste** を選択します。
6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** にあるログファイルの最後に、以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集して、アーカイブを抽出した際に作成された新しいディレクトリーに、**ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. **<broker_instance_dir>\etc\artemis.profile.cmd** 設定ファイルを編集し、適切なログマネージャーのバージョンを参照するように JAVA_ARGS 環境を設定します。

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

9. **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集し、適切なログマネージャーバージョンを参照するようにブートストラップクラスパスの開始引数を設定します。

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

10. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、ログファイル **<broker_instance_dir>\log\artemis.log** を開くと、以下のような行があります。ブローカーの稼働後にログに表示される新しいバージョン番号に注意してください。

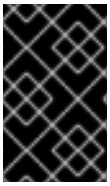
```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.7.0.redhat-00054 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.2.5. ブローカーインスタンスの 7.3.0 から 7.4.0 へのアップグレード

以下のサブセクションでは、異なるオペレーティングシステムの 7.3.0 ブローカーインスタンスを 7.4.0 にアップグレードする方法を説明します。



重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux での 7.3.0 から 7.4.0 へのアップグレード
- Windows での 7.3.0 から 7.4.0 へのアップグレード

2.2.5.1. Linux での 7.3.0 から 7.4.0 へのアップグレード

注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集してください。

   a. アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

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

   b. **JAVA_ARGS** プロパティーを編集します。ログマネージャーに依存するファイルを参照するブートストラップクラスパス引数を追加します。

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

9. **<broker_instance_dir>/etc/bootstrap.xml** 設定ファイルを編集します。**<web>** 設定要素で、AMQ Broker の metrics プラグインファイルへの参照を追加します。

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

10. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

    ```
    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]
    ```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.2.5.2. Windows での 7.3.0 から 7.4.0 へのアップグレード

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。

2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。

3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。

4. ブローカーが実行されている場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。

   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。

   b. 同じウィンドウを右クリックし、**Paste** を選択します。

6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. **<broker_instance_dir>\etc\artemis.profile.cmd** 設定ファイルを編集します。適切なログマネージャーバージョンと依存するファイルを参照するように JAVA_ARGS 環境変数を設定します。

```
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. **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集します。正しいログマネージャーバージョンと依存するファイルを参照するように、ブートストラップクラスパスの開始引数を設定します。

```
<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. **<broker_instance_dir>\etc\bootstrap.xml** 設定ファイルを編集します。**<web>** 設定要素で、AMQ Broker の metrics プラグインファイルへの参照を追加します。

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

11. アップグレードされたブローカーを起動します。

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

12. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

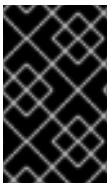
関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.3. ブローカーインスタンスの 7.4.0 から 7.4.X へのアップグレード

重要

AMQ Broker 7.4 は、Long Term Support (LTS) リリースバージョンとして指定されています。バグ修正およびセキュリティーアドバイザリーは、少なくとも 12 カ月間、一連のマイクロリリース (7.4.1、7.4.2 など) で AMQ Broker 7.4 で利用可能になります。つまり、新しいマイナーリリースにアップグレードしなくても、AMQ Broker の最新のバグ修正およびセキュリティーアドバイザリーを取得できます。詳細は、Long Term Support for AMQ Broker を参照してください。



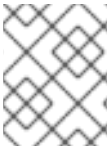
重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

以下のサブセクションでは、異なるオペレーティングシステムの 7.4.0 ブローカーインスタンスを 7.4.x にアップグレードする方法を説明します。

- Linux 上の 7.4.0 から 7.4.x へのアップグレード
- Windows 上の 7.4.0 から 7.4.x へのアップグレード

2.3.1. Linux 上の 7.4.0 から 7.4.x へのアップグレード



注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

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

9. アップグレードされたブローカーを起動します。

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

10. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.3.2. Windows 上の 7.4.0 から 7.4.x へのアップグレード

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。
4. ブローカーが実行されている場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。
   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。
   b. 同じウィンドウを右クリックし、**Paste** を選択します。
6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. アップグレードされたブローカーを起動します。

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

9. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.4. ブローカーインスタンスの 7.4.X から 7.5.0 へのアップグレード

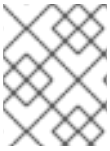
以下のサブセクションでは、異なるオペレーティングシステムの 7.4.x ブローカーインスタンスを 7.5.0 にアップグレードする方法を説明します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux での 7.4.x から 7.5.0 へのアップグレード
- Windows 上の 7.4.x から 7.5.0 へのアップグレード

2.4.1. Linux での 7.4.x から 7.5.0 へのアップグレード



注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集してください。
   a. アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

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

b. **JAVA_ARGS** プロパティーを編集します。ログマネージャーに依存するファイルを参照するブートストラップクラスパス引数を追加します。

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

9. アップグレードされたブローカーを起動します。

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

10. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.4.2. Windows 上の 7.4.x から 7.5.0 へのアップグレード

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。
4. ブローカーが実行されている場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。
   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。

b. 同じウィンドウを右クリックし、**Paste** を選択します。

6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. **<broker_instance_dir>\etc\artemis.profile.cmd** 設定ファイルを編集します。正しいログマネージャーバージョンと依存ファイルを参照するように、**JAVA_ARGS** 環境変数を設定します。

```
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. **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集します。正しいログマネージャーバージョンと依存するファイルを参照するように、ブートストラップクラスパスの開始引数を設定します。

```
<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. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後に

カスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.5. 7.5.0 から 7.6.0 へのブローカーインスタンスのアップグレード

以下のサブセクションでは、異なるオペレーティングシステムの 7.5.0 ブローカーインスタンスを 7.6.0 にアップグレードする方法を説明します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux での 7.5.0 から 7.6.0 へのアップグレード
- Windows での 7.5.0 から 7.6.0 へのアップグレード

2.5.1. Linux での 7.5.0 から 7.6.0 へのアップグレード

注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集してください。

   a. アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

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

   b. **JAVA_ARGS** プロパティーを編集します。ログマネージャーに依存するファイルを参照するブートストラップクラスパス引数を追加します。

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

9. アップグレードされたブローカーを起動します。

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

10. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティー

を更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.5.2. Windows での 7.5.0 から 7.6.0 へのアップグレード

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。
4. ブローカーが実行されている場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。
   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。
   b. 同じウィンドウを右クリックし、**Paste** を選択します。
6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. **<broker_instance_dir>\etc\artemis.profile.cmd** 設定ファイルを編集します。正しいログマネージャーバージョンと依存ファイルを参照するように、**JAVA_ARGS** 環境変数を設定します。

```
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. **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集します。正しいログマネージャーバージョンと依存するファイルを参照するように、ブートストラップクラスパスの開始引数を設定します。

```
<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. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.6. ブローカーインスタンスの 7.6.0 から 7.7.0 へのアップグレード

以下のサブセクションでは、異なるオペレーティングシステムの 7.6.0 ブローカーインスタンスを 7.7.0 にアップグレードする方法を説明します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux で 7.6.0 から 7.7.0 へのアップグレード
- Windows で 7.6.0 から 7.7.0 へのアップグレード

2.6.1. Linux で 7.6.0 から 7.7.0 へのアップグレード

注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。

2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集してください。

   a. アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。以下に例を示します。

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

   b. **JAVA_ARGS** プロパティーを探します。以下に示すように、ブートストラップクラスパスの引数が、ログマネージャーの依存するファイルに必要なバージョンを参照することを確認します。

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

9. **<broker_instance_dir>/etc/logging.properties** 設定ファイルを編集します。

a. 設定する追加のロガーのリストに、AMQ Broker 7.7.0 で追加された **org.apache.activemq.audit.resource** リソースロガーを含めます。

```
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. **Console ハンドラー設定** セクションの前に、リソースロガーのデフォルト設定を追加します。

```
..

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. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.6.2. Windows で 7.6.0 から 7.7.0 へのアップグレード

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。
4. ブローカーが実行されている場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。
   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。
   b. 同じウィンドウを右クリックし、**Paste** を選択します。
6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. **<broker_instance_dir>\etc\artemis.profile.cmd** 設定ファイルを編集します。以下に示すように、**JAVA_ARGS** 環境変数が、ログマネージャーおよび依存するファイルの正しいバージョンを参照することを確認します。

```
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. **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集します。以下に示すように、ブートストラップクラスパスの開始引数がログマネージャーおよび依存するファイルの正しいバージョンを参照することを確認します。

```
<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. **<broker_instance_dir>\etc\logging.properties** 設定ファイルを編集します。
    a. 設定する追加のロガーのリストに、AMQ Broker 7.7.0 で追加された **org.apache.activemq.audit.resource** リソースロガーを含めます。

```
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.aud
```

```
it.base,org.apache.activemq.audit.message,org.apache.activemq.audit.resource
```

b. **Console ハンドラー設定** セクションの前に、リソースロガーのデフォルト設定を追加します。

```
..

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. アップグレードされたブローカーを起動します。

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

12. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.7. ブローカーインスタンスの 7.7.0 から 7.8.0 へのアップグレード

以下のサブセクションでは、異なるオペレーティングシステムの 7.7.0 ブローカーインスタンスを 7.8.0 にアップグレードする方法を説明します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux で 7.7.0 から 7.8.0 へのアップグレード
- Windows で 7.7.0 から 7.8.0 へのアップグレード

2.7.1. Linux で 7.7.0 から 7.8.0 へのアップグレード

注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集してください。

a. アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。以下に例を示します。

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

b. **JAVA_ARGS** プロパティーを探します。以下に示すように、ブートストラップクラスパスの引数が、ログマネージャーの依存するファイルに必要なバージョンを参照することを確認します。

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

9. **<broker_instance_dir>/etc/bootstrap.xml** 設定ファイルを編集します。**web** 要素で、7.8 の AMQ 管理コンソールに必要な **.war** ファイルの名前を更新します。

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

10. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.7.2. Windows で 7.7.0 から 7.8.0 へのアップグレード

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。

2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。

3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。

4. ブローカーが実行されている場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。

   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。

   b. 同じウィンドウを右クリックし、**Paste** を選択します。

6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. **<broker_instance_dir>\etc\artemis.profile.cmd** 設定ファイルを編集します。以下に示すように、**JAVA_ARGS** 環境変数が、ログマネージャーおよび依存するファイルの正しいバージョンを参照することを確認します。

```
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. **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集します。以下に示すように、ブートストラップクラスパスの開始引数がログマネージャーおよび依存するファイルの正しいバージョンを参照することを確認します。

```
<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. **<broker_instance_dir>\etc\bootstrap.xml** 設定ファイルを編集します。**web** 要素で、7.8 の AMQ 管理コンソールに必要な **.war** ファイルの名前を更新します。

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

```
    ...
</web>
```

11. アップグレードされたブローカーを起動します。

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

12. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.8. ブローカーインスタンスの 7.8.X から 7.9.Y へのアップグレード

以下のサブセクションでは、異なるオペレーティングシステムの 7.8.x ブローカーインスタンスを 7.9.x にアップグレードする方法を説明します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux の 7.8.x から 7.9.x へのアップグレード
- Windows の 7.8.x から 7.9.x へのアップグレード

2.8.1. Linux の 7.8.x から 7.9.x へのアップグレード

注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。

2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集してください。

   a. アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。以下に例を示します。

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

   b. **JAVA_ARGS** プロパティーを探します。以下に示すように、ブートストラップクラスパスの引数が、ログマネージャーの依存するファイルに必要なバージョンを参照することを確認します。

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

9. **<broker_instance_dir>/etc/bootstrap.xml** 設定ファイルを編集します。**web** 要素で、7.9 の AMQ 管理コンソールに必要な **.war** ファイルの名前を更新します。

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

10. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.8.2. Windows の 7.8.x から 7.9.x へのアップグレード

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。
4. ブローカーが実行されている場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。

   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy**を選択します。

   b. 同じウィンドウを右クリックし、**Paste** を選択します。

6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

```
ARTEMIS_HOME=<install_dir>
```

8. **<broker_instance_dir>\etc\artemis.profile.cmd** 設定ファイルを編集します。以下に示すように、**JAVA_ARGS** 環境変数が、ログマネージャーおよび依存するファイルの正しいバージョンを参照することを確認します。

```
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. **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集します。以下に示すように、ブートストラップクラスパスの開始引数がログマネージャーおよび依存するファイルの正しいバージョンを参照することを確認します。

```
<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. **<broker_instance_dir>\etc\bootstrap.xml** 設定ファイルを編集します。**web** 要素で、7.9 の AMQ 管理コンソールに必要な **.war** ファイルの名前を更新します。

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

11. アップグレードされたブローカーを起動します。

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

12. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.9. ブローカーインスタンスの 7.9.X から 7.10.X へのアップグレード

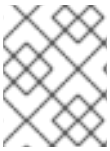
以下のサブセクションでは、異なるオペレーティングシステムの 7.9.x ブローカーインスタンスを 7.10.x にアップグレードする方法を説明します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux の 7.9.x から 7.10.x へのアップグレード
- Windows の 7.9.x から 7.10.x へのアップグレード

2.9.1. Linux の 7.9.x から 7.10.x へのアップグレード



注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

前提条件

- AMQ Broker 7.11 を実行するには、少なくとも Java バージョン 11 が必要です。各 AMQ Broker ホストが Java バージョン 11 以降を実行していることを確認します。サポートされている設定の詳細は、Red Hat AMQ Broker 7 Supported Configurations を参照してください。
- AMQ Broker 7.9 がメッセージデータをデータベースに保存するように設定されている場合、HOLDER_EXPIRATION_TIME 列のデータ型はノードマネージャーデータベーステーブルの **timestamp** です。AMQ Broker 7.11 では、列のデータ型が number に変更されました。AMQ Broker 7.11 にアップグレードする前に、ノードマネージャーテーブルを削除する、つまりデータベースから削除する必要があります。テーブルを削除した後、アップグレードされたブローカーを再起動すると、新しいスキーマでテーブルが再作成されます。共有ストアの高可用性 (HA) 設定では、ノードマネージャーテーブルはブローカー間で共有されます。したがって、

テーブルを削除する前に、テーブルを共有するすべてのブローカーが停止していることを確認する必要があります。次の例では、**NODE_MANAGER_TABLE** というノードマネージャーテーブルを削除します。

```
DROP TABLE NODE_MANAGER_TABLE
```

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. アーカイブの所有者を、AMQ Broker インストールが含まれるのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

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

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

7. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. **<broker_instance_dir>/etc/artemis.profile** 設定ファイルを編集してください。
   a. アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。以下に例を示します。

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

9. **<broker_instance_dir>/etc/bootstrap.xml** 設定ファイルを編集します。
Web 要素で、7.10 の AMQ 管理コンソールに必要な **.war** ファイルの名前を更新します。

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

+ **broker xmlns** 要素で、スキーマ値を **"http://activemq.org/schema"** から **"http://activemq.apache.org/schema"** に変更します。

+

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

1. **<broker_instance_dir>/etc/management.xml** ファイルを編集します。
management-context xmlns 要素で、スキーマ値を **"http://activemq.org/schema"** から **"http://activemq.apache.org/schema"** に変更します。

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

2. アップグレードされたブローカーを起動します。

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

3. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.9.2. Windows の 7.9.x から 7.10.x へのアップグレード

前提条件

- AMQ Broker 7.11 を実行するには、少なくとも Java バージョン 11 が必要です。各 AMQ Broker ホストが Java バージョン 11 以降を実行していることを確認します。サポートされている設定の詳細は、Red Hat AMQ Broker 7 Supported Configurations を参照してください。

- AMQ Broker 7.9 がメッセージデータをデータベースに保存するように設定されている場合、HOLDER_EXPIRATION_TIME 列のデータ型はノードマネージャーデータベーステーブルの **timestamp** です。AMQ Broker 7.11 では、列のデータ型が number に変更されました。AMQ Broker 7.11 にアップグレードする前に、ノードマネージャーテーブルを削除する、つまりデータベースから削除する必要があります。テーブルを削除した後、アップグレードされたブローカーを再起動すると、新しいスキーマでテーブルが再作成されます。共有ストアの高可用性 (HA) 設定では、ノードマネージャーテーブルはブローカー間で共有されます。したがって、テーブルを削除する前に、テーブルを共有するすべてのブローカーが停止していることを確認する必要があります。次の例では、**NODE_MANAGER_TABLE** というノードマネージャーテーブルを削除します。

```
DROP TABLE NODE_MANAGER_TABLE
```

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。

2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。

3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。

4. ブローカーが実行されている場合は停止します。

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

5. ファイルマネージャーを使用してブローカーをバックアップします。

   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。

   b. 同じウィンドウを右クリックし、**Paste** を選択します。

6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

   ```
   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. **<broker_instance_dir>\etc\artemis.profile.cmd** および **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集してください。アーカイブの抽出時に作成された新しいディレクトリーに **ARTEMIS_HOME** プロパティーを設定します。

   ```
   ARTEMIS_HOME=<install_dir>
   ```

8. **<broker_instance_dir>\etc\artemis.profile.cmd** 設定ファイルを編集します。以下に示すように、**JAVA_ARGS** 環境変数が、ログマネージャーおよび依存するファイルの正しいバージョンを参照することを確認します。

```
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. **<broker_instance_dir>\bin\artemis-service.xml** 設定ファイルを編集します。以下に示すように、ブートストラップクラスパスの開始引数がログマネージャーおよび依存するファイルの正しいバージョンを参照することを確認します。

```
<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. **<broker_instance_dir>\etc\bootstrap.xml** 設定ファイルを編集します。
Web 要素で、7.10 の AMQ 管理コンソールに必要な **.war** ファイルの名前を更新します。

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

broker xmlns 要素で、スキーマ値を **"http://activemq.org/schema"** から **"http://activemq.apache.org/schema"** に変更します。

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

11. **<broker_instance_dir>/etc/management.xml** ファイルを編集します。
management-context xmlns 要素で、スキーマ値を **"http://activemq.org/schema"** から **"http://activemq.apache.org/schema"** に変更します。

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

12. アップグレードされたブローカーを起動します。

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

13. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

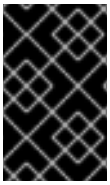
```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.21.0.redhat-00025 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.10. ブローカーインスタンスの 7.10.X から 7.11.X へのアップグレード

次のサブセクションでは、さまざまなオペレーティングシステムで 7.10.x ブローカーインスタンスを 7.11.x にアップグレードする方法について説明します。



重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux での 7.10.x から 7.11.x へのアップグレード
- Windows での 7.10.x から 7.11.x へのアップグレード

2.10.1. Linux での 7.10.x から 7.11.x へのアップグレード

注記

ダウンロードするアーカイブの名前は、以下の例で使用されているものとは異なる場合があります。

前提条件

- AMQ Broker 7.11 を実行するには、少なくとも Java バージョン 11 が必要です。各 AMQ Broker ホストが Java バージョン 11 以降を実行していることを確認します。サポートされている設定の詳細は、Red Hat AMQ Broker 7 Supported Configurations を参照してください。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. ダウンロードしたアーカイブの所有者を、アップグレードする AMQ Broker インストールを所有するのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

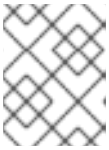
```
sudo chown amq-broker:amq-broker amq-broker-7.x.x-bin.zip
```

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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



注記

アーカイブの内容は、現在のディレクトリーの **apache-artemis-2.28.0.redhat-00019** というディレクトリーに展開されます。

5. ブローカーが実行されている場合は停止します。

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

6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
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. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

8. 圧縮アーカイブの内容を展開したディレクトリーに移動します。

```
cd /opt/redhat/apache-artemis-2.33.0.redhat-00010/bin
```

9. **artemis upgrade** コマンドを実行して、既存のブローカーをアップグレードします。次の例では、**/var/opt/amq-broker/mybroker** ディレクトリー内のブローカーインスタンスをアップグレードします。

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

artemis upgrade コマンドは、ブローカーをアップグレードするための次の手順を完了します。

- アップグレードするブローカーのブローカーインスタンスディレクトリーの **old-config-bkp.<n>** サブディレクトリーに変更する各ファイルのバックアップを作成します。
- **<broker_instance_dir>/etc/artemis.profile** ファイル内の **ARTEMIS_HOME** プロパティーをアーカイブの展開時に作成された新しいディレクトリーに設定します。

- 以前のバージョンで使用されていた JBoss Logging フレームワークの代わりに、AMQ Broker 7.11 にバンドルされている Apache Log4j 2 ログユーティリティーを使用するように **<broker_instance_dir>bin/artemis** スクリプトを更新します。

- JBoss によって使用される既存の **<broker_instance_dir>/etc/logging.properties** ファイルを削除し、Apache Log4j 2 ログユーティリティー用の新しい **<broker_instance_dir>/etc/log4j2.properties** ファイルを作成します。

10. AMQ Broker に含まれる Prometheus メトリックプラグインが 7.10.x で有効になっている場合は、プラグインのクラス名を **org.apache.activemq.artemis.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin** から **com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin** に変更します。これは AMQ Broker 7.11 のプラグインの新しいクラス名です。

    a. **<broker_instance_dir>/etc/broker.xml** 設定ファイルを開きます。

    b. **<metrics>** 要素の **<plugin>** サブ要素で、プラグインクラス名を **com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin** に更新します。

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

    c. **broker.xml** 設定ファイルを保存します。

11. アップグレードされたブローカーを起動します。

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

12. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような行を見つけます。ブローカーの起動後にログに表示される新しいバージョン番号に注意してください。

    ```
    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]
    ```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。

- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティー

を更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** ディレクトリーおよび **data/** ディレクトリーにのみ保存できました。

2.10.2. Windows での 7.10.x から 7.11.x へのアップグレード

前提条件

- AMQ Broker 7.11 を実行するには、少なくとも Java バージョン 11 が必要です。各 AMQ Broker ホストが Java バージョン 11 以降を実行していることを確認します。サポートされている設定の詳細は、Red Hat AMQ Broker 7 Supported Configurations を参照してください。

手順

1. AMQ Broker アーカイブのダウンロード に記載されている手順に従って、AMQ Broker アーカイブをダウンロードします。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。

注記

アーカイブの内容は、現在のフォルダー内の **apache-artemis-2.28.0.redhat-00019** というフォルダーに展開されます。

4. ブローカーが実行されている場合は停止します。

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

5. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
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. ファイルマネージャーを使用してブローカーをバックアップします。
   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。
   b. 同じウィンドウを右クリックし、**Paste** を選択します。
7. 圧縮アーカイブの内容を展開したディレクトリーに移動します。以下に例を示します。

```
cd \redhat\amq-broker\apache-artemis-2.33.0.redhat-00010\bin
```

8. **artemis upgrade** コマンドを実行して、既存のブローカーをアップグレードします。次の例では、**C:\redhat\amq-broker\mybroker** ディレクトリー内のブローカーインスタンスをアップグレードします。

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

artemis upgrade コマンドは、ブローカーをアップグレードするための次の手順を完了します。

- アップグレードするブローカーのブローカーインスタンスディレクトリーの **old-config-bkp.<n>** サブディレクトリーに変更する各ファイルのバックアップを作成します。
- **<broker_instance_dir>\etc\artemis.cmd.profile** ファイル内の **ARTEMIS_HOME** プロパティーをアーカイブの展開時に作成された新しいディレクトリーに設定します。
- **<broker_instance_dir>\bin\artemis.cmd** スクリプトを更新して、以前のバージョンで使用された JBoss Logging フレームワークの代わりに、AMQ Broker 7.11 にバンドルされている Apache Log4j 2 ログユーティリティーを使用します。
- JBoss によって使用される既存の **<broker_instance_dir>\etc\logging.properties** ファイルを削除し、Apache Log4j 2 ログユーティリティー用の新しい **<broker_instance_dir>\etc\log4j2.properties** ファイルを作成します。

9. AMQ Broker に含まれる Prometheus メトリックプラグインが 7.10.x で有効になっている場合は、プラグインのクラス名を **org.apache.activemq.artemis.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin** から **com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin** に変更します。これは 7.11 のプラグインの新しいクラス名です。

   a. **<broker_instance_dir>\etc\broker.xml** 設定ファイルを開きます。

   b. **<metrics>** 要素の **<plugin>** サブ要素で、プラグインクラス名を **com.redhat.amq.broker.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin** に更新します。

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

   c. **broker.xml** 設定ファイルを保存します。

10. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
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]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

2.11. ブローカーインスタンスの 7.10.X から 7.11.X へのアップグレード

次のサブセクションでは、さまざまなオペレーティングシステムで 7.10.x ブローカーインスタンスを 7.11.x にアップグレードする方法について説明します。

重要

AMQ Broker 7.1.0 以降では、デフォルトでローカルホストからのみ AMQ 管理コンソールにアクセスできます。コンソールにリモートアクセスを設定する方法は、Configuring local and remote access to AMQ Management Console を参照してください。

- Linux での 7.10.x から 7.11.x へのアップグレード
- Windows での 7.10.x から 7.11.x へのアップグレード

2.11.1. Linux での 7.10.x から 7.11.x へのアップグレード

前提条件

- AMQ Broker 7.11 を実行するには、少なくとも Java バージョン 11 が必要です。各 AMQ Broker ホストが Java バージョン 11 以降を実行していることを確認します。サポートされている設定の詳細は、Red Hat AMQ 7 でサポートされている設定 [Red Hat AMQ Broker 7 でサポートされている設定] を参照してください。

手順

1. Red Hat カスタマーポータルから必要なアーカイブをダウンロードします。AMQ Broker アーカイブのダウンロード に記載されている手順に従います。
2. ダウンロードしたアーカイブの所有者を、アップグレードする AMQ Broker インストールを所有するのと同じユーザーに変更します。以下の例では、**amq-broker** というユーザーを設定しています。

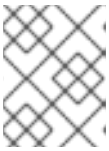
```
sudo chown amq-broker:amq-broker amq-broker-7.x.x-bin.zip
```

3. AMQ Broker の元のインストール時に作成されたディレクトリーにアーカイブを移動します。以下の例では、**/opt/redhat** を使用しています。

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

4. ディレクトリーの所有者は、圧縮アーカイブのコンテンツをデプロイメントします。以下の例では、ユーザー **amq-broker** は **unzip** コマンドを使用してアーカイブをデプロイメントします。

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



注記

アーカイブの内容は、現在のディレクトリーの **apache-artemis-2.28.0.redhat-00019** というディレクトリーに展開されます。

5. ブローカーが実行されている場合は停止します。

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

6. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>/log/artemis.log** ファイルの最後に以下のような行が表示されます。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.28.0.redhat-00019 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

7. 現在のユーザーのホームディレクトリーにコピーして、ブローカーのインスタンスディレクトリーをバックアップします。

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

8. 圧縮アーカイブの内容を展開したディレクトリーに移動します。

```
cd /opt/redhat/apache-artemis-2.33.0.redhat-00010/bin
```

9. **artemis upgrade** コマンドを実行して、既存のブローカーをアップグレードします。次の例では、**/var/opt/amq-broker/mybroker** ディレクトリー内のブローカーインスタンスをアップグレードします。

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

artemis upgrade コマンドは、ブローカーをアップグレードするための次の手順を完了します。

- アップグレードするブローカーのブローカーインスタンスディレクトリーの **old-config-bkp.<n>** サブディレクトリーに変更する各ファイルのバックアップを作成します。
- **<broker_instance_dir>/etc/artemis.profile** ファイル内の **ARTEMIS_HOME** プロパティーをアーカイブの展開時に作成された新しいディレクトリーに設定します。
- 以前のバージョンで使用されていた JBoss Logging フレームワークの代わりに、AMQ Broker 7.11 にバンドルされている Apache Log4j 2 ログユーティリティーを使用するように **<broker_instance_dir>bin/artemis** スクリプトを更新します。

- JBoss によって使用される既存の **<broker_instance_dir>/etc/logging.properties** ファイルを削除し、Apache Log4j 2 ログユーティリティー用の新しい **<broker_instance_dir>/etc/log4j2.properties** ファイルを作成します。

10. アップグレードされたブローカーを起動します。

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

11. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動したら、**<broker_instance_dir>/log/artemis.log** ファイルを開きます。以下のような行を見つけます。ブローカーの起動後にログに表示される新しいバージョン番号に注意してください。

```
2023-02-08 20:53:50,128 INFO  [org.apache.activemq.artemis.integration.bootstrap]
AMQ101000: Starting ActiveMQ Artemis Server version 2.33.0.redhat-00010
2023-02-08 20:53:51,077 INFO  [org.apache.activemq.artemis.core.server] AMQ221001:
Apache ActiveMQ Artemis Message Broker version 2.33.0.redhat-00010 [0.0.0.0,
nodeID=be02a2b2-3e42-11ec-9b8a-4c796e887ecb]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。**<broker_instance_dir>/etc/artemis.profile** ファイルで、ブローカーインスタンスの作成後のカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。AMQ Broker の以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **etc/** および **data/** ディレクトリーにのみ保存できました。

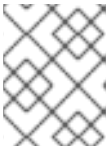
2.11.2. Windows での 7.10.x から 7.11.x へのアップグレード

前提条件

- AMQ Broker 7.11 を実行するには、少なくとも Java バージョン 11 が必要です。各 AMQ Broker ホストが Java バージョン 11 以降を実行していることを確認します。サポートされている設定の詳細は、Red Hat AMQ Broker 7 Supported Configurations を参照してください。

手順

1. AMQ Broker アーカイブのダウンロード に記載されている手順に従って、AMQ Broker アーカイブをダウンロードします。
2. ファイルマネージャーを使用して、アーカイブを AMQ Broker の最後のインストール時に作成したフォルダーに移動します。
3. アーカイブの内容をデプロイメントします。.zip ファイルを右クリックし、**Extract All** を選択します。



注記

アーカイブの内容は、現在のフォルダー内の **apache-artemis-2.28.0.redhat-00019** というフォルダーに展開されます。

4. ブローカーが実行されている場合は停止します。

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

5. (オプション) ブローカーの現行バージョンをメモします。ブローカーが停止すると、**<broker_instance_dir>\log\artemis.log** の末尾に以下のような行が表示されます。

```
INFO  [org.apache.activemq.artemis.core.server] AMQ221002: Apache ActiveMQ Artemis Message Broker version 2.28.0.redhat-00019[4782d50d-47a2-11e7-a160-9801a793ea45] stopped, uptime 28 minutes
```

6. ファイルマネージャーを使用してブローカーをバックアップします。

   a. **<broker_instance_dir>** フォルダーを右クリックし、**Copy** を選択します。

   b. 同じウィンドウを右クリックし、**Paste** を選択します。

7. 圧縮アーカイブの内容を展開したディレクトリーに移動します。以下に例を示します。

```
cd \redhat\amq-broker\apache-artemis-2.33.0.redhat-00010\bin
```

8. **artemis upgrade** コマンドを実行して、既存のブローカーをアップグレードします。次の例では、**C:\redhat\amq-broker\mybroker** ディレクトリー内のブローカーインスタンスをアップグレードします。

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

   artemis upgrade コマンドは、ブローカーをアップグレードするための次の手順を完了します。

   - アップグレードするブローカーのブローカーインスタンスディレクトリーの **old-config-bkp.<n>** サブディレクトリーに変更する各ファイルのバックアップを作成します。
   - **<broker_instance_dir>\etc\artemis.cmd.profile** ファイル内の **ARTEMIS_HOME** プロパティーをアーカイブの展開時に作成された新しいディレクトリーに設定します。
   - **<broker_instance_dir>\bin\artemis.cmd** スクリプトを更新して、以前のバージョンで使用された JBoss Logging フレームワークの代わりに、AMQ Broker 7.11 にバンドルされている Apache Log4j 2 ログユーティリティーを使用します。
   - JBoss によって使用される既存の **<broker_instance_dir>\etc\logging.properties** ファイルを削除し、Apache Log4j 2 ログユーティリティー用の新しい **<broker_instance_dir>\etc\log4j2.properties** ファイルを作成します。

9. アップグレードされたブローカーを起動します。

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

10. (オプション) ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2

つの行を見つけます。ブローカーの稼働時にログに表示される新しいバージョン番号に注意してください。

```
2023-02-08 20:53:50,128 INFO  [org.apache.activemq.artemis.integration.bootstrap]
AMQ101000: Starting ActiveMQ Artemis Server version 2.33.0.redhat-00010
2023-02-08 20:53:51,077 INFO  [org.apache.activemq.artemis.core.server] AMQ221001:
Apache ActiveMQ Artemis Message Broker version 2.33.0.redhat-00010 [0.0.0.0,
nodeID=be02a2b2-3e42-11ec-9b8a-4c796e887ecb]
```

関連情報

- ブローカーのインスタンス作成に関する詳細は、ブローカーインスタンスの作成を参照してください。
- ブローカーインスタンスの設定ファイルおよびデータを、ブローカーインスタンスのディレクトリー外の場所を含む、カスタムディレクトリーに格納できるようになりました。In the **<broker_instance_dir>\etc\artemis.profile** ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI** プロパティーを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカーインスタンスのディレクトリー内の **\etc** ディレクトリーおよび **\data** ディレクトリーにのみ保存できました。

第3章 コマンドラインインターフェイスの使用

コマンドラインインターフェイス (CLI) は、対話式ターミナルを使用してメッセージブローカーとの対話を可能にします。CLI を使用すると、ブローカーのアクションを管理したり、メッセージを設定したり、ファイルにユーザーとロールを追加したり、その他の便利なコマンドを入力したりできます。

CLI を使用して、Bash または Zsh シェル、あるいはカスタム **artemis** シェルでブローカーと対話できます。各シェルで同じコマンドが使用できます。デフォルトでは、**artemis** シェルにはコマンドとコマンドパラメーターの自動補完機能が組み込まれています。CLI コマンドとコマンドパラメーターの自動補完を Bash または Zsh シェルに追加することもできます。

3.1. ARTEMIS SHELL で CLI を使用する

artemis シェルインターフェイスは、**artemis** コマンドで使用できるコマンドとパラメーターの自動補完機能を提供します。シェルは、ブローカー URI やログイン認証情報などの提供された接続情報を、同じシェルセッションで実行する後続のコマンドにも再利用します。

手順

1. インストール中に作成したユーザーアカウントに切り替えます。例:

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

2. シェルを起動するには、**artemis** コマンドを使用します。例:

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

artemis shell を起動するときにブローカー接続の詳細を指定する場合は、**artemis shell** コマンドを使用します。以下に例を示します。

```
$ /var/opt/amq-broker/mybroker/bin/artemis shell --user myuser --password mypassword --
url tcp://localhost:61616
```

提供された認証情報と URI は、ブローカーによる認証を必要とするシェルで実行される後続のコマンドに再利用されます。

3. Tab キーを押すと、シェル内の任意の場所に自動補完情報が表示されます。以下に例を示します。

- **artemis** シェルで使用できるコマンドのリストを表示するには、シェルプロンプトで Tab キーを押します。
- コマンドのサブコマンドを表示するには、コマンドの後に Tab キーを押します。たとえば、**check** と入力して Tab キーを押すと、**check** コマンドのサブコマンドが表示されます。自動補完情報によると、**check** コマンドは **cluster**、**node**、**queue** の 3 つのサブコマンドをサポートしています。
- サブコマンドの自動補完情報を表示するには、サブコマンドの後に Tab キーを押します。たとえば、**check node** と入力して Tab キーを押します。それ以上の自動補完情報が表示されない場合は、**--** (例: **check node --)** と入力し、Tab キーを押して、コマンドに使用できるパラメーターを表示します。

3.2. BASH または ZSH シェルでの自動補完の設定

Bash または Zsh シェルで CLI コマンドとコマンドパラメーターの自動補完を設定できます。

手順

1. インストール中に作成したユーザーアカウントに切り替えます。例:

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

2. 次のコマンドを使用して、**auto-complete-artemis.sh** というコマンド自動補完スクリプトを作成します。

```
$ /var/opt/amq-broker/mybroker/bin/artemis auto-complete
```

3. **auto-complete-artemis.sh** を実行するには、次のコマンドを使用します。

```
$ source /var/opt/amq-broker/mybroker/bin/artemis/auto-complete-artemis.sh
```

4. **artemis** コマンドの自動補完情報を表示するには、**/var/opt/amq-broker/mybroker/artemis** と入力して Tab キーを押します。**artemis** コマンドで使用できるコマンドのリストが表示されます。

5. 使用可能なコマンドの自動補完情報を表示するには、コマンドの後に Tab キーを押します。たとえば、**/var/opt/amq-broker/mybroker/artemis check node** と入力し、Tab キーを押します。それ以上の自動補完情報が表示されない場合は、**--** (例: **check node --)** と入力し、Tab キーを押して、コマンドに使用できるパラメーターを表示します。

3.3. ブローカーインスタンスの起動

ブローカーインスタンスは、ログやデータファイルなどのすべての設定およびランタイムデータが含まれるディレクトリーです。ランタイムデータは一意のブローカープロセスに関連付けられます。

artemis スクリプトを Linux サービスまたは Windows サービスとして使用することで、フォアグラウンドでブローカーを起動することができます。

3.3.1. ブローカーインスタンスの起動

ブローカーインスタンスの作成後に、**artemis run** コマンドを使用して起動します。

手順

1. インストール時に作成したユーザーのアカウントに切り替えます。

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

2. **artemis run** コマンドを使用してブローカーインスタンスを起動します。

```
$ /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
...
```

ブローカーが起動し、以下の情報が含まれるログ出力が表示されます。

- トランザクションログとクラスター設定の場所。
- メッセージの永続性に使用されるジャーナルのタイプ (この場合は AIO)。
- クライアント接続を許可できる URI。
  デフォルトでは、ポート 61616 はサポートされるプロトコル (CORE、MQTT、AMQP、STOMP、HORNETQ、および OPENWIRE) からの接続を受け入れることができます。各プロトコルには個別のポートもあります。
- Web コンソールは、http://localhost:8161 から入手できます。
- Jolokia サービス (JMX over REST) には、http://localhost:8161/jolokia からアクセスできます。

3.3.2. Linux サービスとしてブローカーの起動

ブローカーが Linux にインストールされている場合は、サービスとして実行できます。

手順

1. **/etc/systemd/system/** ディレクトリーに **amq-broker.service** ファイルを新規に作成します。
2. 以下のテキストをファイルにコピーします。
   ブローカーインスタンスの作成時に提供される情報に応じて、パスおよびユーザーフィールドを変更します。以下の例では、ユーザー **amq-broker** が、**/var/opt/amq-broker/mybroker/** ディレクトリーの下にインストールされたブローカーサービスを起動します。

```
[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. 端末を開きます。
4. 以下のコマンドを使用してブローカーサービスを有効にします。

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

5. 以下のコマンドを使用してブローカーサービスを実行します。

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

3.3.3. Windows サービスとしてブローカーの起動

ブローカーが Windows にインストールされている場合は、サービスとして実行できます。

手順

1. コマンドプロンプトを開いてコマンドを入力する
2. 以下のコマンドを使用して、ブローカーをサービスとしてインストールします。

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

3. 以下のコマンドを使用してサービスを起動します。

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

4. (オプション) サービスをアンインストールします。

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

3.4. ブローカーインスタンスの停止

ブローカーインスタンスを手動で停止するか、ブローカーを正常にシャットダウンするように設定します。

3.4.1. ブローカーインスタンスの停止

スタンドアロンブローカーを作成し、テストメッセージを生成および消費した後、ブローカーインスタンスを停止できます。

この手順では、ブローカーを手動で停止し、クライアント接続をすべて強制的に閉じます。実稼働環境では、クライアント接続を適切に閉じるようにブローカーを正常に停止するようにブローカーを設定する必要があります。

手順

- **artemis stop** コマンドを使用してブローカーインスタンスを停止します。

```
$ /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!
```

3.4.2. ブローカーインスタンスの正常な停止

手動シャットダウンは、**stop** コマンドを入力すると、すべてのクライアントを強制的に切断します。別の方法として、**graceful-shutdown-enabled** 設定要素を使用して、ブローカーが正常にシャットダウンするように設定します。

graceful-shutdown-enabled が **true** に設定されている場合、**stop** コマンドが入力された後、新しいクライアントの接続は許可されません。ただし、シャットダウンプロセスを開始する前に、既存の接続はクライアント側で閉じることができます。**graceful-shutdown-enabled** のデフォルト値は **false** です。

graceful-shutdown-timeout 設定要素を使用して、接続がブローカー側から強制的に閉じられる前にクライアントが切断する時間の長さをミリ秒単位で設定します。すべての接続が閉じられると、シャットダウンプロセスが開始します。**graceful-shutdown-timeout** を使用する利点の 1 つは、クライアントの接続によるシャットダウンの遅延を防ぐことができることです。**graceful-shutdown-timeout** のデフォルト値は **-1** で、これは、クライアントが切断するまでブローカーが無期限に待機することを意味します。

以下の手順は、タイムアウトを使用する正常なシャットダウンを設定する方法を示しています。

手順

1. **<broker_instance_dir>\etc\broker.xml** 設定ファイルを開きます。
2. **graceful-shutdown-enabled** 設定要素を追加し、値を **true** に設定します。

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

3. **graceful-shutdown-timeout** 設定要素を追加し、タイムアウトの値をミリ秒単位で設定します。以下の例では、**stop** コマンドが実行されてから 30 秒 (**30000** ミリ秒) 後に、クライアント接続が強制的に閉じられます。

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

```
        ...
    </core>
</configuration>
```

3.5. パケットをインターセプトしてメッセージの監査

ブローカーの出入力または終了パケットをインターセプトして、パケットの監査またはメッセージのフィルターを行います。インターセプターは、インターセプトするパケットを変更します。これによりインターセプターは強力になりますが、危険にさらされる可能性もあります。

ビジネス要件を満たすためのインターセプターを開発します。インターセプターはプロトコル固有であるため、適切なインターフェイスを実装する必要があります。

インターセプターは、ブール値を返す **intercept()** メソッドを実装する必要があります。値が **true** の場合、メッセージパケットは続行されます。**false** の場合、プロセスは中止され、他のインターセプターは呼び出されず、メッセージパケットはこれ以上処理されません。

3.5.1. インターセプターの作成

インターセプターは、傍受するパケットを変更できます。独自の受信インターセプターおよび発信インターセプターを作成できます。すべてのインターセプターはプロトコル固有で、サーバーに出入りするパケットに対して呼び出されます。これにより、監査パケットなどのビジネス要件を満たすインターセプターを作成できます。

インターセプターとその依存関係は、ブローカーの Java クラスに配置する必要があります。**<broker_instance_dir>/lib** ディレクトリーは、デフォルトでクラスパスの一部となっているため、使用することができます。

以下の例は、渡された各パケットのサイズをチェックするインターセプターを作成する方法を示しています。

注記

この例では、プロトコルごとに特定のインターフェイスを実装します。

手順

1. 適切なインターフェイスを実装し、その **intercept()** メソッドをオーバーライドします。

   a. AMQP プロトコルを使用している場合は、**org.apache.activemq.artemis.protocol.amqp.broker.AmqpInterceptor** インターフェイスを実装してください。

```
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. Core Protocol を使用している場合、インターセプターは、**org.apache.artemis.activemq.api.core.Interceptor** インターフェイスを実装する必要があります。

```
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. MQTT プロトコルを使用している場合は、**org.apache.activemq.artemis.core.protocol.mqtt.MQTTInterceptor** インターフェイスを実装してください。

```
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. STOMP プロトコルを使用している場合は、**org.apache.activemq.artemis.core.protocol.stomp.StompFrameInterceptor** インターフェイスを実装してください。

```
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;
  }
}
```

3.5.2. インターセプターを使用するためのブローカーの設定

前提条件

- インターセプタークラスを作成し、そのクラス (およびその依存関係) をブローカーの Java クラスパスに追加します。**<broker_instance_dir>/lib** ディレクトリーは、デフォルトでクラスパスに含まれているため、使用することができます。

手順

1. **<broker_instance_dir>/etc/broker.xml** を開きます。
2. ブローカーがインターセプターを使用するように設定するには、**<broker_instance_dir>/etc/broker.xml** に設定を追加します。
   a. インターセプターが着信メッセージを対象としている場合は、その **class-name** を **remoting-incoming-interceptors** のリストに追加します。

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

b. インターセプターが発信メッセージを対象としている場合は、その **class-name** を **remoting-outgoing-interceptors** のリストに追加します。

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

3.5.3. クライアントサイドのインターセプター

クライアントはインターセプターを使用して、クライアントからサーバーに送信したパケットを、またはサーバーがクライアントへインターセプトできます。ブローカー側のインターセプターが **false** の値を返す場合、他のインターセプターは呼び出されず、クライアントは追加のパケットを処理しません。このプロセスは、発信パケットが **blocking** 方式で送信されない限り、透過的に行われます。この場合、呼び出し元に **ActiveMQException** が出力されます。出力された **ActiveMQException** には、**false** 値を返したインターセプターの名前が含まれています。

サーバーでは、クライアントインターセプタークラスとその依存関係を適切にインスタンス化および呼び出すには、クライアントの Java クラスに追加する必要があります。

3.6. ブローカー、キュー、クラスターの健全性を確認する

AMQ Broker には、ブローカートポロジーのブローカーおよびキューでさまざまなヘルスチェックを実行できるコマンドラインユーティリティーが含まれています。ブローカーがクラスター化されている場合は、このユーティリティーを使用してクラスタートポロジーの健全性を確認することもできます。

手順

1. ブローカートポロジーの特定ブローカー (**ノード**) に対して実行できるチェックのリストを参照してください。

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

artemis check node コマンドで使用できるオプションのセットを説明した出力が表示されます。

```
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. たとえば、ローカルブローカーのディスク使用量がブローカーに設定された最大ディスク使用量を下回ることを確認します。

```
$ <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
```

上記の例では、**--diskUsage** オプションに **-1** の値を指定すると、ユーティリティーはブローカーに設定された **最大** ディスク使用量に対してディスクの使用量をチェックすることを意味します。ブローカーの最大ディスク使用量は、**broker.xml** 設定ファイルの **max-disk-usage** パラメーターを使用して設定されます。**max-disk-usage** に指定された値は、ブローカーが消費できる利用可能な物理ディスク領域の割合を表します。

3. ブローカートポロジーの特定キューに対して実行できるチェックのリストを参照してください。

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

artemis check queue コマンドで、使用できるオプションのセットを説明した出力が表示されます。

```
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. このユーティリティーは、1 つのコマンドで複数のオプションを実行できます。たとえば、ローカルブローカーのデフォルトの **helloworld** キューで、1000 個のメッセージの生成、参照、消費を確認するには、以下のコマンドを使用します。

```
$ <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
```

上記の例では、キューチェックの実行時にブローカー URL を指定していないことを確認します。URL を明示的に指定しないと、ユーティリティーはデフォルト値の **tcp://localhost:61616** を使用します。

5. ブローカーがクラスター化されている場合は、クラスタートポロジーに対して実行できるチェックのリストを参照してください。

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

artemis check queue コマンドで、使用できるオプションのセットを説明した出力が表示されます。

```
NAME
        artemis check cluster - Verify if all the nodes on the cluster match the same topology and
time
        configuration.
```

```
SYNOPSIS
        artemis check cluster [--acceptor=<acceptor>]
                [--clientID=<clientID>] [--password=<password>]
                [--protocol=<protocol>] [--silent] [--url=<brokerURL>]
                [--user=<user>] [--variance=<variance>] [--verbose]
OPTIONS
      --acceptor=<acceptor>
          Name used to find the default connection URL on the acceptor list. If an acceptor with
that name cannot
          be found, the command looks for a connector with the same name.

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

        --password <password>
            Password used to connect.

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

        --silent
            Disables all the input options and make a best guess for any required input.

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

        --user <user>
            User used to connect.

        --variance <variance>
            Allowed time variance in milliseconds after which check is deemed to have failed
(default=1000).

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

3.7. コマンドラインツール

AMQ Broker にはコマンドラインインターフェイス (CLI) ツールのセットが含まれるため、メッセージングジャーナルを管理できます。以下の表は、各ツールの名前と対応する説明をリスト表示しています。

ツール	説明
address	ツールグループのアドレス指定 (create/delete/update/show) (例: **./artemis address create**)
ブラウザー	インスタンスのメッセージを参照します。
consumer	インスタンスでメッセージを消費します。

ツール	説明
data	ジャーナルレコードとデータの圧縮に関するレポートを出力します。
decode	エンコードから内部ジャーナル形式をインポートします。
encode	String にエンコードされるジャーナルの内部形式を示しています。
exp	特別な XML 形式および独立した XML 形式を使用して、メッセージデータをエクスポートします。
help	ヘルプ情報を表示します。
imp	**exp** によって提供された出力を使用して、ジャーナルを稼働中のブローカーにインポートします。
kill	--allow-kill で開始するブローカーインスタンスを強制終了します。
mask	パスワードをマスクし、これを出力します。
perf-journal	現在のデータフォルダーで使用する journal-buffer タイムアウトを計算します。
queue	キューのツールグループ (create/delete/update/stat) (例: **./artemis queue create**)
run	ブローカーインスタンスを実行します。
stop	ブローカーインスタンスを停止します。
user	デフォルトのファイルベースのユーザー管 (add/rm/list/reset) (例: **./artemis user list**)

各ツールで利用可能なコマンドの全リストについては、**help** パラメーターの後にツール名を使用してください。たとえば、以下の例で CLI 出力には、ユーザーが **./artemis help data** コマンドを入力すると、**data** ツールで利用可能なコマンドがすべて表示されます。

```
$ ./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)

        ...
```

各コマンドを実行する方法の詳細については、**help** パラメーターを使用します。たとえば、CLI は、ユーザーが **./artemis help data print** の入力後に **data print** コマンドに関する詳細情報をリスト表示します。

```
$ ./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 AMQ 管理コンソールの使用

AMQ 管理コンソールは、AMQ Broker インストールに含まれる Web コンソールであり、Web ブラウザーを使用して AMQ Broker を管理できます。

AMQ 管理コンソールは hawtio をベースとしています。

4.1. 概要

AMQ Broker はフル機能のメッセージ指向ミドルウェアブローカーです。特殊なキュー処理動作、メッセージの永続性、および管理性を提供します。複数のプロトコルおよびクライアント言語をサポートし、多くのアプリケーションアセットを自由に使用できます。

AMQ Broker の主な機能を使用すると、以下が可能になります。

- AMQ Broker およびクライアントの監視
  - トポロジーの表示
  - glance でのネットワークの正常性の表示
- 以下を使用して AMQ Broker を管理します。
  - AMQ 管理コンソール
  - コマンドラインインターフェイス (CLI)
  - 管理 API

AMQ 管理コンソールでサポートされている Web ブラウザーは、Firefox と Chrome です。サポートされるブラウザーバージョンの詳細は、AMQ 7 でサポートされる設定 を参照してください。

4.2. AMQ 管理コンソールへのローカルおよびリモートアクセスの設定

本セクションの手順では、AMQ 管理コンソールへのローカルおよびリモートアクセスを設定する方法を説明します。

コンソールへのリモートアクセスには、以下の 2 つの形式を使用できます。

- ローカルブローカーのコンソールセッション内では、**Connect** タブを使用して別のリモートブローカーに接続します。
- リモートホストから、ローカルブローカーの外部からアクセスできる IP アドレスを使用して、ローカルブローカーのコンソールに接続します。

前提条件

- 少なくとも AMQ Broker 7.1.0 にアップグレードする必要があります。このアップグレードの一環として、**jolokia-access.xml** という名前のアクセス管理設定ファイルをブローカーインスタンスに追加します。アップグレードについての詳細は、Upgrading a Broker instance from 7.0.x to 7.1.0 を参照してください。

手順

1. **<broker_instance_dir>/etc/bootstrap.xml** ファイルを開きます。

2. **web** 要素内で、Web ポートはデフォルトで **localhost** にのみバインドされていることを確認します。

```
<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. リモートホストからローカルブローカーのコンソールへの接続を有効にするには、Web ポートバインディングをネットワーク到達可能なインターフェイスに変更します。以下に例を示します。

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

上記の例では、**0.0.0.0** を指定することで、Web ポートをローカルブローカーの **すべて**のインターフェイスにバインドします。

4. **bootstrap.xml** ファイルを保存します。

5. **<broker_instance_dir>/etc/jolokia-access.xml** ファイルを開きます。

6. **<cors>** (**Cross-Origin Resource Sharing**) 要素内に、コンソールへのアクセスを許可する各 HTTP origin リクエストヘッダーに **allow-origin** エントリーを追加します。以下に例を示します。

```
<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>
```

上記の設定では、以下の接続が許可されるように指定します。

- ローカルホストからコンソールへの接続 (つまり、ローカルブローカーインスタンスのホストマシン)。
  - 最初のアスタリスク (*) ワイルドカード文字では、セキュアな接続にコンソールを設定したかどうかに基づいて、**http** または **https** スキームのいずれかを接続要求で指定できます。
  - 2 つ目のアスタリスクワイルドカード文字を使用すると、ホストマシン上の任意のポートを接続に使用できます。
- ローカルブローカーの外部からアクセスできる IP アドレスを使用して、リモートホストからローカルブローカーのコンソールへの接続。この場合、ローカルブローカーの外部からアクセスできる IP アドレスは **192.168.0.49** です。
- 別のリモートブローカーで開いたコンソールセッション内からローカルブローカーへの接続。この場合、リモートブローカーの IP アドレスは **192.168.0.51** です。

7. **jolokia-access.xml** ファイルを保存します。
8. **<broker_instance_dir>/etc/artemis.profile** ファイルを開きます。
9. コンソールのConnectタブを有効にするには、**Dhawtio.disableProxy** 引数の値を **false** に設定します。

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

重要

コンソールがセキュアなネットワークに公開されている **場合のみ**、コンソールからのリモート接続を有効にすることが推奨されます (つまり、**Dhawtio.disableProxy** 引数の値を **false** に設定)。

10. Java システム引数の **JAVA_ARGS** リストに、新しい引数 **Dhawtio.proxyWhitelist** を追加します。コンマ区切りリストとして、ローカルブローカーから接続するリモートブローカーの IP アドレスを指定します (つまり、ローカルブローカーで実行しているコンソールセッション内の **Connect** タブを使用)。以下に例を示します。

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

上記の設定に基づいて、ローカルブローカーのコンソールセッション内の **Connect** タブを使用して、IP アドレスが **192.168.0.51** の別のリモートブローカーに接続できます。

11. **aretmis.profile** ファイルを保存します。

関連情報

- コンソールへのアクセス方法は、「AMQ 管理コンソールへのアクセス」 を参照してください。
- 詳細情報:
  - クロスオリジンリソース共有については、W3C Recommendations を参照してください。
  - Jolokia のセキュリティーについては、Jolokia Protocols を参照してください。
  - コンソールへの接続のセキュリティー保護については、「AMQ 管理コンソールへのネットワークアクセスのセキュリティー保護」 を参照してください。

4.3. AMQ 管理コンソールへのアクセス

本セクションの手順では、以下の方法を説明します。

- ローカルブローカーからの AMQ 管理コンソールを開く
- ローカルブローカーのコンソールセッション内から他のブローカーに接続する
- ローカルブローカーの外部からアクセスできる IP アドレスを使用して、リモートホストからローカルブローカーのコンソールインスタンスを開きます。

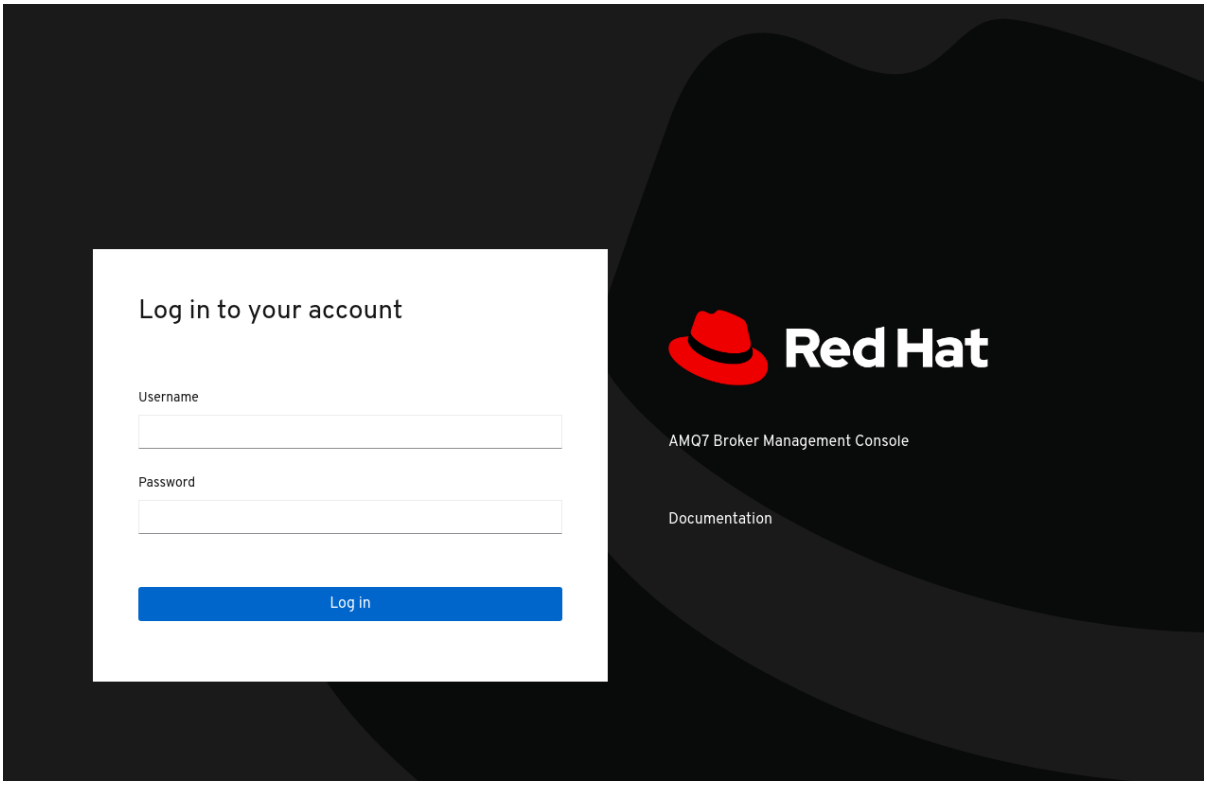
前提条件

- コンソールへのローカルおよびリモートアクセスがすでに設定されている必要があります。詳細は、「AMQ 管理コンソールへのローカルおよびリモートアクセスの設定」 を参照してください。

手順

1. Web ブラウザーで、ローカルブローカーのコンソールアドレスに移動します。
コンソールのアドレスは **http://<host:port>/console/login** です。デフォルトのアドレスを使用している場合は、http://localhost:8161/console/login に移動します。それ以外の場合は、**<broker_instance_dir>/etc/bootstrap.xml** 設定ファイルの **web** 要素の **bind** 属性に定義されているホストおよびポートの値を使用します。

図4.1 コンソールのログインページ



2. ブローカーの作成時に作成したデフォルトのユーザー名とパスワードを使用して、AMQ 管理コンソールにログインします。

3. 別のリモートブローカーに接続するには、ローカルブローカーのコンソールセッションからリモートブローカーに接続します。

   a. 左側のメニューで、**Connect** タブをクリックします。

   b. メインペインで、**リモート** タブの **Add connection** ボタンをクリックします。

   c. **接続の追加** ダイアログボックス で、以下の詳細を指定します。

   Name

   my_other_broker のようなリモート接続の名前。

   スキーム

   リモート接続に使用するプロトコル。非セキュアな接続の場合は **http** を、セキュアな接続の場合は **https** を選択してください。

   ホスト

リモートブローカーの IP アドレス。このリモートブローカーのコンソールアクセスがすでに設定されている必要があります。

ポート

リモート接続に使用するローカルブローカーのポート。**<broker_instance_dir>/etc/bootstrap.xml** 設定ファイルの **web** 要素の **bind** 属性に定義されているポート値を指定してください。デフォルト値は **8161** です。

パス

コンソールアクセスに使用するパス。**console/jolokia** を指定します。

d. 接続をテストするには、**Test Connection** ボタンをクリックします。
接続テストに成功した場合は、**Add** ボタンをクリックします。接続テストに失敗した場合は、必要に応じて接続の詳細を確認し、変更します。接続を再度テストします。

e. **Remote** ページで、追加した接続の **Connect** ボタンをクリックします。
リモートブローカーのコンソールインスタンスに対して、新しい Web ブラウザータブが開きます。

f. **ログイン ダイアログボックスで**、リモートブローカーのユーザー名とパスワードを入力します。**Log In** をクリックします。
リモートブローカーのコンソールインスタンスが開きます。

4. リモートホストからローカルブローカーのコンソールに接続するには、Web ブラウザーでローカルブローカーの Jolokia エンドポイントを指定します。このエンドポイントには、リモートコンソールアクセスの設定時にローカルブローカーに指定した外部からアクセスできる IP アドレスが含まれます。以下に例を示します。

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

4.4. AMQ 管理コンソールの設定

ユーザーアクセスを設定し、ブローカーのリソースへのアクセスを要求します。

4.4.1. Red Hat Single Sign-On を使用した AMQ 管理コンソールの保護

前提条件

- Red Hat Single Sign-On 7.4

手順

1. Red Hat Single Sign-On の設定

a. AMQ 管理コンソールを保護するために使用する Red Hat Single Sign-On のレルムに移動します。Red Hat Single Sign-On の各レルムには、**Broker** という名前のクライアントが含まれます。このクライアントは AMQ とは関係ありません。

b. Red Hat Single Sign-On で、**artemis-console** のような新しいクライアントを作成します。

c. クライアント設定ページに移動し、次を設定します。

- AMQ 管理コンソール URL への **有効なリダイレクト URI** の後に * が続きます。例を以下に示します。

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

- **Web Origins** を **Valid Redirect URIs** と同じ値にします。Red Hat Single Sign-On では、**+** を入力できます。これは、許可される CORS オリジンに **Valid Redirect URIs** の値が含まれていることを示します。

d. クライアントのロールを作成します (例:**guest**)。

e. AMQ Management Console へのアクセスが必要なすべてのユーザーに上記のロールが割り当てられていることを確認します (たとえば、Red Hat Single Sign-On グループを使用)。

2. AMQ Broker インスタンスを設定します。

a. **<broker-instance-dir>/instances/broker0/etc/login.config** ファイルに以下を追加して、AMQ Management Console が Red Hat Single Sign-On を使用するように設定します。

```
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
   ;
};
```

この設定を追加すると、JAAS プリンシパルと、Red Hat Single Sign-On からのベアラートークンの要件が設定されます。次のステップで説明するように、Red Hat Single Sign-On への接続は **keycloak-bearer-token.json** ファイルで定義されます。

b. 以下の内容で **<broker-instance-dir>/etc/keycloak-bearer-token.json** ファイルを作成し、ベアラートークンの交換に使用する Red Hat Single Sign-On への接続を指定します。

```
{
  "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 のレルムの名前

<client-name>

Red Hat Single Sign-On でのクライアントの名前

<RHSSO-URL>

Red Hat Single Sign-On の URL

c. 以下の内容で **<broker-instance-dir>/etc/keycloak-js-token.json** ファイルを作成し、Red Hat Single Sign-On 認証エンドポイントを指定します。

```
{
  "realm": "<realm-name>",
  "clientId": "<client-name>",
```

```
  "url": "<RHSSO-URL>/auth"
}
```

d. **<broker-instance-dir>/etc/broker.xml** ファイルを編集して、セキュリティー設定を行います。
たとえば、**amq** ロールを持つユーザーがメッセージを消費できるようにし、**guest** ロールを持つユーザーがメッセージを送信できるようにするには、以下を追加します。

```
<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. AMQ Broker インスタンスを実行し、AMQ 管理コンソールの設定を検証します。

4.4.2. AMQ 管理コンソールへのユーザーアクセスの設定

ブローカーのログインクレデンシャルを使用して、AMQ 管理コンソールにアクセスできます。以下の表は、AMQ 管理コンソールにアクセスするためにブローカーユーザーを追加するさまざまな方法について説明します。

認証方法	説明
ゲスト認証	匿名アクセスを有効にします。この設定では、クレデンシャルなしまたは誤ったクレデンシャルで接続するユーザーは自動的に認証され、特定のユーザーとロールが割り当てられます。 詳細は、**Configuring AMQ Broker** の Configuring guest access を参照してください。
基本的なユーザーとパスワード認証	各ユーザーに、ユーザー名とパスワードを定義してセキュリティーロールを割り当てる必要があります。ユーザーは、これらのクレデンシャルを使用して AMQ 管理コンソールにのみログインできます。 詳細は、**Configuring AMQ Broker** の Configuring basic user and password authentication を参照してください。
LDAP 認証	ユーザーは、中央の X.500 ディレクトリーサーバーに保存されているユーザーデータに対してクレデンシャルをチェックして認証および認可されます。 詳細は、**Configuring AMQ Broker** の Configuring LDAP to authenticate clients を参照してください。

4.4.3. AMQ 管理コンソールへのネットワークアクセスのセキュリティー保護

コンソールが WAN またはインターネット経由でアクセスされる際に AMQ 管理コンソールのセキュリティーを保護するには、ネットワークアクセスが **http** ではなく **https** を使用するように SSL で指定します。

前提条件

以下は、**<broker_instance_dir>/etc/** ディレクトリーに配置されている必要があります。

- Java キーストア
- Java トラストストア (クライアント認証が必要な場合のみ必要)

手順

1. **<broker_instance_dir>/etc/bootstrap.xml** ファイルを開きます。
2. **<web>** 要素に以下の属性を追加します。

```
<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
: コンソールへのセキュアな接続では、URI スキームを **https** に変更します。

keyStorePath
: キーストアファイルのパス。以下に例を示します。

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

keyStorePassword
: キーストアのパスワード。このパスワードは暗号化できます。

clientAuth
: クライアント認証が必要であるかどうかを指定します。デフォルト値は **false** です。

trustStorePath
: トラストストアファイルのパス。**clientAuth** が **true** に設定されている場合のみ、この属性を定義する必要があります。

trustStorePassword
: トラストストアのパスワード。このパスワードは暗号化できます。

関連情報

- **bootstrap.xml** などのブローカー設定ファイル内のパスワードの暗号化の詳細については、設定ファイル内のパスワードの暗号化 を参照してください。

4.4.4. 証明書ベースの認証を使用するように AMQ 管理コンソールを設定する

パスワードの代わりに証明書を使用してユーザーを認証するように AMQ Management Console を設定できます。

手順

1. 信頼できる認証局からブローカーとクライアントの証明書を取得するか、自己署名証明書を生成します。自己署名証明書を生成する場合は、次の手順を実行します。

   a. ブローカーの自己署名証明書の生成

   ```
   $ 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. ブローカーキーストアから証明書をエクスポートし、クライアントと共有できるようにします。

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

   c. クライアントで、ブローカー証明書をクライアントのトラストストアにインポートします。

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

   d. クライアントで、クライアントの自己署名証明書を生成します。

   ```
   $ 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. クライアント証明書をクライアント鍵ストアからファイルにエクスポートして、ブローカーのトラストストアに追加できるようにします。

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

   f. クライアント証明書をブローカーのトラストストアにインポートします。

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

   注記

   ブローカマシンで、キーストアファイルとトラストストアファイルがブローカからアクセスできる場所にあることを確認します。

2. **<broker_instance_dir>/etc/bootstrap.xml** ファイルで、Web 設定を更新して、ブローカーコンソールの HTTPS プロトコルとクライアント認証を有効にします。以下に例を示します。

   ```
   ...
   <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

https プロトコルを指定して SSL を有効にし、ホスト名とポートを追加します。

keystorePath

ブローカー証明書がインストールされているキーストアへのパス。

keystorePassword

ブローカー証明書がインストールされているキーストアのパスワード。

ClientAuth

クライアントがブローカーコンソールに接続しようとしたときに各クライアントが証明書を提示することを要求するようにブローカーを設定するには、true に設定します。

trustStorePath

クライアントが自己署名証明書を使用している場合は、クライアント証明書がインストールされているトラストストアへのパスを指定します。

trustStorePassword

クライアントが自己署名証明書を使用している場合は、クライアント証明書がインストールされているトラストストアのパスワードを指定します。
注記:クライアントが自己署名証明書を使用している場合にのみ、**trustStorePath** および **trustStorePassword** プロパティーを設定する必要があります。

3. 各クライアント証明書からサブジェクト **識別名** (DN) を取得して、各クライアント証明書とブローカーユーザー間のマッピングを作成できるようにします。

   a. 各クライアント証明書をクライアントのキーストアファイルから一時ファイルにエクスポートします。以下に例を示します。

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

   b. エクスポートされた証明書の内容を出力します。

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

   出力は以下のようになります。

   ```
   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 エントリーは Subject DN です。Subject DN の入力の使用形式はプラットフォーム

Owner エントリーは Subject DN です。Subject DN の入力の使用形式はプラットフォームによって異なります。上記の文字列は、以下のように表現することもできます。

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

4. ブローカーのコンソールに対して証明書ベースの認証を有効にします。

   a. **<broker_instance_dir>/etc/login.config** 設定ファイルを開きます。証明書ログインモジュールを追加し、ユーザーとロールのプロパティーファイルを参照します。以下に例を示します。

```
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

   実装クラス。

   org.apache.activemq.jaas.textfiledn.user

   ログイン設定ファイルが含まれるディレクトリーを基準としたユーザープロパティーファイルの場所を指定します。

   org.apache.activemq.jaas.textfiledn.role

   ユーザーをログインモジュール実装に定義されたロールにマップするプロパティーファイルを指定します。

   注記

   <broker_instance_dir>/etc/login.config ファイルの証明書ログインモジュール設定のデフォルト名を変更する場合は、**<broker_instance_dir>/etc/artemis.profile** ファイルの **-dhawtio.realm** 引数の値を次のように更新する必要があります。新しい名前に合わせます。デフォルト名は **activemq** です。

   b. **<broker_instance_dir>/etc/artemis-users.properties** ファイルを開きます。各クライアント証明書から取得したサブジェクト DNS をブローカーユーザーに追加して、クライアント証明書とブローカーユーザー間のマッピングを作成します。以下に例を示します。

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

   この例では、user1 ブローカーユーザーは、サブジェクト識別名 CN=user1,O=Progress,C=US サブジェクト DN を持つクライアント証明書にマップされます。クライアント証明書とブローカーユーザー間のマッピングを作成すると、ブローカーは証明書を使用してユーザーを認証できます。

   c. **<broker_instance_dir>/etc/artemis-roles.properties** ファイルを開きます。**<broker_instance_dir>/etc/artemis.profile** ファイルの HAWTIO_ROLE 変数に指定されているロールにユーザーを追加して、コンソールにログインする権限をユーザーに付与します。HAWTIO_ROLE 変数のデフォルト値は **amq** です。以下に例を示します。

```
amq=user1, user2
```

5. HTTPS プロトコルの次の推奨セキュリティープロパティーを設定します。

   a. **<broker_instance_dir>/etc/artemis.profile** ファイルを開きます。

   b. **hawtio.http.strictTransportSecurity** プロパティーを設定して、AMQ 管理コンソールへの HTTPS リクエストのみを許可し、すべての HTTP リクエストを HTTPS に変換します。以下に例を示します。

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

   c. **hawtio.http.publicKeyPins** プロパティーを設定して、特定の暗号化公開鍵を AMQ 管理コンソールに関連付けるよう Web ブラウザーに指示し、偽造された証明書を使用した中間者攻撃のリスクを減らします。以下に例を示します。

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

4.4.5. X 転送ヘッダーを処理するための AMQ 管理コンソールの設定

AMQ 管理コンソールへのリクエストがプロキシーサーバー経由でルーティングされる場合は、AMQ 管理コンソールをホスティングする AMQ Broker 組み込み Web サーバーを X-Forwarded ヘッダーを処理するように設定できます。X-Forwarded ヘッダーを処理することにより、AMQ 管理コンソールは、プロキシーがリクエストのパスに関与している場合に変更または失われるヘッダー情報を受け取ることができます。たとえば、プロキシーは HTTPS を使用して AMQ 管理コンソールを公開でき、HTTP を使用する AMQ 管理コンソールは、ブラウザーとプロキシーの間の接続が HTTPS を使用していることを X-Forwarded ヘッダーから識別し、ブラウザーのリクエストを処理するために HTTPS に切り替えることができます。

手順

1. **<broker_instance_dir>/etc/bootstrap.xml** ファイルを開きます。
2. **<web>** 要素に、**org.eclipse.jetty.server.ForwardedRequestCustomizer** の値を持つ **customizer** 属性を追加します。以下に例を示します。

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

3. **bootstrap.xml** ファイルを保存します。
4. 次のコマンドを入力して、ブローカーを起動または再起動します。
   - Linux の場合: **<broker_instance_dir>/bin/artemis run**
   - Windows の場合: **<broker_instance_dir>\bin\artemis-service.exe start**

4.5. AMQ 管理コンソールを使用したブローカーの管理

AMQ 管理コンソールを使用して、稼働中のブローカーに関する情報を表示し、以下のリソースを管理できます。

- 着信ネットワーク接続 (アクセプター)

- アドレス
- Queues

4.5.1. ブローカーの詳細の表示

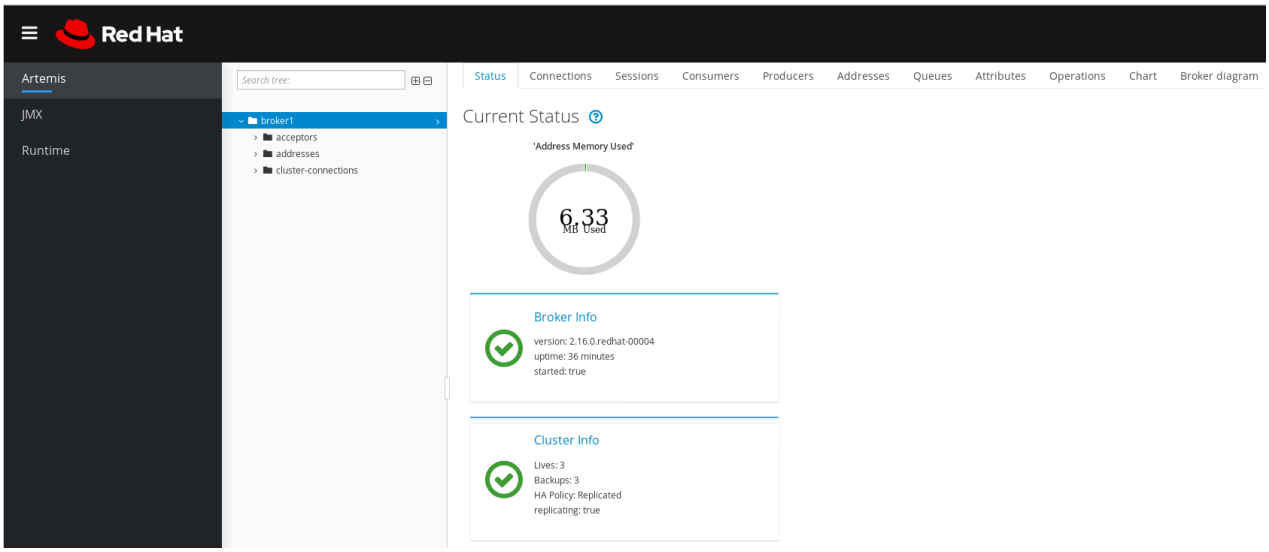
ブローカーの設定方法を確認するには、左側のメニューで **Artemis** をクリックします。フォルダーツリーでは、ローカルブローカーがデフォルトで選択されます。

メインペインで、以下のタブが利用できます。

ステータス

アップタイムやクラスター情報などのブローカーの現在のステータスに関する情報を表示します。また、ブローカーが現在使用しているアドレスメモリー容量も表示します。グラフでは、この値を **global-max-size** コ設定パラメーターに対する割合で示しています。

図4.2 Status タブ



Connections

クライアント、クラスター、ブリッジ接続などのブローカー接続に関する情報を表示します。

セッション

ブローカーで現在開いているすべてのセッションに関する情報を表示します。

Consumers

ブローカーで現在開いているすべてのコンシューマーに関する情報を表示します。

Producers

ブローカーで現在開いているプロデューサーに関する情報を表示します。

アドレス

ブローカーのアドレスに関する情報を表示します。これには、store-and-forward アドレスなどの内部アドレスが含まれます。

Queues

ブローカーのキューに関する情報を表示します。これには、store-and-forward キューなどの内部キューが含まれます。

属性

ブローカーに設定された属性に関する詳細情報を表示します。

操作

コンソールからブローカーで実行できる JMX 操作を表示します。操作をクリックすると、ダイアログボックスが開き、操作のパラメーター値を指定できます。

チャート

ブローカーに設定された属性のリアルタイムデータを表示します。チャートを編集して、チャートに含まれる属性を指定できます。

Broker diagram

クラスタートポロジーの図を表示します。これには、クラスター内のすべてのブローカーと、ローカルブローカーのアドレスおよびキューが含まれます。

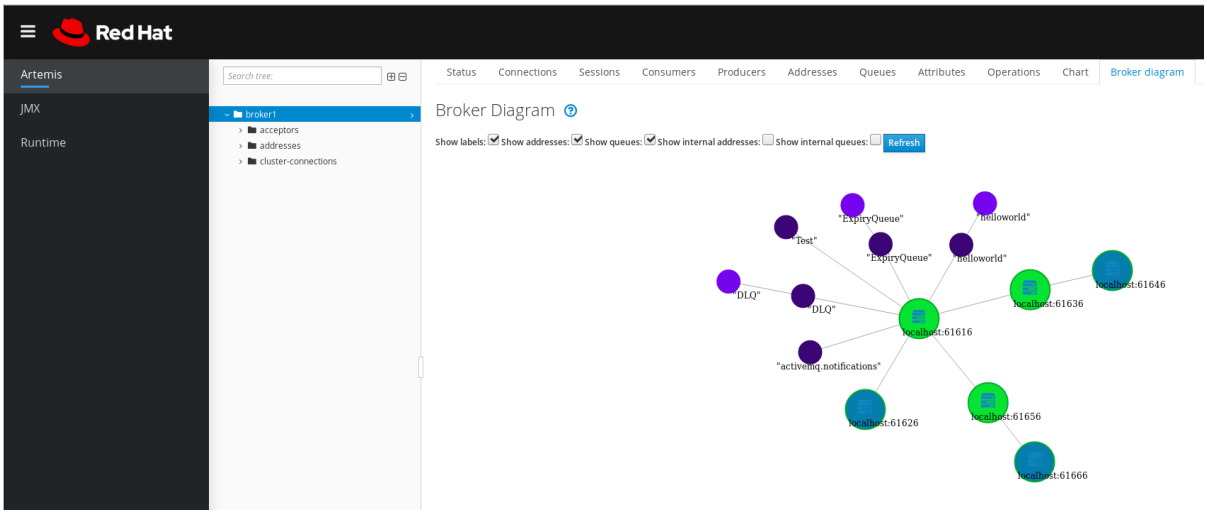
4.5.2. ブローカーダイアグラムの表示

ブローカー (ライブおよびバックアップブローカー)、プロデューサーおよびコンシューマー、アドレス、およびキューなど、トポロジー内のすべての AMQ Broker リソースの図を表示できます。

手順

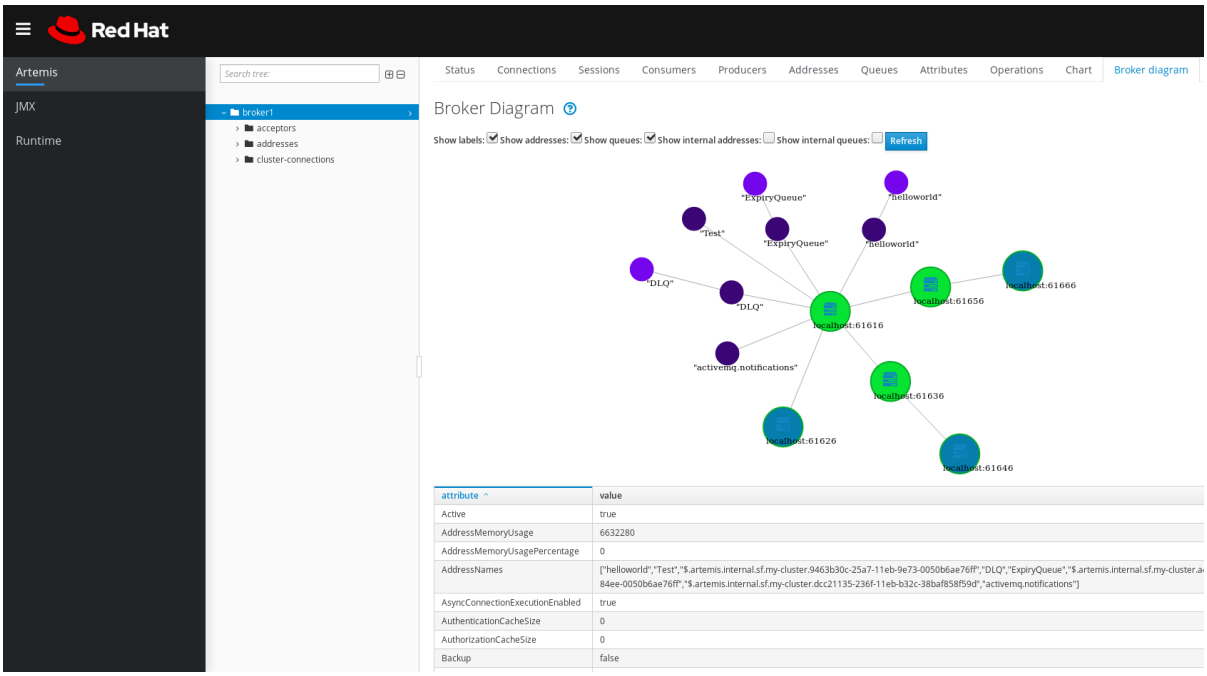
1. 左側のメニューで **Artemis** をクリックします。
2. メインペインで、**Broker diagram** タブをクリックします。
   コンソールは、クラスタートポロジーの図を表示します。これには、図に示すように、クラスター内のすべてのブローカーとローカルブローカーのアドレスおよびキューが含まれます。

   図4.3 ブローカーのダイアグラム タブ



3. ダイアグラムに表示される項目を変更するには、ダイアグラムの上部にあるチェックボックスを使用します。**Refresh** をクリックします。
4. ローカルブローカーの属性、または接続先のアドレスまたはキューの属性を表示するには、ダイアグラムのそのノードをクリックします。たとえば、以下の図は、ローカルブローカーの属性も含まれる図を示しています。

図4.4 属性を含むブローカーダイアグラム タブ



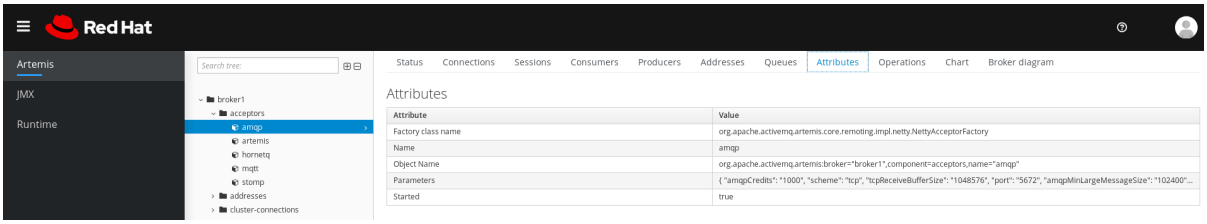
4.5.3. アクセプターの表示

ブローカーに設定されたアクセプターの詳細を表示できます。

手順

1. 左側のメニューで **Artemis** をクリックします。
2. フォルダーツリーで、**acceptors** をクリックします。
3. アクセプターの設定方法の詳細を表示するには、アクセプターをクリックします。
コンソールには、図に示すように **Attributes** タブの対応する属性が表示されます。

図4.5 AMQP アクセプター属性



4. 属性の詳細を表示するには、属性をクリックします。詳細を表示する追加のウィンドウが開きます。

4.5.4. アドレスおよびキューの管理

アドレスはメッセージングエンドポイントを表します。設定内で、通常のアドレスには一意の名前が指定されます。

キューがアドレスに関連付けられます。アドレスごとに複数のキューが存在する場合があります。受信メッセージがアドレスにマッチすると、設定されたルーティングタイプに応じて、メッセージは 1 つ以上のキューに送信されます。キューは、自動作成および削除ができるように設定できます。

4.5.4.1. アドレスの作成

一般的なアドレスには、一意の名前、ゼロ以上のキュー、およびルーティングタイプが指定されます。

ルーティングタイプは、アドレスに関連付けられたキューへメッセージが送信される方法を決定します。アドレスは、2 つの異なるルーティングタイプで設定できます。

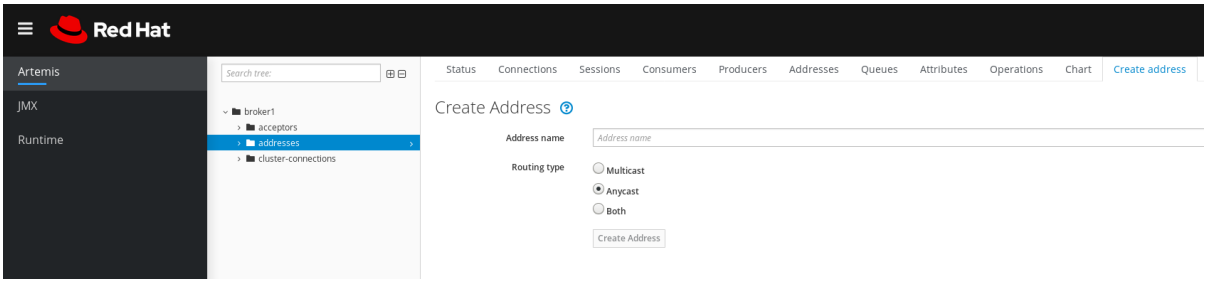
メッセージをルーティング先とルーティングする場合	このルーティングタイプを使用する...
ポイントツーポイント方式で、一致するアドレス内の単一キュー。	anycast
パブリッシュ/サブスクライブ方式で、一致するアドレス内のすべてのキュー。	マルチキャスト

アドレスおよびキューを作成および設定し、使用されていない場合はそれらを削除できます。

手順

1. 左側のメニューで **Artemis** をクリックします。
2. フォルダーツリーで **addresses** をクリックします。
3. メインペインで、**Create address** タブをクリックします。
   図に示すように、アドレスを作成するページが表示されます。

   図4.6 Create Address ページ



4. 以下のフィールドに入力します。

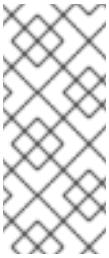
   Address name

   アドレスのルーティング名。

   Routing type

   以下のオプションのいずれかを選択します。

   - **Multicast**: アドレスに送信されたメッセージは、パブリッシュサブスクライブ方式ですべてのサブスクライバーに配信されます。
   - **Anycast**: このアドレスに送信されたメッセージは、ポイントツーポイント方式で 1 人のサブスクライバーにのみ配信されます。
   - **Both**: アドレスごとに複数のルーティングタイプを定義できます。通常、これによりアンチパターンが発生するため、推奨されません。



注記

アドレスが両方のルーティングタイプを使用し、クライアントがどちらにも優先していない場合、ブローカーはデフォルトで **anycast** ルーティングタイプに設定されます。1 つの例外は、クライアントが MQTT プロトコルを使用する場合です。この場合、デフォルトのルーティングタイプは **multicast** です。

5. **Create Address** をクリックします。

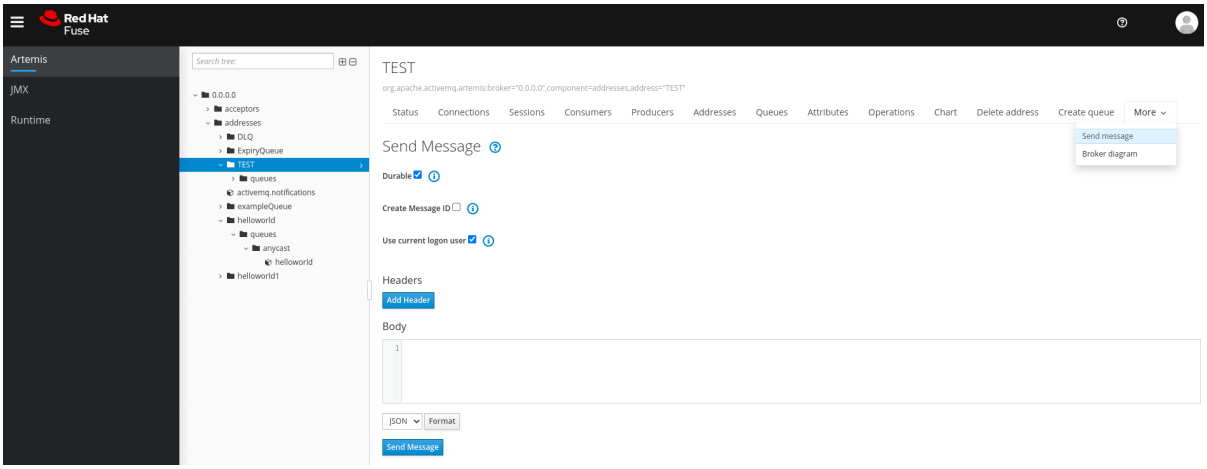
4.5.4.2. アドレスへのメッセージの送信

以下の手順では、コンソールを使用してメッセージをアドレスに送信する方法を説明します。

手順

1. 左側のメニューで **Artemis** をクリックします。
2. フォルダーツリーで、アドレスを選択します。
3. メインペインのナビゲーションバーで **More** → **Send message** をクリックします。
   図に示すように、メッセージを作成するページが表示されます。

図4.7 Send Message ページ



4. デフォルトでは、メッセージは AMQ 管理コンソールへのログインに使用した認証情報を使用して送信されます。別の認証情報を使用する場合は、**Use current logon user** チェックボックスをオフにし、チェックボックスをオフにすると表示される **Username** フィールドと **Password** フィールドに値を指定します。
5. 必要な場合は、**Add Header** ボタンをクリックしてメッセージヘッダー情報を追加します。
6. メッセージのボディーを入力します。
7. **Format** ドロップダウンメニューで、メッセージのボディーのフォーマットのオプションを選択し、**Format** をクリックします。メッセージ本文は、選択した形式向けに人間が判読できるスタイルでフォーマットされます。
8. **Send message** をクリックします。
   メッセージは送信されます。

9. 追加のメッセージを送信するには、入力した情報のいずれかを変更し、**Send message** をクリックします。

4.5.4.3. キューの作成

キューは、プロデューサーとコンシューマー間のチャネルを提供します。

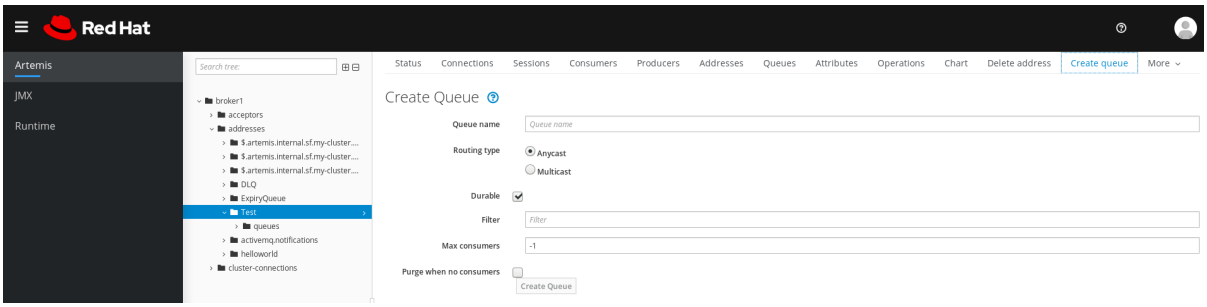
前提条件

- キューをバインドするアドレスが存在する必要があります。コンソールを使用してアドレスを作成する方法は、「アドレスの作成」 を参照してください。

手順

1. 左側のメニューで **Artemis** をクリックします。
2. フォルダーツリーで、キューをバインドするアドレスを選択します。
3. メインペインで **Create queue** タブをクリックします。
図に示すように、キューを作成するページが表示されます。

図4.8 Create Queue ページ



4. 以下のフィールドに入力します。

Queue name
キューの一意の名前。

Routing type
以下のオプションのいずれかを選択します。

- **Multicast**: 親アドレスに送信されたメッセージは、アドレスにバインドされるすべてのキューに送信されます。
- **Anycast**: 親アドレスにバインドされた 1 つのキューのみがメッセージのコピーを受信します。メッセージはアドレスにバインドされたすべてのキューに均等に分散されます。

永続性
このオプションを選択すると、キューとそのメッセージは永続化されます。

Filter
ブローカーへの接続時に使用されるユーザー名。

Max Consumers
一度にキューにアクセスできるコンシューマーの最大数。

Purge when no consumers

選択した場合、コンシューマーが接続されていない場合にキューがパージされます。

5. **Create Queue** をクリックします。

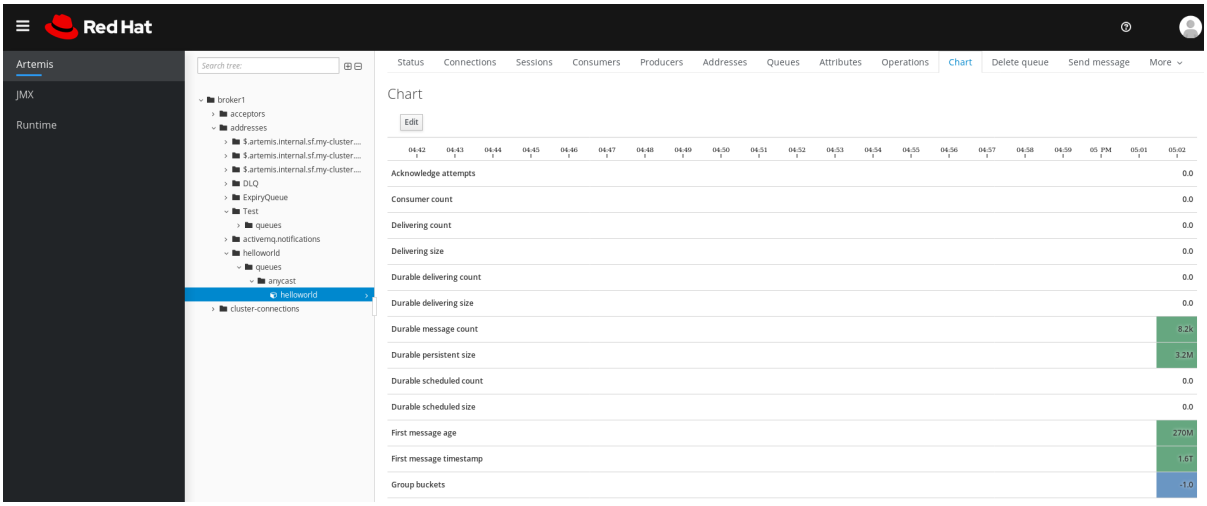
4.5.4.4. キューのステータスの確認

チャートは、ブローカーのキューのステータスのリアルタイムビューを提供します。

手順

1. 左側のメニューで **Artemis** をクリックします。
2. フォルダーツリーで、キューに移動します。
3. メインペインで、**Chart** タブをクリックします。
コンソールは、すべてのキュー属性のリアルタイムデータを表示するチャートを表示します。

図4.9 キューのチャートタブ



注記

あるアドレスの複数のキューのチャートを表示するには、キューを含む **anycast** または **multicast** のフォルダーを選択します。

4. 必要に応じて、チャートに異なる基準を選択します。
   a. メインペインで、**Edit** をクリックします。
   b. **Attributes** リストで、チャートに追加する属性を 1 つ以上選択します。複数の属性を選択するには、**Ctrl** キーを押して保持し、各属性を選択します。
   c. **View Chart** ボタンをクリックします。チャートは選択した属性に基づいて更新されます。

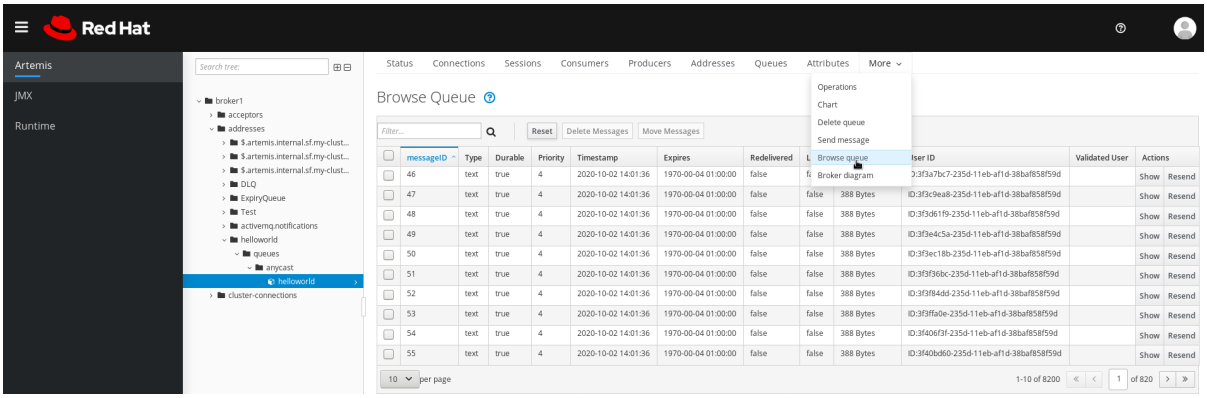
4.5.4.5. キューの参照

キューを参照すると、キュー内のすべてのメッセージが表示されます。また、リストをフィルタリングしてソートして、特定のメッセージを見つけることもできます。

手順

1. 左側のメニューで **Artemis** をクリックします。
2. フォルダーツリーで、キューに移動します。
   キューはバインドされるアドレス内にあります。
3. メインペインのナビゲーションバーで **More → Browse queue** をクリックします。
   キューのメッセージが表示されます。デフォルトでは、最初の 200 メッセージが表示されます。

図4.10 Browse Queue ページ



4. 特定のメッセージまたはメッセージのグループを参照するには、以下のいずれかを行います。

次を行う場合:	こちらを実行します。
メッセージリストをフィルターします。	**Filter...** テキストフィールドで、フィルター基準を入力します。検索 (虫眼鏡) アイコンをクリックします。
メッセージリストを並べ替えます。	メッセージのリストで、列ヘッダーをクリックします。メッセージを降順に並び替えるには、ヘッダーを 2 回クリックします。

5. メッセージの内容を表示するには、**Show** ボタンをクリックします。
   メッセージヘッダー、プロパティー、およびボディーを表示できます。

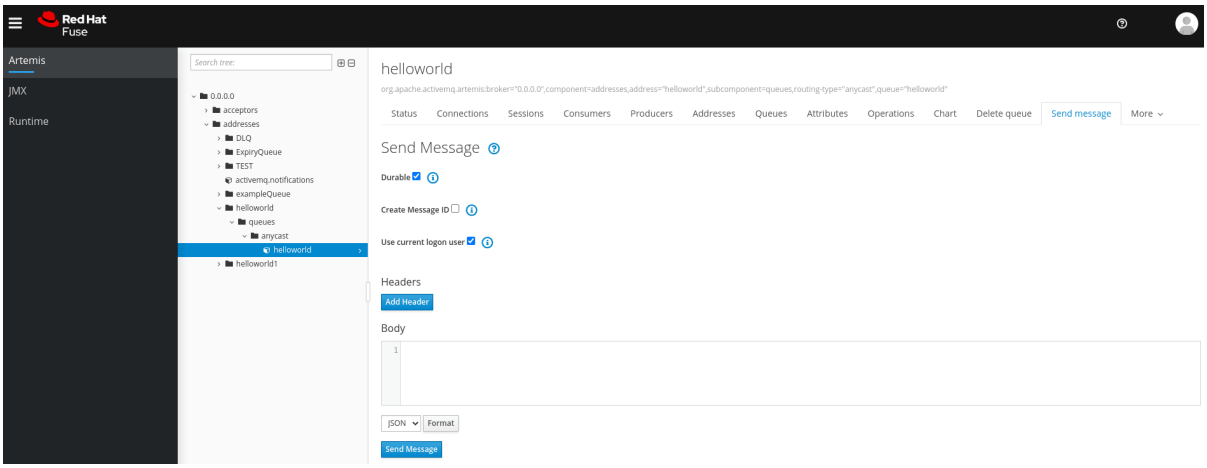
4.5.4.6. キューへのメッセージの送信

キューの作成後、メッセージを送信できます。以下の手順は、メッセージを既存のキューに送信するのに必要な手順の概要を説明します。

手順

1. 左側のメニューで **Artemis** をクリックします。
2. フォルダーツリーで、キューに移動します。
3. メインペインで **Send message** タブをクリックします。
   メッセージを作成するページが表示されます。

図4.11 キューのメッセージページの送信



4. デフォルトでは、メッセージは AMQ 管理コンソールへのログインに使用した認証情報を使用して送信されます。別の認証情報を使用する場合は、**Use current logon user** チェックボックスをオフにし、チェックボックスをオフにすると表示される **Username** フィールドと **Password** フィールドに値を指定します。

5. 必要な場合は、**Add Header** ボタンをクリックしてメッセージヘッダー情報を追加します。

6. メッセージのボディーを入力します。

7. **Format** ドロップダウンメニューで、メッセージのボディーのフォーマットのオプションを選択し、**Format** をクリックします。メッセージ本文は、選択した形式向けに人間が判読できるスタイルでフォーマットされます。

8. **Send message** をクリックします。メッセージは送信されます。

9. 追加のメッセージを送信するには、入力した情報のいずれかを変更し、**Send message** をクリックします。

4.5.4.7. キューへのメッセージの再送信

以前送信されたメッセージを再送信できます。

手順

1. 再送信するメッセージを参照します。

2. 再送信するメッセージの横にあるチェックボックスをクリックします。

3. **Resend** ボタンをクリックします。メッセージが表示されます。

4. 必要に応じて メッセージヘッダーとボディー を更新し、**Send message** をクリックします。

4.5.4.8. 別のキューへのメッセージの移動

キュー内の 1 つ以上のメッセージを別のキューに移動できます。

手順

1. 移動するメッセージを参照します。

2. 移動する各メッセージの横にあるチェックボックスをクリックします。
3. ナビゲーションバーで、**Move Messages** をクリックします。
   確認ダイアログボックスが表示されます。
4. ドロップダウンメニューから、メッセージを移動するキューの名前を選択します。**Move** をクリックします。

4.5.4.9. メッセージまたはキューの削除

キューからキューを削除したり、すべてのメッセージをパージしたりすることができます。

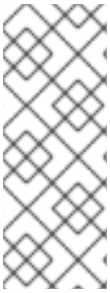
手順

1. 削除またはパージするキューを参照します。
2. 次のいずれかを行います。

次を行う場合:	こちらを実行します。
キューからメッセージを削除します。	1. 削除する各メッセージの横にあるチェックボックスをクリックします。 2. **Delete** ボタンをクリックします。
キューからすべてのメッセージをパージします。	1. メインペインのナビゲーションバーで、**Delete queue** をクリックします。 2. **Purge Queue** ボタンをクリックします。
キューの削除	1. メインペインのナビゲーションバーで、**Delete queue** をクリックします。 2. **Delete Queue** ボタンをクリックします。

第5章 ブローカーランタイムメトリクスのモニタリング

AMQ Broker のインストール時に、Prometheus メトリックプラグインはインストールに含まれます。Prometheus は、大規模でスケーラブルなシステムを監視し、長期間に履歴ランタイムデータを保存するために構築されたソフトウェアです。プラグインを有効にするには、ブローカー設定を変更する必要があります。有効にすると、プラグインはブローカーのランタイムメトリックを収集して Prometheus 形式でエクスポートします。その後、Prometheus を使用してメトリックを確認できます。Grafana などのグラフィカルツールを使用して、データのより高度な可視化を設定することもできます。



注記

Prometheus メトリックプラグインを使用すると、ブローカーメトリックを Prometheus**形式**で収集およびエクスポートできます。ただし、Red Hat では、Prometheus 自体のインストールや設定、または Grafana などの視覚化ツールは、**サポートしていません**。Prometheus または Grafana のインストール、設定、または実行に関するサポートが必要な場合は、製品の Web サイトにアクセスして、コミュニティーのサポートやドキュメントなどのリソースを入手してください。

Prometheus プラグインによって収集されるブローカーメトリックのほかに、ブローカー設定を変更して、ブローカーの Java 仮想マシン (JVM) ホストに関連するメトリックの標準セットをキャプチャーできます。具体的には、ガベージコレクション (GC)、メモリー、スレッドの JVM メトリックをキャプチャーできます。

これ以降のセクションで以下を説明します。

- Prometheus プラグインがエクスポートするメトリック
- Prometheus プラグインを有効にする方法
- JVM メトリックを収集するようにブローカーを設定する方法

5.1. メトリックの概要

AMQ Broker の Prometheus プラグインを使用し、ブローカーのランタイムメトリックを監視および保存して、ブローカーインスタンスの正常性とパフォーマンスを監視できます。AMQ Broker Prometheus プラグインは、ブローカーのランタイムメトリックを Prometheus 形式にエクスポートし、Prometheus 自体を使用してデータのクエリーを視覚化および実行できるようにします。

Grafana などのグラフィカルツールを使用して、Prometheus プラグインが収集するメトリックをさらに詳細にわたり視覚化する設定や、ダッシュボードの設定も行うことができます。

プラグインが Prometheus 形式にエクスポートするメトリックを以下に説明します。

ブローカーメトリック

artemis_address_memory_usage

メモリーメッセージ向けに、このブローカの全アドレスにより使用されるバイト数。

artemis_address_memory_usage_percentage

このブローカ上のすべてのアドレスで使用されるメモリーを、**global-max-size** パラメーターの割合で示したもの。

artemis_connection_count

このブローカーに接続されているクライアントの数。

artemis_total_connection_count

開始してから、このブローカーに接続しているクライアントの数。

アドレスメトリック

artemis_routed_message_count

1つ以上のキューバインディングにルーティングされたメッセージの数。

artemis_unrouted_message_count

キューバインディングにルーティングされ**なかった**メッセージの数。

キューメトリック

artemis_consumer_count

特定のキューからのメッセージを消費しているクライアントの数。

artemis_delivering_durable_message_count

特定のキューが現在コンシューマーに配信している永続メッセージの数。

artemis_delivering_durable_persistent_size

特定のキューが現在コンシューマーに配信している永続メッセージの永続サイズ。

artemis_delivering_message_count

特定のキューが現在コンシューマーに配信しているメッセージの数。

artemis_delivering_persistent_size

特定のキューが現在コンシューマーに配信しているメッセージの永続サイズ。

artemis_durable_message_count

特定のキューに現存する永続メッセージの数。これには、スケジュールされたメッセージ、ページングされたメッセージ、および配信中のメッセージが含まれます。

artemis_durable_persistent_size

現在特定のキューにある永続メッセージの永続サイズ。これには、スケジュールされたメッセージ、ページングされたメッセージ、および配信中のメッセージが含まれます。

artemis_messages_acknowledged

キューが作成されてから、特定のキューから確認応答されたメッセージの数。

artemis_messages_added

キューが作成されてから特定のキューに追加されたメッセージの数。

artemis_message_count

特定のキューに現在あるメッセージの数。これには、スケジュールされたメッセージ、ページングされたメッセージ、および配信中のメッセージが含まれます。

artemis_messages_killed

キューが作成されてからその特定のキューから削除されたメッセージの数。メッセージが設定済みの最大配信試行回数を超えると、ブローカはメッセージを強制終了します。

artemis_messages_expired

キューが作成されてから、その特定のキューから期限切れになったメッセージの数。

artemis_persistent_size

現在特定のキューにある全メッセージ (永続および一時) の永続サイズ。これには、スケジュールされたメッセージ、ページングされたメッセージ、および配信中のメッセージが含まれます。

artemis_scheduled_durable_message_count

指定のキューにスケジュールされた永続メッセージの数。

artemis_scheduled_durable_persistent_size

特定のキューにあるスケジュールされた永続メッセージの永続サイズ。

artemis_scheduled_message_count

特定のキューでスケジュールされたメッセージの数。

artemis_scheduled_persistent_size

特定のキューでスケジュールされたメッセージの永続サイズ。

上記にリストされていない上位レベルのブローカーメトリックについては、下位レベルのメトリックを集計することで算出できます。たとえば、メッセージの合計数を算出するには、ブローカーデプロイメントのすべてのキューから **artemis_message_count** メトリックを集約できます。

AMQ Broker のオンプレミスデプロイメントの場合には、ブローカーをホストする Java 仮想マシン (JVM) のメトリックも Prometheus 形式にエクスポートされます。これは、OpenShift Container Platform での AMQ Broker のデプロイには適用されません。

5.2. AMQ BROKER の PROMETHEUS メトリックプラグインの有効化

AMQ Broker のインストール時に、Prometheus メトリックプラグインはインストールに含まれます。プラグインはすでに使用用に設定されていますが、ブローカー設定でプラグインを有効にする必要があります。有効にすると、プラグインはブローカーのランタイムメトリックを収集して Prometheus 形式でエクスポートします。

次の手順は、AMQ Broker の Prometheus プラグインを有効にする方法を示しています。

手順

1. **<broker_instance_dir>/etc/broker.xml** 設定ファイルを開きます。
2. ブローカー設定で Prometheus プラグインを有効にします。以下のように設定された **<plugin>** サブ要素を持つ **<metrics>** 要素を追加します。

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

3. **broker.xml** 設定ファイルを保存します。メトリックプラグインは、Prometheus 形式でブローカーランタイムメトリックの収集を開始します。

5.3. JVM メトリックを収集するためのブローカーの設定

以下の手順では、ガベージコレクション (GC)、メモリー、およびスレッドの Java 仮想マシン (JVM) メトリックを収集するようにブローカーを設定する方法を説明します。

前提条件

- ブローカー設定で Prometheus メトリックプラグインを有効にしている。詳細は、「AMQ Broker の Prometheus メトリックプラグインの有効化」 を参照してください。

手順

1. **<broker_instance_dir>/etc/broker.xml** 設定ファイルを開きます。

2. Prometheus メトリックプラグインを有効にする際に設定に追加した **<metrics>** 要素で、ブローカーがガベージコレクション (GC)、メモリー、およびスレッドに対して JVM メトリックを収集するかどうかを指定します。以下に例を示します。

```
<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>
```

注記

設定に **jvm-memory** パラメーターを明示的に追加したり、値を指定したりしない場合、ブローカーはデフォルト値の **true** を使用します。これは、ブローカーはデフォルトで JVM メモリーメトリックをエクスポートすることを意味します。**jvm-gc** および **jvm-threads** パラメーターのデフォルト値は **false** です。

3. **broker.xml** 設定ファイルを保存します。ブローカーは、有効にした JVM メトリックの収集を開始します。これらのメトリックは Prometheus 形式にもエクスポートされます。

5.4. 特定のアドレスのメトリックコレクションの無効化

AMQ Broker のメトリックプラグインを設定する場合 (Prometheus メトリックプラグインなど)、メトリックコレクションはデフォルトで有効にされます。ただし、特定のアドレスまたはアドレス **セット** の **address-setting** 設定要素内で、メトリックコレクションを明示的に無効にできます。

以下の手順では、特定のアドレスまたはアドレス **セット** のメトリックコレクションを無効にする方法を説明します。

手順

1. **<broker_instance_dir>/etc/broker.xml** 設定ファイルを開きます。

2. 一致するアドレスまたはアドレス **セット** の **address-setting** 要素で、**enable-metrics** パラメーターを追加し、パラメーターの値を **false** に設定します。たとえば、以下の設定では、**orders** というアドレスのメトリックコレクションを無効にします。

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

```
    ...
  </core>
</configuration>
```

5.5. PROMETHEUS を使用したブローカーランタイムデータへのアクセス

前提条件

- Prometheus プラグインによって収集されるブローカーランタイムデータをクエリーおよび可視化するには、Prometheus をインストールする必要があります。詳細は、Prometheus ドキュメントの Installing Prometheus を参照してください。

手順

1. Prometheus をインストールしたディレクトリーで、**prometheus.yml** 設定ファイルを開きます。
2. 設定ファイルの **static_configs** セクションで、**targets** 要素を **localhost:8161** に変更します。この場所は、ブローカーが Web サーバーを実行する場所です。デフォルトでは、**/metrics** はこのホスト名に追加され、ブローカー Web サーバーに保存されているメトリックへの完全パスを形成します。
3. Prometheus プラグインによって収集されるブローカーランタイムメトリックを表示するには、Web ブラウザーで **localhost:8161/metrics** を開きます。
   生成される Web ページで、ブローカーに設定したキューおよびアドレスに基づいて、プラグインによって収集されるメトリックの現在の値が表示されます。JVM に複数の実行中のブローカーインスタンスがある場合、各ブローカーのメトリックが表示されます。
4. Prometheus インストールディレクトリーから、Prometheus を実行します。

   ```
   $ ./prometheus
   ```

   Prometheus が起動すると、シェル出力に以下の行が含まれます。

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

   上記の行は、Prometheus がポート 9090 で HTTP トラフィックをリッスンしていることを示しています。
5. Prometheus Web コンソールにアクセスするには、Web ブラウザーで **127.0.0.1:9090** を開きます。
6. Prometheus Web コンソールで、**Expression** フィールドを使用して、ブローカーデータにクエリーを作成できます。作成するクエリーは、Prometheus クエリー言語 PromQL に基づいています。クエリーの挿入に使用できるブローカーメトリックは **Insert metric** ドロップダウンリストにあります。
   簡単な例として、DLQ キューのメッセージ数を経時的にクエリーするとします。この場合、メトリックのドロップダウンリストから **artemis_message_count** を選択します。DLQ キュー名とアドレスを指定してクエリーを完了します。このクエリーの例を以下に示します。

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

   さらに高度な視覚化を行うには、正規表現を使用して、複数のメトリックをオーバーレイする複雑なクエリーを作成できます。または、多数のメトリックで数学的な操作 (集計など) を実行

することもできます。Prometheus クエリーの作成に関する詳細は、Prometheus ドキュメントの Prometheus のクエリー を参照してください。

第6章 管理 API の使用

AMQ Broker には、ブローカーの設定変更、新規リソースの作成 (アドレスやキューなど) の作成、これらのリソース (現在のキューに現在保持されるメッセージ数など) を検査し、それらと対話する (たとえば、キューからメッセージを削除するために使用できる) 豊富な管理 API があります。

さらに、クライアントは管理 API を使用してブローカーを管理し、管理通知にサブスクライブできます。

6.1. 管理 API を使用した AMQ BROKER の管理方法

管理 API を使用してブローカーを管理する方法は 2 つあります。

- JMX (JMX) の使用は、Java アプリケーションを管理する標準的な方法です。
- JMS メッセージと AMQ JMS クライアントを使用して JMS APIListenerExternalmanagement 操作を使用するとブローカーに送信されます。

ブローカーを管理する方法は 2 つありますが、各 API は同じ機能をサポートします。JMX を使用してリソースを管理する可能性がある場合は、JMS メッセージおよび AMQ JMS クライアントを使用して同じ結果を実現することもできます。

この選択は、特定の要件、アプリケーション設定、および環境によって異なります。管理操作の呼び出し方法に関係なく、管理 API は同じになります。

マネージドリソースごとに、このタイプのリソースに対して呼び出すことができる Java インターフェイスが存在します。ブローカーは、**org.apache.activemq.artemis.api.core.management** パッケージで管理されたリソースを公開します。管理操作を呼び出す方法は、JMX メッセージまたは JMS メッセージと AMQ JMS クライアントが使用されるかどうかによって異なります。

注記

管理操作によっては、操作の影響を受けるメッセージを選択するために **filter** パラメーターが必要なものもあります。**null** または空の文字列を渡すと、**すべてのメッセージ** で管理操作が実行されることを意味します。

6.2. JMX を使用した AMQ BROKER の管理

JMX (Java Management Extensions) を使用してブローカーを管理できます。管理 API は、MBean インターフェイスを使用してブローカーによって公開されます。ブローカーは、リソースをドメイン **org.apache.activemq** に登録します。

たとえば、**exampleQueue** という名前のキューを管理するための **ObjectName** は次のとおりです。

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

MBean は以下のようになります。

```
org.apache.activemq.artemis.api.management.QueueControl
```

MBean の **ObjectName** は、ヘルパークラス **org.apache.activemq.artemis.api.core.management.ObjectNameBuilder**.を使用して構築されます。jconsole を使用して、管理する MBean の **ObjectName** を見つけることもできます。

JMX を使用したブローカーの管理は、JMX を使用した Java アプリケーションの管理と同じです。これは、リフレクションまたは MBean のプロキシーを作成して実行できます。

6.2.1. JMX 管理の設定

デフォルトでは、JMX はブローカーの管理に有効になっています。JMX 管理を有効または無効にするには、**broker.xml** 設定ファイルで **jmx-management-enabled** プロパティーを設定します。

手順

1. **<broker_instance_dir>/etc/broker.xml** 設定ファイルを開きます。
2. **<jmx-management-enabled>** を設定します。

```
<jmx-management-enabled>true</jmx-management-enabled>
```

JMX が有効になっている場合、ブローカーは **jconsole** を使用してローカルで管理できます。

注記

セキュリティー上の理由から、JMX へのリモート接続はデフォルトで有効になっていません。

3. 同じ **MBeanServer** から複数のブローカーを管理する場合は、各ブローカーに JMX ドメインを設定します。
デフォルトでは、ブローカーは JMX ドメイン **org.apache.activemq.artemis** を使用します。

```
<jmx-domain>my.org.apache.activemq</jmx-domain>
```

注記

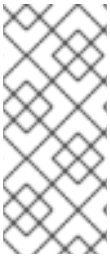
Windows システムで AMQ Broker を使用している場合は、**artemis** または **artemis.cmd** でシステムプロパティーを設定する必要があります。シェルスクリプトは **<install_dir>/bin** の下にあります。

関連情報

- リモート管理用のブローカーの設定に関する詳細は、Oracle の Java Management Guide を参照してください。

6.2.2. JMX 管理アクセスの設定

デフォルトでは、セキュリティー上の理由から、ブローカへのリモート JMX アクセスは無効になっています。ただし、AMQ Broker には、JMX MBean へのリモートアクセスを許可する JMX エージェントがあります。ブローカー **management.xml** 設定ファイルに connector 要素を設定して JMX アクセスを有効にします。



注記

com.sun.management.jmxremoteJVM システムプロパティーを使用して JMX アクセスを有効にすることもできますが、その方法はサポートされておらず、安全ではありません。その JVM システムプロパティーを変更すると、ブローカーで RBAC をバイパスできます。セキュリティーリスクを最小限に抑えるために、localhost へのアクセスを制限することを検討してください。

重要

リモート管理のためにブローカーの JMX エージェントを公開すると、セキュリティーに影響があります。

この手順で説明されているように設定を保護するには、次のようにします。

- すべての接続に SSL を使用します。
- コネクターホスト、つまり、エージェントを公開するホストとポートを明示的に定義します。
- RMI (Remote Method Invocation) レジストリーがバインドするポートを明示的に定義します。

前提条件

- 作業中のブローカーインスタンス
- Java の **jconsole** ユーティリティー

手順

1. **<broker-instance-dir>/etc/management.xml** 設定ファイルを開きます。
2. JMX エージェントのコネクターを定義します。connector-port 設定は、jconsole などのクライアントが JMX コネクターサーバーに対してクエリーを実行する RMI レジストリーを確立します。たとえば、ポート 1099 でのリモートアクセスを許可するには、次のようにします。

```
<connector connector-port="1099"/>
```

3. **jconsole** を使用して、JMX エージェントへの接続を確認します。

```
service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi
```

4. 以下で説明するように、コネクターに追加のプロパティーを定義します。

connector-host

エージェントを公開するブローカーサーバーホスト。リモートアクセスを防ぐには、**connector-host** を **127.0.0.1** (localhost) に設定します。

rmi-registry-port

JMX RMI コネクターサーバーがバインドするポート。設定されていない場合、ポートは常にランダムです。このプロパティーを設定して、ファイアウォールを通過するリモート JMX 接続の問題を回避します。

jmx-realm

認証に使用する JMX レルム。デフォルト値は、JAAS 設定に一致する **activemq** です。

object-name

リモートコネクターを公開するオブジェクト名。デフォルト値は **connector:name=rmi** です。

安全な

コネクターが SSL を使用して保護されているかどうかを指定します。デフォルト値は **false** です。通信をセキュアにするには、値を **true** に設定します。

key-store-path

キーストアの場所。**secured="true"** を設定している場合は必須です。

key-store-password

キーストアパスワード**secured="true"** を設定している場合は必須です。このパスワードは暗号化できます。

key-store-provider

キーストアプロバイダー。**secured="true"** を設定している場合は必須です。デフォルト値は **JKS** です。

trust-store-path

トラストストアの場所。**secured="true"** を設定している場合は必須です。

trust-store-password

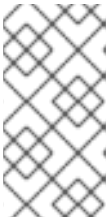
トラストストアのパスワード**secured="true"** を設定している場合は必須です。このパスワードは暗号化できます。

trust-store-provider

トラストストアプロバイダー。**secured="true"** を設定している場合は必須です。デフォルト値は **JKS** です。

password-codec

使用するパスワードコーデックの完全修飾クラス名。この仕組みの詳細については、以下にリンクされているパスワードマスキングのドキュメントを参照してください。



注記

RMI レジストリーは、バインドする IP アドレスを選択します。システムに複数の IP アドレス/NIC が存在する場合は、**artemis.profile** ファイルに次の内容を追加して、使用する IP アドレスを選択できます: **-Djava.rmi.server.hostname=localhost**

5. Java Platform ドキュメント に記載されているように、**jdk.serialFilter** を使用してエンドポイントシリアライゼーションに適切な値を設定します。

関連情報

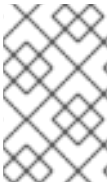
- 設定ファイル内の暗号化されたパスワードの詳細については、設定ファイル内のパスワードの暗号化 を参照してください。

6.2.3. MBeanServer の設定

ブローカーがスタンドアロンモードで動作しているときは、Java 仮想マシンの **Platform MBeanServer** を使用して MBeans を登録します。デフォルトでは、 Jolokia もデプロイされ、REST を使用した MBean サーバーへのアクセスを許可します。

6.2.4. Jolokia で JMX を公開する方法

デフォルトでは、AMQ Broker には Web アプリケーションとしてデプロイされた Jolokia HTTP エージェントが同梱されます。Jolokia は、MBean を公開する HTTP ブリッジ上のリモート JMX です。



注記

Jolokia を使用するには、ユーザーは **<broker_instance_dir>/etc/artemis.profile** 設定ファイルの **hawtio.role** システムプロパティーによって定義されたロールに属している必要があります。デフォルトでは、このロールは **amq** です。

例6.1 Jolokia を使用したブローカーのバージョンのクエリー

この例では、Jolokia REST URL を使用してブローカーのバージョンを検索します。**Origin** フラグは、ブローカーサーバーのドメイン名または DNS ホスト名を指定する必要があります。さらに、**Origin** に指定する値は、Jolokia Cross-Origin Resource Sharing (CORS) 仕様の **<allow-origin>** のエントリーに対応している必要があります。

```
$ 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}
```

関連情報

- JMX-HTTP ブリッジの使用に関する詳細は、Jolokia のドキュメント を参照してください。
- ユーザーをロールに割り当てる方法は、ユーザーの追加 を参照してください。
- CORS(Jolokia Cross-Origin Resource Sharing) の指定に関する詳細は、セキュリティー のセクション 4.1.5 を参照してください。

6.2.5. JMX 管理通知のサブスクライブ

お使いの環境で JMX が有効になっている場合は、管理通知にサブスクライブできます。

手順

- **ObjectName org.apache.activemq.artemis:broker="<broker-name>"** にサブスクライブします。

関連情報

- 管理通知の詳細は、「管理通知」 を参照してください。

6.3. JMS API を使用した AMQ BROKER の管理

Java Message Service (JMS) API を使用すると、メッセージの作成、送信、受信、読み取りが可能です。JMS および AMQ JMS クライアントを使用してブローカーを管理できます。

6.3.1. JMS メッセージおよび AMQ JMS クライアントを使用したブローカー管理の設定

JMS を使用してブローカーを管理するには、まず **manage** パーミッションでブローカーの管理アドレスを設定する必要があります。

手順

1. **<broker_instance_dir>/etc/broker.xml** 設定ファイルを開きます。
2. **<management-address>** 要素を追加し、管理アドレスを指定します。
   デフォルトでは、管理アドレスは **activemq.management** です。デフォルトを使用しない場合は、別のアドレスを指定する必要があります。

```
<management-address>my.management.address</management-address>
```

3. 管理アドレスに **manage** ユーザーパーミッションタイプを指定します。
   このパーミッションタイプにより、管理アドレスの管理メッセージが受信および処理できるようになります。

```
<security-setting-match="activemq.management">
   <permission-type="manage" roles="admin"/>
</security-setting>
```

6.3.2. JMS API および AMQ JMS クライアントを使用したブローカーの管理

JMS メッセージを使用して管理操作を呼び出すには、AMQ JMS クライアントは特別な管理キューをインスタンス化する必要があります。

手順

1. **QueueRequestor** を作成して、管理アドレスにメッセージを送信し、返信を受信します。
2. **Message** を作成します。
3. ヘルパークラス **org.apache.activemq.artemis.api.jms.management.JMSManagementHelper** を使用して、メッセージに管理プロパティーを記入します。
4. **QueueRequestor** を使用してメッセージを送信します。
5. ヘルパークラス **org.apache.activemq.artemis.api.jms.management.JMSManagementHelper** を使用して、管理応答から操作結果を取得します。

例6.2 キュー内のメッセージ数の表示

以下の例は、JMS API を使用して JMS キュー **exampleQueue** でメッセージの数を表示する方法を示しています。

```
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

JMX または JMS メッセージを使用して AMQ Broker を管理する場合でも、同じ API 管理操作を使用できます。管理 API を使用すると、ブローカー、アドレス、およびキューを管理できます。

6.4.1. ブローカー管理操作

管理 API を使用してブローカーを管理できます。

キューのリスト表示、作成、デプロイ、破棄

デプロイされたキューのリストは、**getQueueNames()** メソッドで取得できます。
キューは、**ActiveMQServerControl** の管理操作 **createQueue()**、**deployQueue()**、または **destroyQueue()** を使用して作成または破棄できます (**ObjectName org.apache.activemq.artemis:broker="BROKER_NAME"** またはリソース名 **server** を使用)。

deployQueue が何もしない間、キューがすでに存在する場合、**createQueue** は失敗します。

キューの一時停止および再開

QueueControl は、基礎となるキューを一時停止したり、再開したりすることができます。キューが一時停止すると、メッセージは受信されますが、配信されません。再開すると、キューに格納されたメッセージの配信を開始します (存在する場合)。

リモート接続のリスト表示および閉じる

listRemoteAddresses() を使用して、クライアントのリモートアドレスを取得します。また、**closeConnectionsForAddress()** メソッドを使用して、リモートアドレスに関連する接続を閉じることも可能です。
または、**listConnectionIDs()** を使用して接続 ID をリスト表示し、**listSessions()** を使用して指定の接続 ID の全セッションをリスト表示します。

トランザクションの管理

ブローカーがクラッシュした場合、ブローカーの再起動時に一部のトランザクションを手動で介入する必要がある場合があります。以下の方法を使用して、発生した問題を解決します。
listPreparedTransactions() メソッドリストを使用して、準備済み状態のトランザクションをリスト表示します (トランザクションは不透明な Base64 文字列として表されます)。

commitPreparedTransaction() または **rollbackPreparedTransaction()** を使用して、指定された準備済みトランザクションをコミットまたはロールバックし、ヒューリスティックトランザクションを解決します。

listHeuristicCommittedTransactions() メソッドおよび **listHeuristicRolledBackTransactions** メソッドを使用して、ヒューリスティックに完了したトランザクションをリスト表示します。

メッセージカウンターの有効化およびリセット

enableMessageCounters() または **disableMessageCounters()** メソッドを使用して、メッセージカウンターを有効または無効にします。

resetAllMessageCounters() メソッドと **resetAllMessageCounterHistories()** メソッドを使用して、メッセージカウンターをリセットします。

ブローカー設定および属性の取得

ActiveMQServerControl は、そのすべての属性 (たとえば、ブローカーのバージョンを取得するための **getVersion()** メソッドなど) を通じて、ブローカーの設定を公開します。

コアブリッジおよび迂回のリスト表示、作成、破棄

デプロイされた Core Bridge をリスト表示し、**getBridgeNames()** と **getDivertNames()** メソッドをそれぞれ使用して迂回します。
ActiveMQServerControl で、**createBridge()** と **destroyBridge()** または **createDivert()** と **destroyDivert()** を使用するブリッジと迂回を使用して、作成または破棄します (**ObjectName org.apache.activemq.artemis:broker="BROKER_NAME"** またはリソース名 **server** を使用)。

ブローカーを停止し、現在割り当てられているクライアントでフェイルオーバーを強制する

ActiveMQServerControl の **forceFailover()** を使用します (**ObjectName org.apache.activemq.artemis:broker="BROKER_NAME"** またはリソース名 **server** を使用)。

注記

このメソッドは実際にブローカーを停止するため、エラーが発生する可能性が高くなります。正確なエラーは、メソッドの呼び出しに使用した管理サービスによって異なります。

6.4.2. アドレス管理操作

管理 API を使用してアドレスを管理できます。

アドレスの管理には、**ObjectName org.apache.activemq.artemis:broker="<broker-name>", component=addresses,address="<address-name>"** またはリソース名 **address.<address-name>** の **AddressControl** クラスを使用します。

addRole() メソッドや **removeRole()** メソッドを使用して、アドレスのロールやパーミッションを変更します。**getRoles()** メソッドで、キューに関連付けられたすべてのロールをリスト表示できます。

6.4.3. キュー管理操作

管理 API を使用してキューを管理できます。

コア管理 API はキューを処理します。**QueueControl** クラスは、キューの管理操作を定義します (**ObjectName,org.apache.activemq.artemis:broker="<broker-name>",component=addresses,address="<bound-address>",subcomponent=queues,routing-type="<routing-type>",queue="<queue-name>"** またはリソース名 **queue.<queue-name>** を使用)。

キューの管理操作のほとんどは、単一のメッセージ ID (単一のメッセージを削除するなど) またはフィルター (指定のプロパティーですべてのメッセージを期限切れにするなど) を取ります。

期限切れで、デッドレターアドレスに送信し、メッセージの移動

expireMessages() メソッドを使用して、キューのメッセージを失効させます。期限切れアドレスが定義されている場合、メッセージはこのアドレスに送信されます。それ以外の場合、メッセージは破棄されます。**broker.xml** 設定ファイルの **address-settings** 要素で、アドレスまたはアドレス の

セット (つまり、これらのアドレスにバインドされるキュー) の期限切れアドレスを定義できます。例については、デフォルトのブローカー設定について の「デフォルトのメッセージアドレス設定」セクションを参照してください。
sendMessagesToDeadLetterAddress() メソッドを使用して、デッドレターアドレスにメッセージを送信します。このメソッドは、デッドレターアドレスに送信されたメッセージの数を返します。デッドレターアドレスが定義されている場合、メッセージはこのアドレスに送信されます。一致しない場合、メッセージはキューから削除され、破棄されます。**broker.xml** 設定ファイルの **address-settings** 要素で、アドレスまたはアドレス の **セット** (つまり、これらのアドレスにバインドされるキュー) のデッドレターアドレスを定義できます。例については、デフォルトのブローカー設定について の「デフォルトのメッセージアドレス設定」セクションを参照してください。

moveMessages() メソッドを使用して、あるキューから別のキューにメッセージを移動します。

メッセージのリスト表示と削除

listMessages() メソッドを使用して、あるキューからメッセージをリスト表示します。**Map** の配列 (各メッセージに対して 1 つの **Map**) を返します。
removeMessages() メソッドを使用してキューからメッセージを削除します。これは、単一のメッセージ ID バリアントの **boolean**、またはフィルターバリアントの削除されたメッセージの数を返します。このメソッドは、**filter** 引数を取り、フィルターされたメッセージのみを削除します。フィルターを空の文字列に設定すると、すべてのメッセージが削除されます。

メッセージのカウント

キューに入っているメッセージの数は、**getMessageCount()** メソッドで返されます。または、**countMessages()** は指定のフィルターに一致するキュー内のメッセージの数を返します。

メッセージの優先度の変更

メッセージの優先度は、単一のメッセージ ID バリアントの **boolean** またはフィルターバリアントの更新されたメッセージの数を返す **changeMessagesPriority()** メソッドを使用して変更できます。

メッセージカウンター

メッセージカウンターは、**listMessageCounter()** および **listMessageCounterHistory()** メソッドを使用して、キューに対してリスト表示することができます (「メッセージカウンターの使用」を参照)。メッセージカウンターは、**resetMessageCounter()** メソッドを使用して、1 つのキューに対してリセットすることもできます。

キュー属性の取得

QueueControl は、属性を使用してキュー設定を公開します (たとえば、キューのフィルターが作成されている場合はこれを取得するために **getFilter()** を使用、キューが永続的かどうかを知るためには **isDurable()** を使用など)。

キューの一時停止および再開

QueueControl は、基礎となるキューを一時停止したり、再開したりすることができます。キューが一時停止すると、メッセージは受信されますが、配信されません。再開すると、キューに格納されたメッセージの配信を開始します (存在する場合)。

6.4.4. リモートリソース管理操作

管理 API を使用してブローカーのリモートリソース (アクセプター、迂回、ブリッジなど) を起動および停止し、ブローカーを完全に停止せずに特定の期間にオフラインにすることができます。

アクセプター

start() またはを使用してアクセプターを開始または停止します。**AcceptorControl** クラスの **stop()** メソッド (**ObjectName org.apache.activemq.artemis:broker="<broker-name>",component=acceptors,name="<acceptor-name>"** またはリソース名

acceptor.<address-name> を使用)。アクセプターパラメーターは、**AcceptorControl** 属性を使用して取得できます。アクセプターの詳細については、ネットワーク接続: アクセプターとコネクターを参照してください。

Diverts

DivertControlクラスの**start()**または**stop()**メソッドを使用して、ダイバートを開始または停止します (**ObjectName org.apache.activemq.artemis:broker="<broker-name>",component=diverts,name="<divert-name>"**またはリソース名**divert.<divert-name>**を使用)。迂回パラメーターは、**DivertControl** 属性を使用して取得できます。

ブリッジ

start() (リスピン) を使用して、ブリッジを開始または停止します。**BridgeControl** クラスの **stop()** メソッド (**ObjectName org.apache.activemq.artemis:broker="<broker-name>",component=bridge,name="<bridge-name>"** or the resource name **bridge.<bridge-name>** を伴う)。ブリッジのパラメーターは、**BridgeControl** 属性を使用して取得することができます。

ブロードキャストグループ

BroadcastGroupControl クラスの **start()** または **stop()** メソッドを使用して、ブロードキャストグループを開始または停止します (**ObjectName org.apache.activemq.artemis:broker="<broker-name>",component=broadcast-group,name="<broadcast-group-name>"** またはリソース名 **broadcastgroup.<broadcast-group-name>** を使用)。ブロードキャストグループのパラメーターは、**BroadcastGroupControl** 属性を使用して取得することができます。詳細は 、ブローカー検出メソッド を参照してください。

検出グループ

DiscoveryGroupControl クラスの **start()** または **stop()** メソッドを使用して、ディスカバリーグループを開始または停止します (**ObjectName org.apache.activemq.artemis:broker="<broker-name>",component=discovery-group,name="<discovery-group-name>"** またはリソース名 **discovery.<discovery-group-name>** を使用)。ディスカバリーグループのパラメーターは、**DiscoveryGroupControl** 属性を使用して取得することができます。詳細は 、ブローカー検出メソッド を参照してください。

クラスター接続

ClusterConnectionControl クラスの **start()** または **stop()** メソッドを使用して、クラスター接続を開始または停止します (h the **ObjectName org.apache.activemq.artemis:broker="<broker-name>",component=cluster-connection,name="<cluster-connection-name>"** またはリソース名 **clusterconnection.<cluster-connection-name>** を使用)。クラスター接続パラメーターは、**ClusterConnectionControl** 属性を使用して取得できます。詳細は、ブローカークラスターの作成を参照してください。

6.5. 管理通知

以下は、あらゆる種類の通知とメッセージにあるヘッダーのリストです。すべての通知には、**_AMQ_NotifType**(カッコ内に示されている値) と **_AMQ_NotifTimestamp** ヘッダがあります。タイムスタンプは、**java.lang.System.currentTimeMillis()** への呼び出しの結果で、フォーマットされていません。

通知タイプ	ヘッダー

通知タイプ	ヘッダー
BINDING_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**

通知タイプ	ヘッダー
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**

通知タイプ	ヘッダー
ACCEPTOR_STOPPED (17)	**factory** **id**
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. メッセージカウンターの使用

メッセージカウンターを使用して、キューについての情報を経時的に取得します。これは、特に確認が困難な傾向を特定するのに役立ちます。

たとえば、メッセージカウンターを使用して、特定のキューが時間とともにどのように使用されるかを決定できます。また、管理 API を使用してキュー内のメッセージ数に対して一定間隔でクエリーを試行することもできますが、キューがどのように使用されているかを示すことはできません。キュー内のメッセージ数は、クライアントが送信または受信されないか、キューに送信されたメッセージの数が消費されるメッセージの数と等しくないため、定数を維持できます。どちらの場合も、キュー内のメッセージ数は、非常に異なる方法で使用される場合でも同じになります。

6.6.1. メッセージカウンターのタイプ

メッセージカウンターは、ブローカーのキューに関する追加情報を提供します。

count

ブローカーが起動してからキューに追加されたメッセージの合計数。

countDelta

最後のメッセージカウンターの更新以降にキューに追加されたメッセージの数。

lastAckTimestamp

キューからのメッセージが最後に確認された時刻のタイムスタンプ。

lastAddTimestamp

メッセージが最後にキューに追加されたタイムスタンプ。

messageCount

キューの現在のメッセージ数。

messageCountDelta

最後のメッセージカウンターの更新以降にキューから追加/削除されたメッセージの合計数。たとえば、**messageCountDelta** が **-10** の場合、全部で 10 個のメッセージがキューから削除されました。

udpateTimestamp

最後のメッセージカウンター更新のタイムスタンプ。

注記

メッセージカウンターを組み合わせて、他の意味のあるデータも判断できます。たとえば、最後の更新以降にキューから消費されたメッセージの数を正確に知るには、**countDelta** から **messageCountDelta** を減算します。

6.6.2. メッセージカウンターの有効化

メッセージカウンターはブローカーのメモリーに若干影響を与える可能性があるため、デフォルトでは無効になっています。メッセージカウンターを使用するには、まずメッセージカウンターを有効にする必要があります。

手順

1. **<broker_instance_dir>/etc/broker.xml** 設定ファイルを開きます。
2. メッセージカウンターを有効にします。

```
<message-counter-enabled>true</message-counter-enabled>
```

3. メッセージカウンター履歴およびサンプリング期間を設定します。

```
<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

キューメトリックを保存する必要のある日数。デフォルトは 10 日です。

message-counter-sample-period

メトリックを収集するためにブローカーがキューをサンプリングする頻度 (ミリ秒単位)。デフォルトは 10000 ミリ秒です。

6.6.3. メッセージカウンターの取得

管理 API を使用してメッセージカウンターを取得できます。

前提条件

- ブローカーでメッセージカウンターを有効にする必要があります。
  詳細は、「メッセージカウンターの有効化」 を参照してください。

手順

- 管理 API を使用してメッセージカウンターを取得します。

```
// 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());
```

関連情報

- メッセージカウンターの詳細は、「キュー管理操作」 を参照してください。

第7章 問題についてのブローカーの監視

AMQ Broker には、デッドロック状態などの問題についてブローカーをアクティブに監視する **Critical Analyzer** と呼ばれる内部ツールが含まれています。実稼働環境では、IO エラー、不具合のあるディスク、メモリー不足、または他のプロセスによって発生する CPU 使用率など、デッドロック状態などの問題が発生することがあります。

Critical Analyzer は、キュー配信 (ブローカーのキューへの追加) やジャーナル操作などの重要な操作の応答時間を定期的に測定します。チェックされた操作の応答時間が設定可能なタイムアウト値を超えると、ブローカーは不安定とみなされます。この場合、Critical Analyzer を設定して、ブローカーをシャットダウンするか、ブローカーを実行している仮想マシンの停止など、ブローカーを保護するアクションを実行できます。

7.1. CRITICAL ANALYZER の設定

以下の手順は、問題がないかブローカーを監視するように Critical Analyzer を設定する方法を示しています。

手順

1. **<broker_instance_dir>/etc/broker.xml** 設定ファイルを開きます。
Critical Analyzer のデフォルト設定を以下に示します。

```
<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. 以下に示すように、パラメーター値を指定します。

critical-analyzer

Critical Analyzer ツールを有効または無効にするかどうかを指定します。デフォルト値は **true** です。これは、ツールが有効であることを意味します。

critical-analyzer-timeout

Critical Analyzer によるチェックのタイムアウト (ミリ秒単位)。チェックされた操作のいずれかでかかった時間がこの値を超えると、ブローカーは不安定とみなされます。

critical-analyzer-check-period

各オペレーションの Critical Analyzer による連続チェックの間隔 (ミリ秒単位)。

critical-analyzer-policy

ブローカーがチェックに失敗し、不安定であるとみなされる場合、このパラメーターは、ブローカーがメッセージをログに記録するか (**LOG**)、ブローカーをホストする仮想マシンを停止するか (**HALT**)、ブローカーをシャットダウン (**SHUTDOWN**) します。

設定したポリシーオプションに基づいて、重要な操作の応答時間が設定されたタイムアウト値を超えると、以下のような出力が表示されます。

critical-analyzer-policy=LOG

```
[Artemis Critical Analyzer] 18:11:52,145 ERROR [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
```

また、以下のようなブローカーにスレッドダンプが表示されます。

```
[Artemis Critical Analyzer] 18:10:00,836 ERROR  [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$ConditionObject.await(AbstractQueuedSynchro
nizer.java:2039)
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolEx
ecutor.java:1088)
java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolEx
ecutor.java:809) java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1067)
java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1127)
java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
java.lang.Thread.run(Thread.java:745)
================================================================================
..... .....
================================================================================
AMQ119003: End Thread dump *
```

改訂日時: 2024-05-21