Red Hat AMQ 2021.Q2

Red Hat AMQ 7 への移行

Red Hat AMQ 2021.Q2 向け

Last Updated: 2021-08-26

Red Hat AMQ 2021.Q2 Red Hat AMQ 7 への移行

Red Hat AMQ 2021.Q2 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に準備できます。

1.1. 移行の前にサポートを要求するタイミング

実稼働環境を移行予定の場合には、Red Hat サポート担当者からさらにサポートおよびガイダンスを要求する必要があります。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説明します。

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)
  本ガイドでは、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方法を説明します。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 で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 機能エリアで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的な違いがあります。さらに、本リリースには新しいメッセージアドレス指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的な変更

AMQ 7 は、ブローカーへの受信ネットワーク接続の方法、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する主要なアーキテクチャー的な変更を提供します。

受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 (ブロック以外) 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受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 では、メッセージストアに KahaDB が使用されました。これは、高速で連続するメッセージの保存用のメッセージジャーナル、および必要に応じてメッセージを取得するインデックスの両方で構成されます。

AMQ 7 には、追加のみのメッセージジャーナルで構成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、「メッセージの永続化およびページング」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は 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 は Apache Karaf コンテナーに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 ブローカー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、任意のメッセージングプロトコル (または JMS の場合 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指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す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、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、メッセージをアドレスおよびルーティングメカニズムに基づいて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および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できます。マルチキャストルーティングは、パブリッシュ-サブスクライブパターンを実装します。この場合、アドレスに送信されたメッセージをアドレスのすべてのサブスクライバーが受信します。または、エニーキャストルーティングは、ポイント-ツー-ポイントパターンを実装します。この場合、単一のキューのみがアドレスにアタッチされ、そのキューをサブスクライブする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します。

関連情報

- AMQ Broker 7 の新しいアドレス指定モデルに関する詳細は、『**Configuring AMQ Broker**』の「Configuring addresses and queues」 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。その一覧は、『Release Notes for Red Hat AMQ Broker 7.8』 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以下の表記方法を使用します。

sudo コマンド
本書では、root 権限を必要とするコマンドには **sudo** が使用されています。**sudo** を使用する場合は、変更を加えるとシステム全体に影響する可能性があるため、常に注意が必要です。

sudo の使用に関する詳細は、「Managing **sudo** access」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が 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

各機能エリアで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。

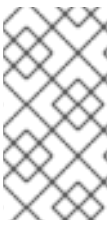
AMQ 7 の要件

- 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 のサポートされる構成は、https://access.redhat.com/articles/2791941 に説明があります。
- 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、『**Getting Started with AMQ Broker**』の「Installing AMQ Broker」 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 ブローカー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 での設定の違いを把握することで、このブローカーインスタンスを設定できます。

AMQ Broker をインストールした際に、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には、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したら、必要な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、必要に応じてブローカーを管理およびアップグレードするのが容易になります。

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. Broker 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. 以下の操作のいずれかを実施して、ブローカーインスタンスを作成します。

状況	操作
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** コマンドを使用して、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 と 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 と 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、新しい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を作成し、必要な値の入力を求めるプロンプトを出します。 `$ sudo INSTALL_DIR/bin/artemis create mybroker` ActiveMQ Artemis インスタンスの作成: /var/lib/amq7/mybroker --user: この設定では必須です。デフォルトのユーザー名 user を指定してください。 --password: この設定では必須です。デフォルトのパスワード password を指定してください。 --role: この設定では必須です。デフォルトのロール amq を指定してください。 --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、『**Getting Started with AMQ Broker**』の「Creating a broker instance」を参照してください。

2.3. BROKER 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ます。ディレクトリーの内容、および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している必要があります。

ブローカーインスタンスを作成すると、以下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

Broker 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異なる場所です。

/bin

ブローカーインスタンスの起動および停止用のシェルスクリプト。

/data

メッセージストアなどのブローカーの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。これらは、ブローカーインスタンスを設定するのにアクセスする必要のあるファイルです。

/lock

cli.lock ファイルが含まれます。

/log

Broker インスタンスのログファイル。

/tmp

一時ファイルのユーティリティーディレクトリー。

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の設定方法と、編集が必要な設定ファイルを理解する必要があります。

AMQ 6 のように、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、実際の変更 (通常は、XML 要素と属性の追加または削除) を行う必要があります。

BROKER_INSTANCE_DIR/etc 内には、編集できる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	メインの設定ファイル。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の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、メインのブローカー設定ファイルの場所を変更し、Web サーバーを設定し、セキュリティー設定を行います。

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 に移行するには、主に **broker.xml** ファイルを編集する必要があります。**broker.xml** 構造およびデフォルト設定の詳細は、『**Configuring AMQ Broker**』の「Understanding the default broker configuration」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

既存のクライアントが作成したブローカーインスタンスに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て、テストメッセージを送信する必要があります。

手順

1. 以下のコマンドのいずれかを使用して、ブローカーインスタンスを起動します。

機能	使用するコマンド
フォアグラウンドでブローカーを起動する。	$ sudo **BROKER_INSTANCE_DIR**/bin/artemis run
ブローカーをサービスとして起動する	$ 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 ブローカーと同じポートでブローカーインスタンスで起動します。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はずです。

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、ローカルホストでホストされる 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、デフォルトのアクセプターをリッスンします。

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

   ブローカーインスタンスの作成時にポート番号をオフセットした場合は (**--port-offset** を使用して)、ブローカー URL に正しいポート番号を使用するようにしてください。たとえば、ポートオフセットを 100 に設定した場合は、**--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使用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
```

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停止します。

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

第3章 受信接続の許可

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 とは機能的に異なり、異なる方法で設定されます。

3.1. 受信ネットワーク接続の変更

AMQ 6 および 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 と呼ばれます。

AMQ 6 では、トランスポート層の複数の実装 (TCP、NIO など) が提供されました。したがって、クライアント接続ポイントがブロッキングトランスポートまたはノン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デフォルトで非ブロッキングの Netty のみを使用します。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、同じサーバーか物理的にリモートかに関係なく、クライアントとブローカーが異なる仮想マシンにある場合に使用されます。
Netty はデフォルトで非ブロッキング (Java NIO) を使用します。したがって、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は非ブロッキングです。また、WebSocket の組み込みサポートもあります。

In-VM

in-VM 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。

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もありました。AMQ Broker 7 では、低レベルのトランスポート (TCP または in-VM のいずれか) は、クライアントによって使用されるメッセージングプロトコル (AMQP、MQTT 等) とは独立しています。つまり、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ます。実際、使用できるプロトコルを明示的に制限しない限り、アクセプターはサポートされ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受信クライアント接続を受け入れるようアクセプターを設定します。

broker.xml 設定ファイルには、**<acceptors>** セクションに以下のデフォルトアクセプターが含まれます。

```
<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) サポートされるすべてのメッセージングプロトコルについて受信クライアント接続を受け入れるデフォルトのアクセプター。

ブローカーインスタンスの受信クライアント接続を設定するには、デフォルトのアクセプターのいずれかの設定プロパティーを変更したり、新しいアクセプターを追加したりできます。以下の例は、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の詳細は、『**Configuring AMQ Broker**』の「Default acceptor settings」を参照してください。
- アクセプター設定のステップごとの詳細は、『**Configuring AMQ Broker**』の「Network Connections: Acceptors and Connectors」を参照してください。
- アクセプターの設定に使用できるすべてのプロパティーの説明は、『**Configuring AMQ Broker**』の「Acceptor and Connector Configuration Parameters」を参照してください。

第4章 ユーザー認証

ユーザー認証では、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、ユーザーのアイデンティティーを検証できます。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、設定構文にはいくつかの違いがあります。相違点を理解すると、ブローカーインスタンスへのユーザーアクセスの設定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 の両方の認証は、Java Authentication and Authorization Service (JAAS) をベースとしたプラグ可能なログインモジュールによって提供されます。ただし、AMQ 6 の グループ は AMQ Broker 7 では ロール と呼ばれます。

さらに、ログインモジュールの名前と場所が AMQ Broker 7 で変更になりました。

ログインモジュール	AMQ 6 での場所	AMQ Broker 7 での場所
ユーザー	**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構文も異なります。

AMQ 6 の場合

特権のないユーザーを追加して、**users.properties** ファイルでパスワードおよびセキュリティーロールを割り当てることができました。

```
USER=PASSWORD,ROLE
```

AMQ Broker 7 の場合

ユーザーとロールは、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、以下のいずれかの方法で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しかブローカーインスタンスアクセスできません。 詳細は、**『Configuring AMQ Broker』**の「Configuring user and password authentication based on properties files」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、**『Configuring AMQ Broker』**の「Configuring certificate-based 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「Using LDAP for authentication and authorization」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、任意のメッセージングプロトコルで機能する標準メッセージングパターンを定義できる、新しい柔軟なアドレス指定モデルが導入されました。そのため、キューや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. アドレス指定の変更

AMQ 6 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 の概念を実装しました。

例: AMQ 6 のデフォルトのキューおよびトピック設定

```
<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、キューとトピックのような動作を実現します。アドレス 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が配信される方法を定義します。ルーティングタイプには 2 種類あります。エニーキャスト の場合、マッチ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 の場合、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。

キューをアドレスおよび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 (トピック的) などの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 のポイントツーポイントアドレス設定

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は、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 のパブリッシュ-サブスクライブアドレス設定

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に関する詳細は、『**Configuring AMQ Broker**』の「Configuring addresses and queues」を参照してください。

5.2. アドレス指定の設定方法

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およびキューを設定します。

broker.xml 設定ファイルには、**<addresses>** セクションに以下のデフォルトアドレス指定設定が含まれます。Dead Letter Queue (**DLQ**) および Expiry Queue (**ExpiryQueue**) には、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>
```

以下の方法のいずれかを使用して、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およびキューを定義します。アドレスは以下の方法で設定できます。 ● 『**Configuring AMQ Broker**』の「Configuring basic point-to-point messaging」 ● 『**Configuring AMQ Broker**』の「Configuring point-to-point messaging for multiple queues」 ● 『**Configuring AMQ Broker**』の「Configuring addresses for publish-subscribe messaging」 ● 『**Configuring AMQ Broker**』の「Configuring an address for both point-to-point and publish-subscribe messaging」 ● 『**Configuring AMQ Broker**』の「Configuring subscription queues」

メソッド	説明
アドレスを自動的に作成するブローカーの設定	自動作成されるアドレスのプレフィックスおよびルーティングタイプを指定します。ブローカーが接頭辞とマッチ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。また、アドレスのすべてのキューが削除されると自動的にアドレスが削除されるように指定することや、アドレスのキューにコンシューマーやメッセージがない場合にキューが自動的に削除されるように指定することもできます。 詳細は、**『Configuring AMQ Broker』**の「Creating and deleting addresses and queues automatically」を参照してください。

第6章 セキュリティー

AMQ Broker 7 は、トランスポート層セキュリティーを提供して受信ネットワーク接続をセキュアにし、承認によりそれぞれのアドレスに基づいてキューへのアクセスをセキュアにします。これら両方の領域で、セキュリティーモデルは AMQ 6 と非常に似ています。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 を使用すると、SSL/TLS を使用して受信ネットワーク接続をセキュア化できます。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、キーストアとトラストストアを定義する SSL コンテキストを作成し、セキュリティー保護する各トランスポートコネクターに SSL 属性を追加して、トランスポート層セキュリティーが設定されていました。

AMQ Broker 7 では、トランスポート層は SSL をネイティブで使用する Netty をベースにしています。そのため、トランスポート層セキュリティーを設定するには、セキュリティー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、以下の設定では、OpenWire クライアントからのセキュアな接続を受け入れます。

AMQ 6 の場合

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、OpenWire クライアントからのセキュアな接続を受け入れるようにトランスポートコネクターを作成します。

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

AMQ Broker 7 の場合

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

```
<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 TLS のいずれかを設定できます。以下の表では、その方法について説明しています。

メソッド	説明

メソッド	説明
一方向 TLS	ブローカーのみ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に Java KeyStore が必要です。 詳細は、**『Configuring AMQ Broker』**の「Securing connections」を参照してください。
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、**『Configuring AMQ Broker』**の「Securing connections」を参照してください。

注記

AMQ Broker 7 用に既存のキーストアおよび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リストは、**『Configuring AMQ Broker』**の「Netty TLS Parameters」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 と似ていますが、パーミッションとワイルドカード構文が異なり、さらに承認の設定方法が異なります。

6.2.1. 承認の変更

AMQ Broker 7 は、AMQ 6 とは異なるパーミッションのセット、および若干異なるワイルドカード構文を使用します。

AMQ 6 および AMQ Broker 7 で適用可能な異なるタイプのパーミッションを以下の表に示します。

AMQ 6 のパーミッション	対応する AMQ Broker 7 のパーミッション
write	**send**
read	**consume** **browse**

AMQ 6 のパーミッション	対応する AMQ Broker 7 のパーミッション
admin	**createAddress** **deleteAddress** **createNonDurableQueue** **deleteNonDurableQueue** **createDurableQueue** **deleteDurableQueue** **manage**

AMQ Broker 7 のパーミッションに関する詳細は、『**Configuring AMQ Broker**』の「Configuring user- and role-based authorization」を参照してください。

マッチするアドレス用のワイルドカード構文も、AMQ Broker 7 では異なります。

機能	AMQ 6 の場合	AMQ Broker 7 の場合
パス内の単語を区切る	**.**	**.**
1 つの単語にマッチする	*****	*****
再帰的に任意の単語にマッチする	**>**	**#**

6.2.2. 承認の設定方法

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、以下のデフォルトのセキュリティー設定が含まれています。これは、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ロールのすべてのアドレスおよび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. セキュリティーパーミッションのセットが適用されるアドレスまたはアドレスプレフィックス。パーミッションは、アドレスにマッチするキューのセットに適用されます。この例では、**#** ワイルドカードはすべてのアドレスとマッチします。

2. ロールに付与されるパーミッション。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パーミッションが付与されます。

キューにマッチするアドレスを指定し、各パーミッションタイプを付与するロールを指定して、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

- 『**Configuring AMQ Broker**』の「Configuring user- and role-based authorization」

第7章 リソースの制限およびポリシー

リソース制限および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に関する重要な側面を制御できます。AMQ Broker 7 でこれらのリソース制限およびポリシーを設定するプロセスは AMQ 6 とは異なり、設定プロパティーの多くが変更になりました。

7.1. リソース制限およびポリシーの設定方法

AMQ 6 では、リソースの制限および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の セット のリソース制限および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定義されたリソース制限およびポリシーがメッセージに適用されます。

AMQ Broker 7 でリソース制限および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、以下のデフォルトアドレス設定が含まれます。

```
<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およびポリシーは、**activemq.management#** にマッチするアドレスを持つ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マッチするため、定義されたリソース制限およびポリシーはすべてのメッセージに適用されます。

リソース制限およびポリシーを設定するには、アドレスまたはアドレス セット を指定します (**<address-setting>** 使用して)、リソース制限およびポリシープロパティーをこれに追加します。これらのプロパティーは、指定したアドレス (またはアドレスセット) に送信された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たアドレスのセットのマッチに関する詳細は、『**Configuring AMQ Broker**』の「AMQ Broker wildcard syntax」 を参照してください。

7.2. リソース制限およびポリシー設定プロパティー

AMQ 6 と同様に、AMQ Broker 7 では、リソース制限およびポリシーを追加して、メッセージの配信方法およびタイミング、配信試行の数、およびメッセージが期限切れになるタイミングなど、ブローカーが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および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は、『**Configuring AMQ Broker**』の「Address Setting Configuration Elements」 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

以下の表は、AMQ 6 のキュー管理設定プロパティーを AMQ Broker 7 の同等のプロパティーと比較します。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
メモリー制限	**memoryLimit** **宛先** のメモリー制限を設定します。デフォルトは **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** (無制限) です。
キュー内の優先度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順位が付けられるため、コンシューマー上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の優先度順に自動的に順序付けられます。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
ブローカーが期限切れの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。
ブローカー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。	同等のものはありません。ただし、自動作成されるキューの場合、最後のコンシューマーが切り離された時点で自動的にキューが削除されるように設定できます。詳細は、**『Configuring AMQ Broker』**の「Configuring automatic creation and deletion of addresses and queues」を参照してください。
非アクティブによるタイムアウト	**inactiveTimeoutBeforeGC** デフォルトは 60 秒です。	同等のものはありません。ただし、自動作成されるキューの場合、最後のコンシューマーが切り離された時点で自動的にキューが削除されるように設定できます。詳細は、**『Configuring AMQ Broker』**の「Creating and Deleting Queues and Addresses Automatically」を参照してください。
キューからディスパッチする際に、ブローカーが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。	アドレスまたはキューには設定できません。ただし、メッセージが到達する受信接続から制御することができます。アクセプターまたはコネクターで **directDeliver** プロパティーを使用して、到達した同じスレッドにメッセージを配信するかどうかを制御します。詳細は、**『Configuring AMQ Broker』**の「Acceptor and Connector Configuration Parameters」を参照してください。

7.2.2. プロデューサーポリシー設定プロパティー

以下の表は、AMQ 6 のプロデューサーポリシー設定プロパティーを AMQ Broker 7 の同等のプロパティーと比較します。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
プロデューサーフロー制御	**producerFlowControl** プロデューサーにスロットリングを適用するようにブローカーを設定します。プロデューサーの確認応答を保留するか、**javax.jms.ResourceAllocationException** 例外を発生させてローカルリソース (メモリーやストレージなど) を使い果たした際にクライアントに戻すことにより、スロットリングを行います。デフォルトは **true** です。	アドレスの場合、**<max-size-bytes>** をプロデューサーにスロットリングを適用するサイズに設定し、**<address-full-policy>** を **BLOCK** に設定します。 これらの 2 つのプロパティーを設定すると、既存の AMQ 6 OpenWire プロデューサーにもスロットリングが適用されます。
プロデューサーが一度に要求できるクレジットの量	同等のものはありません。	**<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「インフライト」で実行できるバイト数を制限し、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設定プロパティー

以下の表は、AMQ 6 のサーバー側宛先ポリシー設定プロパティーを AMQ Broker 7 の同等のプロパティーと比較します。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。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
キューの事前フェッチ	**queuePrefetch**	ブローカーには同等のプロパティーはありません。ただし、接続 URL に **consumerWindowSize** を設定するか、**ActiveMQConnectionFactory** API に直接設定すると、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際にコンシューマーの優先度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 に存在しません。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
前のメッセージが配信され、確認応答されていない場合に、ブローカーが事前フェッチメッセージをディスパッチできるように事前フェッチエクステンション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 に存在しません。
初期再配信の遅延	**initialRedeliveryDelay** デフォルトは 1000 ミリ秒です。	同等のものはありません。ブローカーインスタンスはこれを自動的に処理します。
キャンセルメッセージの再配信を試みるまで待機する時間	**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。	同等のものはありません。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>** 再配信の遅延に適用する乗数。デフォルトは 1.0 です。
ブローカーの Dead Letter Queue に戻される前にキャンセルされたメッセージが再配信される最大回数	**maximumRedeliveries** デフォルトは 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。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
クライアントが 1 秒で消費できるメッセージの数	同等のものはありません。	ブローカーには同等のプロパティーはありません。ただし、接続 URL に **consumerMaxRate** を設定するか、**ActiveMQConnectionFactory** API に直接設定して、コンシューマーでこれを設定できます。 **consumerMaxRate** プロパティーは、クライアントのバッファー内のメッセージ数には影響を与えません。そのため、クライアントの流量制御が遅く、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がメッセージですぐに一杯になります。

7.2.4. 低速なコンシューマー処理設定プロパティー

AMQ 6 と同様に、AMQ Broker 7 は低速なコンシューマーを検出し、継続して低速な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。

ブローカーがコンシューマーを「低速」と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応答したメッセージの数に基づいてコンシューマーを低速と見なします。AMQ 6 では、事前フェッチバッファーの充足度に基づいてコンシューマーを低速と見なしました (バッファーが継続して一杯である場合、クライアントのメッセージ消費速度が遅すぎる可能性があります)。

以下の表は、AMQ 6 の低速なコンシューマー処理設定プロパティーを AMQ Broker 7 の同等のプロパティーと比較します。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
コンシューマーが停止されるまでに低速と見なされる回数。	**maxSlowCount** デフォルトは -1 (制限なし) です。	同等のものはありません。他の低速なコンシューマー処理プロパティーを使用して、低速なコンシューマーを制御できます。
コンシューマーが停止されるまで継続的に低速でいられる時間	**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「低速」とみなされるまで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。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
コンシューマーが低速かどうかを再度確認するまでにブローカーが待機する時間	**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。
ブローカーが低速なコンシューマーと共に接続を終了するかどうか。	**abortConnection** デフォルトは **false** です。	同等のものはありません。AMQ Broker 7 では、低速なコンシューマーが停止されると、接続も終了されます。
低速なコンシューマーが検出された場合に適用するポリシー。	同等のものはありません。	**<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終了することもできます。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及ぼします。

関連情報

- 低速なコンシューマーの処理方法についての詳細は、『**Configuring AMQ Broker**』の「Handling Slow Consumers」を参照してください。

7.2.5. メッセージページング設定プロパティー

AMQ Broker 7 では、ブローカーがメッセージをメモリーに保管し、それらのメッセージをディスクに保管するプロセスが AMQ 6 とは大きく異なります。そのため、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バッファーにページングされます。アドレスに十分な容量がある場合、キューは 1 ページずつページング解除します。

以下の表は、AMQ 6 のメッセージページングサイズの制限を AMQ Broker 7 の同等のプロパティーと比較します。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
ページングサイズ	**maxPageSize** これはメッセージ数で測定され、利用可能なメッセージの数を基にして変化します。	**<page-size-bytes>** これは、(メッセージではなく) 物理ページサイズ (バイト単位) で測定されます。

7.2.6. デッドレターポリシー設定プロパティー

AMQ Broker 7 は、AMQ 6 とは大きく異なる方法で配信不可能なメッセージと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アドレスに適用され (宛先ではなく)、デッドレターの宛先および有効期限の宛先は独立して (単一のデッドレターキューではなく)、デッドレターポリシー設定は大幅に異なります。

AMQ 6 でのデッドレターポリシー
AMQ 6 では、期限切れまたは配信不可能なメッセージは、各メッセージの宛先に設定された デッドレターキュー (DLQ) に送信されます。宛先に DLQ を設定するには、以下のデッドレターポリシーのいずれかを使用します。

sharedDeadLetterStrategy

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

individualDeadLetterStrategy

宛先の配信不可能なメッセージは、この宛先の専用の DLQ に送信されます。

discardingDeadLetterStrategy

宛先の配信不可能な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、以下の設定プロパティーを追加して、宛先の DLQ に送信される必要があるメッセージの種類を制御できます。

AMQ 6 の設定プロパティー	説明
processNotPersistent	非永続的なメッセージを宛先の DLQ に送信すべきかどうか。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 に送信すべきかどうか。デフォルトは **true** です。
expiration	期限切れを宛先の DLQ に送信されたメッセージに適用すべきかどうか。デフォルトは 0 です。

AMQ 7 でのデッドレターポリシー
AMQ Broker 7 では、配信不可能なメッセージは該当する デッドレターアドレス 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を指定します。配信不可能および期限切れメッセージは、これらの設定によって指定された宛先に送信されます。

```
...
<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、デッドレターと期限切れアドレスは、**<addresses>** セクションで定義される **DLQ** および **ExpiryQueue** 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>
...
```

アドレスにデフォルト以外のデッドレターポリシーを設定するには、**<dead-letter-address>** および **<expiry-address>** をアドレスの **<address-setting>** に追加し、使用する DLQ と期限切れ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の有効期限を設定することはできません。また、永続メッセージと非永続メッセージは、どちらもアドレスの **<dead-letter-address>** で指定された DLQ に送信されます。

第8章 メッセージの永続化およびページング

AMQ Broker 7 は、メッセージジャーナルまたは JDBC ストアのいずれかで永続性を提供します。ブローカーがメッセージを保管しそれらをディスクにページングする方法は AMQ 6 とは異なります。また、メッセージの永続化の設定に使用する設定プロパティーが変更されました。

8.1. メッセージ永続化の変更

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 では、メッセージストアに KahaDB が使用され、メッセージジャーナルインデックスを保持し、ジャーナル内の各メッセージの位置を追跡しました。このインデックスにより、ブローカーはジャーナルからページングされたメッセージをバッチでプルし、キャッシュに配置できるました。

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インメモリーメッセージジャーナルを使用します。そのため、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。ブローカー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ーに到達するものの、キューに置かれる前にページングされます。これらのメッセージページファイルは、到達したのと同じ順序でディスクに順次保存されます。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します。ジャーナルは順番に読み取られるため、ジャーナルにメッセージのインデックスを保持する必要はありません。

さらに、AMQ Broker 7 では、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あります。

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

- NFSv4
- GFS2

関連情報

- デフォルトのインメモリーメッセージジャーナルに関する詳細は、『**Configuring AMQ Broker**』の「About Journal-based Persistence」を参照してください。
- 新しい JDBC ベースの永続化オプションの詳細は、『**Configuring AMQ Broker**』の「Configuring JDBC Persistence」を参照してください。

8.2. メッセージ永続化の設定方法

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、以下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には、いずれかのジャーナル設定プロパティーのデフォルト値を変更します。設定プロパティーをさらに追加することもできます。

8.3. メッセージ永続化設定プロパティーの変更

AMQ 6 および AMQ Broker 7 はどちらも、ブローカーがメッセージを永続化する方法を制御する複数の設定プロパティーを提供します。ここでは、AMQ 6 KahaDB ジャーナルの設定プロパティーを、AMQ Broker 7 のインメモリー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。

インメモリーメッセージジャーナルの各メッセージ永続化設定プロパティーの詳細は、以下を参照してください。

- 『**Configuring AMQ Broker**』の「The Bindings Journal」
- 『**Configuring AMQ Broker**』の「Messaging Journal Configuration Elements」

8.3.1. ジャーナルサイズおよび管理

以下の表は、AMQ 6 のジャーナルサイズおよび管理設定プロパティーを AMQ Broker 7 の同等のプロパティーと比較します。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
使用されていない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。	同等のものはありません。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。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
コンパクション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のあるメッセージストア GC サイクルの数	**compactAcksAfterNoGC**	同等のものはありません。AMQ Broker 7 では、コンパクション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増大する間にコンパクションを実行するべきかどうか、または増大が止まった時にのみ実行するべきか。	**compactAcksIgnoresStoreGrowth** デフォルトは **false** です。	同等のものはありません。
コンパクション前にブローカーに格納できるジャーナルファイルの最小数。	同等のものはありません。	**<journal-compact-min-files>** デフォルトは 10 です。この値を 0 に設定すると、コンパクションは無効になります。
コンパクションの開始前に達するしきい値	同等のものはありません。	**<journal-compact-percentage>** デフォルトは 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、コンパクションが開始されます。
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、ジャーナルのタイプ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。
バインディングディレクトリーがまだ存在しない場合に自動的に作成されるかどうか	同等のものはありません。	**<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されるかどうか	同等のものはありません。	**<create-journal-dir>** デフォルトは **true** です。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
メッセージストアがメッセージの確認応答のみが含まれる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**	同等のものはありません。
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の事前割り当てに使用するポリシー	**preallocationStrategy** デフォルトは **sparse_file** です。	同等のものはありません。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 **entire_journal** です。	ブローカーインスタンスの起動時に、AMQ Broker 7 は **<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割り当てします。
ジャーナルタイプ (NIO または AIO のいずれか)	同等のものはありません。	**<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	同等のものはありません。	**<journal-min-files>**
ファイルを開放する際にブローカーが保持するジャーナルファイルの数	同等のものはありません。	**<journal-pool-files>** デフォルトは -1 で、一旦作成されたらブローカーインスタンスはジャーナル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

以下の表は、AMQ 6 の書き込み境界設定プロパティーを AMQ Broker 7 の同等のプロパティーと比較します。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
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。	同等のものはありません。
メッセージストアがメッセージストレージと同時にクライアントにキューメッセージをディスパッチするかどうか	**concurrentStoreAndDispatchQueues** デフォルトは **true** です。	同等のものはありません。
メッセージストアがメッセージストレージと同時に対象のクライアントにトピックメッセージをディスパッチするかどうか	**concurrentStoreAndDispatchTopics** デフォルトは **false** です。	同等のものはありません。
トランザクション以外の各ジャーナル書き込みの後にディスク同期を実行するべきかどうか	**enableJournalDiskSyncs** デフォルトは **true** です。	**<journal-sync-transactional>** トランザクション境界に到達するたびに、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 (コミット、準備、およびロールバック)。デフォルトは **true** です。 **<journal-sync-nontransactional>** トランザクション以外のメッセージデータをディスクにフラッシュします (送信と確認応答)。デフォルトは **true** です。
ジャーナルバッファー全体をフラッシュするタイミング	同等のものはありません。	**<journal-buffer-timeout>** NIO のデフォルト値は 3,333,333 ナノ秒であり、AIO のデフォルト値は 500,000 ナノ秒です。
ジャーナルディスク書き込み間にバッファーするデータ量	**journalMaxWriteBatchSize** デフォルトは 4000 バイトです。	同等のものはありません。
ジャーナルの書き込みリクエストをバッファー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 キューにある書き込みリクエストの最大数を制御します。NIO のデフォルト値は 1 で、AIO のデフォルト値は 500 です。
ジャーナル書き込みで **fdatasync** を使用するかどうか	同等のものはありません。	**<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複数の設定プロパティーがあります。AMQ Broker 7 では journal 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で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、アーカイブされるファイルやアーカイブの保存先を制御する複数の設定プロパティーがあります。ただし、AMQ Broker 7 では、古いジャーナルファイルが必要なくなった場合に、ブローカーはアーカイブせずにそれらを再利用します。そのため、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. ジャーナルリカバリー

AMQ 6 には、ブローカーが破損したジャーナルファイルの有無を確認する方法、およびジャーナルファイルの不足に遭遇した際の処理を制御する複数の設定プロパティーがあります。

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リカバリープロパティーを設定する必要はありません。AMQ Broker 7 のジャーナルファイルには異なる形式が使用され、ジャーナルの破損したエントリーがジャーナルファイル全体を破損するのを防ぎます。ジャーナルが部分的に破損しても、ブローカーは引き続き破損していないエントリーからデータを抽出できます。

第9章 ブローカークラスター

ブローカーを接続すると、クラスターを形成できます。ブローカークラスターにより、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、それらはクライアントが接続できるブローカーの数を増やしてフォールトトレランスを提供します。

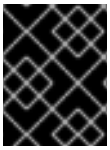
9.1. ブローカーのクラスター化の変更

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によって接続されます (**connector** 要素を参照)。クラスターのメンバーは、(UDP または JGroups を使用して) 動的に、または (クラスターメンバーの一覧を手動で指定して) 静的に、相互に検出するように設定できます。

クラスター構成は、高可用性 (HA) に必須の前提条件です。クラスターが単一のライブブローカーのみで構成される場合でも、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。

ブローカークラスターは、多くの異なるトポロジーで設定できますが、対称およびチェーンクラスターは最も一般的なものです。(メッセージをクラスター内の別のブローカーに送信するようにブローカーを設定している限り) トポロジーに関係なく、メッセージ損失なしにクラスターのスケールアップおよびスケールダウンを実行できます。

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、コンシューマーの興味に基づいて、あるブローから別のブローカーにプルされました。AMQ Broker 7 では、キューの定義およびコンシューマーはクラスター全体で共有され、メッセージが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

同じクラスターで AMQ 6 ブローカーと AMQ Broker 7 ブローカーを組み合わせないでください。

9.2. ブローカークラスターの設定方法

クラスターの各メンバーに ブローカーインスタンスを作成して から、クラスター設定を各ブローカーインスタンスに追加して、ブローカークラスターを設定します。

クラスター設定は以下で構成されます。

検出グループ

動的検出で使用する場合、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どのように検出するかを定義します。検出は、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場合、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内の他のメンバーにクラスター関連の情報を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。メッセージ再配布および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、静的検出を使用する基本的な 2 つのブローカーで構成されるクラスターを作成する方法を説明します。

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 番目のメンバーに 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競合を避ける必要があります。

   この例では、**broker2** という 2 つ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 **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。
静的検出の場合は、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の **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、**broker2** が **broker1** に接続するように設定します。

```
<!-- 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、ならびにメッセージの再配布およびクライアントの負荷分散の設定に関する詳細は、『**Configuring AMQ Broker**積』の「Setting up a broker cluster」を参照してください。

9.2.2. 追加のブローカークラスタートポロジー

ブローカークラスターには、多くの異なるトポロジーで接続できます。AMQ Broker 7 では、対称クラスターおよびチェーンクラスターは最も一般的なものになります。

例: 対称クラスター

完全な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、クラスター内のブローカーはすべて、他のすべてのブローカーから 2 ホップ以上離れていません。

この例では、動的な検出を使用して、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されます。

```
<!-- 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、ブローカーは両端がブローカーの直線状の「チェーン」を形成し、他のすべてのブローカーはチェーンの前のブローカーおよび次のブローカーに接続されます (例: 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され、**max-hops** は **2** に設定され、メッセージがチェーン全体で流れるようにします。

最初のブローカーは以下のように設定されます。

```
<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以下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以下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設定プロパティー

以下の表は、AMQ 6 のブローカーネットワーク設定プロパティーを AMQ Broker 7 の同等の **cluster-connection** プロパティーと比較します。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
除外された宛先	**excludedDestinations**	同等のものはありません。
メッセージがクラスターを通過できるホップの数	**networkTTL** デフォルトは **1** で、メッセージが隣り合うブローカーへの 1 ホップしか移動できないことを意味します。	**<max-hops>** このブローカーインスタンスが、接続されたブローカーに、または直接接続されていないチェーン内の他のブローカー (仲介ブローカー) に、メッセージの負荷を分散するように設定します。デフォルトは **1** で、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配布されることを意味します。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
コンシューマーがない場合のメッセージの再生	**replayWhenNoConsumers**	同等のものはありません。ただし、**<redistribution-delay>** を設定して、コンシューマーのない時間 (ミリ秒単位) を定義できます。この時間が経過すると、メッセージは初めて到達したかのように再配信されます。
クラスター内の一時宛先の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、リクエストリプライメッセージ用に作成された一時的な宛先に使用されました。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でき、引き続き **JMSReplyTo** ヘッダーに指定された一時的な宛先に返信を送信できました。	同等のものはありません。AMQ Broker 7 では、一時的な宛先はクラスター化されません。
このブローカーをリモートブローカーに対して認証するために使用するクレデンシャル	**userNamepassword**	**<cluster-user><cluster-password>**
コネクターのルーティング優先度の設定	**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** で、ネットワークの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 減らされます。	同等のものはありません。
クラスター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	同等のものはありません。	**<message-load-balancing>** これは、**OFF** (負荷分散なし)、**STRICT** (マッチするキューを持つクラスター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マッチするセレクターを持つクラスターのブローカーのみへ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。

設定の対象	AMQ 6 の場合	AMQ Broker 7 の場合
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の有効化	**duplex** デフォルトでは、ネットワークコネクターは一方向です。ただし、これらを duplex に設定し、両方の方向でメッセージが流れるようにすることができます。これは通常、ハブがファイアウォールの裏に隠されるハブ/スポークのネットワークに使用されました。	同等のものはありません。クラスター接続は一方向のみになります。ただし、各ブローカー間で、1 つのクラスター接続 (各エンドから) ペアを設定できます。ブローカークラスターの設定に関する詳細は、**『Configuring AMQ Broker』**の「Setting up a broker cluster」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

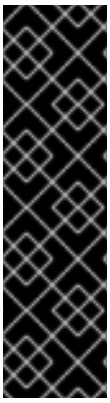
クラスター構成の作成後に、ブローカーインスタンス同士をリンクさせて高可用性 (HA) ペアを形成できます。HA ペアは、クライアントリクエストに対応するマスターブローカーと、マスターがクライアントと通信できなくなった場合にマスターを置き換える 1 つ以上のスレーブブローカーで構成されます。

AMQ Broker 7 では、HA にクラスター構成が必要になります。ブローカークラスターは、非 HA ブローカーのセットまたは HA ペアのいずれかで構成されます。

AMQ Broker 7 は以下の HA ポリシーを提供します。

レプリケーション

レプリケーションでは、ネットワークを通じてマスターとスレーブブローカー間でデータを同期します。レプリケーションにより、障害発生後にマスターブローカーがオンラインに戻ると、制御をマスターに戻すフェイルバックが可能で、クライアントはマスターにフェイルバックすることができます。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し、マスターブローカーと同じ JVM にスレーブブローカーが共存する HA グループを作成することもできます。



重要

7.5 以降では、以前はレプリケーション HA ポリシーで利用できたネットワークの ping 送信は非推奨機能となりました。ネットワークの ping 送信は、復元不可能なメッセージ損失の原因となるネットワーク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。この機能は今後のリリースで削除されます。Red Hat は、ネットワークの ping 送信を使用する既存の AMQ Broker デプロイメントを引き続きサポートします。ただし、Red Hat では、新規デプロイメントでのネットワーク ping 送信の使用を推奨しません。高可用性のためのブローカークラスターの設定およびネットワーク分離の問題の回避についてのガイドラインは、「Implementing high availability」を参照してください。

共有ストア

共有ストアは、マスタ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場所を提供します。レプリケーションに比べて以下の利点を提供するため、共有ストアを使用することは一般的に推奨されます。

- パフォーマンス (共有ストアはより高速です)
- スプリットブレインの問題なし
- クォーラムを維持するために必要なブローカー数が少ない (レプリケーショでは少なくとも 3 つが必要です)
  レプリケーションと同様に、障害発生後に制御をマスターブローカーに戻すフェイルバックが可能で、クライアントはマスターにフェイルバック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共存させることができます。

HA およびフェイルオーバーに関する詳細は、『**Configuring AMQ Broker**』の「Implementing high availability」を参照してください。

10.1. 高可用性およびフェイルオーバーの変更

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関して、AMQ Broker 7 での高可用性は AMQ 6 とは異なります。

AMQ Broker 7 では、マスターロールおよびスレーブロールは固定です。マスターであるブローカーインスタンスを指定し、特定の条件下でのみスレーブがアクティブになります。AMQ 6 では、マスターロールおよびスレーブロールは固定ではありませんでした。代わりに、HA ペアのブローカーはロックについて競争し、勝者がマスターになりました。

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

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

例: 共有ストアを持つ **HA** ペア

マスターブローカーは、以下のように設定されます。**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すると、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>
```

スレーブブローカーは、以下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すると、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 ポリシーの設定に関する詳細は、『**Configuring AMQ Broker**』の以下のトピックを参照してください。

- Configuring shared store high availability
- Configuring replication high availability
- Configuring limited high availability with live-only
- Configuring high availability with colocated backups

改訂日時: 2021-08-01 15:37:49 +1000