Red Hat AMQ 2021.Q3

AMQブロカーの管理

AMQ Broker 7.9での使用について

Last Updated: 2023-02-23

Red Hat AMQ 2021.Q3 AMQブロカーの管理

AMQ Broker 7.9での使用について

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

このガイドでは、AMQ Brokerの監視、管理、およびアップグレードの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、弊社 の CTO、Chris Wright のメッセージを参照してください。

第1章 概要

AMQ Broker は、ActiveMQ Artemis をベースにした高性能なメッセージング実装です。ジャーナルベースの高速なメッセージパーシスタンスを持ち、複数の言語、プロトコル、プラットフォームをサポートしています。

AMQ Brokerは、管理コンソール、管理API、コマンドライン・インターフェースなど、ブローカーインスタンスを管理したり操作したりするための複数のインターフェースを提供します。さらに、ランタイムメトリクスを収集してブローカー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監視するようにブローカーを設定したり、ブローカー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することができます。

本ガイドでは、以下のような典型的なブローカー管理タスクについて詳細な情報を提供しています。

- 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ースと管理APIの使用
- ブローカーやキューの健全性の確認
- ブローカのランタイムメトリクスの収集
- ブローカーの重要なオペレーションをプロアクティブに監視

1.1. サポートされる構成

AMQ Broker のサポートされる構成に関する最新情報は、Red Hat Customer Portal の記事Red Hat AMQ 7 Supported Configurationsを参照してください。

1.2. 本書の表記慣例

本書では、**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
本書では、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** コマンド」を参照してください。

本書におけるファイルパスの使用
本書では、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 の新バージョンをCustomer Portal に公開しました。ブローカーを最新のバージョンにアップデートして、最新の機能強化や修正を行ってください。一般的に、Red Hat は AMQ Broker の新バージョンを次の 3 つの方法でリリースしています。

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 1 つの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には、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は、本書では扱いません。これまでの AMQ Broker リリースからアップグレードする方法は、「 Migrating to Red Hat AMQ 7 」を参照してください。

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。マイナーリリースには新機能が含まれる更新、バグ修正およびセキュリティー修正が含まれます。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 では、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変更を必要としませんが、一部のリリースには設定変更が必要になる場合があります。

2.2. 旧 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 をバージョン 7.0 のいずれかのバージョンから別のバージョン 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、抽出します。

以下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拡張機能、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展開します。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展開します。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現行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拡張機能、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抽出します。.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に、**ARTEMIS_HOME** プロパティ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。

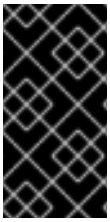
   ```
   <broker_instance_dir>\bin\artemis-service.exe start
   ```

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、さまざまな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拡張機能、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、まずバージョン 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. 現行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。作成手順により、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**ディレクトリから、7.0.xブローカーの**<broker_instance_dir>/etc/**ディレクトリ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拡張機能、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、まずバージョン 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。作成手順により、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**ディレクトリから、7.0.xブローカーの**<broker_instance_dir>/etc/**ディレクトリ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
```

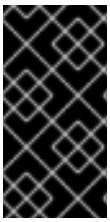
関連情報

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 7.2.0 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、さまざまな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。

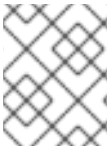


重要

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展開します。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展開します。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現行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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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抽出します。.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に、**ARTEMIS_HOME**プロパティを設定します。

```
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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。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. 7.2.x から 7.3.0 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、さまざまな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しても、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。

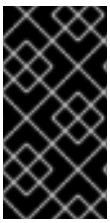
```
2019-04-11 18:00:41,334 WARN  [org.eclipse.jetty.webapp.WebAppContext] Failed startup of context o.e.j.w.WebAppContext@1ef3efa8{/dispatch-hawtio-console,null,null}{/opt/amqbroker/amq-broker-7.3.0/web/dispatch-hawtio-console.war}: java.io.FileNotFoundException: /opt/amqbroker/amq-broker-7.3.0/web/dispatch-hawtio-console.war.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

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。

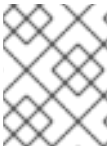


重要

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展開します。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展開します。

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

5. ブローカーが実行されている場合は停止します。

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

6. 現行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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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抽出します。.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に、**ARTEMIS_HOME**プロパティを設定します。

```
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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。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展開します。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展開します。

```
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. 現行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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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抽出します。.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-1.5.1.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
```

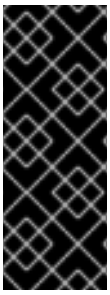
12. （オプション）ブローカーが実行され、バージョンが変更されたことを確認します。ブローカーを起動した後、**<broker_instance_dir>\log\artemis.log** ファイルを開きます。以下のような 2 つの行を見つけます。ブローカーがライブの場合にログに表示される新しいバージョン番号に注意してください。

```
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

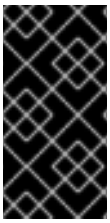
- ブローカーのインスタンス作成に関する詳細は、「ブローカーインスタンスの作成」を 参照してください。
- ブローカーインスタンスの設定ファイルおよびデータをカスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。In the **<broker_instance_dir>\etc\artemis.profile**ファイルで、ブローカーインスタンスの作成後にカスタムディレクトリーの場所を指定し、**ARTEMIS_INSTANCE_ETC_URI**プロパティを更新します。以前のバージョンでは、これらの設定ファイルとデータは、ブローカー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 は長期サポート(LTS)リリースバージョンとして指定されています。バグ修正と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 で利用できます。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。



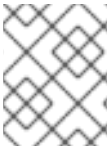
重要

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展開します。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展開します。

```
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. 現行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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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抽出します。.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。ブローカーがライブの場合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。

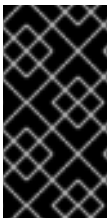
```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.9.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。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。

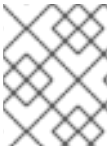


重要

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展開します。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展開します。

```
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. 現行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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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抽出します。.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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。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 上の 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展開します。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展開します。

```
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. 現行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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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 上の 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抽出します。.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。ブローカーがライブの場合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。

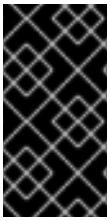
```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.11.0.redhat-00001 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。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展開します。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展開します。

```
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. 現行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。以下は例になります。

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

   b. **JAVA_ARGS**プロパティを探します。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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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抽出します。.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.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
..
```

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。ブローカーがライブの場合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。

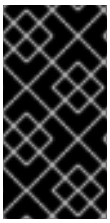
```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.13.0.redhat-00003 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。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。

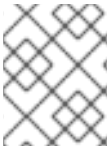


重要

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

2.7.1. Linux 上の 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-7.9.3.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-7.9.3.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展開します。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展開します。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.9.3.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. 現行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。以下は例になります。

```
ARTEMIS_HOME='/opt/redhat/amq-broker-7.9.3-redhat-1'
```

b. **JAVA_ARGS**プロパティを探します。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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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 上の 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抽出します。.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。ブローカーがライブの場合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。

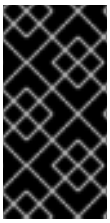
```
INFO  [org.apache.activemq.artemis.core.server] AMQ221007: Server is now live
...
INFO  [org.apache.activemq.artemis.core.server] AMQ221001: Apache ActiveMQ Artemis Message Broker version 2.16.0.redhat-00007 [0.0.0.0, nodeID=554cce00-63d9-11e8-9808-54ee759954c4]
```

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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。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 7.8.0 から 7.9.0 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 7.8.0 ブローカーインスタンスを 7.9.0 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 7.8.0 から 7.9.0 へのアップグレード
- Windows での 7.8.0 から 7.9.0 へのアップグレード

注記

ブローカーによって使用されるジャーナルの形式は、バージョン 7.9.0 で変更されました。そのため、ブローカーをバージョン 7.9.0 にアップグレードした後に、以前のバージョンにダウングレードすることはできません。

2.8.1. Linux での 7.8.0 から 7.9.0 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-7.9.3.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-7.9.3.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展開します。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展開します。

```
su - amq-broker
cd /opt/redhat
unzip amq-broker-7.9.3.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. 現行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。以下は例になります。

```
ARTEMIS_HOME='/opt/redhat/amq-broker-7.9.3-redhat-1'
```

   b. **JAVA_ARGS**プロパティを探します。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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。**<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 での 7.8.0 から 7.9.0 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抽出します。.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。ブローカーがライブの場合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カスタムディレクトリーに格納できます。これには、ブローカーインスタンスのディレクトリー外にある場所が含まれます。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ースの使用

コマンドラインインターフェース(CLI)を使用すると、対話式ターミナルを使用してメッセージブローカーと対話できます。ブローカーアクションの管理、メッセージの設定、および CLI を使用して便利なコマンドを入力できます。

コマンドラインインターフェース(CLI)により、対話プロセスを使用してユーザーおよびロールをファイルに追加できます。

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は、ログやデータファイルなど、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。

3.1.1. 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、以下の情報で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、サポートされる任意のプロトコル (CORE、MQTT、AMQP、STOMP、HORNETQ、および OPEN) からの接続を許可できます。各プロトコルには個別のポートも存在します。
- 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) は http://localhost:8161/jolokia から入手できます。

3.1.2. 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. Open a terminal.
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
```

3.1.3. 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
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
```

3.2. 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。

3.2.1. 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!
```

3.2.2. ブローカーインスタンスを正常に停止

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構成要素を使用して、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の設定方法を表しています。

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>
```

3.3. 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のインターセプトを行い、パケットの監査や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ー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。

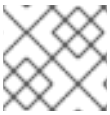
3.3.1. インターセプターの作成

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ースを実装します。

手順

1. 適切なインターフェースを実装し、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**インターフェー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**インターフェー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**インターフェー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;
  }
}
```

3.3.2. 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>
```

3.3.3. 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**がスローされます。スローされた**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クライアントの Java クラスに追加して、正しくインスタンス化および呼び出す必要があります。

3.4. ブローカーやキューの健全性の確認

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。

以下の例は、ユーティリティーを使用してヘルスチェックを実行する方法を示しています。

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一覧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一覧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。

3.5. 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ース(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一覧表示しています。

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**)
browser	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。
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。

ツール	説明
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、それを出力します。
perf-journal	現在のデータフォルダーで使用する必要のある 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一覧については、**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一覧表示します。

```
$ ./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 をベースにしています。

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。特殊なキュー処理の動作、メッセージの永続性、および管理性を提供します。複数のプロトコルと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 ブローカー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 ブローカーを管理します。
  - AMQ 管理コンソール
  - コマンドラインインターフェース(CLI)
  - 管理 API

AMQ 管理コンソールでサポートされる Web ブラウザーは Firefox および Chrome です。対応しているブラウザーのバージョンに関する詳細は、AMQ 7 でサポートされる構成 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で、**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

- 最低でも 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。アップグレードの詳細は、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 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 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>
```

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ースに変更します。以下は例になります。

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

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**のインターフェー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。以下は例になります。

```
<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>
```

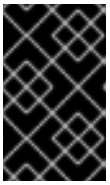
上記の設定では、以下の接続が許可されるように指定します。

- 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からローカルブローカーへ開かれたコンソールセッション内から接続この場合、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されるコンソールセッション内の **Connect** タブを使用して行います）。以下は例になります。

```
-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ア化」 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、以下を行う方法を説明します。

- ローカルブローカーからの AMQ 管理コンソールの表示
- 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。

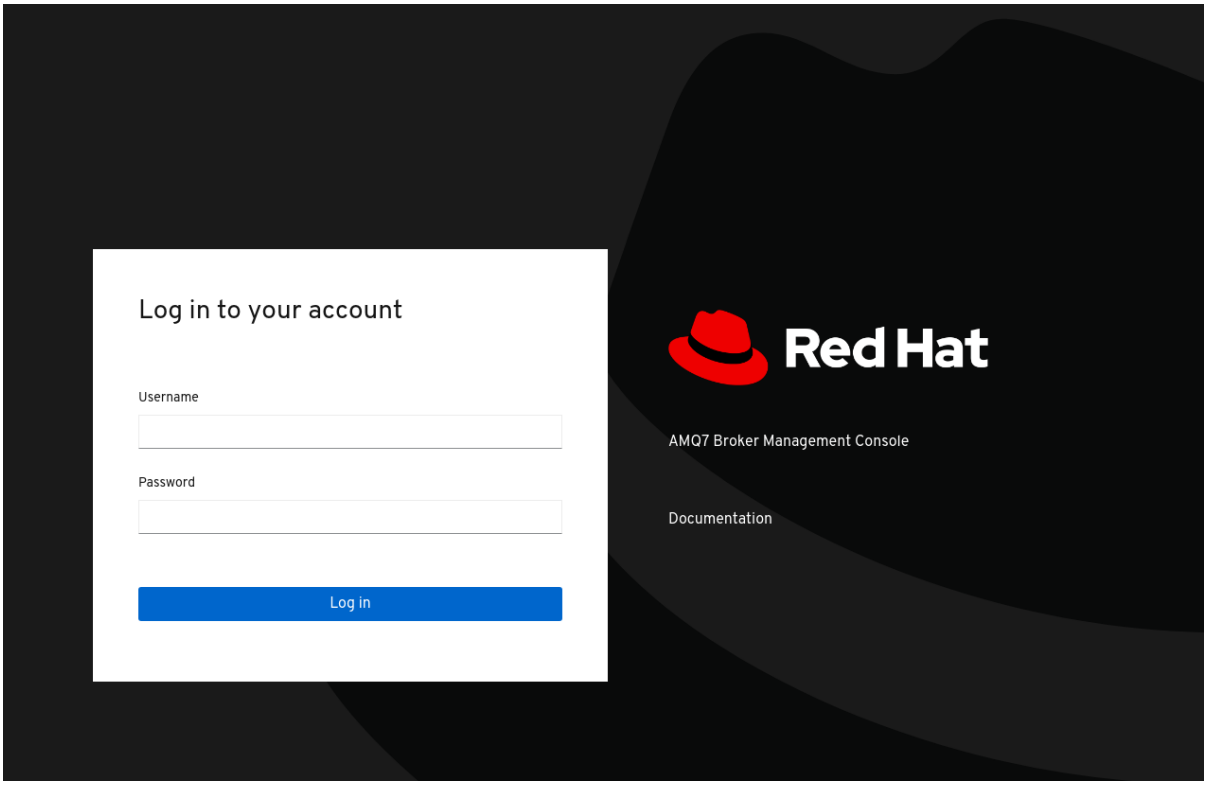
前提条件

- 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。

      名前

      my_other_brokerのようなリモート接続の名前。

      Scheme

      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。
接続テストが成功した場合は、**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。**ログイン** 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。以下は例になります。

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

4.4. AMQ 管理コンソールの設定

ユーザーアクセスおよび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 管理コンソールのセキュア化

前提条件

- Red Hat Single Sign-On 7.4

手順

1. Red Hat Single Sign-On を設定します。

a. AMQ 管理コンソールのセキュア化に使用する 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 に関連しません。

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、以下を設定します。

- 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 は Valid Redirect URIs と同じ値に発行されます。**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は、Red Hat Single Sign-On グループなどを使用して上記のロールが割り当てられていることを確認してください。

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、Red Hat Single Sign-On からベアラートークンの JAAS プリンシパルと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 のレルム名

<client-name>

Red Hat Single Sign-On 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/bootstrap.xml**ファイルを編集して、セキュリティ設定を行います。
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、クレデンシャルがない、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ア化

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 bind="https://0.0.0.0:8161"
   path="web"
   keyStorePath="<path_to_keystore>"
   keyStorePassword="<password>"
   clientAuth="<true/false>"
   trustStorePath="<path_to_truststore>"
   trustStorePassword="<password>">
   ...
</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。以下は例になります。

```
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** を含むブローカー設定ファイル のパスワードを暗号化する方法は、「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。

4.5. 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
- キュー

4.5.1. ブローカーの詳細の表示

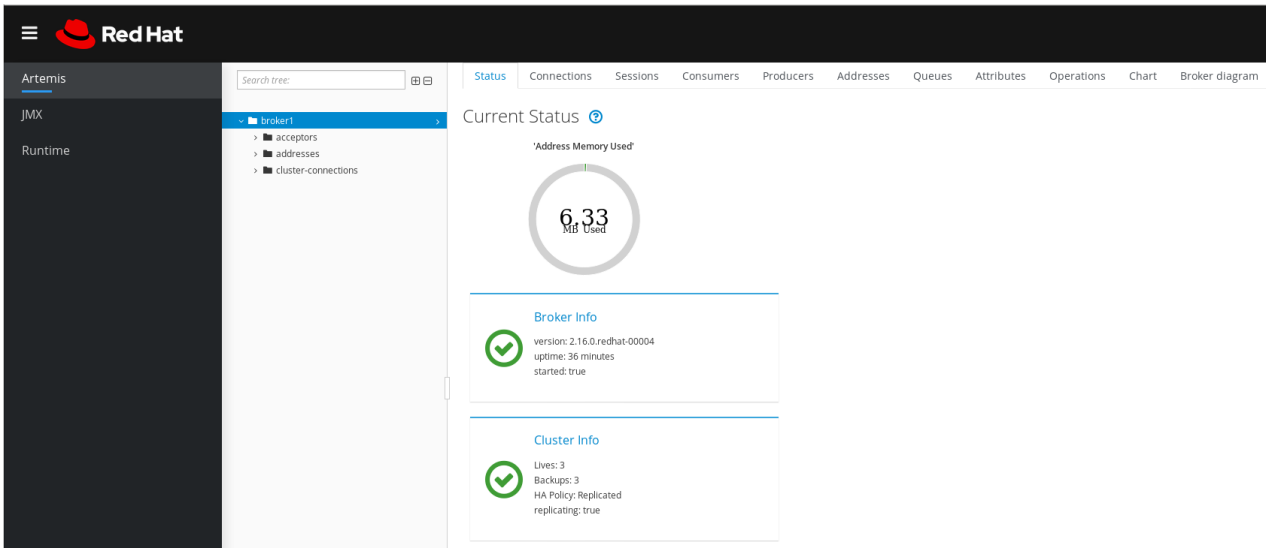
ブローカーの設定方法を確認するには、左側のメニューで **Artemis** をクリックします。フォルダーツリーでは、ローカルブローカーがデフォルトで選択されます。

メインペインで、以下のタブが利用できます。

状態

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ついての情報を表示します。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



接続

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。

コンシューマー

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。

プロデューサー

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。

キュー

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。

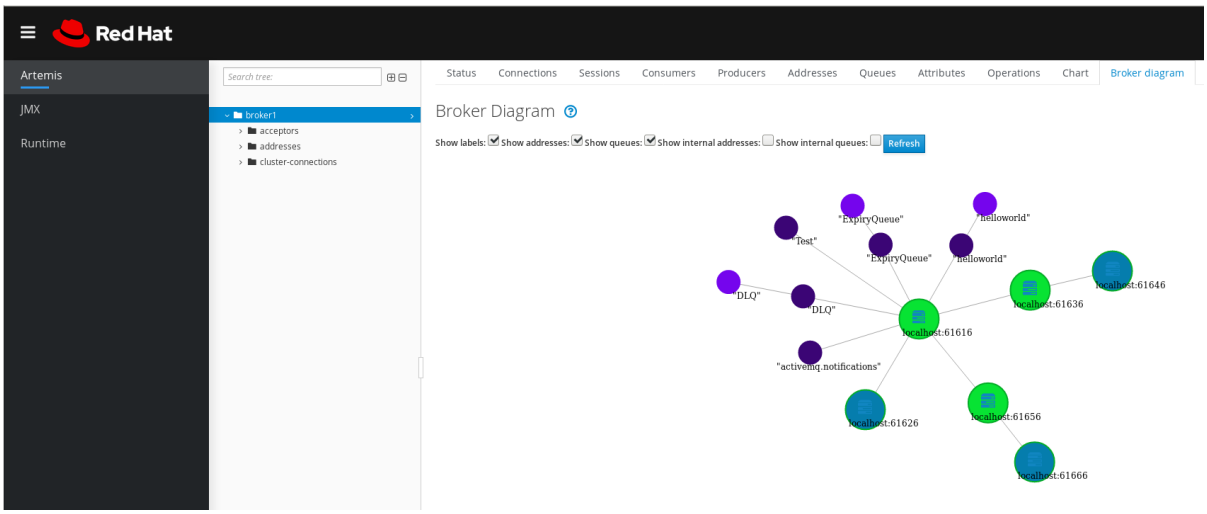
4.5.2. ブローカーダイアグラムの表示

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、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

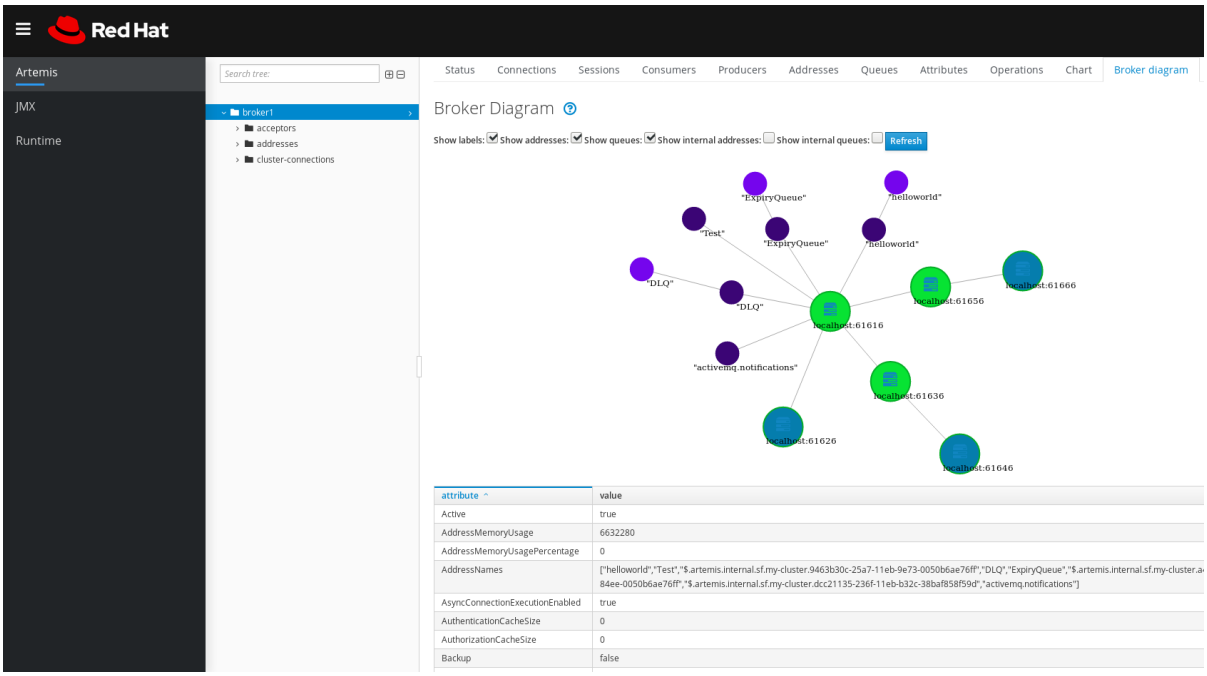
1. 左側のメニューで **Artemis** をクリックします。
2. メインペインで、Broker **diagram** タブをクリックします。
コンソールは、クラスタートポロジーの図を表示します。これには、図に示すように、クラスター内のすべてのブローカーとローカルブローカーのアドレスおよびキューが含まれます。

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、または接続されている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、以下の図では、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



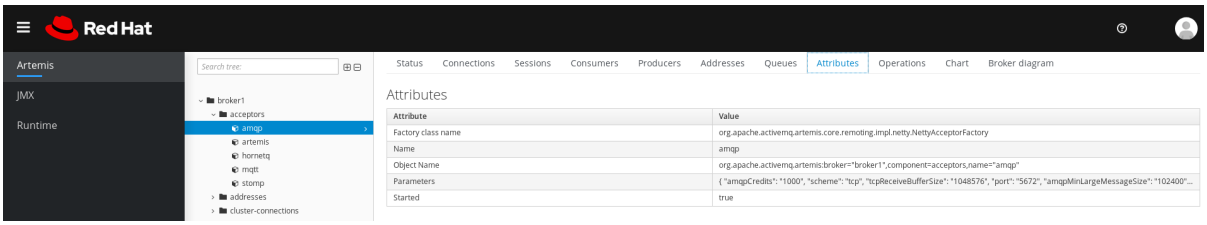
4.5.3. アクセプターの表示

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、**アクセプター** 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** タブで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。

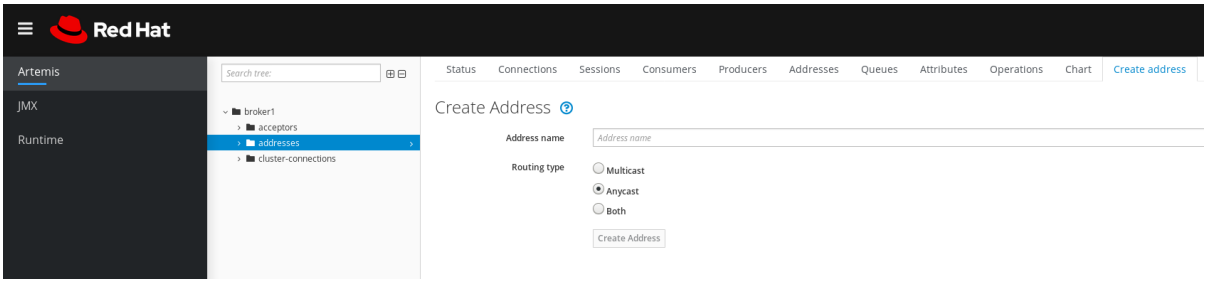
メッセージをルーティング先とルーティングする場合	このルーティングタイプを使用する...
ポイントツーポイントのため、一致するアドレス内の単一キュー。	anycast
パブリッシュ/サブスクライブ方式で、一致するアドレス内のすべてのキュー。	マルチキャスト

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、**address** 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、アドレス作成の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。クライアントが MQTT プロトコルを使用する場合の例外が 1 つ例外になります。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。

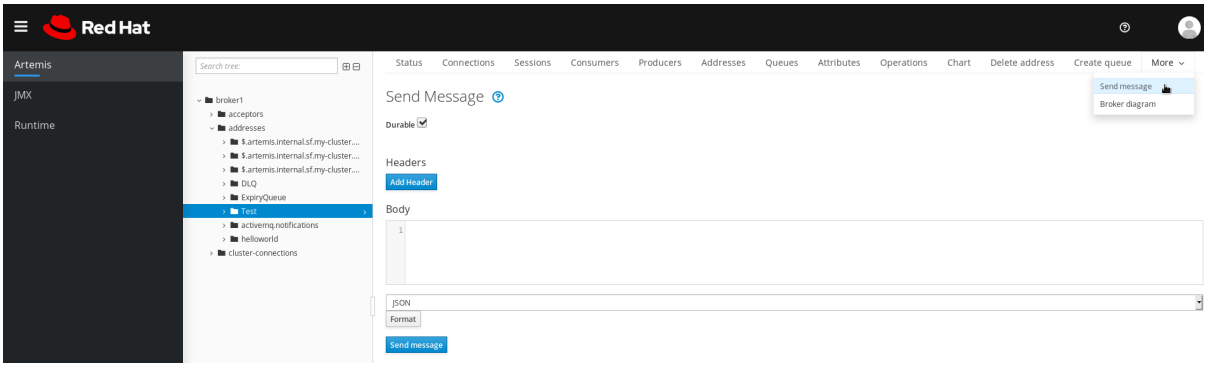
4.5.4.2. アドレスへのメッセージの送信

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、メッセージ作成のページが表示されます。

   図4.7 Send Message ページ



4. 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。
5. メッセージのボディーを入力します。
6. **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。メッセージのボディーは、選択した形式のために人間が判読可能なスタイルでフォーマットされます。
7. **Send message** をクリックします。
   メッセージは送信されます。
8. 追加のメッセージを送信するには、入力した情報を変更し、**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。

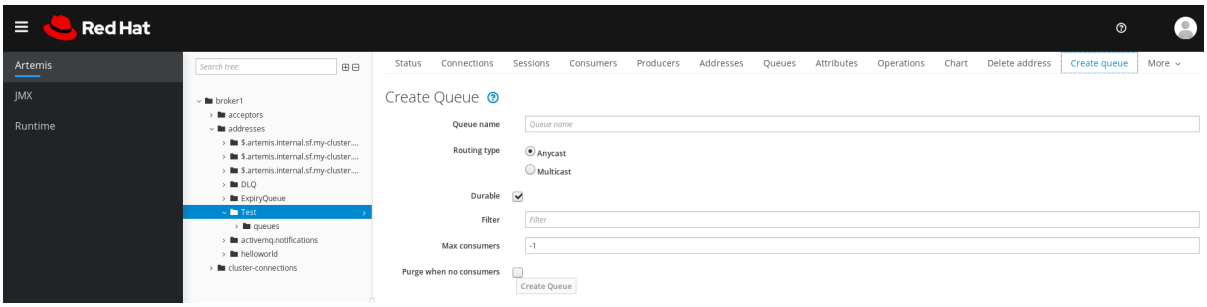
前提条件

- 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。

   フィルター

   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

   選択されている場合には、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。

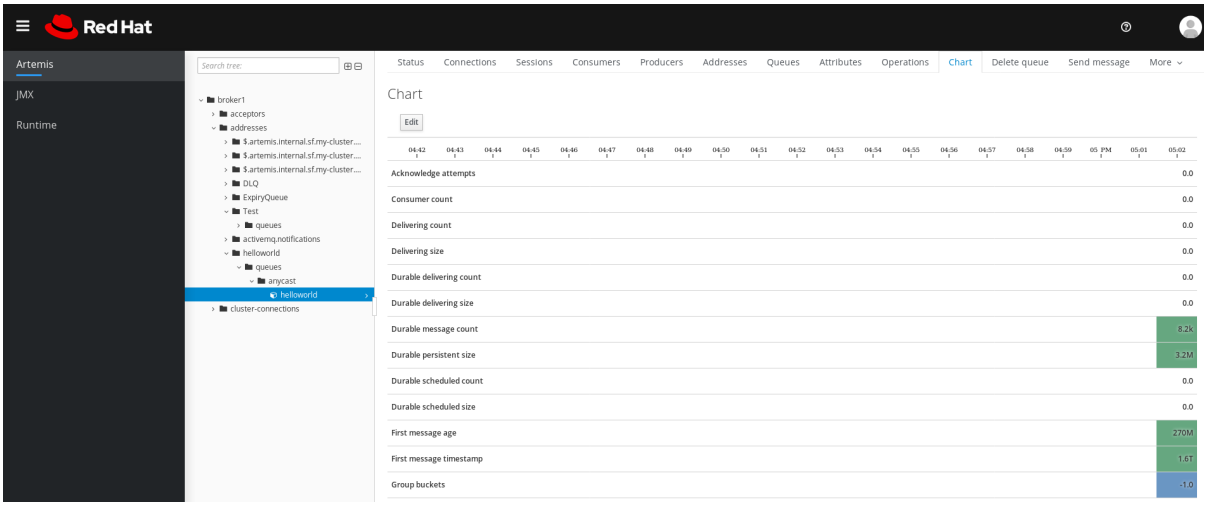
4.5.4.4. キューのステータスの確認

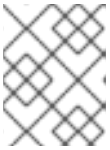
チャートはブローカー上のキューのステータスのリアルタイムビューを提供します。

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、**編集** をクリックします。
   b. **Attributes** 一覧で、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** キーを押して保持すると、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。

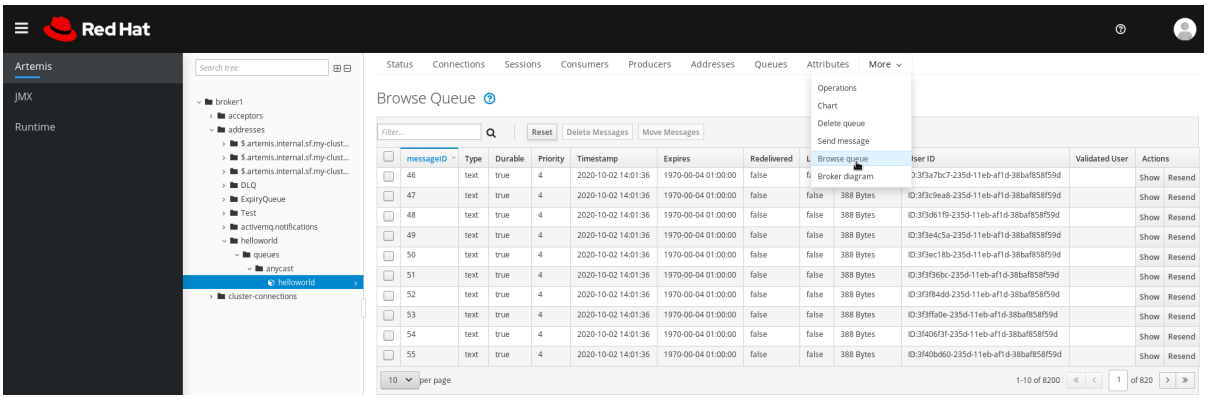
4.5.4.5. キューの参照

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 page



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。

以下を行う場合	以下を行います
メッセージの一覧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。	メッセージの一覧で、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。

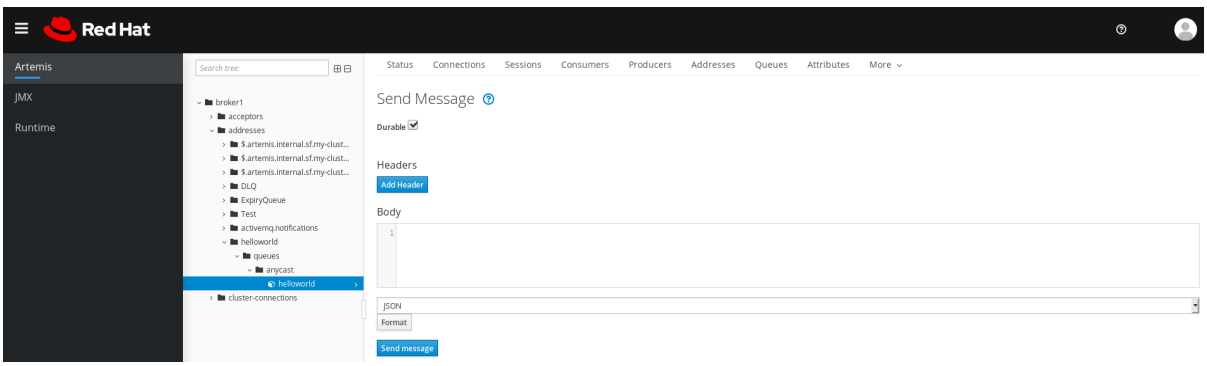
4.5.4.6. キューへのメッセージの送信

キューの作成後、メッセージを送信できます。以下の手順では、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. 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。
5. メッセージのボディーを入力します。
6. **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。メッセージのボディーは、選択した形式のために人間が判読可能なスタイルでフォーマットされます。
7. **Send message** をクリックします。メッセージは送信されます。
8. 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

以前に送信されたメッセージを再送信できます。

手順

1. 再送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異なる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するか、すべてのメッセージをパージすることができます。

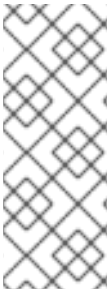
手順

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。

以下を行う場合	以下を行います
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. **削除** 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 メトリクスプラグインはインストールに含まれます。Prometheus は、大規模な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格納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クスを収集し、それらを Prometheus 形式にエクスポートします。その後、Prometheus を使用してメトリクスを確認できます。Grafana などのグラフィカルツールを使用して、データの高度な可視化を設定することもできます。



注記

Prometheus メトリクスプラグインを使用すると、Prometheus **形式** でブローカーメトリクスを収集およびエクスポートできます。ただし、Red Hat は、Prometheus 自体のインストールや設定、Grafana などの可視化ツールに対するサポートを提供**しません**。Prometheus または Grafana のインストール、設定、または実行のサポートが必要な場合は、コミュニティーサポートやドキュメントなどのリソースの製品の Web サイトを参照してください。

Prometheus プラグインによって収集されるブローカーメトリクスのほかに、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)のホストに関連するメトリクス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、およびスレッドの JVM メトリクス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クス
- Prometheus プラグインを有効にする方法
- JVM メトリクスを収集するようにブローカーを設定する方法

5.1. メトリクスの概要

ブローカーインスタンスの正常性およびパフォーマンスを監視するには、AMQ Broker に Prometheus プラグインを使用してブローカーランタイムメトリクスを監視および保存できます。AMQ Broker Prometheus プラグインはブローカーランタイムメトリクスを Prometheus 形式でエクスポートし、Prometheus 自体を使用してデータを視覚化し、クエリーを実行することができます。

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クスのより高度な可視化およびダッシュボードを設定することもできます。

プラグインが Prometheus 形式にエクスポートするメトリクスを以下に示します。

ブローカーメトリクス

artemis_address_memory_usage

インメモリーメッセージに対してこのブローカーのすべてのアドレスによって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を、**global-max-size**パラメータ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

起動してからこのブローカーに接続していたクライアントの数。

アドレスメトリクス

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るメッセージの数。

artemis_unrouted_message_count

キューバインディングにルーティング **されない** メッセージの数。

キューメトリクス

artemis_consumer_count

指定のキューからメッセージを消費するクライアントの数。

artemis_delivering_durable_message_count

指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

指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

指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

現在指定のキューにあ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的なサイズ。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

キューの作成後に指定のキューから確認応答されたメッセージの数。

artemis_messages_added

キューの作成以降、指定のキューに追加されたメッセージの数。

artemis_message_count

指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

キューの作成以降、指定のキューから削除されたメッセージの数。ブローカーは、設定された最大配信試行回数を超えたときにメッセージを強制終了します。

artemis_messages_expired

キューの作成以降、指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

指定されたキューに現在、すべてのメッセージの永続サイズ（永続および非永続の両方）。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

指定されたキューにスケジュールされたメッセージの数。

artemis_scheduled_durable_persistent_size

指定されたキューにスケジュールされたメッセージの永続サイズ。

artemis_scheduled_message_count

指定のキューにスケジュールされたメッセージの数。

artemis_scheduled_persistent_size

指定のキューにスケジュールされたメッセージの永続サイズ。

上記に記載されていない高レベルのブローカーメトリクスの場合は、低いレベルのメトリックを集計してこのメトリクスを計算できます。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** メトリクスを集約できます。

AMQ Broker のオンプレミスデプロイメントでは、ブローカーをホストする Java 仮想マシン (JVM) のメトリクスも Prometheus 形式でエクスポートされます。これは OpenShift Container Platform 上の AMQ Broker のデプロイメントには適用されません。

5.2. AMQ BROKER の PROMETHEUS メトリクスプラグインの有効化

AMQ Broker のインストール時に、Prometheus メトリクス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クスを収集し、それらを Prometheus 形式にエクスポートします。

以下の手順では、AMQ Broker で Prometheus プラグインを有効にする方法を説明します。

手順

1. AMQ Broker 7.9 の抽出したアーカイブから、ブローカーインスタンスの **lib** ディレクトリーに、Prometheus メトリクスプラグイン **.jar** ファイルをコピーします。

```
$ cp amq-broker-7.9.0/lib/artemis-prometheus-metrics-plugin-1.0.0.CR1-redhat-00010.jar
<broker_instance_dir>/lib
```

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

3. 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="org.apache.activemq.artemis.core.server.metrics.plugins.ArtemisPrometheusMetricsPlugin"/>
</metrics>
```

4. **broker.xml**設定ファイルを保存します。metrics プラグインは、Prometheus 形式でブローカーランタイムメトリクスを収集を開始します。

5.3. JVM メトリクスを収集するようにブローカーを設定する

以下の手順では、ガベージコレクション(GC)、メモリー、およびスレッドに対して Java 仮想マシン(JVM)メトリクス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 メトリクスプラグインを有効にしている。詳細は、「AMQ Broker の Prometheus メトリクス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 メトリクス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 メトリクスを収集するかどうかを指定します。以下は例になります。

```
<metrics>
   <jvm-gc>true</jvm-gc>
   <jvm-memory>true</jvm-memory
   <jvm-threads>true</jvm-threads>
   <plugin class-
name="org.apache.activemq.artemis.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 メモリーメトリクス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、有効化した JVM メトリクスの収集を開始します。これらのメトリクスは Prometheus 形式にもエクスポートされます。

5.4. 特定アドレスのメトリクスコレクションの無効化

AMQ Broker の metrics プラグインを設定する場合（Prometheus メトリックスプラグインなど）、メトリクス収集はデフォルトで有効になっています。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、メトリクス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** メトリクス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クス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 プラグインで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クス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クスを表示するには、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と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クス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クスが表示されます。
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 の 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クスは **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、時間の経過とともにクエリーするとします。この場合、メトリクス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、複数のメトリクス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（集計）で数学的な操作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 querying Prometheus 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（たとえばキューからメッセージを削除）を行う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 のアプリケーションを管理する標準的な方法は、JMX(JMX)を使用することです。
- JMS メッセージと AMQ JMS クライアントを使用して、JMS API させたmanagement 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 メッセージと 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。

各管理リソースには、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 インターフェースが存在します。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 ブローカーの管理

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 インターフェー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**.を使って構築されます。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、リフレクションや 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

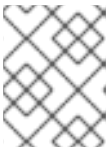
デフォルトでは、JMX はブローカーの管理に有効になっています。JMX管理を有効または無効にするには、**broker.xml**構成ファイルで**jmx-management-enabled**プロパティを設定します。

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、ブローカーへのリモート JMX アクセスは無効になっています。しかし、AMQ Broker には JMX MBean へのリモートアクセスを可能にする 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

また、'com.sun.management.jmxremote ' JVM システムプロパティーを使用した JMX アクセスを有効にすることも可能ですが、その方法はサポートされておらず、安全ではありません。JVM システムプロパティーを変更すると、ブローカーの RBAC がバイパスされる可能性があります。セキュリティーリスクを最小限に抑えるために、ローカルホストへのアクセスが限定されることを検討してください。

重要

リモート管理にブローカーの JMX エージェントを公開すると、セキュリティーに影響を及ぼします。

この手順で説明されているように設定のセキュリティーを保護するには、以下を実行します。

- 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および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

- 稼働中のブローカーインスタンス
- Javaの**jconsole**ユーティリティ

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、JMX コネクターサーバーの jconsole クエリーなどのクライアントが RMI レジストリーを確立します。たとえば、ポート 1099 でリモートアクセスを許可するには、次のコマンドを実行します。

```
<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. 以下で説明されているように、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になります。リモート JMX 接続のトンネルがファイアウォール経由で行われるのを回避するには、このプロパティーを設定します。

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ためのオブジェクト名。デフォルト値は **connector:name=rmi** です。

secured

コネクターが SSL を使用してセキュア化され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。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。パスワードは暗号化できます。

trust-store-provider

truststore 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 codec の完全修飾クラス名。これがどのように機能するかについての詳細は、パスワードマスクのドキュメント（以下にリンク）を参照してください。

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

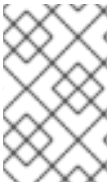
- 設定ファイルの暗号化されたパスワードの詳細は、設定ファイル の パスワードの暗号化 を参照してください。

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、Security 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

- 管理通知についての詳しい情報は、 「管理通知」 を参照してください。

6.3. JMS API を使用した AMQ ブローカーの管理

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 **queue.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、管理アドレスが管理メッセージを送受信できるようになります。

```
<security-setting-match="queue.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を記入します。
4. **QueueRequestor**を使っ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。

キューの一覧表示、作成、デプロイ、および破棄

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。キューが一時停止されると、メッセージを受信しますが、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（ある場合）。

リモート接続の一覧表示および閉じる

listRemoteAddresses()を使って、クライアントのリモートアドレスを取得します。また、**closeConnectionsForAddress()**メソッドを使って、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 を一覧表示し、 **listSessions()** を使用して指定の接続 ID の全セッションを一覧表示します。

トランザクションの管理

ブローカーがクラッシュすると、ブローカーを再起動すると、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一覧表示します（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一覧表示します。

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一覧表示、作成、破棄

デプロイされた Core Bridge を一覧表示し、**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()**メソッドを使って、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一覧表示できます。

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（例： 単一のメッセージを削除）またはフィルター（特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され、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。一致しない場合、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。たとえば、デフォルトのブローカー設定について の「Default message address settings」セクションを 参照してください。
sendMessagesToDeadLetterAddress()メソッドを使って、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。たとえば、デフォルトのブローカー設定について の「Default message address settings」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。

メッセージの一覧表示と削除

listMessages()メソッドを使用して、あるキューからメッセージを一覧表示します。**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は、**listMessageCounter()**および**listMessageCounterHistory()**メソッドを使用して、キューに対して一覧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()**メソッドを使って、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。キューが一時停止されると、メッセージを受信しますが、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（ある場合）。

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は、「 Network Connections: Acceptors and Connectors 」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) を使って、ブリッジを開始または停止します。**BridgeControl** クラスの **stop（）**（ **ObjectName org.apache.activemq.artemis:broker="<broker-name>",component=bridge,name="<bridge-name>"** またはリソース名 **ブリッジ）の stop（）メソッド。**ブリッジのパラメータは、**BridgeControl**属性を使っ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は、**BroadcastGroupControl**属性を使っ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は、**DiscoveryGroupControl**属性を使っ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一覧です。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
BINDING_ADDED (0)	**_AMQ_Binding_Type** **_AMQ_Address** **_AMQ_ClusterName** **_AMQ_RoutingName** **_AMQ_Binding_ID** **_AMQ_Distance** **_AMQ_FilterString**

通知タイプ	ヘッダー
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**
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**

通知タイプ	ヘッダー
SECURITY_PERMISSION_VIOLATION (7)	**_AMQ_Address** **_AMQ_CheckType** **_AMQ_User**
DISCOVERY_GROUP_STARTED (8)	**name**
DISCOVERY_GROUP_STOPPED (9)	**name**
BROADCAST_GROUP_STARTED (10)	**name**
BROADCAST_GROUP_STOPPED (11)	**name**
BRIDGE_STARTED (12)	**name**
BRIDGE_STOPPED (13)	**name**
CLUSTER_CONNECTION_STARTED (14)	**name**
CLUSTER_CONNECTION_STOPPED (15)	**name**
ACCEPTOR_STARTED (16)	**factory** **id**
ACCEPTOR_STOPPED (17)	**factory** **id**
PROPOSAL (18)	**_JBM_ProposalGroupId** **_JBM_ProposalValue** **_AMQ_Binding_Type** **_AMQ_Address** **_AMQ_Distance**

通知タイプ	ヘッダー
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、キューに関する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判別できます。また、管理 API を使用してキュー内のメッセージ数の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は示唆されません。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、またはキューに送信されたメッセージの数が、それから消費されるメッセージの数と等しいため、定数を保ち続けることができます。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、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

キューメトリクス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

メトリクスを収集するためにブローカーがキューをサンプルする頻度（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、他のプロセスによって生じる 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場合、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。
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 による連続的なチェックの間隔（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 WARN [org.apache.activemq.artemis.core.server]
AMQ224081: The component
org.apache.activemq.artemis.tests.integration.critical.CriticalSimpleTest$2@5af97850 is not
responsive
```

critical-analyzer-policy=HALT

> [Artemis Critical Analyzer] 18:10:00,831 ERROR [org.apache.activemq.artemis.core.server] AMQ224079: The process for the virtual machine will be killed, as component org.apache.activemq.artemis.tests.integration.critical.CriticalSimpleTest$2@5af97850 is not responsive

critical-analyzer-policy=SHUTDOWN

> [Artemis Critical Analyzer] 18:07:53,475 ERROR [org.apache.activemq.artemis.core.server] AMQ224080: The server process will now be stopped, as component org.apache.activemq.artemis.tests.integration.critical.CriticalSimpleTest$2@5af97850 is not responsive

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 WARN [org.apache.activemq.artemis.core.server] AMQ222199: Thread dump: AMQ119001: Generating thread dump
> *
> ==========
> ==== AMQ119002: Thread Thread[Thread-1 (ActiveMQ-scheduled-threads),5,main] name = Thread-1 (ActiveMQ-scheduled-threads) id = 19 group = java.lang.ThreadGroup[name=main,maxpri=10] sun.misc.Unsafe.park(Native Method)
> java.util.concurrent.locks.LockSupport.park(LockSupport.java:175)
> java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.await(AbstractQueuedSynchronizer.java:2039)
> java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:1088)
> java.util.concurrent.ScheduledThreadPoolExecutor$DelayedWorkQueue.take(ScheduledThreadPoolExecutor.java:809)
> java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1067)
> java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1127)
> java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
> java.lang.Thread.run(Thread.java:745)
> ==========
> ==== ....
> ==========
> ==== AMQ119003: End Thread dump *

改訂日時： 2022-04-03 11:52:11 +1000