Red Hat AMQ 7.7

Red Hat AMQ 7 への移行

Red Hat AMQ 7.7 向け

Last Updated: 2023-02-03

Red Hat AMQ 7.7 Red Hat AMQ 7 への移行

Red Hat AMQ 7.7 向け

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

本ガイドでは、AMQ 6 から AMQ 7 に移行する際に、注意が必要な重要な変更を説明します。

目次

第1章 はじめに

本ガイドでは、AMQ 7 の新機能や動作の変更について説明します。既存の AMQ 6 環境がある場合、本書は AMQ 7 の相違点を理解するのに役立ちます。これにより、AMQ 7 で新しいブローカーインスタンスを設定するための準備ができます。

1.1. 移行前にサポートを得る場合

実稼働環境を移行する予定の場合には、Red Hat サポート担当者からの詳細なサポートおよびガイダンスを求める必要があります。https://access.redhat.com/support/ からサポートケースを作成することができます。

1.2. サポート対象の移行パス

本書を使用して、既存の OpenWire JMS クライアントが接続できる AMQ Broker 7 設定の作成に必要な設定変更について理解することができます。

本ガイドでは、以下の機能の移行方法を説明していません。

- メッセージストア
  本ガイドでは、新しい AMQ 7 ブローカーインスタンスの設定に役立つ設定変更に関する情報を提供します。AMQ 6 ブローカーに保存されているメッセージなどのデータは移行されません。

- クライアント (OpenWire JMS クライアント以外)
  本ガイドは、既存の OpenWire JMS クライアントが接続できる AMQ 7 ブローカーインスタンスを設定するのに役立ちます。AMQ 7 ブローカーに接続できる新規クライアントの作成に関する詳細は、Red Hat カスタマーポータル のクライアントガイドを参照してください。

1.3. AMQ 7 の重要な新コンセプトについて

各 AMQ 機能エリアの特定の設定変更について理解する前に、最初に AMQ 6 と AMQ 7 の重要な概念的な相違点を理解しておく必要があります。

AMQ 7 には、いくつかの主要なアーキテクチャーの違いがあります。さらに、本リリースには新しいメッセージアドレス設定とルーティングモデルが実装されました。

1.3.1. AMQ 7 のアーキテクチャーの変更

AMQ 7 は、ブローカーへの着信ネットワーク接続、メッセージストア、およびブローカーのデプロイ方法に関して、主要なアーキテクチャーの変更を提供します。

着信接続のトランスポートコネクターの変更
AMQ 6 では、TCP (同期) や Java NIO (ノンブロック) などの異なるタイプのトランスポートコネクターを使用していました。

AMQ 7 では、使用するトランスポートタイプを選択する必要がなくなりました。異なる仮想マシンのエンティティー間の着信ネットワーク接続はすべて Netty 接続を使用します。Netty は、Java IO、Java NIO、TCP ソケット、SSL/TLS、HTTP、および HTTPS を使用するようネットワーク接続を設定できる、高パフォーマンスの低レベルネットワークライブラリーです。

メッセージストアおよびページングの変更
AMQ 7 では、ブローカーがメッセージをメモリーに保管し、それらのメッセージをディスクにページングするプロセスが異なります。

AMQ 6 でメッセージストアに使用されていた KahaDB は、高速で連続するメッセージを保存するメッセージジャーナルと、必要に応じてメッセージを取得するためのインデックスの両方で設定されました。

AMQ 7 には、追加のみのメッセージジャーナルで設定される独自のビルトインメッセージストアが含まれています。インデックスは使用されません。

これらの変更の詳細は、メッセージの永続性 を参照してください。

ブローカーデプロイメントの変更
AMQ Broker 7 では、以下のようにブローカーのデプロイメントが AMQ 6 とは異なります。

- デプロイメントメカニズム
  AMQ 6 はデフォルトで Apache Karaf コンテナーにデプロイされました。AMQ Broker 7 ではデプロイされません。

- 複数のブローカーのデプロイ
  AMQ 6 で複数のブローカーをデプロイするには、スタンドアロンブローカーのコレクションをデプロイするか (各ブローカーを個別にインストールおよび設定する必要がある)、JBoss Fuse Fabric を使用して AMQ Broker のファブリックをデプロイする必要がありました。

  AMQ Broker 7 で複数のブローカーをデプロイするには AMQ Broker 7 を一度インストールしてから、同じマシンに、必要なだけ ブローカーインスタンス を作成します。AMQ Broker 7 は、ファブリックを使用してデプロイされる想定ではありません。

1.3.2. AMQ 7 でのメッセージアドレスの変更

AMQ 7 では、すべてのメッセージングプロトコル (または JMS の場合は API) について、メッセージルーティングセマンティクスを設定する新しいアドレス設定およびルーティングモデルが導入されました。しかし、このモデルは AMQ 6 とは異なる方法でアドレス、キュー、トピック、およびルーティング機能を設定する必要があります。移行計画の一部として、新しいアドレス指定モデルとその設定要素を慎重に確認できるように準備する必要があります。

AMQ Broker 7 は、JMS 設定と非 JMS 設定を区別しません。AMQ Broker 7 は、アドレス設定、ルーティングメカニズム、およびキューを実装します。メッセージは、アドレスおよびルーティングメカニズムに基づいてメッセージをキューにルーティングすることで配信されます。

2 つの新たなルーティングメカニズムであるマルチキャストと エニーキャストにより、AMQ Broker 7 は標準のメッセージングパターンでメッセージをルーティングすることができます。マルチキャストルーティングは、アドレスのすべてのサブスクライバーがそのアドレスに送信されたメッセージを受信する、パブリッシュ/サブスクライブパターンを実装します。または、エニーキャストルーティングは、単一のキューのみがアドレスにアタッチされ、コンシューマーはラウンドロビン順でメッセージを受信するためにそのキューをサブスクライブする、ポイントツーポイントパターンを実装します。

関連情報

- AMQ Broker 7 の新しいアドレス設定モデルに関する詳細は、**AMQ Broker** の設定の アドレス、キュー、およびトピック を参照してください。

- AMQ Broker 7 でメッセージのアドレス指定が設定される方法についての詳細は、メッセージアドレスとキュー を参照してください。

1.4. AMQ 7 の新機能および既知の問題の確認

AMQ 7 に移行する前に、主な新機能、改良された機能、および既知の問題を理解する必要があります。この一覧は、Red Hat AMQ Broker 7.7 のリリースノート を参照してください。

1.5. 本書の表記慣例

本書では、**sudo** コマンドおよびファイルパスについて、以下の表記慣例を使用します。

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\...**)。

第2章 移行の準備

各機能エリアの設定変更を把握する前に、環境が移行要件を満たしていることを確認し、ブローカーインスタンスが AMQ Broker 7 でどのように設定されるかを理解する必要があります。

2.1. 移行の要件

AMQ 7 に移行する前に、お使いの環境が以下の要件を満たす必要があります。

AMQ 6 の要件

- AMQ 6.2.x 以降を実行している必要があります。
- OpenWire クライアントは、OpenWire バージョン 10 以降を使用する必要があります。

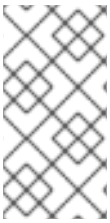
AMQ 7 の要件

- サポート対象のオペレーティングシステムおよび JVM が必要です。
  AMQ 7 のサポートされる設定は、https://access.redhat.com/articles/2791941 で確認できます。
- AMQ Broker 7 がインストールされている必要があります。
  詳細は、**AMQ Broker** の使用の AMQ Broker のインストール を参照してください。

2.2. ブローカーインスタンスの作成

AMQ 7 に移行する前に、AMQ Broker インスタンスを作成する必要があります。本書で説明されている AMQ 7 の設定の違いを理解するにつれ、このブローカーインスタンス設定することができます。

AMQ Broker をインストールした場合、AMQ Broker の実行に必要なバイナリー、ライブラリー、およびその他の重要なファイルがインストールされています。ただし、AMQ 7 では、新規ブローカーが必要な場合は、常にブローカーインスタンスを明示的に作成する必要があります。各ブローカーインスタンスは、独自の設定およびランタイムデータが含まれる個別のディレクトリーです。



注記

ブローカーのインストールと設定を個別に維持すると、中央の場所に AMQ Broker を一度インストールすれば、必要な数だけブローカーインスタンスを作成することができます。さらに、インストールと設定を個別に維持すると、必要に応じたブローカーの管理およびアップグレードが容易になります。

前提条件

- AMQ Broker 7 がインストールされている必要があります。

手順

1. ブローカーインスタンスを作成する場所に移動します。

```
$ sudo mkdir /var/lib/amq7
$ cd /var/lib/amq7
```

2. 以下のいずれかの手順を実施して、ブローカーインスタンスを作成します。

設定	実施する手順
AMQ Broker 7 が AMQ 6 と同じマシンにインストールされる	**--port-offset** パラメーターと共に **artemis create** コマンドを使用し、既存の AMQ 6 ブローカーと競合しない新しいブローカーインスタンスを作成します。  注記 AMQ Broker 7 および AMQ 6 の両方が、同じデフォルトポートのセットでクライアントトラフィックをリッスンします。したがって、潜在的な競合を回避するために、AMQ Broker ブローカーインスタンスのデフォルトのポートをオフセットする必要があります。 この例では、AMQ 6 ブローカーとは異なるポートでクライアントトラフィックをリッスンする新しいブローカーインスタンスを作成します。 $ sudo **INSTALL_DIR**/bin/artemis create mybroker --port-offset 100 --user admin --password pass --role amq --allow-anonymous true
AMQ Broker 7 および AMQ 6 が別のマシンにインストールされる	**artemis create** コマンドを使用して新規ブローカーインスタンスを作成します。 この例では、新しいブローカーインスタンスが作成され、必要な値の入力が求められます。 $ sudo **INSTALL_DIR**/bin/artemis create mybroker Creating ActiveMQ Artemis instance at: /var/lib/amq7/mybroker --user: is mandatory with this configuration: Please provide the default username: user --password: is mandatory with this configuration: Please provide the default password: password --role: is mandatory with this configuration: Please provide the default role: amq --allow-anonymous

関連情報

ブローカーインスタンスの作成に関する詳細は、**AMQ Broker** の使用の ブローカーインスタンスの作成 を参照してください。

2.3. ブローカーインスタンスのディレクトリー構造について

各 AMQ 7 ブローカーインスタンスには、独自のディレクトリーが含まれています。ディレクトリーの内容と、作成したブローカーインスタンスの設定ファイルの場所を理解する必要があります。

ブローカーインスタンスを作成すると、次のディレクトリー構造が作成されます。

```
$ ls /var/lib/amq7/mybroker
bin  data  etc  lock  log  tmp
```

BROKER_INSTANCE_DIR

ブローカーインスタンスが作成された場所。これは、AMQ Broker のインストールとは別の場所です。

/bin

ブローカーインスタンスを開始および停止するためのシェルスクリプトです。

/data

メッセージストアなどのブローカ状態データが含まれます。

/etc

ブローカーインスタンスの設定ファイルです。これらは、ブローカーインスタンスを設定するためにアクセスする必要があるファイルです。

/lock

cli.lock ファイルが含まれています。

/log

ブローカーインスタンスのログファイル。

/tmp

一時ファイル用のユーティリティーディレクトリーです。

2.4. AMQ 7 でのブローカーの設定方法

作成したブローカーインスタンスをどのように設定する必要があるか、およびどの設定ファイルを編集する必要があるかを理解する必要があります。

AMQ 6 と同様に、プレーンテキストと XML ファイルを編集して AMQ 7 ブローカーインスタンスを設定します。ブローカーの設定を変更するには、ブローカーインスタンスのディレクトリーで適切な設定ファイルを開き、XML 階層で適切な要素を見つけてから、実際の変更を行う必要があります。これには通常、XML 要素と属性の追加または削除が含まれます。

BROKER_INSTANCE_DIR/etc 内には、編集可能な設定ファイルがいくつかあります。

設定ファイル	説明
broker.xml	主要設定ファイル。AMQ 6 の **activemq.xml** と同様に、このファイルを使用して、受信ネットワーク接続のアクセプター、セキュリティー設定、メッセージアドレスなど、ブローカーのほとんどの側面を設定します。
bootstrap.xml	AMQ Broker がブローカーインスタンスを起動するために使用するファイル。これを使用して、メインブローカ設定ファイルの場所を変更し、Web サーバーを設定し、いくつかのセキュリティー設定を設定します。

設定ファイル	説明
logging.properties	このファイルを使用して、ブローカーインスタンスのロギングプロパティーを設定します。このファイルは、AMQ 6 の **org.ops4j.pax.logging.cfg** ファイルに似ています。
JAAS 設定ファイル (**login.config**、**users.properties**、**roles.properties**)	これらのファイルを使用して、ブローカーインスタンスへのユーザーアクセスの認証を設定します。

AMQ 7 への移行には、主に **broker.xml** ファイルの編集が含まれます。**broker.xml** 構造とデフォルトの設定設定の詳細については、**AMQ** ブローカーの設定 の デフォルトのブローカ設定について を参照してください。

2.5. クライアントがブローカーインスタンスに接続できることの確認

作成したブローカーインスタンスに既存のクライアントが接続できることを確認するには、ブローカーインスタンスを起動し、いくつかのテストメッセージを送信する必要があります。

手順

1. 次のいずれかのコマンドを使用して、ブローカーインスタンスを開始します。

次を行う場合:	このコマンド...
ブローカーをフォアグラウンドで開始します。	$ sudo **BROKER_INSTANCE_DIR**/bin/artemis run
ブローカーをサービスとして開始します。	$ sudo **BROKER_INSTANCE_DIR**/bin/artemis-service start

   ブローカーインスタンスが起動します。デフォルトでは、OpenWire コネクターは AMQ 6 ブローカと同じポートのブローカインスタンスで開始されます。これにより、既存のクライアントがブローカーインスタンスに接続できるようになります。

2. ブローカーインスタンスのステータスを確認する場合は、**BROKER_INSTANCE_DIR/logs/artemis.log** ファイルを開きます。

3. AMQ 6 ブローカーで、**producer** コマンドを使用して、いくつかのテストメッセージを AMQ 7 ブローカーインスタンスに送信します。
   このコマンドは、localhost でホストされ、デフォルトのアクセプターをリッスンする AMQ 7 ブローカーインスタンスに 5 つのテストメッセージを送信します。

```
JBossA-MQ:karaf@root> producer --brokerUrl tcp://0.0.0.0:61616 --message "Test message" --messageCount 5
```

   (**--port-offset** を使用して) ブローカーインスタンスを作成したときにポート番号をオフセットした場合は、ブローカー URL に正しいポート番号を使用していることを確認してください。たとえば、ポートオフセットを 100 に設定した場合、**--brokerUrl** を **tcp://0.0.0.0:61716** に設定す

る必要があります。

4. AMQ 6 ブローカーで、**consumer** コマンドを使用して、AMQ 7 ブローカーインスタンスに送信したテストメッセージを消費できることを確認します。
このコマンドは、AMQ 7 ブローカーインスタンスに送信された 5 つのテストメッセージを受け取ります。

```
JBossA-MQ:karaf@root> consumer --brokerUrl tcp://0.0.0.0:61616
```

また、AMQ 6 ブローカーの **INSTALL_DIR/data/log/amq.log** ファイルをチェックして、メッセージが送受信されたことを確認することもできます。

5. ブローカーインスタンスを実行します。

```
$ BROKER_INSTANCE_DIR/bin/artemis stop
```

第3章 内向き接続の許可

ネットワーク接続は、クライアントがブローカーインスタンスに接続する方法を定義します。AMQ 7 では、これらの接続は AMQ 6 とは異なる方法で機能し、異なる方法で設定されます。

3.1. 着信ネットワーク接続の変更

AMQ 6 と AMQ Broker 7 の両方で、クライアントがブローカーに接続する方法を定義できます。これらの接続ポイントは、AMQ 6 では トランスポートコネクター と呼ばれていましたが、AMQ Broker 7 では アクセプター と呼ばれるようになりました。

AMQ 6 は、トランスポート層 (TCP や NIO など) の複数の実装を提供しました。つまり、クライアント接続ポイントでブロッキングまたは非ブロッキングトランスポートを使用するかどうかに応じて、異なるトランスポートコネクターを使用する必要がありました。AMQ Broker 7 では、トランスポート層は Netty のみを使用します。これは、デフォルトでノンブロッキングです。AMQ Broker 7 には 2 種類のアクセプターがあります。

TCP

Netty TCP 接続は、クライアントとブローカーが同じサーバー上または物理的に離れた場所にある別の仮想マシンに配置されている場合に使用されます。
Netty はデフォルトでノンブロッキング (Java NIO) を使用します。これは、ブローカーインスタンスへのすべてのクライアント接続がノンブロッキングであることを意味します。また、WebSocket のサポートも組み込まれています。

仮想マシン内

仮想マシン内の接続は、クライアント (アプリケーションかサーバーかに関係なく) がブローカーと同じ仮想マシンにある場合に使用されます。

また、AMQ 6 では、メッセージングプロトコルごとに個別のトランスポートコネクターを使用する必要がありました。AMQ Broker 7 では、低レベルのトランスポート (TCP または仮想マシン内) は、クライアントが使用するメッセージングプロトコル (AMQP、MQTT など) とは異なります。これは、1 つのアクセプターが同じポートで複数のプロトコルを使用できることを意味します。実際、アクセプターは、使用できるプロトコルを明示的に制限しない限り、サポートされているすべてのメッセージプロトコルを受け入れます。

たとえば、AMQ Broker 7 のデフォルトのアクセプターは、すべてのメッセージプロトコルを自動的に受け入れます。

```
<acceptor name="artemis">tcp://0.0.0.0:61616?
tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=CORE,AMQP,STOMP,HORNETQ,MQTT,OPENWIRE;useEpoll=true;amqpCredits=1000;amqpLowCredits=300</acceptor>
```

3.2. アクセプターの設定方法

BROKER_INSTANCE_DIR/etc/broker.xml 設定ファイルを使用して、ブローカーインスタンスの着信クライアント接続を受け入れるようにアクセプターを設定します。

broker.xml 設定ファイルの **<acceptors>** セクションには、次のデフォルトのアクセプターが含まれています。

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

```
...
   <acceptors>
      <!-- Acceptor for every supported protocol -->
      <acceptor name="artemis">tcp://0.0.0.0:61616?
tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=CORE,AMQP,STOMP,HORNE
TQ,MQTT,OPENWIRE;useEpoll=true;amqpCredits=1000;amqpLowCredits=300</acceptor> 1

      <!-- AMQP Acceptor.  Listens on default AMQP port for AMQP traffic.-->
      <acceptor name="amqp">tcp://0.0.0.0:5672?
tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=AMQP;useEpoll=true;amqpCre
dits=1000;amqpMinCredits=300</acceptor>

      <!-- STOMP Acceptor. -->
      <acceptor name="stomp">tcp://0.0.0.0:61613?
tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=STOMP;useEpoll=true</acce
ptor>

      <!-- HornetQ Compatibility Acceptor.  Enables HornetQ Core and STOMP for legacy HornetQ
clients. -->
      <acceptor name="hornetq">tcp://0.0.0.0:5445?
protocols=HORNETQ,STOMP;useEpoll=true</acceptor>

      <!-- MQTT Acceptor -->
      <acceptor name="mqtt">tcp://0.0.0.0:1883?
tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=MQTT;useEpoll=true</accept
or>
      ...
  </core>
</configuration>
```

1 サポートされている任意のメッセージングプロトコルの着信クライアント接続を受け入れる既定のアクセプター。

ブローカーインスタンスの着信クライアント接続を設定するには、既定のアクセプターの設定プロパティーを変更するか、新しいアクセプターを追加します。この例は、OpenWire プロトコルを使用して TCP 接続を受け入れるように設定された新しいアクセプターを示しています。

```
<acceptor name="my_acceptor">tcp://0.0.0.0:61613?
tcpSendBufferSize=1048576;tcpReceiveBufferSize=1048576;protocols=OPENWIRE;useEpoll=true</
acceptor>
```

関連情報

- デフォルトのアクセプター設定について詳しくは、AMQ ブローカーの設定の デフォルトのアクセプター設定 を参照してください。
- アクセプターの設定に関する段階的な詳細は、AMQ Broker の設定の ネットワーク接続: アクセプターおよびコネクター を参照してください。
- アクセプターを設定するために使用できるすべてのプロパティーの説明については、AMQ Brokerの設定 の アクセプターおよびコネクター設定パラメーター を参照してください。

第4章 ユーザー認証

ユーザー認証を使用すると、ユーザー名を追加してセキュリティーロールに割り当てることで、ユーザーの身元を確認できます。AMQ Broker 7 では、このプロセスは AMQ 6 に似ています。ただし、用語、設定ファイルの場所、および設定構文にはいくつかの違いがあります。違いを理解したら、ブローカーインスタンスへのユーザーアクセスを設定するために使用できる方法がいくつかあります。

4.1. ユーザー認証の変更

AMQ Broker 7 と AMQ 6 の両方で、Java Authentication and Authorization Service (JAAS) に基づくプラグイン可能なログインモジュールによって認証が提供されます。ただし、AMQ 6 の **グループ** は、AMQ Broker 7 では **ロール** と呼ばれるようになりました。

さらに、ログインモジュールの名前と場所が AMQ Broker 7 で変更されました。

ログインモジュール	AMQ 6 の場所	AMQ Broker 7 の場所
Users	**etc/users.properties**	**BROKER_INSTANCE_DIR/etc/artemis-users.properties**
ロール (グループ)	**etc/groups.properties**	**BROKER_INSTANCE_DIR/etc/artemis-roles.properties**

ユーザーとロールを追加するための構文も異なります。

AMQ 6 で

権限のないユーザーを追加して、**users.properties** ファイルにパスワードとセキュリティーロールを割り当てることができます。

```
USER=PASSWORD,ROLE
```

AMQ Broker 7 で

ユーザーとロールは、個別のログインモジュールで割り当てられます。**artemis-users.properties** ファイルにユーザーを追加します。

```
USER=PASSWORD
```

artemis-roles.properties ファイルでユーザーをセキュリティーロールに割り当てます。

```
ROLE=USER
```

4.2. ユーザー認証の設定方法

ブローカーインスタンス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 7 ブローカーインスタンスにアクセスできます。追加のユーザーがブローカーインスタンスにアクセスできるようにするには、次のいずれかの方法を使用して、ブローカーのユーザー認証を設定できます。

認証方法	説明
ゲスト認証	匿名アクセスを有効にします。この設定では、クレデンシャルなしまたは誤ったクレデンシャルで接続するユーザーは自動的に認証され、特定のユーザーとロールが割り当てられます。 詳細は、**AMQ Broker の設定** の ゲストアクセスの設定 を参照してください。
基本的なユーザーとパスワード認証	各ユーザーに、ユーザー名とパスワードを定義してセキュリティーロールを割り当てる必要があります。ユーザーは、これらの認証情報を使用してのみブローカーインスタンスにアクセスできます。 詳細については、**AMQ Broker の設定** の ユーザーの追加 を参照してください。
証明書ベースの認証	ユーザーは SSL 証明書を使用して認証されます。 詳細は、**AMQ Broker の設定** の 証明書ベースの認証の追加 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、**AMQ Broker の設定** の 認証に LDAP を使用 を参照してください。

第5章 メッセージアドレスとキュー

AMQ 7 では、新しい柔軟なアドレッシングモデルが導入され、あらゆるメッセージングプロトコルで機能する標準のメッセージングパターンを定義できます。そのため、キューとトピックのような動作を設定するプロセスが大幅に変更されました。

5.1. 変更への対応

AMQ 6 では、直接設定可能な宛先として、キュー、トピック、永続サブスクリプションなどの JMS の概念が実装されました。

例: AMQ 6 のデフォルトのキューとトピックの設定

```
<destinations>
    <queue physicalName="my-queue" />
    <topic physicalName="my-topic" />
</destinations>
```

AMQ Broker 7 は、アドレス、ルーティングタイプ、およびキューを使用して、キューおよびトピックのような動作を実現します。**address** はメッセージングエンドポイントを表します。キュー はアドレスに関連付けられています。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にメッセージを配信する方法を定義します。2 つのルーティングタイプがあります。エニーキャスト は一致するアドレス内の単一のキューにメッセージを配信し、マルチキャスト はアドレスに関連付けられたすべてのキューにメッセージを配信します。

キューをアドレスとルーティングタイプに関連付けることで、ポイントツーポイント (キュー) やパブリッシュ/サブスクライブ (トピックのような) など、さまざまなメッセージングパターンを実装できます。

例: AMQ Broker 7 でのポイントツーポイントアドレスの設定

この例では、ブローカーが **address.foo** でメッセージを受信すると、メッセージは **my-queue** にルーティングされます。複数のエニーキャストキューがアドレスに関連付けられている場合、メッセージはキュー全体に均等に分散されます。

```
<address name="address.foo">
  <anycast>
    <queue name="my-queue"/>
  </anycast>
</address>
```

例: AMQ Broker 7 でのパブリッシュ/サブスクライブアドレスの設定

この例では、ブローカーが **topic.foo** でメッセージを受信すると、メッセージのコピーが **my-topic-1** と **my-topic-2** の両方にルーティングされます。

```
<address name="topic.foo">
  <multicast>
    <queue name="my-topic-1"/>
    <queue name="my-topic-2"/>
  </multicast>
</address>
```

関連情報

- AMQ Broker 7 のアドレス指定モデルの詳細については、**AMQ Broker** の設定の アドレス、キュー、およびトピック を参照してください。

5.2. アドレスの設定方法

BROKER_INSTANCE_DIR/etc/broker.xml 設定ファイルを使用して、ブローカーインスタンスのアドレスとキューを設定します。

broker.xml 設定ファイルの **<addresses>** セクションには、次のデフォルトのアドレス指定設定が含まれています。Dead Letter Queue (**DLQ**) と Expiry Queue (**ExpiryQueue**) のデフォルトエントリーがあります。

```
<addresses>
  <address name="DLQ">
    <anycast>
      <queue name="DLQ" />
    </anycast>
  </address>
    <address name="ExpiryQueue">
      <anycast>
        <queue name="ExpiryQueue" />
      </anycast>
  </address>
</addresses>
```

次のいずれかの方法を使用して、ブローカーインスタンスのアドレス指定を設定できます。

メソッド	説明
アドレスを手動で設定する	アドレスでメッセージを受信するときにブローカーが使用するルーティングタイプとキューを定義します。次の方法でアドレスを設定できます。 - **AMQ Broker の設定** の ポイントツーポイントメッセージング用のアドレスの設定 - **AMQ Broker の設定** の 2 つのキューを使用したポイントツーポイントアドレスの設定 - **AMQ Broker の設定** の パブリッシュ/サブスクライブメッセージングのアドレスの設定 - **AMQ Broker の設定** の ポイントツーポイントおよびパブリッシュ - サブスクライブを使用するためのアドレスの設定 - **AMQ ブローカ** の 設定のサブスクリプションキューの設定
アドレスを自動的に作成するようにブローカーを設定する	自動的に作成するアドレスの接頭辞とルーティングタイプを指定します。ブローカーが接頭辞に一致するアドレスでメッセージを受信すると、アドレスとルーティングタイプが自動的に作成されます。すべてのキューが削除されたときにアドレスが自動的に削除されるように指定したり、コンシューマーやメッセージがなくなったときにそのキューが自動的に削除されるように指定したりすることもできます。 詳細については、**AMQ Broker の設定** の キューとアドレスの自動作成と削除 を参照してください。

メソッド	説明

第6章 セキュリティー

AMQ Broker 7 は、着信ネットワーク接続を保護するためのトランスポート層セキュリティーと、それぞれのアドレスに基づいてキューへのアクセスを保護するための承認を提供します。これらの領域の両方で、セキュリティーモデルは AMQ 6 に非常に似ています。ただし、設定プロセスは異なります。

6.1. トランスポート層セキュリティーの設定方法

AMQ 6 と同様に、AMQ Broker 7 では、SSL/TLS を使用して着信ネットワーク接続を保護できます。ただし、設定構文と設定プロパティーにはいくつかの違いがあります。

AMQ 6 では、SSL コンテキストを作成してキーストアとトラストストアを定義し、保護する各トランスポートコネクターに SSL 属性を追加することで、トランスポート層セキュリティーを設定していました。

AMQ Broker 7 では、トランスポート層は SSL をネイティブに使用する Netty に基づいています。つまり、トランスポート層のセキュリティーを設定するには、保護する各アクセプターに必要な SSL 属性を追加するだけです。別の SSL コンテキストを追加する必要はありません。

たとえば、次の設定は、OpenWire クライアントからの安全な接続を受け入れます。

AMQ 6 で

1. **INSTALL_DIR/etc/activemq.xml** ファイルで SSL コンテキストを定義します。

```
<sslContext>
   <sslContext keyStore="file:${activemq.conf}/broker.ks" keyStorePassword="password"/>
</sslContext>
```

2. ブローカー設定ファイルで、トランスポートコネクターを作成して、OpenWire クライアントからの安全な接続を受け入れます。

```
<transportConnector name="ssl" uri="ssl://localhost:61617?transport.needClientAuth=true"/>
```

AMQ Broker 7 で

- **BROKER_INSTANCE_DIR/etc/broker.xml** 設定ファイルで、アクセプターを作成または更新して、OpenWire クライアントからのセキュアな接続を受け入れます。

```
<acceptor name="netty-ssl-acceptor">tcp://localhost:61617?
sslEnabled=true;keyStorePath=${data.dir}/../etc/broker.ks;keyStorePassword=password;needCl
ientAuth=true</acceptor>
```

一方向または双方向の TLS を設定できます。次の表で、これらの方法について説明します。

メソッド	説明
一方向 TLS	ブローカーのみが証明書を表示します。この方法では、サーバー側の証明書用の Java KeyStore が必要です。 詳細は、**AMQ Broker の設定** の ネットワーク接続の保護 を参照してください。

メソッド	説明
双方向 TLS (相互認証)	ブローカーとクライアントの両方が証明書を提示します。この方法では、サーバー側の証明書用の Java KeyStore と、ブローカーが信頼するクライアントのキーを保持する TrustStore が必要です。 詳細は、**AMQ Broker の設定** の ネットワーク接続の保護 を参照してください。

注記

AMQ Broker 7 の既存のキーストアとトラストストアを再利用するには、それらを AMQ Broker 7 ブローカーインスタンスにコピーします。

関連情報

- すべてのトランスポート層セキュリティー設定プロパティーの完全なリストについては、**AMQ Broker** の設定 の Netty TLS パラメーター を参照してください。

6.2. 承認

AMQ Broker 7 は、アドレスに基づいてセキュリティー設定をキューに適用するロールベースのセキュリティーモデルを提供します。このセキュリティーモデルは AMQ 6 に似ています。ただし、権限とワイルドカードの構文は異なり、承認の設定も異なります。

6.2.1. 承認の変更

AMQ Broker 7 は、AMQ 6 とは異なる一連の権限と、わずかに異なるワイルドカード構文を使用します。

次の表では、AMQ 6 および AMQ Broker 7 で適用できるさまざまな種類のアクセス許可について説明します。

AMQ 6 の許可	AMQ Broker 7 の対応する権限
write	**send**
read	**consume** **browse**

AMQ 6 の許可	AMQ Broker 7 の対応する権限
admin	**createAddress** **deleteAddress** **createNonDurableQueue** **deleteNonDurableQueue** **createDurableQueue** **deleteDurableQueue** **manage**

AMQ Broker 7 のアクセス許可の詳細については、**AMQ Broker** の設定の アクセス許可の設定 を参照してください。

AMQ Broker 7 では、一致するアドレスのワイルドカード構文も異なります。

次を行う場合:	AMQ 6 で	AMQ Broker 7 で
パス内の単語を区切る	**.**	**.**
1 つの単語に一致する	*****	*****
任意の単語を再帰的に一致させる	**>**	**#**

6.2.2. 認可の設定方法

BROKER_INSTANCE_DIR/etc/broker.xml 設定ファイルを使用して、セキュリティー設定をキューに割り当てます。

broker.xml 設定ファイルには、次のデフォルトのセキュリティー設定が含まれており、ブローカインスタンスの作成時に作成したデフォルトロールのすべてのアドレスとキューへの完全なアクセスを提供します。

```
<configuration ...>
  <core ...>
    ...
    <security-settings>
      <security-setting match="#"> 1
        <permission type="createNonDurableQueue" roles="admin"/> 2
        <permission type="deleteNonDurableQueue" roles="admin"/>
        <permission type="createDurableQueue" roles="admin"/>
        <permission type="deleteDurableQueue" roles="admin"/>
        <permission type="createAddress" roles="admin"/>
        <permission type="deleteAddress" roles="admin"/>
        <permission type="consume" roles="admin"/>
        <permission type="browse" roles="admin"/>
```

```
      <permission type="send" roles="admin"/>
      <permission type="manage" roles="admin"/>
    </security-setting>
  </security-settings>
  ...
 </core>
</configuration>
```

1. 一連のセキュリティー許可が適用されるアドレスまたはアドレス 接頭辞。アクセス許可は、アドレスに一致する一連のキューに適用されます。この例では、**#** ワイルドカードはすべてのアドレスに一致します。

2. ロールに付与される権限。この例では、**admin** ロールに属するすべてのユーザーに、非永続キューを作成する権限が付与されています。

キューに一致するアドレスを指定し、各権限タイプに付与する必要があるロールを指定することで、キューまたはキューのセットの承認を設定できます。

関連情報

- AMQ Broker の設定で 権限を設定する

第7章 リソースの制限とポリシー

リソース制限とポリシーを定義して、ブローカーインスタンスがメッセージを処理する方法の重要な側面を制御できます。これらのリソース制限とポリシーを設定するプロセスは、AMQ Broker 7 と AMQ 6 では異なり、多くの設定プロパティーが変更されています。

7.1. リソース制限とポリシーの設定方法

AMQ 6 では、リソースの制限とポリシーは、ブローカーの設定ファイルで宛先ポリシーとして設定されていました。

AMQ Broker 7 では、アドレスまたはアドレスセットのリソース制限とポリシーを定義します。ブローカーインスタンスがメッセージを受信すると、メッセージのアドレスに対して定義されたリソース制限とポリシーがメッセージに適用されます。

AMQ Broker 7 でリソース制限とポリシーを設定するには、**BROKER_INSTANCE_DIR/etc/broker.xml** 設定ファイルを使用して、適切な設定プロパティーで **<address-setting>** 要素を定義します。

broker.xml 設定ファイルには、次のデフォルトのアドレス設定設定が含まれています。

```
<address-settings>
   <!-- if you define auto-create on certain queues, management has to be auto-create -->
   <address-setting match="activemq.management#"> (1)
      <dead-letter-address>DLQ</dead-letter-address>
      <expiry-address>ExpiryQueue</expiry-address>
      <redelivery-delay>0</redelivery-delay>
      <!-- with -1 only the global-max-size is in use for limiting -->
      <max-size-bytes>-1</max-size-bytes>
      <message-counter-history-day-limit>10</message-counter-history-day-limit>
      <address-full-policy>PAGE</address-full-policy>
      <auto-create-queues>true</auto-create-queues>
      <auto-create-addresses>true</auto-create-addresses>
      <auto-create-jms-queues>true</auto-create-jms-queues>
      <auto-create-jms-topics>true</auto-create-jms-topics>
   </address-setting>
   <!--default for catch all-->
   <address-setting match="#"> (2)
      <dead-letter-address>DLQ</dead-letter-address>
      <expiry-address>ExpiryQueue</expiry-address>
      <redelivery-delay>0</redelivery-delay>
      <!-- with -1 only the global-max-size is in use for limiting -->
      <max-size-bytes>-1</max-size-bytes>
      <message-counter-history-day-limit>10</message-counter-history-day-limit>
      <address-full-policy>PAGE</address-full-policy>
      <auto-create-queues>true</auto-create-queues>
      <auto-create-addresses>true</auto-create-addresses>
      <auto-create-jms-queues>true</auto-create-jms-queues>
      <auto-create-jms-topics>true</auto-create-jms-topics>
   </address-setting>
</address-settings>
```

(1) デフォルトの管理アドレス設定。ネストされたリソース制限とポリシーは、アドレスが **activemq.management#** と一致するすべてのメッセージに適用されます。

(2) デフォルトのアドレス設定**#** ワイルドカードはすべてのアドレスに一致するため、定義されたリ

ソース制限とポリシーがすべてのメッセージに適用されます。

リソースの制限とポリシーを設定するには、(**<address-setting>** を使用して) アドレスまたはアドレスのセットを指定し、リソースの制限とポリシーのプロパティーをそれに追加します。これらのプロパティーは、指定したアドレス (または複数のアドレス) に送信される各メッセージに適用されます。

関連情報

- ワイルドカードを使用して一連のアドレスを照合する方法の詳細については、**AMQ Broker** の設定 の AMQ Broker ワイルドカードの構文 を参照してください。

7.2. リソース制限とポリシー設定プロパティー

AMQ 6 と同様に、AMQ Broker 7 ではリソース制限とポリシーを追加して、ブローカーがメッセージの配信方法と配信時期、配信の試行回数、メッセージの有効期限などの特定の側面を処理する方法を制御できます。ただし、これらのリソース制限とポリシーを定義するために使用する設定プロパティーは、AMQ Broker 7 では異なります。

このセクションでは、AMQ 6 の **<policyEntry>** 設定プロパティーを AMQ Broker 7 の同等の **<address-setting>** プロパティーと比較します。AMQ Broker 7 の各設定プロパティーの詳細については、**AMQ Broker** の設定の アドレス設定の構成要素 を参照してください。

7.2.1. キュー管理設定プロパティー

次の表では、AMQ 6 のキュー管理設定プロパティーと AMQ Broker 7 の同等のプロパティーを比較しています。

設定するには	AMQ 6 で	AMQ Broker 7 で
メモリー制限	**memoryLimit** destination のメモリー制限を設定します。デフォルト値は **none** です。	**<max-size-bytes>** **アドレス** のメモリー制限を設定します。デフォルトは **-1** (制限なし) です。
キュー内の優先順位によるメッセージの順序	**prioritizedMessages** これはデフォルトではオフになっています。つまり、メッセージは (ブローカーではなく) コンシューマーで優先されるため、コンシューマーでのメッセージの優先度に基づいて順序付けられます。	メッセージは、キュー内で優先度順に自動的に並べられます。
ブローカが期限切れメッセージをスキャンする頻度	**expiredMessagesPeriod**	**<message-expiry-scan-period>** デフォルトは 30000 ミリ秒です。

設定するには	AMQ 6 で	AMQ Broker 7 で
ブローカが一定期間非アクティブな宛先を削除するかどうか	**gcInactiveDestinations** デフォルトは **false** です。	該当なし。ただし、自動作成されたキューの場合、最後のコンシューマーが切り離されたときにキューが自動的に削除されるように設定できます。詳細については、**AMQ Broker の設定** の キューとアドレスの自動作成と削除 を参照してください。
非アクティブタイムアウト	**inactiveTimeoutBeforeGC** デフォルトは 60 秒です。	該当なし。ただし、自動作成されたキューの場合、最後のコンシューマーが切り離されたときにキューが自動的に削除されるように設定できます。詳細については、**AMQ Broker の設定** の キューとアドレスの自動作成と削除 を参照してください。
ブローカがキューからディスパッチするときに別のスレッドを使用するかどうか	**optimizedDispatch** デフォルトは **false** です。	アドレスやキューには設定できません。ただし、メッセージが到着する着信接続から制御できます。アクセプタまたはコネクターで **directDeliver** プロパティーを使用して、メッセージが到着したのと同じスレッドでメッセージを配信するかどうかを制御します。詳細については、**AMQ Broker の設定** の アクセプターとコネクターの設定パラメータ を参照してください。

7.2.2. プロデューサーポリシーの設定プロパティー

次の表では、AMQ 6 のプロデューサーポリシー設定プロパティーと AMQ Broker 7 の同等のプロパティーを比較しています。

設定するには	AMQ 6 で	AMQ Broker 7 で

設定するには	AMQ 6 で	AMQ Broker 7 で
プロデューサーフロー制御	**producerFlowControl** プロデューサーを調整するようにブローカーを設定します。プロデューサーの承認を差し控えるか、**javax.jms.ResourceAllocationException** 例外を発生させ、ローカルリソース (メモリーやストレージなど) が使い果たされたときにそれをクライアントに伝播することによって、スロットリングが実現されます。デフォルトは **true** です。	アドレスについては、**<max-size-bytes>** をプロデューサーを調整するサイズに設定してから、**<address-full-policy>** を **BLOCK** に設定します。 これら 2 つのプロパティーを設定すると、既存の AMQ 6 OpenWire プロデューサーも抑制されます。
プロデューサーが一度にリクエストできるクレジットの量	該当なし。	**<producer-window-size>** ウィンドウサイズを制限すると、プロデューサーが一度に実行中に保持できるバイト数に制限が設定され、リモート接続が過負荷になるのを防ぐことができます。

7.2.3. コンシューマーポリシーの設定プロパティー

次の表では、AMQ 6 のサーバー側の宛先ポリシー設定プロパティーと AMQ Broker 7 の同等のプロパティーを比較しています。これらのプロパティーは、OpenWire クライアントにのみ適用されます。

設定するには	AMQ 6 で	AMQ Broker 7 で
キューのプリフェッチ	**queuePrefetch**	ブローカーには同等のものはありません。ただし、接続 URL で、または **ActiveMQConnectionFactory** API で直接 **consumerWindowSize** を設定することにより、コンシューマーでバッファーされるメッセージの最大サイズ (バイト単位) を設定できます。
キューからメッセージをディスパッチするときにコンシューマーの優先順位を使用するかどうか	**useConsumerPriority** デフォルトは **true** です。	この機能は AMQ Broker 7 にはありません。

設定するには	AMQ 6 で	AMQ Broker 7 で
前のメッセージが配信されたが確認応答されていない場合に、ブローカーがプリフェッチされたメッセージをディスパッチできるようにするために、プリフェッチ拡張機能を使用するかどうか	**usePrefetchExtension** デフォルトは **true** です。	この機能は AMQ Broker 7 にはありません。
最初の再配達の遅延	**initialRedeliveryDelay** デフォルトは 1000 ミリ秒です。	該当なし。ブローカーインスタンスはこれを自動的に処理します。
キャンセルされたメッセージの再配信を試みるまでの待機時間	**redeliveryDelay** **initialRedeliveryDelay** が **0** に設定されている場合の配信遅延。デフォルトは 1000 ミリ秒です。	**<redelivery-delay>** デフォルトは 0 ミリ秒です。
指数バックオフ	**useExponentialBackoff** デフォルトは **false** です。	該当なし。他のコンシューマーポリシー設定プロパティーのいずれかを使用して、コンシューマーの再配信を設定できます。
バックオフ乗数	**backOffMultiplier** デフォルトは 5 です。	**<redelivery-multiplier>** redelivery-delay に適用する乗数。デフォルト値は 1.0 です。
キャンセルされたメッセージがブローカーの配信不能キューに返される前に再配信できる最大回数	**maximumRedeliveries** デフォルト値は 6 です。	**<max-delivery-attempts>** デフォルトは 10 です。
再配信遅延の最大値	**maximumRedeliveryDelay** これは、**useExponentialBackoff** プロパティーが設定されている場合にのみ適用されます。デフォルトは -1 (最大再配信遅延なし) です。	**<max-redelivery-delay>** デフォルトは 0 です。

設定するには	AMQ 6 で	AMQ Broker 7 で
クライアントが 1 秒間に消費できるメッセージの数	該当なし。	ブローカーには同等のものはありません。ただし、**consumerMaxRate** を接続 URL に設定するか、**ActiveMQConnectionFactory** API に直接設定することで、これをコンシューマーに設定できます。 **consumerMaxRate** プロパティーは、クライアントがバッファーに保持するメッセージの数には影響しません。したがって、クライアントのレート制限が遅く、ウィンドウサイズが大きい場合、クライアントの内部バッファーはすぐにメッセージでいっぱいになります。

7.2.4. 遅いコンシューマー処理の設定プロパティー

AMQ 6 と同様に、AMQ Broker 7 は遅いコンシューマーを検出し、一貫して遅いコンシューマーを自動的に停止できます。これは、AMQ 6 ではデフォルトで有効になっていましたが、AMQ Broker 7 ではデフォルトで無効になっています。

コンシューマーが遅いとブローカが判断する方法も異なります。AMQ Broker 7 では、コンシューマーが確認したメッセージの数に基づいて、コンシューマーは遅いと見なされます。AMQ 6 では、コンシューマーはプリフェッチバッファーがいっぱいであることに基づいて遅いと見なされていました (バッファーが常にいっぱいである場合、クライアントがメッセージを消費するのが遅すぎる可能性があります)。

次の表では、AMQ 6 の遅いコンシューマー処理の設定プロパティーと、AMQ Broker 7 の同等のプロパティーを比較しています。

設定するには	AMQ 6 で	AMQ Broker 7 で
コンシューマーが中止される前に遅いと見なされる回数	**maxSlowCount** デフォルトは -1(制限なし) です。	該当なし。他の出力 コンシューマー処理プロパティーを使用して、出力 コンシューマーを制御できます。
コンシューマーがアボートされるまでに継続的に低速になることができる時間の長さ	**maxSlowDuration** デフォルトは 30000 ミリ秒です。	**<slow-consumer-threshold>** AMQ Broker 7 では、これはコンシューマーが遅いと見なされる前のメッセージ消費の最小レートです (1 秒あたりのメッセージ数で測定)。デフォルトは -1 (しきい値なし) です。

設定するには	AMQ 6 で	AMQ Broker 7 で
ブローカが遅いコンシューマーの別のチェックを実行するまで待機する時間	**checkPeriod** デフォルトは 30000 ミリ秒です。	**<slow-consumer-check-period>** AMQ Broker 7 では、これは秒単位で測定されます。デフォルトは 5 です。
ブローカーが低速のコンシューマーと共に接続を閉じる必要があるかどうか	**abortConnection** デフォルトは **false** です。	該当なし。AMQ Broker 7 では、低速のコンシューマーが中止されると、接続も閉じられます。
遅いコンシューマーが検出された場合に適用するポリシー。	該当なし。	**<slow-consumer-policy>** デフォルトは **NOTIFY** で、**CONSUMER_SLOW** 管理通知をアプリケーションに送信します。 **KILL** ポリシーを使用して、コンシューマーの接続を閉じることもできます。ただし、これはその接続を使用する他のクライアントスレッドに影響を与えます。

関連情報

- 低速のコンシューマーの処理方法の詳細については、**AMQ Broker** の設定の 低速のコンシューマーの処理 を参照してください。

7.2.5. メッセージページングの設定プロパティー

AMQ Broker 7 では、ブローカーがメッセージをメモリーに保存してディスクに保存するプロセスが AMQ 6 とは大きく異なります。したがって、AMQ 6 のページング設定プロパティーのほとんどは、AMQ Broker 7 には適用されません。

AMQ Broker 7 では、ページングはメッセージアドレスで設定されます。各アドレスは、最大バイト数を使用するように設定されています。この制限に達すると、そのアドレスに送信されたメッセージは、キューに到達する前にディスク上のバッファーにページングされます。アドレスに十分な使用可能なスペースがある場合、キューは一度に 1 ページずつページング解除されます。

次の表では、AMQ 6 のメッセージページングサイズの制限と、AMQ Broker 7 の同等のプロパティーを比較しています。

設定するには	AMQ 6 で	AMQ Broker 7 で

設定するには	AMQ 6 で	AMQ Broker 7 で
ページングサイズ。	**maxPageSize** これはメッセージの数で測定され、使用可能なメッセージの数に基づいて変化します。	**<page-size-bytes>** これは、バイト単位の物理ページサイズで測定されます (メッセージではありません)。

7.2.6. デッドレターポリシーの設定プロパティー

AMQ Broker 7 は、AMQ 6 とは大きく異なる方法で、配信不能および期限切れのメッセージを処理します。デッドレターポリシーは (宛先ではなく) アドレスに適用され、(単一のデッドレターキューではなく) 個別のデッドレターと有効期限の宛先があります。デッドレターポリシーの設定は大きく異なります。

AMQ 6 の配信不能ポリシー
AMQ 6 では、期限切れまたは配信不能のメッセージは、メッセージの宛先ごとに設定された 配信不能キュー (DLQ) に送信されます。宛先の DLQ を設定するには、次のデッドレターポリシーのいずれかを使用できます。

sharedDeadLetterStrategy

宛先の配信不能メッセージは、**ActiveMQ.DLQ** と呼ばれる共有のデフォルト DLQ に送信されます。

individualDeadLetterStrategy

宛先の配信不能メッセージは、この宛先専用の DLQ に送信されます。

discardingDeadLetterStrategy

宛先の配信不能メッセージは破棄されます。

宛先のデッドレターポリシー内で、次の設定プロパティーを追加して、宛先の DLQ に送信する必要があるメッセージの種類を制御できます。

AMQ 6 設定プロパティー	説明
processNotPersistent	非持続メッセージを宛先の DLQ に送信するかどうか。デフォルトは **false** です。
processExpired	期限切れのメッセージを宛先の DLQ に送信するかどうか。デフォルトは **true** です。
有効期限	宛先の DLQ に送信されるメッセージに有効期限を適用するかどうか。デフォルトは 0 です。

AMQ 7 の配信不能ポリシー
AMQ Broker 7 では、配信不能メッセージは該当する 配信不能アドレス に送信され、期限切れのメッセージは該当する 期限切れアドレス に送信されます。

broker.xml 設定ファイルでは、デフォルトのアドレス設定でデッドレターアドレスと期限切れアドレスが指定されています。配信不能および期限切れのメッセージは、次の設定で指定された宛先に配信されます。

```
...
<address-settings>
   <address-setting match="#">
      <dead-letter-address>DLQ</dead-letter-address>
      <expiry-address>ExpiryQueue</expiry-address>
      ...
   </address-setting>
   ...
</address-settings>
...
```

デフォルトでは、配信不能アドレスと有効期限アドレスは、**<addresses>** セクションで定義されている **DLQ** と **ExpiryQueue** の宛先を指定します。

```
...
<addresses>
   <address name="DLQ">
      <anycast>
         <queue name="DLQ" />
      </anycast>
   </address>
   <address name="ExpiryQueue">
      <anycast>
         <queue name="ExpiryQueue" />
      </anycast>
   </address>
...
</addresses>
...
```

アドレスのデフォルト以外の配信不能ポリシーを設定するには、アドレスの **<address-setting>** に **<dead-letter-address>** と **<expiry-address>** を追加し、使用する DLQ と有効期限キューを指定します。

AMQ 6 とは異なり、AMQ Broker 7 では、DLQ に送信されるメッセージに有効期限を設定できません。さらに、永続メッセージと非永続メッセージの両方が、アドレスの **<dead-letter-address>** で指定された DLQ に送信されます。

第8章 メッセージの永続性およびページング

AMQ Broker 7 は、メッセージジャーナルまたは JDBC ストアのいずれかを介して永続性を提供します。ブローカーがメッセージを保存してディスクにページングする方法は AMQ 6 とは異なり、メッセージの永続性を設定するために使用する設定プロパティーが変更されています。

8.1. メッセージの永続性の変更

AMQ Broker 7 は、AMQ 6 とは異なるタイプのメッセージジャーナルを使用し、ジャーナルインデックスを使用しません。

AMQ 6 はメッセージストアに KahaDB を使用し、ジャーナル内の各メッセージの位置を追跡するメッセージジャーナルインデックスを維持していました。このインデックスにより、ブローカはページングされたメッセージをジャーナルからバッチでプルし、キャッシュに配置できました。

デフォルトでは、AMQ Broker 7 は、ブローカーがメッセージをディスパッチできるメモリー内メッセージジャーナルを使用します。したがって、AMQ Broker 7 はメッセージジャーナルインデックスを使用しません。ブローカインスタンスがメモリー不足になると、メッセージはブローカに到着したときにページングされますが、キューに入れられる前です。これらのメッセージページファイルは、到着した順序でディスクに保存されます。次に、ブローカーでメモリーが解放されると、メッセージはページファイルからブローカーのジャーナルに移動されます。ジャーナルは順次読み取られるため、メッセージのインデックスをジャーナルに保持する必要はありません。

さらに、AMQ Broker 7 は、AMQ 6 では利用できなかった別の JDBC ベースのメッセージジャーナルオプションも提供します。

AMQ Broker 7 メッセージジャーナルは、次の共有ファイルシステムをサポートしています。

- NFSv4
- GFS2

関連情報

- デフォルトのメモリー内メッセージジャーナルの詳細については、**AMQ Broker** の設定の ジャーナルベースの持続性について を参照してください。
- 新しい JDBC ベースの永続性オプションの詳細については、**AMQ Broker** の設定の JDBC 永続性の設定 を参照してください。

8.2. メッセージ持続性の設定方法

BROKER_INSTANCE_DIR/etc/broker.xml 設定ファイルを使用して、ブローカーインスタンスのメッセージジャーナルを設定します。

broker.xml 設定ファイルには、次のデフォルトのメッセージジャーナル設定プロパティーが含まれています。

```
<core>
   <name>0.0.0.0</name>
   <persistence-enabled>true</persistence-enabled>
   <journal-type>ASYNCIO</journal-type>
```

```
    <paging-directory>./data/paging</paging-directory>
    <bindings-directory>./data/bindings</bindings-directory>
    <journal-directory>./data/journal</journal-directory>
    <large-messages-directory>./data/large-messages</large-messages-directory>
    <journal-datasync>true</journal-datasync>
    <journal-min-files>2</journal-min-files>
    <journal-pool-files>-1</journal-pool-files>
    <journal-buffer-timeout>744000</journal-buffer-timeout>
    <disk-scan-period>5000</disk-scan-period>
    <max-disk-usage>90</max-disk-usage>
    <global-max-size>104857600</global-max-size>
    ...
</core>
```

メッセージジャーナルを設定するために、任意のジャーナル設定プロパティーの既定値を変更できます。設定プロパティーを追加することもできます。

8.3. メッセージ持続性設定プロパティーの変更

AMQ 6 と AMQ Broker 7 は両方とも、ブローカーがメッセージを永続化する方法を制御するための多くの設定プロパティーを提供します。このセクションでは、AMQ 6 KahaDB ジャーナルの設定プロパティーを、AMQ Broker 7 のメモリー内メッセージジャーナルの同等のプロパティーと比較します。

メモリー内メッセージジャーナルの各メッセージパーシスタンス設定プロパティーの詳細については、以下を参照してください。

- **AMQ Broker** の設定の バインディングジャーナル
- **AMQ Broker**の設定の メッセージングジャーナル設定要素

8.3.1. ジャーナルのサイズと管理

次の表では、AMQ 6 のジャーナルサイズと管理設定のプロパティーを、AMQ Broker 7 の同等のプロパティーと比較しています。

設定するには	AMQ 6 で	AMQ Broker 7 で
使用されなくなったデータログをクリーンアップする間隔	**cleanupInterval** デフォルトは 30000 ミリ秒です。	該当なし。AMQ Broker 7 では、プールサイズを超えるジャーナルファイルは使用されなくなりました。

設定するには	AMQ 6 で	AMQ Broker 7 で
圧縮がトリガーされる前に、他のファイルをクリーンアップせずに完了する必要があるメッセージストア GC サイクルの数	**compactAcksAfterNoGC**	該当なし。AMQ Broker 7 では、圧縮は特定のレコードタイプとは関係ありません。
メッセージストアがまだ拡大しているときに圧縮を実行するか、それとも拡大が停止したときにのみ圧縮を実行するか	**compactAcksIgnoresStoreGrowth** デフォルトは **false** です。	該当なし。
ブローカが圧縮する前にブローカに保存できるジャーナルファイルの最小数	該当なし。	**<journal-compact-min-files>** デフォルトは 10 です。この値を 0 に設定すると、圧縮は無効になります。
圧縮が開始される前に到達するしきい値	該当なし。	**<journal-compact-percentage>** デフォルト値は 30% です。このパーセンテージ未満がライブデータと見なされると、圧縮が開始されます。
メッセージストアのデータファイルを保持する最上位フォルダーへのパス	**directory**	AMQ Broker 7 には、ジャーナルの種類ごとに個別のディレクトリーがあります。 ● **<journal-directory>** - デフォルトは **/data/journal** です。 ● **<bindings-directory>** - デフォルトは **/data/bindings** です。 ● **<paging-directory>** - デフォルトは **/data/paging** です。 ● **<large-message-directory>** - デフォルトは **/data/large-messages** です。
bindings ディレクトリーが存在しない場合に自動的に作成するかどうか	該当なし。	**<create-bindings-dir>** デフォルトは **true** です。
ジャーナルディレクトリーがまだ存在しない場合に自動的に作成するかどうか	該当なし。	**<create-journal-dir>** デフォルトは **true** です。

設定するには	AMQ 6 で	AMQ Broker 7 で
メッセージストアが、メッセージ確認のみを含む古いジャーナルログファイルを定期的に圧縮するかどうか	**enableAckCompaction**	該当なし。
データログファイルの最大サイズ	**journalMaxFileLength** デフォルトは 32 MB です。	**<journal-file-size>** デフォルトは 10485760 バイト (10 MiB) です。
新しいジャーナルファイルが必要な場合に、ブローカーがジャーナルファイルを事前に割り当てるために使用するポリシー	**preallocationStrategy** デフォルトは **sparse_file** です。	該当なし。デフォルトでは、事前に割り当てられたジャーナルファイルは通常ゼロで埋められますが、ファイルシステムによって異なる場合があります。
ブローカーがジャーナルファイルの事前割り当てに使用するポリシー	**preallocationScope** デフォルトは **whole_journal** です。	AMQ Broker 7 は、ブローカーインスタンスの起動時に **<journal-min-files>** で指定されたジャーナルファイルを自動的に事前に割り当てます。
ジャーナルのタイプ (NIO または AIO)	該当なし。	**<journal-type>** **NIO** (Java NIO ジャーナル) または **ASYNCIO** (Linux 非同期 I/O ジャーナル) のいずれかを選択できます。
ジャーナルが保持するファイルの最小数	該当なし。	**<journal-min-files>**
ファイルを再利用するときにブローカーが保持する必要があるジャーナルファイルの数	該当なし。	**<journal-pool-files>** デフォルトは -1 です。これは、ブローカインスタンスが一度作成されたジャーナル上のファイルを決して削除しないことを意味します。

8.3.2. 書き込み境界

次の表では、AMQ 6 の書き込み境界設定プロパティーと AMQ Broker 7 の同等のプロパティーを比較しています。

設定するには	AMQ 6 で	AMQ Broker 7 で
メタデータキャッシュをディスクに書き込む間隔	**checkpointInterval** デフォルトは 5000 ミリ秒です。	該当なし。
メッセージストアがメッセージストレージと同時にキューメッセージをクライアントにディスパッチするかどうか	**concurrentStoreAndDispatchQueues** デフォルトは **true** です。	該当なし。
メッセージストアがメッセージストレージと同時にトピックメッセージを関心のあるクライアントにディスパッチする必要があるかどうか	**concurrentStoreAndDispatchTopics** デフォルトは **false** です。	該当なし。
各非トランザクションジャーナル書き込み後にディスク同期を実行するかどうか	**enableJournalDiskSyncs** デフォルトは **true** です。	**<journal-sync-transactional>** トランザクション境界 (コミット、準備、およびロールバック) に達するたびに、トランザクションデータをディスクにフラッシュします。デフォルトは **true** です。 **<journal-sync-nontransactional>** 非トランザクションメッセージデータをディスクにフラッシュします (送信と確認)。デフォルトは **true** です。
ジャーナルバッファー全体をいつフラッシュするか	該当なし。	**<journal-buffer-timeout>** NIO のデフォルトは 3,333,333 ナノ秒で、AIO のデフォルトは 500,000 ナノ秒です。
ジャーナルディスク書き込みの間にバッファーするデータの量	**journalMaxWriteBatchSize** デフォルトは 4000 バイトです。	該当なし。
ジャーナルの書き込み要求をバッファーリングするために使用されるタスクキューのサイズ	**maxAsyncJobs** デフォルト値は 10000 です。	**<journal-max-io>** このプロパティーは、任意の時点で I/O キューに入れることができる書き込み要求の最大数を制御します。NIO のデフォルト値は 1 で、ASYNCIO のデフォルト値は 500 です。

設定するには	AMQ 6 で	AMQ Broker 7 で
ジャーナル書き込みで **fdatasync** を使用するかどうか	該当なし。	**<journal-datasync>** デフォルトは **true** です。

8.3.3. インデックス設定

AMQ 6 には、ジャーナルインデックスを設定するための多くの設定プロパティーがあります。AMQ Broker 7 はジャーナルインデックスを使用しないため、ブローカーインスタンスに対してこれらのプロパティーを設定する必要はありません。

8.3.4. ジャーナルアーカイブ

AMQ 6 には、アーカイブするファイルとアーカイブの保存場所を制御するための設定プロパティーがいくつかあります。ただし、AMQ Broker 7 では、古いジャーナルファイルが不要になると、ブローカーはそれらをアーカイブする代わりに再利用します。したがって、ブローカーインスタンスのジャーナルアーカイブプロパティーを設定する必要はありません。

8.3.5. ジャーナルの回復

AMQ 6 には、ブローカーが破損したジャーナルファイルをチェックする方法と、欠落しているジャーナルファイルが検出された場合の対処方法を制御するための設定プロパティーがいくつかあります。

ただし、AMQ Broker 7 では、ブローカーインスタンスのジャーナル回復プロパティーを設定する必要はありません。AMQ Broker 7 ではジャーナルファイルの形式が異なるため、ジャーナル内の破損したエントリーによってジャーナルファイル全体が破損するのを防ぐことができます。ジャーナルが部分的に破損している場合でも、ブローカーは破損していないエントリーからデータを抽出できるはずです。

第9章 ブローカークラスター

ブローカーを接続してクラスターを形成できます。ブローカークラスターを使用すると、メッセージ処理の負荷を分散し、クライアント接続のバランスを取ることができます。また、クライアントが接続できるブローカーの数を増やすことで、フォールトトレランスを提供します。

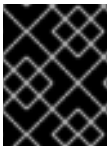
9.1. ブローカークラスターリングの変更

AMQ Broker 7 では、ブローカーネットワークはブローカークラスターと呼ばれます。クラスター内のブローカーは、クラスター接続 (**connector** 要素を参照) によって接続されます。クラスターのメンバーは、相互に動的に (UDP または JGroups を使用して)、または静的に (クラスターメンバーのリストを手動で指定して) 検出するように設定できます。

クラスター設定は、高可用性 (HA) の必須前提条件です。クラスターが 1 つの Live Broker のみで設定されている場合でも、HA を設定する前にクラスターを設定する必要があります。

対称クラスターとチェーンクラスターが最も一般的ですが、さまざまなトポロジーでブローカークラスターを設定できます。トポロジーに関係なく、メッセージを失うことなくクラスターをスケールアップおよびスケールダウンできます (クラスター内の別のブローカーにメッセージを送信するようにブローカーを設定している場合)。

ブローカークラスターは、AMQ 6 のブローカーネットワークとは異なる方法でメッセージを配布 (および再配布) します。AMQ 6 では、メッセージは常に特定のキューに到着し、コンシューマーの関心に基づいてあるブローカーから別のブローカーにプルされました。AMQ Broker 7 では、キュー定義とコンシューマーがクラスター全体で共有され、メッセージはブローカーで受信されるとクラスター全体にルーティングされます。



重要

同じクラスター内で AMQ 6 ブローカーと AMQ Broker 7 ブローカーを組み合わせようとしないでください。

9.2. ブローカークラスターの設定方法

クラスターのメンバーごとに ブローカーインスタンスを作成 し、各ブローカーインスタンスにクラスター設定を追加して、ブローカークラスターを設定します。

クラスター設定は、次のもので設定されます。

検出グループ

動的検出で使用するために、検出グループは、ブローカーインスタンスがクラスター内の他のメンバーを検出する方法を定義します。Discovery は、UDP または JGroups のいずれかを使用できます。

ブロードキャストグループ

動的検出で使用するために、ブロードキャストグループは、ブローカーインスタンスがクラスター関連情報をクラスター内の他のメンバーに送信する方法を定義します。ブロードキャストは UDP または JGroups のいずれかを使用できますが、対応する検出グループと一致する必要があります。

クラスター接続

ブローカーインスタンスがクラスターの他のメンバーに接続する方法。検出グループまたはクラスターメンバーの静的リストを指定できます。メッセージの再配布と最大ホッププロパティーを指定することもできます。

9.2.1. ブローカークラスターの作成

この手順では、静的検出を使用して基本的な 2 ブローカークラスターを作成する方法を示します。

手順

1. **artemis create** コマンドを使用して、最初のブローカーインスタンスを作成します。
この例では、**broker1** という名前の新しいブローカーインスタンスを作成します。

```
$ sudo INSTALL_DIR/bin/artemis create broker1 --user user --password pass --role amq
```

2. クラスターの 2 番目のメンバーの 2 番目のブローカーインスタンスを作成します。
追加のブローカーインスタンスごとに、**--port-offset** パラメーターを使用して、以前のブローカーインスタンスとのポートの衝突を回避する必要があります。

   この例では、**broker2** という 2 番目のブローカーインスタンスを作成します。

```
$ sudo INSTALL_DIR/bin/artemis create broker2 --port-offset 100 --user user --password pass --role amq
```

3. 最初のブローカーインスタンスの場合、**BROKER_INSTANCE_DIR/etc/broker.xml** 設定ファイルを開き、クラスター設定を追加します。
静的検出の場合、コネクターと静的クラスター接続を追加する必要があります。この例では、**broker1** が **broker2** に接続するように設定します。

```
<!-- Connectors -->
<connectors>
   <connector name="netty-connector">tcp://localhost:61616</connector>
   <!-- connector to broker2 -->
   <connector name="broker2-connector">tcp://localhost:61617</connector>
</connectors>

<!-- Clustering configuration -->
<cluster-connections>
   <cluster-connection name="my-cluster">
      <connector-ref>netty-connector</connector-ref>
      <retry-interval>500</retry-interval>
      <use-duplicate-detection>true</use-duplicate-detection>
      <message-load-balancing>STRICT</message-load-balancing>
      <max-hops>1</max-hops>
      <static-connectors>
         <connector-ref>broker2-connector</connector-ref>
      </static-connectors>
   </cluster-connection>
</cluster-connections>
```

4. 2 番目のブローカーインスタンスについては、**BROKER_INSTANCE_DIR/etc/broker.xml** 設定ファイルを開き、クラスター設定を追加します。
この例では、**broker1** に接続するように **broker2** を設定します。

```
<!-- Connectors -->
<connectors>
   <connector name="netty-connector">tcp://localhost:61617</connector>
   <!-- connector to broker1 -->
```

```
    <connector name="broker1-connector">tcp://localhost:61616</connector>
</connectors>

<!-- Clustering configuration -->
<cluster-connections>
    <cluster-connection name="my-cluster">
        <connector-ref>netty-connector</connector-ref>
        <retry-interval>500</retry-interval>
        <use-duplicate-detection>true</use-duplicate-detection>
        <message-load-balancing>STRICT</message-load-balancing>
        <max-hops>1</max-hops>
        <static-connectors>
            <connector-ref>broker1-connector</connector-ref>
        </static-connectors>
    </cluster-connection>
</cluster-connections>
```

関連情報

- ブローカークラスターの作成と、メッセージの再配布とクライアントの負荷分散の設定の詳細は、**AMQ Broker** の設定の ブローカークラスターの設定 を参照してください。

9.2.2. 追加のブローカークラストポロジー

ブローカークラスターは、さまざまなトポロジーで接続できます。AMQ Broker 7 では、対称クラスターとチェーンクラスターが最も一般的です。

例: 対称クラスター

フルメッシュトポロジーでは、各ブローカーはクラスター内の他のすべてのブローカーに接続されます。つまり、クラスター内のすべてのブローカーは他のすべてのブローカーから複数のホップを削除できません。

この例では、動的検出を使用して、クラスター内のブローカーが相互に検出できるようにします。**max-hops** を **1** に設定すると、各ブローカーは他のすべてのブローカーに接続します。

```
<!-- Clustering configuration -->
<broadcast-groups>
    <broadcast-group name="my-broadcast-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <broadcast-period>100</broadcast-period>
        <connector-ref>netty-connector</connector-ref>
    </broadcast-group>
</broadcast-groups>

<discovery-groups>
    <discovery-group name="my-discovery-group">
        <group-address>${udp-address:231.7.7.7}</group-address>
        <group-port>9876</group-port>
        <refresh-timeout>10000</refresh-timeout>
    </discovery-group>
</discovery-groups>

<cluster-connections>
```

```
    <cluster-connection name="my-cluster">
      <connector-ref>netty-connector</connector-ref>
      <retry-interval>500</retry-interval>
      <use-duplicate-detection>true</use-duplicate-detection>
      <message-load-balancing>ON_DEMAND</message-load-balancing>
      <max-hops>1</max-hops>
      <discovery-group-ref discovery-group-name="my-discovery-group"/>
    </cluster-connection>
</cluster-connections>
```

例: チェーンクラスター

チェーンクラスターでは、ブローカーは両端にブローカーがあり、他のすべてのブローカーがチェーン内の前のブローカーと次のブローカーに接続する直線的なチェーンを形成します (たとえば、A→B→C)。

この例では、静的検出を使用して、3 つのブローカーをチェーンクラスターに接続します。各ブローカーはチェーン内の次のブローカーに接続し、**max-hops** は **2** に設定されて、メッセージがチェーン全体を流れるようにします。

最初のブローカーは次のように設定されています。

```
<connectors>
  <connector name="netty-connector">tcp://localhost:61616</connector>
  <!-- connector to broker2 -->
  <connector name="broker2-connector">tcp://localhost:61716</connector>
</connectors>


<cluster-connections>
  <cluster-connection name="my-cluster">
    <address>jms</address>
    <connector-ref>netty-connector</connector-ref>
    <retry-interval>500</retry-interval>
    <use-duplicate-detection>true</use-duplicate-detection>
    <message-load-balancing>STRICT</message-load-balancing>
    <max-hops>2</max-hops>
    <static-connectors allow-direct-connections-only="true">
      <connector-ref>broker2-connector</connector-ref>
    </static-connectors>
  </cluster-connection>
</cluster-connections>
```

2 番目のブローカーは次のように設定されます。

```
<connectors>
  <connector name="netty-connector">tcp://localhost:61716</connector>
  <!-- connector to broker3 -->
  <connector name="broker3-connector">tcp://localhost:61816</connector>
</connectors>


<cluster-connections>
  <cluster-connection name="my-cluster">
    <address>jms</address>
    <connector-ref>netty-connector</connector-ref>
```

```
      <retry-interval>500</retry-interval>
      <use-duplicate-detection>true</use-duplicate-detection>
      <message-load-balancing>STRICT</message-load-balancing>
      <max-hops>1</max-hops>
      <static-connectors allow-direct-connections-only="true">
        <connector-ref>broker3-connector</connector-ref>
      </static-connectors>
    </cluster-connection>
  </cluster-connections>
```

最後に、3 番目のブローカーは次のように設定されます。

```
<connectors>
  <connector name="netty-connector">tcp://localhost:61816</connector>
</connectors>

<cluster-connections>
  <cluster-connection name="my-cluster">
    <address>jms</address>
    <connector-ref>netty-connector</connector-ref>
    <retry-interval>500</retry-interval>
    <use-duplicate-detection>true</use-duplicate-detection>
    <message-load-balancing>STRICT</message-load-balancing>
    <max-hops>0</max-hops>
  </cluster-connection>
</cluster-connections>
```

9.3. ブローカークラスターの設定プロパティー

次の表では、AMQ 6 のブローカーネットワーク設定プロパティーと AMQ Broker 7 の同等の **cluster-connection** プロパティーを比較しています。

設定するには	AMQ 6 で	AMQ Broker 7 で
除外された宛先	**excludedDestinations**	該当なし。
メッセージがクラスターを通過できるホップ数	**networkTTL** デフォルトは **1** です。これは、メッセージが近隣のブローカーに 1 回だけホップできることを意味します。	**<max-hops>** このブローカーインスタンスを設定して、間接的に接続されている可能性のあるブローカーにメッセージを負荷分散します。他のブローカーはチェーンの仲介者です。デフォルト値は **1** です。つまり、メッセージはこのブローカーインスタンスに直接接続された他のブローカーにのみ分散されます。

設定するには	AMQ 6 で	AMQ Broker 7 で
コンシューマーがいないときにメッセージを再生する	**replayWhenNoConsumers**	該当なし。ただし、**<redistribution-delay>** を設定して、メッセージが初めて到着したかのように再配信されるまでの時間 (ミリ秒単位) を定義することができます。
クラスター内の一時的な宛先にアドバイザリーメッセージをブロードキャストするかどうか	**bridgeTempDestinations** デフォルトは **true** です。このプロパティーは、通常、要求応答メッセージ用に作成された一時的な宛先に使用されていました。これにより、これらのメッセージのコンシューマーがネットワーク内の別のブローカーに接続され、**JMSReplyTo** ヘッダーで指定された一時的な宛先に応答を送信できるようになります。	該当なし。AMQ Broker 7 では、一時的な宛先がクラスター化されることはありません。
このブローカーをリモートブローカーで認証するために使用する認証情報	**userName password**	**<cluster-user> <cluster-password>**
コネクターのルート優先度を設定する	**decreaseNetworkConsumerPriority** デフォルトは **false** です。**true** に設定すると、ローカルコンシューマーの優先度は **0** になり、ネットワークサブスクリプションの優先度は **-5** になります。さらに、ネットワークサブスクリプションの優先度は、通過するネットワークホップごとに 1 ずつ減少します。	該当なし。

設定するには	AMQ 6 で	AMQ Broker 7 で
クラスター内の他のブローカー間でメッセージを分散するかどうか、およびその方法	該当なし。	**<message-load-balancing>** これは、**OFF** (負荷分散なし)、**STRICT** (一致するキューを持つクラスター内のすべてのブローカーにメッセージを転送する)、または **ON_DEMAND** (アクティブなコンシューマーまたは一致するセレクターを持つクラスター内のブローカーにのみメッセージを転送する) に設定できます。デフォルトは **ON_DEMAND** です。
メッセージを生成および消費するためのクラスターネットワーク接続を有効にする	**duplex** デフォルトでは、ネットワークコネクターは単方向です。ただし、双方向にメッセージが流れるように二重に設定することもできます。これは通常、ハブがファイアウォールの背後にあるハブアンドスポークネットワークに使用されていました。	該当なし。クラスター接続は単方向のみです。ただし、両端から 1 つずつ、各ブローカー間にクラスター接続のペアを設定できます。ブローカークラスターのセットアップの詳細は、**AMQ Broker の設定** の ブローカークラスターのセットアップ を参照してください。

第10章 高可用性とフェイルオーバー

クラスター設定を作成したら、ブローカーインスタンスをリンクして、高可用性 (HA) ペアを形成できます。HA ペアは、クライアント要求を処理するマスターブローカーと、クライアントと通信できなくなった場合にマスターを置き換える 1 つ以上のスレーブブローカーで設定されます。

AMQ Broker 7 では、HA のためにクラスター設定が必要です。ブローカークラスターは、一連の非 HA ブローカーまたは HA ペアのいずれかで設定できます。

AMQ Broker 7 は、次の HA ポリシーを提供します。

レプリケーション

レプリケーションは、ネットワークを介してマスターブローカーとスレーブブローカーの間でデータを同期します。レプリケーションを使用すると、フェイルバックを有効にして、障害イベント後にマスターブローカーがオンラインに戻ったときにマスターブローカーに制御を戻し、クライアントがマスターブローカーにフェールバックできるようにすることができます。また、複数のマスターブローカーが 1 つ以上のスレーブブローカーを共有する HA グループを作成し、スレーブブローカーをマスターブローカーと同じ JVM に配置することもできます。

重要

7.5 以降、以前はレプリケーション HA ポリシーで使用できたネットワーク ping は非推奨の機能です。ネットワークの ping は、ネットワークの分離の問題からブローカークラスターを保護することができません。これにより、修復不能なメッセージが失われることがあります。この機能は今後のリリースで削除されます。Red Hat では、ネットワークの ping を使用する既存の AMQ Broker デプロイメントは引き続きサポートされます。ただし、Red Hat は、新しいデプロイメントでネットワーク ping を使用することは推奨しません。高可用性のためのブローカークラスターの設定と、ネットワーク分離の問題を回避するためのガイダンスについては、高可用性の実装 を参照してください。

共有ストア

共有ストアは、マスターブローカーとスレーブブローカーがメッセージングデータを共有するための場所を提供します。レプリケーションよりも次の利点があるため、共有ストアを使用することをお勧めします。

- パフォーマンス (共有ストアの方が高速)
- スプリットブレインではない問題
- クォーラムを維持するために必要なブローカーが少ない (レプリケーションには少なくとも 3 つ必要)
  レプリケーションと同様に、フェールバックを有効にして、障害イベントの後に制御をマスターブローカーに戻し、クライアントがマスターブローカーにフェールバックできるようにすることができます。マスターブローカーに複数のスレーブブローカーを設定し、スレーブブローカーを併置することができます。

HA とフェイルオーバーの詳細については、**AMQ Broker** の設定の 高可用性の実装 を参照してください。

10.1. 高可用性とフェイルオーバーの変更点

AMQ Broker 7 の高可用性は、マスターの決定方法とブローカー接続がアクティブになるタイミングに基づいて、AMQ 6 とは異なります。

AMQ Broker 7 では、マスターとスレーブのロールが固定されています。どのブローカーインスタンスがマスターであるかを指定すると、スレーブは特定の条件でのみアクティブになります。AMQ 6 では、マスターとスレーブのロールは固定されていませんでした。代わりに、HA ペアのブローカーがロックを求めて競合し、勝者がマスターになります。

AMQ Broker 7 では、HA ペアで、スレーブブローカーのアクセプターは、ブローカーが非アクティブであってもアクティブです。AMQ 6 では、スレーブブローカーのトランスポートコネクターは、ブローカーがアクティブになるまでアクティブになりませんでした。

10.2. 高可用性の設定方法

各ブローカーの **BROKER_INSTANCE_DIR/etc/broker.xml** 設定ファイルに HA ポリシー設定を追加して、HA を設定します。

例: 共有ストアとの **HA** ペア

マスターブローカーはこのように設定されています。**failover-on-shutdown** を **true** に設定すると、マスターブローカーがシャットダウンされた場合に HA ペアがスレーブブローカーにフェイルオーバーします。

```
<configuration>
  <core>
    ...
    <ha-policy>
      <shared-store>
        <master/>
          <failover-on-shutdown>true</failover-on-shutdown>
      </shared-store>
    </ha-policy>
    ...
  </core>
</configuration>
```

スレーブブローカは次のように設定されます。**failover-on-shutdown** を **true** に設定すると、現在のマスターブローカーがシャットダウンされた場合に、このスレーブブローカーがマスターになります。

```
<configuration>
  <core>
    ...
    <ha-policy>
      <shared-store>
        <slave/>
          <failover-on-shutdown>true</failover-on-shutdown>
      </shared-store>
    </ha-policy>
    ...
  </core>
</configuration>
```

関連情報

HA ポリシーの設定の詳細については、次のトピックを参照してください。

- **AMQ Broker** の設定の 高可用性の設定

改訂日時：2023-01-28 11:51:43 +1000