Sun Java™ System

# Message Queue 3
# 管理ガイド

2005Q4

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819-3560

# 目 次

# 図目次

# 表目次

# 手順一覧

# はじめに

この『Sun Java™ System Message Queue 管理ガイド』では、Sun Java System Message Queue 3 2005Q4 (Message Queue 3.6) について説明し、Message Queue メッセージングシステムの管理に必要な情報を提供しています。

ここでは、次の節について説明します。



## 対象読者

このマニュアルは、Message Queue 管理タスクを実行する必要がある管理者およびアプリケーション開発者を対象としています。Message Queue 管理者とは、Message Queue メッセージングシステム、特にシステムの中枢となるメッセージサーバーの設定および管理の担当者です。

# お読みになる前に

このマニュアルを読む前に、『Message Queue 技術の概要』に目を通し、Java メッセージの仕様書に従った Message Queue の実装、Message Queue サービスのコンポーネント、Message Queue アプリケーションの開発、配備、管理の基本的なプロセスに慣れてください。

# マニュアルの構成

表 1 に、このマニュアルの内容について簡単に説明しています。

**表 1** このマニュアルの内容

| パート / 章 | 説明 |
|---|---|
| **第 I 部「Message Queue 管理の概要」** | |
| 第 1 章「管理タスクと管理ツール」 | Message Queue 管理タスクとツールを説明します。 |
| 第 2 章「クイックスタートチュートリアル」 | Message Queue 管理コンソールに精通するための実践的なチュートリアルを提供します。 |
| **第 II 部「管理タスク」** | |
| 第 3 章「ブローカとクライアントの起動」 | Message Queue ブローカとクライアントの起動方法を説明します。 |
| 第 4 章「ブローカの設定」 | 設定プロパティーの設定と読み込みの方法を説明し、ブローカの設定可能な部分に関する概略を示します。また、ファイルまたはデータベースを設定して持続機能を実行する方法も説明します。 |
| 第 5 章「ブローカの管理」 | ブローカ管理タスクを説明します。 |
| 第 6 章「物理的送信先の管理」 | 物理的送信先に関する管理タスクを説明します。 |
| 第 7 章「セキュリティーの管理」 | パスワードファイル、認証、承認、および暗号化の管理など、セキュリティー関連のタスクを説明します。 |
| 第 8 章「管理対象オブジェクトの管理」 | オブジェクトストアを説明し、管理対象オブジェクト ( 接続ファクトリと送信先 ) に関連したタスクの実行方法について説明します。 |
| 第 9 章「ブローカクラスタを使用した作業」 | Message Queue ブローカのクラスタの設定方法および管理方法を説明します。 |
| 第 10 章「メッセージサーバーの監視」 | Message Queue の監視機能の設定方法および使用方法を説明します。 |

表 1　このマニュアルの内容 ( 続き )

| パート / 章 | 説明 |
|---|---|
| 第 11 章「メッセージサービスの分析と調整」 | メッセージサーバーのパフォーマンスを分析し、最適化する技術を説明します。 |
| 第 12 章「問題のトラブルシューティング」 | Message Queue の共通の問題の原因の判断、および問題の解決に用いられる処置に関して提案を行います。 |
| **第 III 部「リファレンス」** | |
| 第 13 章「コマンド行のリファレンス」 | Message Queue のコマンドユーティリティーの構文と詳細を示します。 |
| 第 14 章「ブローカのプロパティーのリファレンス」 | ブローカの設定に使用できるプロパティーとその説明を一覧表示しています。 |
| 第 15 章「物理的送信先のプロパティーのリファレンス」 | 物理的送信先の設定に使用できるプロパティーとその説明を一覧表示しています。 |
| 第 16 章「管理対象オブジェクト属性のリファレンス」 | 管理対象オブジェクト ( 接続ファクトリと送信先 ) の設定に使用できるプロパティーとその説明を一覧表示しています。 |
| 第 17 章「JMS リソースアダプタのプロパティーのリファレンス」 | Message Queue リソースアダプタを、アプリケーションサーバー向けに設定する場合に使用できるプロパティーとその説明を一覧表示しています。 |
| 第 18 章「メトリックスのリファレンス」 | Message Queue ブローカで生成されるメトリックスとその説明を一覧表示しています。 |
| **第 IV 部「付録」** | |
| 付録 A「プラットフォームごとの Message Queue データの場所」 | Message Queue ファイルの、サポートされる各プラットフォームでの場所を一覧表示しています。 |
| 付録 B「Message Queue インタフェースの安定度」 | さまざまな Message Queue インタフェースの安定性を説明します。 |
| 付録 C「HTTP/HTTPS のサポート」 | Message Queue 通信に使用する HTTP (Hypertext Transaction protocol) の設定および使用方法を説明します。 |
| 付録 D「よく使用するコマンドユーティリティーのコマンド」 | 使用頻度の高い Message Queue コマンドユーティリティー (imqcmd) コマンドを一覧表示しています。 |

# 表記規則

ここでは、このマニュアルで使用されている表記規則について説明します。

## テキストの表記規則

表 2 に、このマニュアルで使用されている表記規則をまとめています。

**表 2** テキストの表記規則

| 書式 | 説明 |
|---|---|
| *斜体* | 斜体で表記されたテキストは適切な節や値に置き換える必要があります。強調するマニュアル名や説明の対象となる語句や項目に対しても使用されます。 |
| モノスペース | コード例、コマンド行に入力するコマンド、ディレクトリ、ファイルまたはパス名、エラーメッセージテキスト、クラス名、メソッド名 ( シグネチャの全要素を含む )、パッケージ名、予約語、および URL を表します。 |
| [] | コマンド行の構文ステートメントのオプションの値を示します。 |
| すべて大文字 | すべて大文字のテキストは環境変数 (IMQ_HOME など ) または頭文字 (JMS、GIF、HTML など ) を表します。 |
| キー + キー | 同時キーストロークはプラス記号で結合します。Ctrl+A は、Ctrl と A のキーを同時に押すことを表します。 |
| キー - キー | 連続するキーストロークはハイフンで結合します。Esc-S は、Esc キーを押してから離し、次に S キーを押すことを表します。 |

## ディレクトリ変数の表記規則

Message Queue では 3 種類のディレクトリ変数が使用されますが、その設定方法は、プラットフォームによって異なります。表 3 では、これらの変数について説明し、Solaris™、Linux、および Windows の各プラットフォームでの使用方法についても説明します。

**注** このマニュアルでは、ディレクトリ変数を通常のプラットフォーム固有の構文 (UNIX での $IMQ_HOME など ) に関係なく示されています。パス名には、通常、UNIX のディレクトリ区切り文字の表記法 (/) が使用されています。

表 3　Message Queue ディレクトリ変数

| 変数 | 説明 |
|---|---|
| **IMQ_HOME** | Message Queue 基本ディレクトリ ( ルートインストールディレクトリ ) を指します。<br>• Solaris および Linux では使用されず、Message Queue 基本ディレクトリは存在しません。<br>• Windows の場合は、Message Queue インストーラによって設定されます ( デフォルトでは C:¥Program Files¥Sun¥MessageQueue3)。<br>• Solaris および Windows での Sun Java System Application Server の場合、Application Server の基本ディレクトリの下の /imq に設定されます。 |
| **IMQ_VARHOME** | Message Queue の一時的な、または動的に作成された設定ファイルやデータファイルが格納され、任意のディレクトリを指す環境変数として設定可能なディレクトリです。<br>• Solaris の場合、デフォルトは /var/imq に設定されます。<br>• Linux の場合、デフォルトは /var/opt/sun/mq ディレクトリに設定されます。<br>• Windows の場合、デフォルトは IMQ_HOME¥var に設定されます。<br>• Solaris での Sun Java System Application Server の Evaluation Edition の場合、デフォルトは IMQ_HOME/var に設定されます。<br>• Windows での Sun Java System Application Server の場合、デフォルトは IMQ_HOME¥var に設定されます。 |

表 3　Message Queue ディレクトリ変数 ( 続き )

| 変数 | 説明 |
|---|---|
| **IMQ_JAVAHOME** | Message Queue 実行可能ファイルに必要な Java™ ランタイム (JRE) の場所を指します。デフォルトで、次の場所をここに示す順序で検索するように設定されますが、オプションで、必要な JRE が置かれた場所に設定することもできます。<br>• Solaris 8 または 9 の場合 :<br>`/usr/jdk/entsys-j2se`<br>`/usr/jdk/jdk1.5.*`<br>`/usr/jdk/j2sdk1.5.*`<br>`/usr/j2se`<br>• Solaris 10 の場合 :<br>`/usr/jdk/entsys-j2se`<br>`/usr/java`<br>`/usr/j2se`<br>• Linux の場合 :<br>`/usr/jdk/entsys-j2se`<br>`/usr/java/jre1.5.*`<br>`/usr/java/jdk1.5.*`<br>`/usr/java/jre1.4.2*`<br>`/usr/java/j2sdk1.4.2*`<br>• Windows の場合 :<br>`IMQ_HOME¥jre*` |

# 関連マニュアル

この節に挙げる情報リソースは、このマニュアルの情報に加えて、Message Queue についてさらに詳しい情報を提供しています。

## Message Queue マニュアルセット

Message Queue マニュアルセットは、表 4 に示すマニュアルで構成されています。

表 4　Message Queue マニュアル セット

| マニュアル | 対象読者 | 説明 |
|---|---|---|
| 『Message Queue Installation Guide』 | 開発者および管理者 | Message Queue ソフトウェアの Solaris、Linux、Windows の各プラットフォームへのインストール方法を説明します。 |
| 『Message Queue リリースノート』 | 開発者および管理者 | 新機能、制限、既知のバグ、および技術的な注意点を収録します。 |
| 『Message Queue 技術の概要』 | 開発者および管理者 | Message Queue の概念、機能、コンポーネントを説明します。 |
| 『Message Queue 管理ガイド』 | 管理者と開発者 | Message Queue 管理ツールを使用した管理タスクの実行に必要な基本情報を提供します。 |
| 『Message Queue Developer's Guide for Java Clients』 | 開発者 | Message Queue の JMS および SOAP/JAXM 仕様による実装を使用する Java クライアントプログラムの開発に関する情報を提供します。 |
| 『Message Queue Developer's Guide for C Clients』 | 開発者 | Message Queue メッセージサービスとの C アプリケーションプログラミングインタフェース (C-API) を使用した C クライアントプログラムの開発に関する情報を提供します。 |

# Java Message Service 仕様書

Message Queue メッセージサービスは Java Message Service (JMS) アプリケーションプログラミングインタフェースに準拠しており、Java Message Service 仕様書に説明されています。このマニュアルは次の URL にあります。

http://java.sun.com/products/jms/docs.html

# オンラインヘルプ

Message Queue コマンド行ユーティリティーのオンラインヘルプを使用できます。詳細については、第 13 章「コマンド行のリファレンス」を参照してください。Message Queue グラフィカルユーザーインタフェース (GUI) 管理ツールの管理コンソールにも操作状況に合わせて表示できるオンラインヘルプ機能が用意されています。40 ページの「管理コンソールのオンラインヘルプ」を参照してください。

# JavaDoc

JavaDoc 形式の JMS と Message Queue API マニュアルは、表 5 に示す場所に保存されています。このマニュアルは、Netscape または Internet Explorer などの任意の HTML ブラウザで表示できます。このマニュアルには、標準の JMS API マニュアルおよび Message Queue 固有の API が含まれています。

表 5　JavaDoc の場所

| プラットフォーム | ロケーション |
|---|---|
| Solaris | /usr/share/javadoc/imq/index.html |
| Linux | /opt/sun/mq/javadoc/index.html/ |
| Windows | IMQ_HOME/javadoc/index.html |

# クライアントアプリケーション例

Message Queue インストールには、いくつかのクライアントアプリケーションの例を格納したディレクトリがあります。プラットフォームごとの正確な場所については、付録 A「プラットフォームごとの Message Queue データの場所」を参照してください。そのディレクトリには README ファイルがあり、各サブディレクトリにはアプリケーション例に関する説明もあります。

# 関連するサードパーティーの Web サイト

このマニュアルの関連箇所で、追加の関連情報を提供するサードパーティーの URL を参照しています。

---

**注** サンマイクロシステムズ株式会社は、このマニュアルに記載されたサードパーティーの Web サイトの可用性については一切責任を負いません。また、このようなサイトまたはリソースで提供されているコンテンツ、広告、製品、そのほかのマテリアルを支持するわけではなく、それらに対する責任も一切負いません。サンマイクロシステムズ株式会社は、このようなサイトまたはリソースで提供されているコンテンツ、商品、またはサービスを使用もしくは信用したことで、実際に引き起こされた、または引き起こされたと考えられる損害や損失についても一切の責任は負いません。

---

# ご意見をお寄せください

Sun では常時、マニュアルの品質向上のために、お客様からのコメントやご意見をお待ちしています。ご意見がありましたら、次の URL にアクセスしてください。

`http://docs.sun.com`

「**コメントの送信**」をクリックしてください。表示されたオンラインフォームで、ご意見とともにマニュアルのタイトルとパート番号をお知らせください。パート番号は、マニュアルのタイトルページまたは表紙にある 7 桁または 9 桁の番号です。

ご意見をお寄せください

第 I 部

# Message Queue 管理の概要

第 1 章

# 管理タスクと管理ツール

この章では、Sun Java™ System Message Queue 管理タスクの概要と、コマンド行管理ユーティリティーの共通機能に的を絞って、管理タスクを実行するためのツールについて説明します。この章は、次の節から構成されています。



## 管理タスク

実行する一般的な管理タスクは、Message Queue を実行している環境の性質によって異なります。Message Queue アプリケーションを開発およびテストするソフトウェア開発環境に対する要求は、実用的な作業を実行するためにそうしたアプリケーションを展開する本稼動環境に対する要求とは異なります。次の節では、こうした 2 種類の環境の一般的な管理要件の概要を示します。

### 開発環境の管理

開発環境では、柔軟性が重視されます。Message Queue メッセージサーバーは、主に開発中のアプリケーションのテストのために必要となります。概して管理は最小限であり、プログラマが個人用のシステムを管理する場合が多くなります。そのような環境は、一般に次の特性によって識別されます。

- テストで使用するブローカの簡単な起動
- 管理対象のオブジェクトは、管理者によって作成されるのではなく、クライアントコード内でインスタンス化される
- 自動作成された送信先
- ファイルシステムオブジェクトストア

- ファイルベースの持続
- ファイルベースのユーザーリポジトリ
- マルチブローカクラスタ内にマスターブローカがない

# 本稼動環境の管理

本稼動環境では、アプリケーションは確実に配置および実行される必要があるため、管理はより重要になります。実行する管理タスクは、メッセージングシステムの複雑さとメッセージングシステムがサポートするアプリケーションの複雑さによって異なります。それらのタスクは、セットアップ操作とメンテナンス操作の 2 つの大まかな種類に分けることができます。

## セットアップ操作

本稼動環境での管理セットアップ操作には、一般に次の一部またはすべての操作が含まれます。

**管理者のセキュリティー :**

- デフォルトの管理ユーザー (`admin`) のパスワードの設定 (141 ページの「デフォルトの管理者パスワードの変更」)
- 個人またはグループの管理接続サービスへのアクセス (148 ページの「接続サービスのアクセス制御」) とデッドメッセージキュー (149 ページの「物理的な送信先のアクセス制御」) の制御
- 管理グループのファイルベースまたは LDAP (Lightweight Directory Access Protocol) ユーザーリポジトリへのアクセスの制御 (139 ページの「グループ」、143 ページの「管理者のアクセス制御の設定」)

**全般的なセキュリティー :**

- ファイルベースのユーザーリポジトリの内容の管理 (140 ページの「ユーザーリポジトリの設定と管理」) または既存の LDAP ユーザーリポジトリを使用するブローカの設定 (142 ページの「インスタンス設定ファイルの編集」)
- 各ユーザーまたはグループの実行が承認される操作の制御 (144 ページの「ユーザーの承認 : アクセス制御プロパティーファイル」)
- SSL (Secure Socket Layer) を使用した暗号化サービスのセットアップ (151 ページの「SSL ベースのサービスの操作」)

**管理対象オブジェクト :**

- LDAP オブジェクトストアのセットアップと設定 (166 ページの「LDAP サーバーオブジェクトストア」)

- 接続ファクトリおよび送信先の作成 (178 ページの「管理対象オブジェクトの追加」)

**ブローカクラスタ :**

- クラスタ設定ファイルの作成 (186 ページの「クラスタ設定ファイルの使用」)
- マスターブローカの指定 (190 ページの「マスターブローカ」)

**持続性 :**

- 持続ストアを使用するブローカの設定 (95 ページの「持続データストアの設定」)

**メモリー管理 :**

- メモリー利用率を最適化する送信先の設定プロパティーの設定 (125 ページの「物理的送信先のプロパティーの更新」、第 15 章「物理的送信先のプロパティーのリファレンス」)

## メンテナンス操作

本稼働環境では、アプリケーションのパフォーマンス、信頼性、およびセキュリティーが重視されるため、次のような継続的な管理メンテナンス操作によって、メッセージサーバーのリソースを厳しく監視し、制御する必要があります。

**ブローカの管理および調整 :**

- ブローカのメトリックスを使用したブローカの調整および再設定 ( 第 11 章「メッセージサービスの分析と調整」)
- ブローカのメモリーリソースの管理 (81 ページの「ルーティングサービス」)
- メッセージ負荷のバランスを取るブローカクラスタの作成と管理 ( 第 9 章「ブローカクラスタを使用した作業」)
- 障害が発生したブローカの復元 (68 ページの「ブローカの起動」)

**管理対象オブジェクト :**

- クライアントアプリケーションの正常な動作を確保するための接続ファクトリ属性の調整 (169 ページの「接続ファクトリの属性」)
- 物理的送信先の監視と管理 ( 第 6 章「物理的送信先の管理」)
- 送信先へのユーザーアクセスの制御 (149 ページの「物理的な送信先のアクセス制御」)

**クライアント管理 :**

- 永続サブスクリプションの監視と管理 (114 ページの「永続サブスクリプションの管理」を参照 )

- トランザクションの監視と管理 (116 ページの「トランザクションの管理」を参照 )

# 管理ツール

Message Queue の管理ツールは 2 つの種類に分けられます。

- コマンド行ユーティリティー
- グラフィカル管理コンソール

## コマンド行ユーティリティー

Message Queue のユーティリティーはすべて、コマンド行インタフェースからアクセスできます。ユーティリティーコマンドは、共通の形式、構文規則、およびオプションを共有します。これらのユーティリティーには次のものが含まれます。

- ブローカユーティリティー (`imqbrokerd`) はブローカを起動し、それらを クラスタに接続するなどの設定プロパティーを指定します。
- コマンドユーティリティー (`imqcmd`) はブローカとそれらのリソースを制御し、物理的送信先を管理します。
- オブジェクトマネージャーユーティリティー (`imqobjmgr`) は、JNDI (Java Naming and Directory Interface) によってアクセス可能なオブジェクトストア内のプロバイダに依存しない 管理対象オブジェクトを管理します。
- データベースマネージャーユーティリティー (`imqdbmgr`) は JDBC (Java Database Connectivity) 規格を遵守する持続ストレージ用のデータベースを作成し、管理します。
- ユーザーマネージャーユーティリティー (`imqusermgr`) は、ユーザー認証および承認のためのファイルベースのユーザーリポジトリを設定します。
- サービス管理ユーティリティー (`imqsvcadmin`) は Windows のサービスとして、ブローカをインストールし、管理します。
- キーツールユーティリティー (`imqkeytool`) は SSL (Secure Socket Layer) 認証用の自己署名型証明書を生成します。

これらのユーティリティーの使い方の詳細については第 13 章「コマンド行のリファレンス」を参照してください。

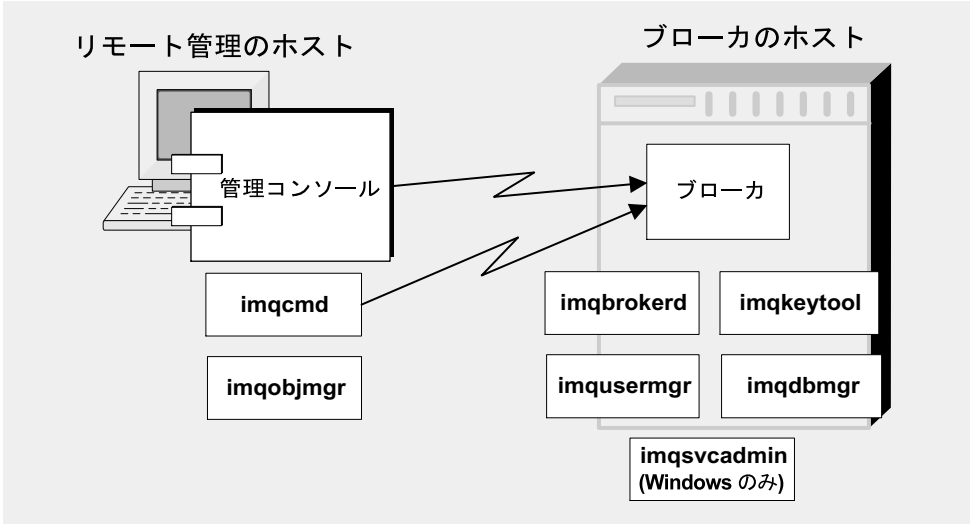
# 管理コンソール

Message Queue 管理コンソールは、コマンドおよびオブジェクトマネージャーユーティリティーの機能を組み合わせたものです。管理コンソールを使用して、次の作業を実行できます。

- ブローカにリモートで接続し、制御します
- 物理的送信先を作成し、管理します
- JNDI オブジェクトストア内の管理対象のオブジェクトを作成し、管理します

ただし、管理コンソールを使用して、ブローカの起動、ブローカクラスタの作成、JDBC データベースまたはユーザーリポジトリの管理、Windows サービスとしてのブローカのインストール、SSL 証明書の生成などの作業は実行できません。これらの作業には、他のコマンド行ユーティリティー ( ブローカ、データベースマネージャー、ユーザーマネージャー、サービス管理、およびキーツール ) が必要ですが、それらはリモートで操作できないため、管理するブローカと同じホスト上で実行する必要があります ( 図 1-1 を参照 )。

**図 1-1**　ローカルおよびリモートの管理ユーティリティー




管理コンソールの簡単で実践的な手引きについては、第 2 章「クイックスタートチュートリアル」を参照してください。管理コンソールの使い方の詳細については、管理コンソールのヘルプ機能を参照してください。

第 2 章

# クイックスタートチュートリアル

このクイックスタートチュートリアルでは、Message Queue の管理の概要を紹介します。メッセージブローカとオブジェクトストアを管理するためのグラフィカルインタフェース Message Queue 管理コンソールを使用した、基本的な管理タスクの実行方法を順を追って説明します。この章は、次の節で構成されています。



このチュートリアルでは、簡単な JMS 互換アプリケーションである `HelloWorldMessageJNDI` の実行に必要な物理的送信先と管理対象オブジェクトを設定します。このアプリケーションは、アプリケーションディレクトリ例 (Solaris と Windows プラットフォームの場合は `demo`、Linux の場合は `examples`、付録 A「プラットフォームごとの Message Queue データの場所」を参照 ) の `helloworld` サブディレクトリから利用できます。チュートリアルの最後の部分では、このアプリケーションを実行します。

**注** チュートリアルを実行するには、Message Queue 製品がインストールされている必要があります。必要に応じて、手順については『Message Queue Installation Guide』を参照してください。

チュートリアルは、基本的な手引きにすぎないため、チュートリアルとは別に、マニュアルを参照するようにしてください。チュートリアルの手順に従うことで、次の作業を実行する方法を学びます。

- メッセージブローカを起動する
- ブローカに接続し、管理コンソールを使用してブローカを管理する
- ブローカに物理的送信先を作成する
- オブジェクトストアを作成し、管理コンソールを使用して接続する
- オブジェクトストアに管理対象オブジェクトを追加し、それらのプロパティーを表示する

---

**注**　このチュートリアルの手順は、Windows プラットフォームに固有のものです。必要に応じて、他のプラットフォームのユーザー向けに補足の注記を追加しています。

---

一部の管理タスクは、管理コンソールを使用して実行できません。次のようなタスクの実行には、コマンド行ユーティリティーを使用します。

- ブローカを起動する
- ブローカクラスタを作成する
- 特定の物理的送信先プロパティーを設定する
- 持続ストレージ用の JDBC データベースを管理する
- ユーザーリポジトリを管理する
- Windows のサービスとしてブローカをインストールする
- SSL 証明書を生成する

これらすべての作業については、このマニュアルの後半の章で説明します。

# 管理コンソールの起動

管理コンソールを起動するには、次の方法のいずれかを使用します。

- Solaris の場合、次のコマンドを入力します。

  ```
  /usr/bin/imqadmin
  ```

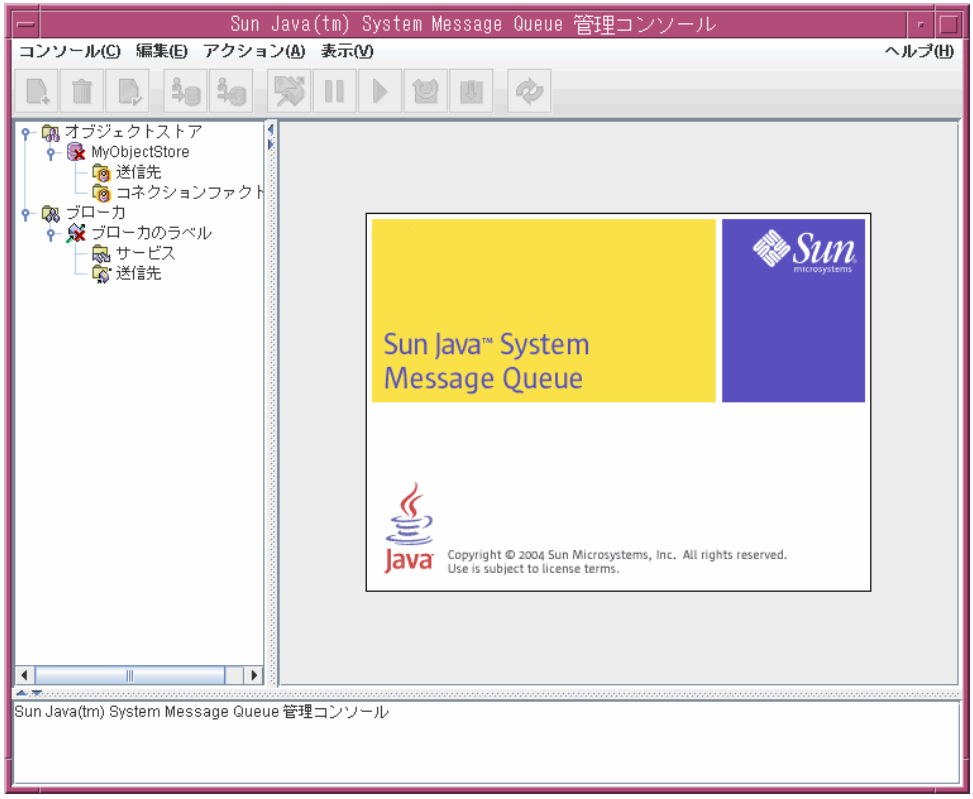
- Linux の場合、次のコマンドを入力します。

  ```
  /opt/sun/mq/bin/imqadmin
  ```

- Windows の場合、「スタート」>「すべてのプログラム」>「Sun Microsystems」>「Sun Java System Message Queue 3.6」>「管理」の順に選択します。

管理コンソールのウィンドウが表示されるまで、数秒かかることがあります ( 図 2-1 を参照 )。

**図 2-1**　管理コンソールウィンドウ



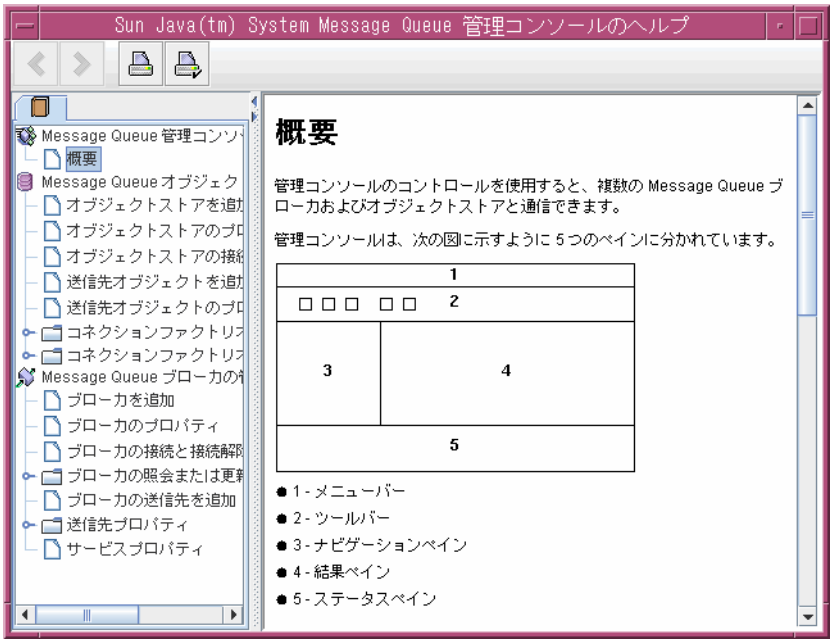

管理コンソールのウィンドウの確認に少し時間を取ってください。コンソールには、一番上にメニューバー、メニューバーのすぐ下にツールバー、左側にナビゲーションの区画、右側に結果を表す区画 ( この図では Sun Java System Message Queue 製品を表すグラフィックが表示されている )、および一番下に状態の区画があります。

**注**　管理コンソールを操作しているときは、「View ( 表示 )」メニューの「Refresh ( 更新 )」コマンドを使用して、ブローカやオブジェクトストアのリストなどのあらゆる要素または要素グループの視覚的表示を更新できます。

# 管理コンソールのオンラインヘルプ

管理コンソールには、コンソールを使用して管理タスクを実行する方法に関するすべての情報を含むヘルプ機能が用意されています。ヘルプ機能を使用するには、メニューバーの右端にあるヘルプメニューをプルダウンし、「Overview ( 概要 )」を選択します。管理コンソールのヘルプウィンドウ ( 図 2-2) が表示されます。

**図 2-2**　管理コンソールヘルプウィンドウ




ヘルプウィンドウの左側にあるナビゲーション区画では、トピックが Message Queue 管理コンソール、Message Queue オブジェクトストア管理、および Message Queue ブローカ管理の 3 つの分野に分類されています。それぞれの領域に、ファイルやフォルダがあります。フォルダは複数のタブが付いたダイアログボックスのヘルプを示し、ファイルは単一のダイアログボックスまたは個別のタブのヘルプを示します。ナビゲーション区画のアイテムを選択すると、右側の結果の区画にそのアイテムの内容が表示されます。「Overview ( 概要 )」アイテムを選択すると、図に示すように、結果区画に管理コンソールウィンドウの各区画を識別する構成図が表示されます。

管理コンソールを使用した最初のタスクは、ブローカへの参照を作成することです。ただし、開始する前に、ヘルプウィンドウの情報を確認します。ヘルプウィンドウのナビゲーション区画にある「Add Broker ( ブローカを追加 )」をクリックすると、結果区画の内容が、ブローカを追加するとはどういうことかを説明し、「Add Broker ( ブローカを追加 )」ダイアログボックスにある各フィールドの使用方法を示すテキストの表示に変わります。ヘルプテキストを読み、ヘルプウィンドウを閉じてください。

# ブローカの操作

この節では、管理コンソールを使用して、メッセージブローカに接続し、管理する方法を説明します。

## ブローカの起動

管理コンソールを使用してブローカを起動することはできません。代わりに次の方法のいずれかを使用します。

- Solaris の場合、次のコマンドを入力します。

  ```
  /usr/bin/imqbrokerd
  ```

- Linux の場合、次のコマンドを入力します。

  ```
  /opt/sun/mq/bin/imqbrokerd
  ```

- Windows の場合、「スタート」>「すべてのプログラム」>「Sun Microsystems > Sun Java System Message Queue 3.6」>「Message Broker」を選択します。

Windows の「スタート」メニューを使用した場合、コマンドウィンドウが表示され、次のような行が表示されて、ブローカの準備ができていることが示されます。

```
Loading persistent data...
Broker "imqbroker@stan:7676 ready.
```

管理コンソールウィンドウを再びアクティブにします。これでコンソールにブローカを追加し、接続する準備ができました。管理コンソールでブローカへの参照を追加する前に、ブローカを起動する必要はありませんが、接続する前にはブローカを起動する必要があります。

## 管理コンソールへのブローカの追加

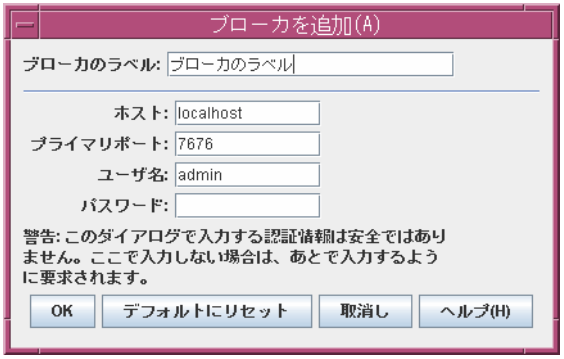
ブローカを追加すると、そのブローカへの参照が管理コンソール内に作成されます。ブローカを追加したあと、そのブローカへ接続できます。

### ➤ 管理コンソールにブローカを追加する

1. 管理コンソールウィンドウのナビゲーション区画にある「Brokers ( ブローカ )」アイテムをクリックし、「Actions ( アクション )」メニューから「Add Broker ( ブローカを追加 )」を選択します。

   または、「Brokers ( ブローカ )」を右クリックし、ポップアップコンテキストメニューから「Add Broker ( ブローカを追加 )」を選択します。どちらの場合も「Add Broker ( ブローカを追加 )」ダイアログボックス ( 図 2-3) が表示されます。

図 2-3　「Add Broker ( ブローカを追加 )」ダイアログボックス




2. 「Broker Label ( ブローカのラベル )」フィールドにブローカの名前を入力します。

   これにより、管理コンソールでブローカを識別するラベルが作成されます。

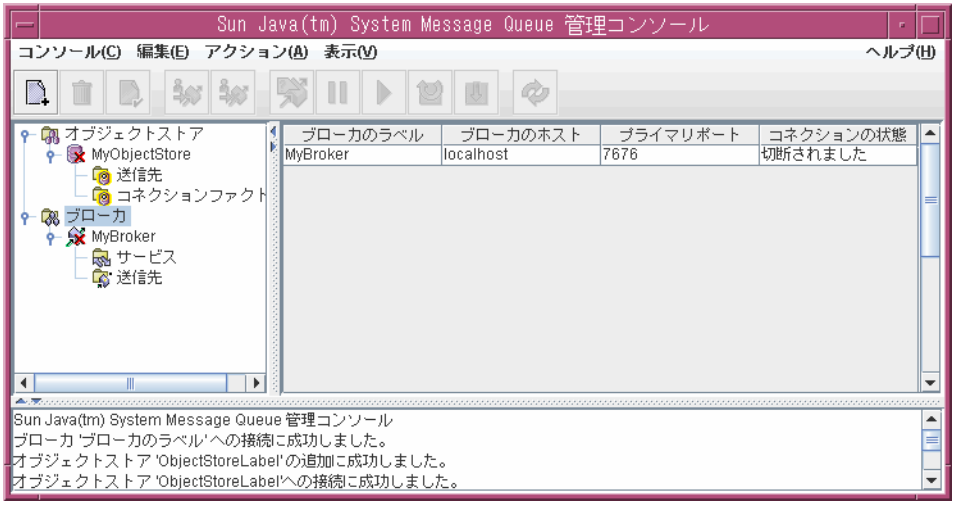
   ダイアログボックスで指定されたデフォルトのホスト名 (`localhost`) と、プライマリポート (`7676`) を控えておきます。これらの値はあとで、クライアントがこのブローカへの接続を作成するのに必要な接続ファクトリを設定するときに、この値を指定する必要があります。

   この演習では、「Broker Label ( ブローカのラベル )」フィールドに `MyBroker` という名前を入力します。「Password ( パスワード )」フィールドは空欄のままにしておきます。パスワードは、接続を実行するときに指定したほうが安全です。

3. 「OK」をクリックして、ブローカを追加し、ダイアログボックスを閉じます。

   図 2-4 に示すように、新しいブローカがナビゲーション区画の「Brokers ( ブローカ )」の下に表示されます。ブローカアイコンの上についている赤い×印は、そのブローカが現在管理コンソールに接続されていないことを表しています。

図 2-4　管理コンソールウィンドウに表示されたブローカ




ブローカを追加したら、「Actions ( アクション )」メニューの「Properties ( プロパティー )」コマンドまたはポップアップコンテキストメニューを使用して、図 2-3 に示す「Add Broker ( ブローカを追加 )」ダイアログボックスに類似した「Broker Properties ( ブローカのプロパティー )」ダイアログボックスを表示して、ブローカのプロパティーを表示または変更できます。

# ブローカへの接続

これで、管理コンソールにブローカが追加され、ブローカへの接続に進むことができます。

### ➤ ブローカに接続する

1. 管理コンソールウィンドウのナビゲーション区画にあるブローカの名前をクリックし、「Actions ( アクション )」メニューから「Connect to Broker ( ブローカに接続 )」を選択します。

   または、ブローカの名前を右クリックし、ポップアップコンテキストメニューから「Connect to Broker ( ブローカに接続 )」を選択します。どちらの場合も「Connect to Broker ( ブローカに接続 )」ダイアログボックス ( 図 2-5) が表示されます。

図 2-5　「Connect to Broker ( ブローカに接続 )」ダイアログボックス


2. ブローカに接続するためのユーザー名とパスワードを入力します。

   ダイアログボックスには最初にデフォルトのユーザー名 admin が表示されます。この演習では、デフォルトの値を使用しますが、実際の環境では、できるだけ早く安全なユーザー名とパスワードを設定する必要があります (135 ページの「ユーザーの認証」を参照 )。

   デフォルトのユーザー名に関連付けられているパスワードも admin であり、ダイアログボックスの「Password ( パスワード )」フィールドにこのパスワードを入力します。これにより、管理者権限でブローカに接続されます。

3. 「OK」をクリックして、ブローカに接続し、ダイアログボックスを閉じます。

ブローカに接続したら、「Actions ( アクション )」メニューのコマンドまたはコンテキストメニューを使用して、選択したブローカに対して次の操作を実行できます。

- 「Pause Broker ( ブローカを一時停止 )」は実行中のブローカの処理を一時的に中断します。
- 「Resume Broker ( ブローカを再開する )」は停止しているブローカの処理を再開します。
- 「Restart Broker ( ブローカを再起動する )」はブローカを再初期化し、再起動します。
- 「Shut Down Broker ( ブローカを停止する )」はブローカの処理を停止します。
- 「Query/Update Broker ( ブローカの照会または更新 )」は、ブローカの設定プロパティーを表示または変更します。
- 「Disconnect from Broker ( ブローカから切断する )」は、ブローカと管理コンソール間の接続を終了します。

# 接続サービスを表示する

ブローカは、提供する接続サービスと、サポートする物理的送信先とで識別されます。

## ➤ 利用可能な接続サービスを表示する

1. 管理コンソールウィンドウのナビゲーション区画のブローカ名の下の「Services ( サービス )」を選択します。

   利用できるサービスの一覧が結果区画に表示され ( 図 2-6 を参照 )、各サービスの名前、ポート番号、現在の状態が示されます。

図 2-6　　接続サービスを表示する

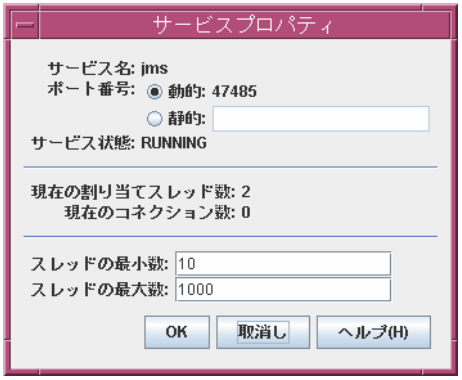

2. 結果区画のサービスの名前をクリックして、サービスを選択します。

   この演習では、名前 jms を選択します。

3. 「Actions ( アクション )」メニューから「Properties ( プロパティー )」を選択します。

   「Service Properties ( サービスプロパティー )」ダイアログボックス ( 図 2-7) が表示されます。このダイアログボックスを使用して、サービスに静的ポート番号を割り当て、それに割り当てられるスレッドの最小数および最大数を変更できます。

図 2-7 「Service Properties ( サービスプロパティー )」ダイアログボックス




この演習では、接続サービスプロパティーを変更しないでください。

4. 「OK」をクリックして、新しいプロパティー値を受け入れ、ダイアログボックスを閉じます。

「Actions ( アクション )」メニューには、サービスを停止し、再開するためのコマンドもあります。ただし、管理サービスを選択し、「Actions ( アクション )」メニューをプルダウンすると、「Pause Service ( サービスを一時停止 )」コマンドが無効にされています。これは、管理サービスがブローカへの管理コンソールのリンクであるためです。管理サービスを停止すると、ブローカにアクセスできなくなります。

# 物理的送信先の操作

物理的送信先は、メッセージプロデューサから受信したメッセージをあとで 1 つ以上のメッセージコンシューマに配信するために保存されているメッセージブローカ上の場所です。送信先は、使用しているメッセージングドメイン、つまり キュー ( ポイントツーポイントドメイン ) とトピック ( パブリッシュ / サブスクライブドメイン ) によって、2 種類あります。メッセージングドメインおよびそれらに関連付けられる送信先の詳細については、『Message Queue 技術の概要』を参照してください。

# 物理的送信先の作成

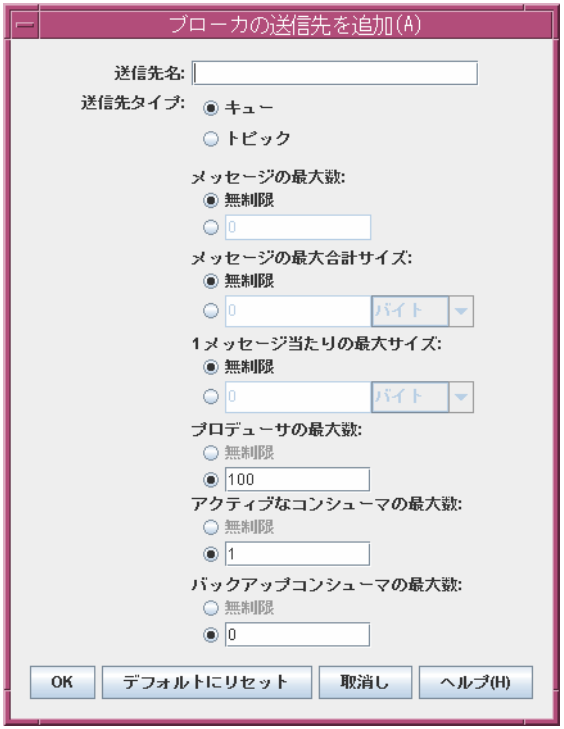
デフォルトで、メッセージプロデューサまたはコンシューマが、存在しない送信先にアクセスしようとした場合に、新しい物理的送信先を自動的に作成するように、メッセージブローカを設定できます。それらの自動作成される送信先は、ソフトウェア開発環境でクライアントコードをテストしている場合に使用すると便利です。ただし、運用時の設定では、送信先の自動作成を無効にし、すべての送信先を管理者が明示的に作成することをお勧めします。次の手順に、管理者が作成した送信先をブローカに追加する方法を示します。

### ➤ ブローカに物理的送信先を追加する

1. 管理コンソールウィンドウのナビゲーション区画にあるブローカの名前の下の「Destinations ( 送信先 )」アイテムをクリックし、「Actions ( アクション )」メニューから「Add Broker Destination ( ブローカの送信先を追加 )」を選択します。

   または、「Destinations ( 送信先 )」を右クリックし、ポップアップコンテキストメニューから「Add Broker Destination ( ブローカの送信先を追加 )」を選択します。どちらの場合も「Add Broker Destination ( ブローカの送信先を追加 )」ダイアログボックス ( 図 2-8) が表示されます。

図 2-8 「Add Broker Destination ( ブローカの送信先を追加 )」ダイアログボックス




2. 「Destination Name ( 送信先名 )」フィールドに物理的送信先の名前を入力します。

   送信先に割り当てた名前を控えておきます。あとでこの物理的送信先に対応する管理対象オブジェクトを作成するときに必要となります。

   この演習では、`MyQueueDest` と入力します。

3. 「Queue ( キュー )」または「Topic ( トピック )」のラジオボタンを選択し、作成する送信先のタイプを指定します。

   この演習では、「Queue ( キュー )」ラジオボタンが選択されていなければ選択します。

4. 「OK」をクリックして、物理的送信先を追加し、ダイアログボックスを閉じます。

   新しい送信先が結果区画に表示されます。

# 物理的送信先のプロパティーの表示

管理コンソールの「Actions ( アクション )」メニューの「Properties ( プロパティー )」コマンドを使用して、物理的送信先のプロパティーを表示または変更できます。

## ➤ 物理的送信先のプロパティーを表示または変更する

1. 管理コンソールウィンドウのナビゲーション区画のブローカ名の下の「Destinations ( 送信先 )」を選択します。

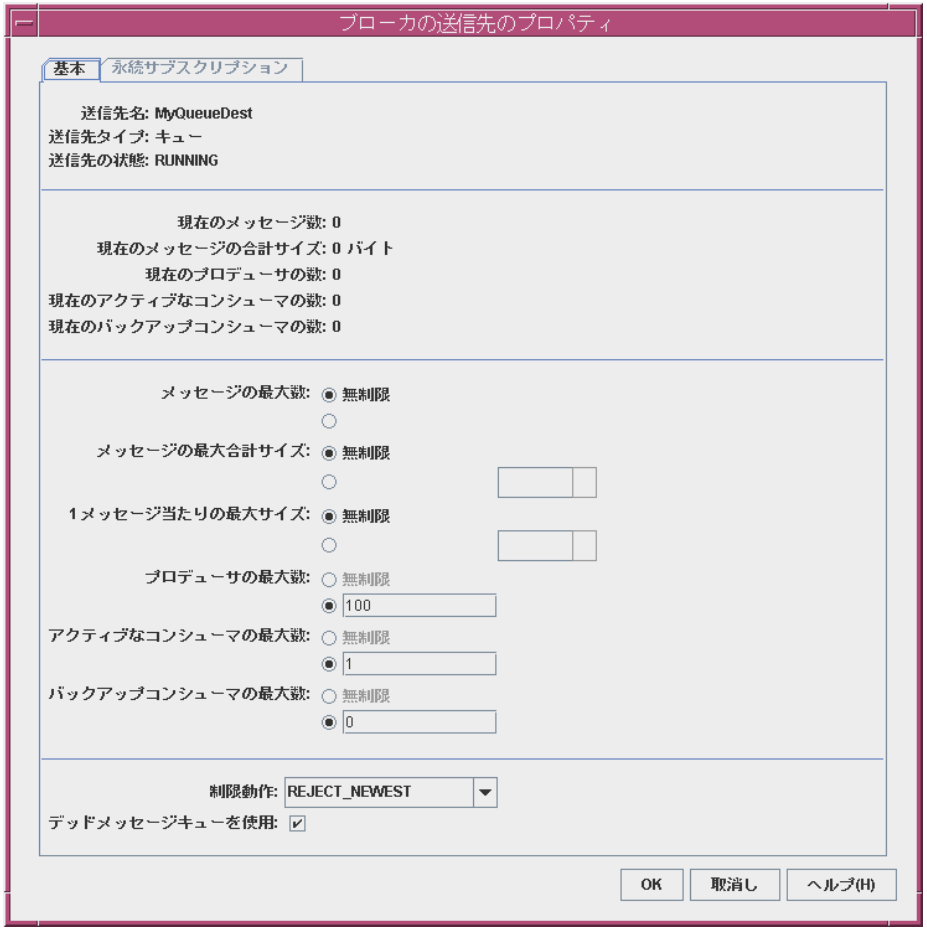
   利用できる物理的送信先の一覧が結果区画に表示され、各送信先の名前、タイプ、現在の状態が示されます。

2. 結果区画の送信先の名前をクリックして、送信先を選択します。

3. 「Actions ( アクション )」メニューから「Properties ( プロパティー )」を選択します。

   「Broker Destination Properties ( ブローカの送信先のプロパティー )」ダイアログボックス ) ( 図 2-9) が表示され、選択した物理的送信先の現在の状態および設定情報が示されます。このダイアログボックスを使用して、送信先で対応可能なメッセージ、プロデューサ、コンシューマの最大数などのさまざまな設定プロパティーを変更できます。

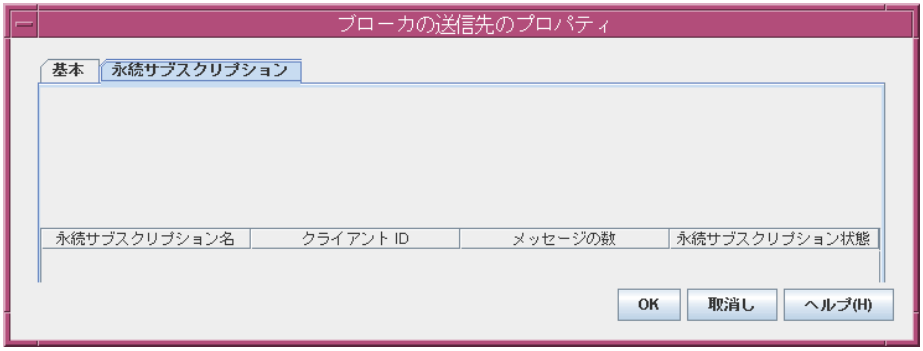
図 2-9 「Broker Destination Properties ( ブローカの送信先のプロパティー )」ダイアログボックス




この演習では、送信先プロパティーを変更しないでください。

トピック送信先の場合、「Broker Destination Properties ( ブローカの送信先のプロパティー )」ダイアログボックスには、追加のタブ「Durable Subscriptions ( 永続サブスクリプション )」が表示されます。このタブをクリックすると、「Durable Subscriptions ( 永続サブスクリプション )」パネル ( 図 2-10) が表示され、現在特定のトピックに関連付けられているすべての永続サブスクリプションに関する情報が示されます。

図 2-10　「Durable Subscriptions ( 永続サブスクリプション )」パネル




「Durable Subscriptions ( 永続サブスクリプション )」パネルの「Purge ( 消去 )」ボタンおよび「Delete ( 削除 )」ボタンを使用して、次の操作を実行できます。

- 永続サブスクリプションに関連付けられているすべての保留中のメッセージを消去する
- トピックから永続サブスクリプションを削除する

キュー送信先の場合、「Durable Subscriptions ( 永続サブスクリプション )」タブは無効になります。

4. 「OK」をクリックして、新しいプロパティー値を受け入れ、ダイアログボックスを閉じます。

# 物理的送信先からのメッセージの消去

物理的送信先からメッセージを消去すると、送信先に関連付けられている保留中のすべてのメッセージが削除され、送信先が空になります。

### ➤ 物理的送信先からメッセージを消去する

1. 管理コンソールウィンドウのナビゲーション区画のブローカ名の下の「Destinations ( 送信先 )」を選択します。

   利用できる物理的送信先の一覧が結果区画に表示され、各送信先の名前、タイプ、現在の状態が示されます。

2. 結果区画の送信先の名前をクリックして、送信先を選択します。
3. 「Actions ( アクション )」メニューから「Purge Messages ( メッセージを消去する )」を選択します。

   確認のダイアログボックスが表示され、処理を続けることを確認するように求められます。

4. 「Yes ( はい )」をクリックして処理を確認し、確認ダイアログを閉じます。

## 物理的送信先の削除

物理的送信先の削除を行うと、そのメッセージがすべて消去されたあと、送信先自体も破棄され、それが属するブローカから完全に削除されます。

### ➤ 物理的送信先を削除する

1. 管理コンソールウィンドウのナビゲーション区画のブローカ名の下の「Destinations ( 送信先 )」を選択します。

   利用できる送信先の一覧が結果区画に表示され、各送信先の名前、タイプ、現在の状態が示されます。

2. 結果区画の送信先の名前をクリックして、送信先を選択します。
3. 「Edit ( 編集 )」メニューから「Delete ( 削除 )」を選択します。

   確認のダイアログボックスが表示され、処理を続けることを確認するように求められます。

4. 「Yes ( はい )」をクリックして処理を確認し、確認ダイアログを閉じます。

   この演習では、前に作成した送信先 `MyQueueDest` を削除しないで、「No ( いいえ )」をクリックし、削除操作を実行せずに、確認のダイアログを閉じてください。

# オブジェクトストアの操作

オブジェクトストアは、Message Queue 管理対象オブジェクトを格納するために使用され、特定の Message Queue プロバイダに固有の実装および設定情報をカプセル化します。オブジェクトストアは、LDAP (Lightweight Directory Access Protocol) ディレクトリサーバーかローカルファイルシステムのディレクトリのいずれかになります。

クライアントアプリケーションのコード内から管理対象のオブジェクトを直接インスタンス化し、設定することは可能ですが、一般に管理者がこれらのオブジェクトを作成し、設定して、クライアントアプリケーションから JNDI (Java Naming and Directory Interface) を使用してアクセス可能なオブジェクトストアに保存することをお勧めします。これにより、クライアントコード自体はプロバイダに依存しなくなります。

# オブジェクトストアを追加する

管理コンソールを使用して、オブジェクトストアを管理できますが、作成することはできません。オブジェクトストアとして使用される LDAP サーバーまたはファイルシステムディレクトリは事前に存在している必要があります。この既存のオブジェクトストアを管理コンソールに追加して、オブジェクトストアへの参照を作成し、コンソール内から操作できるようにすることができます。

---

**注**　この章で使用するサンプルアプリケーションでは、オブジェクトストアが C ドライブの `Temp` というディレクトリに保持されていることが前提となっています。C ドライブに `Temp` という名前のフォルダが存在しない場合は、次の演習に進む前に、作成しておきます。Windows 以外のプラットフォームの場合、既存の `/tmp` ディレクトリを使用できます。

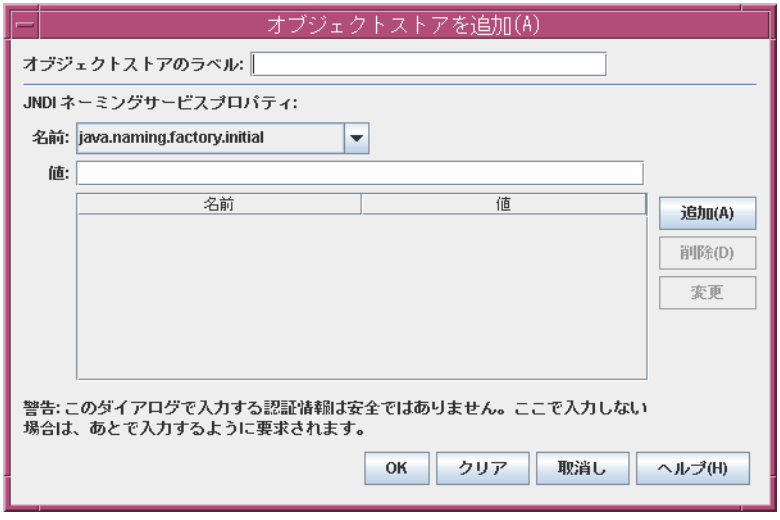
---

## ➤ 管理コンソールにオブジェクトストアを追加する

1. 管理コンソールウィンドウのナビゲーション区画にある「Object Stores ( オブジェクトストア )」アイテムをクリックし、「Actions ( アクション )」メニューから「Add Object Store ( オブジェクトストアを追加 )」を選択します。

   または、オブジェクトストアを右クリックし、ポップアップコンテキストメニューから「Add Object Store ( オブジェクトストアを追加 )」を選択します。どちらの場合も「Add Object Store ( オブジェクトストアを追加 )」ダイアログボックス ( 図 2-11) が表示されます。

図 2-11　「Add Object Store ( オブジェクトストアを追加 )」ダイアログボックス



2. 「Object Store Label ( オブジェクトストアのラベル )」フィールドにオブジェクトストアの名前を入力します。

   これにより、管理コンソールでオブジェクトストアを識別するラベルが作成されます。

   この演習では、名前 `MyObjectStore` を入力します。

3. 管理対象オブジェクトを検索するとき使用する JNDI 属性の値を入力します。

   a. 「Name ( 名前 )」プルダウンメニューから、指定する属性の名前を選択します。

   b. 「Value ( 値 )」フィールドに属性の値を入力します。

   c. 「Add ( 追加 )」ボタンをクリックして、指定した属性値を追加します。

      プロパティーの概要区画にプロパティーとその値が表示されます。

   設定する必要がある属性の数だけ、a から c の手順を繰り返します。

   この演習では、`java.naming.factory.initial` 属性を次のように設定します。

   ```
   com.sun.jndi.fscontext.RefFSContextFactory
   ```

   さらに、`java.naming.provider.url` 属性を次のように設定します。

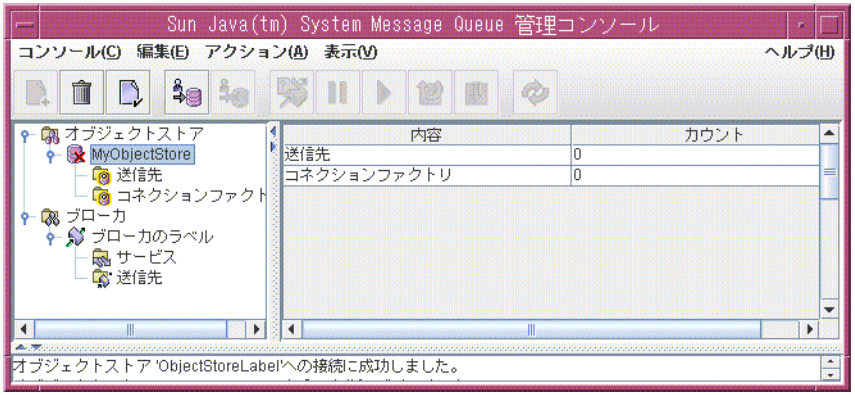
   ```
   file:///C:/Temp
   ```

   または、Solaris または Linux プラットフォームでは `file:///tmp` に設定します。これらは、ファイルシステムオブジェクトストアを使用している場合に設定する必要がある属性です。LDAP ストアに必要な属性値については、166 ページの「LDAP サーバーオブジェクトストア」を参照してください。

4. 「OK」をクリックして、オブジェクトストアを追加し、ダイアログボックスを閉じます。

   図 2-12 に示すように、新しいオブジェクトストアがナビゲーション区画のオブジェクトストアの下に表示されます。オブジェクトストアのアイコンの上についている赤い×印は、そのオブジェクトストアが現在管理コンソールに接続されていないことを表しています。

図 2-12　管理コンソールウィンドウに表示されたオブジェクトストア




ナビゲーション区画のオブジェクトストアをクリックすると、結果の区画にその内容が表示されます。オブジェクトストアにはまだ管理対象オブジェクトを追加していないため、送信先と接続ファクトリのどちらについても「Count ( カウント )」列に 0 が表示されます。

オブジェクトストアを追加したら、「Actions ( アクション )」メニューまたはポップアップコンテキストメニューの「Properties ( プロパティー )」コマンドを使用して、図 2-11 に示す「Add Object Store ( オブジェクトストアを追加 )」ダイアログに類似した「Object Store Properties ( オブジェクトストアのプロパティー )」ダイアログボックスを表示して、プロパティーを表示または変更できます。

# オブジェクトストアに接続する

オブジェクトストアを管理コンソールに追加したので、管理対象オブジェクトを追加するために、オブジェクトストアに接続する必要があります。

### ➤ オブジェクトストアに接続する

1. 管理コンソールウィンドウのナビゲーション区画にあるオブジェクトストアの名前をクリックし、「Actions ( アクション )」メニューから「Connect to Object Store ( オブジェクトストアに接続 )」を選択します。

   または、オブジェクトストアの名前を右クリックし、ポップアップコンテキストメニューから「Connect to Object Store ( オブジェクトストアに接続 )」を選択します。どちらの場合も、オブジェクトストアのアイコンから赤い×印が消去され、管理コンソールに接続されたことが示されます。

# 管理対象オブジェクトの操作

オブジェクトストアを管理コンソールに接続したら、管理対象オブジェクト ( 接続ファクトリおよび送信先 ) の追加に進むことができます。この節ではその方法について説明します。

| | |
|---|---|
| **注** | 管理コンソールには Message Queue 管理対象オブジェクトだけが表示されます。オブジェクトストアに、追加したい管理対象オブジェクトと同じ検索名の Message Queue 以外のオブジェクトが含まれている場合は、追加操作を実行しようとするとエラーが表示されます。 |

## 接続ファクトリの追加

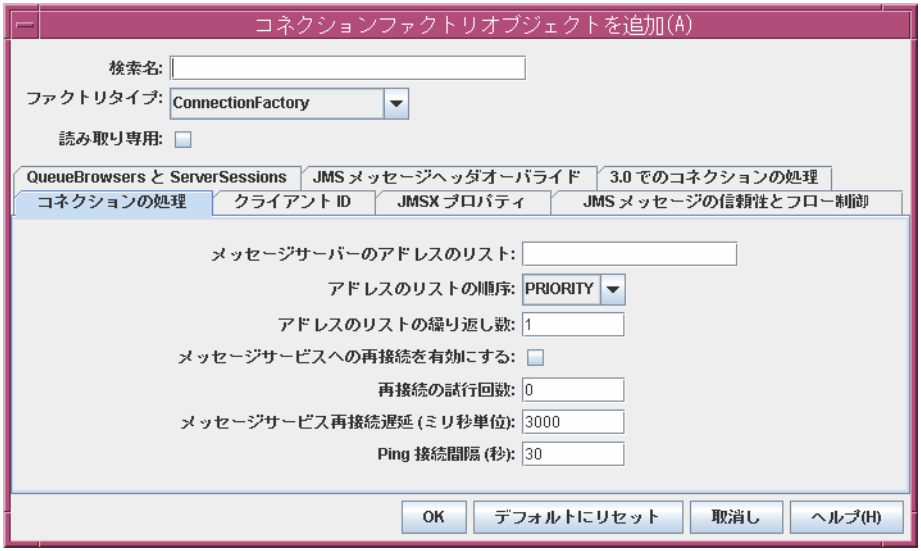
接続ファクトリは、クライアントアプリケーションがブローカへの接続を作成するために使用します。接続ファクトリを設定すると、作成する接続のプロパティーを制御できます。

### ➤ 接続ファクトリをオブジェクトストアに追加する

1. オブジェクトストアが管理コンソールに接続されていることを確認します (55 ページの「オブジェクトストアに接続する」を参照 )。
2. 管理コンソールウィンドウのナビゲーション区画にあるオブジェクトストアの名前の下の「Connection Factories ( 接続ファクトリ )」アイテムをクリックし、「Actions ( アクション )」メニューから「Add Connection Factory Object ( 接続ファクトリオブジェクトを追加 )」を選択します。

   または、「Connection Factories ( 接続ファクトリ )」を右クリックし、ポップアップコンテキストメニューから「Add Connection Factory Object ( 接続ファクトリオブジェクトを追加 )」を選択します。どちらの場合も「Add Connection Factory Object ( 接続ファクトリオブジェクトを追加 )」ダイアログボックス ( 図 2-13) が表示されます。

図 2-13　「Add Connection Factory Object ( 接続ファクトリオブジェクトを追加 )」ダイアログボックス




3. 「Lookup Name ( 検索名 )」フィールドに接続ファクトリの名前を入力します。

   この名前は、クライアントアプリケーションが JNDI によって接続ファクトリを検索するときに使用する名前です。

   この演習では、`MyQueueConnectionFactory` と入力します。

4. 「Factory Type ( ファクトリタイプ )」プルダウンメニューから、作成する接続ファクトリのタイプを選択します。

   この演習では、QueueConnectionFactory を選択します。

5. 「Connection Handling ( 接続の処理 )」タブをクリックします。

   図 2-13 に示すように、「Connection Handling ( 接続の処理 )」パネルが表示されます。

6. 「Message Server Address List ( メッセージサーバーのアドレスのリスト )」フィールドに、この接続ファクトリで接続を作成するブローカのアドレスを入力します。

   このアドレスリストには、1 つのブローカまたは ( ブローカクラスタの場合に ) 複数のブローカを指定できます。ブローカごとに、ブローカの接続サービス、ホスト名、ポート番号などの情報を指定します。指定する情報の実際の性質や構文は、使用される接続サービスによって異なります。仕様については、330 ページの「接続の処理」を参照してください。

サンプルアプリケーション HelloWorldMessageJNDI は、接続ファクトリが、デフォルトで自動的に設定される標準アドレスリストの属性 ( 接続サービス jms、ホスト名 localhost、ポート番号 7676) を使用することを想定しています。したがって、この演習では、「Message Server Address List ( メッセージサーバーのアドレスのリスト )」フィールドに何も入力する必要はありません。

7. 必要に応じて、接続ファクトリのその他の属性を設定します。

   「Add Connection Factory Object ( 接続ファクトリオブジェクトを追加 )」ダイアログボックスに、「Connection Handling ( 接続の処理 )」以外の多数のパネルが表示されます。これらを使用して、接続ファクトリのさまざまな属性を設定できます。

   この演習では、他の属性設定を変更しないでください。ただし、他のタブをクリックして、指定可能な設定情報の種類を知ることは有益な場合があります。これらの他の設定パネルの内容について詳しく知るには、「Help ( ヘルプ )」ボタンを使用してください。

8. 必要に応じて、「Read-Only ( 読み取り専用 )」チェックボックスをクリックします。

   これによって、接続ファクトリオブジェクトの設定属性が、作成時に指定された値にロックされます。読み取り専用の管理対象オブジェクトの属性は、クライアントコードからプログラムによっても、コマンド行から管理者によってもオーバーライドできません。

   この演習では、「Read-Only ( 読み取り専用 )」をチェックしないでください。

9. 「OK」をクリックして、接続ファクトリを作成し、それをオブジェクトストアに追加して、ダイアログボックスを閉じます。

   新しい接続ファクトリが結果区画に表示されます。

# 送信先の追加

送信先管理対象オブジェクトは、ブローカ上の物理的送信先を表し、クライアントはプロバイダ固有の設定とネーミング構文とは関係なく、物理的送信先にメッセージを送信できます。クライアントが管理対象のオブジェクトによってアドレス設定されたメッセージを送信すると、ブローカは対応する物理的送信先 ( 存在する場合 ) にメッセージを配信します。物理的送信先が存在しない場合、自動作成が有効ならば、47 ページの「物理的送信先の作成」に説明されているとおりに、ブローカによって自動的に作成され、メッセージが配信されます。自動作成が有効でなければ、メッセージを配信できないことを通知するエラーが生成されます。

次の手順に、既存の物理的送信先に対応するオブジェクトストアに、送信先管理対象オブジェクトを追加する方法を説明します。

### ➤ 送信先をオブジェクトストアに追加する

1. オブジェクトストアが管理コンソールに接続されていることを確認します (55 ページの「オブジェクトストアに接続する」を参照 )。

2. 管理コンソールウィンドウのナビゲーション区画にあるオブジェクトストアの名前の下の「Destinations ( 送信先 )」アイテムをクリックし、「Actions ( アクション )」メニューから「Add Destination Object ( 送信先オブジェクトを追加 )」を選択します。

   または、送信先を右クリックし、ポップアップコンテキストメニューから「Add Destination Object ( 送信先オブジェクトを追加 )」を選択します。どちらの場合も「Add Destination Object ( 送信先オブジェクトを追加 )」ダイアログボックス ( 図 2-14) が表示されます。

   図 2-14　「Add Destination Object ( 送信先オブジェクトを追加 )」ダイアログボックス


3. 「Lookup Name ( 検索名 )」フィールドに送信先管理対象オブジェクトの名前を入力します。

   この名前は、クライアントアプリケーションが JNDI によって送信先を検索するときに使用する名前です。

   この演習では、`MyQueue` と入力します。

4. 「Queue ( キュー )」または「Topic ( トピック )」のラジオボタンを選択し、作成する送信先オブジェクトのタイプを指定します。

   この演習では、「Queue ( キュー )」ラジオボタンが選択されていなければ選択します。

5. 「Destination Name ( 送信先名 )」フィールドに、対応する物理的送信先の名前を入力します。

この名前は、ブローカに物理的送信先を追加したときに指定した名前です (46 ページの「物理的送信先の操作」を参照 )。

この演習では、MyQueueDest と入力します。

6. オプションで、「Destination Description ( 送信先の説明 )」フィールドに、送信先の簡単な説明を入力します。

   このフィールドの内容は、ユーザーが参照することのみを目的としているため、クライアントの処理には影響がありません。

   この演習では、「Destination Description ( 送信先の説明 )」フィールドの内容を削除しても、次のような説明のテキストを入力してもかまいません。

   MQ 管理ガイドチュートリアルの送信先例

7. 必要に応じて、「Read-Only ( 読み取り専用 )」チェックボックスをクリックします。

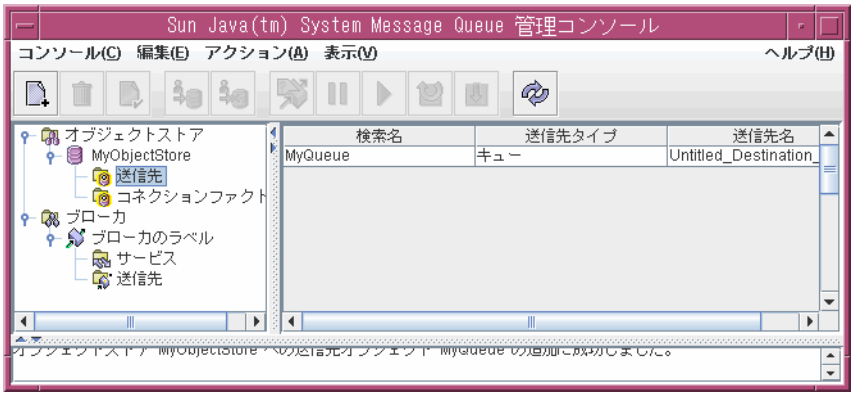
   これによって、送信先オブジェクトの設定属性が、作成時に指定された値にロックされます。読み取り専用の管理対象オブジェクトの属性は、クライアントコードからプログラムによっても、コマンド行から管理者によってもオーバーライドできません。

   この演習では、「Read-Only ( 読み取り専用 )」をチェックしないでください。

8. 「OK」をクリックして、送信先オブジェクトを作成し、それをオブジェクトストアに追加して、ダイアログボックスを閉じます。

   図 2-15 に示すように、新しい送信先オブジェクトが結果区画に表示されます。

**図 2-15** 管理コンソールウィンドウに表示された送信先オブジェクト



# 管理対象オブジェクトのプロパティーの表示

管理コンソールの「Actions ( アクション )」メニューの「Properties ( プロパティー )」コマンドを使用して、管理対象オブジェクトのプロパティーを表示または変更できます。

### ➤ 管理対象オブジェクトのプロパティーを表示または変更する

1. 管理コンソールウィンドウのナビゲーション区画のオブジェクトストア名の下の「Connection Factories ( 接続ファクトリ )」または「Destinations ( 送信先 )」を選択します。

   利用できる接続ファクトリまたは送信先管理対象オブジェクトの一覧が結果区画に表示され、各オブジェクトの検索名とタイプ ( および送信先管理対象オブジェクトの場合は送信先名 ) が示されます。

2. 結果区画の管理対象オブジェクトの名前をクリックして、管理対象オブジェクトを選択します。

3. 「Actions ( アクション )」メニューから「Properties ( プロパティー )」を選択します。

   「Add Connection Factory Object ( 接続ファクトリオブジェクトを追加 )」(57 ページの図 2-13) または「Add Destination Object ( 送信先オブジェクトを追加 )」(59 ページの図 2-14) ダイアログに類似した、「Connection Factory Object Properties ( 接続ファクトリオブジェクトのプロパティー )」または「Destination Object Properties ( 送信先オブジェクトのプロパティー )」ダイアログボックスが表示されます。このダイアログボックスを使用して、選択したオブジェクトの設定属性を変更できます。ただし、オブジェクトの検索名は変更できません。検索名を変更するには、オブジェクトを削除してから、目的の検索名で新しい管理対象オブジェクトを作成する方法しかありません。

4. 「OK」をクリックして、新しい属性値を受け入れ、ダイアログボックスを閉じます。

## 管理対象オブジェクトの削除

管理対象オブジェクトを削除すると、そのオブジェクトが属するオブジェクトストアから完全に削除されます。

### ➤ 管理対象オブジェクトを削除する

1. 管理コンソールウィンドウのナビゲーション区画のオブジェクトストア名の下の「Connection Factories ( 接続ファクトリ )」または「Destinations ( 送信先 )」を選択します。

   利用できる接続ファクトリまたは送信先管理対象オブジェクトの一覧が結果区画に表示され、各オブジェクトの検索名とタイプ ( および送信先管理対象オブジェクトの場合は送信先名 ) が示されます。

2. 結果区画の管理対象オブジェクトの名前をクリックして、管理対象オブジェクトを選択します。

3. 「Edit ( 編集 )」メニューから「Delete ( 削除 )」を選択します。

   確認のダイアログボックスが表示され、処理を続けることを確認するように求められます。

4. 「Yes ( はい )」をクリックして処理を確認し、確認ダイアログを閉じます。

   この演習では、前に作成した管理対象オブジェクト `MyQueue` または `MyQueueConnectionFactory` は削除しません。「No ( いいえ )」をクリックし、削除操作を実行せずに確認のダイアログを閉じてください。

# サンプルアプリケーションを実行する

このチュートリアルでは、サンプルアプリケーション `HelloWorldMessageJNDI` が提供されます。このアプリケーションは、作成した物理的送信先と管理対象オブジェクトを使用します。

- `MyQueueDest` という名前のキューの物理的送信先
- JNDI の検索名が `MyQueueConnectionFactory` であるキューの接続ファクトリ管理対象オブジェクト
- JNDI の検索名が `MyQueue` であるキューの管理対象オブジェクト

コードではキューの送信者と受信者が作成され、「`Hello World`」メッセージが送受信されます。

アプリケーションを実行する前に、ソースファイルの `HelloWorldMessageJNDI.java` を開き、ソース全体に目を通してください。プログラムは短いながら詳細に説明されているため、仕組みを理解するのはそれほど難しくありません。

## ➤ サンプルアプリケーションを実行する

1. 使用しているプラットフォームによって、次のいずれかのコマンドを使用し、HelloWorldmessageJNDI アプリケーションが格納されているディレクトリを現在のディレクトリにします。

   - Solaris の場合 :

     ```
     cd /usr/demo/imq/helloworld/helloworldmessagejndi
     ```

   - Linux の場合 :

     ```
     cd /opt/sun/mq/examples/helloworld/helloworldmessagejndi
     ```

   - Windows の場合 :

     ```
     cd IMQ_HOME¥demo¥helloworld¥helloworldmessagejndi
     ```

   HelloWorldMessageJNDI.class ファイルが存在していることを確認してください。このアプリケーションを変更している場合は、『Message Queue Developer's Guide for Java Clients』に説明されているクライアントアプリケーションのコンパイルの手順を実行して、再コンパイルする必要があります。

2. CLASSPATH 変数に HelloWorldMessageJNDI.class ファイルと Message Queue 製品に組み込まれた次の .jar ファイルを含む現在のディレクトリを追加します。

   ```
   jms.jar
   imq.jar
   jndi.jar
   fscontext.jar
   ```

   CLASSPATH 変数の設定方法については、『Message Queue Developer's Guide for Java Clients』を参照してください。

---

**注** ファイル jndi.jar は JDK 1.4 にバンドルされています。それより前のバージョンの JDK を使用していないかぎり、CLASSPATH にこのファイルを追加する必要はありません。

---

3. 使用しているプラットフォームによって、次のいずれかのコマンドを実行し、HelloWorldMessageJNDI アプリケーションを実行します。

   - Solaris または Linux の場合 :

     ```
     % java HelloWorldMessageJNDI file:///tmp
     ```

   - Windows の場合 :

     ```
     java HelloWorldMessageJNDI
     ```

   アプリケーションが問題なく実行されると、コード例 2-1 に示す出力が表示されるはずです。

**コード例 2-1** サンプルアプリケーションからの出力

```
java HelloWorldMessageJNDI
Using file:///C:/Temp for Context.PROVIDER_URL


Looking up Queue Connection Factory object with lookup
name:MyQueueConnectionFactory
Queue Connection Factory object found.
Looking up Queue object with lookup name:MyQueue
Queue object found.


Creating connection to broker.
Connection to broker created.

Publishing a message to Queue: MyQueueDest
Received the following message:Hello World
```

第 II 部

# 管理タスク

第 3 章

# ブローカとクライアントの起動

Sun Java System Message Queue をインストールし、いくつかの準備手順を実行した後、ブローカとクライアントを起動できます。ブローカの設定は、設定ファイルセットと、ブローカユーティリティー (imqbrokerd) に渡されるコマンド行オプションによって決まります。詳細については、第 4 章「ブローカの設定」を参照してください。

この章では、次の節について説明します。



## システムリソースの準備

ブローカを起動する前に、2 つの予備的なシステムレベルのタスク、すなわちシステムクロックの同期と、Solaris または Linux プラットフォームの場合のファイル記述子の制限の設定を行います。次の節では、これらのタスクについて説明します。

### システムクロックの同期

ブローカまたはクライアントを起動する前に、Message Queue システムと対話するすべてのホスト上のクロックを同期する必要があります。メッセージの有効期限 ( 生存時間 ) を使用する場合には、同期は特に重要です。同期されていないクロックのタイムスタンプは、メッセージの有効期限が予想どおりに機能するのを阻害し、メッセージの配信を阻害します。同期はブローカクラスタにとっても重要です。

システムを設定し、Simple Network Time Protocol (SNTP) などの時間同期プロトコルを実行するようにします。時間同期は一般に、Solaris と Linux の場合は xntpd デーモンで、Windows の場合は W32Time サービスでサポートされます。このサービスの設定に関する詳細は、オペレーティングシステムのマニュアルを参照してください。ブローカを実行した後、システムクロックが逆戻り設定されるのを防いでください。

## ファイル記述子制限の設定

Solaris および Linux プラットフォームでは、クライアントやブローカが実行されるシェルによって、プロセスで使用できるファイル記述子の数に対する弱い制限値があります。Message Queue では、クライアントが行う接続、あるいはブローカが受け付ける接続はすべて、これらのファイル記述子のどれかを使用します。持続メッセージを持つ物理的な送信先もすべて、ファイル記述子を使用します。

その結果、ファイル記述子の制限によって、ブローカまたはクライアントの接続数が制限されます。デフォルトで最大の接続数は Solaris で 256、Linux で 1024 です。実際には、持続性のためにファイル記述子を使用することから、接続数の制限はこの値より小さくなります。これ以上の接続を必要とする場合は、クライアントまたはブローカが実行する各シェルのファイル記述子制限を拡大する必要があります。この方法については、ulimit マニュアルページを参照してください。

# ブローカの起動

Message Queue コマンド行ユーティリティーまたは Windows の「スタート」メニューを使用して、ブローカをインタラクティブに起動できます。または、システムの起動時に自動的に起動するように調整できます。次の節で、この方法について説明します。

## ブローカのインタラクティブな起動

ブローカユーティリティー (imqbrokerd) を使用すると、コマンド行からブローカをインタラクティブに起動できます または、Windows の場合は「スタート」メニューからブローカを起動できます。ブローカの起動に管理コンソール (imqadmin) や コマンドユーティリティー (imqcmd) を使用できません。これらのツールを使用する前に、ブローカが実行されている必要があります。

Solaris と Linux プラットフォームでは、ブローカインスタンスは必ずブローカを最初に起動したユーザーと同一のユーザーが起動します。各ブローカインスタンスは、固有の設定プロファイルとファイルベースのメッセージストアを保有します。ブローカインスタンスを最初に起動する場合、Message Queue はユーザーのファイル作成モードマスク (umask) を使用して、ブローカインスタンスの設定情報と持続データを格納するディレクトリに、アクセス権を設定します。

ブローカインスタンスには、デフォルトでインスタンス名 imqbroker が割り当てられます。この名前とデフォルト設定を使用してコマンド行からブローカを起動するには、次のコマンドを使用します。

```
imqbrokerd
```

このコマンドにより、デフォルトポート 7676 のポートマッパーを持つローカルマシン上にある、ブローカのインスタンス (imqbroker) が起動されます (79 ページの「ポートマッパー」を参照 )。

デフォルト以外のインスタンス名を指定する場合は、imqbrokerd コマンドに -name オプションを使用します。次のコマンドは、インスタンス名 myBroker を持つブローカを起動します。

```
imqbrokerd -name myBroker
```

imqbrokerd コマンド行では、ブローカの操作のさまざまな面を制御するその他のオプションも使用できます。次の例では、-tty オプションを使用してコマンドウィンドウにエラーと警告を送信します ( 標準出力 )。

```
imqbrokerd -name myBroker -tty
```

コマンド行で -D オプションを使用しても、ブローカのインスタンス設定ファイル (config.properties) で指定されたプロパティーの値を上書きすることができます。この例では、imq.jms.max_threads プロパティーを設定して、jms 接続サービスが利用できる最大スレッド数を 2000 に上げています。

```
imqbrokerd -name myBroker -Dimq.jms.max_threads=2000
```

imqbrokerd コマンドの構文、サブコマンド、オプションの詳細は、272 ページの「ブローカユーティリティー」を参照してください。この情報の簡単な概要については、次のコマンドで確認します。

```
imqbrokerd -help
```

**注** Sun Java System Message Queue Platform Edition ライセンスを保有している場合は、imqbrokerd コマンドの -license オプションを使用して、Enterprise Edition の試用ライセンスをアクティブにして、Enterprise Edition の機能を 90 日間試用できます。ライセンス名に try を指定します。

```
imqbrokerd -license try
```

ブローカを起動するたびにこのオプションを使用する必要があります。使用しない場合、デフォルトで Platform Edition の標準ライセンスに戻ります。

# ブローカの自動起動

コマンド行からブローカを明示的に起動する代わりに、システムの起動時に自動的にブローカが起動するように設定できます。この方法は、ブローカを実行するプラットフォーム (Solaris、Linux、または Windows) により異なります。

## Solaris と Linux での自動起動

Solaris と Linux システムの場合、自動起動を有効にするスクリプトを Message Queue のインストール時に /etc/rc* ディレクトリツリーに配置します。このスクリプトの使用を有効にする場合、設定ファイル /etc/imq/imqbrokerd.conf (Solaris) または /etc/opt/sun/mq/imqbrokerd.conf (Linux) を次のように編集します。

- システムの起動時に自動的にブローカが起動するには、AUTOSTART プロパティーを YES に設定します。
- 異常終了の後、ブローカを自動的に再起動するには、RESTART プロパティーを YES に設定します。
- ブローカの起動コマンド行引数を設定するには、ARGS プロパティーに 1 つ以上の値を指定します。

## Windows での自動起動

Windows システムの起動時にブローカを自動的に起動するには、ブローカを Windows サービスとして定義する必要があります。そうすることで、ブローカはシステムの起動時に起動し、システムがシャットダウンされるまで、バックグランドで実行します。したがって、別のインスタンスを起動する必要がないかぎり、ブローカを起動するのに imqbrokerd コマンドを使用することはありません。

システムで Windows サービスとして実行できるブローカは 1 つのみです。タスクマネージャーには、そうしたブローカが 2 つの実行可能プロセスとして表示されます。

- Windows のネイティブサービスラッパー、imqbrokersvc.exe
- ブローカを実行中の Java ランタイム

Windows システムで Message Queue をインストールしている場合、ブローカをサービスとしてインストールできます。インストール後、サービス管理ユーティリティー imqsvcadmin を使用して、次の操作を実行します。

- Windows のサービスとしてブローカを追加
- ブローカサービスの起動オプションを決定
- Windows サービスとして実行中のブローカを削除

ブローカに起動オプションを渡すには、imqsvcadmin コマンドに -args 引数を使用します。これは 68 ページの「ブローカの起動」で説明するように、imqbrokerd コマンドの -D オプションと同じように機能します。ブローカの動作を通常どおり制御するには、コマンドユーティリティー (imqcmd) を使用します。

imqsvcadmin コマンドの構文、サブコマンド、オプションの詳細は、293 ページの「サービス管理ユーティリティー」を参照してください。

### *ブローカサービスの再設定*

Windows サービスとしてインストールしたブローカを再設定する手順は次のとおりです。

#### ➤ Windows サービスとして実行中のブローカを再設定する

1. サービスを停止します。
   a. Windows の「スタート」メニューのサブメニュー「設定」から、「コントロール パネル」を選択します。
   b. 「管理ツール」コントロールパネルを開きます。
   c. 「サービス」ツールのアイコンを選択し、「ファイル」メニューから「開く」、またはポップアップコンテキストメニューから選択するか、単にアイコンをダブルクリックして、サービスツールを実行します。
   d. 「サービス ( ローカル )」の下の「Message Queue Broker」サービスを選択し、「操作」メニューから「プロパティ」を選択します。

      または、「Message Queue Broker」を右クリックし、ポップアップコンテキストメニューから「プロパティ」を選択するか、単に「Message Queue Broker」をダブルクリックします。どちらの場合も「Message Queue Broker のプロパティ」ダイアログボックスが表示されます。
   e. 「Message Queue Broker のプロパティ」ダイアログの「全般」タブで、「停止」をクリックして、ブローカサービスを停止します。
2. サービスを削除します。

コマンド行で、次のコマンドを入力します。

```
imqsvcadmin remove
```

3. サービスを再インストールし、異なるブローカ起動オプション -args、または異なる Java バージョン引数、-vmargs オプションを指定します。

   たとえば、サービスのホスト名とポート番号を broker1 と 7878 に変更する場合、次のコマンドを使用します。

```
imqsvcadmin install -args "-name broker1 -port 7878"
```

### *代替 Java ランタイムの使用*

代替の Java ランタイムの場所を指定する場合、imqsvcadmin コマンドの -javahome オプション、または -jrehome オプションのどちらかを使用することができます。これらのオプションは、サービスの「プロパティ」ダイアログウィンドウの「全般」タブの「開始パラメータ」フィールドに指定することもできます。

---

**注** 「開始パラメータ」フィールドでは、円記号 (¥) がエスケープ文字として処理されるため、パスの区切り文字として使用する場合、次のように円記号を 2 つ入力してください。

```
-javahome c:¥¥j2sdk1.4.0
```

---

### *ブローカサービス起動オプションの表示*

ブローカサービスの起動オプションを指定するには、コード例 3-1 に示すように、imqsvcadmin コマンドの query オプションを使用します。

**コード例 3-1** ブローカサービス起動オプションの表示

```
imqsvcadmin query

Service Message Queue Broker is installed.
Display Name: Message Queue Broker
Start Type: Automatic
Binary location: C:¥Sun¥MessageQueue¥bin¥imqbrokersvc.exe
JavaHome: c:¥j2sdk1.4.0
Broker Args: -name broker1 -port 7878
```

### *サービス開始時の問題のトラブルシューティング*

ブローカを Windows サービスとして開始しようとしたときにエラーが発生する場合、記録されているエラーイベントを確認できます。

➤ **記録されているサービスのエラーイベントを表示する**

1. Windows の「管理ツール」コントロールパネルを開きます。
2. 「イベントビューア」ツールを起動します。
3. 「アプリケーション」イベントログを選択します。
4. 「操作」メニューから「最新の情報に更新」を選択して、エラーイベントを表示します。

# ブローカの削除

ブローカを削除する手順もプラットフォームによって異なります。次の節で説明します。

## Solaris または Linux でのブローカの削除

Solaris または Linux プラットフォームでのブローカインスタンスを削除するには、`imqbrokerd` コマンドと `-remove` オプションを使用します。このコマンドの形式は次のようになります。

```
imqbrokerd [options...] -remove instance
```

たとえば、ブローカの名前が `myBroker` の場合、コマンドは次のようになります。

```
imqbrokerd -name myBroker -remove instance
```

このコマンドは、指定されたブローカのインスタンスディレクトリ全体を削除します。

システムの起動時に自動起動するようにブローカが設定されている場合、設定ファイル `/etc/imq/imqbrokerd.conf` (Solaris) または `/etc/opt/sun/mq/imqbrokerd.conf` (Linux) を編集し、`AUTOSTART` プロパティーを `NO` に設定します。

`imqbrokerd` コマンドの構文、サブコマンド、オプションの詳細は、272 ページの「ブローカユーティリティー」を参照してください。この情報の簡単な概要については、次のコマンドで確認します。

## Windows ブローカサービスの削除

Windows サービスとして実行中のブローカを削除するには、次のコマンドを使用して、

```
imqcmd shutdown bkr
```

ブローカをシャットダウンし、続けて次のコマンドを使用して、

```
imqsvcadmin remove
```

サービスを削除します。

または、管理ツールコントロールパネルから到達できる Windows サービスツールを使用して、ブローカサービスを停止し、削除することもできます。

ブローカサービスを削除したら、コンピュータを再起動します。

# クライアントの起動

クライアントアプリケーションを起動する前に、アプリケーション開発者からシステムの設定方法に関する情報を入手します。Java クライアントアプリケーションを起動する場合、CLASSPATH 変数を適切に設定し、正しい .jar ファイルがインストールされていることを確認します。システムの設定の一般的な手順については、『Message Queue Developer's Guide for Java Clients』で説明していますが、開発者が追加情報を提供する場合があります。

Java クライアントアプリケーションを起動するには、次のコマンド行形式を使用します。

```
java clientAppName
```

C クライアントアプリケーションを起動するには、アプリケーション開発者が提供した形式を使用します。

アプリケーションのマニュアルには、アプリケーションで設定される属性値に関する情報が提供されています。これらの属性値をコマンド行からオーバーライドできます。また、JNDI (Java Naming and Directory Interface) 検索により接続ファクトリを検索する Java クライアントに対して、コマンド行で属性を指定することもできます。検索でアプリケーションよりも古い接続ファクトリが戻される場合、その接続ファクトリは最新の属性をサポートしない可能性があります。そのような場合、Message Queue はこれらの属性にデフォルト値を設定します。必要に応じて、コマンド行を使用して、これらのデフォルト値をオーバーライドできます。

コマンド行から、Java アプリケーションの属性値を指定するには、次の構文を使用します。

```
java [[-Dattribute=value]...] clientAppName
```

*attribute* の値は、第 16 章「管理対象オブジェクト属性のリファレンス」で説明するように、接続ファクトリの管理対象オブジェクトの属性になる必要があります。値にスペースが入る場合は、コマンド行の *attribute=value* 部分を引用符で囲みます。

次の例は、`MyMQClient` というクライアントアプリケーションを起動し、ホスト `OtherHost` のポート `7677` のブローカに接続します。

```
java -DimqAddressList=mq://OherHost:7677/jms MyMQClient
```

コマンド行で指定したホスト名およびポートによって、アプリケーション自体で設定された属性値がオーバーライドされます。

場合によっては、コマンド行で属性値を指定できません。管理者は読み取りアクセス専用を許可するように管理対象オブジェクトを設定できます。または、アプリケーション開発者が、クライアントアプリケーションに読み取り専用を許可するようにコーディングできます。アプリケーション開発者との通信は、クライアントプログラムの起動の最適な方法を理解するのに必要です。

第 4 章

# ブローカの設定

ブローカの設定は、一連の設定ファイルおよび起動時に `imqbrokerd` コマンドに渡されるオプションにより制御されます。この章では、使用可能な設定プロパティーと、それらを使用してブローカを設定する方法について説明します。

この章では、次の節について説明します。



ブローカ設定プロパティーの詳細は、第 14 章「ブローカのプロパティーのリファレンス」を参照してください。

## ブローカサービス

ブローカ設定プロパティーは、影響を受けるサービスやブローカコンポーネントに応じて、いくつかのカテゴリに分類できます。

- **接続サービス**はブローカとそのクライアント間の物理的な接続を管理し、送受信メッセージの転送を行います。
- **ルーティングサービス**は JMS ペイロードメッセージをルーティングし、配信するほか、メッセージサービスによって使用されるメッセージを制御し、信頼性の高い配信をサポートします。
- **持続サービス**は持続ストレージへのデータの書き込みと、持続ストレージからのデータの取得を管理します。
- **セキュリティーサービス**はブローカに接続するユーザーを認証し、ユーザーの操作を承認します。
- **監視サービス**はブローカのパフォーマンスに関するメトリックスと診断情報を生成します。

次の節では、これらの各サービスおよび、特定のニーズに合わせてサービスをカスタマイズするために使用できるプロパティーについて説明します。

# 接続サービス

メッセージブローカは、さまざまなトランスポートプロトコルを使用して、アプリケーションと管理クライアントの両方をサポートするさまざまな接続サービスを提供できます。接続サービスに関連するブローカ設定プロパティーについては、295 ページの「接続のプロパティー」に示しています。

表 4-1 に使用できる接続サービスを示します。それらは次の 2 つの特性によって識別されます。

- *service type* は、JMS メッセージ配信サービス (NORMAL)、または Message Queue 管理サービス (ADMIN) のどちらを提供するのかを指定します。
- プロトコルタイプは基礎となるトランスポートプロトコルを指定します。

**表 4-1** Message Queue 接続サービス

| サービス名 | サービスタイプ | プロトコルタイプ |
|---|---|---|
| jms | NORMAL | TCP |
| ssljms (Enterprise Edition) | NORMAL | TLS (SSL ベースセキュリティー ) |
| httpjms (Enterprise Edition) | NORMAL | HTTP |
| httpsjms (Enterprise Edition) | NORMAL | HTTPS (SSL ベースセキュリティー ) |
| admin | ADMIN | TCP |
| ssladmin | ADMIN | TLS (SSL ベースセキュリティー ) |

ブローカの imq.service.activelist プロパティーを設定することで、これらの接続サービスの一部、またはすべてを実行することができます。このプロパティーの値は、ブローカの起動時にアクティブにする接続サービスのリストであり、このプロパティーを明示的に指定しないと、jms および admin サービスがデフォルトでアクティブにされます。

各接続サービスは、特定の認証および承認機能もサポートします。詳細については 86 ページの「セキュリティーサービス」を参照してください。

## ポートマッパー

各接続サービスは、ホスト名 ( または IP アドレス ) とポート番号によって指定された特定のポートで使用できます。サービスには、静的なポートを明示的に指定するか、またはブローカのポートマッパーによって動的にポートを割り当てることができます。ポートマッパー自体は、通常、標準ポート番号 7676 にあるブローカのプライマリポートに常駐します。必要に応じて、ブローカの設定プロパティー `imq.portmapper.port` を使用して、このポートを別のポート番号でオーバーライドできます。デフォルトで、各接続サービスは、起動時に自身をポートマッパーに登録します。クライアントがブローカとの接続を作成すると、Message Queue クライアントランタイムはまずポートマッパーに接続し、目的の接続サービスのポート番号を要求します。

または、ポートマッパーをオーバーライドし、`imq.`*`serviceName`*`.`*`protocolType`*`.port` 設定プロパティー ( この場合 *serviceName* と *protocolType* は表 4-1 に示すように、特定の接続サービスを示す ) を使用して、接続サービスに静的ポート番号を明示的に割り当てることができます。この方法で設定できるのは、`jms`、`ssljms`、`admin`、および `ssladmin` 接続サービスのみです。`httpjms` および `httpsjms` サービスでは、付録 C 「HTTP/HTTPS のサポート」に説明するように、異なる設定プロパティーが使用されます。ただし、静的ポートは、一般に、ファイアウォールを介した接続を作成する場合などの特定の状況でのみ使用し、一般的な使用にはお勧めしません。

**注** 複数のホストを使用できる場合 ( コンピュータに複数のネットワークカードがインストールされている場合など )、ブローカプロパティーを使用して、接続サービスのバインド先にするホストを指定できます。`imq.hostname` プロパティーは、すべての接続 サービス用の単一のデフォルトのホストを指定し、必要に応じて、`imq.`*`serviceName`*`.`*`protocolType`*`.hostname` (`jms`、`ssljms`、`admin`、または `ssl` 管理サービス ) または `imq.portmapper.hostname` ( ポートマッパー自体 ) で、オーバーライドできます。

複数のポートマネージャー要求が同時に受け取られた場合、それらはアクションの待機中に、オペレーティングシステムのバックログに保存されます。`imq.portmapper.backlog` プロパティーは、そうしたバックログされた要求の最大数を指定します。この制限を超えると、バックログが縮小するまで、その後の要求が拒否されます。

## スレッドプール管理

各接続サービスは、複数の接続をサポートするマルチスレッドです。これらの接続に必要なスレッドは、ブローカによって、サービスごとに個別のスレッドプールに保存されます。接続にスレッドが必要なときは、その接続をサポートするサービスのスレッドプールにスレッドが追加されます。

選択するスレッドモデルは、スレッドが 1 つの接続専用であるか、複数の接続で共有するかどうかを指定します。

- 専用モデルでは、ブローカへの接続ごとに受信メッセージ用と出力メッセージ用の 2 つのスレッドが必要です。これによって、サポート可能な接続数が制限されますが、パフォーマンスは向上します。
- 共有モデルでは、メッセージが送信または受信されると、接続が共有スレッドによって処理されます。接続ごとに専用スレッドが必要ないため、このモデルでは使用可能な接続数が増加しますが、それと引き換えに、スレッド管理に追加のオーバーヘッドが必要なため、パフォーマンスが低下します。

ブローカの `imq.`*`serviceName`*`.threadpool_model` プロパティーは、特定の接続サービスに 2 つのうちどちらのモデルを使用するかを指定します。このプロパティーは、`dedicated` または `shared` の文字列値のどちらかになります。プロパティーを明示的に設定しない場合は、デフォルトで `dedicated` が使用されます。

ブローカプロパティー `imq.`*`serviceName`*`.min_threads` および `imq.`*`serviceName`*`.max_threads` を設定して、サービスのスレッドプール内のスレッドの最小数と最大数を指定することもできます。使用可能なスレッド数が、指定された最小のしきい値より少なくなると、Message Queue は、再び最小値に達するまで、スレッドをシャットダウンして、スレッドを解放します。これによって、メモリーのリソースが節約されます。負荷が重い場合、スレッドの数がプールの最大数まで増加する可能性があります。最大数に達すると、スレッドが利用できるようになるまで、新しい接続は拒否されます。

共有スレッドモデルでは、ディストリビュータスレッドを使用して、スレッドをアクティブな接続に割り当てます。ブローカプロパティー `imq.shared.connectionMonitor_limit` は、1 つのディストリビュータスレッドで監視できる接続の最大数を指定します。このプロパティーの値を小さくするほど、接続にスレッドを割り当てる速度が向上します。`imq.ping.interval` プロパティーは、時間間隔を秒単位で指定します。ブローカが定期的に接続をテスト ("ping") して、接続がまだアクティブであるか確認することによって、メッセージ送信に失敗する前に、事前に接続の障害を検出できます。

# ルーティングサービス

クライアントがブローカに接続したら、メッセージのルーティングおよび配信を処理できるようになります。この段階では、ブローカはさまざまな種類の物理的送信先の作成および管理を担当し、メッセージのスムーズなフローを確保し、リソースを効率的に使用します。298 ページの「ルーティングのプロパティー」で説明するブローカ設定プロパティーを使用して、それぞれのアプリケーションのニーズに合わせて、これらのタスクを管理できます。

ブローカのパフォーマンスと安定性は、使用できるメモリーなどのシステムリソースとリソースの使用効率によって異なります。設定プロパティーを設定して、受信メッセージによるブローカの過負荷やメモリー不足を防止することができます。これらのプロパティーは、リソースが不十分になってもメッセージサービスの動作を維持できるように 3 つのレベルで機能します。

- **システム全体のメッセージ制限**は、システム上のすべての物理的送信先にまとめて適用されます。これらには、ブローカが保持するメッセージの最大数 (`imq.system.max_count`) やそれらのメッセージが占有する最大合計バイト数 (`imq.system.max_size`) などがあります。これらのいずれかの制限に達した場合、ブローカは、保留メッセージが制限以下になるまで、新しいメッセージを拒否します。各メッセージの最大サイズ (`imq.message.max_size`) や期限切れメッセージの再利用の間隔 (`imq.message.expiration.interval`) にも制限があります。
- **個々の送信先の制限**は特定の物理的送信先へのメッセージを制限します。これらの制限を制御する設定プロパティーについては、第 15 章「物理的送信先のプロパティーのリファレンス」で説明しています。たとえば、送信先で保持するメッセージの数とサイズ、作成可能なメッセージのプロデューサとコンシューマの数、送信先にバッチ配信できるメッセージの数の制限などがあります。

  送信先は、メッセージプロデューサによるメッセージの配信速度を遅くするか、新しい受信メッセージを拒否するか、もっとも古いか優先度がもっとも低い既存のメッセージを破棄するかによって、メモリー制限に対処するように設定できます。この方法で送信先から削除されるメッセージは、完全に削除するのではなく、オプションでデッドメッセージキューに移動できます。ブローカプロパティー `imq.destination.DMQ.truncateBody` が、デッドメッセージキューにメッセージ本体全体を保存するのか、ヘッダーとプロパティーデータのみを保存するのかを制御します。

  アプリケーションの開発とテスト時の利便性のため、メッセージプロデューサまたはコンシューマが存在しない送信先にアクセスしようとした場合に、新しい物理的送信先を自動的に作成するように、メッセージブローカを設定できます。299 ページの表 14-3 にまとめられているブローカプロパティーは、ここで説明したプロパティーとほとんど同じですが、管理者によって作成される送信先ではなく、自動的に作成される送信先に適用されます。
- **システムメモリーのしきい値**は、ブローカがメモリーの過負荷を防ぐために段階的に重大なアクションをとるメモリー使用率のレベルを定義します。それらの利用率のレベルが 4 つ定義されています。

- Green: 使用可能なメモリーが十分にあります。
- Yellow: ブローカメモリーが不足し始めています。
- Orange: ブローカのメモリーが不十分です。
- Red: ブローカのメモリーが不足しています。

これらのレベルを定義するメモリー利用率は、ブローカプロパティー `imq.green.threshold`、`imq.yellow.threshold`、`imq.orange.threshold`、および `imq.red.threshold` でそれぞれ指定します。green のデフォルト値は 0%、yellow のデフォルト値は 80%、orange のデフォルト値は 90%、red のデフォルト値は 98% です。

メモリー利用率が次のレベルに進むと、ブローカは漸進的に対応します。まず、メッセージをアクティブなメモリーから持続ストレージにスワップし、次に、持続的でないメッセージのプロデューサの処理速度を低下させ、最終的にブローカへのメッセージフローを止めます。これらのアクションはどちらもブローカのパフォーマンスを低下させます。メッセージの生成を徐々に減らすには、配信される各バッチのサイズを、プロパティー `imq.`*resourceState*`.count` によって指定されたメッセージ数に制限します。ここで、*resourceState* はそれぞれ `green`、`yellow`、`orange`、または `red` です。

**注** こうしたシステムメモリーのしきい値がトリガーされることは、システム全体および送信先のメッセージ制限の設定が高すぎることを示しています。メモリーしきい値によって、潜在するメモリーの過負荷を必ずしもタイミングよく検出できるとは限らないため、メモリー利用率を制御するためにそれらに依存することは避け、メモリーリソースを最大限に活用できるように、システム全体および送信先の制限を再設定する必要があります。

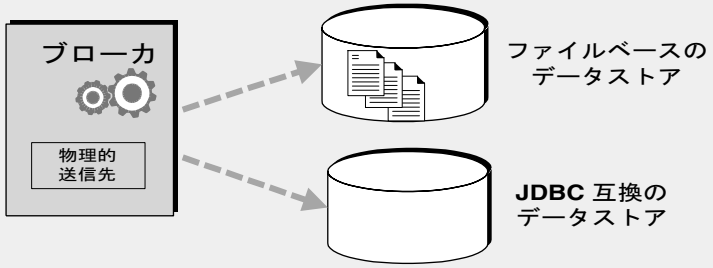
# 持続サービス

障害が発生したブローカを復元するには、メッセージの配信処理の状態を作成し直す必要があります。この操作を行うには、ブローカが持続データストアに状態情報を保存する必要があります。ブローカの再起動時、ブローカは保存されたデータを使用して、送信先と永続サブスクリプションを再作成し、持続メッセージを復元して、開いているトランザクションをロールバックし、配信されていないメッセージのルーティングテーブルを再構築します。この後に、メッセージの配信を再開します。

Message Queue は、ファイルベースの持続モジュールと JDBC ベースの持続モジュールの両方をサポートしています ( 図 4-1 を参照 )。ファイルベースの持続は、持続データを保存するために個別のファイルを使用し、JDBC ベースの持続では、JDBC™ (Java Database Connectivity) インタフェースを使用し、JDBC 互換のデータストアにブロー

カを接続します。ファイルベースの持続は一般に JDBC ベースより高速ですが、JDBC 互換ストアによって実現される冗長性や管理者による制御を好むユーザーもいます。ブローカ設定プロパティー `imq.persist.store` (303 ページの表 14-4 を参照 ) は 2 つの持続形式のどちらを使用するかを指定します。

**図 4-1**　持続データストレージ




## ファイルベースの持続

デフォルトで、Message Queue はファイルベースの持続データストアを使用します。これは、メッセージ、送信先、永続サブスクリプション、およびトランザクションなどの持続データを個別のファイルに保存します。ファイルベースの持続に関連するブローカ設定プロパティーについては、304 ページの「ファイルベースの持続」に示しています。

ファイルベースのストアは、そのデータストアが属するブローカインスタンスの名前 (*instanceName*) によって識別されるディレクトリに配置されます。

*.../*`instances/`*instanceName*`/fs350/`

`instances` ディレクトリの場所については、付録 A 「プラットフォームごとの Message Queue データの場所」を参照してください。ブローカの各送信先には、その送信先に配信されるメッセージを保持する個別のサブディレクトリがあります。

**注**　持続データストアには機密事項を扱うメッセージや財産的価値のあるメッセージが含まれることがあるため、*.../*`instances/`*instanceName*`/fs350/` ディレクトリは承認されていないアクセスから保護するようにしてください。97 ページの「持続データの保護」を参照してください。

送信先、永続サブスクリプション、トランザクション状態情報などのメッセージ以外の持続データは、すべて個別のファイルに格納されます。大半のメッセージは、可変長レコードから構成されるシングルファイルに格納されます。可変長レコードファイルを圧縮し、メッセージが追加および削除されたときの断片化を緩和することができ

ます (128 ページの「物理的送信先の圧縮」を参照 )。さらに、特定のしきい値を超えるサイズのメッセージは、可変長レコードファイルではなく、個別のファイルに格納されます。このしきい値のサイズは、ブローカプロパティー imq.persist.file.message.max_record_size で設定できます。

ブローカは、これらの各メッセージファイル用のファイルプールを維持します。ファイルは必要なくなった場合にも削除されず、送信先ディレクトリの空きファイルのプールに戻されるため、あとで別のメッセージに再利用できます。ブローカプロパティー imq.persist.file.destination.message.filepool.limit はプール内のファイルの最大数を指定します。送信先の各メッセージファイル数がこの制限を超えた場合、ファイルが不要になると、プールに戻されずに削除されます。

ファイルをファイルプールに戻すと、ブローカは前の内容を削除せずに、再利用可能なファイルとしてタグ付けすることにより、保存領域と引き換えに時間を節約できます。imq.persist.file.message.filepool.cleanratio ブローカプロパティーを使用して、単に再利用可能とマークするだけでなく、「クリーン」( 空き ) 状態で維持する必要がある各送信先のファイルプール内のファイルのパーセンテージを指定します。この値を大きくするほど、ファイルプールに必要な領域は少なくなりますが、ファイルをプールに戻す際にファイルの内容を空にするために必要なオーバーヘッドが大きくなります。ブローカの imq.persist.file.message.cleanup プロパティーが true の場合、ブローカのシャットダウン時にプール内のすべてのファイルが空にされ、クリーン状態のままにされるため、保存領域は節約できますが、シャットダウンの処理速度が遅くなります。

持続ストアへのデータの書き込みにおいて、オペレーティングシステムには、データを同時に書き込むか、またはレイジーに」( 非同期的に ) 書き込むかを選択できます。レイジーストレージでは、システムクラッシュの発生時に、ブローカが持続ストレージにデータが書き込まれていなくても書き込まれたものと考えた場合に、データが損失する可能性があります。パフォーマンスと引き換えに、完全な信頼性を確保するには、ブローカプロパティー imq.persist.file.sync.enabled を true に設定して、すべてのデータを同時に書き込む必要があります。この場合、システムがクラッシュ後に回復した際に、データの利用が保証されるため、ブローカは確実に処理を再開できます。ただし、データは失われませんが、クラスタ構成のブローカでは現在データを共有していないため、クラスタ内の他のブローカからは使用できません。

## JDBC ベースの持続

ファイルベースの持続を使用する代わりに、JDBC 互換ドライバを介してアクセスが可能な任意のデータストアにアクセスするように、ブローカを設定できます。この作業には、該当する JDBC 関連のブローカ設定プロパティーの設定、データベースマネージャーユーティリティー (imqdbmgr) を使用した適切なスキーマでのデータストアの作成が含まれます。仕様については、96 ページの「JDBC ベースのストアの設定」を参照してください。

JDBC データベースを使用するように、ブローカを設定するプロパティーについては、306 ページの「JDBC ベースの持続」を参照してください。これらのプロパティーを指定するには、各ブローカインスタンスのインスタンス設定ファイル (config.properties) か、ブローカユーティリティー (imqbrokerd) またはデータベースマネージャーユーティリティー (imqdbmgr) に -D コマンド ライン オプションを使用します。

imq.persist.jdbc.driver プロパティーは、データベースへの接続に使用する JDBC ドライバの Java クラス名を指定します。既存のデータベースへの接続 (imq.persist.jdbc.opendburl)、新しいデータベースの作成 (imq.persist.jdbc.createdburl)、およびデータベース接続のクローズ (imq.persist.jdbc.closedburl) のための URL を指定するプロパティーもあります。

imq.persist.jdbc.user および imq.persist.jdbc.password プロパティーは、データベースにアクセスするためのユーザー名とパスワードを指定し、imq.persist.jdbc.needpassword はパスワードが必要かどうかを指定するブール型のフラグです。セキュリティー上の理由から、パスワードは、-passfile コマンド行オプションで指定するパスワードファイルにのみ指定する必要があります。そうしたパスワードファイルを指定しないと、imqbrokerd および imqdbmgr コマンドではインタラクティブにパスワードの入力が要求されます。同様に、imqbrokerd コマンドに -dbuser オプション、または imqdbmgr コマンドに -u オプションを使用して、コマンド行からユーザー名を指定できます。

複数のブローカインスタンスによって共有される JDBC データベースでは、設定プロパティー imq.persist.jdbc.brokerid は、データベーステーブル名に追加される各データベースの一意のインスタンス識別子を指定します。このプロパティーは、1 つのブローカインスタンスのデータのみを格納する組み込みデータベースでは、一般に必要ありません。その他の JDBC 関連の設定プロパティーは、データベーススキーマを作成する SQL コードをカスタマイズするために使用し、1 つのプロパティーが 1 つのデータベーステーブルに対応します。たとえば、imq.persist.jdbc.table.IMQSV35 プロパティーはバージョンテーブル、imq.persist.jdbc.table.IMQCCREC35 は設定変更レコードテーブル、imq.persist.jdbc.table.IMQDEST35 は送信先テーブルを作成するための SQL コマンドを指定します。すべてのリストについては、306 ページの表 14-6 を参照してください。

**注** データベースシステムによって、必要とされる正確な SQL 構文が異なるため、詳細についてはデータベースベンダーのマニュアルを参照してください。

# セキュリティーサービス

Message Queue には、ユーザーアクセス制御 ( 認証と承認 ) および暗号化のためのセキュリティーサービスが用意されています。

- 認証は、確認されたユーザーのみがブローカとの接続を確立できるようにします。
- 承認は、リソースにアクセスし、特定の操作を実行する権限を持つユーザーやグループを指定します。
- 暗号化は、接続によるメッセージの配信中の書き換えを防止します。

Message Queue 管理者は、ユーザーを認証し、ユーザーの操作を承認するために必要な情報をブローカに設定する責任があります。セキュリティーサービスに属するブローカプロパティーを 309 ページの「セキュリティーのプロパティー」に示します。ブール型のプロパティー `imq.accesscontrol.enabled` はアクセス制御をブローカ全体に適用するかどうかを制御するマスタースイッチとして機能しますが、厳密に制御する場合は、`imq.`*serviceName*`.accesscontrol.enabled` プロパティーを設定することによって、特定の接続サービスのこの設定をオーバーライドができます。ここで *serviceName* は 78 ページの表 4-1 に示す接続サービスの名前で、たとえば `imq.httpjms.accesscontrol.enabled` などになります。

図 4-2 に、ブローカが認証サービスおよび承認サービスを提供するために必要なコンポーネントを示します。これらのサービスは、メッセージングシステムのユーザーに関する情報 ( 名前、パスワード、グループメンバーシップ ) を格納するユーザーリポジトリを使用します。さらに、ユーザーまたはグループの特定の操作を承認するため、ブローカは、ユーザーやグループが実行できる操作を指定したアクセス制御プロパティーファイルを参照します。ブローカ全体で 1 つのアクセス制御プロパティーファイルを指定する場合は、設定プロパティー `imq.accesscontrol.file.filename` を使用し、1 つの接続サービスのアクセス制御プロパティーファイルを指定する場合は、`imq.`*serviceName*`.accesscontrol.file.filename` を使用します。

図 4-2　セキュリティーのサポート

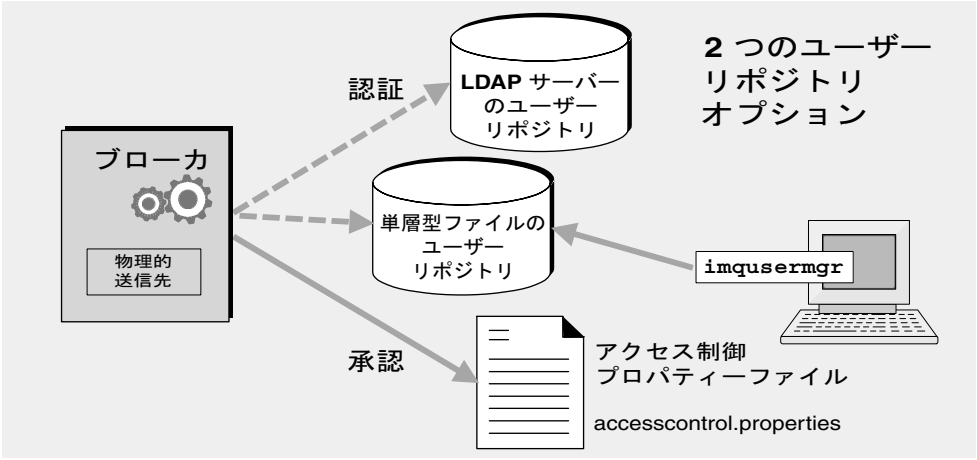



図 4-2 に示すように、Message Queue サービスによって提供された単層型ファイルユーザーリポジトリにユーザーデータを格納するか、または、既存の LDAP (Lightweight Directory Access Protocol) リポジトリに接続できます。

- 単層型ファイルリポジトリを選択した場合は、Message Queue ユーザーマネージャーユーティリティー (`imqusermgr`) を使用して、リポジトリを管理する必要があります。このオプションは組み込みであるため、簡単に使用できます。
- 既存の LDAP サーバーを使用する場合は、LDAP ベンダーから提供されているツールを使用して、ユーザーリポジトリを設定、管理します。さらに、ブローカがユーザーとグループに関する情報について LDAP サーバーをクエリーできるようにするため、ブローカのインスタンス設定ファイルにもプロパティーを設定する必要があります。

ブローカの `imq.authentication.basic.user_repository` プロパティーは、どちらの種類のリポジトリを使用するかを指定します。一般に、拡張性が重要な場合、またはブローカクラスタを使用するなど、さまざまなブローカでリポジトリを共有する必要がある場合は、LDAP リポジトリをお勧めします。単層型ファイルリポジトリまたは LDAP ユーザーリポジトリの設定の詳細については、135 ページの「ユーザーの認証」を参照してください。

## 認証

ブローカへの接続を要求するクライアントは、ユーザー名とパスワードを入力する必要があります。ブローカはそれらをユーザーリポジトリに保存されているユーザー名とパスワードと比較します。クライアントからブローカに送信されるパスワードは、Base-64 ( 単層型ファイルリポジトリの場合 ) か、メッセージダイジェスト (MD5) ハッシュ (LDAP リポジトリの場合 ) を使用して暗号化されます。この選択は、ブローカ全

体としては imq.authentication.type プロパティーで、または特定の接続サービスに対しては imq.*serviceName*. authentication.type で制御します。imq.authentication.client.response.timeout プロパティーは、認証要求のタイムアウトの間隔を設定します。

161 ページの「パスワードファイルの使用」で説明するように、パスワードはインテラクティブに入力を求める代わりに、パスワードファイルに指定することができます。ブール型のブローカプロパティー imq.passfile.enabled によってこのオプションを制御します。このプロパティーが true の場合、imq.passfile.dirpath および imq.passfile.name プロパティーで、パスワードファイルのディレクトリパスとファイル名を指定します。imq.imqcmd.password プロパティー ( パスワードファイルに埋め込み可能 ) は管理ユーザーが、ブローカ、接続サービス、接続、物理的送信先、永続サブスクリプション、トランザクションの管理にコマンドユーティリティー (imqcmd) を使用することを認証するためのパスワードを指定します。

LDAP ベースのユーザーリポジトリを使用する場合は、LDAP 検索のさまざまな側面を設定するために、あらゆる範囲のブローカプロパティーを使用できます。LDAP サーバー自体のアドレス ( ホスト名とポート番号 ) は imq.user_repository.ldap.server で指定します。imq.user_repository.ldap.principal プロパティーは、LDAP リポジトリにバインドするための識別名を指定し、imq.user_repository.ldap.password は関連パスワードを指定します。その他のプロパティーは、個々のユーザーおよびグループ検索用のディレクトリベースとオプションの JNDI フィルタ、ユーザーおよびグループ名のプロバイダ固有の属性識別子などを指定します。詳細については、309 ページの「セキュリティーのプロパティー」を参照してください。

## 承認

ユーザーは認証されると、さまざまな Message Queue 関連のアクティビティを実行することを承認されます。Message Queue 管理者は、ユーザーグループを定義し、各ユーザーにグループのメンバーシップを割り当てることができます。デフォルトのアクセス制御プロパティーファイルは、admin という 1 つのグループだけを明示的に参照します (139 ページの「グループ」を参照 )。このグループのユーザーは、admin 接続サービスの接続アクセス権を持ち、送信先の作成、ブローカの監視と制御などの管理機能を実行できます その他のグループに定義されたユーザーは、デフォルトでは admin サービスの接続アクセス権を取得できません。

ユーザーがある操作を実行しようとすると、ブローカが、ユーザーリポジトリ内のユーザー名とグループのメンバーシップを、アクセス制御プロパティーファイル内のその操作へのアクセスに指定されたユーザー名とグループのメンバーシップと照らし合わせます。アクセス制御プロパティーファイルでは、次の操作に対するユーザーまたはグループのアクセス権を指定します。

- ブローカへの接続

- 送信先へのアクセス : 特定の送信先、またはすべての送信先に対してのコンシューマ、プロデューサ、またはキューブラウザの作成
- 送信先の自動作成

### 暗号化

クライアントとブローカ間で送信されるメッセージを暗号化するには、SSL (Secure Socket Layer) 標準に基づいた接続サービスを使用する必要があります。SSL は、SSL 対応のブローカとクライアント間で暗号化された接続を確立して、接続レベルのセキュリティーを提供します。

SSL ベースの Message Queue 接続サービスを使用するには、キーツールユーティリティー (`imqkeytool`) を使用して、非公開キーと公開キーのペアを生成します。このユーティリティーは、自己署名型証明書に公開キーを組み込んで、それを Message Queue のキーストアに配置します。キーストア自体は、パスワードによって保護されているため、起動時に、`imq.keystore.password` プロパティーに指定されたキーストアのパスワードを入力して、ロックを解除する必要があります。キーストアのロックが解除されると、ブローカは、接続を要求しているクライアントに証明書を渡すことができます。証明書を受け取ると、クライアントはその証明書を使用して暗号化されたブローカへの接続を設定します。

`imq.audit.enabled` ブローカプロパティーは、Message Queue ブローカログファイルへの監査レコードのロギングを制御します。詳細については、163 ページの「監査ログの作成」を参照してください。

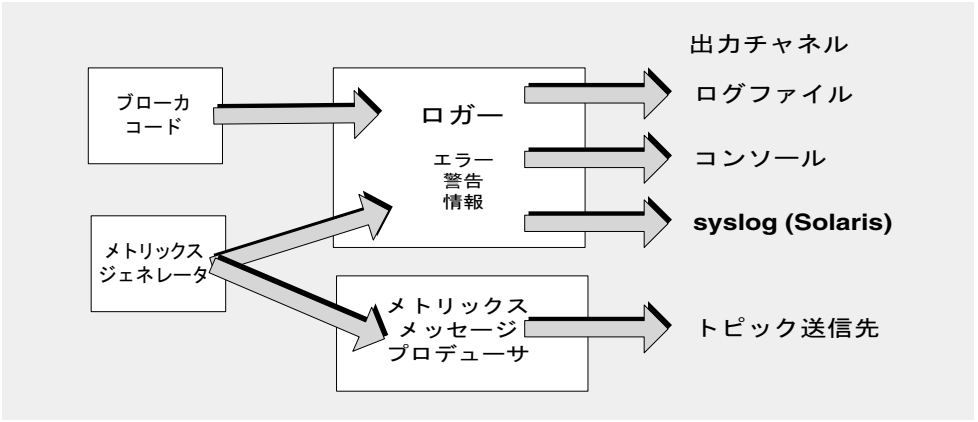
## 監視サービス

ブローカには、アプリケーションおよびブローカのパフォーマンスを監視し、診断するためのコンポーネントが含まれます。次のコンポーネントがあります。

- データを生成するコンポーネント ( イベントを記録するメトリックスジェネレータとブローカコード )
- 多数の出力チャネルに情報を書き込むロガーコンポーネント
- メトリックス情報を含む JMS メッセージを、JMS 監視クライアントによって消費させるためにトピック送信先へ送るメトリックスメッセージプロデューサ

この仕組みの概略を、図 4-3 に示します。管理サービスを設定するブローカプロパティーを、313 ページの「監視のプロパティー」に示します。

図 4-3　監視のサポート




## メトリックスジェネレータ

メトリックスジェネレータは、ブローカとの間で入出力されるメッセージフロー、ブローカメモリー内のメッセージ数とそれらが消費するメモリー量、開かれている接続の数、使用中のスレッドの数など、ブローカのアクティビティーに関する情報を提供します。ブール型のブローカプロパティー imq.metrics.enabled はそれらの情報を記録するかどうかを制御し、imq.metrics.interval は記録する頻度を指定します。

## ロガー

エラーの発生時に、ロガーで、ブローカコードおよびメトリックスジェネレータによって生成された情報が取得され、その情報が標準出力 ( コンソール )、ログファイル、Solaris プラットフォーム、syslog デーモンプロセスに書き込まれます。使用するログファイルは imq.log.file.dirpath および imq.log.file.filename ブローカプロパティーで指定し、imq.log.console.stream はコンソールの出力を stdout または stderr のどちらに書き込むかを指定します。

imq.log.level プロパティーは、ロガーが収集するメトリックス情報のカテゴリ (ERROR、WARNING、または INFO など ) を制御します。各レベルにはそれ以上のレベルも含むため、たとえばロギングレベルとして WARNING を指定すると、エラーメッセージも記録されます。imq.log.console.output および imq.log.file.output プロパティーは、指定したカテゴリのどれをコンソールに書き込み、どれをログファイルに書き込むかを制御します。ただし、この場合は、カテゴリにそれ以上のレベルが含まれないため、たとえば、エラーと警告の両方をログファイルに書き込み、情報メッセージをコンソールに書き込む場合は明示的に、imq.log.file.output を ERROR|WARNING に設定し、imq.log.console.output を INFO に設定する必要があります。Solaris プラットフォームでは、別のプロパティー imq.log.syslog.output で、

syslog デーモンに書き込むメトリックス情報のカテゴリを指定します。デッドメッセージが破棄された場合、またはデッドメッセージキューに移動された場合にログに記録するかどうかを指定する imq.destination.logDeadMsgs プロパティーもあります。

ログファイルの場合、ファイルを閉じて出力が新しいファイルにロールオーバーされる時点を指定できます。ログファイルが指定したサイズ (imq.log.file.rolloverbytes) または有効期間 (imq.log.file.rolloversecs) に達すると、保存されて新しいログファイルが作成されます。

ログに関連するその他のブローカプロパティーについては、313 ページの「監視のプロパティー」を参照してください。また、ロガーの設定方法およびロガーを使用してパフォーマンス情報を取得する方法の詳細については、195 ページの「ブローカロギングの設定と使用」を参照してください。

## メトリックスメッセージプロデューサ (Enterprise Edition)

メトリックスメッセージプロデューサは、メトリックスジェネレータから定期的に情報を受け取り、メトリックスメッセージに情報を書き込みます。メトリックスメッセージは、メッセージに含まれるメッセージ情報のタイプに応じて、多数のメトリックストピック送信先のいずれかに送信されます ( 表 4-2 を参照 )。これらのメトリックストピック送信先にサブスクライブされている Message Queue クライアントはメッセージを消費し、それらに含まれるメトリックスを処理できます。これにより、開発者はカスタム監視ツールを作成して、メッセージングアプリケーションをサポートできます。各タイプのメトリックスメッセージで報告されるメトリックス数の詳細は、『Message Queue Developer's Guide for Java Clients』を参照してください。

表 4-2　メトリックスのトピック送信先

| トピック名 | メトリックス情報のタイプ |
|---|---|
| mq.metrics.broker | ブローカのメトリックス |
| mq.metrics.jvm | Java 仮想マシンのメトリックス |
| mq.metrics.destination_list | 送信先とそれらのタイプのリスト |
| mq.metrics.destination.queue.*queueName* | 指定したキューの送信先メトリックス |
| mq.metrics.destination.topic.*topicName* | 指定したトピックの送信先メトリックス |

ブローカプロパティー imq.metrics.topic.enabled および imq.metrics.topic.interval はそれぞれ、メッセージをメトリックストピック送信先に送信するかどうか、その頻度を制御します。imq.metrics.topic.timetolive および imq.metrics.topic.persist プロパティーは、それらのメッセージの有効期間とそれらが持続メッセージであるかどうかを指定します。

メトリックスメッセージの本体に含まれる情報以外に、各メッセージのヘッダーには次の追加情報を提供するプロパティーが含まれます。

- メッセージのタイプ
- メッセージを送信したブローカのアドレス ( ホスト名とポート番号 )
- メトリックスサンプルを採取した時間

これらのプロパティーは、異なる種類または異なるブローカからのメトリックスメッセージを処理するクライアントアプリケーションに有用です。

# ブローカのプロパティーの設定

ブローカの設定プロパティーは、次のいずれかの方法で指定できます。

- ブローカの設定ファイルを編集する
- コマンド行から直接プロパティー値を入力する

次の 2 つの節で、この 2 つのブローカの設定方法について説明します。

## 設定ファイル

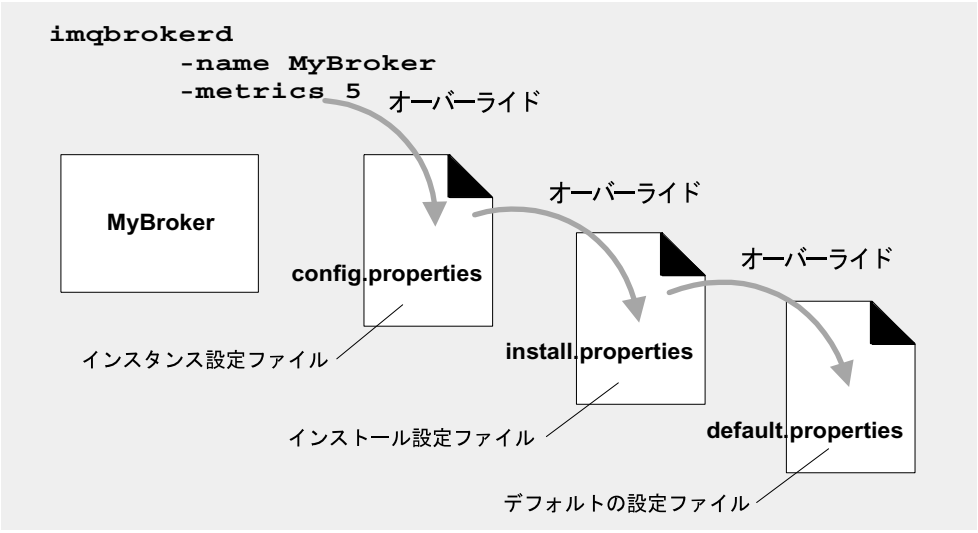
ブローカ設定ファイルには、ブローカを設定するプロパティー設定が格納されます。それらのファイルは、オペレーティングシステムプラットフォームによって異なる場所のディレクトリに保存されます。詳細については、付録 A「プラットフォームごとの Message Queue データの場所」を参照してください。このディレクトリには、次のファイルが格納されています。

- 起動時に読み込まれるデフォルトの設定ファイル default.properties。このファイルは編集できませんが、デフォルトの設定を決定したり、変更するプロパティーの正確な名前を検索したりする場合、このファイルに目を通すといいでしょう。
- Message Queue のインストール時に指定されたプロパティーを格納するインストール設定ファイル install.properties。このファイルはインストール後に編集できません。

さらに、後述するように、ブローカインスタンスごとに個別のインスタンス設定ファイルがあります。クラスタでブローカインスタンスを接続する場合、クラスタ設定ファイルを使用して、クラスタの設定情報を指定する必要があります。詳細については、318 ページの「クラスタ設定プロパティー」を参照してください。

起動時に、ブローカはさまざまな設定ファイルのプロパティー値をマージします。図 4-4 に示すように、ファイルは階層を形成し、インスタンス設定ファイルに指定された値によって、インストール設定ファイルの値がオーバーライドされ、さらに、デフォルトの設定ファイルの値によってそれらがオーバーライドされます。階層の最上部で、`imqbrokerd` コマンドにコマンド行オプションを使用して、設定ファイルに指定されている任意のプロパティー値を手動でオーバーライドできます。

**図 4-4**　ブローカ設定ファイル




## インスタンス設定ファイルの編集

最初にブローカを実行したときに、その特定のブローカインスタンスの設定プロパティーを格納するインスタンス設定ファイルが作成されます。インスタンス設定ファイルは `config.properties` と呼ばれ、設定ファイルが属するブローカインスタンスの名前によって識別されるディレクトリに格納されます。

*.../*`instances`*/instanceName/*`props/config.properties`

`instances` ディレクトリの場所については、付録 A 「プラットフォームごとの Message Queue データの場所」を参照してください。ファイルが存在しない場合、ブローカの起動時に `-name` オプションを使用して (272 ページの「ブローカユーティリティー」を参照 )、Message Queue がファイルの作成に使用できるインスタンス名を指定する必要があります。

| 注 | instances/*instanceName* ディレクトリとインスタンス設定ファイルは、対応するブローカインスタンスを作成したユーザーが所有します。ブローカインスタンスは、常に同じユーザーにより再起動されます。 |
|---|---|

インスタンス設定ファイルは、ブローカインスタンスによって管理され、Message Queue 管理ユーティリティーを使用して設定に変更を加えた場合に変更されます。インスタンス設定ファイルを手作業で編集して、ブローカの動作とリソースの使用をカスタマイズできます 手作業で編集するには、instances/*instanceName* ディレクトリの所有権が必要です。所有権がなければ、root としてログインしてディレクトリのアクセス権限を変更する必要があります。

ブローカがインスタンス設定ファイルを読み込むのは起動時だけです。ブローカの設定の変更を確定するには、ブローカをシャットダウンして、ファイルを編集し、ブローカを再起動する必要があります。ファイル ( または任意の設定ファイル ) 内のプロパティーの定義では、次の構文が使われます。

*propertyName*=*value*[[ , *value1*]...]

たとえば、次のエントリは、ブローカが追加メッセージを拒否するまでに、メモリーと持続ストレージに最大 50,000 メッセージを保持するように指定します。

```
imq.system.max_count=50000
```

次のエントリは、毎日、つまり 86,400 秒ごとに新しいログファイルを作成するように指定します。

```
imq.log.file.rolloversecs=86400
```

使用可能なブローカ設定プロパティーとそれらのデフォルト値については、77 ページの「ブローカサービス」および第 14 章「ブローカのプロパティーのリファレンス」を参照してください。

## コマンド行からの設定オプションの設定

ブローカの起動時、または起動後に、コマンド行からブローカ設定オプションを入力できます。

起動時に、ブローカユーティリティー (imqbrokerd) を使用してブローカインスタンスを起動します。コマンドの -D オプションを使用すると、ブローカの設定プロパティーとその値を指定できます。詳細については、68 ページの「ブローカの起動」および 272 ページの「ブローカユーティリティー」を参照してください。サービス管理ユーティリティー (imqsvcadmin) を使用して、Windows サービスとしてブローカを起動している場合、-args オプションを使用して起動時設定プロパティーを指定します。293 ページの「サービス管理ユーティリティー」を参照してください。

また、ブローカインスタンスの実行中に、特定のブローカプロパティーを変更できます。実行中のブローカの設定を変更するには、コマンドユーティリティーの `imqcmd update bkr` コマンドを使用します。104 ページの「ブローカのプロパティーの更新」および280 ページの「ブローカ管理」を参照してください。

# 持続データストアの設定

ブローカの持続データストアには、物理的送信先、永続サブスクリプション、メッセージ、トランザクション、および通知に関する情報が格納されます。Message Queue ブローカはデフォルトで、ファイルベースの持続ストアを使用するように設定されますが、JDBC 互換ドライバからアクセス可能な任意のデータストアに接続するようにブローカを設定し直すことができます。ブローカ設定プロパティー `imq.persist.store` (303 ページの表 14-4 を参照 ) は 2 つの持続形式のどちらを使用するかを指定します。

この節では、持続ストアを使用するようにブローカを設定する方法を説明します。次のトピックが含まれます。

- 95 ページの「ファイルベースのストアの設定」
- 96 ページの「JDBC ベースのストアの設定」
- 97 ページの「持続データの保護」

## ファイルベースのストアの設定

ファイルベースのデータストアは、ブローカインスタンスの作成時に自動的に作成されます。このストアは、ブローカのインスタンスディレクトリに配置されます。正確な場所については、付録 A 「プラットフォームごとの Message Queue データの場所」を参照してください。

デフォルトでは、Message Queue はディスクへの非同期の書き込み操作を実行します。オペレーティングシステムは、このような操作をバッファリングし、パフォーマンスを高めることができます。ただし、不測のシステム障害が書き込み操作の間に発生した場合、メッセージは失われる可能性があります。信頼性を高めるために ( パフォーマンスの低下と引き換えに )、データを同時に書き込むように、ブローカプロパティー `imq.persist.file.sync` を設定できます。このプロパティーの詳細な説明については、83 ページの「ファイルベースの持続」および304 ページの表 14-5 を参照してください。

ブローカインスタンスを起動すると、`imqbrokerd` コマンドの `-reset` オプションを使用してファイルシステムストアを消去できます。このオプションおよびサブオプションの詳細は、272 ページの「ブローカユーティリティー」を参照してください。

# JDBC ベースのストアの設定

JDBC ベースの持続を使用するようにブローカを設定するには、ブローカインスタンス設定ファイルで JDBC 関連のプロパティーを設定し、適切なデータベーススキーマを作成します。Message Queue のデータベースマネージャーユーティリティー (`imqdbmgr`) は、JDBC ドライバとブローカ設定プロパティーを使用してデータベースを作成し、管理します。さらに、破損したテーブルをデータベースから削除する場合や別のデータベースをデータストアとして使用する場合に、データベースマネージャーを使用することもできます。詳細については、290 ページの「データベースマネージャーユーティリティー」を参照してください。

**注** Oracle および PointBase データベース製品の設定例を参照できます。これらのファイルの場所は、プラットフォームによって異なり、付録 A 「プラットフォームごとの Message Queue データの場所」の関連する表「アプリケーションと設定の例」に記載されています。さらに、PointBase の組み込みバージョン、PointBase サーバーバージョン、および Oracle の例は、インスタンス設定ファイル `config.properties` 内でコメントアウトされた値として提供されています。

## ➤ JDBC ベースのデータストアを設定する

1. ブローカの設定ファイルに、JDBC 関連のプロパティーを設定します。

   関連プロパティーについては、84 ページの「JDBC ベースの持続」で説明し、306 ページの表 14-6 に示しています。特に、ブローカの `imq.persist.store` プロパティーを `jdbc` に設定する必要があります (303 ページの表 14-4 を参照 )。

2. 次の場所の JDBC ドライバの `.jar` ファイルにコピーまたはシンボリックリンクを配置します。

   `/usr/share/lib/imq/ext/` (Solaris)
   `/opt/sun/mq/share/lib/` (Linux)
   `IMQ_VARHOME¥lib¥ext` (Windows)

   たとえば、Solaris システムで PointBase を使用している場合、次のコマンドでドライバの `.jar` ファイルを適切な場所にコピーします。

   ```
   % cp j2eeSDKInstallDirectory/pointbase/lib/pointbase.jar
   /usr/share/lib/imq/ext
   ```

   次のコマンドはシンボリックリンクを作成します。

   ```
   % ln -s j2eeSDKInstallDirectory/lib/pointbase/pointbase.jar
   /usr/share/lib/imq/ext
   ```

3. Message Queue の持続に必要なデータベーススキーマを作成します。

組み込みデータベース用の `imqdbmgr create all` コマンドまたは外部データベース用の `imqdbmgr create tbl` コマンドを使用します。290 ページの「データベースマネージャーユーティリティー」を参照してください。

a. `imqdbmgr` がある場所にディレクトリを変更します。

```
cd /usr/bin (Solaris)
cd /opt/sun/mq/bin (Linux)
cd IMQ_HOME¥bin (Windows)
```

b. `imqdbmgr` コマンドを入力します。

```
imqdbmgr create all
```

**注** 組み込みデータベースを使用している場合、次のディレクトリ内に作成するのが最適です。

*.../*`instances/`*instanceName*`/dbstore/`*databaseName*

組み込みデータベースは、ユーザー名とパスワードで保護されていない場合、ファイルシステムのアクセス権によって保護される可能性があります。ブローカが確実にデータベースに対して読み取りと書き込みを実行できるようにするため、ブローカを実行するユーザーは、`imqdbmgr` コマンドを使用して組み込みデータベースを作成したユーザーと同一でなければなりません。

# 持続データの保護

持続ストアにはほかの情報とともに、一時的に保存されるメッセージファイルを保存できます。これらのメッセージには専有情報が保持されている場合があるため、認可されていないアクセスからデータストアを保護することをお勧めします。この節では、ファイルベースまたは JDBC ベースのデータストアでデータを保護する方法を説明します。

## ファイルベースのストアの保護

ファイルベースの持続を使用するブローカは、プラットフォームにより場所が異なる単層型ファイルのデータストアに持続データを書き込みます ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

*.../*`instances/`*instanceName*`/fs350/`

*instanceName* には、ブローカインスタンスを識別する名前が入ります。

*instanceName*`/fs350/` ディレクトリは、ブローカインスタンスがはじめて開始されたときに作成されます。このディレクトリを保護するための手順は、ブローカを実行しているオペレーティングシステムプラットフォームによって異なります。

- Solaris および Linux の場合、ディレクトリのアクセス権は、ブローカインスタンスを開始したユーザーのファイルモード作成マスク (umask) によって決定されます。したがって、ブローカインスタンスの開始および持続ファイルの読み取りを行うためのアクセス権は、マスクを適切に設定することによって制限できることになります。あるいは、スーパーユーザーである管理者は、instances ディレクトリのアクセス権を 700 に設定することによって、持続データを保護できます。
- Windows の場合、ディレクトリのアクセス権は、Windows オペレーティングシステムが提供するメカニズムを使って設定できます。この操作では、通常そのディレクトリの「プロパティー」ダイアログを開きます。

## JDBC ベースのストアの保護

JDBC ベースの持続を使用するブローカは、JDBC 互換データベースに持続データを書き込みます。Oracle などのデータベースサーバーによって管理されるデータベースについては、Message Queue のデータベーステーブル (IMQ で始まる名前が付けられたテーブル ) にアクセスするためのユーザー名とパスワードを作成することをお勧めします。データベースで個々のテーブルの保護ができない場合、Message Queue ブローカだけが使用する専用のデータベースを作成します。ユーザー名とパスワードのアクセス権を作成する方法については、データベースベンダーによって提供されているマニュアルを参照してください。

データベース接続を開くためにブローカが求めるユーザー名とパスワードは、ブローカ設定プロパティーとして与えることができます。ただし、imqbrokerd コマンドの -dbuser および -dbpassword オプションを使用して、ブローカの起動時にコマンド行オプションとして入力するほうがより安全です (272 ページの「ブローカユーティリティー」を参照 )。

データベースの JDBC ドライバを使用してブローカが直接アクセスする組み込みデータベースの場合、「ファイルベースのストアの保護」で説明したように、通常は持続データが格納されるディレクトリにファイルアクセス権を設定することでセキュリティーが確保されます。ただし、データベースをブローカとデータベースマネージャーユーティリティーの両方から読み取り可能および書き込み可能にするためには、いずれも同じユーザーにより実行される必要があります。

第 5 章

# ブローカの管理

この章では、imqcmd ユーティリティーを使用して、ブローカおよびそのサービスを管理する方法について説明します。この章では、次の節について説明します。



この章ではブローカの管理に関連したすべてのトピックは扱いません。その他のトピックは、次の章で個別に扱っています。

- ブローカでの物理的送信先の管理。物理的送信先の作成、表示、更新、破棄の方法、およびデッドメッセージキューの使い方といったトピックの詳細は、第 6 章「物理的送信先の管理」を参照してください。
- ブローカのセキュリティー設定。ユーザー認証、アクセス制御、暗号化、パスワードファイル、監査ロギングなどのトピックの詳細は、第 7 章「セキュリティーの管理」を参照してください。

# 前提条件

ブローカの管理には、imqcmd および imqusermgr コマンド行ユーティリティーを使用します。ブローカを管理する前に、次の作業が必要です。

- imqbrokerd ユーティリティーコマンドを使用して、ブローカを起動する。ブローカを実行するまで、ほかのコマンド行ユーティリティーは使用できません。
- Message Queue 管理ユーザーを設定するか、デフォルトアカウントを使用するかを決定する。管理コマンドを使用する場合、ユーザー名とパスワードを指定する必要があります。

  Message Queue をインストールすると、デフォルトの単層ファイルのユーザーリポジトリがインストールされます。リポジトリは 2 つのデフォルトエントリである、管理ユーザーとゲストユーザーと一緒に出荷されます。Message Queue をテストする場合、デフォルトのユーザー名とパスワード (admin/admin) を使用して、imqcmd ユーティリティーを実行できます。

  本稼動システムをセットアップする場合は、管理ユーザーの認証および認可を設定する必要があります。ファイルベースのユーザーリポジトリの設定、または LDAP ディレクトリサーバーを使用する設定の詳細は、第 7 章「セキュリティーの管理」を参照してください。本稼働環境では、セキュリティー上の理由によりデフォルト以外のユーザー名とパスワードを使用することをお勧めします。
- ブローカとの安全な接続を使用する場合、ターゲットブローカインスタンスで ssladmin サービスを設定し有効化します。詳細は、151 ページの「SSL ベースのサービスの操作」を参照してください。

# imqcmd ユーティリティーの使用

imqcmd ユーティリティーを使用すると、ブローカとブローカのサービスを管理できます。

imqcmd コマンドの構文、サブコマンド、オプションの詳細は、271 ページの第 13 章「コマンド行のリファレンス」を参照してください。物理的送信先の管理の詳細は、325 ページの第 15 章「物理的送信先のプロパティーのリファレンス」で個別に扱っています。

# ヘルプの表示

imqcmd ユーティリティーでヘルプを表示するには、-h オプションまたは -H オプションを使用し、サブコマンドは使用しません。特定のサブコマンドのヘルプは表示されません。

たとえば、次のコマンドは imqcmd に関するヘルプを表示します。

```
imqcmd -H
```

サブコマンドまたはその他のオプションに加えて、-h オプションまたは -H オプションを指定してコマンド行を入力した場合、imqcmd ユーティリティーは -h オプションまたは -H オプションのみを処理します。コマンド行のほかのすべての項目は無視されます。

# 製品のバージョンの表示

Message Queue の製品のバージョンを表示するには、-v オプションを使用します。たとえば、次のように指定します。

```
imqcmd -v
```

サブコマンドまたはその他のオプションに加えて、-v オプションを指定してコマンド行を入力した場合、imqcmd ユーティリティーは -v オプションのみを処理します。コマンド行のほかのすべての項目は無視されます。

# ユーザー名とパスワードを指定する

それぞれの imqcmd サブコマンドはユーザーリポジトリに対して認証されるため、ユーザー名とパスワードが必要になります。ただし、ヘルプを表示するための -h または -H オプションを使用するコマンド、および製品のバージョンを表示するための -v オプションを使用するコマンドには必要ありません。

## ユーザー名を指定する

管理ユーザー名を指定する場合は、-u オプションを使用します。ユーザー名を省略すると、コマンドから入力が要求されます。たとえば、次のコマンドはデフォルトのブローカに関する情報を表示します。

```
imqcmd query bkr -u admin
```

この章の例を読みやすくするために、デフォルトのユーザー名 admin は -u オプションの引数として示しています。本稼動環境では、カスタムユーザー名を使用します。

### パスワードを指定する

パスワードは次のいずれかの方法で指定します。

- パスワードファイル (passfile) を作成し、そのファイルにパスワードを入力する。コマンド行で、-passfile オプションを使用してパスワードファイルの名前を指定する。
- コマンドからパスワードの入力を要求する。

これまでのバージョンの Message Queue では、-p オプションを使用して imqcmd コマンド行にパスワードを指定できました。このオプションは異論が多く、今後のバージョンでは削除される予定です。

## ブローカ名とポートを指定する

imqcmd のデフォルトブローカは、ローカルホストで実行中のブローカであり、デフォルトポートは 7676 です。

リモートホストで実行中のブローカまたはデフォルト以外のポートで待機中のブローカ、あるいはその両方にコマンドを発行する場合、-b オプションを使用してブローカのホストとポートを指定する必要があります。

## 例

この節の例は、imqcmd の使い方を表しています。

最初の例では、localhost のポート 7676 で実行中のブローカのプロパティーを一覧表示しているため、-b オプションは不要です。このコマンドはデフォルトの管理ユーザー名 (admin) を使用してパスワードを省略しています。したがってコマンドで入力が要求されています。

```
imqcmd query bkr -u admin
```

次の例では、ホスト myserver のポート 1564 で実行中のブローカのプロパティーを一覧表示しています。ユーザー名は aladdin です。このコマンドが機能するためには、ユーザーリポジトリを更新して、aladdin を admin グループに追加する必要がある場合があります。

```
imqcmd query bkr -b myserver:1564 -u aladdin
```

次の例では、localhost のポート 7676 で実行中のブローカのプロパティーを一覧表示しています。このコマンドの最初のタイムアウトは 20 秒に設定され、タイムアウト後の再試行回数が 7 回に設定されています。ユーザーのパスワードは、コマンドを呼び出したときに現在のディレクトリにある myPassfile と呼ばれるパスワードファイル内に格納されています。

```
imqcmd query bkr -u admin -passfile myPassfile -rtm 20 -rtr 7
```

ブローカとの安全な接続を確立するために、次の例では -secure オプションを指定しています。ssladmin サービスが設定および起動されていれば、imqcmd は -secure オプションを指定したときに ssladmin サービスを使用します。

# ブローカ情報の表示

シングルブローカに関する情報のクエリーと表示を行うには、query bkr サブコマンドを使用します。

次に示すのは、query bkr サブコマンドの構文です。

```
imqcmd query bkr -b hostName:portNumber
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカの現在のプロパティーの設定を一覧表示します。また、特定のブローカに接続している実行中のブローカ ( マルチブローカクラスタ内のブローカ ) のリストも表示されます。

たとえば、次のように指定します。

```
imqcmd query bkr -u admin
```

パスワードの入力を要求した後、コマンドは次のような出力を生成します。

```
Version                                                 3.6
Instance Name                                           imqbroker
Primary Port                                            7676

Current Number of Messages in System                    0
Current Total Message Bytes in System                   0

Current Number of Messages in Dead Message Queue        0
Current Total Message Bytes in Dead Message Queue       0

Log Dead Messages                                       true
Truncate Message Body in Dead Message Queue             false

Max Number of Messages in System                        unlimited
(-1)
Max Total Message Bytes in System                       unlimited
(-1)
Max Message Size                                        70m

Auto Create Queues                                      true
Auto Create Topics                                      true
Auto Created Queue Max Number of Active Consumers       1
Auto Created Queue Max Number of Backup Consumers       0
```

```
Cluster Broker List (active)
Cluster Broker List (configured)
Cluster Master Broker
Cluster URL

Log Level                                                INFO
Log Rollover Interval (seconds)                          604800
Log Rollover Size (bytes)                                unlimited
(-1)
```

# ブローカのプロパティーの更新

次のブローカのプロパティーを更新する場合は、update bkr サブコマンドを使用します。

- imq.autocreate.queue
- imq.autocreate.topic
- imq.autocreate.queue.maxNumActiveConsumers
- imq.autocreate.queue.maxNumBackupConsumers
- imq.cluster.url
- imq.destination.DMQ.truncateBody
- imq.destination.logDeadMsgs
- imq.log.level
- imq.log.file.rolloversecs
- imq.log.file.rolloverbytes
- imq.system.max_count
- imq.system.max_size
- imq.message.max_size
- imq.portmapper.port

次に示すのは、update bkr サブコマンドの構文です。

```
imqcmd update bkr [-b hostName:portNumber]-o attribute=value [[-o
attribute=value1]...]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカに対して、指定した属性を変更します。たとえば、次のコマンドはキュー送信先の自動作成を無効にします。

```
imqcmd update bkr -o "imq.autocreate.queue=false" -u admin
```

プロパティーは、第 14 章「ブローカのプロパティーのリファレンス」で説明しています。

# ブローカの停止および再開

ブローカの起動後に、imqcmd のサブコマンドを使用して、ブローカの状態を制御できます。

## ブローカを停止する

ブローカを停止すると、ブローカの接続サービススレッドが中断されるため、ブローカは接続ポートでの待機をやめます。その結果、ブローカはそれ以上、新しい接続の受け入れ、メッセージの受信、メッセージのディスパッチは行いません。

ただし、ブローカを停止しても admin 接続サービスは中断されないため、ブローカへのメッセージを制限するために必要な管理タスクは実行できます。ブローカを停止しても、cluster 接続サービスは継続されます。ただし、クラスタ内のメッセージ配信は、クラスタ内のブローカによって実行される配信機能によって異なります。そのため、クラスタ内のブローカを停止すると、一部のメッセージトラフィックが遅くなる可能性があります。

次に示すのは、pause bkr サブコマンドの構文です。

```
imqcmd pause bkr [-b hostName:portNumber]
```

このコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカを停止します。

次のコマンドでは、myhost のポート 1588 で実行しているブローカが停止されます。

```
imqcmd pause bkr -b myhost:1588 -u admin
```

個々の接続サービス、および個々の物理的送信先も停止できます。詳細は、112 ページの「接続サービスの停止および再開」と 126 ページの「物理的送信先の停止と再開」を参照してください。

## ブローカを再開する

ブローカを再開すると、ブローカのサービススレッドが再び有効になり、ブローカはポートでの待機を再開します。

次に示すのは、resume bkr サブコマンドの構文です。

```
imqcmd resume bkr [-b hostName:portNumber]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカを再開します。

次のコマンドでは、localhost のポート 7676 で実行していたブローカが再開されます。

```
imqcmd resume bkr -u admin
```

# ブローカのシャットダウンと再起動

ブローカをシャットダウンすると、正常にブローカプロセスを終了することができます。ブローカは新しい接続やメッセージを受け入れるのをやめて、既存のメッセージの配信を完了し、ブローカプロセスを終了します。

次に示すのは、shutdown bkr サブコマンドの構文です。

```
imqcmd shutdown bkr [-b hostName:portNumber]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカをシャットダウンします。

次のコマンドでは、ctrlsrv のポート 1572 で実行していたブローカがシャットダウンされます。

```
imqcmd shutdown bkr -b ctrlsrv:1572 -u admin
```

restart bkr サブコマンドを使用して、ブローカをシャットダウンし、再起動します。次に示すのは、restart bkr サブコマンドの構文です。

```
imqcmd restart bkr [-b hostName:portNumber]
```

このサブコマンドは、最初にブローカを起動したときに指定されたオプションを使用して、デフォルトのブローカ、または指定されたホストとポートのブローカをシャットダウンし、再起動します。別のオプションを選択する場合は、必要なオプションを指定して、ブローカをシャットダウンしてから再起動します。

# ブローカのメトリックスの表示

ブローカに関するメトリックス情報を表示するには、metrics bkr サブコマンドを使用します。

次に示すのは、metrics bkr サブコマンドの構文です。

```
imqcmd metrics bkr [-b hostName:portNumber]
        [-m metricType] [-int interval] [-msp numSamples]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカに対して、ブローカのメトリックスを表示します。

表示するメトリックスのタイプを次の中から指定するには、-m オプションを使用します。

- **ttl** ブローカとの間で入出力されているメッセージとパケットのフローに関するメトリックスを表示します ( デフォルトのメトリックスタイプ )。
- **rts** ブローカとの間で入出力されているメッセージとパケットの1秒あたりのフローレートに関するメトリックスを表示します。
- **cxn** 接続、仮想メモリーヒープ、およびスレッドを表示します。

メトリックスを表示する間隔を秒単位で指定するには、-int オプションを使用します。デフォルトは 5 秒です。

出力で表示するサンプル数を指定するには、-msp オプションを使用します。デフォルトは無制限です ( 無限 )。

たとえば、ブローカに入力するメッセージフローとブローカから出力されるメッセージのフローレートを 10 秒間隔で取得するには、次のコマンドを使用します。

```
imqcmd metrics bkr -m rts -int 10 -u admin
```

このコマンドでは、次のような情報が出力されます。

```
------------------------------------------------------------
 Msgs/sec    Msg Bytes/sec    Pkts/sec     Pkt Bytes/sec
 In   Out      In       Out      In   Out      In       Out
------------------------------------------------------------
 0     0       27       56       0     0       38       66
 10    0      7365      56       10    10     7457     1132
 0     0       27       56       0     0       38       73
 0     10      27      7402      10    20     1400     8459
 0     0       27       56       0     0       38       73
```

ブローカによって収集され、レポートされるデータの詳細については、350 ページの「ブローカ全体のメトリックス」を参照してください。

# 接続サービスの管理

imqcmd ユーティリティーには、次の接続サービス管理タスクを実行するために使用できるサブコマンドが含まれています。

- 接続サービスの一覧表示
- 接続サービス情報の表示
- 接続サービスのプロパティーの更新
- 接続サービスのメトリックスの表示
- 接続サービスの停止および再開

ブローカは、アプリケーションクライアントと管理クライアントの両方からの通信をサポートしています。Message Queue のブローカで現在使用できる接続サービスを、表 5-1 に示します。表が示すように、各サービスは使用するサービスタイプ ( アプリケーションクライアントの場合は NORMAL、管理クライアントの場合は ADMIN) と基礎となるトランスポートプロトコルに関連付けられます。

**表 5-1** ブローカがサポートする接続サービス

| サービス名 | サービスタイプ | プロトコルタイプ |
|---|---|---|
| jms | NORMAL | tcp |
| ssljms (Enterprise Edition) | NORMAL | tls (SSL ベースセキュリティー ) |
| httpjms (Enterprise Edition) | NORMAL | http |
| httpsjms (Enterprise Edition) | NORMAL | https (SSL ベースセキュリティー ) |
| admin | ADMIN | tcp |
| ssladmin (Enterprise Edition) | ADMIN | tls (SSL ベースセキュリティー ) |

imqcmd サブコマンドを使用して、接続サービス全体を管理するか、または特定の接続サービスを管理することができます。サブコマンドの対象が特定のサービスの場合は、-n オプションを使用して、表 5-1 の「サービス名」列に示されたいずれかの名前を指定します。

# 接続サービスの一覧表示

ブローカで使用できる接続サービスを一覧表示するには、list svc サブコマンドを使用します。

次に示すのは、list svc サブコマンドの構文です。

```
imqcmd list svc [-b hostName:portNumber]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカのすべての接続サービスを一覧表示します。

次のコマンドでは、localhost のポート 7676 で実行しているブローカのすべてのサービスが一覧表示されます。

```
imqcmd list svc -u admin
```

このコマンドでは、次のような情報が出力されます。

```
-----------------------------------------------
Service Name    Port Number         Service State
-----------------------------------------------
admin           41844 (dynamic)     RUNNING
httpjms         -                   UNKNOWN
httpsjms        -                   UNKNOWN
jms             41843 (dynamic)     RUNNING
ssladmin         dynamic             UNKNOWN
ssljms          dynamic             UNKNOWN
```

# 接続サービス情報の表示

シングルサービスに関する情報のクエリーと表示を行うには、query サブコマンドを使用します。

次に示すのは、query svc サブコマンドの構文です。

```
imqcmd query svc -n serviceName [-b hostName:portNumber]
```

query svc サブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカで実行している特定のサービスに関する情報を一覧表示します。

たとえば、次のように指定します。

```
imqcmd query svc -n jms -u admin
```

パスワードの入力を要求した後、コマンドは次のような出力を生成します。

```
Service Name                              jms
Service State                             RUNNING
Port Number                               60920 (dynamic)

Current Number of Allocated Threads       0
Current Number of Connections             0

Min Number of Threads                     10
Max Number of Threads                     1000
```

# 接続サービスのプロパティーの更新

表 5-2 に示す 1 つ以上のサービスのプロパティーの値を変更するには、update サブコマンドを使用します。

表 5-2　imqcmd によって更新される接続サービスプロパティー

| プロパティー | 説明 |
|---|---|
| port | 更新するサービスに割り当てられるポートです (httpjms または httpsjms には適用しない )。値 0 は、ポートマッパーによって動的に割り当てられるポートを示しています。 |
| minThreads | サービスに割り当てられるスレッドの最小数 |
| maxThreads | サービスに割り当てられるスレッドの最大数 |

次に示すのは、update サブコマンドの構文です。

```
imqcmd update svc -n serviceName [-b hostName:portNumber]
          -o attribute=value  [-o attribute=value1]...
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカで実行している特定のサービスの特定の属性を更新します。サービスの属性については、295 ページの「接続のプロパティー」を参照してください。

次のコマンドでは、jms サービスに割り当てられたスレッドの最小数が 20 に変更されます。

```
imqcmd update svc -n jms -o "minThreads=20" -u admin
```

# 接続サービスのメトリックスの表示

シングルサービスに関するメトリックス情報を表示するには、metrics サブコマンドを使用します。

次に示すのは、metrics サブコマンドの構文です。

```
imqcmd metrics svc -n serviceName [-b hostName:portNumber] [-m metricType]
    [-int interval] [-msp numSamples]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカで実行している特定のサービスのメトリックスを表示します。

表示するメトリックスのタイプを次の中から指定するには、-m オプションを使用します。

- **ttl**　指定した接続サービスを使ってブローカとの間で入出力されているメッセージとパケットのフローに関するメトリックスを表示します ( デフォルトのメトリックスタイプ )。
- **rts**　指定した接続サービスを使ってブローカとの間で入出力されているメッセージとパケットのフローレートに関するメトリックスを表示します。
- **cxn**　接続、仮想メモリーヒープ、およびスレッドを表示します。

メトリックスを表示する間隔を秒単位で指定するには、-int オプションを使用します。デフォルトは 5 秒です。

出力で表示するサンプル数を指定するには、-msp オプションを使用します。デフォルトは無制限です ( 無限 )。

たとえば、jms 接続サービスによって処理されたメッセージとパケットの累計数を取得するには、次のコマンドを使用します。

```
imqcmd metrics svc -n jms -m ttl -u admin
```

パスワードの入力を要求した後、コマンドは次のような出力を生成します。

```
-----------------------------------------------
  Msgs       Msg Bytes       Pkts       Pkt Bytes
In   Out     In      Out    In   Out     In      Out
-----------------------------------------------
164  100   120704   73600   282  383   135967   102127
657  100   483552   73600   775  876   498815   149948
```

imqcmd を使用して接続サービスのメトリックスをレポートする方法の詳細は、352 ページの「接続サービスのメトリックス」を参照してください。

# 接続サービスの停止および再開

管理サービス ( 停止することが禁止されているサービス ) 以外のサービスを停止するには、pause svc サブコマンドと resume svc サブコマンドを使用します。

次に示すのは、pause svc サブコマンドの構文です。

```
imqcmd pause svc -n serviceName [-b hostName:portNumber]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカで実行している特定のサービスを停止します。たとえば、次のコマンドは、デフォルトのブローカで実行している httpjms サービスを停止します。

```
imqcmd pause svc -n httpjms -u admin
```

サービスを停止すると、次のような結果になります。

- ブローカは、停止したサービスでの新たなクライアント接続の受け入れをやめます。Message Queue クライアントが新しい接続を開こうとすると、例外が発生します。
- 停止したサービスの既存の接続はすべて維持されますが、ブローカはサービスが再開されるまでこれらの接続のすべてのメッセージ処理を中断します。たとえば、クライアントがメッセージを送信しようとしても、サービスが再開されるまでは、send メソッドがそれを阻止します。
- すでにブローカが受信済みのメッセージのメッセージ配信状態は維持されます。たとえば、トランザクションは中断されず、サービスが再開された時点でメッセージ配信も再開されます。

サービスを再開するには、resume svc サブコマンドを使用します。

次に示すのは、resume svc サブコマンドの構文です。

```
imqcmd resume svc -n serviceName[-b hostName:portNumber]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカで実行している特定のサービスを再開します。

# 接続情報の入手

imqcmd ユーティリティーには、接続に関する情報を一覧表示し取得するために使用できるサブコマンドが含まれています。

list cxn サブコマンドは、指定されたサービス名のすべての接続を一覧表示します。次に示すのは、list cxn サブコマンドの構文です。

```
imqcmd list cxn [-svn serviceName] [-b hostName:portNumber]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカの指定したサービス名の接続をすべて一覧表示します。サービス名が指定しない場合は、すべての接続が一覧表示されます。

たとえば、次のコマンドはデフォルトのブローカのすべての接続を表示します。

```
imqcmd list cxn -u admin
```

パスワードの入力を要求した後、コマンドは次のような出力を生成します。

```
Listing all the connections on the broker specified by:
-------------------------------
Host                   Primary Port
-----------------------------------
localhost              7676

--------------------------------------------------------------------------------
Connection ID         User    Service   Producers  Consumers    Host
--------------------------------------------------------------------------------
1964412264455443200   guest   jms       0          1            127.0.0.1
1964412264493829311   admin   admin     1          1            127.0.0.1

Successfully listed connections.
```

シングル接続サービスに関する情報のクエリーと表示を行うには、query サブコマンドを使用します。

```
query cxn -n connectionID [-b hostName:portNumber]
```

このサブコマンドは、デフォルトのブローカ、または指定したホストとポートのブローカの指定した接続に関する情報を表示します。

たとえば、次のように指定します。

```
imqcmd query cxn -n 421085509902214374 -u admin
```

パスワードの入力を要求した後、コマンドは次のような出力を生成します。

```
Connection ID      421085509902214374
User               guest
Service            jms
Producers          0
Consumers          1
Host               111.22.333.444
Port               60953
Client ID
Client Platform
```

# 永続サブスクリプションの管理

imqcmd サブコマンドを使用して、次のような操作を実行して、ブローカの永続サブスクリプションを管理できます。

- 永続サブスクリプションを一覧表示する
- 永続サブスクリプションのすべてのメッセージを消去する
- 永続サブスクリプションを破棄する

永続サブスクリプションとは、クライアントによって、永続的であると登録されたトピックのサブスクリプションのことです。このサブスクリプションには固有の識別情報があり、コンシューマがアクティブになっていないときでも、サブスクリプションのメッセージを保持するブローカが必要となります。通常、ブローカはメッセージの有効期限が切れたときだけ、保持していた永続サブスクライバのメッセージを削除します。

指定された物理的送信先の永続サブスクリプションを一覧表示するには、`list dur` サブコマンドを使用します。次に示すのは、list dur サブコマンドの構文です。

```
imqcmd list dur -d destName
```

たとえば、次のコマンドはローカルホストのデフォルトポートのブローカを使用する、トピック `SPQuotes` のすべての永続サブスクリプションを一覧表示します。

```
imqcmd list dur -d SPQuotes
```

`list dur` サブコマンドでは、トピックの永続サブスクリプションごとに、永続サブスクリプションの名前、ユーザーのクライアント ID、このトピックのキューに入っているメッセージの数、および永続サブスクリプションの状態 ( アクティブまたは非アクティブ ) を返します。たとえば、次のように指定します。

```
Name         Client ID       Number of    Durable Sub
                             Messages       State
-------------------------------------------------------------
myDurable    myClientID       1            INACTIVE
```

list dur サブコマンドから返される情報を使用して、破棄する必要がある永続サブスクリプションやメッセージを消去する必要がある永続サブスクリプションを識別することができます。

purge dur サブコマンドは、指定されたクライアント識別子を持つ特定の永続サブスクリプションのすべてのメッセージを消去します。次に示すのは、purge dur サブコマンドの構文です。

```
imqcmd purge dur -n subscrName -c clientID
```

destroy dur サブコマンドは、指定されたクライアント識別子を持つ特定の永続サブスクリプションを破棄します。次に示すのは、destroy dur サブコマンドの構文です。

```
imqcmd destroy dur -n subscrName -c clientID
```

たとえば、次のコマンドは、永続サブスクリプション myDurable と clientID myClientID を破棄します。

```
imqcmd destroy dur -n myDurable -c myClientID
```

# トランザクションの管理

クライアントアプリケーションによって開始されたトランザクションはすべてブローカによって記録されます。これらは、分散トランザクション (XA リソース ) マネージャーによって管理される Message Queue の単純なトランザクション、または分散トランザクションです。

各トランザクションには、Message Queue トランザクション ID が付けられています。これは、ブローカのトランザクションを一意に識別するための 64 ビットの数字です。また、分散トランザクションには、分散トランザクションマネージャーによって割り当てられる最大 128 バイトの分散トランザクション ID (XID) が付けられます。Message Queue は、Message Queue トランザクション ID と XID の関連付けを保持します。

分散トランザクションの場合、障害が発生すると、トランザクションがコミットされずに PREPARED 状態のままになる可能性があります。このため、管理者は監視を行い、PREPARED 状態のトランザクションをロールバックするか、またはコミットする必要があります。

ブローカが追跡するすべてのトランザクションを一覧表示するには、`list txn` コマンドを使用します。次に示すのは、`list tx` サブコマンドの構文です。

```
imqcmd list txn
```

たとえば、次のコマンドでは、ブローカのすべてのトランザクションが一覧表示されます。

```
imqcmd list txn
```

トランザクションごとに、`list` サブコマンドは、トランザクション ID、状態、ユーザー名、メッセージまたは通知の数、および作成時間を返します。たとえば、次のように指定します。

```
-----------------------------------------------------------------
Transaction ID  State     User name   # Msgs/   Creation time
                                      # Acks
-----------------------------------------------------------------

64248349708800  PREPARED  guest       4/0       1/30/02 10:08:31 AM
64248371287808  PREPARED  guest       0/4       1/30/02 10:09:55 AM
```

このコマンドを使用すると、ブローカ内のローカルと分散の両方のトランザクションがすべて表示されます。PREPARED 状態のトランザクションだけをコミット、またはロールバックすることができます。これを実行するのは、障害の発生でトランザクションが PREPARED 状態になり、分散トランザクションマネージャーによってコミットされるプロセスになっていないことがわかっている場合だけです。

たとえば、ブローカの自動ロールバックプロパティーを false に設定した場合 (298 ページの表 14-2 を参照 )、ブローカの起動時に、PREPARED 状態のトランザクションを手動でコミット、またはロールバックする必要があります。

list サブコマンドは、トランザクションで生成されたメッセージの数とトランザクションで通知されたメッセージの数 (#Msgs/#Acks) も表示します。トランザクションがコミットされるまで、これらのメッセージは配信されず、通知は処理されません。

query サブコマンドを使用すると、同じ情報のほかに、クライアント ID、接続識別子、分散トランザクション ID (XID) などの多数の追加された値を確認できます。次に示すのは、query txn サブコマンドの構文です。

```
imqcmd query txn -n transactionID
```

たとえば、次の例では以下のような出力が生成されます。

```
imqcmd query txn -n 64248349708800
```

```
Client ID
Connection                  guest@192.18.116.219:62209->jms:62195
Creation time               1/30/02 10:08:31 AM
Number of acknowledgments 0
Number of messages          4
State                       PREPARED
Transaction ID              64248349708800
User name                   guest
XID
6469706F6C7369646577696E6465723130313234313431313030373230
```

分散トランザクションをコミット、またはロールバックするには、commit サブコマンドと rollback サブコマンドを使用します。前述したように、PREPARED 状態のトランザクションだけをコミット、またはロールバックできます。

次に示すのは、commit サブコマンドの構文です。

```
imqcmd commit txn -n transactionID
```

たとえば、次のように指定します。

```
imqcmd commit txn -n 64248349708800
```

次に示すのは、rollback サブコマンドの構文です。

```
imqcmd rollback txn -n transactionID
```

詳細は、298 ページの表 14-2 の `imq.transaction.autorollback` プロパティーを参照してください。

ブローカの起動時に、`PREPARED` 状態のトランザクションが自動的にロールバックされるように、ブローカを設定することも可能です。

第 6 章

# 物理的送信先の管理

この章では、imqcmd ユーティリティーを使用して、物理的送信先を管理する方法を説明します。Message Queue メッセージは、ブローカ上の物理的送信先によりコンシューマクライアントにルーティングされます。ブローカは物理的送信先に関連したメモリーと持続ストレージを管理し、その動作を設定します。

クラスタで、1 つのブローカ上に物理的送信先を作成すると、クラスタはその物理的送信先をすべてのブローカに伝えます。アプリケーションクライアントは、トピックにサブスクライブするか、クラスタ内の任意のブローカにあるキューから消費できます。これはブローカが共同作業でクラスタ間のメッセージをルーティングするためです。ただし、最初にメッセージが生成されたブローカだけは、そのメッセージの持続性と通知を管理します。

この章では、次のトピックについて説明します。



表 13-5 に、物理的送信先を管理し、そのタスクを実行するための imqcmd サブコマンドに関する詳細を示します。

物理的送信先の概要については、『Message Queue 技術の概要』を参照してください。

| 注 | クライアントアプリケーションは、物理的送信先と対話する場合は常に Destination オブジェクトを使用します。プロバイダへの非依存性と移植性のために、クライアントは通常は管理者が作成した送信先オブジェクトを使用し、これは送信先管理対象オブジェクトと呼ばれます。第 8 章「管理対象オブジェクトの管理」で説明するように、管理対象オブジェクトはクライアントアプリケーションで使用できるように設定できます。 |
|---|---|

# コマンドユーティリティーの使用

Message Queue コマンドユーティリティー (imqcmd) を使用して、物理的送信先を管理します。imqcmd コマンドの構文は、ほかのブローカサービスの管理に使用する場合と同じです。

imqcmd とそのサブコマンド、オプションについての詳細は、271 ページの第 13 章「コマンド行のリファレンス」で説明しています。

## サブコマンド

表 6-1 には imqcmd サブコマンドが掲載されています。この章では、その使用法について説明します。これらのサブコマンドの詳細は、283 ページの「物理的送信先管理」を参照してください。

表 6-1　コマンドユーティリティーの物理的送信先のサブコマンド

| サブコマンドと引数 | 説明 |
|---|---|
| compact dst | 1 つ以上の物理的送信先に対応するファイルベースのデータストアを圧縮します。 |
| create dst | 物理的送信先を作成します。 |
| destroy dst | 物理的送信先を廃棄します。 |
| list dst | ブローカの物理的送信先を一覧表示します。 |
| metrics dst | 物理的送信先のメトリックスを表示します。 |
| pause dst | ブローカの 1 つ以上の物理的送信先を停止します。 |
| purge dst | 物理的送信先のすべてのメッセージを、物理的送信先を破棄せずに消去します。 |
| query dst | 物理的送信先の情報をクエリーおよび表示します。 |
| resume dst | ブローカの 1 つ以上の停止された物理的送信先を再開します。 |

表 6-1　コマンドユーティリティーの物理的送信先のサブコマンド ( 続き )

| サブコマンドと引数 | 説明 |
|---|---|
| update dst | 送信先のプロパティーを更新します。 |

# 物理的送信先の作成

物理的送信先を作成するには、imqcmd create サブコマンドを使用します。次に示すのは、create サブコマンドの構文です。

```
create dst -t destType -n destName [-o property=value] [-o property=value1]...
```

物理的送信先を作成するときには、次の情報を指定する必要があります。

- 物理的送信先のタイプ。t ( トピック )、または q ( キュー ) のいずれか。
- 物理的送信先名。次のような命名規則がある。
  - 名前には英数字のみを使用する。スペースは使用できない。
  - 名前は英字、下線文字 (_)、ドル記号 ($) のいずれかで始める。文字列「mq.」で開始することはできない。
- 物理的送信先のプロパティーには、デフォルト以外の値を指定する。

また、物理的送信先を更新する場合、プロパティーも設定できます。

物理的送信先の多くのプロパティーが、ブローカのメモリーリソースおよびメッセージフローに影響します。たとえば、物理的送信先に送信できるプロデューサの数、送信可能なメッセージの数とサイズ、および物理的送信先の制限に達したときにブローカが行う応答を指定できます。この制限は、ブローカの設定プロパティーによって制御されるブローカ全体の制限に似ています。

次のプロパティーは、キューの送信先とトピックの送信先のいずれにも使用します。

- maxNumMsgs。物理的送信先で許容されるコンシューマ配信されないメッセージの最大数を指定します。
- maxTotalMsgBytes。物理的送信先でコンシューマ配信されないメッセージ用として許容されるメモリーの最大量をバイト単位で指定します。
- limitBehavior。メモリー制限のしきい値に達したときのブローカの応答方法を指定します。
- maxBytesPerMsg。物理的送信先で許容されるシングルメッセージの最大サイズをバイト単位で指定します。
- maxNumProducers。物理的送信先のプロデューサの最大数を指定します。
- consumerFlowLimit。1 つのバッチでコンシューマに配信されるメッセージの最大数を指定します。

- isLocalOnly。ブローカクラスタに対してのみ適用します。物理的送信先がそのほかのブローカに複製されないように指定します。つまり、メッセージの配信をローカルコンシューマ ( 物理的送信先の作成元にあるブローカに接続されたコンシューマ ) だけに制限します。
- useDMQ。物理的送信先のデッドメッセージを破棄するか、デッドメッセージのキューに配置するかを指定します。

次のプロパティーは、キューの送信先にのみ使用します。

- maxNumActiveConsumers。ロードバランスされたキュー送信先からの配信でアクティブにできるコンシューマの最大数を指定します。
- maxNumBackupConsumers。キュー送信先からのロードバランスされた配信で障害が生じた場合に、アクティブコンシューマに代わることができるバックアップコンシューマの最大数を指定します。
- localDeliveryPreferred。ブローカクラスタ内のロードバランスされたキュー配信にのみ適用します。ローカルブローカ上にコンシューマが存在しない場合にだけ、メッセージがリモートコンシューマに配信されるように指定します。

物理的送信先のプロパティーについての詳細は、325 ページの第 15 章「物理的送信先のプロパティーのリファレンス」を参照してください。

自動作成される送信先の場合は、ブローカのインスタンス設定ファイルにデフォルトのプロパティー値を設定します。自動作成されるプロパティーの詳細を、299 ページの表 14-3 に示しています。

### ➤ 物理的送信先を作成する

- キューの送信先を作成するには、次のようなコマンドを入力します。

```
imqcmd create dst -n myQueue -t q -o "maxNumActiveConsumers=5"
```

- トピックの送信先を作成するには、次のようなコマンドを入力します。

```
imqcmd create dst -n myTopic -t t -o "maxBytesPerMsg=5000"
```

# 物理的送信先の一覧表示

物理的送信先の現在のプロパティー値、物理的送信先に関連付けられているプロデューサまたはコンシューマの数、物理的送信先内のメッセージの数とサイズなどのメッセージングメトリックスに関する情報を取得できます。

情報を入手する物理的送信先を探す場合は、`list dst` サブコマンドを使用して、ブローカのすべての物理的送信先を一覧表示します。次に示すのは、`list dst` サブコマンドの構文です。

```
list dst [-t destType] [-tmp]
```

このコマンドは、指定されたタイプの物理的送信先を一覧表示します。送信先のタイプ (`-t`) オプションの値は、`q` ( キュー ) または `t` ( トピック ) のいずれかになります。

送信先のタイプを指定しない場合は、すべてのタイプの物理的送信先が一覧表示されます。

`list dst` サブコマンドを使用すると、任意で、一覧表示する送信先のタイプを指定したり、一時的送信先を含めたりすることができます (`-tmp` オプションを使用 )。一時的送信先はクライアントによって作成され、通常は、そのほかのクライアントへ送信されたメッセージへの返信を受信することを目的としています。

たとえば、`myHost` のポート `4545` 上で実行しているブローカ上の物理的送信先すべてのリストを取得するには次のコマンドを入力します。

```
imqcmd list dst -b myHost:4545
```

送信先のタイプ `t` でトピックのみを指定している場合を除き、デッドメッセージキューの情報 `mq.sys.dmq` がほかの物理的送信先と一緒に常に表示されます。

# 物理的送信先の情報の表示

物理的送信先の現在のプロパティーに関する情報を入手するには、query dst サブコマンドを使用します。次に示すのは、query dst サブコマンドの構文です。

```
query dst -t destType -n destName
```

このコマンドは、特定のタイプと名前の送信先に関する情報を一覧表示します。たとえば、次のコマンドはキューの送信先 XQueue に関する情報を表示します。

```
imqcmd query dst -t q -n XQueue -u admin
```

このコマンドでは、次のような情報が出力されます。

```
-------------------------------------
Destination Name    Destination Type
-------------------------------------
XQueue              Queue

On the broker specified by:

-------------------------
Host         Primary Port
-------------------------
localhost    7676

Destination Name                      XQueue
Destination Type                      Queue
Destination State                     RUNNING
Created Administratively              true

Current Number of Messages            0
Current Total Message Bytes           0
Current Number of Producers           0
Current Number of Active Consumers    0
Current Number of Backup Consumers    0

Max Number of Messages                unlimited (-1)
Max Total Message Bytes               unlimited (-1)
Max Bytes per Message                 unlimited (-1)
Max Number of Producers               100
Max Number of Active Consumers        1
Max Number of Backup Consumers        0

Limit Behavior                        REJECT_NEWEST
Consumer Flow Limit                   1000
Is Local Destination                  false
Local Delivery is Preferred           false
Use Dead Message Queue                true
```

また、出力は送信先に関連付けられたプロデューサとコンシューマの数を示しています。キューの送信先について、数字にはアクティブなコンシューマとバックアップコンシューマが含まれます。

update dst サブコマンドを使用すると、1 つ以上のプロパティーの値を変更できます (125 ページの「物理的送信先のプロパティーの更新」を参照 )。

# 物理的送信先のプロパティーの更新

物理的送信先のプロパティーを変更するには、update dst サブコマンドと -o オプションを使用して、更新するプロパティーを指定します。次に示すのは、update dst サブコマンドの構文です。

```
update dst -t destType -n destName -o property=value [[-o property=value1]...]
```

このコマンドは、指定した送信先の特定のプロパティー値を更新します。プロパティー名は、表 15-1 で説明しているいずれかのプロパティーになります。

複数の -o オプションを使用すると、複数のプロパティーを更新できます。たとえば、次のコマンドでは maxBytesPerMsg プロパティーが 1000 に、MaxNumMsgs プロパティーが 2000 にそれぞれ変更されます。

```
imqcmd update dst -t q -n myQueue -o "maxBytesPerMsg=1000"
              -o "maxNumMsgs=2000" -u admin
```

更新が可能なプロパティーについては、第 15 章「物理的送信先のプロパティーのリファレンス」を参照してください。

物理的送信先の *type* や isLocalOnly プロパティーを更新する場合、update dst サブコマンドは使用できません。

**注** デッドメッセージキューは、特殊な物理的送信先であり、プロパティーがその他の送信先のプロパティーと多少異なります。詳細は、130 ページの「デッドメッセージキューの使用の設定」を参照してください。

# 物理的送信先の停止と再開

プロデューサから送信先、送信先からコンシューマ、またはその両方のメッセージの配信を制御するために、物理的送信先を停止できます。特に、メッセージの生成が消費よりかなり高速な場合に、送信先がメッセージによって過負荷にならないように、送信先へのメッセージフローを停止できます。圧縮する前に物理的送信先を停止する必要があります。

物理的送信先との間で配信されるメッセージを停止するには、pause dst サブコマンドを使用します。次に示すのは、pause dst サブコマンドの構文です。

```
pause dst [-t destType -n destName] [-pst pauseType]
```

このサブコマンドは、特定のタイプと名前の送信先について、コンシューマへのメッセージ (-pst CONSUMERS)、プロデューサからのメッセージ (-pst PRODUCERS)、またはその両方 (-pst ALL) を停止します。送信先のタイプと名前が指定されない場合、すべての物理的送信先が停止します。デフォルト値は ALL です。

例 :

```
imqcmd pause dst -n myQueue -t q -pst PRODUCERS -u admin
```

```
imqcmd pause dst -n myTopic -t t -pst CONSUMERS -u admin
```

停止した送信先への配信を再開するには、resume dst サブコマンドを使用します。次に示すのは、resume dst サブコマンドの構文です。

```
resume dst [-t destType -n destName]
```

このサブコマンドは、特定のタイプと名前の停止された送信先についてメッセージの配信を再開します。送信先のタイプと名前が指定されていない場合は、すべての送信先が再開されます。

例 :

```
imqcmd resume dst -n myQueue -t q
```

ブローカクラスタでは、物理的送信先のインスタンスはクラスタ内の各ブローカに常駐します。各インスタンスを個別に停止する必要があります。

# 物理的送信先の消去

物理的送信先のキューに現在入っているメッセージは、すべて消去することが可能です。物理的送信先を消去すると、送信先に保存されているすべてのメッセージが削除されます。

累積されたメッセージによって、システムのリソースが大幅に消費される場合に、これらのメッセージを消去することができます。これは、登録済みのコンシューマクライアントがキューに入っていない場合やキューが多数のメッセージを受信する場合に発生する可能性があります。また、トピックの永続サブスクライバが、アクティブにならない場合にも発生する可能性があります。どちらの場合も、メッセージが必要以上に保持されます。

物理的送信先でメッセージを消去するには、`purge dst` サブコマンドを使用します。次に示すのは、`purge dst` サブコマンドの構文です。

```
purge dst -t destType -n destName
```

このサブコマンドは、特定のタイプと名前の物理的送信先のメッセージを消去します。

例 :

```
imqcmd purge dst -n myQueue -t q -u admin

imqcmd purge dst -n myTopic -t t -u admin
```

ブローカをシャットダウンした後、再起動するときに、古いメッセージを配信する必要がない場合は、`-reset messges` オプションを使用して、古いメッセージを消去します。たとえば、次のとおりです。

```
imqbrokerd -reset messages -u admin
```

これで、ブローカを再起動すると、送信先の消去に関する問題が解消されます。

ブローカクラスタでは、物理的送信先のインスタンスはクラスタ内の各ブローカに常駐します。これらの送信先はそれぞれ個別に消去する必要があります。

# 物理的送信先の破棄

物理的送信先を破棄するには、destroy dst サブコマンドを使用します。次に示すのは、destroy dst サブコマンドの構文です。

```
destroy dst -t destType -n destName
```

このサブコマンドは、特定のタイプと名前の物理的送信先のメッセージを破棄します。

例 :

```
imqcmd destroy dst -t q -n myQueue -u admin
```

物理的送信先を破棄すると、その送信先のすべてのメッセージが消去され、ブローカからその送信先がなくなるため、操作を元に戻すことはできません。

デッドメッセージキューを破棄することはできません。

# 物理的送信先の圧縮

メッセージの持続ストアとして、ファイルベースのデータストアを使用している場合は、ディスク利用率を監視し、必要に応じてディスクを圧縮できます。

ファイルベースのメッセージストアは、保持される物理的送信先に応じてメッセージがディレクトリに格納されるように構成されています。各物理的送信先のディレクトリでは、大半のメッセージが可変長のレコードから成る 1 つのファイル、つまり可変長のレコードファイルに格納されます。断片化を減らすため、サイズが設定可能なしきい値を超えているメッセージは専用の個別のファイルに格納されます。

可変サイズのメッセージが保持されていて、その後可変長のレコードファイルから削除された場合、空きレコードが再利用されていないファイルに空白ができることがあります。

未使用の空きレコードを管理するために、コマンドユーティリティーには、物理的送信先ごとにディスク利用率を監視したり、利用率の低下時に空きディスクスペースを再利用したりするためのサブコマンドが含まれています。

## 物理的送信先のディスク利用率の監視

物理的送信先のディスク利用率を監視するには、次のようなコマンドを使用します。

```
imqcmd metrics dst -t q -n myQueue -m dsk -u admin
```

このコマンドでは、次のような情報が出力されます。

```
--------------------------------------
Reserved   Used      Utilization Ratio
--------------------------------------
806400     804096    99
1793024    1793024   100
2544640    2518272   98
```

サブコマンド出力の各列の意味は次のとおりです。

表 6-2　物理的送信先ディスク利用率のメトリックス

| メトリックス | 説明 |
|---|---|
| Reserved ( 予約済み ) | すべてのレコードによって使用されるディスクスペース ( バイト単位 )。アクティブメッセージを保持するレコードと再利用可能な空きレコードが含まれます。 |
| Used ( 使用中 ) | アクティブメッセージを保持しているレコードによって使用されるディスクスペース ( バイト単位 )。 |
| Utilization Ratio ( 利用率 ) | 使用されているディスクスペースを予約済みのディスクスペースで割ったときの商。割合が高いほど、アクティブメッセージを保持するためにより多くのディスクスペースが使用されています。 |

## 未使用の物理的送信先ディスクスペースの再利用

ディスク利用率のパターンは、特定の物理的送信先を使用しているメッセージングアプリケーションの特性によって異なります。また、物理的送信先との間でやり取りされる相対的なメッセージフローとメッセージの相対的なサイズに応じて、時間の経過とともに予約済みディスクスペースが拡大することがあります。

メッセージの生成レートがメッセージの消費レートを上回る場合は、一般に、空きレコードが再利用され利用率が高くなります。ただし、メッセージの生成レートがメッセージの消費レートと同程度かそれより低い場合は、利用率は低いと予測できます。

一般に、予約済みディスクスペースは安定化させ、利用率は高いまま維持させる必要があります。一般的に、システムが安定して、予約済みディスクスペースがほぼ一定になり利用率が高い (75% を超える ) 状態に達した場合には、未使用のディスクスペースを再利用する必要はありません。システムが安定した状態になったが利用率が低い (50% を下回る ) 場合は、ディスクを圧縮し、空きレコードが占有しているディスクスペースを再利用できます。

データストアを圧縮する場合は、compact dst サブコマンドを使用します。次に示すのは、compact dst サブコマンドの構文です。

```
compact dst [-t destType -n destName]
```

このサブコマンドは、特定のタイプと名前の物理的送信先に対応するファイルベースのデータストアを圧縮します。送信先のタイプと名前が指定されていない場合は、すべての送信先が圧縮されます。圧縮する前に、物理的送信先を停止する必要があります。

予約済みのディスクスペースが時間の経過とともに増え続けている場合は、送信先メモリーの制限プロパティーと制限動作を設定して送信先のメモリー管理を設定し直す必要があります (325 ページの表 15-1 を参照 )。

### ➤ 未使用の物理的送信先ディスクスペースを再利用する

1. 送信先を停止します。

   ```
   imqcmd pause dst -t q -n myQueue -u admin
   ```

2. ディスクを圧縮します。

   ```
   imqcmd compact dst -t q -n myQueue -u admin
   ```

3. 物理的送信先を再開します。

   ```
   imqcmd resume dst -t q -n myQueue -u admin
   ```

送信先のタイプと名前が指定されなかった場合、これらの操作はすべての物理的送信先に対して実行されます。

# デッドメッセージキューの使用の設定

デッドメッセージキュー mq.sys.dmq は、ブローカとブローカのその他の物理的送信先のデッドメッセージを保持する、システムで生成された物理的送信先です。デッドメッセージキューは、監視、システムの効率性の調整、トラブルシューティングに使用するツールです。「デッドメッセージ」の定義と、デッドメッセージキューの概要については、『Message Queue 技術の概要』を参照してください。

ブローカは起動時に自動的にデッドメッセージキューを作成します。ブローカは処理できないメッセージ、または生存期間を過ぎたメッセージを、キューに配置します。さらに、その他の物理的送信先が廃棄したメッセージの保持にデッドメッセージキューを使用することもあります。デッドメッセージキューを使用することで、システムのトラブルシューティングに有益な情報がもたらされます。

# デッドメッセージキューの使用の設定

デフォルトでは、物理的送信先は、デッドメッセージキューを有効に設定しています。物理的送信先がデッドメッセージキューを使用しないように設定できます。あるいは物理的送信先プロパティー useDMQ を設定して有効にすることもできます。

次の例では、デフォルトでデッドメッセージキューを使用する、myDist と呼ばれるキューが作成されます。

```
imqcmd create dst -n -myDist -t q
```

次の例では、同じキューに対してデッドメッセージキューの使用が無効になります。

```
imqcmd update dst -n myDist -t q -o useDMQ=false
```

ブローカ上の自動作成されたすべての物理的送信先で、デッドメッセージキューの使用を有効にしたり、imq.autocreate.destination.useDMQ ブローカプロパティーを設定して、デッドメッセージキューの使用を無効にしたりできます。

# デッドメッセージキューを設定し管理する

Message Queue コマンドユーティリティー (imqcmd) を使用して、デッドメッセージキューをほかのキューと同じように管理できますが、いくつかの相違点があります。たとえば、デッドメッセージキューはシステムで生成されるため、作成、停止、破棄の操作は行えません。さらに、表 6-3 に示すように、デッドメッセージキューのデフォルト値は通常のキューと異なる場合があります。

## デッドメッセージキューのプロパティー

デッドメッセージキューは、ほかのキューの設定と同様に設定しますが、特定の物理的送信先のプロパティーは適用されません。あるいは別のデフォルト値が指定されます。表 6-3 にデッドメッセージキューが独自の方法で処理するキュープロパティーを一覧表示しています。

**表 6-3** 標準の物理的送信先プロパティーのデッドメッセージキューの処理

| プロパティー | デッドメッセージキューによる固有の処理 |
|---|---|
| limitBehavior | デッドメッセージキューのデフォルト値は、REMOVE_OLDEST です。その他のキューのデフォルト値は REJECT_NEWEST です。デッドメッセージキューでは、フロー制御はサポートされません。 |
| localDeliveryPreferred | デッドメッセージキューに適用されません。 |
| maxNumMsgs | デッドメッセージキューのデフォルト値は、1000 です。その他のキューのデフォルト値は -1 ( 無制限 ) です。 |

表 6-3　標準の物理的送信先プロパティーのデッドメッセージキューの処理 ( 続き )

| プロパティー | デッドメッセージキューによる固有の処理 |
|---|---|
| maxNumProducers | デッドメッセージキューに適用されません。 |
| maxTotalMsgBytes | デッドメッセージキューのデフォルト値は、10M バイトです。その他のキューのデフォルト値は -1 ( 無制限 ) です。 |
| isLocalOnly | ブローカクラスタで、デッドメッセージキューは常にローカルの物理的送信先になり、このプロパティーは永続的に true に設定されます。ただし、ローカルブローカがメッセージをデッドメッセージとマークしている場合、ローカルブローカのデッドメッセージキューには、クラスタのほかのブローカのクライアントが生成したメッセージが格納される場合があります。 |

### メッセージの内容

ブローカはメッセージ全体をデッドメッセージキューに配置できます。あるいはヘッダーとプロパティーデータのみを残して、メッセージ本体の内容を破棄できます。デフォルトでは、デッドメッセージキューはメッセージ全体を格納します。

デッドメッセージキューのサイズを減らし、デッドメッセージを復元する予定がない場合は、imq.destination.DMQ.truncateBody ブローカプロパティーを true に設定することを検討してください。

```
imqcmd update bkr -o imq.destination.DMQ.truncateBody=true
```

これにより、メッセージ本体が破棄され、ヘッダーとプロパティーデータのみが残されます。

## デッドメッセージのロギングを有効にする

デッドメッセージのロギングは、デフォルトでは無効になっています。デッドメッセージのロギングを有効にすると、ブローカが次のイベントを記録するようになります。

- ブローカがデッドメッセージキューにメッセージを移動する
- ブローカがデッドメッセージキューとデッドメッセージキューを使用していない物理的送信先からメッセージを破棄する
- 物理的送信先が制限に達する

次のコマンドでは、デッドメッセージのロギングを有効にしています。

```
imqcmd update bkr -o imq.destination.logDeadMsgs=true
```

デッドメッセージのロギングは、デッドメッセージキューを使用するすべての物理的送信先に適用されます。物理的送信先の個々については、ロギングを有効または無効に設定できません。

第 7 章

# セキュリティーの管理

管理者は、ユーザーの認証のためのユーザーリポジトリの設定、アクセス制御の定義、クライアントブローカ間の通信を暗号化する SSL (Secure Socket Layer) 接続サービスの設定、ブローカ起動時に使用するパスワードファイルの設定を行います。

この章では、次の節について説明します。



## ユーザーの認証

ユーザーがブローカへの接続を試みると、ブローカは提供された名前とパスワードを調べて、ユーザーを認証します。ブローカは、その名前とパスワードが、それぞれ参照するように設定されているブローカ固有のユーザーリポジトリにある名前とパスワードに一致した場合に、接続を許可します。

管理者は、ユーザー、ユーザーグループ、およびパスワードのリストをユーザーリポジトリに保持しておく責任があります。ブローカインスタンスごとに異なるユーザーリポジトリを使用できます。この節では、リポジトリの作成、設定、および管理の方法を説明します。

リポジトリは次のいずれかのタイプになります。

- Message Queue に付属している単層型ファイルリポジトリ

このタイプのユーザーリポジトリは、非常に簡単に扱えます。ユーザーマネージャーユーティリティー (imqusermgr) を使用してリポジトリを設定および管理します。認証を有効にするには、ユーザーリポジトリにユーザー名、パスワード、およびユーザーグループの名前を設定します。

ユーザーリポジトリの設定と管理の詳細については、「単層型ファイルユーザーリポジトリを使用する」を参照してください。

- LDAP サーバー

  このリポジトリは、LDAP v2 または v3 プロトコルを使用する既存または新規の LDAP ディレクトリサーバーです。単層型ファイルリポジトリほど使用方法は簡単ではありませんが、よりスケーラブルなため、本稼動環境に適しています。

  LDAP ユーザーリポジトリを使用している場合、LDAP ベンダーから提供されているツールを使用して、ユーザーリポジトリを設定、管理します。詳細は、142 ページの「ユーザーリポジトリに LDAP サーバーを使用する」を参照してください。

# 単層型ファイルユーザーリポジトリを使用する

Message Queue には、単層型ファイルユーザーリポジトリ、コマンド行ツール、および単層型ファイルユーザーリポジトリの設定と管理ができるユーザーマネージャーユーティリティー (imqusermgr) が用意されています。次の節では、単層型ファイルユーザーリポジトリと、そのリポジトリを設定し管理するユーザーマネージャーユーティリティーの使用方法について説明します。

## ユーザーリポジトリの作成

単層型ファイルユーザーリポジトリは、インスタンス固有です。起動するブローカインスタンスごとに、passwd という名前のデフォルトのユーザーリポジトリが自動的に作成されます。このユーザーリポジトリは、そのリポジトリが関連付けられているブローカインスタンスの名前によって識別されたディレクトリに書き込まれます ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

```
.../instances/instanceName/etc/passwd
```

リポジトリは 2 つのエントリから構成されます。表 7-1 の各行にエントリを示します。

**表 7-1** ユーザーリポジトリでの初期エントリ

| ユーザー名 | パスワード | グループ | 状態 |
|---|---|---|---|
| admin | admin | admin | アクティブ |
| guest | guest | anonymous | アクティブ |

これらの初期エントリにより、管理者が介入しなくても、インストール後直ちに Message Queue ブローカを使用できます。

- 初期設定された guest ユーザーエントリを使って、クライアントはデフォルトの guest ユーザー名とパスワードでブローカインスタンスに接続できます。
- 初期設定された admin ユーザーエントリでは、デフォルトのユーザー名とパスワード admin で、imqcmd コマンドを使用してブローカを管理できます。この初期エントリを更新して、パスワードを変更する必要があります (141 ページの「デフォルトの管理者パスワードの変更」を参照 )。

次の節では、単層型ユーザーリポジトリの設定、および管理方法について説明します。

## ユーザーマネージャーユーティリティー

Message Queue ユーザーマネージャーユーティリティー (imqusermgr) を使って、単層型ファイルユーザーリポジトリを編集したり設定できます。この節では、ユーザーマネージャユーティリティーについて説明します。後続の節では、imqusermgr サブコマンドを使用して、特定のタスクを実行する方法について説明します。

imqusermgr コマンドの詳細は、第 13 章「コマンド行のリファレンス」を参照してください。

ユーザーマネージャーの使用の先立ち、次の点に留意してください。

- ブローカ固有のユーザーリポジトリが存在していない場合は、それを作成するために該当するブローカインスタンスを起動する必要があります。
- imqusermgr コマンドは、ブローカがインストールされているホスト上で実行する必要があります。
- リポジトリへの書き込みに関する適切なアクセス権を持つ必要があります。たとえば、Solaris と Linux の場合、root ユーザーまたはブローカインスタンスを最初に作成したユーザーになる必要があります。

**注**　次の節の例は、デフォルトのブローカインスタンスを前提としています。

### *サブコマンド*

imqusermgr コマンドには、add、delete、list、update といったサブコマンドがあります。

- add サブコマンドは、ユーザーとそのパスワードを指定した、またはデフォルトのブローカインスタンスリポジトリに追加し、オプションでユーザーグループを指定します。このサブコマンドの構文は次のようになります。

  add [-i *instanceName*] -u *userName* -p *passwd* [-g *group*] [-s]

- `delete` サブコマンドは、指定したユーザーを、指定した、またはデフォルトのブローカインスタンスリポジトリから削除します。このサブコマンドの構文は次のようになります。

  `delete [-i` *instanceName*`] -u` *userName* `[-s] [-f]`

- `list` サブコマンドは、指定した、またはデフォルトのブローカインスタンスリポジトリの指定したユーザーまたはすべてのユーザーに関する情報を表示します。このサブコマンドの構文は次のようになります。

  `list [-i` *instanceName*`] [-u` *userName*`]`

- `update` サブコマンドは、指定した、またはデフォルトのブローカインスタンスリポジトリの指定ユーザーのパスワードまたは状態、もしくは両方を更新します。このサブコマンドの構文は次のようになります。

  `update [-i` *instanceName*`] -u` *userName* `-p` *passwd* `[-a` *state*`] [-s] [-f]`

  `update [-i` *instanceName*`] -u` *userName* `-a` *state* `[-p` *passwd*`] [-s] [-f]`

## *コマンドオプション*

表 7-2 に `imqusermgr` コマンドのオプションを一覧表示します。

**表 7-2** `imqusermgr` オプション

| オプション | 説明 |
|---|---|
| `-a` *activeState* | ユーザーの状態をアクティブにするかどうかを指定します (`true/false`)。値が `true` の場合、状態はアクティブです。デフォルト値 は **true** です。 |
| `-f` | ユーザーの確認なしで、アクションを実行します。 |
| `-h` | 使用方法に関するヘルプを表示します。コマンド行ではそれ以外のことは実行されません。 |
| `-i` *instanceName* | コマンドを適用するブローカインスタンス名を指定します。指定しない場合は、デフォルトのインスタンス名 `imqbroker` が使用されます。 |
| `-p` *passwd* | ユーザーのパスワードを指定します。 |
| `-g` *group* | ユーザーグループを指定します。指定できる値は、`admin`、`user`、`anonymous` です。 |
| `-s` | サイレントモードに設定します。 |
| `-u` *userName* | ユーザー名を指定します。 |
| `-v` | バージョン情報を表示します。コマンド行ではそれ以外のことは実行されません。 |

## グループ

ブローカインスタンスのユーザーリポジトリにユーザーエントリを追加する場合、事前に定義された 3 つのグループである admin、user、anonymous のいずれかを指定できます。グループが指定されない場合、デフォルトのグループ user が割り当てられます。グループが次のように割り当てられるはずです。

- **admin グループ**: ブローカの管理者用です。このグループに割り当てられたユーザーは、デフォルトでブローカを設定および管理できるようになっています。管理者は、複数のユーザーを admin グループに割り当てることができます。
- **user グループ**: 管理ユーザーでない通常の Message Queue クライアントユーザー用です。ほとんどのクライアントユーザーは user グループに所属します。デフォルトでは、このグループのユーザーはすべてのトピックとキューへのメッセージを生成し、すべてのトピックとキューからのメッセージを消費し、すべてのキューのメッセージを検索します。
- **anonymous グループ**: ブローカが認識しているユーザー名を使用しない Message Queue クライアント用です。クライアントアプリケーションが実際に使用するユーザー名を認識していないなどの理由がある場合に使います。このアカウントは、多くの FTP サーバーにある匿名アカウントに似ています。一度に anonymous グループに割り当てられるのは、1 人のユーザーだけです。このグループのアクセス権限を user グループよりも制限したり、配置時にグループからユーザーを削除する必要があります。

ユーザーが属するグループを変更するには、そのユーザーのエントリを削除してから、そのユーザーの別のエントリを追加し、新しいグループを指定します。

このようなシステムで生成されたグループの名前の変更や削除、および新しいグループの作成は行えません。ただし、そのグループのメンバーがどの操作を実行するかを定義するアクセス規則を指定できます。詳細は、144 ページの「ユーザーの承認 : アクセス制御プロパティーファイル」を参照してください。

## ユーザーの状態

ユーザーをリポジトリに追加した場合、デフォルトのユーザーの状態はアクティブです。ユーザーを非アクティブに変更するには、更新コマンドを使用する必要があります。たとえば、次のコマンドでは、ユーザーの JoeD が非アクティブになります。

```
imqusermgr update -u JoeD -a false
```

非アクティブになったユーザーのエントリは、リポジトリに保持されますが、非アクティブのユーザーは、新規接続を開くことはできません。ユーザーが非アクティブのときに、同じ名前を持つ別のユーザーを追加すると、操作に障害が発生します。非アクティブなユーザーのエントリを削除するか、新しいユーザーのユーザー名を変更するか、あるいは新しいユーザーに別のユーザー名を使用する必要があります。このようにして、重複するユーザー名が追加されるのを防ぎます。

## ユーザー名とパスワードの形式

ユーザー名とパスワードは、次の規則に従う必要があります。

- ユーザー名にアスタリスク (*)、コンマ (,)、コロン (:)、改行やキャリッジリターン文字を使用することはできません。
- ユーザー名やパスワードは、1 文字以上であることが必要です。
- ユーザー名やパスワードに空白を入れる場合は、ユーザー名やパスワード全体を引用符で囲みます。
- コマンド行で入力可能な最大文字数で、コマンドシェルにより制限されない限り、パスワードやユーザー名の長さに制限はありません。

## ユーザーリポジトリの設定と管理

add サブコマンドを使用して、ユーザーをリポジトリに追加します。たとえば、次のコマンドでは、sesame というパスワードを持つ Katharine というユーザーがデフォルトのブローカインスタンスユーザーリポジトリに追加されます。

```
imqusermgr add -u Katharine -p sesame -g user
```

delete サブコマンドを使用して、ユーザーをリポジトリから削除します。たとえば、次のコマンドでは Bob というユーザーが削除されます。

```
imqusermgr delete -u Bob
```

update サブコマンドを使用して、ユーザーのパスワードまたは状態を変更します。たとえば、次のコマンドでは、Katharine のパスワードが alladin に変更されます。

```
imqusermgr update -u Katharine -p aladdin
```

1 人またはすべてのユーザーに関する情報を一覧表示するには、list コマンドを使用します。次のコマンドでは、isa という名前のユーザーに関する情報が表示されます。

```
imqusermgr list -u isa
```

```
% imqusermgr list -u isa

User repository for broker instance: imqbroker
----------------------------------
User Name    Group     Active State
----------------------------------
isa          admin    true
```

次のコマンドでは、すべてのユーザーに関する情報が表示されます。

```
imqusermgr list
```

```
% imqusermgr list

User repository for broker instance: imqbroker
-------------------------------------
User Name         Group     Active State
-------------------------------------
admin         admin         true
guest         anonymous     true
isa           admin         true
testuser1     user           true
testuser2     user           true
testuser3     user           true
testuser4     user           false
testuser5     user           false
```

## デフォルトの管理者パスワードの変更

セキュリティーのために、admin のデフォルトパスワードを自分だけが知っているパスワードに変更する必要があります。次のコマンドは、mybroker ブローカインスタンスのデフォルトの管理者パスワードを admin から grandpoobah に変更します。

```
imqusermgr update mybroker -u admin -p grandpoobah
```

ブローカインスタンスの実行中に任意のコマンド行ツールを実行すれば、この変更が反映されていることをすぐに確認できます。たとえば、次のコマンドはパスワードの入力を要求します。

```
imqcmd list svc mybroker -u admin
```

新しいパスワード (grandpoobah) は入力可能であり、古いパスワードでは失敗します。

パスワードを変更したあとは、管理コンソールなどのあらゆる Message Queue 管理ツールを使用するときに、新しいパスワードを使用してください。

# ユーザーリポジトリに LDAP サーバーを使用する

ユーザーリポジトリに LDAP サーバーを使用する場合、次の作業を実行します。

- インスタンス設定ファイルの編集
- 管理者のアクセス制御の設定

## インスタンス設定ファイルの編集

ブローカでディレクトリサーバーを使用する場合、ブローカインスタンス設定ファイル `config.properties` で特定のプロパティーの値を設定します。これらのプロパティーを使用すると、ユーザーがブローカインスタンスへの接続を試みているか、メッセージング操作を実行しようとしている場合に、ブローカインスタンスが LDAP サーバーに対して、ユーザーやグループに関する情報をクエリーできるようになります。

インスタンス設定ファイルは、ブローカインスタンスディレクトリのディレクトリ内にあります。パスは次のような形式です。

*.../*`instances/`*instanceName*`/props/config.properties`

オペレーティングシステム別のインスタンスディレクトリの場所については、付録 A「プラットフォームごとの Message Queue データの場所」を参照してください。

### ➤ 設定ファイルを編集して、LDAP サーバーを使用する

1. 次のプロパティーを設定して、LDAP ユーザーリポジトリの使用を指定します。

   ```
   imq.authentication.basic.user_repository=ldap
   ```

2. `imq.authentication.type` プロパティーを設定して、クライアントからブローカへのパスワードの受け渡しに、base64 (`basic`) 符号化方式を使用するか、MD5 (`digest`) 符号化方式を使用するかを決定します。LDAP ディレクトリサーバーをユーザーリポジトリに使用する場合、認証タイプに `basic` を設定する必要があります。たとえば、次のように指定します。

   ```
   imq.authentication.type=basic
   ```

3. LDAP へのアクセスを制御するブローカプロパティーを設定する必要もあります。このプロパティーは、ブローカインスタンス設定ファイル内にあります。プロパティーについては、86 ページの「セキュリティーサービス」で説明し、309 ページの「セキュリティーのプロパティー」にまとめています。

   Message Queue は JNDI API を使用して、LDAP ディレクトリサーバーと対話します。このプロパティーで使用されている構文と用語の詳細については JNDI のドキュメントを参照してください。Message Queue は、Sun JNDI LDAP プロバイダと簡単な認証を使用しています。

Message Queue は LDAP 認証のフェイルオーバーをサポートします。認証が試みられる LDAP ディレクトリサーバーのリストを指定できます (imq.user.repos.ldap.server プロパティーの詳細を参照 )。

LDAP ユーザーリポジトリに関連したプロパティーの設定方法の例については、ブローカの config.properties ファイルを参照してください。

4. 必要に応じて、アクセス制御プロパティーファイルにあるユーザーまたはグループ、および規則を編集する必要があります。アクセス制御プロパティーファイルの使用に関する詳細は、144 ページの「ユーザーの承認 : アクセス制御プロパティーファイル」を参照してください。
5. 接続の認証やグループ検索の間、ブローカに SSL を使用して LDAP ディレクトリとの通信を行わせる場合、LDAP サーバーの SSL をアクティブにし、ブローカ設定ファイルで次のプロパティーを設定します。
   - LDAP サーバーが SSL 通信に使用するポートを指定します。たとえば、次のように指定します。

     ```
     imq.user_repository.ldap.server=myhost:7878
     ```

   - ブローカプロパティーの imq.user_repository.ldap.ssl.enabled を true に設定します。

     複数の LDAP ディレクトリサーバーを使用している場合は、ldap:// を使用して、追加の各ディレクトリサーバーを指定します。たとえば、次のように指定します。

     imq.user_repository.ldap.server=*myHost*:7878 ldap://*otherHost*:7878 ...

     追加の各ディレクトリサーバーはスペースで区切ります。リスト内のすべてのディレクトリサーバーは、LDAP 関連プロパティーに同じ値を使用する必要があります。

## 管理者のアクセス制御の設定

管理ユーザーを作成するには、アクセス制御プロパティーファイルで、ADMIN 接続を作成できるユーザーとグループを指定します。これらのユーザーとグループは、LDAP ディレクトリで事前に定義されている必要があります。

ADMIN 接続を作成できるユーザーまたはグループは、管理コマンドを発行できます。

### ➤ 管理ユーザーを設定する

1. アクセス制御ファイルの使用を有効にするには、ブローカプロパティー imq.accesscontrol.enabled を、デフォルト値である true に設定します。

   imq.accesscontrol.enabled プロパティーにより、アクセス制御ファイルの使用が有効になります。

2. アクセス制御ファイル accesscontrol.properties を開きます。このファイルの場所については、付録 A 「プラットフォームごとの Message Queue データの場所」の一覧を参照してください。

   このファイルには、次のようなエントリが収められています。

```
サービス接続アクセス制御
##################################
connection.NORMAL.allow.user=*
connection.ADMIN.allow.group=admin
```

   上記のエントリは一例です。admin グループはファイルベースのユーザーリポジトリに存在しますが、デフォルトでは LDAP ディレクトリに存在しないことに注意してください。LDAP ディレクトリで定義される、Message Queue 管理者権限を付与するグループの名前は変更する必要があります。

3. Message Queue 管理者権限をユーザーに付与するには、ユーザー名を次のように入力します。

   connection.ADMIN.allow.user=*userName*[[,*userName2*]...]

4. Message Queue 管理者権限をグループに付与するには、グループ名を次のように入力します。

   connection.ADMIN.allow.group=*groupName*[[,*groupName2*]...]

# ユーザーの承認 : アクセス制御プロパティーファイル

アクセス制御プロパティーファイル (ACL ファイル ) には、ユーザーおよびユーザーグループがどの操作を実行できるかを指定する規則が収められています。ACL ファイルを編集して、操作を特定のユーザーやグループに制限します。ブローカインスタンスごとに異なる ACL ファイルを使用できます。

ユーザー情報が単層型ファイルのユーザーリポジトリにある場合でも LDAP ユーザーリポジトリにある場合でも、ACL ファイルが使用されます。ブローカは、クライアントアプリケーションが次の操作のいずれかを実行するときに ACL ファイルをチェックします。

- 接続の作成

- プロデューサの作成
- コンシューマの作成
- キューの検索

ブローカは ACL ファイルをチェックして、要求を生成したユーザーまたはユーザーが所属するグループに対して、操作の実行を許可するかどうかを決定します。

ACL ファイルを編集する場合、次にブローカがファイルをチェックして認証を検証するまで、新しい設定は有効になりません。ファイルの編集後、ブローカを再起動する必要はありません。

# アクセス制御プロパティーファイルの作成

ACL ファイルはインスタンス固有です。ブローカインスタンスを起動する場合は常に、インスタンスディレクトリでデフォルトファイル `accesscontrol.properties` が作成されます。ファイルのパスは次のような形式です ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

```
.../instances/brokerInstanceName/etc/accesscontrol.properties
```

ACL ファイルは、Java プロパティーファイルのような形式になっています。ACL ファイルは、ファイルのバージョンを指定すると起動し、次の 3 つのセクションのアクセス制御規則を指定します。

- 接続のアクセス制御
- 物理的送信先のアクセス制御
- 物理的送信先の自動作成アクセス制御

`version` プロパティーでは、ACL プロパティーファイルのバージョンが定義されるので、このエントリを変更しないでください。

```
version=JMQFileAccessControlModel/100
```

アクセス制御を指定する ACL ファイルの 3 つのセクションについては、アクセス規則の基本構文およびアクセス権の計算方法に続いて、説明します。

# アクセス規則の構文

ACL プロパティーファイルでは、アクセス制御は、特定のユーザーやグループが物理的送信先や接続サービスといった保護されたリソースに対してどのアクセスを持っているのかを定義します。アクセス制御は、それぞれ Java プロパティーとして提示されている規則、または規則のセットで表現されます。

これらの規則の基本的な構文は次のとおりです。

*resourceType*.*resourceVariant*.*operation*.*access*.*principalType*=*principals*

表 7-3 に構文規則の各要素を示します。

**表 7-3** アクセス規則の構文要素

| 要素 | 説明 |
|---|---|
| *resourceType* | `connection`、`queue`、`topic` のいずれか |
| *resourceVariant* | *resourceType* で指定されたタイプのインスタンス。たとえば、myQueue。ワイルドカードの文字 (*) を、すべての接続サービス、またはすべての物理的な送信先を表すのに使用できます。 |
| *operation* | 公式化されているアクセス規則の種類に依存する値です。 |
| *access* | `allow` か `deny` のどちらかです。 |
| *principalType* | `user` か `group` のどちらかです。詳細は、139 ページの「グループ」を参照してください。 |
| *principals* | 規則の左側で指定されるアクセス権を保持するユーザーを示します。ここでは、`principalType` が `user` の場合は個々のユーザーまたはコンマで区切られたユーザーのリストとなり、`principalType` が `group` の場合は 1 つのグループまたはコンマで区切られたグループのリストとなります。ワイルドカードの文字 (*) を、すべてのユーザーまたはすべてのグループを表すのに使用できます。 |

ここで、アクセス規則の例をいくつか紹介します。

- 次の規則では、あらゆるユーザーがメッセージを q1 という名前のキューに送信します。

  ```
  queue.q1.produce.allow.user=*
  ```

- 次の規則では、あらゆるユーザーがあらゆるキューにメッセージを送信します。

  ```
  queue.*.produce.allow.user=*
  ```

**注** ASCII でないユーザー、グループ、または送信先の名前を指定するには、Unicode エスケープ (¥uXXXX) の表記法を使用します。ASCII コードではない名前を含む ACL ファイルを編集して保存した場合、Java `native2ascii` ツールを使用して、ファイルを ASCII に変換できます。詳細は、次を参照してください。

http://java.sun.com/j2se/1.4/docs/guide/intl/faq.html

# 権限の計算方法

ファイル内に複数のアクセス規則が存在する場合、権限を次のように計算します。

- 特定のアクセス規則は、一般的なアクセス規則をオーバーライドします。次の 2 つの規則が適用されると、ユーザーはだれでもすべてのキューに送信できますが、Bob は `tq1` に送信できません。

```
queue.*.produce.allow.user=*

queue.tq1.produce.deny.user=Bob
```

- 明示的な *principal* に指定されたアクセスは、* *principal* に指定されたアクセスをオーバーライドします。次の規則で、Bob は `tq1` へのメッセージを生成できませんが、その他のユーザーはメッセージを生成できます。

```
queue.tq1.produce.allow.user=*

queue.tq1.produce.deny.user=Bob
```

- ユーザーの * *principal* 規則は、グループの対応する * *principal* 規則をオーバーライドします。たとえば、次の 2 つの規則では、すべての認証済みユーザーがメッセージを `tq1` に送信できます。

```
queue.tq1.produce.allow.user=*

queue.tq1.produce.deny.group=*
```

- ユーザーに許可されたアクセスは、ユーザーのグループに許可されたアクセスをオーバーライドします。次の例では、Bob が User のメンバーである場合でも、`tq1` へのメッセージは生成できません。User のその他のメンバーはすべて、メッセージを生成できます。

```
queue.tq1.produce.allow.group=User

queue.tq1.produce.deny.user=Bob
```

- アクセスを介して明示的に指定されていないアクセス権は、暗黙的に拒否されます。たとえば、ACL ファイルにアクセス規則がない場合、すべてのユーザーはすべての操作を実行できません。

- 同じユーザーまたはグループにアクセス権の拒否と許可を行うと、すべてが取り消されます。たとえば、次の 2 つの規則が適用されると、Bob は q1 を検索できなくなります。

  ```
  queue.q1.browse.allow.user=Bob
  queue.q1.browse.deny.user=Bob
  ```

  次の 2 つの規則は、グループ User が q5 でメッセージを消費するのを禁止します。

  ```
  queue.q5.consume.allow.group=User
  queue.q5.consume.deny.group=User
  ```

- 同じ左側の規則が複数ある場合、最後のエントリが有効になります。

# 接続サービスのアクセス制御

ACL プロパティーファイルの接続アクセス制御のセクションには、ブローカの接続サービスのアクセス制御規則が含まれます。接続アクセス制御規則の構文は次のとおりです。

```
connection.resourceVariant.access.principalType=principals
```

*resourceVariant* には、NORMAL と ADMIN の 2 つの値が定義されています。これらの定義済みの値は、アクセス権を付与できる唯一のタイプの接続サービスです。

デフォルトの ACL プロパティーファイルは、すべてのユーザーに NORMAL 接続サービスへのアクセス権を付与し、グループ admin のユーザーに ADMIN 接続サービスへのアクセス権を付与します。

```
connection.NORMAL.allow.user=*
connection.ADMIN.allow.group=admin
```

ファイルベースのユーザーリポジトリを使用している場合、ユーザーマネージャーユーティリティーによりデフォルトのグループ admin が作成されます。LDAP ユーザーリポジトリを使用している場合、次のいずれかを実行して、デフォルトの ACL プロパティーファイルを使用します。

- LDAP ディレクトリでグループ admin を定義する。
- ACL プロパティーファイルの名前 admin を、LDAP ディレクトリで定義される 1 つ以上のグループの名前と置換する。

接続アクセス権限を制限できます。たとえば、次の規則では Bob が NORMAL にアクセスすることは拒否されますが、ほかのユーザーはすべてアクセスが許可されます。

```
connection.NORMAL.deny.user=Bob
connection.NORMAL.allow.user=*
```

アスタリスク (*) 文字を使用して、すべての認証済みユーザーまたはグループを指定できます。

ACL プロパティーファイルを使用して ADMIN 接続へのアクセスを付与する方法は、次のように、ファイルベースのユーザーリポジトリと LDAP ユーザーリポジトリでは異なります。

- **ファイルベースのユーザーリポジトリ**
  - アクセス制御が無効に設定されている場合、グループ admin には ADMIN 接続権限が割り当てられます。
  - アクセス制御が有効な場合、ACL ファイルを編集します。ユーザーまたはグループに、ADMIN 接続サービスへのアクセス権を明示的に付与します。
- **LDAP ユーザーリポジトリ。**LDAP ユーザーリポジトリを使用している場合、次のいずれかを実行します。
  - アクセス制御を有効にします。
  - ACL ファイルを編集して、ADMIN 接続を作成できるユーザーまたはグループの名前を指定します。LDAP ディレクトリサーバーで定義されるユーザーまたはグループを指定します。

# 物理的な送信先のアクセス制御

アクセス制御プロパティーファイルの送信先アクセス制御セクションには、物理的送信先ベースのアクセス制御規則が含まれます。これらの規則では、誰 ( ユーザーまたはグループ ) が何 ( 操作 ) をどこ ( 物理的送信先 ) に行うかが決定されます。これらの規則で統制されるアクセスのタイプには、キューへのメッセージの送信、トピックへのメッセージの発行、キューからのメッセージの受信、トピックへのサブスクライブ、キューでのメッセージの検索が含まれます。

デフォルトでは、あらゆるユーザーまたはグループが、任意の物理的送信先に対してあらゆるタイプのアクセス権を保持できます。さらに詳細な送信先アクセス規則を追加したり、デフォルトの規則を編集したりできます。この節の残りの部分では、自分自身の規則を記述するために理解しておく必要のある物理的送信先アクセス規則の構文について説明します。

送信先規則の構文は次のとおりです。

*resourceType . resourceVariant . operation . access . principalType=principals*

表 7-4 にこれらの要素の説明を示します。

表 7-4　送信先アクセス制御規則の要素

| コンポーネント | 説明 |
|---|---|
| *resourceType* | queue か topic のいずれかです。 |
| *resourceVariant* | 物理的送信先名、またはすべてのキューやすべてのトピックを表す、すべての物理的送信先 (*) です。 |
| *operation* | produce、consume、または browse のいずれかです。 |
| *access* | allow か deny のいずれかです。 |
| *principalType* | user か group のいずれかです。 |

アクセス権は、1 人以上のユーザーまたは 1 つ以上のグループ、あるいはその両方に対して指定できます。

次の例では、さまざまな種類の物理的送信先のアクセス制御規則を示します。

- すべてのユーザーが、あらゆるキュー送信先に対してメッセージを送信できる。

  ```
  queue.*.produce.allow.user=*
  ```

- user グループのメンバーがトピック Admissions にサブスクライブする機能を拒否する。

  ```
  topic.Admissions.consume.deny.group=user
  ```

# 自動作成された物理的送信先のアクセス制御

ACL プロパティーファイルの最後のセクションには、どのユーザーおよびグループに対してブローカが物理的送信先を自動作成するのかを指定するアクセス規則が含まれます。

ユーザーがまだ存在していない物理的送信先でプロデューサまたはコンシューマを作成すると、ブローカの自動作成プロパティーが有効になっている場合、ブローカは送信先を作成します。

デフォルトでは、任意のユーザーやグループは、ブローカに物理的送信先を自動作成させる権限を持っています。この権限は、次の規則で指定されます。

```
queue.create.allow.user=*

topic.create.allow.user=*
```

ACL ファイルを編集して、このタイプのアクセスを制限できます。

物理的送信先の自動作成アクセス規則の一般的な構文は、次のとおりです。

*resourceType*.create.*access*.*principalType*=*principals*

*resourceType* の部分には、queue か topic が表示されます。

たとえば次の規則により、ブローカは Snoopy 以外の全員に対してトピック送信先を自動作成できます。

```
topic.create.allow.user=*

topic.create.deny.user=Snoopy
```

物理的送信先の自動作成規則の結果は、物理的送信先のアクセス規則の影響と一致している必要があります。たとえば、1) 送信先アクセス規則を変更して、どのユーザーも送信先にメッセージを送信できないようにしてから、2) 送信先の自動作成を有効にすると、ブローカは物理的送信先が存在しない場合、物理的送信先を作成しますが、メッセージの配信は行いません。

# SSL ベースのサービスの操作

SSL (Secure Socket Layer) 標準に基づく接続サービスは、クライアントとブローカ間で暗号化されるメッセージを送信します。この節では、SSL ベースの接続サービスの設定方法を説明します。

Message Queue は、SSL (Secure Socket Layer) 規格に基づく次の接続サービスをサポートしています。

- ssljms、ssladmin、および cluster は、TCP/IP で使用される。
- httpsjms は HTTP で使用される。

これらの接続サービスにより、クライアントとブローカ間で送信されるメッセージが暗号化されます。Message Queue は、自己署名型サーバー証明書または自己署名型証明書に基づく SSL 暗号化をサポートしています。

SSL ベースの接続サービスを使用するには、キーツールユーティリティー (imqkeytool) を使用して、非公開キーと公開キーのペアを生成します。このユーティリティーは、ブローカへの接続を要求しているクライアントに渡される自己署名の証明書に公開キーを埋め込み、クライアントはこの証明書を使用して、接続を暗号化します。

Message Queue の SSL ベースの接続サービスはこれと同様の概念ですが、設定の方法に若干の違いがあります。

この節の後半では、TCP/IP で安全な接続を設定する方法について説明します。

HTTP を使用するユーザー向けの SSL ベースの接続サービス `httpsjms` では、クライアントとブローカが HTTPS トンネルサーブレットを使用して安全な接続を確立できます。HTTP を使用した安全な接続の確立の詳細は、369 ページの付録 C「HTTP/HTTPS のサポート」を参照してください。

# TCP/IP を介した安全な接続サービス

次の SSL ベースの接続サービスは、TCP/IP を介した直接的で安全な接続を提供します。

- `ssljms` サービスは、クライアントとブローカ間で安全な暗号化接続を介してメッセージを配信します。
- `ssladmin` サービスは、Message Queue コマンドユーティリティー (`imqcmd`) とブローカ間に安全な暗号化接続を作成します。安全な接続は、管理コンソール (`imqadmin`) に対してサポートされていません。
- `cluster` サービスは、メッセージを配信し、クラスタ内のブローカ間に安全な暗号化接続を介したブローカ間通信を確立します (188 ページの「ブローカ間の安全な接続」を参照 )。

# 自己署名型証明書の使用の設定

この節では、自己署名型証明書を使用した SSL ベースのサービスの設定方法を説明します。

認証を強力なものにするために、認証局が検証する署名付き証明書を使用できます。まずこの節の手順に従い、次に 158 ページの「署名付き証明書の使用の設定」に進んで、追加手順を実行します。

### ➤ SSL ベースの接続サービスを設定する

1. 自己署名型証明書を生成する。
2. ブローカで `ssljms`、`ssladmin`、`cluster` のいずれかの接続サービスを有効にする。
3. ブローカを起動する。
4. クライアントを設定し実行する (`ssljms` 接続サービスだけに適用 )。

`ssljms` および `ssladmin` 接続サービスを設定する手順は、手順 4 のクライアントの設定と実行以外は同じです。

各手順については、次で詳しく説明します。

## 手順 1: 自己署名型証明書の生成

Message Queue の自己署名型証明書による SSL Support は、クライアントが既知の信頼されたサーバーと通信することを前提に、ネットワーク上のデータを保護することを目的としています。

キーツールユーティリティーを実行し、ブローカの自己署名型証明書を生成します。UNIX システムでは、キーストアを作成するアクセス権を取得するためにスーパーユーザー (root) としてキーツールを実行する必要があります。

ssljms、ssladmin、cluster の接続サービスに対して、同じ証明書を使用できます。

コマンドプロンプトで次のとおり入力します。

```
imqkeytool -broker
```

ユーティリティーがキーストアのパスワードの入力を要求します。

```
Generating keystore for the broker ...
Enter keystore password:
```

次に、ユーティリティーは証明書の所有者である、ブローカを識別する情報の入力を要求します。指定する情報は、X.500 識別名になります。次の表にプロンプトの一覧とその説明、および各プロンプトの例を示します。値は大文字と小文字を区別し、空白を使用できます。

**表 7-5** 自己署名型証明書に必要な識別名情報

| プロンプト | 説明 | 例 |
|---|---|---|
| 氏名 | X.500 commonName (CN)。ブローカを実行しているサーバーの完全修飾名を入力します。 | myhost.sun.com |
| 組織名 | X.500 organizationUnit (OU)。部署または部門の名前を入力します。 | purchasing |
| 組織名 | X.500 organizationName (ON)。企業または政府機関などの大規模組織の名前です。 | My Company, Inc. |
| 市町村名 | X.500 localityName (L)。 | San Francisco |
| 州名または県名 | X.500 stateName (ST)。頭語ではなく、州または県の完全名を入力します。 | California |
| 組織の 2 文字国コード | X.500 country (C)。 | US |

情報を入力し終えたら、キーツールにより確認の画面が表示されます。たとえば、次のように表示されます。

```
Is CN=mqserver.sun.com, OU=purchasing, O=My Company, Inc., L=San
Francisco, ST=California, C=US correct?
```

値を再入力する場合は、デフォルトを使用するか no を入力します。現在の値でよければ、先に進み、yes を入力します。確認後、キーツールがキーの組み合わせを生成する間、このツールは停止します。

次に、キーツールから特定のキーの組み合わせをロックするためのパスワードの入力が要求されます ( キーパスワード )。このプロンプトに対して Return キーを押し、キーパスワードおよびキーストアパスワードと同じパスワードを使用します。

**注** 指定したパスワードは覚えておいてください。ブローカの起動時にこのパスワードを指定し、ブローカがキーストアを開けるようにする必要があります。キーストアパスワードはパスワードファイルに保存できます (161 ページの「パスワードファイルの使用」を参照 )。

imqkeytool コマンドにより、JDK keytool ユーティリティーが実行されて、自己署名型証明書が生成されます。生成された証明書は、付録 A 「プラットフォームごとの Message Queue データの場所」に記載されているとおり、オペレーティングシステムに応じたディレクトリにある Message Queue のキーストアに配置されます。

キーストアは、JDK1.2 keytool ユーティリティーでサポートされているのと同じ形式になっています。

次に示すのは Message Queue キーストアの設定可能なプロパティーです。

- imq.keystore.file.dirpath。SSL ベースのサービスの場合は、キーストアファイルが配置されているディレクトリへのパスを指定します。デフォルト値については、付録 A 「プラットフォームごとの Message Queue データの場所」を参照してください。
- imq.keystore.file.name。SSL ベースのサービスの場合は、キーストアファイル名を指定します。
- imq.keystore.password。SSL ベースのサービスの場合は、キーストアのパスワードを指定します。

たとえば特定の問題を解決するには、キーの組み合わせを生成し直す必要があります。

- キーストアのパスワードを忘れてしまった。
- ブローカの起動時に例外 java.security.UnrecoverableKeyException: Cannot recover key が発生し、SSL ベースのサービスの初期化に失敗した。

  この例外は、自己署名型証明書を 153 ページの「手順 1: 自己署名型証明書の生成」で生成するときに、キーストアのパスワードと違ったものをキーのパスワードに設定したことが原因で発生する場合があります。

### ➤ キーの組み合わせを生成し直す

1. 付録 A 「プラットフォームごとの Message Queue データの場所」に示すとおり、ブローカのキーストアを削除します。
2. `imqkeytool` をもう一度実行し、153 ページの「手順 1: 自己署名型証明書の生成」の説明に従って、キーの組み合わせを生成します。

## 手順 2: ブローカでの SSL ベースのサービスを有効にする

ブローカでの SSL ベースのサービスを有効にするには、`ssljms` ( または、`ssladmin`) を `imq.service.activelist` プロパティーに追加する必要があります。

---

**注** `imq.service.activelist` プロパティーではなく、`imq.cluster.transport` プロパティーを使用して、SSL ベースの `cluster` 接続サービスを有効にします。188 ページの「ブローカ間の安全な接続」を参照してください。

---

### ➤ ブローカで SSL ベースのサービスを有効にする

1. ブローカのインスタンス設定ファイルを開きます。

   インスタンス設定ファイルは、その設定ファイルが関連付けられているブローカインスタンスの名前 (*instanceName*) によって識別されたディレクトリに書き込まれます ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

   `.../instances/`*instanceName*`/props/config.properties`

2. 存在していない場合は、`imq.service.activelist` プロパティーのエントリを追加し、SSL ベースのサービスをリストに含めます。

   デフォルトでは、プロパティーには jms 接続サービスと admin 接続サービスが含まれます。アクティブ化するサービスに応じて、ssljms 接続サービスか ssladmin 接続サービス、またはその両方を追加する必要があります。

   ```
   imq.service.activelist=jms,admin,ssljms,ssladmin
   ```

## 手順 3: ブローカを起動する

キーストアパスワードを入力して、ブローカを起動します。パスワードの入力は、次のいずれかの方法で行います。

- ブローカの起動時にパスワードを要求するように許可します

  ```
  imqbrokerd
  Please enter Keystore password: myPassword
  ```

- 161 ページの「パスワードファイルの使用」の説明に従って、パスワードをパスワードファイルに格納します。パスワードをパスワードファイルに格納し、プロパティー `imq.passfile.enabled=true` を設定したあとで次のいずれかを実行します。
  - `imqbrokerd` コマンドにパスワードファイルの場所を渡します。

    `imqbrokerd -passfile /tmp/myPassfile`
  - `-passfile` オプションを使用せず、次の 2 つのブローカ設定プロパティーを使用するパスワードファイルの場所を指定して、ブローカを起動します。

    `imq.passfile.dirpath=/tmp`

    `imq.passfile.name=myPassfile`

SSL を使用してブローカまたはクライアントを起動するとき、多くの CPU サイクルが数秒間消費されます。これは、Message Queue が JSSE (Java Secure Socket Extension) を使用して SSL を実装するためです。JSSE は `java.security.SecureRandom` を使用して、ランダムな数を生成します。このメソッドで初期ランダム番号シードを作成するにはかなりの時間がかかり、そのために CPU の使用率が増加します。シードが作成されたあと、CPU レベルは通常に戻ります。

## 手順 4: SSL ベースのクライアントを設定および実行する

最後に安全な接続サービスを使用するように、クライアントを設定します。TCP/IP を使用した安全な接続には、2 つのシナリオが考えられます。

- `ssljms` を使用するアプリケーションクライアント
- `ssladmin` を使用する Message Queue 管理クライアント (`imqcmd` など )

後続の節では、これらについて個別に説明します。

### *ssljms を使用するアプリケーションクライアント*

クライアントが必要な JSSE (Java Secure Socket Extension) jar ファイルをクラスパスに保持していることを確認し、このファイルに `ssljms` 接続サービスを使用するよう指示する必要があります。

1. クライアントが JSSE と JNDI のサポートを組み込んだ J2SDK1.4 を使用していない場合、クライアントのクラスパスに次の `.jar` ファイルがあることを確認します。

   `jsse.jar、jnet.jar、jcert.jar、jndi.jar`
2. クライアントのクラスパスに次の Message Queue `.jar` ファイルがあることを確認します。

   `imq.jar、jms.jar`

3. クライアントを起動し、ブローカの ssljms サービスに接続します。これを行う 1 つの方法として、次のようなコマンドを入力します。

   ```
   java -DimqConnectionType=TLS clientAppName
   ```

   imqConnectionType を設定すると、接続に SSL を使用するよう指示が出されます。

   クライアントアプリケーションでの ssljms 接続サービスの使用についての詳細は、『Message Queue Developer's Guide for Java Clients』の管理対象オブジェクトの使用に関する章を参照してください。

### *ssladmin を使用する管理クライアント (imqcmd)*

imqcmd を使用するときに -secure オプションを含めると、安全な管理接続を確立できます。たとえば、次のように指定します。

imqcmd list svc -b *hostName*:*portNumber* -u *adminName* -secure

*adminName* には Message Queue ユーザーリポジトリの有効なエントリが入り、コマンドからパスワードの入力が要求されます。単層型ファイルリポジトリを使用している場合は、141 ページの「デフォルトの管理者パスワードの変更」を参照してください。

接続サービスを一覧表示すると、次の出力のように、ssladmin サービスが実行中で、安全な管理接続が問題なく確立されたことが示されます。

```
Listing all the services on the broker specified by:

Host                    Primary Port
localhost               7676

Service Name      Port Number        Service State
admin             33984 (dynamic)    RUNNING
httpjms           -                  UNKNOWN
httpsjms          -                  UNKNOWN
jms               33983 (dynamic)    RUNNING
ssladmin          35988 (dynamic)    RUNNING
ssljms            dynamic            UNKNOWN

Successfully listed services.
```

# 署名付き証明書の使用の設定

署名付き証明書は、自己署名型証明書よりも強力なサーバー認証をもたらします。署名付き証明書を実装するには、キーストアに署名付き証明書をインストールし、Message Queue クライアントが `imqbrokerd` との SSL 接続を確立するときに、署名付き証明書を要求するように設定します。

署名付き証明書はクライアントとブローカ間でのみ実装可能で、クラスタ内の複数のブローカ間に実装できません。

次の手順では、152 ページの「自己署名型証明書の使用の設定」に文書化した手順を実行しているものと仮定しています。手順に従う際に、J2SE キーツール証明書と X.509 証明書に関する情報を `http://java.sun.com` で確認しておくと役に立つ場合があります。

## 手順 1: 署名付き証明書の取得とインストール

### ➤ 署名付き証明書を取得する

1. J2SE キーツールを使用して、前節で作成した自己署名型証明書の CSR (Certificate Signing Request) を生成します。

   次に例を示します。

   ```
   keytool -certreq -keyalg RSA -alias imq -file certreq.csr
           -keystore /etc/imq/keystore -storepass myStorePassword
   ```

   CSR は次にファイル `certreq.csr` に証明書をカプセル化します。

2. 次のいずれかの方法で、署名付き証明書を作成するか要求します。

   - Thawte や Verisign などの非常に著名な認証局 (CA) から、証明書に署名してもらいます。このプロセスの詳細は、CA のマニュアルを参照してください。
   - SSL 署名ソフトウェアパッケージを使用して、証明書に自己署名を行います。

   最終的な署名付き証明書は、ASCII 文字列が連続したものになります。CA から署名付き証明書を受け取る場合、電子メールの添付またはテキスト形式のメッセージとして到着します。

3. 署名付き証明書を取得した場合、ファイルに保存します。

   次の手順では、ブローカの証明書に `broker.cer` という名前が使用されています。

### ➤ 署名付き証明書をインストールする

1. `$JAVA_HOME/lib/security/cacerts` で、次のコマンドを使用して、J2SE がデフォルトで使用中の CA をサポートしているかどうかを確認します。

   ```
   keytool -v -list -keystore $JAVA_HOME/lib/security/cacerts
   ```

   このコマンドは、システムキーストアの root CA を一覧表示します。

使用中の CA がリスト内に見つかったら、次の手順は省略してください。

2. 使用中の CA が J2SE でサポートされていない場合、認証局の root 証明書を imqbrokerd キーストアにインポートします。

   次に例を示します。

   ```
   keytool -import -alias ca -file ca.cer -noprompt -trustcacerts
           -keystore /etc/imq/keystore -storepass myStorePassword
   ```

   値 ca.cer は、CA から入手した CA root 証明書です。

   CA テスト証明書を使用している場合、Test CA Root 証明書をインポートする必要があるかもしれません。CA には、Test CA Root のコピーの入手方法に関する手順が示されているはずです。

3. 署名付き証明書をキーストアにインポートし、オリジナルの自己署名型証明書と置き換えます。

   たとえば、次のように指定します。

   ```
   keytool -import -alias imq -file broker.cer -noprompt -trustcacerts
           -keystore /etc/imq/keystore -storepass myStorePassword
   ```

   値 broker.cer は、CA から受け取った署名付き証明書を含むファイルです。

imqbrokerd キーストアに、SSL 接続に使用できる署名付き証明書が格納されました。

## 手順 2: 署名付き証明書を要求するクライアントランタイムの設定

### ➤ Java クライアントランタイムを設定する

デフォルトでは、Message Queue クライアントランタイムは imqbrokerd を信頼し、提示されるすべての証明書を受け付けます。次に署名付き証明書を要求するクライアントランタイムを設定し、クライアントが証明書に署名した CA を確実に信頼する必要があります。

1. imqbrokerd から有効な署名付き証明書を要求するようにクライアントを設定するには、クライアントの ConnectionFactory オブジェクトに対して imqSSLIsHostTrusted 属性を false に設定します。
2. 156 ページの「手順 4: SSL ベースのクライアントを設定および実行する」の説明に従って、imqbrokrd との SSL 接続の確立を試みます。

   broker の証明書に著名な CA が署名している場合、接続は成功すると見られ、次の手順は省略してもかまいません。接続が証明書の有効性検証エラーにより失敗した場合、次の手順を実行します。

3. 次の節の説明に従って、クライアントのトラストストアに署名 CA の root 証明書をインストールします。

トラストストアへのクライアントの設定方法は 3 通りあります。

- root CA をデフォルトシステム cacerts ファイルにインストールします。
- root CA を代替システム jssecacerts ファイルにインストールします。これは推奨オプションです。
- root CA を任意のキーストアファイルにインストールし、これをトラストストアとして使用するようにクライアントを設定します。

次の節では、上記のオプションを使用した Verisign Test Root CA のインストール方法の例を説明します。root CA はファイル testrootca.cer 内にあります。この例では、J2SE が /usr/j2se にインストールされていると仮定しています。

### *デフォルトシステム cacerts ファイルへのインストール*

この例では root CA をファイル $JAVA_HOME/usr/jre/lib/security/cacerts にインストールしています。

```
keytool -import -keystore /usr/j2se/jre/lib/security/cacerts
         -alias VerisignTestCA -file testrootca.cer -noprompt
         -trustcacerts -storepass myStorePassword
```

クライアントはデフォルトでこのキーストアを検索するため、その他のクライアント設定は不要です。

### *jssecacerts へのインストール*

この例では root CA をファイル $JAVA_HOME/usr/jre/lib/security/jssecacerts にインストールしています。

```
keytool -import -keystore /usr/j2se/jre/lib/security/jssecacerts
        -alias VerisignTestCA -file testrootca.cer -noprompt
        -trustcacerts -storepass myStorePassword
```

クライアントはデフォルトでこのキーストアを検索するため、その他のクライアント設定は不要です。

### *その他のファイルへのインストール*

この例では、root CA をファイル /home/smith/.keystore にインストールしています。

```
keytool -import -keystore /home/smith/.keystore
        -alias VerisignTestCA -file testrootca.cer -noprompt
        -trustcacerts -storepass myStorePassword
```

クライアントはデフォルトでこのキーストアを検索しないため、トラストストアの場所をクライアントに知らせる必要があります。この場合、クライアントの実行後に Java システムプロパティー javax.net.ssl.trustStore を設定します。たとえば、次のように指定します。

```
javax.net.ssl.trustStore=/home/smith/.keystore
```

# パスワードファイルの使用

コマンドにはパスワードを必要とするものがいくつかあります。表 7-6 では、最初の列にパスワードを必要とするコマンド、2 番目の列にパスワードが必要な理由を一覧表示しています。

表 7-6　パスワードを使用するコマンド

| コマンド | 目的 | パスワードの目的 |
|---|---|---|
| imqbrokerd | ブローカを起動する | JDBC ベースの持続データストア、SSL 証明書キーストア、または LDAP ユーザーリポジトリにアクセスします |
| imqcmd | ブローカを管理する | コマンドの使用が許可された管理ユーザーを認証します |
| imqdbmgr | JDBC ベースのデータストアを管理する | データストアにアクセスします |

パスワードファイルでこれらのパスワードを指定し、-passfile オプションを使用してファイル名を指定します。次に示すのは、-passfile オプションの形式です。

```
imqbrokerd -passfile myPassfile
```

**注**　以前のリリースでは、-p、-password、-dbpassword、-ldappassword のオプションを使用してコマンド行でパスワードを指定できました。これらのオプションには異論が多く、今後のリリースでは削除される予定です。現在のリリースでは、これらのオプションのいずれかのコマンド行の値は、パスワードファイルの対応する値よりも優先されます。

# セキュリティー上の問題

プロンプトに応じて、パスワードをインタラクティブに指定するのは、ほかのユーザーにモニタリングが表示されていなければ、もっとも安全なパスワード指定の方法です。またコマンド行でパスワードファイルも指定できます。ただし、コマンドをインタラクティブではない方法で使用する場合、パスワードファイルを使用する必要があります。

パスワードファイルは暗号化されず、このため不正なアクセスから保護するためにパスファイルにアクセス権を設定する必要があります。ファイルを表示できるユーザーを制限するが、ブローカを起動するユーザーに読み取りアクセスを許可するようにアクセス権を設定します。

# パスワードファイルの内容

パスワードファイルは、一連のプロパティーと値を収めた簡単なテキストファイルです。それぞれの値はコマンドで使用されるパスワードです。

パスワードファイルには、表 7-7 に示すパスワードを含めることができます。

**表 7-7** パスワードファイル内のパスワード

| パスワード | 影響を受けるコマンド | 説明 |
|---|---|---|
| `imq.imqcmd.password` | `imqcmd` | `imqcmd` コマンド行の管理者パスワードを指定します。パスワードはコマンドごとに認証されます。 |
| `imq.keystore.password` | `imqbrokerd` | SSL ベースのサービスにキーストアパスワードを指定します。 |
| `imq.persist.jdbc.password` | `imqbrokerd`<br>`imdbmgr` | 必要に応じて、データベース接続を開くときに使用するパスワードを指定します。 |
| `imq.user_repository.ldap.password` | `imqbrokerd` | 設定された LDAP ユーザーリポジトリにバインドするためにブローカに割り当てられた、識別名に関連するパスワードを指定します。 |

サンプルパスワードファイルは、Message Queue 製品に組み込まれています。サンプルファイルの場所については、付録 A 「プラットフォームごとの Message Queue データの場所」を参照してください。

# 監査ログの作成

Message Queue は Enterprise Edition でのみ監査ロギングをサポートします。監査ロギングを有効にすると、Message Queue は次のタイプのイベントにレコードを生成します。

- ブローカインスタンスの起動、シャットダウン、再起動、削除
- ユーザーの認証と承認
- 持続ストアのリセット
- 物理的送信先の作成、消去、破棄
- 永続的サブスクライバの管理上の破棄

Message Queue ブローカログファイルにレコードの監査ロギングを作成するには、`imq.audit.enabled` ブローカプロパティーを `true` に設定します。ログ内のすべての監査レコードには、キーワード `AUDIT` が含まれています。

第 8 章

# 管理対象オブジェクトの管理

管理対象オブジェクトでは、プロバイダ固有の設定およびネーミング情報をカプセル化することにより、JMS プロバイダ間で移植可能なクライアントアプリケーションを開発できます。Message Queue の管理者は、通常、クライアントアプリケーションがメッセージの送受信のためのブローカ接続を取得する際に使用する管理対象オブジェクトを作成します。

この章では、オブジェクトマネージャーユーティリティー (`imqobjmgr`) を使用して、管理対象オブジェクトを作成および管理する方法について説明します。この章は、次の節から構成されています。



## オブジェクトストア

管理対象オブジェクトは、即時に使用可能なオブジェクトストアに配置されます。クライアントアプリケーションは、JNDI (Java Naming and Directory Interface) を介して、このオブジェクトストアに配置された管理対象オブジェクトにアクセスします。使用できるオブジェクトストアには、標準 LDAP (Lightweight Directory Access Protocol) ディレクトリサーバーとローカルファイルシステムのディレクトリの 2 種類があります。

# LDAP サーバーオブジェクトストア

LDAP サーバーは、運用メッセージングシステム用のオブジェクトストアとしてお勧めします。LDAP サーバーは、分散システムでの使用を考慮した設計になっており、本稼働環境で役立つセキュリティー機能を備えています。

LDAP 実装は、多数のベンダーによってサポートされています。Message Queue の管理ツールで LDAP サーバー上のオブジェクトストアを管理するには、最初に、Java オブジェクトを格納して JNDI 検索を実行するようにサーバーを設定する必要がある場合があります。詳細については、使用する LDAP 実装に付属のマニュアルを参照してください。

LDAP サーバーをオブジェクトストアとして使用するには、表 8-1 に示す属性を指定する必要があります。これらの属性は、次のように分類されます。

- **初期コンテキスト**：`java.naming.factory.initial` 属性は、サーバーでの JNDI 検索の初期コンテキストを指定します。LDAP オブジェクトストアの場合、この属性の値は固定です。
- **ロケーション**：`java.naming.provider.url` 属性は、LDAP サーバーの URL とディレクトリパスを指定します。指定したパスが存在することを確認する必要があります。
- **セキュリティー**：`java.naming.security.principal`、`java.naming.security.credentials`、および `java.naming.security.authentication` 属性は、オブジェクトストアにアクセスを試みる呼び出し元の認証を制御します。これらの属性の正確な形式と値は、LDAP サービスプロバイダによって異なります。詳細について、およびセキュリティー情報がすべての操作に対して必要か、または格納されたデータを変更する操作に対してのみ必要かについては、使用する LDAP 実装に付属のマニュアルを参照してください。

表 8-1　LDAP オブジェクトストアの属性

| 属性 | 説明 |
|---|---|
| `java.naming.factory.initial` | JNDI 検索の初期コンテキスト<br>例：<br>`com.sun.jndi.ldap.LdapCtxFactory` |

表 8-1　LDAP オブジェクトストアの属性 ( 続き )

| 属性 | 説明 |
|---|---|
| java.naming.provider.url | サーバーの URL とディレクトリパス<br>例 :<br>ldap://mydomain.com:389/ou=mqobjs,o=myapp<br>この場合、管理対象オブジェクトは、/myapp/mqobjs ディレクトリに格納されます。 |
| java.naming.security.principal | 呼び出し元を認証するための主体の識別情報<br>この属性の形式は、認証スキーマによって異なります。たとえば、次のように指定します。<br>uid=homerSimpson,ou=People,o=mq<br>この属性を指定しない場合は、LDAP サービスプロバイダによって動作が決定されます。 |
| java.naming.security.credentials | 認証主体の資格情報<br>この属性の値は、認証スキーマによって異なります。たとえば、ハッシュ化されたパスワード、クリアテキストのパスワード、キー、証明書などになります。<br>このプロパティーを指定しない場合は、LDAP サービスプロバイダによって動作が決定されます。 |
| java.naming.security.authentication | 認証のセキュリティーレベル<br>この属性の値は、キーワード none、simple、または strong のいずれかになります。たとえば、simple を指定した場合は、未指定の主体または資格情報の値を入力するよう要求されます。これによって、識別情報をより安全に提供することが可能となります。<br>このプロパティーを指定しない場合は、LDAP サービスプロバイダによって動作が決定されます。 |

# ファイルシステムオブジェクトストア

Message Queue では、管理対象オブジェクトのオブジェクトストアとしてローカルファイルシステムのディレクトリを使用することもサポートされています。この方法は、本稼動システムにはお勧めしませんが、開発環境で非常に簡単に扱えるという利点があります。ただし、複数のコンピュータノードに配備されているクライアントに対して、一元化されたオブジェクトストアとしてディレクトリを使用する場合は、それらすべてのクライアントがディレクトリへのアクセス権を持っている必要があります。また、ディレクトリへのアクセス権を持つユーザーはすべて、Message Queue の管理ツールを使用して管理対象オブジェクトを作成および管理できます。

ファイルシステムのディレクトリをオブジェクトストアとして使用するには、表 8-2 に示す属性を指定する必要があります。これらの属性の意味は、前述の LDAP オブジェクトストアの場合とほとんど同じですが、`java.naming.provider.url` 属性では、オブジェクトストアを格納するディレクトリのディレクトリパスを指定します。このディレクトリが存在していて、Message Queue の管理ツールのユーザーと、ストアにアクセスするクライアントアプリケーションのユーザーが、適切なアクセス権を持っている必要があります。

表 8-2　ファイルシステムオブジェクトストアの属性

| 属性 | 説明 |
|---|---|
| `java.naming.factory.initial` | JNDI 検索の初期コンテキスト<br>例 :<br>`com.sun.jndi.fscontext.RefFSContextFactory` |
| `java.naming.provider.url` | ディレクトリパス<br>例 :<br>`file:///C:/myapp/mqobjs` |

# 管理対象オブジェクトの属性

Message Queue の管理対象オブジェクトには 2 つの基本的な種類があります。

- 接続ファクトリ。ブローカへの接続を作成するために、クライアントアプリケーションが使用します。
- 送信先。クライアントアプリケーションがメッセージを交換 ( 送受信 ) するブローカ上の場所を表す。

これらの種類の管理対象オブジェクトはそれぞれ、オブジェクトのプロパティーと動作を決める特定の属性を持ちます。この節では、コマンド行ユーティリティーであるオブジェクトマネージャー (`imqobjmgr`) を使用してこれらの属性を設定する方法について説明します。また、第 2 章で説明したように、GUI 管理コンソールで属性を設定することもできます (56 ページの「管理対象オブジェクトの操作」を参照 )。

## 接続ファクトリの属性

クライアントアプリケーションは、接続ファクトリ管理対象オブジェクトを使用して、ブローカとメッセージを交換するための接続を作成します。接続ファクトリの属性は、その接続ファクトリが作成するすべての接続のプロパティーを定義します。接続が作成されたあとは、そのプロパティーを変更できません。つまり、接続のプロパティーを設定するには、その作成に使用される接続ファクトリの属性を設定する必要があります。

Message Queue では、2 つのクラスの接続ファクトリオブジェクトが定義されています。

- `ConnectionFactory` オブジェクト。通常のメッセージングおよび分散していないトランザクションをサポート。
- `XAConnectionFactory` オブジェクト。分散トランザクションをサポート。

どちらのクラスも同じ設定属性を持ち、それらを使用して、リソース、パフォーマンス、およびメッセージスループットを最適化できます。これらの属性のリストと詳細な説明については、第 16 章「管理対象オブジェクト属性のリファレンス」を参照してください。次の節では、これらの属性について取り上げます。


- 170 ページの「接続の処理」
- 172 ページの「クライアントの識別」
- 174 ページの「信頼性およびフロー制御」
- 175 ページの「キューブラウザとサーバーセッション」
- 175 ページの「標準メッセージプロパティー」
- 175 ページの「メッセージヘッダーのオーバーライド」

## 接続の処理

接続処理の属性では、接続先のメッセージサーバーのアドレス、および必要に応じて、接続障害の検出方法と再接続の試行方法を指定します。これらの要約については、330 ページの表 16-1 を参照してください。

### *メッセージサーバーのアドレスリスト*

もっとも重要な接続処理の属性は、接続の確立先となる 1 つ以上のブローカを指定する `imqAddressList` です。この属性の値は、メッセージサーバーのアドレスを 1 つ含む文字列、またはブローカクラスタの場合はコンマ区切りで複数含む文字列です。サーバーのアドレスには、使用する接続サービス (78 ページの「接続サービス」を参照 ) と接続の確立方法に応じて、さまざまなアドレススキーマを使用できます。

- `mq`。`jms` または `ssljms` 接続サービスで、ブローカのポートマッパーを使用してポートを動的に割り当てます。
- `mqtcp`。`jms` 接続サービスを使用し、ポートマッパーをバイパスして、指定されたポートに直接接続します。
- `mqssl`。`ssljms` 接続サービスを使用し、指定されたポートへの SSL (Secure Socket Layer) 接続を作成します。
- `http`。`httpjms` 接続サービスを使用し、指定された URL の Message Queue トンネルサーブレットへの HTTP (Hypertext Transport Protocol) 接続を作成します。
- `https`。`httpsjms` 接続サービスを使用し、指定された URL の Message Queue トンネルサーブレットへの HTTPS (Secure Hypertext Transport Protocol) 接続を作成します。

これらのアドレススキーマの要約については、332 ページの表 16-2 を参照してください。

各メッセージサーバーアドレスの一般的な形式は、次のようになります。

*scheme*`://`*address*

*scheme* は、前述のアドレススキーマのいずれかで、*address* は、サーバーのアドレス自体を示します。アドレスを指定するための正確な構文は、表 16-2 の最後の列に示すように、アドレススキーマによって異なります。さまざまなアドレス形式の例については、333 ページの表 16-3 を参照してください。

複数のブローカによるクラスタ環境では、アドレスリストに複数のサーバーアドレスを含めることができます。最初の接続試行に失敗すると、Message Queue クライアントランタイムは、リスト内の別のアドレスに接続を試行し、成功するまでリスト内のすべてのアドレスに順に接続を試みます。2 つの追加の接続ファクトリ属性によって、この方法を制御します。

- imqAddressListBehavior。指定したアドレスへの試行順序を指定します。この属性を文字列 PRIORITY に設定すると、アドレスリストに表示されている順序でアドレスが試行されます。属性値を RANDOM にすると、ランダムな順序でアドレスが試行されます。これは、たとえば、多数の Message Queue クライアントが同じ接続ファクトリオブジェクトを共有しているときに、すべてのクライアントが同じサーバーアドレスに接続を試行するのを避けるためなどに便利です。
- imqAddressListIterations。試行を中止して失敗を報告するまでの、リストの繰り返し回数を指定します。値 -1 は、無限に繰り返すことを示します。つまり、クライアントランタイムは、接続の確立に成功するか時間切れになるまで試行を続けます。

### *自動再接続*

接続ファクトリの imqReconnectEnabled 属性を true に設定することで、接続に障害が発生した場合にクライアントがブローカに自動的に再接続するように設定できます。imqReconnectAttempts 属性では、特定のサーバーアドレスに再接続を試行する回数を制御します。imqReconnectInterval 属性では、試行の間隔をミリ秒単位で指定します。

メッセージサーバーのアドレスリスト (imqAddressList) で複数のアドレスを指定するブローカクラスタでは、障害の発生した接続を、元のブローカだけでなく、クラスタ内の別のブローカでも復元できます。元のブローカへの再接続に失敗すると、クライアントランタイムは、リスト内のほかのアドレスを試行します。前述のとおり、imqAddressListBehavior および imqAddressListIterations 属性で、アドレスの試行順序とリストの繰り返し回数を制御します。各アドレスは、imqReconnectInterval ミリ秒の間隔で、imqReconnectAttempts を最大回数として、繰り返し試行されます。

自動再接続では、メッセージの消費に関するすべてのクライアント通知モードがサポートされます。接続が再確立されると、ブローカは、以前に配信した未通知メッセージをすべて再配信し、それらに Redeliver フラグを付けます。アプリケーションコードでは、このフラグを使用して、特定のメッセージが消費されたが未通知であるかどうかを判断できます。ただし、永続的でないサブスクライバの場合、メッセージサーバーは、接続が閉じられたあとはメッセージを保持しません。そのため、接続がダウンしている間にそれらのサブスクライバに対して生成されたメッセージは、再接続後に配信できず、失われることになります。自動再接続中は、メッセージ生成がブロックされます。メッセージプロデューサは、接続の再確立が完了するまで、サーバーにメッセージを送信できません。

自動再接続は、接続のフェイルオーバーを提供しますが、データのフェイルオーバーは実行しません。障害の発生したブローカまたは切断されたブローカが保持する持続メッセージ、その他の状態情報は、クライアントが別のブローカインスタンスに再接続すると失われることがあります。接続の再確立の試行中、Message Queue によって、クライアントランタイムが提供したオブジェクト ( セッション、メッセージコンシューマ、メッセージプロデューサなど ) は維持されます。一時的送信先も、接続に障害が発生したときは、クライアントがそれらの送信先に再接続して再度アクセスする可能性があるため、当面は維持されます。再接続してそれらの送信先を使用するための時間をクライアントに与えたあとで、ブローカはそれらを削除します。再接続時

にブローカでクライアント側の状態を完全に復元できない場合 ( たとえば、接続の間のみ存在する処理済セッションを使用している場合など ) は、自動再接続は行われず、代わりに接続の例外ハンドラが呼び出されます。その後の例外のキャッチ、再接続、および状態の復元は、アプリケーションコードに任されます。

### *接続の定期的なテスト (ping)*

Message Queue クライアントランタイムは、接続を定期的にテストする、つまり「ping」を実行するように設定できます。これにより、メッセージ転送に失敗する前に、予防的に接続の障害を検出できます。メッセージを消費するだけで生成しないクライアントアプリケーションでは、接続の障害を検出する手段がほかにないため、このようなテストが特に重要です。メッセージをたまに生成するだけのクライアントでも、この機能が役立ちます。

接続ファクトリ属性 `imqPingInterval` では、接続で ping を実行する頻度を秒単位で指定します。デフォルトでは、この間隔は 30 秒に設定されます。値 `-1` を指定すると、ping の実行が無効になります。

失敗した ping への対応は、オペレーションシステムのプラットフォームによって異なります。一部のオペレーティングシステムでは、ただちに、クライアントアプリケーションの例外リスナーに例外がスローされます。クライアントに例外リスナーがない場合は、その接続を使用するための次回の試行に失敗します。また、オペレーティングシステムによっては、ブローカへの接続の確立を試行し続け、ping が成功するかバッファーがオーバーフローするまで、一連の ping をバッファリングするものもあります。

## クライアントの識別

334 ページの表 16-4 に示されている接続ファクトリ属性は、クライアント認証と、永続サブスクライバのクライアント識別子の設定をサポートしています。

### *クライアント認証*

ブローカへのすべての接続試行は、ユーザー名とパスワードによって、メッセージサーバーが管理するユーザーリポジトリに対して認証される必要があります。接続ファクトリ属性 `imqDefaultUsername` および `imqDefaultPassword` では、クライアントが接続の作成時にユーザー名とパスワードを明示的に提供しなかった場合に使用するデフォルトのユーザー名とパスワードを指定します。

開発者がアプリケーションの開発およびテストの際にユーザーリポジトリを設定する手間を省くことができるように、Message Queue には、ユーザー名とパスワードがどちらも `guest` に設定されたゲストユーザーアカウントが用意されています。これは、`imqDefaultUsername` および `imqDefaultPassword` 属性のデフォルト値でもあるため、これらが明示的に指定されていない場合、クライアントは常にゲストアカウントで接続を取得できます。本稼働環境では、ブローカ接続へのアクセスは、ユーザーリポジトリに明示的に登録されているユーザーのみに制限するべきです。

### *クライアント識別子*

『Java Message Service 仕様書』では、メッセージサーバーがクライアントに代わって持続状態を維持する必要があるときは常に、接続で一意のクライアント識別子を提供することが要求されています。Message Queue は、このクライアント識別子を使用して、トピック送信先への永続サブスクライバを追跡します。永続サブスクライバが非アクティブになると、ブローカは、そのトピックのすべての受信メッセージを保持し、サブスクライバが再度アクティブになったときにそれらを配信します。ブローカは、クライアント識別子によってサブスクライバを識別します。

クライアントアプリケーションが接続オブジェクトの `setClientID` メソッドを使用してプログラムによって独自のクライアント識別子を設定することは可能ですが、この場合、クライアント識別子が互いに一意になるように調整するのが難しくなります。一般的には、Message Queue が、クライアントに代わって接続を作成するときに一意の識別子を自動的に割り当てるようにすることをお勧めします。このためには、接続ファクトリの `imqConfiguredClientID` 属性を、次の形式の値に設定します。

`${u}`*factoryID*

この属性値の先頭の 4 文字は必ず、`${u}` になります。括弧内に `u` 以外の文字があると、接続の作成時に例外がスローされます。先頭以外の位置では、これらの文字は特別な意味を持たず、プレーンテキストとして扱われます。*factoryID* の値は、この接続ファクトリオブジェクトに一意に関連付ける文字列です。

特定のクライアントの接続を作成するときに、Message Queue は、文字列 `${u}` を `u:`*userName* に置き換えることによってクライアント識別子を生成します。*userName* は、接続で認証されたユーザー名です。これにより、特定の接続ファクトリによって作成された接続がそれぞれ、ほかのすべての面で同一であっても、一意のクライアント識別子を持つことが保証されます。たとえば、ユーザー名が `Calvin` で、接続ファクトリの `imqConfiguredClientID` 属性に指定された文字列が `${u}Hobbes` である場合、割り当てられるクライアント識別子は `u:CalvinHobbes` になります。

---

**注**　2 つのクライアントがデフォルトユーザー名 `guest` を使用して接続を取得しようとした場合、それぞれが同じ `${u}` 要素を含むクライアント識別子を持つことになるため、このスキーマは動作しません。この場合は、先に接続を要求したクライアントだけが接続を取得します。Message Queue は同じクライアント識別子を持つ 2 つの接続を作成できないため、あとのクライアントの接続試行は失敗します。

---

imqConfiguredClientID を使用してクライアント識別子を指定する場合でも、クライアントアプリケーションは、接続メソッド setClientID を使用してこの設定をオーバーライドできます。接続ファクトリの imqDisableSetClientID 属性を true に設定することで、これを避けることができます。永続サブスクライバを使用するアプリケーションでは、管理のために imqConfiguredClientID を使用するか、プログラムによって setClientID を使用して、クライアント識別子を必ず設定する必要があります。

## 信頼性およびフロー制御

クライアントが送受信する「ペイロード」メッセージと Message Queue 自体が使用する制御メッセージ ( ブローカ通知など ) は、クライアントとブローカ間の同じ接続でやり取りされるため、ペイロードのトラフィックが過剰になると、制御メッセージの配信が妨げられる可能性があります。この問題を軽減するために、335 ページの表 16-5 に示されている接続ファクトリ属性を使用して、2 種類のメッセージの相対的なフローを管理できます。これらの属性は、4 つのカテゴリに分類されます。

- **通知タイムアウト** : 例外をスローするまでのブローカ通知の最大待ち時間 (imqAckTimeout) を指定します。
- **接続フロー測定** : ペイロードメッセージの転送を、指定したサイズ (imqConnectionFlowCount) のバッチに制限します。これにより、累積された制御メッセージを配信する機会が定期的に得られます。
- **接続フロー制御** : 特定の接続で、消費を待って保留状態になるペイロードメッセージの数 (imqConnectionFlowLimit) を制限します。制限に達すると、消費待ちのメッセージの数が制限を下回るまで、その接続へのペイロードメッセージの配信が一時停止されます。この機能の使用は、ブール型のフラグ (imqConnectionFlowLimitEnabled) によって制御します。
- **コンシューマフロー制御** : 単一のコンシューマに対して、消費を待って保留状態になるペイロードメッセージの数 (imqConsumerFlowLimit) を制限します。この制限は、特定のキュー送信先のプロパティー (consumerFlowLimit) として指定することもできます。制限に達すると、消費待ちのメッセージの数が、imqConsumerFlowLimit に対する割合として imqConsumerFlowThreshold 属性で指定した制限を下回るまで、そのコンシューマへのペイロードメッセージの配信が一時停止されます。同じ接続上で 1 つのコンシューマがほかのコンシューマの通信を妨げるのを回避することによって、複数のコンシューマ間のロードバランスを向上させるのに役立ちます。

これらのフロー制御技術のいずれを使用する場合でも、信頼性とスループットの兼ね合いを考慮する必要があります。詳細は、236 ページの「クライアントランタイムのメッセージフローの調整」を参照してください。

## キューブラウザとサーバーセッション

336 ページの表 16-6 に、クライアントのキュー検索とサーバーセッションに関する接続ファクトリ属性が示されています。imqQueueBrowserMaxMessagesPerRetrieve 属性では、キュー送信先の内容の検索時に一度に取得するメッセージの最大数を指定します。imqQueueBrowserRetrieveTimeout 属性では、メッセージ取得の最大待ち時間を指定します。ブール型の imqLoadMaxToServerSession 属性では、アプリケーションサーバーセッションでの接続コンシューマの動作を制御します。この属性の値が true の場合、クライアントは、サーバーセッションに対して最大数までのメッセージを読み込みます。false の場合は、一度に 1 つのメッセージだけを読み込みます。

## 標準メッセージプロパティー

『Java Message Service 仕様書』では、Message Queue などの JMS プロバイダが任意でサポートできる、いくつかの標準メッセージプロパティーを定義しています。規定によって、これらの標準プロパティーの名前はすべて、JMSX という文字列で始まります。337 ページの表 16-7 に示されている接続ファクトリ属性では、Message Queue クライアントランタイムがこれらいずれかの標準プロパティーを設定するかどうかを制御します。生成されたメッセージについては、次のプロパティーがあります。

| | |
|---|---|
| JMSXUserID | メッセージを送信するユーザーの識別情報 |
| JMSXAppID | メッセージを送信するアプリケーションの識別情報 |
| JMSXProducerTXID | メッセージが生成されたトランザクションのトランザクション識別子 |

消費されたメッセージについては、次のものがあります。

| | |
|---|---|
| JMSXConsumerTXID | メッセージが消費されたトランザクションのトランザクション識別子 |
| JMSXRcvTimestamp | コンシューマにメッセージが配信された時刻 |

## メッセージヘッダーのオーバーライド

338 ページの表 16-8 に示されている接続ファクトリ属性を使用して、クライアントが特定の JMS メッセージヘッダーフィールドに設定した値をオーバーライドできます。指定した設定は、その接続ファクトリから取得された接続が生成するすべてのメッセージに使用されます。この方法でオーバーライドできるヘッダーフィールドには次のものがあります。

- JMSDeliveryMode　配信モード ( 持続的または持続性なし )
- JMSExpiration　有効期限
- JMSPriority　優先度のレベル

これらのフィールドにはそれぞれ、フィールドがオーバーライド可能であるかどうかを制御するブールの属性と、フィールドの値を指定する属性の 2 つがあります。たとえば、優先度のレベルを設定するための属性は、imqOverrideJMSPriority と imqJMSPriority です。さらに、オーバーライドの値を一時的送信先に適用するかどうかを制御する imqOverrideJMSHeadersToTemporaryDestinations 属性もあります。

**注** メッセージヘッダーをオーバーライドすると、特定のアプリケーションの要件に支障を及ぼす可能性があるため、これらの属性は、必ずアプリケーションの設計者またはユーザーに相談した上で使用することをお勧めします。

## 送信先の属性

物理的なキューまたはトピック送信先を識別する送信先管理対象オブジェクトには、339 ページの表 16-9 に示されている 2 つの属性だけがあります。重要な属性である imqDestinationName では、この管理対象オブジェクトが表す物理的送信先の名前を指定します。これは、その物理的送信先を作成した imqcmd create dst コマンドの -n オプションで指定された名前です。オプションの説明文字列である imqDestinationDescription もあります。これを使用して、送信先オブジェクトを識別しやすくしたり、作成済みのほかの送信先オブジェクトと区別しやすくしたりできます。

# オブジェクトマネージャーユーティリティーの使用

Message Queue のオブジェクトマネージャーユーティリティー (imqobjmgr) を使用して、管理対象オブジェクトを作成および管理できます。imqobjmgr コマンドには、管理対象オブジェクトに対してさまざまな操作を実行するための、次のサブコマンドが用意されています。

| | |
|---|---|
| add | 管理対象オブジェクトをオブジェクトストアに追加します |
| delete | 管理対象オブジェクトをオブジェクトストアから削除します |
| list | オブジェクトストア内の既存の管理対象オブジェクトを一覧表示します |
| query | 管理対象オブジェクトに関する情報を表示します |
| update | 管理対象オブジェクトの属性を変更します |

imqobjmgr コマンドの構文、サブコマンド、およびオプションに関する参照情報については、288 ページの「オブジェクトマネージャーユーティリティー」を参照してください。

オブジェクトマネージャーのほとんどの操作で、imqobjmgr コマンドのオプションとして次の情報を指定する必要があります。

- **管理対象オブジェクトの JNDI 検索名** (-l オプション)

  これは、クライアントアプリケーションが Java Naming and Directory Interface を使ってオブジェクトストア内の管理対象オブジェクトを検索するときに使用する論理名です。

- **JNDI オブジェクトストアの属性** (-j オプション)

  使用可能な属性とそれらの値については、165 ページの「オブジェクトストア」を参照してください。

- **管理対象オブジェクトのタイプ** (-t オプション)

  使用可能なタイプには次のものがあります。

  | | |
  |---|---|
  | q | キュー送信先 |
  | t | トピック送信先 |
  | cf | 接続ファクトリ |
  | qf | キュー接続ファクトリ |
  | tf | トピック接続ファクトリ |
  | xcf | 分散トランザクションの接続ファクトリ |
  | xqf | 分散トランザクションのキュー接続ファクトリ |
  | xtf | 分散トランザクションのトピック接続ファクトリ |
  | e | SOAP の端点 |

- **管理対象オブジェクトの属性** (-o オプション)

  使用可能な属性とそれらの値については、169 ページの「管理対象オブジェクトの属性」を参照してください。

# 管理対象オブジェクトの追加

imqobjmgr コマンドの add サブコマンドでは、接続ファクトリおよびトピックまたはキュー送信先管理対象オブジェクトをオブジェクトストアに追加します。LDAP オブジェクトストアに格納する管理対象オブジェクトには、接頭辞 cn= で始まる検索名を割り当てる必要があります。ファイルシステムオブジェクトストアでは、検索名を特定の接頭辞で始める必要はありませんが、スラッシュ文字 (/) を含めてはいけません。

**注** オブジェクトマネージャーは、Message Queue 管理対象オブジェクトだけを一覧表示します。オブジェクトストアに、追加したい管理対象オブジェクトと同じ検索名の Message Queue 以外のオブジェクトが含まれている場合は、追加操作を実行するとエラーが表示されます。

## 接続ファクトリの追加

クライアントアプリケーションがブローカ接続を作成できるようにするには、作成される接続のタイプに応じた接続ファクトリ管理対象オブジェクト、つまりキュー接続ファクトリまたはトピック接続ファクトリを追加します。コード例 8-1 に、キュー接続ファクトリ ( 管理対象オブジェクトのタイプ qf) を LDAP オブジェクトストアに追加するコマンドを示します。このオブジェクトは、検索名が cn=myQCF で、ホスト myHost のポート番号 7272 で実行するブローカに、jms 接続サービスを使用して接続します。

**コード例 8-1** 接続ファクトリの追加

```
imqobjmgr add
   -l "cn=myQCF"
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
   -t qf
   -o "imqAddressList=mq://myHost:7272/jms"
```

## 送信先の追加

送信先を表す管理対象オブジェクトを作成するときは、最初に物理的送信先を作成してから、管理対象オブジェクトをオブジェクトストアに追加することをお勧めします。物理的送信先を作成するには、121 ページの「物理的送信先の作成」で説明しているように、コマンドユーティリティー (imqcmd) を使用します。

コード例 8-2 に示すコマンドでは、トピック送信先を表す管理対象オブジェクトを LDAP オブジェクトストアに追加しています。検索名は myTopic で、物理的送信先の名前は physTopic です。キュー送信先を追加するためのコマンドも同様になりますが、管理対象オブジェクトのタイプ (-t オプション ) を、トピック (topic) 送信先を示す t の代わりに、キュー (queue) 送信先を示す q にします。

**コード例 8-2** LDAP オブジェクトストアへの送信先の追加

```
imqobjmgr add
   -l "cn=myTopic"
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
   -t t
   -o "imqDestinationName=physTopic"
```

コード例 8-3 に、同じコマンドで、LDAP サーバーではなく Solaris ファイルシステムに管理対象オブジェクトを格納する場合の例を示します。

**コード例 8-3** ファイルシステムオブジェクトストアへの送信先の追加

```
imqobjmgr add
   -l "cn=myTopic"
   -j "java.naming.factory.initial=
          com.sun.jndi.fscontext.RefFSContextFactory"
   -j "java.naming.provider.url=file:///home/foo/imq_admin_objects"
   -t t
   -o "imqDestinationName=physTopic"
```

# 管理対象オブジェクトの削除

管理対象オブジェクトをオブジェクトストアから削除するには、imqobjmgr コマンドの delete サブコマンドを使用して、削除するオブジェクトの検索名、タイプ、および場所を指定します。コード例 8-4 に示すコマンドでは、前述のコード例 8-2 で追加したオブジェクトを削除しています。

**コード例 8-4** 管理対象オブジェクトの削除

```
imqobjmgr delete
   -l "cn=myTopic"
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
   -t t
```

# 管理対象オブジェクトの一覧表示

オブジェクトマネージャーの `list` サブコマンドを使用して、オブジェクトストア内のすべての管理対象オブジェクトまたは特定のタイプの管理対象オブジェクトを一覧表示できます。コード例 8-5 に、LDAP サーバー上のすべての管理対象オブジェクトを一覧表示する例を示します。

**コード例 8-5** すべての管理対象オブジェクトの一覧表示

```
imqobjmgr list
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
```

コード例 8-6 では、すべてのキュー送信先 ( タイプ `q`) を一覧表示しています。

**コード例 8-6** 特定のタイプの管理対象オブジェクトの一覧表示

```
imqobjmgr list
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
   -t q
```

# 管理対象オブジェクトの情報の表示

query サブコマンドでは、検索名および格納先のオブジェクトストアの属性によって指定した管理対象オブジェクトに関する情報が表示されます。コード例 8-7 では、検索名が cn=myTopic であるオブジェクトの情報を表示しています。

**コード例 8-7** 管理対象オブジェクトの情報の表示

```
imqobjmgr query
   -l "cn=myTopic"
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
```

# 管理対象オブジェクトの属性の変更

管理対象オブジェクトの属性を変更するには、imqobjmgr update サブコマンドを使用します。オブジェクトの検索名と場所を指定し、-o オプションを使用して新しい属性値を指定します。

コード例 8-8 では、178 ページのコード例 8-1 でオブジェクトストアに追加したキュー接続ファクトリの imqReconnectAttempts 属性の値を変更しています。

**コード例 8-8** 管理対象オブジェクトの属性の変更

```
imqobjmgr update
   -l "cn=myQCF"
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
   -t qf
   -o "imqReconnectAttempts=3"
```

# コマンドファイルの使用

imqobjmgr コマンドの -i オプションでは、Java プロパティーファイルの構文を使用してサブコマンド節の全部または一部を示したコマンドファイルの名前を指定できます。この機能は特に、通常は多くの入力が必要で、imqobjmgr を何度も呼び出す場合はたいてい同じになる、オブジェクトストアの属性を指定するときに便利です。また、コマンドファイルを使用すると、コマンド行で許可されている最大文字数を超えてしまうのを避けることもできます。

コード例 8-9 に、オブジェクトマネージャーコマンドファイルの一般的な構文を示します。version プロパティーはコマンド行オプションではありません。これは、コマンドファイル自体のバージョン (Message Queue 製品のバージョンではない ) を示し、値を 2.0 に設定する必要があります。

**コード例 8-9**　　オブジェクトマネージャーコマンドファイルの構文

```
version=2.0
cmdtype=[ add | delete | list | query | update ]
obj.lookupName=lookup name
objstore.attrs.objStoreAttrName1=value1
objstore.attrs.objStoreAttrName2=value2
   . . .
objstore.attrs.objStoreAttrNameN=valueN
obj.type=[ q | t | cf | qf | tf | xcf | xqf | xtf | e ]
obj.attrs.objAttrName1=value1
obj.attrs.objAttrName2=value2
   . . .
obj.attrs.objAttrNameN=valueN
```

前述の 178 ページのコード例 8-1 で示した、LDAP オブジェクトストアにキュー接続ファクトリを追加するオブジェクトマネージャーコマンドを例として考えてみます。このコマンドは、コード例 8-10 に示すようにコマンドファイルでカプセル化できます。コマンドファイルの名前を MyCmdFile とした場合、次のコマンド行でコマンドを実行できます。

```
imqobjmgr -i MyCmdFile
```

**コード例 8-10** コマンドファイルの例

```
version=2.0
cmdtype=add
obj.lookupName=cn=myQCF
objstore.attrs.java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=ldap://mydomain.com:389/o=imq
objstore.attrs.java.naming.security.principal=¥
                                        uid=homerSimpson,ou=People,o=imq
objstore.attrs.java.naming.security.credentials=doh
objstore.attrs.java.naming.security.authentication=simple
obj.type=qf
obj.attrs.imqAddressList=mq://myHost:7272/jms
```

コマンドファイルを使用して、imqobjmgr サブコマンド節の一部だけを指定し、残りの部分はコマンド行で明示的に指定することもできます。たとえば、コード例 8-12 に示すコマンドファイルでは、LDAP オブジェクトストアの属性値だけを指定しています。

**コード例 8-11** 部分的なコマンドファイル

```
version=2.0
objstore.attrs.java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=ldap://mydomain.com:389/o=imq
objstore.attrs.java.naming.security.principal=¥
                                        uid=homerSimpson,ou=People,o=imq
objstore.attrs.java.naming.security.credentials=doh
objstore.attrs.java.naming.security.authentication=simple
```

次に、コード例 8-12 に示すように、このコマンドファイルを使用して imqobjmgr コマンドでオブジェクトストアを指定し、残りのオプションを明示的に指定できます。

**コード例 8-12** 部分的なコマンドファイルの使用

```
imqobjmgr add
   -l "cn=myQCF"
   -i MyCmdFile
   -t qf
   -o "imqAddressList=mq://myHost:7272/jms"
```

コマンドファイルのその他の例は、プラットフォームに応じて次の場所で参照できます。

Solaris: `/usr/demo/imq/imqobjmgr`
Linux: `/opt/sun/mq/examples/imqobjmgr`
Windows: `IMQ_HOME/demo/imqobjmgr`

第 9 章

# ブローカクラスタを使用した作業

Message Queue Enterprise Edition ではブローカクラスタの使用がサポートされています。ブローカクラスタでは、ブローカのグループの連動により、メッセージ配信サービスがクライアントに提供されます。メッセージサーバーでは、クラスタにより、複数のブローカ間でクライアント接続を分散し、メッセージトラフィックのボリュームで処理を拡張できます。クラスタとその動作方法の概要については、『Message Queue 技術の概要』を参照してください。

この章では、ブローカクラスタを管理する方法、ブローカをブローカクラスタに接続する方法、ブローカクラスタを設定する方法について説明します。この章は、次の節から構成されています。



## クラスタ設定プロパティー

クラスタを定義するには、メンバーブローカごとにクラスタ設定プロパティーを指定します。このプロパティーは、クラスタのブローカごとに個別に設定できますが、このプロパティーを中央のクラスタ設定ファイルに集めて、すべてのブローカに参照させる方が一般的に便利です。このようにすると、設定の不一致を防止し、クラスタのすべてのブローカで同一の一貫した設定情報を共有できます。

クラスタ設定プロパティーについては、318 ページの表 14-9 で詳しく説明します。クラスタ設定プロパティーには次のものが含まれます。

- `imq.cluster.brokerlist` では、クラスタに属しているすべてのブローカのホスト名とポート番号を指定します。
- `imq.cluster.masterbroker` では、どのブローカをマスターブローカにするかを必要に応じて指定します。マスターブローカでは状態変更を追跡します。

- imq.cluster.url では、必要に応じてクラスタ設定ファイルの場所を指定します。
- imq.cluster.hostname では、cluster 接続サービスのホスト名か IP アドレスを指定します。これは、クラスタのブローカ間の内部通信に使用されます。複数のホストを使用できる場合、この設定は便利です。たとえば、複数のネットワークインタフェースカードが 1 台のコンピュータに含まれる場合に便利です。
- imq.cluster.port では、cluster 接続サービスのポート番号を指定します。
- imq.cluster.transport では、tcp や ssl など、cluster 接続サービスで使用するトランスポートプロトコルを指定します。

hostname プロパティーと port プロパティーはブローカごとに個別に設定できますが、brokerlist、masterbroker、url、transport は、クラスタのすべてのブローカで同一の値にする必要があります。

次の節では、クラスタのブローカごとに個別に、またはクラスタ設定ファイルを使用して中央で、ブローカのクラスタ設定プロパティーを設定する方法について説明します。

# ブローカごとのクラスタプロパティーの設定

ブローカのクラスタ設定プロパティーは、インスタンス設定ファイルで、またはブローカの起動時にコマンド行で設定できます。たとえば、host1 のポート 9876、host2 のポート 5000、ctrlhost のデフォルトポート 7676 のブローカから構成されるクラスタを作成するには、3 つすべてのブローカのインスタンス設定ファイルに次のプロパティーを含めます。

```
imq.cluster.brokerlist=host1:9876,host2:5000,ctrlhost
```

この手法では、クラスタ設定を変更する必要がある場合、クラスタのブローカごとにインスタンス設定ファイルを更新する必要があることに注意してください。

# クラスタ設定ファイルの使用

一貫性を保って保守しやすくするため、ブローカごとに共有クラスタ設定プロパティーを設定する代わりに、すべての共有クラスタ設定プロパティーを 1 つのクラスタ設定ファイルに集めることをお勧めします。この手法では、それぞれのブローカのインスタンス設定ファイルで imq.cluster.url プロパティーを設定し、クラスタ設定ファイルの場所を指定する必要があります。たとえば次のように指定します。

```
imq.cluster.url=file:/home/cluster.properties
```

クラスタ設定ファイルでは、接続するブローカのリスト (imq.cluster.brokerlist)、cluster 接続サービスに使用するトランスポートプロトコル (imq.cluster.transport)、任意でマスターブローカのアドレス (imq.cluster.masterbroker) など、クラスタに属しているすべてのブローカの共有設定プロパティーを定義します。次のコードでは、前の例と同じクラスタが定義され、ctrlhost で動作するブローカがマスターブローカになります。

```
imq.cluster.brokerlist=host1:9876,host2:5000,ctrlhost
imq.cluster.masterbroker=ctrlhost
```

# クラスタ管理

この節では、ブローカのセットを接続してクラスタを形成する方法、既存クラスタに新しいブローカを追加する方法、クラスタからブローカを削除する方法について説明します。

## ブローカの接続

一般的にブローカを接続してクラスタを形成する方法には、コマンド行から行う方法 (-cluster オプションを使用 )、またはクラスタ設定ファイルで imq.cluster.brokerlist プロパティーを設定する方法の 2 つがあります。どちらの方法を使用しても、起動するそれぞれのブローカは、5 秒ごとにその他のブローカとの接続を試み、設定されている場合はマスターブローカが起動すると接続されます。マスターブローカの前にクラスタのブローカを起動すると、マスターブローカが起動するまで、そのブローカは保留状態になり、クライアント接続を拒否します。マスターブローカが起動すると、保留状態のブローカは自動的に完全に機能するようになります。

ブローカクラスタをコマンド行から設定するには、それぞれのブローカの起動時に、imqbrokerd コマンドの -cluster オプションを使用して、クラスタのブローカの完全なリストを指定します。たとえば次のコマンドでは、新しいブローカが起動し、host1 のデフォルトポート 7676、host2 のポート 5000、デフォルトホスト localhost のポート 9876 で動作しているブローカに接続されます。

```
imqbrokerd -cluster host1,host2:5000,:9876
```

本稼動システムに適した別の方法として、クラスタ設定ファイルを作成し、imq.cluster.brokerlist プロパティーを使用して、接続するブローカのリストを指定する方法があります。クラスタのそれぞれのブローカでは、独自の imq.cluster.url プロパティーを設定し、このクラスタ設定ファイルの場所を指定する必要があります。

### Linux の前提条件 : IP アドレスの設定

Linux システムでブローカを接続してクラスタを形成する場合は、特別な前提条件があります。一部の Linux インストーラでは、localhost エントリが、ネットワークループバック IP アドレス 127.0.0.1 に自動的に設定されます。クラスタのすべてのブローカでアドレスを適切にするには、システムの IP アドレスを設定する必要があります。

クラスタに加わるすべての Linux システムでは、クラスタ設定の一環として /etc/hosts ファイルをチェックしてください。システムで固定 IP アドレスを使用している場合は、/etc/hosts ファイルを編集し、localhost の正しいアドレスを指定します。アドレスがドメインネームサービス (DNS) に登録されている場合は、/etc/nsswitch.conf ファイルを編集してエントリの順序を変更し、システムが DNS 検索を実行してから、ローカルの hosts ファイルを参照するように設定します。/etc/nsswitch.conf ファイルの行は次のようになります。

```
hosts: dns files
```

### ブローカ間の安全な接続

安全で暗号化されたメッセージ配信がクラスタのブローカ間で必要である場合は、SSL ベースのトランスポートプロトコルを使用するように cluster 接続サービスを設定します。151 ページの「SSL ベースのサービスの操作」で説明するように、クラスタのブローカごとに、SSL ベースの接続サービスを設定します。次にそれぞれのブローカの imq.cluster.transport プロパティーを、クラスタ設定ファイルでまとめて、またはブローカごとに個別に、ssl に設定します。

## クラスタへのブローカの追加

新しいブローカをクラスタに追加する手順は、クラスタでクラスタ設定ファイルを使用しているかどうかによって決まります。

### ➤ クラスタ設定ファイルを使用して新しいブローカをクラスタに追加する

1. クラスタ設定ファイルにある imq.cluster.brokerlist プロパティーに、新しいブローカを追加します。
2. クラスタ内の各ブローカに次のコマンドを実行します。

   ```
   imqcmd reload cls
   ```

   それぞれのブローカでクラスタ設定が再読み込みされ、クラスタに属しているブローカのすべての一貫した情報が最新になります。
3. ( 任意指定 ) ブローカの config.properties ファイルで imq.cluster.url プロパティーの値をクラスタ設定ファイルの場所に設定します。

4. 新しいブローカを起動します。

   手順 3 を実行しなかった場合は、imqbrokerd コマンド行で -D オプションを使用し、imq.cluster.url の値を設定します。

### ➤ クラスタ設定ファイルを使用せずに新しいブローカをクラスタに追加する

config.properties ファイルを編集するか、imqbrokerd コマンド行で -D オプションを使用し、次のプロパティー値を設定します。

- imq.cluster.brokerlist
- imq.cluster.masterbroker ( 必要に応じて )
- imq.cluster.transport ( 安全な cluster 接続サービスを使用している場合 )

# クラスタからのブローカの削除

クラスタからブローカを削除する方法は、最初にコマンド行でクラスタを作成したか、中央のクラスタ設定ファイルによって作成したかによって決まります。

## コマンド行を使用したブローカの削除

コマンド行から imqbrokerd コマンドを使用してブローカをクラスタに接続した場合は、それぞれのブローカを停止してから、コマンド行に新しいクラスタメンバーセットを指定してブローカを再起動する必要があります。その手順は次のとおりです。

### ➤ コマンド行を使用してクラスタからブローカを削除する

1. imqcmd コマンドを使用し、クラスタのそれぞれのブローカを停止します。
2. imqbrokerd コマンドの -cluster オプションを使用し、クラスタに残すブローカのみを指定してそれらのブローカを再起動します。

   たとえば、次のコマンドを使用して、*A*、*B*、*C* というそれぞれのブローカを起動し、その 3 つのブローカから構成されるクラスタを最初に作成したとします。

   ```
   imqbrokerd -cluster A,B,C
   ```

   ブローカ *A* をクラスタから削除するには、次のコマンドを使用してブローカ *B* と *C* を再起動します。

   ```
   imqbrokerd -cluster B,C
   ```

### クラスタ設定ファイルを使用したブローカの削除

中央のクラスタ設定ファイルの imq.cluster.brokerlist プロパティーでメンバーブローカを指定してクラスタを最初に作成した場合、ブローカを停止してメンバーのうち 1 つのブローカを削除する必要はありません。単純に設定ファイルを編集して削除したいブローカを除外し、残りのクラスタメンバーにクラスタ設定を再読み込みさせます。除外するブローカは、同じクラスタ設定ファイルの場所を指定しないように再設定します。手順は次のとおりです。

#### ➤ クラスタ設定ファイルを使用してクラスタからブローカを削除する

1. クラスタ設定ファイルを編集し、imq.cluster.brokerlist プロパティーに指定しているリストから除外対象ブローカを削除します。
2. クラスタ内の残りのブローカに次のコマンドを実行します。

   ```
   imqcmd reload cls
   ```

   ブローカがクラスタ設定を再読み込みします。
3. クラスタから削除するブローカを停止します。
4. そのブローカの config.properties ファイルを編集し、imq.cluster.url プロパティーを削除するか、別の値を指定します。

# マスターブローカ

クラスタには、1 つのマスターブローカを任意に含めることができます。マスターブローカでは設定変更レコードが維持され、クラスタの持続的な状態の変更が追跡されます。マスターブローカは、クラスタ設定ファイル、またはそれぞれのブローカのインスタンス設定ファイルで、imq.cluster.masterbroker 設定プロパティーによって識別されます。

設定変更レコードには、永続サブスクリプション、および管理者が作成した物理的送信先など、クラスタに関連する持続エンティティーの変更に関する情報が含まれます。クラスタのすべてのブローカは、起動中にマスターブローカを参照し、この持続エンティティーに関する情報を更新します。このような同期は、マスターブローカの障害によって不可能になります。詳細については、191 ページの「マスターブローカを使用できない場合」を参照してください。

# 設定変更レコードの管理

設定変更レコードには重要な情報が含まれるので、定期的にバックアップして、障害が発生した場合に復元できるようにすることが重要です。バックアップから復元しても、バックアップ以降に発生したクラスタの持続的な状態の変更は失われますが、頻繁にバックアップすれば、情報喪失の可能性を最小限に抑えることができます。バックアップ操作と復元操作には、時間の経過とともに増大していく可能性がある設定変更レコード内の変更履歴を、圧縮して最適化するという肯定的な効果もあります。

### ➤ 設定変更レコードをバックアップする

imqbrokerd コマンドの -backup オプションを使用し、バックアップファイルの名前を指定します。たとえば、次のように指定します。

```
imqbrokerd -backup mybackuplog
```

### ➤ 設定変更レコードを復元する

1. クラスタにあるすべてのブローカをシャットダウンします。
2. 次のコマンドを使用し、マスターブローカの設定変更レコードをバックアップファイルから復元します。

   ```
   imqbrokerd -restore mybackuplog
   ```

3. 新しい名前やポート番号をマスターブローカに割り当てる場合は、クラスタ設定ファイルの imq.cluster.brokerlist プロパティーと imq.cluster.masterbroker プロパティーを相応に更新します。
4. クラスタにあるすべてのブローカを再起動します。

# マスターブローカを使用できない場合

クラスタのすべてのブローカでは、持続的な操作を実行するためにマスターブローカが必要になるので、マスターブローカを使用できない場合、クラスタのすべてのブローカでは次の imqcmd サブコマンドがエラーになります。

- create dst
- destroy dst
- update dst
- destroy dur

自動作成の物理的送信先および一時的送信先は影響されません。

マスターブローカがない場合、永続サブスクライバを作成したり、永続サブスクリプションから登録解除しようとするすべてのクライアントアプリケーションではエラーが発生します。ただしクライアントは、既存の永続サブスクリプションを指定したり、既存の永続サブスクリプションとやり取りしたりすることはできます。

第 10 章

# メッセージサーバーの監視

この章では、メッセージサーバーの監視に使用できるツール、およびメトリックスデータの取得方法について説明します。この章では、次の節について説明します。



特定メトリックスの詳細については、第 18 章「メトリックスのリファレンス」を参照してください。

## 監視ツールの概要

Message Queue 情報の監視インタフェースには、ログファイル、対話型コマンド、メトリックスを取得できるクライアント API の 3 つがあります。それぞれのツールには、次のような長所と短所があります。

- ログファイルでは、メトリックスデータの長期間の記録が提供されますが、簡単には解析できません。
- コマンドでは、ニーズに合った情報を迅速にサンプル抽出できますが、履歴情報を調べたり、プログラムでデータを操作したりすることはできません。
- クライアント API では、情報の抽出、処理、データの操作、グラフ表示、警告の送信を行うことができます。ただし、クライアント API を使用するには、カスタムアプリケーションを作成して、データの取得と分析を行う必要があります。

表 10-1 は、さまざまなツールの比較です。

表 10-1　メトリックス監視ツールの長所と短所

| メトリックス監視ツール | 長所 | 短所 |
|---|---|---|
| imqcmd metrics | リモート監視<br>スポット検査に適しています<br>報告間隔はコマンドのオプションで設定されるため、即座に変更可能です<br>対象となる特定のデータを容易に選択できます<br>わかりやすい表形式でデータを表示します | シングルコマンドではすべてのデータを取得できません<br>データ分析のプログラム化が困難です<br>履歴レコードを作成しません<br>履歴的な傾向を確認するのが困難です |
| ログファイル | 定期的なサンプリング<br>履歴レコードの作成 | ブローカプロパティーの設定が必要です。有効にするにはブローカをシャットダウンし再起動する必要があります<br>ローカル監視のみ<br>読み取りや解析が非常に困難なデータ形式です。解析ツールはありません<br>報告間隔を即座に変更できません。すべてのメトリックスデータについて同じです<br>柔軟にデータを選択できません<br>ブローカメトリックスのみ。送信先と接続サービスのメトリックスは含まれていません<br>間隔が短過ぎるとパフォーマンスに影響する可能性があります |

表 10-1　メトリックス監視ツールの長所と短所 ( 続き )

| メトリックス監視ツール | 長所 | 短所 |
|---|---|---|
| クライアント API | リモート監視<br>対象となる特定のデータを容易に選択できます<br>データをプログラムで分析し、任意の形式で提示できます | ブローカプロパティーの設定が必要です。有効にするにはブローカをシャットダウンし再起動する必要があります<br>専用のメトリックス監視クライアントをプログラミングする必要があります<br>報告間隔を即座に変更できません。すべてのメトリックスデータについて同じです |

この表に掲載されている違いに加えて、それぞれのツールでは、ブローカによって生成されたメトリックス情報の、多少異なるサブセットが収集されます。どの監視ツールがどのメトリックスデータを収集するかについては、349 ページの第 18 章「メトリックスのリファレンス」を参照してください。

# ブローカロギングの設定と使用

Message Queue ロガーでは、ブローカコード、デバッガ、メトリックスジェネレータによって生成された情報が取得され、その情報が、標準出力 ( コンソール )、ログファイル、Solaris™ オペレーティングシステムの `syslog` デーモンプロセスなど、多くの出力チャネルに書き込まれます。

ロガーが収集する情報のタイプと、各出力チャネルに書き込む情報のタイプを指定できます。特に、メトリックス情報のログファイルへの書き込みを指定できます。

この節では、ブローカのデフォルトロギング設定、代替出力チャネルにログ情報をリダイレクトする方法、ログファイルロールオーバー基準の変更方法、メトリックスデータをログファイルに送信する方法について説明します。

# デフォルトのロギングの設定

ブローカは、ローリングログファイルのセットにログ出力を保存するように自動的に設定されます。ログファイルは、関連付けられたブローカのインスタンス名によって識別されるディレクトリに配置されます ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

```
.../instances/instanceName/log/
```

ログファイルは、簡単なテキストファイルです。ログファイルは、次のように順番に名前が付けられています。

```
log.txt
log_1.txt
log_2.txt
...
log_9.txt
```

デフォルトでは、ログファイルは週に 1 回ロールオーバーされ、システムは 9 つのバックアップファイルを保持します。

- ログファイルが保管されるディレクトリを変更するには、`imq.log.file.dirpath` プロパティーを希望するパスに設定します。
- ログファイルのルート名を `log` から別の名前に変更するには、`imq.log.file.filename` プロパティーを設定します。

ブローカでは、`ERROR`、`WARNING`、`INFO` という、3 つのログレベルがサポートされます。表 10-2 では、それぞれのレベルについて説明します。

表 10-2　ロギングレベル

| レベル | 説明 |
|---|---|
| `ERROR` | システム障害が生じる可能性のある問題点を示すメッセージです。 |
| `WARNING` | システム障害が生じる可能性はないが、留意すべき警告です。 |
| `INFO` | メトリックスおよびその他の情報メッセージの報告です。 |

ロギングレベルを設定すると、そのレベル以上のメッセージが収集されます。デフォルトのログレベルは `INFO` なので、`ERROR` メッセージ、`WARNING` メッセージ、`INFO` メッセージはすべてデフォルトで記録されます。

# ログメッセージの書式設定

ログメッセージは、タイムスタンプ、メッセージコード、メッセージ自体から構成されます。情報量は、設定したログレベルにより異なります。INFO メッセージの例を次に示します。

```
[13/Sep/2000:16:13:36 PDT] B1004 Starting the broker service
using tcp [ 25374,100] with min threads 50 and max threads of 500
```

タイムスタンプのタイムゾーンを変更するには、313 ページの表 14-8 に示す imq.log.timezone プロパティーに関する情報を参照してください。

# ロガー設定の変更

ログ関連のプロパティーについては、313 ページの表 14-8 を参照してください。

### ➤ ブローカのロガー設定を変更する

1. ログレベルを設定します。
2. 1 つまたはそれ以上のロギングカテゴリの出力チャネル ( ファイル、コンソール、またはその両方 ) を設定します。
3. 出力をファイルに記録する場合、ファイルのロールオーバー基準を設定します。

ロガープロパティーを設定すると、手順は完了します。ロガープロパティーの設定は、次のどちらかの方法で行います。

- ブローカを起動する前に、ブローカの config.properties ファイルでロガープロパティーを変更または追加します。
- ブローカを起動する imqbrokerd コマンドでロガーコマンド行オプションを指定します。また、オプション -D を使用して、ロガープロパティー、または任意のブローカプロパティーを変更できます。

コマンド行で渡されたオプションは、ブローカインスタンス設定ファイルで指定されたプロパティーをオーバーライドします。表 10-3 に、ロギングに影響する imqbrokerd オプションを一覧表示します。

表 10-3　imqbrokerd ロガーオプションと対応するプロパティー

| imqbrokerd オプション | 説明 |
|---|---|
| -metrics *interval* | メトリックス情報がロガーに書き込まれる間隔を秒単位で指定します。 |
| -loglevel *level* | ERROR、WARNING、INFO のいずれかにログレベルを設定します。 |
| -silent | コンソールへのロギングをオフにします。 |
| -tty | すべてのメッセージをコンソールに送信します。デフォルトでは、WARNING レベルと ERROR レベルのメッセージだけが表示されます。 |

続いて、デフォルトの設定を変更して、次のことを実行する方法を説明します。

- ログメッセージの送信先である、出力チャネルの変更
- ロールオーバー基準の変更

## 出力チャネルの変更

デフォルトでは、エラーメッセージと警告メッセージは、ログファイルに記録されると同時に、端末に表示されます。Solaris では、エラーメッセージはシステムの syslog デーモンにも書き込まれます。

次の方法で、ログメッセージの出力チャネルを変更できます。

- 指定したレベルのすべてのログカテゴリを画面に表示するには、imqbrokerd コマンドの -tty オプションを使用します。
- ログ出力を画面に表示しないようにするには、imqbrokerd コマンドの -silent オプションを使用します。
- imq.log.file.output プロパティーを使用して、ログファイルに書き込むロギング情報のカテゴリを指定します。たとえば、次のように指定します。

  ```
  imq.log.file.output=ERROR
  ```

- imq.log.console.output プロパティーを使用して、コンソールに書き込むロギング情報のカテゴリを指定します。たとえば、次のように指定します。

  ```
  imq.log.console.output=INFO
  ```

- Solaris の場合、imq.log.syslog.output プロパティーを使用して、Solaris syslog に書き込むロギング情報のカテゴリを指定します。たとえば、次のように指定します。

  ```
  imq.log.syslog.output=NONE
  ```

**注** ロガー出力チャネルを変更する前に、出力チャネルにマッピングされた情報をサポートするレベルにロギングが設定されていることを確認する必要があります。たとえば、ログレベルを ERROR に設定し、imq.log.console.output プロパティーを WARNING に設定すると、WARNING メッセージのロギングが有効になっていないため、どのメッセージも記録されません。

## ログファイルのロールオーバー基準の変更

ログファイルのロールオーバーには、時間とサイズの 2 つの基準があります。デフォルトでは時間の基準が使用され、7 日ごとにファイルがロールオーバーされます。

- 時間の間隔を変更するには、imq.log.file.rolloversecs プロパティーを変更する必要があります。たとえば、次のようにプロパティーを定義すると、間隔が 10 日に変更されます。

  ```
  imq.log.file.rolloversecs=864000
  ```

- ファイルサイズに従ってロールオーバーするように基準を変更するには、imq.log.file.rolloverbytes プロパティーを設定する必要があります。たとえば、次のように定義すると、500,000 バイトの制限に達したあと、ブローカはファイルをロールオーバーするように指示されます。

  ```
  imq.log.file.rolloverbytes=500000
  ```

時間に関連するロールオーバープロパティーとサイズに関連するロールオーバープロパティーの両方が設定されている場合は、どちらかの制限に最初に達したときにロールオーバーが実行されます。前の節でも説明したように、ブローカでは 9 つのロールオーバーファイルが保持されます。

ログファイルのロールオーバープロパティーの設定や変更は、ブローカの動作時に実行できます。このプロパティーを設定するには、imqcmd update bkr コマンドを使用します。

## ログファイルへのメトリックスデータの送信

この節では、ブローカログファイルを使用してメトリックス情報を報告するための手順を説明します。ロガーの設定方法については、195 ページの「ブローカロギングの設定と使用」を参照してください。

### ➤ ログファイルを使用してメトリックス情報を報告する

1. ブローカのメトリックス生成機能を設定します。

   a. imq.metrics.enabled=true に設定されていることを確認します。

      デフォルトでは、ロギング用のメトリックスの生成は有効になっています。

b. メトリックスの生成間隔を適切な秒数に設定します。

```
imq.metrics.interval=interval
```

この値は、config.properties ファイル内で設定するか、またはブローカの起動時に -metrics *interval* コマンド行オプションを使用して設定できます。

2. ロガーがメトリックス情報を収集していることを確認します。

```
imq.log.level=INFO
```

これはデフォルト値です。この値は、config.properties ファイル内で設定するか、またはブローカの起動時に -loglevel *level* コマンド行オプションを使用して設定できます。

3. ロガーが、メトリックス情報をログファイルへ書き込むように設定されていることを確認します。

```
imq.log.file.output=INFO
```

これはデフォルト値です。config.properties ファイル内で設定できます。

4. ブローカを起動します。

ログファイルに出力されたブローカメトリックスの例を次に示します。

```
[21/Jul/2004:11:21:18 PDT]
Connections:0    JVM Heap:8323072 bytes (7226576 free) Threads: 0 (14-1010)
      In:0 msgs (0bytes) 0 pkts (0 bytes)
     Out:0 msgs (0bytes) 0 pkts (0 bytes)
 Rate In: 0 msgs/sec (0 bytes/sec) 0 pkts/sec (0 bytes/sec)
Rate Out: 0 msgs/sec (0 bytes/sec) 0 pkts/sec (0 bytes/sec)
```

メトリックスデータの詳細については、第 18 章「メトリックスのリファレンス」を参照してください。

## デッドメッセージのロギング

ブローカのデッドメッセージロギングを有効にすると、物理的送信先を監視できます。デッドメッセージキューを使用しているかどうかに関係なく、デッドメッセージを記録できます。

デッドメッセージロギングを有効にすると、次のタイプのイベントが、ブローカによって記録されます。

- 物理的送信先が最大サイズを超えた。
- 次のような理由により、ブローカが物理的送信先からメッセージを削除した。
  - 送信先サイズが制限に達した。

- メッセージの生存時間が満了した。
- メッセージが長すぎる。
- ブローカがメッセージを処理しようとしたときにエラーが発生した。

デッドメッセージキューを使用している場合、ロギングには次のタイプのイベントも含まれます。

- ブローカがデッドメッセージキューにメッセージを移動した。
- ブローカがデッドメッセージキューからメッセージを削除して破棄した。

デッドメッセージのロギングは、デフォルトでは無効になっています。有効にするには、ブローカ属性 `imq.destination.logDeadMsgs` を設定します。

# メトリックスの対話型表示

Message Queue ブローカでは、次のタイプのメトリックスが報告されます。

- **Java 仮想マシン (JVM) メトリックス** : JVM ヒープサイズに関する情報です。
- **ブローカ全体のメトリックス** : ブローカに保存されているメッセージ、ブローカで入出力されるメッセージ、メモリー使用に関する情報です。メッセージは、メッセージ数とバイト数の点で追跡されます。
- **接続サービスのメトリックス** : 接続と接続スレッドのリソースに関する情報、および特定の接続サービスのメッセージフローに関する情報です。
- **送信先メトリックス** : 特定の物理的送信先との間のメッセージフロー、物理的送信先のコンシューマ、メモリーとディスクスペースの使用率に関する情報です。

`imqcmd` コマンドでは、ブローカ全体、それぞれの接続サービス、それぞれの物理的送信先のメトリックス情報を取得できます。メトリックスデータを取得するには、一般に、`imqcmd` の `metrics` サブコマンドを使用します。メトリックスデータは、指定した間隔で、または指定した回数だけ、コンソール画面に表示されます。

`query` サブコマンドを使用し、設定情報も含む、同様のデータを表示することもできます。詳細については、206 ページの「imqcmd query」を参照してください。

# imqcmd metrics

imqcmd metrics の構文とオプションを、それぞれ表 10-4 と表 10-5 に示します。

表 10-4　imqcmd metrics サブコマンドの構文

| サブコマンドの構文 | 提供されるメトリックスデータ |
|---|---|
| metrics bkr<br>[-b *hostName*:*portNumber*]<br>[-m *metricType*]<br>[-int *interval*]<br>[-msp *numSamples*] | デフォルトのブローカ、または指定したホストとポートのブローカに関して、ブローカのメトリックスを表示します。 |
| または | |
| metrics svc -n *serviceName*<br>[-b *hostName*:*portNumber*]<br>[-m *metricType*]<br>[-int *interval*]<br>[-msp *numSamples*] | デフォルトのブローカ、または指定したホストとポートのブローカで実行している特定のサービスのメトリックスを表示します。 |
| または | |
| metrics dst -t *destType*<br>-n *destName*<br>[-b *hostName*:*portNumber*]<br>[-m *metricType*]<br>[-int *interval*]<br>[-msp *numSamples*] | 特定のタイプと名前の物理的送信先に関するメトリックス情報を表示します。 |

表 10-5　imqcmd metrics サブコマンドのオプション

| サブコマンドのオプション | 説明 |
|---|---|
| -b *hostName*:*portNumber* | メトリックスデータを報告するホスト名とブローカのポートを指定します。デフォルトは localhost:7676 です。 |
| -int *interval* | メトリックスが表示される間隔を秒単位で指定します。デフォルトは 5 秒です。 |

表 10-5　imqcmd metrics サブコマンドのオプション ( 続き )

| サブコマンドのオプション | 説明 |
|---|---|
| -m *metricType* | 表示するメトリックスのタイプを指定します。<br>**ttl**:　ブローカ、サービス、または送信先で入出力されているメッセージとパケットのフローに関するメトリックスを表示します ( デフォルトのメトリックスタイプ )。<br>**rts**:　ブローカ、接続サービス、または送信先で入出力されているメッセージとパケットのフローレートに関するメトリックスを表示します ( 秒単位 )。<br>**cxn**:　接続、仮想メモリーヒープ、およびスレッドを表示します ( ブローカと接続サービスのみ )。<br>**con**:　コンシューマ関連のメトリックスを表示します ( 送信先のみ )。<br>**dsk**:　ディスク使用量のメトリックスを表示します ( 送信先のみ )。 |
| -msp *numSamples* | 出力に表示するサンプルの数を指定します。デフォルトは無制限です ( 無限 )。 |
| -n *destName* | 必要に応じて、メトリックスデータを報告する物理的送信先の名前を指定します。デフォルトはありません。 |
| -n *serviceName* | 必要に応じて、メトリックスデータを報告する接続サービスを指定します。デフォルトはありません。 |
| -t *destType* | 必要に応じて、メトリックスデータを報告する物理的送信先のタイプ ( キューまたはトピック ) を指定します。デフォルトはありません。 |

# metrics サブコマンドを使用したメトリックスデータの表示

この節では、metrics サブコマンドを使用してメトリックス情報を報告するための手順を説明します。

➤ **metrics サブコマンドを使用する**

1. メトリックス情報が必要なブローカを起動します。

   68 ページの「ブローカの起動」を参照してください。

2. 表 10-4 と表 10-5 に示すオプションを指定して、適切な imqcmd metrics サブコマンドを実行します。

# メトリックスの出力 : imqcmd metrics

この節には、imqcmd metrics サブコマンドの出力例が含まれています。この例では、ブローカ全体のメトリックス、接続サービスのメトリックス、物理的送信先のメトリックスが示されています。

## ブローカ全体のメトリックス

ブローカとの間のメッセージとパケットのフローレートを 10 秒間隔で取得するには、metrics bkr サブコマンドを使用します。

```
imqcmd metrics bkr -m rts -int 10 -u admin
```

このコマンドは、次のような出力を生成します (350 ページの表 18-2 のデータの説明を参照 )。

```
-------------------------------------------------------------
 Msgs/sec    Msg Bytes/sec    Pkts/sec     Pkt Bytes/sec
 In    Out      In       Out      In    Out      In       Out
-------------------------------------------------------------
 0     0        27       56       0     0        38       66
 10    0      7365       56       10    10     7457     1132
 0     0        27       56       0     0        38       73
 0     10       27     7402       10    20     1400     8459
 0     0        27       56       0     0        38       73
```

## 接続サービスのメトリックス

jms 接続サービスが処理したメッセージとパケットの累計を取得するには、metrics svc サブコマンドを使用します。

```
imqcmd metrics svc -n jms -m ttl -u admin
```

このコマンドは、次のような出力を生成します (352 ページの表 18-3 のデータの説明を参照 )。

```
-----------------------------------------------
  Msgs       Msg Bytes       Pkts       Pkt Bytes
In   Out     In      Out    In   Out     In      Out
-----------------------------------------------
164  100   120704  73600   282  383   135967  102127
657  100   483552  73600   775  876   498815  149948
```

## 物理的送信先のメトリックス

物理的送信先に関するメトリックス情報を表示するには、metrics dst サブコマンドを使用します。

```
imqcmd metrics dst -t q -n XQueue -m ttl -u admin
```

このコマンドは、次のような出力を生成します (355 ページの表 18-4 のデータの説明を参照 )。

```
-------------------------------------------------------------------------------
  Msgs       Msg Bytes          Msg Count          Total Msg Bytes
(k)      Largest
In   Out     In      Out     Current  Peak  Avg   Current  Peak      Avg     Msg (k)
-------------------------------------------------------------------------------
200  200   147200  147200      0      200    0       0       143       71         0
300  200   220800  147200    100      200   10      71       143       64         0
300  300   220800  220800      0      200    0       0       143       59         0
```

物理的送信先のコンシューマに関する情報を取得するには、次の metrics dst サブコマンドを使用します。

```
imqcmd metrics dst -t q -n SimpleQueue -m con -u admin
```

このコマンドは、次のような出力を生成します (355 ページの表 18-4 のデータの説明を参照 )。

```
-------------------------------------------------------------------
  Active Consumers          Backup Consumers          Msg Count
Current   Peak   Avg      Current   Peak    Avg    Current  Peak  Avg
-------------------------------------------------------------------
   1        1      0         0        0       0       944    1000  525
```

# imqcmd query

`imqcmd query` の構文とオプションを、コマンドによって提供されるメトリックスデータの説明とともに表 10-6 に示します。

**表 10-6** `imqcmd query` サブコマンドの構文

| サブコマンドの構文 | 提供されるメトリックスデータ |
|---|---|
| `query bkr [-b` *hostName*`:`*portNumber*`]` | ブローカのメモリーと持続ストアに格納されている現在のメッセージ数とメッセージバイト数に関する情報 (103 ページの「ブローカ情報の表示」を参照 )。 |
| または | |
| `query svc -n` *serviceName* `[-b` *hostName*`:`*portNumber*`]` | 指定した接続サービスに現在割り当てられているスレッドの数とそのサービスの接続数に関する情報 (109 ページの「接続サービス情報の表示」を参照 )。 |
| または | |
| `query dst -t` *destType* `-n` *destName* `[-b` *hostName*`:`*portNumber*`]` | 指定した送信先のメモリーと持続ストアに格納されている現在のプロデューサ数、アクティブコンシューマとバックアップコンシューマの数、メッセージ数とメッセージバイト数に関する情報 (124 ページの「物理的送信先の情報の表示」を参照 )。 |

**注** `imqcmd query` は限定されたメトリックスデータを提供するため、このツールは、349 ページの第 18 章「メトリックスのリファレンス」の表には記載されていません。

# ブローカを監視するアプリケーションの作成

Message Queue には、ブローカがメトリックスデータを JMS メッセージへ書き込み、そのメッセージに含まれるメトリックス情報のタイプに応じて、そのメッセージを多数のメトリックストピック送信先のどれかに送信する際に使用できるメトリックス監視機能が用意されています。

メトリックストピックを送信先へサブスクライブし、これらの送信先のメッセージを消費し、メッセージに含まれるメトリックス情報を処理するクライアントアプリケーションをプログラミングすることで、このメトリックス情報にアクセスできます。

5 つのメトリックストピック送信先があります。それらの名前と、各送信先へ配信されるメトリックスメッセージのタイプを表 10-7 に示します。

**表 10-7** メトリックスのトピック送信先

| トピック名 | メトリックスメッセージのタイプ |
|---|---|
| mq.metrics.broker | ブローカのメトリックス |
| mq.metrics.jvm | Java 仮想マシンのメトリックス |
| mq.metrics.destination_list | 送信先とそれらのタイプのリスト |
| mq.metrics.destination.queue.*monitoredDestinationName* | 指定した名前のキューの送信先メトリックス |
| mq.metrics.destination.topic.*monitoredDestinationName* | 指定した名前のトピックの送信先メトリックス |

## メッセージベースの監視の設定

この節では、メッセージベースの監視機能を使用してメトリックス情報を収集するための手順を説明します。手順には、クライアント開発タスクと管理タスクの両方が含まれます。

### ➤ メッセージベースの監視を設定する

1. メトリックス監視クライアントを作成します。

   メトリックストピック送信先へサブスクライブし、メトリックスメッセージを消費し、これらのメッセージからメトリックスデータを抽出するクライアントをプログラミングする手順については、『Message Queue Developer's Guide for Java Clients』を参照してください。

2. `config.properties` ファイルにブローカプロパティー値を設定して、ブローカのメトリックスメッセージプロデューサを設定します。

a. メトリックスメッセージの生成を有効にします。

   `imq.metrics.topic.enabled=true` と設定します。

   デフォルト値は `true` です。

b. メトリックスメッセージを生成する間隔を、秒単位で指定します。

   `imq.metrics.topic.interval=`*interval* と設定します。

   デフォルトは 60 秒です。

c. メトリックスメッセージを持続的にするかどうか、つまり、ブローカ障害時にもそのまま保持するかどうかを指定します。

   `imq.metrics.topic.persist` を設定します。

   デフォルト値は `false` です。

d. 各送信先で、メトリックスメッセージを削除するまでに保持しておく期間を指定します。

   `imq.metrics.topic.timetolive` を設定します。

   デフォルト値は 300 秒です。

3. メトリックストピック送信先に必要なアクセス制御を設定します。

   設定については次の「セキュリティーとアクセスで考慮すること」を参照してください。

4. メトリックス監視クライアントを起動します。

   コンシューマがメトリックストピックをサブスクライブすると、メトリックストピック送信先が自動的に作成されます。メトリックストピックが作成されると、ブローカのメトリックスメッセージプロデューサがメトリックスメッセージをメトリックストピックへ送信し始めます。

# セキュリティーとアクセスで考慮すること

メトリックストピック送信先へのアクセスを制限する理由は 2 つあります。

- メトリックスデータにブローカとそのリソースに関する機密情報が含まれることがある。
- メトリックストピック送信先へのサブスクリプション数が過剰になると、ブローカのオーバーヘッドが増加し、パフォーマンスに悪影響を及ぼすことがある。

これらの点を考慮して、メトリックストピック送信先へのアクセスは制御することをお勧めします。

監視クライアントは、そのほかのクライアントと同じ認証制御と権限を前提にしています。ブローカへの接続が許可されるのは、Message Queue ユーザーリポジトリに登録されているユーザーだけです。

144 ページの「ユーザーの承認 : アクセス制御プロパティーファイル」に説明されているとおり、アクセス制御プロパティーファイルを使用して特定のメトリックストピック送信先へのアクセスを制限することで、さらに保護を強化できます。

たとえば、`accesscontrol.properties` ファイル内の次のエントリは、user1 と user2 を除き、すべてのユーザーについて `mq.metrics.broker` メトリックストピックへのアクセスを拒否します。

```
topic.mq.metrics.broker.consume.deny.user=*
topic.mq.metrics.broker.consume.allow.user=user1,user2
```

次のエントリは、ユーザー user3 だけにトピック t1 の監視を許可します。

```
topic.mq.metrics.destination.topic.t1.consume.deny.user=*
topic.mq.metrics.destination.topic.t1.consume.allow.user=user3
```

メトリックスデータの機密性に応じて、暗号化された接続を使用してメトリックス監視クライアントをブローカへ接続することもできます。暗号化された接続の使用方法については、151 ページの「SSL ベースのサービスの操作」を参照してください。

# メトリックスの出力 : メトリックスメッセージ

メッセージベースの監視 API を使用して取得したメトリックスデータ出力は、プログラミングしたメトリックス監視クライアントによって異なります。出力されるデータは、ブローカ内のメトリックスジェネレータによって提供されるデータだけに限定されます。このデータの完全なリストは、349 ページの「メトリックスのリファレンス」を参照してください。

第 11 章

# メッセージサービスの分析と調整

この章では、Message Queue サービスの分析と調整を行い、メッセージングアプリケーションのパフォーマンスを最適化する方法に関連するさまざまなトピックを取り上げます。次のトピックが含まれます。



## パフォーマンス関連

この節では、パフォーマンス調整の背景について説明します。

### パフォーマンス調整プロセス

メッセージングアプリケーションのパフォーマンスは、アプリケーションと Message Queue サービスの相互関係に左右されます。そのため、パフォーマンスを最大化するには、アプリケーション開発者と管理者が協力し合う必要があります。

パフォーマンスを最適化するプロセスは、アプリケーションの設計から始まります。アプリケーションが配置されたあとも、継続してメッセージサービスの調整を行います。パフォーマンス調整プロセスには、次の段階があります。

- アプリケーションのパフォーマンス要件を定義します。
- 特に信頼性とパフォーマンスの兼ね合いなど、パフォーマンスに影響する要因を考慮してアプリケーションを設計します。
- パフォーマンスの基準を設けます。
- パフォーマンスを最適化するためにメッセージサービスを調整または再設定します。

通常は、上記に概略したプロセスを繰り返し実行します。アプリケーションの配置時に、Message Queue 管理者は、メッセージサーバーの適合性を評価し、アプリケーションの全般的なパフォーマンス要件を満たしているかどうかを判断します。ベンチマークテストがこれらの要件を満たす場合は、この章で説明するとおり、管理者はシステムの調整段階に入ることができます。一方、ベンチマークテストがパフォーマンス要件を満たしていない場合は、アプリケーションの再設計や配置アーキテクチャーの変更が必要となる場合があります。

# パフォーマンスのさまざまな側面

一般に、パフォーマンスの基準は、メッセージサービスがプロデューサからコンシューマへメッセージを配信するときの速度と効率です。ただし、パフォーマンスには、ニーズに応じて重要度が変わるさまざまな側面があります。

**接続の負荷**：メッセージプロデューサまたはメッセージコンシューマの数、もしくは、システムがサポート可能な同時接続の数です。

**メッセージのスループット**：メッセージングシステムが 1 秒間に扱えるメッセージ数、またはメッセージのバイト数です。

**遅延**：特定のメッセージがメッセージプロデューサからメッセージコンシューマへ配信されるまでに要する時間です。

**安定性**：メッセージサービス全体の可用性、つまり過負荷や障害による影響をどれだけ抑えられるかです。

**効率**：メッセージ配信の効率。使用するコンピュータリソースに関係する、メッセージスループットの評価です。

パフォーマンスのこれらの異なる評価基準は、一般に相互に関連しています。メッセージスループットが高い場合、メッセージがメッセージサーバーへバックログされることは、ほとんどありません。その結果、遅延も短くなり、シングルメッセージはすぐに配信されます。ただし、遅延はさまざまな要因に左右されます。そのような要因の例としては、通信リンクの速度、メッセージサーバーの処理速度、クライアントの処理速度などがあります。

どのような場合でも、パフォーマンスには複数の異なる側面があります。一般に、その中でどれがもっとも重要となるかは、特定のアプリケーションの要件によって決まります。

# ベンチマーク

ベンチマークとは、使用中のメッセージングアプリケーション用のテスト群を作成し、このテスト群を用いてメッセージスループットや、そのほかの観点からパフォーマンスを評価するプロセスです。

たとえば、複数のプロデューシングクライアントを対象に、複数の、接続、セッション、メッセージプロデューサを使用し、標準サイズの持続メッセージまたは持続性のないメッセージを一部のキューやトピック ( すべてメッセージングアプリケーションの設計に依存 ) へ一定レートで送信するテスト群を作成できます。同様に、特定の通知モードでテスト群の物理的送信先においてメッセージを消費する複数の接続、セッション、および特定タイプのメッセージコンシューマを使用し、複数のコンシューミングクライアントをテスト群に含められます。

標準のテスト群を使用することで、メッセージが生成されてから消費されるまでに要する時間やメッセージの平均スループットレートを測定したり、システムを監視して、接続スレッド使用率、メッセージストレージデータ、メッセージフローデータ、そのほかの関連するメトリックスを監視したりできます。その後、パフォーマンスに悪影響が出る上限まで、メッセージの生成レート、メッセージプロデューサの数、その他の変数を増加させることができます。実現可能な最大スループットが、メッセージサービス設定のベンチマークになります。

このベンチマークを基に、テスト群の特性の一部を変更できます。パフォーマンスに影響しそうな要因すべてを慎重に制御すれば (216 ページの「パフォーマンスに影響するアプリケーション設計の要因」を参照 )、これらの要因の変化によるベンチマークへの影響を理解できます。たとえば、接続数またはメッセージ数を 5 倍もしくは 10 倍に増やし、パフォーマンスに与える影響を調べることができます。

逆に、アプリケーションベースの要因を一定に保ち、たとえば、接続プロパティー、スレッドプールプロパティー、JVM メモリー制限、制限の動作、ファイルベースの持続と JDBC ベースの持続などを変更するといった、制御方法でブローカ設定を変更して、これらの変更がパフォーマンスに及ぼす影響を判断することもできます。

アプリケーションのこのようなベンチマークから、メッセージサービスを調整して配置済みのアプリケーションのパフォーマンスを向上させたいときに有用な情報を得られます。ベンチマークによって、1 か所の変更や一連の変更による影響を正確に予測できます。

原則として、ベンチマークは、管理されたテスト環境で、メッセージサービスを安定させるため長期間実施する必要があります。Java コードをマシンコードに変換する JIT コンパイルによる起動時には、パフォーマンスに悪影響が及びます。

# 基準になる使用パターン

メッセージングアプリケーションが配置され稼働されたあとは、基準になる使用パターンを確立することが重要となります。要求のピークがいつ発生するか把握し、その要求の定量化を図ります。たとえば、通常、要求はエンドユーザー数、アクティビティレベル、時間帯、またはこれらすべてによって左右されます。

基準になる使用パターンを確立するには、メッセージサーバーを一定期間監視して、次のようなデータを調べる必要があります。

- 接続数
- ブローカ、または特定の物理的送信先に保存されたメッセージ数
- ブローカ、または特定の物理的送信先で送受信されるメッセージフロー
- アクティブコンシューマの数

また、メトリックスデータにより提供される平均値とピーク値を使用することもできます。

これらの基準になるメトリックスを設計時の期待値と比較することが重要です。それによって、クライアントコードが正常に動作していることを確認します。たとえば、接続が開いたままになっている、または、消費されたメッセージが未通知のままになっているといった状態を確認できます。これらのコーディングエラーはメッセージサーバーのリソースを消費し、パフォーマンスに大きな影響を及ぼします。

基準になる使用パターンは、最適なパフォーマンスを得るためにシステムを調整する方法を決定する上で役立ちます。たとえば、次のように指定します。

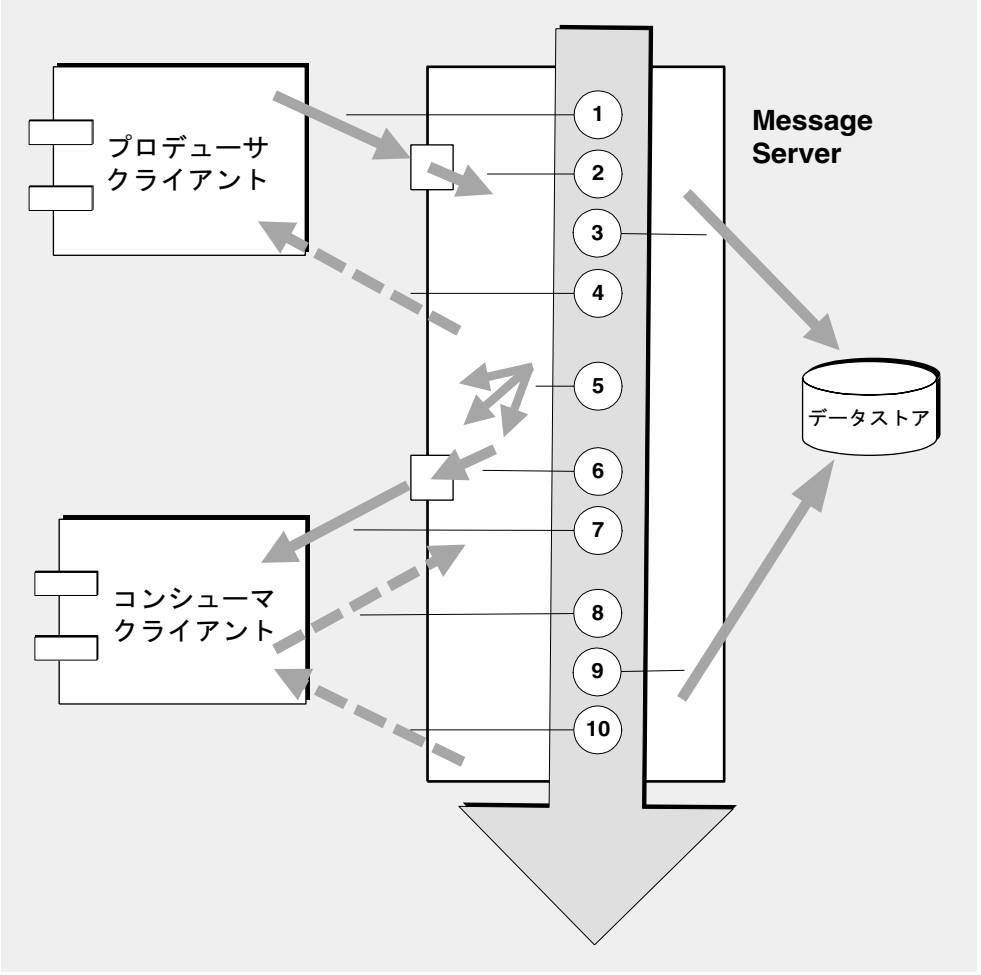
- 1 つの物理的送信先がそのほかの物理的送信先に比べ頻繁に使用されている場合は、その物理的送信先のメッセージメモリー制限をそのほかの送信先より高く設定したり、使用率に応じて制限の動作を調整したりできます。
- 必要な接続数が最大スレッドプールサイズによる許容値を大きく上回る場合は、スレッドプールサイズを増やすか、共有スレッドモデルを採用することができます。
- ピーク時のメッセージフローが平均フローより多い場合は、メモリーが不足したときに使用する制限の動作に影響することがあります。

一般に、使用パターンをより綿密に理解しているほど、より適切に、使用パターンに応じてシステムを調整し、将来ニーズに合わせてプランニングすることができます。

# パフォーマンスに影響する要因

メッセージの遅延とメッセージのスループットは、2 つの主要なパフォーマンスの評価基準です。これらは一般に、標準的なメッセージがメッセージ配信プロセスの各手順を完了するまでに要する時間に依存します。メッセージを持続的で信頼できる方法で配信する場合の各手順は次のとおりです。各手順を以下で図示します。

**図 11-1**　Message Queue サービスを使用したメッセージの配信




1. メッセージはプロデューシングクライアントからメッセージサーバーへ配信される。
2. メッセージサーバーはメッセージの内容を読み取る。

3. メッセージは、信頼性を維持するために持続ストレージに配置される。
4. メッセージサーバーは、信頼性を維持するためにメッセージの受信確認を発行する。
5. メッセージサーバーは、メッセージのルーティングを決定する。
6. メッセージサーバーはメッセージを書き込む。
7. メッセージはメッセージサーバーからコンシューミングクライアントへ配信される。
8. コンシューミングクライアントは、信頼性を維持するためにメッセージの受信確認を発行する。
9. メッセージサーバーは、信頼性を維持するために、クライアントの通知を処理する。
10. メッセージサーバーは、クライアントの通知が処理されたことを通知する。

これらの手順は順次実行されるため、プロデューシングクライアントからコンシューミングクライアントへメッセージを配信する際には、どの手順もボトルネックとなる恐れがあります。これらの手順の大半は、メッセージングシステムの物理的な特性に依存しています。物理的な特性には、ネットワーク帯域幅、コンピュータの処理速度、メッセージサーバーのアーキテクチャーなどが含まれます。ただし、一部の手順は、メッセージングアプリケーションの特性と必要とされる信頼性のレベルにも依存しています。

次の節では、アプリケーション設計の要因とメッセージングシステムの要因の両方がパフォーマンスに及ぼす影響について説明します。アプリケーション設計の要因とメッセージングシステムの要因はメッセージの配信に密接に関係しますが、各カテゴリは個別に考慮します。

# パフォーマンスに影響するアプリケーション設計の要因

アプリケーション設計の決定は、メッセージングのパフォーマンス全体に大きく影響することがあります。

パフォーマンスに影響するもっとも重要な要因は、メッセージ配信の信頼性に影響を及ぼす要因です。次のような要因が含まれています。

- 配信モード ( 持続的 / 持続性のないメッセージ )
- トランザクションの使用
- 通知モード
- 永続サブスクリプションと永続的でないサブスクリプション

そのほかに、パフォーマンスに影響するアプリケーション設計の要因には、次のものがあります。

- セレクタの使用 ( メッセージのフィルタリング )
- メッセージのサイズ
- メッセージ本体のタイプ

以降の節では、これらの各要因がメッセージングパフォーマンスに及ぼす影響について説明します。原則として、パフォーマンスと信頼性は相反しています。つまり、信頼性が高くなるとパフォーマンスは低下します。

表 11-1 は、さまざまなアプリケーション設計の要因が一般にどのようにメッセージングパフォーマンスに影響するかを示しています。表には、信頼性が高くパフォーマンスが低いシナリオと、パフォーマンスが高く信頼性の低いシナリオの 2 つのシナリオと、それぞれを特徴付ける主要なアプリケーション設計の要因を示します。これらの極端なシナリオの間には、信頼性とパフォーマンスの両方に影響する、多数の選択肢と兼ね合いがあります。

**表 11-1** 高信頼性シナリオと高パフォーマンスシナリオの比較

| アプリケーション設計の要因 | 高信頼性<br>低パフォーマンスシナリオ | 高パフォーマンス<br>低信頼性シナリオ |
|---|---|---|
| 配信モード | 持続メッセージ | 持続性のないメッセージ |
| トランザクションの使用 | 処理済みセッション | トランザクションなし |
| 通知モード | `AUTO_ACKNOWLEDGE` または `CLIENT_ACKNOWLEDGE` | `DUPS_OK_ACKNOWLEDGE` |
| 永続的 / 永続的でないサブスクリプション | 永続サブスクリプション | 永続的でないサブスクリプション |
| セレクタの使用 | メッセージのフィルタリング | メッセージのフィルタリングなし |
| メッセージのサイズ | 多数の小さいメッセージ | 少数の大きいメッセージ |
| メッセージ本体のタイプ | 複合本体タイプ | 単純本体タイプ |

**注**

以下のグラフのパフォーマンスデータは、2 基の CPU を搭載した 1002 MHz の Solaris 8 システムでファイルベースの持続を使用して生成されたものです。パフォーマンステストでは、JIT コンパイラにより、システムを最適化し、持続データベースの準備をするために、最初に Message Queue ブローカを起動しました。

ブローカを起動したあと、1 つのプロデューサと 1 つのコンシューマが作成され、メッセージが 30 秒間生成されました。コンシューマがすべての生成されたメッセージを受信するために要した時間が記録され、スループットレートつまり 1 秒あたりのメッセージ数が計算されました。このシナリオは、表 11-1 に示すアプリケーション設計の要因の異なる組み合わせで繰り返し実行されました。

## 配信モード ( 持続的 / 持続性のないメッセージ )

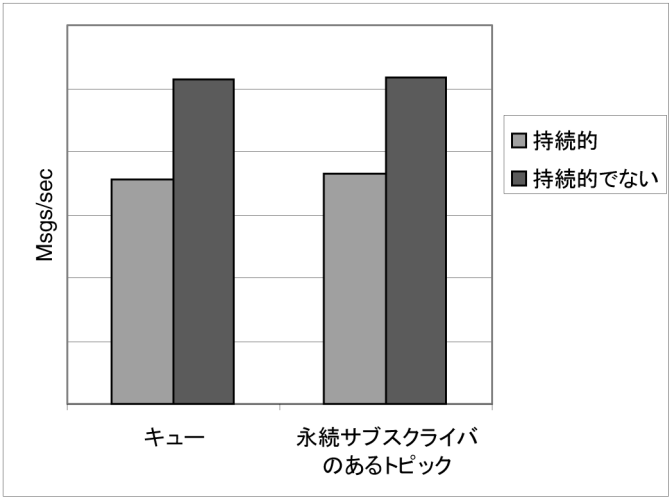
持続メッセージはメッセージサーバーの障害時にもメッセージの配信を保証します。すべての対象のコンシューマが、メッセージを消費したことを通知するまで、ブローカはメッセージを持続ストアに格納します。

持続メッセージのブローカの処理速度は、次の理由から、持続性のないメッセージの場合より低速です。

- ブローカに障害が生じても持続メッセージが失われないように、ブローカは信頼できる方法で持続メッセージを格納する必要があります。
- ブローカは受信した持続メッセージごとに、受信確認をする必要があります。メッセージを生成するメソッドが例外を返さなければ、ブローカへの配信は保証されます。
- クライアントの通知モードによっては、クライアントからの持続メッセージの受信通知が消費されたことを、ブローカが確認しなければならない場合があります。

持続モードと非持続モードではパフォーマンスに大きな差が生じることがあります。図 11-2 は、信頼できる配信を行う 2 つのケースで、持続メッセージと持続性のないメッセージのスループットを比較したものです。2 つのケースとは、永続サブスクリプションを使用して 10K バイトのメッセージをキューに配信した場合とトピックに配信した場合です。どちらの場合も `AUTO_ACKNOWLEDGE` 通知モードを使用しています。

図 11-2　配信モードによるパフォーマンスへの影響




## トランザクションの使用

トランザクションとは、処理済みセッションで生成されたすべてのメッセージと処理済みセッションで消費されたすべてのメッセージが、一体として処理されるか、または一体として処理されない、つまりロールバックされることを保証するものです。

Message Queue では、ローカルと分散の両方のトランザクションがサポートされます。

処理済みセッションでのメッセージの生成または通知の処理速度は、次の理由から、処理済みでないセッションの場合より低速です。

- 生成されたメッセージごとに追加情報を格納する必要がある。
- 状況によっては、通常は格納されないトランザクション内のメッセージを格納する場合がある。たとえば、サブスクリプションを使用しないトピック送信先へ配信された持続メッセージは、通常削除されますが、トランザクションの開始時にサブスクリプションに関する情報が使用できなくなってしまう。
- トランザクションでのメッセージの消費と通知に関する情報は、トランザクションがコミットされた時点で格納し処理する必要がある。

## 通知モード

JMS メッセージの配信の信頼性を保証する手段の 1 つは、Message Queue メッセージサーバーによってクライアントへ配信されたメッセージの消費をクライアントに通知するという方法です。

クライアントがメッセージを通知することなくセッションが閉じられた場合や、通知が処理される前にメッセージサーバーに障害が生じた場合には、ブローカはメッセージを再配信して JMSRedelivered フラグをセットします。

処理済みでないセッションの場合、クライアントは、それぞれ固有のパフォーマンス特性をもつ 3 つの通知モードの中から 1 つを選択できます。

- AUTO_ACKNOWLEDGE。コンシューマがメッセージを処理したあと、システムは自動的にメッセージを通知します。このモードでは、プロバイダで障害が生じたあとは、多くても 1 つのメッセージの再配信を保証するだけです。
- CLIENT_ACKNOWLEDGE。アプリケーションは、メッセージが通知されるポイントを制御します。直前の通知以降にそのセッションで処理されたすべてのメッセージが通知されます。一連の通知の処理中にメッセージサーバーに障害が生じた場合は、そのグループ内の複数のメッセージが再配信されることがあります。
- DUPS_OK_ACKNOWLEDGE。このモードは、時間をかけてメッセージを通知するようにシステムに指示します。プロバイダに障害が生じたあとでも、複数のメッセージを再配信できます。

CLIENT_ACKNOWLEDGE モードの使い方はトランザクションの使い方に似ています。ただし、処理中にプロバイダに障害が生じた場合に、すべての通知が一括して処理されることを保証していない点を除きます。

次の理由により、通知モードはパフォーマンスに影響します。

- AUTO_ACKNOWLEDGE モードと CLIENT_ACKNOWLEDGE モードでは、ブローカとクライアント間で特別な制御メッセージが必要です。追加の制御メッセージは、処理オーバーヘッドを高め、JMS ペイロードメッセージに干渉して処理遅延を引き起こすことがあります。
- AUTO_ACKNOWLEDGE モードと CLIENT_ACKNOWLEDGE モードでは、ブローカがクライアントの通知を処理したことを確認するまで、クライアントは待機する必要があります。その後、クライアントは追加メッセージを消費できるようになります。このブローカの確認によって、何らかの理由でブローカがこれらのメッセージを再配信しないように保証します。
- Message Queue 持続ストアは、コンシューマが受信したすべての持続メッセージに関する通知情報を使って更新する必要があります。そのため、パフォーマンスは低下します。

## 永続サブスクリプションと永続的でないサブスクリプション

トピック送信先へのサブスクライバは、永続サブスクリプションをもつものと、永続的でないサブスクリプションをもつものの 2 つのカテゴリに分かれます。

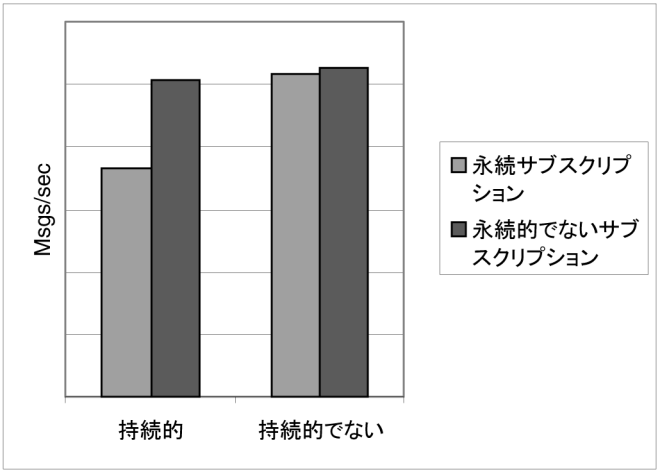
永続サブスクリプションでは、次の理由により、信頼性が高まりますが、スループットが遅くなります。

- Message Queue メッセージサーバーは、メッセージサーバーに障害が生じた場合でも回復後にリストを使用できるように、各永続サブスクリプションに割り当てられたメッセージのリストを持続的に格納する必要があります。
- メッセージサーバーに障害が生じた場合でも、回復後に、対応するコンシューマがアクティブになったときに、メッセージを引き続き配信できるように、永続サブスクリプションの持続メッセージは持続ストアに格納されます。対照的に、永続的でないサブスクリプションの持続メッセージは持続ストアには格納されません。したがって、メッセージサーバーに障害が生じると、対応するコンシューマ接続は失われ、メッセージは配信されません。

図 11-3 では、10K バイトの持続メッセージと持続性のないメッセージの 2 とおりのケースで、永続サブスクリプションと永続的でないサブスクリプションを使用したトピック送信先のスループットを比較しています。どちらの場合も AUTO_ACKNOWLEDGE 通知モードを使用しています。

図 11-3 から、パフォーマンスへの影響が顕著となるのは、永続サブスクリプションを使用した持続メッセージの場合だけであることがわかります。上記のとおり、この場合の影響は、永続サブスクリプションを使用したときに持続メッセージだけが持続ストアに格納されることに起因しています。

**図 11-3** サブスクリプションタイプによるパフォーマンスへの影響




## セレクタの使用 ( メッセージのフィルタリング )

アプリケーション開発者は、通常、特定のコンシューマへの一連のメッセージを対象にしています。それは、一意の物理的送信先への一連のメッセージごとを対象とするか、単一の物理的送信先を使用しコンシューマごとに複数のセレクタを登録することで実現できます。

セレクタは文字列であり、この文字列に一致するプロパティー値を持ったメッセージだけを特定のコンシューマに配信します。たとえば、セレクタ `NumberOfOrders >1` は、`NumberOfOrders` プロパティー値が 2 以上のメッセージだけを配信します。

コンシューマにセレクタを登録すると、各メッセージを取り扱うために追加処理が必要となり、複数の物理的送信先を使用する場合に比べ、パフォーマンスは低下します。以降のメッセージを比較する際にも構文解析できるセレクタを使用する必要があります。さらに、各メッセージがルーティングされるたびに、各メッセージのメッセージプロパティーを読み取り、比較する必要があります。ただし、セレクタを使用すると、メッセージングアプリケーションの柔軟性が向上します。

## メッセージのサイズ

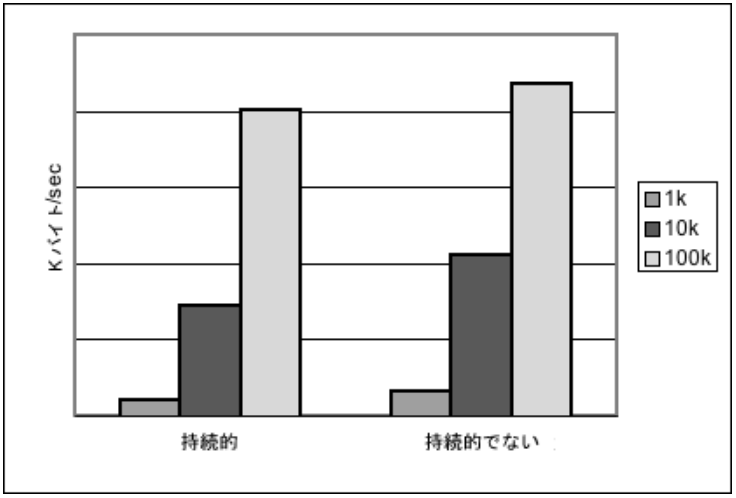
メッセージのサイズはパフォーマンスに影響します。プロデューシングクライアントからブローカへ、さらにブローカからコンシューミングクライアントへは、より多くのデータを渡す必要があり、持続メッセージの場合はサイズの大きいメッセージを格納する必要があるからです。

ただし、複数のサイズの小さいメッセージを 1 つのメッセージにまとめることで、個々のメッセージの転送と処理を最小限に抑え、パフォーマンス全体を向上させることができます。この場合、個々のメッセージの状態に関する情報は失われてしまいます。

図 11-4 は、持続メッセージと持続性のないメッセージの 2 とおりのケースで、1K、10K、および 100K バイトのメッセージのスループットを 1 秒あたりの K バイト数で比較したものです。どのケースも、メッセージはキュー送信先へ送信し、`AUTO_ACKNOWLEDGE` 通知モードを使用しています。

図 11-4 は、両方のケースで、小さいサイズのメッセージの場合に比べ、よりサイズの大きいメッセージを配信するほどオーバーヘッドが低くなることを示しています。また、1K バイトと 10K バイトのメッセージの場合は、持続メッセージと持続性のないメッセージを比べたパフォーマンスの上昇は約 50% ですが、100K バイトのメッセージの場合、この差が維持されないことがわかります。おそらくこれは、100K バイトの場合には、ネットワークの帯域幅がメッセージスループットのボトルネックになるからでしょう。

図 11-4　メッセージのサイズによるパフォーマンスへの影響




## メッセージ本体のタイプ

JMS がサポートするメッセージ本体のタイプ 5 種類の概要を複雑な順に次に示します。

- BytesMessage: 一連のバイトデータを含み、形式はアプリケーションによって決まります。
- TextMessage: 単純な java.lang.String です。
- StreamMessage: Java プリミティブ値によるストリームを含みます。
- MapMessage: 一連の名前と値のペアが含まれます。
- ObjectMessage: Java のシリアライズされたオブジェクトが含まれます。

一般に、メッセージのタイプはアプリケーションのニーズによって決定され、MapMessage や ObjectMessage などのより複雑なタイプほどパフォーマンスは低下します。データのシリアライズとデシリアライズがパフォーマンスを低下させます。パフォーマンスは、データがどの程度単純か、またはどの程度複雑かによって異なります。

# パフォーマンスに影響するメッセージサービスの要因

メッセージングアプリケーションのパフォーマンスは、アプリケーション設計だけでなく、メッセージのルーティングと配信を実行するメッセージサービスによっても影響を受けます。

次の節では、パフォーマンスに影響することのあるさまざまなメッセージサービスの要因について説明します。これらの要因の影響を理解しておくことは、メッセージサービスの内容を変更したり、配置済みのアプリケーションで発生することのあるパフォーマンスボトルネックを診断し解決したりする上で重要となります。

Message Queue サービスのパフォーマンスに影響するもっとも重要な要因は、次のとおりです。

- ハードウェア
- オペレーティングシステム
- Java 仮想マシン (JVM)
- 接続
- ブローカの制限と動作
- メッセージサーバーのアーキテクチャー
- データストアのパフォーマンス
- クライアントランタイムの設定

以降の節では、これらの各要因がメッセージングパフォーマンスに及ぼす影響について説明します。

## ハードウェア

Message Queue メッセージサーバーとクライアントアプリケーションのどちらの場合も、CPU の処理速度と使用可能なメモリーはメッセージサービスのパフォーマンスを決定する主要な要因となります。処理性能を強化して、多数のソフトウェア制限をなくす一方で、メモリーを追加して処理速度と能力を増加させることができます。ただし、一般に、単にハードウェアをアップグレードするだけでボトルネックを解消すると多額の費用がかかります。

## オペレーティングシステム

同じハードウェアプラットフォームを前提とした場合でも、異なるオペレーティングシステムの効率によって、パフォーマンスも変わってきます。たとえば、オペレーティングシステムが採用しているスレッドモデルが、メッセージサーバーがサポート可能な同時接続数に大きく影響することがあります。一般に、すべてのハードウェアが同じであれば、Solaris は通常 Linux より高速で、Linux は Windows より高速です。

## Java 仮想マシン (JVM)

メッセージサーバーは、ホスト JVM 内で実行され、ホスト JVM によってサポートされる Java プロセスです。そのため、JVM 処理は、メッセージサーバーがメッセージをいかに早く効率良くルーティングし配信できるかを決定する重要な要因となります。

特に、JVM のメモリーリソースの管理が不可欠となる場合があります。増加し続けるメモリーの負荷に対応するには、JVM に十分なメモリーを割り当てる必要があります。さらに、JVM は定期的に未使用のメモリーを再利用します。このメモリー再利用がメッセージの処理を遅らせることがあります。JVM のメモリーヒープが大きくなるほど、メモリー再利用時に経験することのある、潜在する遅延も長くなります。

## 接続

クライアントとブローカ間の接続の数と速度は、メッセージサーバーが処理可能なメッセージ数とメッセージ配信速度に影響することがあります。

### *メッセージサーバーの接続の制限*

メッセージサーバーへのアクセスはすべて、接続経由で行われます。同時接続数の制限によって、メッセージサーバーが同時に使用できるプロデューシングクライアントまたはコンシューミングクライアントの数が左右されることがあります。

メッセージサーバーへの接続の数は、一般に、使用可能なスレッド数によって制限されます。Message Queue は、専用スレッドモデルまたは共有スレッドモデルのどちらかをサポートするように設定できます (79 ページの「スレッドプール管理」を参照 )。

専用スレッドモデルは、各接続が専用のスレッドを持つため非常に高速ですが、接続の数は使用可能なスレッド数によって制限されます。この場合、接続ごとに、入力スレッドと出力スレッドが 1 つずつ必要です。共有スレッドモデルには、接続数の制限はありませんが、多数の接続でスレッドを共有するため、オーバーヘッドが増え、スループットが悪化します。これは、多くの接続が使用中のとき特に顕著になります。

### *トランスポートプロトコル*

Message Queue ソフトウェアを使うと、クライアントは各種の低レベルのトランスポートプロトコルを使用してメッセージサーバーと通信できます。Message Queue は、78 ページの「接続サービス」で説明する接続サービスとそれに対応するプロトコルをサポートします。

暗号化、ファイアウォールを介したアクセスなどのプロトコルは、アプリケーション要件に基づいて選択されますが、選択結果はパフォーマンス全体に影響を及ぼします。

**図 11-5**　トランスポートプロトコルの速度

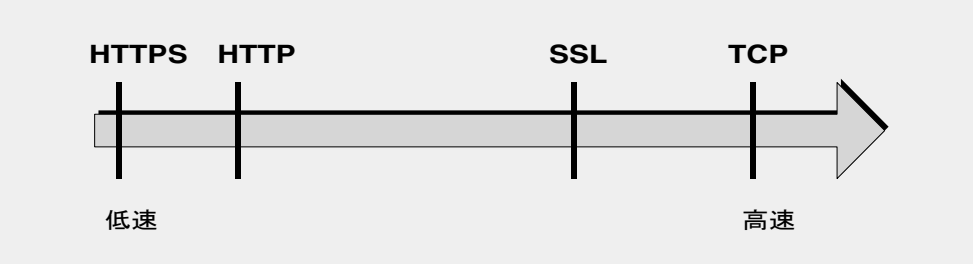


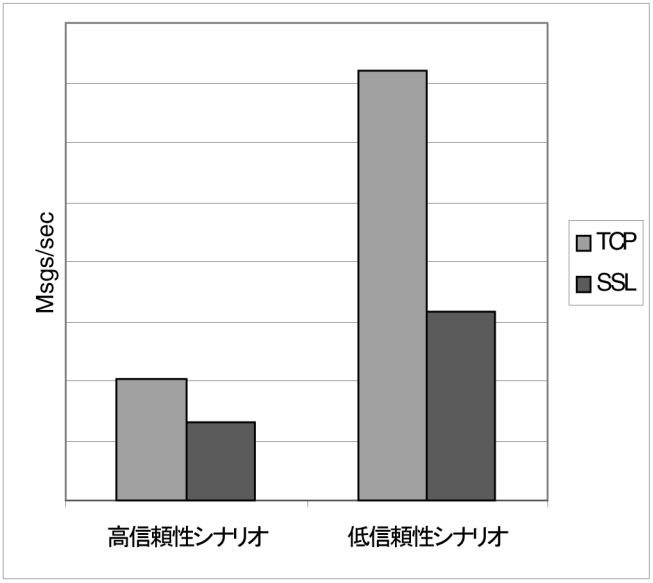

図 11-5 は、さまざまなプロトコルテクノロジのパフォーマンス特性を示しています。

- TCP は、ブローカと通信する最速の方法を提供します。
- SSL は、メッセージの送信と受信に関しては、TCP より 50 ～ 70% 低速です。持続メッセージの場合は 50%、持続性のないメッセージの場合は約 70% です。さらに、初期接続の確立は、SSL を使用した場合の方が低速で数秒かかります。これは、クライアントとブローカ、または HTTPS の場合は Web Server で、送信するデータの暗号化に使う非公開キーの作成が必要なためです。低レベルの各 TCP パケットの暗号化と復号化に必要な追加処理によって、パフォーマンスの低下が引き起こされます。

  図 11-6 は、2 つのケースでの TCP と SSL のスループットを比較したものです。2 つのケースとは、1K バイトの持続メッセージを、永続サブスクリプションと `AUTO_ACKNOWLEDGE` 通知モードを使用しているトピック送信先へ送信する高信頼性シナリオと、1K バイトの持続性のないメッセージを、永続サブスクリプションと `DUPS_OK_ACKNOWLEDGE` 通知モードを使用しているトピック送信先へ送信するハイパフォーマンスシナリオです。

  図 11-6 は、高信頼性ケースの方がプロトコルによる影響は少ないことを示します。これは、高信頼性のケースで必要な持続メッセージのためのオーバーヘッドの方が、プロトコルの速度より、スループットを制限する重要な要因となるからです。

図 11-6　トランスポートプロトコルによるパフォーマンスへの影響




- HTTP は、TCP または SSL より低速です。HTTP ではクライアントとブローカ間のプロキシとして、Web サーバー上で実行しているサーブレットを使用します。パケットを HTTP 要求へカプセル化する必要がある点と、メッセージがブローカへ到達するには、クライアントからサーブレットへ、サーブレットからブローカへという 2 つの段階が必要である点から、パフォーマンスへのオーバーヘッドが発生します。
- HTTPS は HTTP より低速です。これは、クライアントとサーブレット間、およびサーブレットとブローカ間でパケットを暗号化するためにオーバーヘッドが必須となるからです。

## メッセージサーバーのアーキテクチャー

メッセージサーバーは、シングルブローカ、または複数の連結されたブローカインスタンスであるブローカクラスタとして実装できます。

ブローカに接続するクライアントの数や配信されるメッセージの数が増えると、ブローカは最終的に、ファイル記述子、スレッド、メモリーの制限などのリソースの限界を超えてしまいます。増え続ける負荷に対処するための 1 つの方法は、Message Queue メッセージサーバーにブローカインスタンスを追加して、クライアントの接続とメッセージのルーティングおよび配信を複数のブローカに分散することです。

一般に、ブローカインスタンスの追加は、クライアント、特にメッセージプロデューシングクライアントがクラスタ間で均等に分散されている場合に最適に動作します。クラスタ内のブローカ間でのメッセージ配信にはオーバーヘッドが伴うため、接続数とメッセージ配信レートが制限されているクラスタでは、シングルブローカよりパフォーマンスが低くなります。

また、ブローカクラスタを使用してネットワークの帯域幅を最適化することもできます。たとえば、クラスタ内の一連のリモートブローカ間で、速度の遅い、長距離のネットワークリンクを使用する一方で、個々のブローカインスタンスへのクライアントの接続に、高速なリンクを使用することができます。

クラスタの詳細については、第 9 章「ブローカクラスタを使用した作業」を参照してください。

## ブローカの制限と動作

メッセージを処理するためにメッセージサーバーに要求されるメッセージスループットは、メッセージサーバーがサポートするメッセージングアプリケーションの使用パターンによって異なります。ただし、メッセージサーバーでは、メモリーや CPU サイクルなどのリソースに制限があります。リソースの制限により、メッセージサーバーは、過負荷となり、無応答または不安定となるポイントに達してしまうことがあります。

Message Queue メッセージサーバーには、メモリーリソースを管理し、ブローカのメモリー不足を防ぐためのメカニズムが組み込まれています。これらのメカニズムに含まれるのは、ブローカまたは個々の物理的送信先が保持できるメッセージ数またはメッセージのバイト数についての設定可能な制限と、物理的送信先の制限に達したときに起動できる一連の動作です。

これらの設定可能なメカニズムを使用して、システムが過負荷にならないように、メッセージの受信と送信のバランスを取ることができますが、これには注意深い監視と調整が必要です。これらのメカニズムは、オーバーヘッドを増加させ、メッセージのスループットを制限することがありますが、それでも動作の完全性を維持します。

## データストアのパフォーマンス

Message Queue は、ファイルベースの持続モジュールと JDBC ベースの持続モジュールの両方をサポートしています。ファイルベースの持続モジュールでは、持続データを格納するために個別のファイルを使用します。JDBC ベースの持続では、JDBC™ (Java Database Connectivity) インタフェースを使用し、JDBC 互換のデータストアを必要とします。一般的に、ファイルベースの持続は、JDBC ベースの持続よりも高速です。ただし、JDBC 互換のストアによって提供される冗長性や管理機能を好むユーザーもいます。

ファイルベースの持続の場合は、持続的な操作によりメモリー内の状態とデータストアとを同期化するように指定することで、信頼性を高められます。この同期化は、システム破壊によるデータの損失をなくす上で役立ちますが、パフォーマンスが犠牲になります。

### クライアントランタイムの設定

Message Queue クライアントランタイムは、クライアントアプリケーションに Message Queue メッセージサービスへのインタフェースを提供します。クライアントランタイムでは、物理的送信先にメッセージを送信し、物理的送信先からメッセージを受信する場合に、クライアントに必要なすべての処理をサポートします。クライアントランタイムは、接続ファクトリ属性値を使って設定可能で、パフォーマンスとメッセージスループットを向上させるように、接続フロー測定、コンシューマフローの制限、接続フロー制御などのプロパティーと動作を設定できます。これらの機能とそれを設定するために使用される属性の詳細は、236 ページの「クライアントランタイムのメッセージフローの調整」を参照してください。

# パフォーマンスを改善するための設定の調整

## システムの調整

次の節では、オペレーティングシステム、JVM、および通信プロトコルで実行できる調整について説明します。

### Solaris での調整 : CPU 使用率、ページング / スワッピング / ディスク I/O

オペレーティングシステムの調整については、システムのマニュアルを参照してください。

### Java 仮想マシン (JVM) の調整

デフォルトでは、ブローカは 192M バイトの JVM ヒープサイズを使用します。通常、大量のメッセージ負荷がある場合はこのサイズでは小さ過ぎるため、大きくする必要があります。

Java オブジェクトが使用する JVM のヒープ容量を使い果たしそうになると、ブローカは、フロー制御やメッセージスワップなどのさまざまな技術を使用して、メモリーを解放します。極端な状況のもとでは、メモリーを解放し、メッセージの流入を減少させるために、クライアント接続を閉じることもあります。このような状況を避けるため、最大 JVM ヒープ容量を十分に高く設定するようお勧めします。

ただし、Java の最大ヒープ容量がシステムの物理メモリーに対して高くしすぎた場合、ブローカは Java ヒープ容量を増加し続け、システム全体のメモリーを使い果たしてしまうことがあります。これは、パフォーマンスの低下、予期しないブローカのクラッシュにつながり、そのシステムで実行されているほかのアプリケーションやサービスの動作にも影響を与える場合があります。一般に、オペレーティングシステムとそのほかのアプリケーションをマシン上で実行させるために十分な物理メモリーを使用させる必要があります。

一般に、通常時とピーク時のシステムメモリーフットプリントを評価して、十分なパフォーマンスを得られて、しかもシステムメモリーに問題を生じさせるほどではない大きさに Java ヒープサイズを設定するのがよい方法です。

ブローカの最小ヒープサイズと最大ヒープサイズを変更するには、ブローカの起動時に `-vmargs` コマンド行オプションを使用します。たとえば、次のように指定します。

```
/usr/bin/imqbrokerd -vmargs "-Xms256m -Xmx1024m"
```

このコマンドは、起動時の Java ヒープサイズを 256M バイトに、最大 Java ヒープサイズを 1G バイトに設定します。

- Solaris や Linux では、`/etc/rc*`、つまり `/etc/init.d/imq` を介してブローカを起動する場合には、`/etc/imq/imqbrokerd.conf` (Solaris) ファイルまたは `/etc/opt/sun/mq/imqbrokerd.conf` (Linux) ファイルにブローカコマンド行引数を指定します。詳細については、そのファイルのコメントを参照してください。
- Windows では、ブローカを Windows のサービスとして起動する場合には、`imqsvcadmin install` コマンドに `-vmargs` オプションを使用して JVM 引数を指定します。第 13 章「コマンド行のリファレンス」の「サービス管理ユーティリティー」を参照してください。

どのような場合でも、ブローカのログファイルを確認するか、`imqcmd metrics bkr -m cxn` コマンドを使用して設定を検証します。

## トランスポートプロトコルの調整

アプリケーションのニーズを満たすプロトコルが選択されたら、選択されたプロトコルに基づいて調整を加えることでパフォーマンスを改善できます。

プロトコルのパフォーマンスは、次の 3 つのブローカプロパティーを使用して修正できます。

- `imq.protocol.`*protocolType*`.nodelay`
- `imq.protocol.`*protocolType*`.inbufsz`
- `imq.protocol.`*protocolType*`.outbufsz`

TCP と SSL プロトコルの場合、これらのプロパティーがクライアントとブローカ間のメッセージ配信の速度に影響します。HTTP プロトコルと HTTPS プロトコルの場合は、これらのプロパティーが、Web サーバー上で実行している Message Queue トンネルサーブレットとブローカ間のメッセージ配信の速度に影響します。HTTP/HTTPS プロトコルの場合、そのほかにもパフォーマンスに影響することのあるプロパティーがあります (233 ページの「HTTP/HTTPS の調整」を参照)。

プロトコルを調整するためのプロパティーについては、次の節で説明します。

### *nodelay*

`nodelay` プロパティーは、特定のプロトコルの Nagle のアルゴリズム、つまり TCP/IP 上の `TCP_NODELAY` ソケットレベルのオプションの値に影響します。Nagle のアルゴリズムは、広域ネットワーク (WAN) などの低速接続を使用しているシステム上で TCP パフォーマンスを改善するために使用されます。

このアルゴリズムが使用されている場合、TCP は、データをサイズの大きいパケットにバンドルすることで、複数の小さいデータの塊がリモートシステムへ送信されるのを防ぎます。ソケットに書き込まれたデータが必要なバッファーサイズを満たしていない場合、プロトコルは、バッファーが満たされるか、一定の遅延時間が経過するまで、パケットの送信を遅らせます。バッファーがいっぱいになるか、タイムアウトが発生すると、パケットが送信されます。

大半のメッセージングアプリケーションでは、パケットの送信に遅延がない、つまり Nagle のアルゴリズムが無効な場合にパフォーマンスは最適となります。これは、クライアントとブローカ間の大半の対話が、要求 / 応答型の対話だからです。つまり、クライアントはデータのパケットをブローカへ送信し、その応答を待ちます。たとえば、典型的な対話には次のものがあります。

- 接続の作成
- プロデューサまたはコンシューマの作成
- 持続メッセージの送信。ブローカはメッセージの受信を確認します
- `AUTO_ACKNOWLEDGE` セッションまたは `CLIENT_ACKNOWLEDGE` セッションでのクライアント通知の送信。ブローカは通知の処理を確認します

これらの対話では、大半のパケットがバッファーサイズより小さいサイズです。つまり、Nagle のアルゴリズムが使用されている場合は、ブローカは数ミリ秒遅れて、コンシューマに応答を送信します。

ただし、Nalge のアルゴリズムは、接続が低速でブローカの応答が必要ない状況で、パフォーマンスを改善することができます。これは、クライアントが持続性のないメッセージを送信する場合や、クライアント通知がブローカによって確認されない場合 (`DUPS_OK_ACKNOWLEDGE` セッション ) です。

### *inbufsz/outbufsz*

inbufsz プロパティーは、ソケットからのデータを読み取る入力ストリームのバッファーサイズを設定します。同様に、outbufsz は、ブローカがデータをソケットに書き込むために使用する出力ストリームのバッファーサイズを設定します。

一般に、どちらのパラメータも受信パケットまたは送信パケットの平均サイズより多少大きい値に設定する必要があります。経験上、これらのプロパティーはパケットの平均サイズに 1K バイトを足した値 (K バイト単位で四捨五入 ) に設定すると良いでしょう。

たとえば、本体が 1K バイトのパケットをブローカで受信している場合、パケット全体のサイズ ( メッセージ本体 + ヘッダー + プロパティー ) は約 1200 バイトです。inbufsz を 2K バイト (2048 バイト ) にすると、妥当なパフォーマンスが得られます。

inbufsz または outbufsz をそのサイズより大きくすると多少パフォーマンスは改善しますが、接続ごとに必要なメモリーも増えます。

図 11-7 は、1K バイトのパケットの inbufsz を変更した結果を示しています。

**図 11-7　1K バイト (1024 バイト ) パケットの inbufsz を変更した結果**

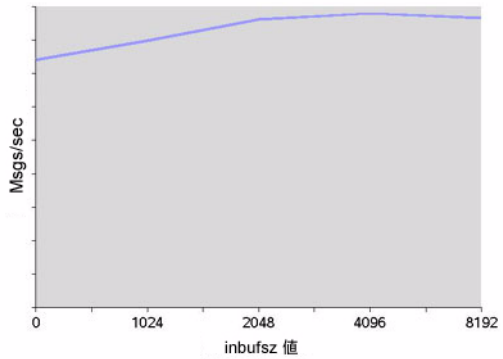
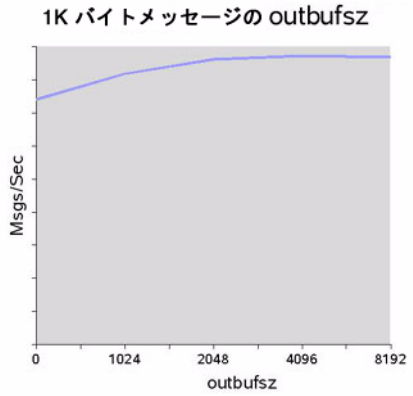



図 11-8 は、1K バイトのパケットの outbufsz を変更した結果を示しています。

図 11-8　1K バイト (1024 バイト ) パケットの outbufsz を変更した結果




### *HTTP/HTTPS の調整*

前の 2 つの節で説明した一般的なプロパティーに加え、HTTP/HTTPS のパフォーマンスは、Message Queue トンネルサーブレットをホスティングする Web サーバーへの HTTP 要求をクライアントが作成する速度によっても制限されます。

Web サーバーは、シングルソケットで複数の要求を処理するように最適化する必要があります。JDK バージョン 1.4 以降では、Web サーバーが複数の HTTP 要求を処理する際に使用するリソースを最小限にするために、Web サーバーへの HTTP 接続 (Web サーバーへのソケット ) は開かれたままになっています。JDK 1.4 を使用しているクライアントアプリケーションのパフォーマンスが JDK の旧リリースで稼働している同じアプリケーションより低速な場合は、パフォーマンスを改善するために Web サーバーのキープアライブ設定パラメータの調整が必要となることがあります。

このような Web サーバーの調整に加え、クライアントが Web サーバーをポーリングする頻度を調整することもできます。HTTP は要求ベースのプロトコルです。つまり、HTTP ベースのプロトコルを使用しているクライアントは、メッセージが待機中かどうかを判断するために Web サーバーを定期的に確認する必要があります。`imq.httpjms.http.pullPeriod` ブローカプロパティーとそれに対応する `imq.httpsjms.https.pullPeriod` プロパティーは、Message Queue クライアントが Web サーバーをポーリングする頻度を指定します。

`pullPeriod` 値が `-1` ( デフォルト値 ) の場合、クライアントランタイムは直前の要求が戻るとすぐにサーバーをポーリングし、個々のクライアントのパフォーマンスを最大化します。その結果、各クライアント接続が Web サーバー内の要求スレッドを 1 つずつ占有するため、Web サーバーのリソースにかなりの負荷がかかる場合があります。

pullPeriod 値が正の数字である場合、クライアントランタイムは要求を定期的に Web サーバーへ送信し、データが保留されているかどうかを確認します。この場合、クライアントは Web サーバーの要求スレッドを占有しません。したがって、多数のクライアントが Web サーバーを使用している場合は、pullPeriod を正の値に設定することで Web サーバーのリソースを節約できます。

### ファイルベースの持続ストアの調整

ファイルベースの持続ストアの調整については、82 ページの「持続サービス」を参照してください。

## ブローカの調整

次の節では、パフォーマンスを改善するためにブローカのプロパティーに対して実行できる調整について説明します。

### メモリー管理 : 負荷のある状態でブローカの安定性を高める

メモリー管理は、送信先単位で、またはシステム全体に対してすべての送信先を一括で、設定できます。

#### *物理的送信先の制限の使い方*

物理的送信先の制限については、第 6 章「物理的送信先の管理」を参照してください。

#### *システム全体の制限の使い方*

メッセージプロデューサの処理速度がメッセージコンシューマの処理速度を上回る傾向がある場合には、メッセージをブローカに蓄積できます。ブローカにはメモリーが不足した場合に、プロデューサの処理速度を低下させ、アクティブメモリーからメッセージをスワップさせるメカニズムが組み込まれていますが、ブローカが保持可能なメッセージの合計数とメッセージのバイトの合計数に厳密な制限を設定した方が賢明です。

imq.system.max_count ブローカプロパティーと imq.system.max_size ブローカプロパティーを設定して、これらの制限を制御します。

たとえば、次のように指定します。

```
imq.system.max_count=5000
```

上記で定義された値は、ブローカが未配信 / 未通知のメッセージを最大 5000 までしか保持しないことを示しています。それ以上のメッセージが送信されると、メッセージはブローカによって拒否されます。メッセージが持続的な場合は、プロデューサがメッセージを送信しようとすると例外を受け取ります。メッセージが持続的でない場合は、ブローカは暗黙のうちにメッセージを廃棄します。

メッセージの送信時に例外が戻った場合、クライアントは一時停止してから、送信を再試行します。この例外は、メッセージ受信に関するブローカの障害が原因ではありません。発生した例外は、送信側のクライアントだけが検出します。

## 複数のコンシューマキューのパフォーマンス

複数のキューコンシューマがキュー送信先でメッセージを処理する能率は、次の設定可能キュー送信先属性によって決まります。

- アクティブなコンシューマの数 (`maxNumActiveConsumers`)
- 1 つのバッチでコンシューマに配信できるメッセージの最大数 (`consumerFlowLimit`)

最適なメッセージスループットを実現するには、十分な数のアクティブコンシューマがキューでのメッセージの生成に遅れずに対応し、消費する割合を最大にするような方法で、キュー内のメッセージをルーティングし、アクティブコンシューマへ配信しなければなりません。メッセージ配信を複数のコンシューマに分散させる一般的なメカニズムについては、『Sun Java System Message Queue 技術の概要』で説明されています。

メッセージがキューに蓄積している場合、メッセージ負荷を処理するアクティブコンシューマの数が不十分であることが考えられます。また、複数のメッセージがバッチサイズでコンシューマに配信されるため、メッセージがコンシューマ上でバックアップされていることも考えられます。たとえば、バッチサイズ (`consumerFlowLimit`) が大き過ぎる場合は、あるコンシューマがキュー内のすべてのメッセージを受信し、そのほかのコンシューマは何も受信していないことがあります。コンシューマが非常に高速であれば、これは問題にはなりません。

ただし、コンシューマが比較的低速で、メッセージをコンシューマに均等に分散させたい場合は、バッチサイズを小さくする必要があります。バッチサイズが小さいほど、メッセージをコンシューマへ配信するのに必要なオーバーヘッドは増加します。それでも、低速なコンシューマの場合は、一般に、小さいバッチサイズを使用した方がパフォーマンスは向上します。

# クライアントランタイムのメッセージフローの調整

この節では、パフォーマンスに影響するフロー制御の動作について説明します (229 ページの「クライアントランタイムの設定」を参照 )。これらの動作は、接続ファクトリの管理対象オブジェクトの属性として設定されます。接続ファクトリ属性の設定方法については、第 8 章「管理対象オブジェクトの管理」を参照してください。

## メッセージフロー測定

クライアントによって送受信されるメッセージ ( ペイロードメッセージ ) と Message Queue 制御メッセージは、同じクライアントとブローカ間の接続を使って伝送されます。ブローカ通知などの制御メッセージの配信における遅延は、ペイロードメッセージの配信によって制御メッセージが保留された場合の結果として生じます。このようなネットワークの輻輳を防止するため、Message Queue は接続全体のペイロードメッセージのフローを測定します。

ペイロードメッセージは、接続ファクトリ属性 `imqConnectionFlowCount` の指定に従い、設定した数のみが配信されるようにバッチされます。バッチが配信されたあと、ペイロードメッセージの配信は中断され、保留中の制御メッセージのみが配信されます。ペイロードメッセージの追加のバッチが配信されるときも、このサイクルが繰り返され、保留中の制御メッセージが続きます。

クライアントが、ブローカからの多数の応答を必要とする操作を実行している場合、たとえば、クライアントが `CLIENT_ACKNOWLEDGE` または `AUTO_ACKNOWLEDGE` モード、持続メッセージ、トランザクション、キューブラウザを使用している場合や、クライアントがコンシューマを追加または削除している場合などには、`imqConnectionFlowCount` の値を小さいままにしておく必要があります。一方、クライアントが `DUPS_OK_ACKNOWLEDGE` モードを使用しており、接続上に単純なコンシューマしかいない場合は、パフォーマンスを犠牲にすることなく `imqConnectionFlowCount` の値を増やすことができます。

## メッセージフロー制限

Message Queue クライアントランタイムがメモリーなどのローカルリソースの上限に達する前に処理可能なペイロードメッセージの数には制限があります。この数に達すると、パフォーマンスに悪影響が出ます。したがって、Message Queue では、コンシューマあたりのメッセージ数または接続あたりのメッセージ数を制限できます。この制限は、接続を介して配信し、クライアントランタイムにバッファリングし、消費を待機できるメッセージの数を示しています。

### *コンシューマフローの制限*

クライアントランタイムへ配信されたペイロードメッセージの数が、いずれかのコンシューマの imqConsumerFlowLimit 値を超えた場合、そのコンシューマへのメッセージ配信は停止します。そのコンシューマの消費されないメッセージの数が、imqConsumerFlowThreshold で設定された値を下回った場合にだけ、配信処理が再開されます。

次の例は、これらの制限の使い方を示しています。トピックコンシューマのデフォルト設定値を前提としています。

```
imqConsumerFlowLimit=1000

imqConsumerFlowThreshold=50
```

コンシューマが作成され、1000 のメッセージがあれば、ブローカはこれらのメッセージを最初のバッチとしてコンシューマに配信します。このとき一時停止はありません。1000 メッセージの送信後、ブローカはクライアントランタイムが追加のメッセージを要求するまで、配信を停止します。アプリケーションがこれらのメッセージを処理するまで、クライアントランタイムはそれらを保持します。その後、クライアントランタイムは、アプリケーションがメッセージバッファー容量の 50% (imqConsumerFlowThreshold) つまり 500 メッセージ以上を消費するのを待ってから、ブローカに次のバッチを送信するように要求します。

同じ状況で、しきい値が 10% の場合、クライアントランタイムは、アプリケーションが少なくとも 900 メッセージを消費してから、次のバッチを要求します。

次のバッチサイズの計算方法 :

```
imqConsumerFlowLimit - ( 現在、バッファーに保留中のメッセージ数 )
```

そのため、imqConsumerFlowThreshold が 50% の場合、次のバッチサイズは、アプリケーションがメッセージを処理する速度に応じて 500 ～ 1000 の間になります。

imqConsumerFlowThreshold がかなり高く (100% 近くに ) 設定された場合、ブローカは比較的小さいサイズのバッチを送信するため、メッセージのスループットは低下することがあります。値が低過ぎる (0% に近い ) 場合は、クライアントは次のセットを配信する前に残りのバッファリングされたメッセージの処理を完了してしまい、やはりメッセージスループットを低下させることがあります。一般に、特定のパフォーマンスや信頼性を考慮しないかぎり、imqConsumerFlowThreshold 属性のデフォルト値を変更する必要はありません。

コンシューマベースのフロー制御、特に、imqConsumerFlowLimit は、クライアントランタイム内のメモリーを管理する最適な手段です。一般に、クライアントアプリケーションに応じて、接続でサポートする必要のあるコンシューマの数、メッセージのサイズ、クライアントランタイムで使用可能なメモリー総量がわかります。

### *接続フローの制限*

一部のクライアントアプリケーションでは、エンドユーザーの選択によって、コンシューマの数が不確定な場合があります。そのような場合は、引き続き、接続レベルのフロー制限を使用してメモリーを管理できます。

接続レベルのフロー制御は、接続上のすべてのコンシューマについてバッファリングされたメッセージの合計数を制限します。この数が `imqConnectionFlowLimit` の値を超えると、合計数が接続の制限を下回るまで、接続経由のメッセージの配信は停止します。`imqConnectionFlowLimit` 属性は、`imqConnectionFlowLimitEnabled` を `true` に設定した場合にだけ使用できます。

1 つのセッションでキューに入るメッセージの数は、そのセッションを使用するメッセージコンシューマの数と、各コンシューマのメッセージ負荷によって決まります。クライアント側のメッセージの生成またはメッセージの消費に遅延が発生する場合は、通常は、アプリケーションを再設計し、より多くのセッションにメッセージプロデューサとメッセージコンシューマを分散し、またはより多くの接続にセッションを分散してパフォーマンスを改善できます。

第 12 章

# 問題のトラブルシューティング

この章では、次の問題を把握して解決する方法について説明します。



問題が発生したら、インストールしている Message Queue ソフトウェアのバージョン番号を調べてください。そのバージョン番号により、ソフトウェアバージョンと一致するバージョンのマニュアルを使用していることを確認します。Sun に問題を報告するときにも、そのバージョン番号が必要になります。バージョン番号を調べるには、次のコマンドを実行します。

```
imqcmd -v
```

# クライアントが接続を確立できない

この問題では、次の症状がみられます。

- クライアントが新しい接続を確立できない。
- クライアントが障害の生じた接続を自動的に再接続できない。

この節では、次の原因について説明します。



## *クライアントアプリケーションが接続を閉じていないため、接続数がリソース制限を超えてしまった*

**この問題の原因を確認するには**

ブローカへの接続をすべて一覧表示します。

```
imqcmd list cxn
```

出力にはすべての接続と各接続の確立元のホストが一覧表示されます。異常な数の接続が開かれている特定のクライアントがわかります。

**問題を解決するには**

原因となっているクライアントが未使用の接続を閉じるようにプログラムし直します。

## *ブローカが実行されていないか、ネットワーク接続の問題が存在している*

**この問題の原因を確認するには**

- ブローカのプライマリポートへ telnet で接続し、ブローカがポートマッパー出力を返すか確認します。プライマリポートのデフォルトは 7676 です。
- ブローカプロセスがホスト上で実行されていることを確認します。

**問題を解決するには**

- ブローカを起動します。

- ネットワーク接続の問題を修復します。

### *接続サービスが非アクティブであるか停止している*

**この問題の原因を確認するには**

すべての接続サービスのステータスを確認します。

```
imqcmd list svc
```

接続サービスのステータスが unknown または paused と表示された場合、クライアントはそのサービスを使用する接続を確立できません。

**問題を解決するには**

- 接続サービスのステータスが unknown と表示された場合、そのサービスはアクティブサービスリスト (imq.service.active) に含まれていません。SSL ベースのサービスの場合は、サービスが不適切に設定されているため、ブローカがブローカログに次のエントリを作成する可能性があります。`ERROR [B3009]: Unable to start service ssljms: [B4001]: Unable to open protocol tls for ssljms service...` 例外の根本的な原因の説明が続きます。

  SSL サービスを適切に設定する方法については、151 ページの「SSL ベースのサービスの操作」を参照してください。

- 接続サービスのステータスが paused と表示された場合は、サービスを再開します (112 ページの「接続サービスの停止および再開」を参照 )。

### *必要な接続数に対して使用可能なスレッドが少なすぎる*

**この問題の原因を確認するには**

ブローカログの次のエントリを確認します。

```
WARNING [B3004]: No threads are available to process a new connection
on service ... Closing the new connection.
```

また、次の形式のうちいずれかを使用し、接続サービスの接続数と現在使用中のスレッド数を確認します。

```
imqcmd query svc -n serviceName
```

```
imqcmd metrics svc -n serviceName -m cxn
```

接続ごとに 2 つのスレッドが必要です。1 つは受信メッセージ用、もう 1 つは出力メッセージ用です (79 ページの「スレッドプール管理」を参照 )。

### 問題を解決するには

- 専用スレッドプールモデル (`imq.`*serviceName*`. threadpool_model=dedicated`) を使用している場合、接続の最大数は、スレッドプールにあるスレッドの最大数の半分です。そのため、接続数を増やすには、スレッドプール (`imq.`*serviceName*`.max_threads`) のサイズを拡大するか、共有スレッドプールモデルに切り換えます。
- 共有スレッドプールモデル (`imq.`*serviceName*`. threadpool_model=shared`) を使用している場合、接続の最大数は、接続監視制限 (`imq.`*serviceName*`.connectionMonitor_limit`) とスレッドの最大数 (`imq.`*serviceName*`.max_threads`) の 2 つのプロパティーの結果の半分です。そのため、接続数を増やすには、スレッドプールのサイズを拡大するか、接続監視制限の値を大きくします。
- 最終的に、サポート可能な接続の数または接続のスループットが入力 / 出力制限に達してしまいます。このような場合は、マルチブローカクラスタを使用して、クラスタ内のブローカインスタンス間で接続を分散します。

## *Solaris または Linux オペレーティングシステム上で必要な接続数に対してファイル記述子が少なすぎる*

この問題については、68 ページの「ファイル記述子制限の設定」を参照してください。

### この問題の原因を確認するには

次のようなブローカログのエントリを確認します。`Too many open files`

### 問題を解決するには

`ulimit` のマニュアルで説明しているとおり、ファイル記述子の制限を増やします。

## *TCP バックログにより、確立可能な新しい同時接続要求の数が制限される*

TCP バックログにより、ポートマッパーが追加の要求を拒否するまでにシステムバックログ (`imq.portmapper.backlog`) に格納可能な同時接続要求の数が制限されます。Windows オペレーティングシステムでは、Windows デスクトップで 5、Windows サーバーで 200 というバックログ制限がハードコードされています。

通常、バックログ制限が原因の要求拒否は過渡的な現象であり、非常に多数の同時接続要求があると発生します。

### この問題の原因を確認するには

ブローカログを調べます。最初に、ブローカが、その他の接続を拒否している期間に接続を受け入れているかどうかを確認します。次に、拒否された接続について説明するメッセージを確認します。このようなメッセージがある場合、TCP バックログが問題ではないと思われます。ブローカは、TCP バックログによる接続拒否をログしないからです。

正常接続がログされ、接続拒否がログされない場合は、TCP バックログが問題と思われます。

**問題を解決するには**

次の手順で、TCP バックログ制限を解消できます。

- クライアントが確立しようとする接続の再試行を短い間隔で行うようにプログラミングします。この問題の過渡的な性質上、このようにプログラミングしても正常に動作します。
- `imq.portmapper.backlog` の値を大きくします。
- クライアントが接続を閉じずに、多くの接続を開いていないか確認します。

### *オペレーティングシステムによって同時接続の数が制限される*

Windows オペレーティングシステムのライセンスは、サポートされる同時リモート接続の数を制限します。

**この問題の原因を確認するには**

`imqcmd query svc` を使用して、接続用のスレッドが十分にあることを調べ、さらに Windows ライセンス契約書の条項を確認します。ローカルクライアントからは接続を確立できるが、リモートクライアントからは確立できない場合は、オペレーティングシステムの制限が問題の原因と考えられます。

**問題を解決するには**

- より多くの接続が許可されるように Windows ライセンスをアップグレードします。
- マルチブローカクラスタを設定して、多数のブローカインスタンスに接続を分散します。

### *ユーザーの認証に失敗するか権限が与えられない*

不正なパスワード、ユーザーリポジトリにユーザーのエントリがない、またはユーザーが接続サービスへのアクセス許可を持っていないといった原因で、認証は失敗することがあります。

**この問題の原因を確認するには**

ブローカログのエントリで `Forbidden` エラーメッセージを確認します。このメッセージは、認証エラーを示しているだけで、その理由は示していません。

- ファイルベースのユーザーリポジトリを使用している場合は、次のコマンドを入力します。

  `imqusermgr list -i` *instanceName* `-u` *userName*

- 出力にユーザーが表示された場合は、不正なパスワードの入力が原因と考えられます。出力に次のエラーが表示された場合は、ユーザーリポジトリにエントリがありません。

  ```
  Error [B3048]: User does not exist in the password file,
  ```

- LDAP サーバーのユーザーリポジトリを使用している場合は、適切なツールを使用して、ユーザーのエントリがあるかどうかを確認します。
- アクセス制御プロパティーファイルで、接続サービスへのアクセスが制限されていないかどうかを確認します。

**問題を解決するには**

- ユーザーリポジトリにユーザーのエントリがない場合は、ユーザーリポジトリにユーザーを追加します (140 ページの「ユーザーリポジトリの設定と管理」を参照 )。
- 不正なパスワードが使用された場合は、正しいパスワードを入力し直します。
- アクセス制御プロパティーが不正に設定されていた場合は、アクセス制御プロパティーファイルを編集し、接続サービスへのアクセス許可を与えます (148 ページの「接続サービスのアクセス制御」を参照 )。

# 接続スループットが遅すぎる

この問題では、次の症状がみられます。

- メッセージスループットが期待どおりでない。
- サポートされるブローカへの接続数が、240 ページの「クライアントが接続を確立できない」で説明されている原因によってではなく、メッセージの入力 / 出力レートによって制限されている。

この節では、次の原因について説明します。

- ネットワーク接続または WAN が遅すぎる
- 接続サービスプロトコルが、TCP に比べて本質的に低速である
- 接続サービスプロトコルが最適に調整されていない
- メッセージのサイズが大きく、多くの帯域幅を占有してしまう
- 接続スループットが低速であるように見えるが、実際は、メッセージ配信プロセスのほかの手順にボトルネックがある

## *ネットワーク接続または WAN が遅すぎる*

**この問題の原因を確認するには**

ネットワークへ ping し、ping が戻るまでに要する時間を確認し、ネットワーク管理者に相談します。また、ローカルクライアントを使用してメッセージを送受信し、ネットワークリンク経由でリモートクライアントを使用した場合と配信時間を比較することもできます。

**問題を解決するには**

接続が遅すぎる場合は、ネットワークリンクをアップグレードします。

### *接続サービスプロトコルが、TCP に比べて本質的に低速である*

たとえば、SSL ベースプロトコルや HTTP ベースプロトコルは、TCP より低速です (226 ページの図 11-5 を参照 )。

**この問題の原因を確認するには**

SSL ベースのプロトコルまたは HTTP ベースのプロトコルを使用している場合は、TCP を使用して配信時間を比較してみます。

**問題を解決するには**

通常、アプリケーション要件によって使用するプロトコルが決定されます。そのため、230 ページの「トランスポートプロトコルの調整」の説明に従いプロトコルを調整する以外に、対処方法はほとんどありません。

### *接続サービスプロトコルが最適に調整されていない*

**この問題の原因を確認するには**

プロトコルを調整し、違いが生じるかどうかを確認します。

**問題を解決するには**

230 ページの「トランスポートプロトコルの調整」の説明にしたがいプロトコルを調整します。

### *メッセージのサイズが大きく、多くの帯域幅を占有してしまう*

**この問題の原因を確認するには**

小さいサイズのメッセージでベンチマークを実行します。

**問題を解決するには**

- メッセージ圧縮機能を使用するように、アプリケーション開発者にアプリケーションを修正してもらいます。『Message Queue Developer's Guide for Java Clients』を参照してください。
- データを送信することの通知としてメッセージを使用し、データの送信には別のプロトコルを使用します。

### *接続スループットが低速であるように見えるが、実際は、メッセージ配信プロセスのほかの手順にボトルネックがある*

**この問題の原因を確認するには**

接続スループットが低速であるように見えるが、前に述べたような原因が見当たらない場合は、215 ページの図 11-1 を参照して、そのほかの考えられるボトルネックを特定し、次の問題に関連する現象が出ていないかどうかを確認します。

- 247 ページの「メッセージの生成が遅れるまたは低速である」
- 250 ページの「メッセージがバックログされる」
- 255 ページの「メッセージサーバーのスループットが散発的である」

**問題を解決するには**

前に述べた問題のトラブルシューティングの節に記載された問題の解決方法に従います。

# クライアントがメッセージプロデューサを作成できない

この問題では、次の症状がみられます。

- メッセージプロデューサが物理的送信先に対して作成できず、クライアントは例外を受け取る。

この節では、次の原因について説明します。

- 限定された数のプロデューサだけを許可するように物理的送信先が設定されている
- アクセス制御プロパティーファイル内の設定により、ユーザーがメッセージプロデューサの作成を承認されていない

### *限定された数のプロデューサだけを許可するように物理的送信先が設定されている*

物理的送信先でのメッセージの蓄積を回避する 1 つの方法は、サポートされるプロデューサの数 (`maxNumProducers`) を限定することです。

**この問題の原因を確認するには**

物理的送信先を確認します (124 ページの「物理的送信先の情報の表示」を参照 )。

```
imqcmd query dst
```

出力に現在のプロデューサ数と maxNumProducers の値が表示されます。2 つの値が同じ場合、プロデューサ数は設定済みの制限に達しています。ブローカは新しいプロデューサを拒否したときには、ResourceAllocationException [C4088]: A JMS destination limit was reached を返し、ブローカログに次のエントリを作成します。[B4183]: Producer can not be added to destination

**問題を解決するには**

maxNumProducers 属性の値を大きくします (125 ページの「物理的送信先のプロパティーの更新」を参照)。

### *アクセス制御プロパティーファイル内の設定により、ユーザーがメッセージプロデューサの作成を承認されていない*

**この問題の原因を確認するには**

ブローカは、新しいプロデューサを拒否したとき、次のメッセージを返します。

```
JMSSecurityException [C4076]: Client does not have permission to
create producer on destination
```

ブローカは、ブローカログに次のエントリも作成します。

[B2041]: Producer on destination denied および [B4051]: Forbidden guest

**問題を解決するには**

ユーザーがメッセージを生成できるようにアクセス制御プロパティーを変更します (149 ページの「物理的な送信先のアクセス制御」を参照)。

## メッセージの生成が遅れるまたは低速である

この問題では、次の症状がみられます。

- 持続メッセージを送信したときに、send メソッドが戻らずクライアントがブロックする。
- 持続メッセージを送信したときに、クライアントが例外を受け取る。
- プロデューシングクライアントの処理速度が低下する。

この節では、次の原因について説明します。

- メッセージサーバーがバックログされ、メッセージプロデューサの処理速度を低下させることによって対処される
- ブローカが持続メッセージをデータストアに保存できない
- ブローカによる通知のタイムアウトが短すぎる
- プロデューシングクライアントが JVM 制限に達している

### *メッセージサーバーがバックログされ、メッセージプロデューサの処理速度を低下させることによって対処される*

バックログされたサーバーでは、ブローカメモリーにメッセージが蓄積します。

物理的送信先メモリー内のメッセージ数またはメッセージのバイト数が設定された制限に達すると、ブローカは指定された制限の動作に従いメモリーリソースを節約しようとします。次の制限の動作により、メッセージプロデューサの処理速度が低下します。

- FLOW_CONTROL: ブローカが持続メッセージの受信を即時に通知せず、プロデューシングクライアントがブロックされる。
- REJECT_NEWEST: ブローカが新しい持続メッセージを拒否する。

同様に、ブローカ全体のメモリー内 ( すべての物理的送信先に対応 ) のメッセージ数またはメッセージのバイト数が設定済みの制限に達すると、ブローカは最新のメッセージを拒否してメモリーリソースを節約しようとします。

また、物理的送信先またはブローカ全体の制限が適切に設定されていないために、システムメモリーの制限に達すると、ブローカはさらに大規模なアクションを実行してメモリーの過負荷を防ぎます。このアクションには、メッセージプロデューサを徐々に減らすことなどがあります。

#### この問題の原因を確認するには

設定済みのメッセージ制限が原因でブローカによってメッセージが拒否された場合は、ブローカが次のメッセージを返します。

```
JMSException [C4036]: A server error occurred
```

ブローカは、ブローカログに次のエントリも作成します。

```
WARNING [B2011]: Storing of JMS message from IMQconn failed
```

このメッセージには、到達した制限を示すメッセージが続きます。物理的送信先上でメッセージが制限されている場合、ブローカは次のようなエントリを作成します。

```
[B4120]: Can not store message on destination destName because capacity
of maxNumMsgs would be exceeded.
```

ブローカ全体でメッセージが制限されている場合、ブローカは次のようなエントリを作成します。

```
[B4024]: The Maximum Number of messages currrently in the system has
been exceeded, rejecting message.
```

より一般的には、拒否が発生する前に、次のようにメッセージ制限条件を確認します。

- 物理的送信先とブローカを照会し、設定されているメッセージ制限の設定を調べます。

- 適切な imqcmd コマンドを使用し、物理的送信先かブローカ全体に現在あるメッセージの数かバイト数を監視します。監視できるメトリックス、およびメトリックスの取得に使用するコマンドについては、第 18 章「メトリックスのリファレンス」を参照してください。

**問題を解決するには**

メッセージがバックログされたことでプロデューサの処理が低下する問題を解消するためのアプローチは多数あります。

- 物理的送信先またはブローカ全体のメッセージ制限を、メモリーリソースを超えないように注意しながら変更します。

  一般に、ブローカ全体のメッセージ制限に達しないように、送信先単位でメモリーを管理する必要があります。詳細は、234 ページの「ブローカの調整」を参照してください。

- メッセージ制限に達したときに、メッセージの生成が低速化しないようにする代わりに、メモリー内のメッセージを廃棄するよう、送信先の制限の動作を変更します。

  たとえば、メモリーに累積されたメッセージを削除する REMOVE_OLDEST および REMOVE_LOW_PRIORITY といった制限の動作を指定できます (325 ページの表 15-1 を参照 )。

### *ブローカが持続メッセージをデータストアに保存できない*

ブローカがデータストアにアクセスできないか、または持続メッセージをデータストアに書き込めない場合は、プロデューシングクライアントがブロックされます。前に述べたとおり、この状態は、送信先またはブローカ全体のメッセージ制限に達したときにも発生します。

**この問題の原因を確認するには**

ブローカは、データストアに書き込めない場合には、ブローカログに次のエントリのどれかを作成します。`[B2011]: Storing of JMS message from connectionID failed...` または `[B4004]: Failed to persist message messageID...`

**問題を解決するには**

- ファイルベースの持続の場合は、ファイルベースのデータストアのディスクスペースを増やしてみます。
- JDBC 互換のデータストアの場合は、JDBC ベースの持続が正しく設定されていることを確認します ( 第 4 章「ブローカの設定」を参照 )。正しく設定されている場合は、データベース管理者にほかのデータベース問題の解決を依頼します。

### *ブローカによる通知のタイムアウトが短すぎる*

低速な接続または、CPU 使用率が高いかメモリーリソースが不十分なためにメッセージサーバーの能力が低下したことが原因で、ブローカが持続メッセージの受信を通知するまでに、接続ファクトリの imqAckTimeout 属性値で許容されている以上の時間を必要としています。

**この問題の原因を確認するには**

imqAckTimeout 値を超えると、ブローカは次のメッセージを返します。

```
JMSException [C4000]: Packet acknowledge failed
```

**問題を解決するには**

imqAckTimeout 接続ファクトリ属性の値を変更します (174 ページの「信頼性およびフロー制御」を参照)。

### *プロデューシングクライアントが JVM 制限に達している*

**この問題の原因を確認するには**

- クライアントアプリケーションがメモリー不足エラーを受け取ったかどうかを確認します。
- freeMemory、MaxMemory、および totalMemory などのランタイムメソッドを使用して JVM ヒープの使用可能な空きメモリーを確認します。

**問題を解決するには**

JVM を調整します (229 ページの「Java 仮想マシン (JVM) の調整」を参照)。

# メッセージがバックログされる

この問題では、次の症状がみられます。

- ブローカまたは特定の送信先のメッセージ数またはメッセージのバイト数が時間の経過とともに徐々に増えていく。

  メッセージが蓄積されているかどうかを確認するため、ブローカ内のメッセージ数またはメッセージのバイト数が時間の経過とともにどのように変化するかを確認し、設定済みの制限と比較します。最初に、設定済みの制限を確認します。

  ```
  imqcmd query bkr
  ```

  **注**：imqcmd metrics bkr サブコマンドは、この情報を表示しません。

  その後、各送信先でのメッセージの蓄積を確認します。

  ```
  imqcmd list dst
  ```

メッセージが設定済みの送信先またはブローカ全体の制限を超えているかどうかを判断するため、ブローカログで次のエントリを確認します。`WARNING [B2011]: Storing of JMS message from...failed.` このエントリには、超過した制限について説明する別のエントリが続きます。

- メッセージの生成が遅い、または生成されたメッセージがブローカによって拒否される。
- メッセージがコンシューマに到達するまでに異常に長い時間がかかる。

この節では、次の原因について説明します。

- トピック送信先に非アクティブな永続サブスクリプションがある
- キュー内のメッセージを消費するための使用可能なコンシューマが少なすぎる
- メッセージプロデューサの処理速度についていくには、メッセージコンシューマの処理速度が遅すぎる
- クライアントの通知処理が、メッセージの消費を遅くする
- 生成されたメッセージの処理にブローカが追いつけない
- クライアントコードの欠陥 : コンシューマがメッセージを通知していない

### *トピック送信先に非アクティブな永続サブスクリプションがある*

永続サブスクリプションが非アクティブな場合は、該当するコンシューマがアクティブになりメッセージを消費できるようになるまで、メッセージは送信先に格納されます。

**この問題の原因を確認するには**

各トピック送信先の永続サブスクリプションの状態を確認します。

```
imqcmd list dur -d destName
```

**問題を解決するには**

次のアクションのどれかを実行できます。

- 原因となっている永続サブスクリプションのすべてのメッセージを消去します (114 ページの「永続サブスクリプションの管理」を参照 )。
- トピックのメッセージの制限と制限の動作属性を指定します (325 ページの表 15-1 を参照 )。たとえば、メモリーに累積されたメッセージを削除する `REMOVE_OLDEST` および `REMOVE_LOW_PRIORITY` といった制限の動作を指定できます。
- 該当する送信先からすべてのメッセージを消去します (127 ページの「物理的送信先の消去」を参照 )。

- メッセージをメモリー内で存続できる時間を制限します。プロデューシングクライアントをプログラムし直し、メッセージごとに生存時間の値を設定できます。imqOverrideJMSExpiration および imqJMSExpiration 接続ファクトリ属性を設定することで、接続を共有するすべてのプロデューサのこれらの設定値をオーバーライドできます (338 ページの「メッセージヘッダーのオーバーライド」を参照 )。

### *キュー内のメッセージを消費するための使用可能なコンシューマが少なすぎる*

メッセージを配信可能なアクティブなコンシューマが少なすぎる場合は、メッセージが蓄積するにつれ、キュー送信先がバックログされる恐れがあります。この状態は、次の理由のどれかが原因で発生することがあります。

- 送信先に対応するアクティブなコンシューマが少なすぎる。
- コンシューミングクライアントが接続の確立に失敗した。
- アクティブなコンシューマがキュー内のメッセージに一致するセレクタを使用していない。

#### この問題の原因を確認するには

コンシューマが使用できない理由を判断するために、送信先のアクティブなコンシューマの数を確認します。

```
imqcmd metrics dst -n destName -t q -m con
```

#### 問題を解決するには

コンシューマが使用できない理由に応じて、次のアクションのどれかを実行できます。

- 追加のコンシューミングクライアントを起動して、キューに対応するアクティブなコンシューマを増やします。
- imq.consumerFlowLimit ブローカプロパティーを調整して、複数のコンシューマへのキュー配信を最適化します (235 ページの「複数のコンシューマキューのパフォーマンス」を参照 )。
- キューのメッセージの制限と制限の動作属性を指定します (325 ページの表 15-1 を参照 )。たとえば、メモリーに累積されたメッセージを削除する REMOVE_OLDEST および REMOVE_LOW_PRIORITY といった制限の動作を指定できます。
- 該当する送信先からすべてのメッセージを消去します (127 ページの「物理的送信先の消去」を参照 )。
- メッセージをメモリー内で存続できる時間を制限します。プロデューシングクライアントをプログラミングし直し、メッセージごとに生存期間の値を設定できます。imqOverrideJMSExpiration および imqJMSExpiration 接続ファクトリ属性を設定することで、接続を共有するすべてのプロデューサのこれらの設定値をオーバーライドできます (338 ページの「メッセージヘッダーのオーバーライド」を参照 )。

### *メッセージプロデューサの処理速度についていくには、メッセージコンシューマの処理速度が遅すぎる*

この場合、トピックのサブスクライバまたはキューの受信側は、プロデューサがメッセージを送信する速度より遅い速度でメッセージを消費しています。この不均衡が原因で、複数の送信先にメッセージがバックログされています。

**この問題の原因を確認するには**

ブローカとの間のメッセージのフローレートを確認します。

```
imqcmd metrics bkr -m rts
```

その後、個々の送信先についてそれぞれのフローレートを確認します。

```
imqcmd metrics bkr -t destType -n destName -m rts
```

**問題を解決するには**

- コンシューミングクライアントコードを最適化します。
- キュー送信先の場合は、アクティブなコンシューマの数を増やします (235 ページの「複数のコンシューマキューのパフォーマンス」を参照 )。

### *クライアントの通知処理が、メッセージの消費を遅くする*

クライアントの通知処理には 2 つの要因が影響しています。

- クライアント通知の処理時に、大量のブローカリソースが使用されることがあります。その結果、このような通知モードでは、ブローカがクライアント通知を確認するまでコンシューミングクライアントがブロックされるので、メッセージの消費が遅くなることがあります。
- JMS ペイロードメッセージと、クライアント通知などの Message Queue 制御メッセージは同じ接続を共有します。その結果、制御メッセージが JMS ペイロードメッセージによって保留され、メッセージの消費を低速化させることがあります。

**この問題の原因を確認するには**

- メッセージのフローをパケットのフローと比較して確認します。1 秒当たりのパケット数がメッセージの数と比例していない場合は、クライアントの通知が問題と考えられます。
- クライアントが次のメッセージを受信したかどうかを確認します。

```
JMSException [C4000]: Packet acknowledge failed
```

**問題を解決するには**

- クライアントの通知モードを変更します。たとえば、`DUPS_OK_ACKNOWLEDGE` または `CLIENT_ACKNOWLEDGE` に切り換えます。
- `CLIENT_ACKNOWLEDGE` または処理済みのセッションを使用している場合は、より多数のメッセージを単一の通知にグループ化します。

- コンシューマと接続のフロー制御パラメータを調整します (236 ページの「クライアントランタイムのメッセージフローの調整」を参照 )。

### *生成されたメッセージの処理にブローカが追いつけない*

この場合、ブローカがメッセージをコンシューマにルーティングおよび配信可能な速度より速く、ブローカにメッセージが流入しています。ブローカの遅滞は、次のどれかまたはすべてにおける制限が原因と考えられます。それは、CPU、ネットワークソケットの読み取り / 書き込み操作、ディスク読み取り / 書き込み操作、メモリーのページング、持続ストア、または JVM メモリー制限です。

#### この問題の原因を確認するには

この問題にそれ以外の原因が関与していないことを確認します。

#### 問題を解決するには

- コンピュータまたはデータストアの速度をアップグレードします。
- ブローカクラスタを使用して、多数のブローカインスタンスに負荷を分散します。

### *クライアントコードの欠陥 : コンシューマがメッセージを通知していない*

メッセージは、すべてのコンシューマによってメッセージの送信先へ通知されるまで、送信先で保持されます。クライアントが消費したメッセージを通知しない場合、メッセージは削除されずに送信先で蓄積されます。

たとえば、クライアントコードは次の欠陥を持っている可能性があります。

- `CLIENT_ACKNOWLEDGEacknowledgment` または処理済みセッションを使用しているコンシューマが、定期的に `Session.acknowledge` または `Session.commit` を呼び出していない。
- `AUTO_ACKNOWLEDGE` セッションを使用しているコンシューマが何らかの理由で停止している。

#### この問題の原因を確認するには

この節で挙げられている、その他すべての考えられる原因を確認します。次に、以下のコマンドを使用し、送信先を一覧表示します。

```
imqcmd list dst
```

ヘッダー「UnAcked」の下に一覧表示されるメッセージの数が、送信先のメッセージの数と同じであるかどうか確認してください。ヘッダー「UnAcked」の下のメッセージはコンシューマに送信されますが、通知されません。この数がメッセージの総数と同じである場合、ブローカはすべてのメッセージを送信し、通知を待機しています。

#### 問題を解決するには

アプリケーション開発者にこの問題をデバッグしてもらうように依頼します。

# メッセージサーバーのスループットが散発的である

この問題では、次の症状がみられます。

- メッセージのスループットがときどき低下し、その後通常のパフォーマンスに戻る。

この節では、次の原因について説明します。

- ブローカのメモリーリソースがかなり不足している
- JVM メモリーの再利用 ( ガベージコレクション ) を実行する
- JVM は JIT コンパイラを使用してパフォーマンスを高速化させる

## *ブローカのメモリーリソースがかなり不足している*

送信先とブローカに制限が適切に設定されなかったため、ブローカはメモリーが過負荷になるのを防ぐためにさらに大規模なアクションを実行します。このため、メッセージのバックログがクリアされるまでは、ブローカの処理がかなり遅くなります。

**この問題の原因を確認するには**

ブローカのログで、メモリー不足の状態になっていないかどうかを確認します。`[B1089]: In low memory condition, broker is attempting to free up resources` に続き、メモリーの最新の状態と、使用中のメモリーの合計を示すエントリが表示されます。

また、JVM ヒープ内の使用可能な空きメモリーも確認します。

```
imqcmd metrics bkr -m cxn
```

JVM メモリーの合計値が JVM メモリーの最大値に近くなると、空きメモリーは不足がちになります。

**問題を解決するには**

- JVM を調整します (229 ページの「Java 仮想マシン (JVM) の調整」を参照 )。
- システムスワップスペースを増やします。

## *JVM メモリーの再利用 ( ガベージコレクション ) を実行する*

定期的なメモリー再利用によりシステム全体を一掃し、メモリーを解放します。これが実行されると、すべてのスレッドがブロックされます。より多くのメモリーが解放され、JVM ヒープサイズがより大きくなるほど、メモリー再利用に起因する遅延も長くなります。

**この問題の原因を確認するには**

コンピュータ上の CPU 使用率を監視します。メモリーが再利用されるとき、CPU 使用率は下がります。

また、次のコマンド行オプションを使用してブローカを起動します。

```
-vmargs -verbose:gc
```

標準出力では、メモリー再利用に要した時間が示されます。

**問題を解決するには**

複数の CPU を持つコンピュータでは、メモリー再利用を並行して実行するように設定します。

```
-XX:+UseParallelGC=true
```

### *JVM は JIT コンパイラを使用してパフォーマンスを高速化させる*

**この問題の原因を確認するには**

この問題にそれ以外の原因が関与していないことを確認します。

**問題を解決するには**

しばらくの間システムを稼働させておくと、パフォーマンスは改善するはずです。

# メッセージがコンシューマに到達しない

この問題では、次の症状がみられます。

- プロデューサによって送信されたメッセージをコンシューマが受信しない

この節では、次の原因について説明します。

- 制限の動作が、ブローカでのメッセージの削除を引き起こしている
- メッセージタイムアウト値が期限切れになる
- クロックが同期化しない
- コンシューミングクライアントが接続でのメッセージ配信の開始に失敗した

### *制限の動作が、ブローカでのメッセージの削除を引き起こしている*

送信先メモリー内のメッセージ数またはメッセージのバイト数が設定済みの制限に達すると、ブローカはメモリーリソースを節約しようとします。これらの制限に達したときにブローカが実行する 3 つの設定可能な動作によって、メッセージが失われることがあります。

- `REMOVE_OLDEST`: もっとも古いメッセージを削除する。

- REMOVE_LOW_PRIORITY: メッセージの有効期間に従いもっとも優先度の低いメッセージを削除する。
- REJECT_NEWEST: 新しい持続メッセージを拒否する。

ブローカのメモリー内のメッセージ数またはメッセージのバイト数が設定済みの制限に達すると、ブローカは最新のメッセージを拒否してメモリーリソースを節約しようとします。

**この問題の原因を確認するには**

260 ページの「デッドメッセージキューにメッセージが含まれる」の説明に従い、デッドメッセージキューを確認します。特に261 ページの「メッセージ数かメッセージサイズが送信先の制限を超える」の指示に従ってください。REMOVE_OLDEST か REMOVE_LOW_PRIORITY の理由を探します。

**問題を解決するには**

送信先の制限を上げます。たとえば、次のように指定します。

```
imqcmd update dst -n MyDest -o maxNumMsgs=1000
```

### *メッセージタイムアウト値が期限切れになる*

ブローカは、タイムアウトして期限切れになったメッセージを削除します。送信先がメッセージで過分にバックログされている場合、生存期間の値が短すぎるメッセージは削除されます。

**この問題の原因を確認するには**

デッドメッセージキューを確認し、メッセージがタイムアウトになったかどうかを確認します。

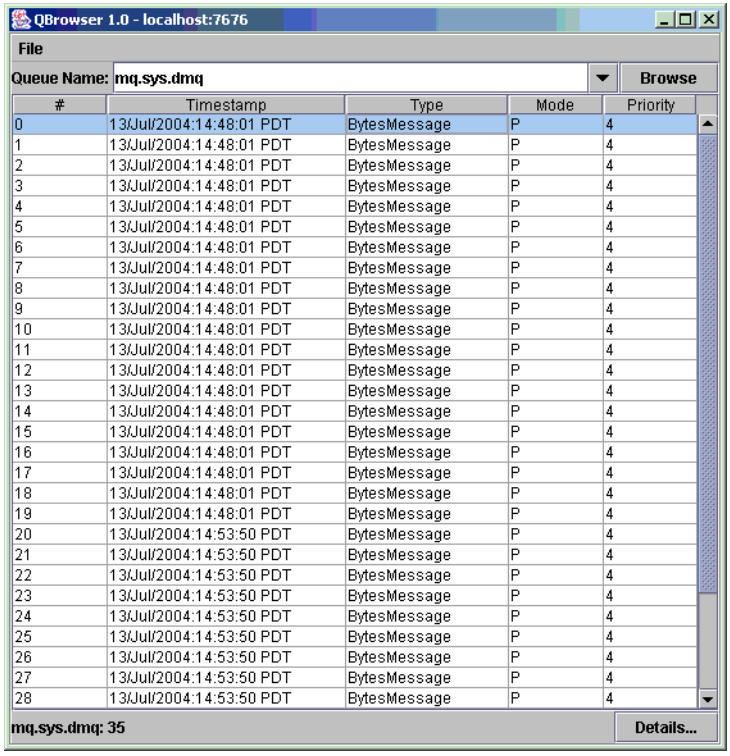
QBrowser デモアプリケーションを使用し、DMQ の内容を調べます。QBrowser デモアプリケーションの場所は、オペレーティングシステムによって異なります。場所については、付録 A「プラットフォームごとの Message Queue データの場所」を参照し、アプリケーション例と場所の表を調べてください。

以下は、Windows における呼び出し例です。

```
cd ¥MessageQueue3¥demo¥applications¥qbrowser java QBrowser
```

QBrowser のメインウィンドウが表示されたら、キュー名 mq.sys.dmq を選択してから「Browse」をクリックします。次のようなリストが表示されます。

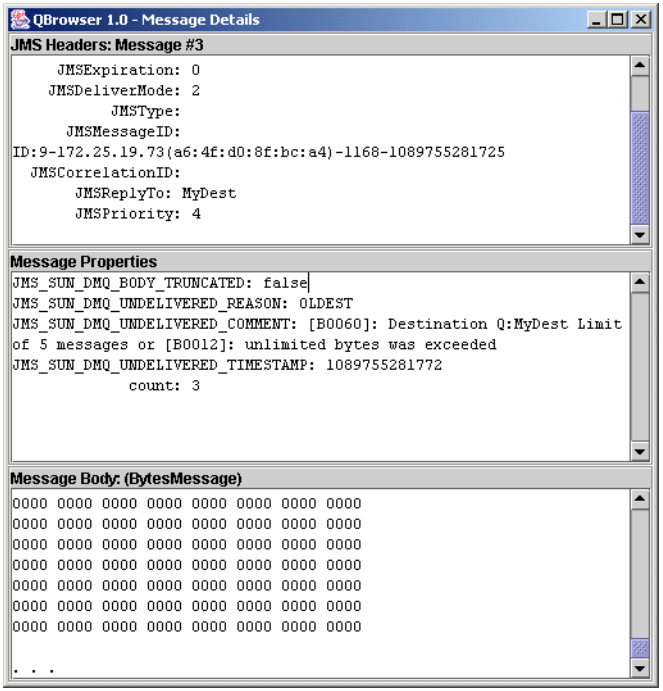
図 12-1　QBrowser のウィンドウ




| # | Timestamp | Type | Mode | Priority |
|---|---|---|---|---|
| 0 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 1 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 2 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 3 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 4 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 5 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 6 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 7 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 8 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 9 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 10 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 11 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 12 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 13 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 14 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 15 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 16 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 17 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 18 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 19 | 13/Jul/2004:14:48:01 PDT | BytesMessage | P | 4 |
| 20 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |
| 21 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |
| 22 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |
| 23 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |
| 24 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |
| 25 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |
| 26 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |
| 27 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |
| 28 | 13/Jul/2004:14:53:50 PDT | BytesMessage | P | 4 |

メッセージをダブルクリックすると、そのメッセージの詳細が表示されます。

図 12-2　QBrowser のメッセージ詳細




メッセージの JMS_SUN_DMQ_UNDELIVERED_REASON プロパティー値が EXPIRED に設定されているかどうか確認してください。

**問題を解決するには**

アプリケーション開発者と相談し、生存時間の値を上げます。

### *クロックが同期化しない*

クロックが同期化されていない場合、ブローカによるメッセージの生存期間の計算が誤りとなり、メッセージが有効期限より早く削除される場合があります。

**この問題の原因を確認するには**

ブローカのログファイルで、B2102、B2103、B2104 のメッセージを探します。このメッセージはすべて、クロックスキューが検出されたことを報告します。

**問題を解決するには**

67 ページの「システムリソースの準備」の説明に従い、時刻同期プログラムが動作していることを確認します。

### *コンシューミングクライアントが接続でのメッセージ配信の開始に失敗した*

クライアントコードが接続を確立し、その接続上でメッセージ配信を開始するまで、メッセージは配信できません。

**この問題の原因を確認するには**

クライアントコードが接続を確立しメッセージ配信を開始したことを確認します。

**問題を解決するには**

接続を確立しメッセージ配信を開始するように、クライアントコードをプログラミングし直します。

# デッドメッセージキューにメッセージが含まれる

この問題では、次の症状がみられます。

- 送信先を一覧表示したとき、デッドメッセージキューにメッセージが含まれていることが表示されます。たとえば次のようなコマンドを実行します。

  ```
  imqcmd list dst
  ```

  ユーザー名とパスワードを入力したあとで、次のような出力が表示されます。

```
Listing all the services on the broker specified by:
--------------------------------
Host         Primary Port
--------------------------------
localhost    7676
-------------------------------------------------------------------------
   Name     Type    State   Producers  Consumers Msgs
                                        Total Count  UnAck  Avg Size
-------------------------------------------------------------------------
MyDest      Queue  RUNNING  0          0          5     0      1177.0
mq.sys.dmq  Queue  RUNNING   0          0         35     0      1422.0
Successfully listed destinations.
```

  この例では、デッドメッセージキュー mq.sys.dmq に 35 個のメッセージが含まれています。

この節では、次の原因について説明します。

- メッセージ数かメッセージサイズが送信先の制限を超える
- ブローカのクロックとプロデューサのクロックが同期化しない
- コンシューマがメッセージを受信せずにメッセージがタイムアウトになる
- コンシューマの数に対してプロデューサが多すぎる

- プロデューサがコンシューマより速い
- コンシューマが遅すぎる
- クライアントがメッセージをコミットしない
- 永続コンシューマがアクティブにならない
- 予期しないブローカエラーが発生する

### *メッセージ数かメッセージサイズが送信先の制限を超える*

**この問題の原因を確認するには**

QBrowser デモアプリケーションを使用し、デッドメッセージキューの内容を調べます。QBrowser デモアプリケーションの場所は、オペレーティングシステムによって異なります。場所については、付録 A「プラットフォームごとの Message Queue データの場所」を参照し、アプリケーション例と場所の表を調べてください。

以下は、Windows における呼び出し例です。

```
cd ¥MessageQueue3¥demo¥applications¥qbrowser java QBrowser
```

QBrowser のメインウィンドウが表示されたら、キュー名 `mq.sys.dmq` を選択してから「Browse」をクリックします。258 ページの図 12-1 に示したようなリストが表示されます。

メッセージをダブルクリックすると、そのメッセージの詳細が表示されます。259 ページの図 12-2 で示したウィンドウが表示されます。

次のメッセージプロパティーの値を確認してください。

- `JMS_SUN_DMQ_UNDELIVERED_REASON`
- `JMS_SUN_DMQ_UNDELIVERED_COMMENT`
- `JMS_SUN_DMQ_UNDELIVERED_TIMESTAMP`

「JMS Headers」の下で `JMSDestination` の値を調べ、メッセージが終了している送信先を判断します。

**問題を解決するには**

送信先の制限を上げます。たとえば、次のように指定します。

```
imqcmd update dst -n MyDest -o maxNumMsgs=1000
```

### *ブローカのクロックとプロデューサのクロックが同期化しない*

この問題の原因を確認するには

QBrowser アプリケーションを使用し、デッドメッセージキューのメッセージのメッセージ詳細を表示します。`JMS_SUN_DMQ_UNDELIVERED_REASON` の値を確認し、理由が `EXPIRED` になっているメッセージを探します。

ブローカのログファイルで、B2102、B2103、B2104 のメッセージを探します。このメッセージはすべて、クロックスキューが検出されたことを報告します。

**問題を解決するには**

67 ページの「システムリソースの準備」の説明に従い、時刻同期プログラムが動作していることを確認します。

### *コンシューマがメッセージを受信せずにメッセージがタイムアウトになる*

**この問題の原因を確認するには**

QBrowser アプリケーションを使用し、デッドメッセージキューのメッセージのメッセージ詳細を表示します。`JMS_SUN_DMQ_UNDELIVERED_REASON` の値を確認し、理由が `EXPIRED` になっているメッセージを探します。

送信先にコンシューマがあるかどうかを確認します。たとえば、次のように指定します。

```
imqcmd query dst -t q -n MyDest
```

Active Consumers の Current Number に一覧表示される値を確認します。アクティブなコンシューマがある場合は、次のうちいずれかに該当します。

- コンシューマの接続が一時停止している。
- コンシューマの実行速度を考慮すると、メッセージのタイムアウトが短すぎる。

**問題を解決するには**

アプリケーション開発者に、メッセージの生存時間の値を上げてもらいます。

### *コンシューマの数に対してプロデューサが多すぎる*

**この問題の原因を確認するには**

QBrowser アプリケーションを使用し、デッドメッセージキューのメッセージのメッセージ詳細を表示します。`JMS_SUN_DMQ_UNDELIVERED_REASON` の値を確認します。

この値が `REMOVE_OLDEST` か `REMOVE_LOW_PRIORITY` である場合は、`imqcmd query dst` コマンドを使用し、送信先のプロデューサ数とコンシューマ数を確認します。プロデューサ数がコンシューマ数より多い場合は、生成レートが消費レートを超えている可能性があります。

### 問題を解決するには

コンシューマクライアントを追加するか、FLOW_CONTROL 制限動作を使用するように送信先を設定します。FLOW_CONTROL 制限動作では、消費レートが使用されて生成レートが制御されます。

次の例のようなコマンドを使用し、フロー制御動作を起動します。

```
imqcmd update dst -n myDst -t q -o consumerFlowLimit=FLOW_CONTROL
```

## *プロデューサがコンシューマより速い*

### この問題の原因を確認するには

低速コンシューマがプロデューサの減速の原因になっているかどうか判断するには、送信先制限動作を FLOW_CONTROL に設定します。FLOW_CONTROL 制限動作では、消費レートが使用されて生成レートが制御されます。

次の例のようなコマンドを使用し、フロー制御動作を起動します。

```
imqcmd update dst -n myDst -t q -o consumerFlowLimit=FLOW_CONTROL
```

次の例のようなコマンドを実行し、メトリックスを使用して、送信先の入力と出力を調べます。

```
imqcmd metrics dst -n myDst -t q -m rts
```

メトリックスの出力で、次の値を調べます。

- Msgs/sec Out

  この値は、ブローカが 1 秒あたりに削除したメッセージ数を示します。すべてのコンシューマがメッセージの受信を通知したとき、ブローカはメッセージを削除するので、このメトリックスには消費レートが反映されます。

- Msgs/sec In

  この値は、ブローカが 1 秒あたりにプロデューサから受信したメッセージ数を示します。このメトリックスには生成レートが反映されます。

フロー制御では生成が消費に調整されるので、生成が低速になるか停止しているか確認します。レートが低速になるか停止している場合は、プロデューサとコンシューマの処理速度に相違があります。

imqcmd list dst コマンドを使用し、未通知 (UnAcked) 送信メッセージの数を確認することもできます。未通知メッセージ数が送信先のサイズより小さい場合、送信先では容量に余裕がありますが、送信先はクライアントフロー制御によって抑制されています。

### 問題を解決するには

生成レートが消費レートより常に速い場合は、フロー制御を定期的に使用することを考慮し、システムを調整します。

また、後続の節を参照し、次の考えられる要因の解決を考慮するか試してください。

- コンシューマが遅すぎる
- クライアントがメッセージをコミットしない
- コンシューマがメッセージを通知しない
- 永続コンシューマがアクティブにならない
- 予期しないブローカエラーが発生する

### *コンシューマが遅すぎる*

#### この問題の原因を確認するには

263 ページの「プロデューサがコンシューマより速い」の説明に従い、メトリックスを使用して、生成と消費のレートを判断します。

#### 問題を解決するには

次のうち 1 つ以上を試します。

- FLOW_CONTROL 制限動作を使用するように送信先を設定します。次のようなコマンドを使用します。

  ```
  imqcmd update dst -n myDst -t q -o consumerFlowLimit=FLOW_CONTROL
  ```

  フロー制御を使用すると、消費のレートまで生成が減速し、ブローカにおけるメッセージの蓄積が防止されます。送信先が適時にメッセージを処理できるようになり、期限切れになる可能性が低くなるまで、プロデューサアプリケーションはメッセージを抑制します。

- プロデューサが安定したレートでメッセージを送信しているか、定期的に大量のメッセージを送信しているかをアプリケーション開発者に尋ねます。

  アプリケーションが大量のメッセージを送信している場合は、次の項目の指示に従い、送信先の制限を上げます。

- メッセージ数かバイト数、またはその両方に基づいて、送信先の制限を上げます。

  送信先のメッセージ数を変更するには、次の形式のコマンドを入力します。

  ```
  imqcmd update dst -n destName -t {q/t} -o maxNumMsgs=number
  ```

  送信先のサイズを変更するには、次の形式のコマンドを入力します。

  ```
  imqcmd update dst -n destName -t {q/t} -o maxTotalMsgBytes=number
  ```

  制限を上げると、ブローカが使用するメモリー量が増えることに注意してください。制限が高すぎる場合は、ブローカでメモリーが不足し、メッセージを処理できなくなることがあります。

- 生成負荷のレベルが高い間、メッセージの喪失を受け入れることができるかどうかを考慮します。

### *クライアントがメッセージをコミットしない*

**この問題の原因を確認するには**

アプリケーション開発者と協力し、アプリケーションでトランザクションが使用されているかどうかを調べます。アプリケーションでトランザクションが使用されている場合は、次のようにアクティブなトランザクションを一覧表示します。

```
imqcmd list txn
```

以下は、コマンド出力の例です。

```
-------------------------------------------------------------------
Transaction ID       State    User name  # Msgs/# Acks   Creation time
-------------------------------------------------------------------
6800151593984248832  STARTED  guest       3/2         7/19/04 11:03:08 AM
```

メッセージ数と通知数に注意してください。

メッセージ数が多い場合は、プロデューサがそれぞれのメッセージを送信しているが、トランザクションのコミットには失敗している可能性があります。ブローカは、コミットを受信するまで、そのトランザクションのメッセージをルーティングしたり配信したりすることができません。

通知数が多い場合は、コンシューマがメッセージごとに通知を送信しているが、トランザクションのコミットには失敗している可能性があります。ブローカは、コミットを受信するまで、そのトランザクションの通知を削除できません。

**問題を解決するには**

アプリケーション開発者に連絡し、コーディングエラーを修正します。

### *コンシューマがメッセージを通知しない*

**この問題の原因を確認するには**

アプリケーション開発者に連絡し、アプリケーションでシステムベースの通知が使用されているか、クライアントベースの通知が使用されているかを判断します。アプリケーションでシステムベースの通知が使用されている場合は、この節を省略してください。

アプリケーションでクライアントベースの通知 (`CLIENT_ACKNOWLEDGE type`) が使用されている場合は、まず、クライアントで保存されるメッセージの数を減らします。次のようなコマンドを使用します。

```
imqcmd update dst -n myDst -t q -o consumerFlowLimit=1
```

次に、コンシューマが遅いためにブローカがメッセージをバッファリングしている状態か、コンシューマがメッセージを高速に処理しているがメッセージを通知していない状態かを判断します。

次のコマンドを使用し、送信先を一覧表示します。

```
imqcmd list dst
```

ユーザー名とパスワードを入力したあとで、次のような出力が表示されます。

```
Listing all the services on the broker specified by:
--------------------------------
Host         Primary Port
--------------------------------
localhost    7676
-------------------------------------------------------------------------
   Name     Type    State    Producers  Consumers Msgs
                                        Total Count  UnAck  Avg Size
-------------------------------------------------------------------------
MyDest      Queue   RUNNING   0          0        5    200    1177.0
mq.sys.dmq  Queue   RUNNING   0          0       35      0    1422.0
Successfully listed destinations.
```

UnAck の数値は、ブローカが送信して通知を待機しているメッセージ数を表します。UnAck の数値が高いか上昇し続けている場合、ブローカはメッセージを送信しているので、遅いコンシューマを待機していません。コンシューマはメッセージを通知していないことになります。

**問題を解決するには**

アプリケーション開発者に連絡し、コーディングエラーを修正します。

### *永続コンシューマがアクティブにならない*

**この問題の原因を確認するには**

次のコマンド形式を使用し、トピックの永続サブスクリプションを調べます。

```
imqcmd list dur -d topicName
```

**問題を解決するには**

- `imqcmd purge dur` コマンドを使用し、永続コンシューマを消去します。
- コンシューマアプリケーションを再起動します。

### *予期しないブローカエラーが発生する*

**この問題の原因を確認するには**

263 ページの「プロデューサがコンシューマより速い」の説明に従い、QBrowser を使用してメッセージを調べます。

JMS_SUN_DMQ_UNDELIVERED_REASON の値が ERROR である場合は、ブローカエラーが発生しています。

**問題を解決するには**

- ブローカのログファイルを調べ、関連エラーを探します。
- Sun テクニカルサポートに連絡し、ブローカの問題について報告します。

第 III 部

# リファレンス

第 13 章

# コマンド行のリファレンス

この章では、Message Queue のコマンド行管理ユーティリティーの使い方に関する参照情報を提供します。この章は、次の節から構成されています。



## コマンド行の構文

Message Queue のコマンド行ユーティリティーはシェルコマンドです。ユーティリティーの名前はコマンドであり、そのサブコマンドやオプションは、そのコマンドに渡される引数です。ユーティリティーを開始または終了するためのコマンドは別途必要ありません。

コマンド行ユーティリティーはすべて、次のコマンド構文を共有します。

*utilityName* [*subcommand*] [*commandArgument*] [[*-optionName* [*-optionArgument*]]...]

*utilityName* は、次のいずれかになります。

- `imqbrokerd` ( ブローカユーティリティー )
- `imqcmd` ( コマンドユーティリティー )

- imqobjmgr ( オブジェクトマネージャーユーティリティー )
- imqdbmgr ( データベースマネージャーユーティリティー )
- imqusermgr ( ユーザーマネージャーユーティリティー )
- imqsvcadmin ( サービス管理ユーティリティー )
- imqkeytool ( キーツールユーティリティー )

サブコマンドおよびコマンドレベル引数を指定する場合は、すべてのオプションとそれらの引数の前に指定する必要があります。オプションは任意の順序で指定できます。サブコマンド、コマンドの引数、オプション、およびオプションの引数はすべて、スペースで区切ります。オプションの引数の値にスペースが含まれる場合は、値全体を引用符で囲む必要があります。属性と値の組み合わせは、通常、引用符で囲んでおきます。

デフォルトブローカを起動する次のコマンドは、サブコマンド節がないコマンド行の例です。

```
imqbrokerd
```

次のコマンドは、より多くの項目を含む例です。

```
imqcmd destroy dst -t q -n myQueue -u admin -f -s
```

このコマンドでは、myQueue という名前のキュー送信先 ( 送信先タイプ q) が破棄されます。認証はユーザー名 admin で実行され、コマンドによってパスワードが要求されます。確認の要求はされず (-f オプション )、出力を表示しないサイレントモードで実行されます (-s オプション )。

# ブローカユーティリティー

ブローカユーティリティー (imqbrokerd) では、ブローカを起動します。コマンド行オプションは、ブローカ設定ファイルの値をオーバーライドします。ただし、オーバーライドの対象は現在のブローカセッションだけです。

表 13-1 に、imqbrokerd コマンドのオプションと、各オプションによってオーバーライドされる設定プロパティー ( ある場合 ) を示します。

**表 13-1** ブローカユーティリティーのオプション

| オプション | オーバーライドされるプロパティー | 説明 |
|---|---|---|
| `-name` *instanceName* | `imq.instancename` | ブローカのインスタンス名<br><br>同じホスト上で実行される複数のブローカインスタンスには、異なるインスタンス名を割り当てる必要があります。<br><br>デフォルト値 : `imqbroker` |
| `-port` *portNumber* | `imq.portmapper.port` | ブローカのポートマッパーのポート番号<br><br>Message Queue クライアントは、このポート番号を使用してブローカに接続します。同じホスト上で実行される複数のブローカインスタンスには、異なるポートマッパーのポート番号を割り当てる必要があります。<br><br>デフォルト値 : `7676` |
| `-cluster` *broker1* [ [ , *broker2* ] ... ] | `imq.cluster.brokerlist` | ブローカを接続してクラスタを形成します[1]<br><br>指定したブローカは、`imq.cluster.brokerlist` プロパティーのリストとマージされます。各ブローカの引数は、次のいずれかの形式になります。<br><br>*hostName* : *portNumber*<br>*hostName*<br>: *portNumber*<br><br>*hostName* を省略した場合、デフォルト値は `localhost` になります。*portNumber* を省略した場合、デフォルト値は `7676` になります。 |
| `-D`*property*`=`*value* | インスタンス設定ファイル内の対応するプロパティー | 設定プロパティーを設定します<br><br>ブローカ設定プロパティーについては、第 14 章「ブローカのプロパティーのリファレンス」を参照してください。<br><br>**注 :** このオプションを使用してプロパティーを設定するときは、スペルと形式をよく確認してください。誤りのある値は無視され、通知や警告は表示されません。 |

**表 13-1** ブローカユーティリティーのオプション ( 続き )

| オプション | オーバーライドされるプロパティー | 説明 |
|---|---|---|
| `-reset props` | なし | 設定プロパティーをリセットします<br><br>ブローカの既存のインスタンス設定ファイル (`config.properties`) を空のファイルに置き換えます。すべてのプロパティーがデフォルト値になります。 |
| `-reset store` | なし | 持続データストアをリセットします<br><br>持続メッセージ、永続サブスクリプション、およびトランザクション情報を含むすべての持続データをデータストアから消去します。これにより、ブローカインスタンスを新規の状態で起動できます。その後の再起動時に持続ストアがリセットされないようにするには、`-reset` オプションを付けずにブローカインスタンスを再起動します。<br><br>持続メッセージまたは永続サブスクリプションだけを消去するには、代わりに `-reset messages` または `-reset durables` を使用します。 |
| `-reset messages` | なし | データストアから持続メッセージを消去します |
| `-reset durables` | なし | データストアから永続サブスクリプションを消去します |
| `-backup` *fileName* | なし | 設定変更レコードをファイルにバックアップします[1]<br><br>詳細については、191 ページの「設定変更レコードの管理」を参照してください。 |
| `-restore` *fileName* | なし | 設定変更レコードをバックアップファイルから復元します[1]<br><br>バックアップファイルは、`-backup` オプションを使用してあらかじめ作成しておく必要があります。<br><br>詳細については、191 ページの「設定変更レコードの管理」を参照してください。 |

表 13-1　ブローカユーティリティーのオプション ( 続き )

| オプション | オーバーライドされるプロパティー | 説明 |
|---|---|---|
| `-remove instance` | なし | ブローカインスタンスを削除します[2]<br>インスタンスに関連付けられたインスタンス設定ファイル、ログファイル、持続ストア、およびその他のファイルとディレクトリを削除します。 |
| `-password` *keyPassword* | `imq.keystore.password` | SSL 証明書キーストアのパスワード[3] |
| `-dbuser` *userName* | `imq.persist.jdbc.user` | JDBC ベースの持続データストアのユーザー名 |
| `-dbpassword` *dbPassword* | `imq.persist.jdbc.password` | JDBC ベースの持続データストアのパスワード[3] |
| `-ldappassword` *ldapPassword* | `imq.user_repository.ldap.password` | LDAP ユーザーリポジトリのパスワード[3] |
| `-passfile` *filePath* | `imq.passfile.enabled`<br>`imq.passfile.dirpath`<br>`imq.passfile.name` | パスワードファイルの場所<br>ブローカの `imq.passfile.enabled` プロパティーを `true` に設定し、`imq.passfile.dirpath` をパスワードファイルを含むパスに設定し、`imq.passfile.name` をファイル名自体に設定します。<br>詳細については、161 ページの「パスワードファイルの使用」を参照してください。 |
| `-shared` | `imq.jms.threadpool_model` | 共有スレッドプールモデルを使用して `jms` 接続サービスを実装します。<br>実行スレッドが接続間で共有されるため、サポートされる接続数が増えます。<br>ブローカの `imq.jms.threadpool_model` プロパティーを `shared` に設定します。 |
| `-javahome` *path* | なし | 代替 Java ランタイムの場所<br>デフォルトでは、システムにインストールされているランタイム、または Message Queue にバンドルされているランタイムが使用されます。 |

**表 13-1** ブローカユーティリティーのオプション ( 続き )

| オプション | オーバーライドされるプロパティー | 説明 |
|---|---|---|
| `-vmargs` *arg1* [ [ *arg2* ] ... ] | なし | Java 仮想マシンに引数を渡します<br><br>引数はスペースで区切ります。複数の引数またはスペースを含む引数を渡すには、引数のリストを引用符で囲みます。<br><br>VM 引数は、コマンド行からのみ渡すことができます。インスタンス設定ファイルには、関連する設定プロパティーはありません。 |
| `-license` [ *licenseName* ] | なし | インストール済み Message Queue 製品エディションのデフォルトとは異なるライセンスを使用する場合に読み込むライセンス<br><br>`pe` 基本機能を備えた Platform Edition<br><br>`try` 企業向け機能 (90 日間のトライアル ) を備えた Platform Edition<br><br>`unl` Enterprise Edition<br><br>ライセンス名を指定しない場合は、システムにインストールされているすべてのライセンスが一覧表示されます。 |
| `-upgrade-store-nobackup` | なし | 非互換バージョンから Message Queue 3.5 または 3.5 SP*x* にアップグレードするときに、古いデータストアを自動的に削除します[2]<br><br>詳細については、『Message Queue Installation Guide』を参照してください。 |
| `-force` | なし | ユーザーの確認なしでアクションを実行します<br><br>このオプションは、通常は確認が必要な `-remove instance` および `-upgrade-store-nobackup` オプションのみに適用されます。 |

**表 13-1** ブローカユーティリティーのオプション ( 続き )

| オプション | オーバーライドされるプロパティー | 説明 |
|---|---|---|
| `-loglevel` *level* | `imq.broker.log.level` | ロギングレベル<br>`NONE`<br>`ERROR`<br>`WARNING`<br>`INFO`<br>デフォルト値 : `INFO` |
| `-metrics` *interval* | `imq.metrics.interval` | ブローカのメトリックスのロギング間隔 ( 秒単位 ) |
| `-tty` | `imq.log.console.output` | コンソールにすべてのメッセージをログ出力します<br>ブローカの `imq.log.console.output` プロパティーを `ALL` に設定します。<br>指定しない場合は、エラーおよび警告メッセージだけがログ出力されます。 |
| `-s` \| `-silent` | `imq.log.console.output` | サイレントモード ( コンソールへのログ出力なし )<br>ブローカの `imq.log.console.output` プロパティーを `NONE` に設定します。 |
| `-version` | なし | バージョン情報を表示します[4] |
| `-h` \| `-help` | なし | 使用方法に関するヘルプを表示します[4] |

1. このオプションはブローカクラスタのみに適用されます。
2. このオプションでは、`-force` を一緒に指定しないかぎり、ユーザーの確認が求められます。
3. このオプションは異論が多く、最終的には削除される予定です。代わりに、パスワードを完全に省略して、インタラクティブにパスワードが要求されるようにするか、または `-passfile` オプションを使用して、パスワードが含まれるパスワードファイルを指定します。
4. コマンド行に指定したその他のオプションはすべて無視されます。

# コマンドユーティリティー

コマンドユーティリティー (imqcmd) は、ブローカ、接続サービス、接続、物理的送信先、永続サブスクリプション、およびトランザクションの管理に使用します。

製品のバージョン情報または使用方法に関するヘルプを表示するための -v または -h オプションを使用する場合を除き、すべての imqcmd コマンドでサブコマンドを指定する必要があります。使用可能なサブコマンドのリストを次に示し、それらの詳細を以下の対応する節で説明します。サブコマンドがブローカアドレス (-b オプション ) を受け付ける場合、ホスト名またはポート番号を指定しないときは常に、値はデフォルトで localhost および 7676 になります。

| | |
|---|---|
| **ブローカ管理** | |
| shutdown bkr | ブローカをシャットダウンします |
| restart bkr | ブローカを再起動します |
| pause bkr | ブローカを停止します |
| resume bkr | ブローカを再開します |
| update bkr | ブローカのプロパティーを設定します |
| reload cls | クラスタ設定を再読み込みします |
| query bkr | ブローカのプロパティー値を一覧表示します |
| metrics bkr | ブローカのメトリックスを表示します |
| **接続サービス管理** | |
| pause svc | 接続サービスを停止します |
| resume svc | 接続サービスを再開します |
| update svc | 接続サービスのプロパティーを設定します |
| list svc | ブローカで使用可能な接続サービスを一覧表示します |
| query svc | 接続サービスのプロパティー値を一覧表示します |
| metrics svc | 接続サービスのメトリックスを表示します |
| **接続管理** | |
| list cxn | ブローカ上の接続を一覧表示します |
| query cxn | 接続情報を表示します |

| | |
|---|---|
| **物理的送信先管理** | |
| `create dst` | 物理的送信先を作成します |
| `destroy dst` | 物理的送信先を破棄します |
| `pause dst` | 物理的送信先のメッセージ配信を停止します |
| `resume dst` | 物理的送信先のメッセージ配信を再開します |
| `update dst` | 物理的送信先のプロパティーを設定します |
| `purge dst` | 物理的送信先からすべてのメッセージを消去します |
| `compact dst` | 物理的送信先を圧縮します |
| `list dst` | 物理的送信先を一覧表示します |
| `query dst` | 物理的送信先のプロパティー値を一覧表示します |
| `metrics dst` | 物理的送信先のメトリックスを表示します |
| **永続サブスクリプション管理** | |
| `destroy dur` | 永続サブスクリプションを破棄します |
| `purge dur` | 永続サブスクリプションのすべてのメッセージを消去します |
| `list dur` | トピックの永続サブスクリプションを一覧表示します |
| **トランザクション管理** | |
| `commit txn` | トランザクションをコミットします |
| `rollback txn` | トランザクションをロールバックします |
| `list txn` | ブローカが追跡しているトランザクションを一覧表示します |
| `query txn` | トランザクション情報を表示します |

# ブローカ管理

コマンドユーティリティーではブローカを起動できません。代わりにブローカユーティリティー (imqbrokerd) を使用します。ブローカの起動後は、表 13-2 に示す imqcmd サブコマンドを使用して、ブローカを管理および制御できます。

**表 13-2** ブローカ管理のためのコマンドユーティリティーサブコマンド

| 構文 | 説明 |
|---|---|
| `shutdown bkr` [`-b` *hostName*`:`*portNumber*] | ブローカをシャットダウンします |
| `restart bkr` [`-b` *hostName*`:`*portNumber*] | ブローカを再起動します<br>ブローカをシャットダウンしてから、そのブローカの起動時に指定されたオプションを使用して再起動します。 |
| `pause bkr` [`-b` *hostName*`:`*portNumber*] | ブローカを停止します<br>詳細については、105 ページの「ブローカを停止する」を参照してください。 |
| `resume bkr` [`-b` *hostName*`:`*portNumber*] | ブローカを再開します |
| `update bkr` [`-b` *hostName*`:`*portNumber*] `-o` *property1=value1* [ [`-o` *property2=value2*] ... ] | ブローカのプロパティーを設定します<br>ブローカのプロパティーについては、第 14 章「ブローカのプロパティーのリファレンス」を参照してください。 |
| `reload cls` | クラスタ設定を再読み込みします[1]<br>すべての持続情報を最新の状態にします。 |
| `query bkr -b` *hostName*`:`*portNumber* | ブローカのプロパティー値を一覧表示します<br>クラスタでは、指定されたブローカに接続している実行中のすべてのブローカも一覧表示します。 |

表 13-2　ブローカ管理のためのコマンドユーティリティーサブコマンド ( 続き )

| 構文 | 説明 |
|---|---|
| `metrics bkr` [`-b` *hostName*:*portNumber*]<br>[`-m` *metricType*]<br>[`-int` *interval*]<br>[`-msp` *numSamples*] | ブローカのメトリックスを表示します<br>`-m` オプションでは、表示するメトリックスのタイプを指定します。<br>`ttl` ブローカで送受信されているメッセージとパケット<br>`rts` ブローカで送受信されているメッセージとパケットのフローレート ( 秒単位 )<br>`cxn` 接続、仮想メモリーヒープ、およびスレッド<br>デフォルト値は `ttl` です。<br>`-int` オプションでは、メトリックスを表示する間隔を秒単位で指定します。デフォルト値は `5` です。<br>`-msp` オプションでは、表示するサンプル数を指定します。デフォルト値は、無制限 ( 無限 ) です。 |

1. このオプションはブローカクラスタのみに適用されます。

# 接続サービス管理

表 13-3 に、接続サービスを管理するための `imqcmd` サブコマンドを示します。

表 13-3　接続サービス管理のためのコマンドユーティリティーサブコマンド

| 構文 | 説明 |
|---|---|
| `pause svc -n` *serviceName*<br>[`-b` *hostName*:*portNumber*] | 接続サービスを停止します<br>`admin` 接続サービスは停止できません。 |
| `resume svc -n` *serviceName*<br>[`-b` *hostName*:*portNumber*] | 接続サービスを再開します |
| `update svc -n` *serviceName*<br>[`-b` *hostName*:*portNumber*]<br>`-o` *property1=value1*<br>[ [`-o` *property2=value2*] ... ] | 接続サービスのプロパティーを設定します<br>接続サービスのプロパティーについては、295 ページの「接続のプロパティー」を参照してください。 |

表 13-3　接続サービス管理のためのコマンドユーティリティーサブコマンド ( 続き )

| 構文 | 説明 |
|---|---|
| `list svc` [`-b` *hostName*:*portNumber*] | ブローカで使用可能な接続サービスを一覧表示します |
| `query svc -n` *serviceName* [`-b` *hostName*:*portNumber*] | 接続サービスのプロパティー値を一覧表示します |
| `metrics svc -n` *serviceName* [`-b` *hostName*:*portNumber*] [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*] | 接続サービスのメトリックスを表示します<br>`-m` オプションでは、表示するメトリックスのタイプを指定します。<br>`ttl` 指定の接続サービスを介してブローカで送受信されているメッセージとパケット<br>`rts` 指定の接続サービスを介してブローカで送受信されているメッセージとパケットのフローレート ( 秒単位 )<br>`cxn` 接続、仮想メモリーヒープ、およびスレッド<br>デフォルト値は `ttl` です。<br>`-int` オプションでは、メトリックスを表示する間隔を秒単位で指定します。デフォルト値は `5` です。<br>`-msp` オプションでは、表示するサンプル数を指定します。デフォルト値は、無制限 ( 無限 ) です。 |

# 接続管理

表 13-4 に、接続を管理するための imqcmd サブコマンドを示します。

**表 13-4** 接続サービス管理のためのコマンドユーティリティーサブコマンド

| 構文 | 説明 |
|---|---|
| `list cxn` [`-svn` *serviceName*] [`-b` *hostName*`:`*portNumber*] | ブローカ上の接続を一覧表示します<br>指定された接続サービスに対する、ブローカ上のすべての接続を一覧表示します。接続サービスを指定しない場合は、すべての接続が一覧表示されます。 |
| `query cxn -n` *connectionID* [`-b` *hostName*`:`*portNumber*] | 接続情報を表示します |

# 物理的送信先管理

表 13-5 に、物理的送信先を管理するための imqcmd サブコマンドを示します。-t ( 送信先タイプ ) オプションには常に、2 つの値のいずれかを指定できます。

```
q   キュー送信先
t   トピック送信先
```

**表 13-5** 物理的送信先管理のためのコマンドユーティリティーサブコマンド

| 構文 | 説明 |
|---|---|
| `create dst -t` *destType* `-n` *destName* [`-o` *property1*`=`*value1*] [ [`-o` *property2*`=`*value2*] ... ] | 物理的送信先を作成します[1]<br>送信先名 *destName* は、スペースを含まない英数字だけを使用でき、英字、下線 (_)、またはドル記号 ($) 文字で始める必要があります。文字列 mq で始めることはできません。 |
| `destroy dst -t` *destType* `-n` *destName* | 物理的送信先を破棄します[1]<br>この操作は、デッドメッセージキューなど、システムが作成した送信先には適用できません。 |

表 13-5　物理的送信先管理のためのコマンドユーティリティーサブコマンド ( 続き )

| 構文 | 説明 |
|---|---|
| pause dst [-t *destType* -n *destName*]<br>[-pst *pauseType*] | 物理的送信先のメッセージ配信を停止します<br><br>-t および -n オプションで指定された物理的送信先のメッセージ配信を停止します。これらのオプションを指定しない場合は、すべての送信先が停止されます。<br><br>pst オプションでは、停止するメッセージ配信のタイプを指定します。<br><br>CONSUMERS　メッセージコンシューマへの配信を停止します<br><br>PRODUCERS　メッセージプロデューサへの配信を停止します<br><br>ALL　すべてのメッセージ配信を停止します<br><br>デフォルト値 : ALL |
| resume dst [-t *destType* -n *destName*] | 物理的送信先のメッセージ配信を再開します<br><br>-t および -n オプションで指定された物理的送信先のメッセージ配信を再開します。これらのオプションを指定しない場合は、すべての送信先が再開されます。 |
| update dst -t *destType* -n *destName*<br>-o *property1=value1*<br>[ [-o *property2=value2*] ... ] | 物理的送信先のプロパティーを設定します<br><br>物理的送信先のプロパティーについては、第 15 章「物理的送信先のプロパティーのリファレンス」を参照してください。 |
| purge dst -t *destType* -n *destName* | 物理的送信先からすべてのメッセージを消去します |
| compact dst [-t *destType* -n *destName*] | 物理的送信先を圧縮します<br><br>-t および -n オプションで指定された物理的送信先のファイルベースの持続データストアを圧縮します。これらのオプションを指定しない場合は、すべての送信先が圧縮されます。<br><br>圧縮する前に、対象の送信先を停止する必要があります。 |

表 13-5　物理的送信先管理のためのコマンドユーティリティーサブコマンド ( 続き )

| 構文 | 説明 |
|---|---|
| `list dst` [`-t` *destType*] [`-tmp`] | 物理的送信先を一覧表示します<br>`-t` オプションで指定されたタイプのすべての物理的送信先を一覧表示します。送信先タイプを指定しない場合は、キューとトピックの両方の送信先が一覧表示されます。`-tmp` オプションを指定した場合は、一時的送信先も一覧表示されます。 |
| `query dst -t` *destType* `-n` *destName* | 物理的送信先のプロパティー値を一覧表示します |
| `metrics dst -t` *destType* `-n` *destName* [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*] | 物理的送信先のメトリックスを表示します<br>`-m` オプションでは、表示するメトリックスのタイプを指定します。<br>`ttl` 送信先で送受信されているメッセージとパケットおよびメモリー内のメッセージとパケット<br>`rts` ブローカで送受信されているメッセージとパケットのフローレート ( 秒単位 )、およびその他のレート情報<br>`con` メッセージコンシューマに関するメトリックス<br>`dsk` ディスク使用量<br>デフォルト値は `ttl` です。<br>`-int` オプションでは、メトリックスを表示する間隔を秒単位で指定します。デフォルト値は `5` です。<br>`-msp` オプションでは、表示するサンプル数を指定します。デフォルト値は、無制限 ( 無限 ) です。 |

1. この操作は、マスターブローカが一時的に使用できなくなっているブローカクラスタでは実行できません。

# 永続サブスクリプション管理

表 13-6 に、永続サブスクリプションを管理するための imqcmd サブコマンドを示します。

表 13-6　永続サブスクリプション管理のためのコマンドユーティリティーサブコマンド

| 構文 | 説明 |
|---|---|
| `destroy dur -c` *clientID* `-n` *subscriberName* | 永続サブスクリプションを破棄します[1] |
| `purge dur -c` *clientID* `-n` *subscriberName* | 永続サブスクリプションのすべてのメッセージを消去します |
| `list dur -d` *topicName* | トピックの永続サブスクリプションを一覧表示します |

1. この操作は、マスターブローカが一時的に使用できなくなっているブローカクラスタでは実行できません。

# トランザクション管理

表 13-7 に、トランザクションを管理するための imqcmd サブコマンドを示します。

表 13-7　トランザクション管理のためのコマンドユーティリティーサブコマンド

| 構文 | 説明 |
|---|---|
| `commit txn -n` *transactionID* | トランザクションをコミットします |
| `rollback txn -n` *transactionID* | トランザクションをロールバックします |
| `list txn` | ブローカが追跡しているトランザクションを一覧表示します |
| `query txn -n` *transactionID* | トランザクション情報を表示します |

# 一般的なコマンドユーティリティーオプション

表 13-8 に示す追加のオプションは、imqcmd コマンドのすべてのサブコマンドに適用できます。

表 13-8　一般的なコマンドユーティリティーオプション

| オプション | 説明 |
|---|---|
| `-secure` | `ssladmin` 接続サービスによるブローカへの安全な接続を使用します |
| `-u` *userName* | 認証のためのユーザー名<br>このオプションを省略すると、コマンドユーティリティーによってインタラクティブに要求されます。 |
| `-p` *password* | 認証のためのパスワード[1] |
| `-passfile` *path* | パスワードファイルの場所<br>詳細については、161 ページの「パスワードファイルの使用」を参照してください。 |
| `-rtm` *timeoutInterval* | 初期タイムアウト間隔 ( 秒単位 )<br>これは、コマンドユーティリティーが要求を再試行するまでの、ブローカからの応答を待つ時間の初期値です。その後の各再試行では、タイムアウト間隔として、この初期間隔の倍数が使用されます。<br>デフォルト値は `10` です。 |
| `-rtr` *numRetries* | ブローカ要求がタイムアウトになったあとの再試行の回数<br>デフォルト値は `5` です。 |
| `-javahome` *path* | 代替 Java ランタイムの場所<br>デフォルトでは、システムにインストールされているランタイム、または Message Queue にバンドルされているランタイムが使用されます。 |
| `-f` | ユーザーの確認なしでアクションを実行します |
| `-s` | サイレントモード ( 出力の表示なし ) |
| `-v` | バージョン情報を表示します[2,3] |
| `-h` | 使用方法に関するヘルプを表示します[2、3] |
| `-H` | 属性リストや例を含む、使用方法に関する詳細なヘルプを表示します[2、3] |

1. このオプションは異論が多く、最終的には削除される予定です。代わりに、パスワードを完全に省略して、インタラクティブにパスワードが要求されるようにするか、または `-passfile` オプションを使用して、パスワードが含まれるパスワードファイルを指定します。
2. コマンド行に指定したその他のオプションはすべて無視されます。
3. このオプションでは、ユーザー名とパスワードは必要ありません。

# オブジェクトマネージャーユーティリティー

オブジェクトマネージャーユーティリティー (imqobjmgr) では、Message Queue 管理対象オブジェクトを作成および管理します。表 13-9 に、使用可能なサブコマンドを示します。

表 13-9　オブジェクトマネージャーのサブコマンド

| サブコマンド | 説明 |
|---|---|
| add | 管理対象オブジェクトをオブジェクトストアに追加します |
| delete | 管理対象オブジェクトをオブジェクトストアから削除します |
| list | オブジェクトストア内の管理対象オブジェクトを一覧表示します |
| query | 管理対象オブジェクトの情報を表示します |
| update | 管理対象オブジェクトを変更します |

表 13-10 に、imqobjmgr コマンドのオプションを示します。

表 13-10　オブジェクトマネージャーのオプション

| オプション | 説明 |
|---|---|
| -l *lookupName* | 管理対象オブジェクトの JNDI 検索名 |
| -j *attribute=value* | JNDI オブジェクトストアの属性 (165 ページの「オブジェクトストア」を参照 ) |
| -t *objectType* | 管理対象オブジェクトのタイプ :<br>q　キュー送信先<br>t　トピック送信先<br>cf　接続ファクトリ<br>qf　キュー接続ファクトリ<br>tf　トピック接続ファクトリ<br>xcf　分散トランザクションの接続ファクトリ<br>xqf　分散トランザクションのキュー接続ファクトリ<br>xtf　分散トランザクションのトピック接続ファクトリ<br>e　SOAP の端点 (『Message Queue Developer's Guide for Java Clients』を参照 ) |

表 13-10　オブジェクトマネージャーのオプション ( 続き )

| オプション | 説明 |
|---|---|
| -o *attribute=value* | 管理対象オブジェクトの属性 (169 ページの「管理対象オブジェクトの属性」および第 16 章「管理対象オブジェクト属性のリファレンス」を参照 ) |
| -r *readOnlyState* | 管理対象オブジェクトが読み取り専用かどうか<br>true の場合、クライアントはオブジェクトの属性を変更できません。デフォルト値は false です。 |
| -i *fileName* | サブコマンド節の全体または一部を含むコマンドファイルの名前 |
| -pre | コマンドを実行せずに結果のプレビューを表示します<br>このオプションは、デフォルト属性の値を確認するときに便利です。 |
| -javahome *path* | 代替 Java ランタイムの場所<br>デフォルトでは、システムにインストールされているランタイム、または Message Queue にバンドルされているランタイムが使用されます。 |
| -f | ユーザーの確認なしでアクションを実行します |
| -s | サイレントモード ( 出力の表示なし ) |
| -v | バージョン情報を表示します[1] |
| -h | 使用方法に関するヘルプを表示します[1] |
| -H | 属性リストや例を含む、使用方法に関する詳細なヘルプを表示します[1] |

1. コマンド行に指定したその他のオプションはすべて無視されます。

# データベースマネージャーユーティリティー

データベースマネージャーユーティリティー (imqdbmgr) では、JDBC ベースの持続データストアのデータベーススキーマを設定します。また、破損した Message Queue データベーステーブルを削除したり、データストアを変更したりすることもできます。表 13-11 に、使用可能なサブコマンドを示します。

**表 13-11** データベースマネージャーのサブコマンド

| サブコマンド | 説明 |
|---|---|
| `create all` | 新しいデータベースと持続ストアのスキーマを作成します<br>組み込みデータベースシステムに使用します。ブローカプロパティー `imq.persist.jdbc.createdburl` を指定する必要があります。 |
| `create tbl` | 既存のデータベースの持続ストアのスキーマを作成します<br>外部データベースシステムに使用します。 |
| `delete tbl` | 現在の持続ストアから Message Queue データベーステーブルを削除します |
| `delete oldtbl` | 以前のバージョンの持続ストアから Message Queue データベーステーブルを削除します<br>持続ストアが Message Queue の現在のバージョンへ自動的に移行されたあとに使用されます。 |
| `recreate tbl` | 持続ストアのスキーマを再作成します<br>現在の持続ストアから既存の Message Queue データベースをすべて削除したあと、スキーマを再作成します。 |
| `reset lck` | 持続ストアのロックをリセットします<br>ほかのプロセスが持続ストアデータベースを使用できるようにロックをリセットします。 |

表 13-12 に、`imqdbmgr` コマンドのオプションを示します。

**表 13-12** データベースマネージャーのオプション

| オプション | 説明 |
|---|---|
| `-b` *instanceName* | ブローカのインスタンス名 |

表 13-12　データベースマネージャーのオプション ( 続き )

| オプション | 説明 |
|---|---|
| -D*property=value* | ブローカ設定プロパティーを設定します<br><br>持続に関連するブローカ設定プロパティーについては、303 ページの「持続のプロパティー」を参照してください。<br><br>**注 :** このオプションを使用してプロパティーを設定するときは、スペルと形式をよく確認してください。誤りのある値は無視され、通知や警告は表示されません。 |
| -u *name* | 認証のためのユーザー名 |
| -p *password* | 認証のためのパスワード[1] |
| -passfile *path* | パスワードファイルの場所<br><br>詳細については、161 ページの「パスワードファイルの使用」を参照してください。 |
| -v | バージョン情報を表示します[2] |
| -h | 使用方法に関するヘルプを表示します[2] |

1. このオプションは異論が多く、最終的には削除される予定です。代わりに、パスワードを完全に省略して、インタラクティブにパスワードが要求されるようにするか、または -passfile オプションを使用して、パスワードが含まれるパスワードファイルを指定します。
2. コマンド行に指定したその他のオプションはすべて無視されます。

# ユーザーマネージャーユーティリティー

ユーザーマネージャーユーティリティー (imqusermgr) は、単層型ファイルユーザーリポジトリを設定または編集するために使用します。このユーティリティーは、対象のブローカがインストールされているホストで実行する必要があります。ブローカ固有のユーザーリポジトリがまだ存在しない場合は、最初に、対応するブローカインスタンスを起動してリポジトリを作成する必要があります。また、リポジトリに書き込むための適切なアクセス権が必要になります。つまり、Solaris および Linux プラットフォームでは、root ユーザーか、対象のブローカインスタンスを作成したユーザーである必要があります。

表 13-13 に、imqusermgr コマンドで使用可能なサブコマンドを示します。-i オプションでは常に、コマンドの適用対象となるユーザーリポジトリを持つブローカのインスタンス名を指定します。指定しない場合は、デフォルト名 imqbroker が使用されます。

表 13-13　ユーザーマネージャーのサブコマンド

| 構文 | 説明 |
|---|---|
| `add` [`-i` *instanceName*] `-u` *userName* `-p` *password* [`-g` *group*] | ユーザーとパスワードをリポジトリに追加します<br>任意指定の `-g` オプションでは、このユーザーを割り当てるグループを指定します。<br>`admin`<br>`user`<br>`anonymous` |
| `delete` [`-i` *instanceName*] `-u` *userName* | ユーザーをリポジトリから削除します |
| `update` [`-i` *instanceName*] `-u` *userName* `-p` *password* [`-a` *activeState*]<br>`update` [`-i` *instanceName*] `-u` *userName* `-a` *activeState* [`-p` *password*] | ユーザーのパスワードまたはアクティブ状態 ( あるいはその両方 ) を設定します<br>`-a` オプションでは、ユーザーをアクティブにする (`true`) か非アクティブにする (`false`) かのブール値を指定します。デフォルト値は `true` です。 |
| `list` [`-i` *instanceName*] [`-u` *userName*] | ユーザー情報を表示します<br>ユーザー名を指定しない場合は、リポジトリ内のすべてのユーザーが一覧表示されます。 |

また、表 13-14 に示すオプションは、`imqusermgr` コマンドのすべてのサブコマンドに適用できます。

表 13-14　一般的なユーザーマネージャーオプション

| オプション | 説明 |
|---|---|
| `-f` | ユーザーの確認なしでアクションを実行します。 |
| `-s` | サイレントモード ( 出力の表示なし ) |
| `-v` | バージョン情報を表示します[1] |
| `-h` | 使用方法に関するヘルプを表示します[1] |

1. コマンド行に指定したその他のオプションはすべて無視されます。

# サービス管理ユーティリティー

サービス管理ユーティリティー (imqsvcadmin) では、ブローカを Windows サービスとしてインストールします。表 13-15 に、使用可能なサブコマンドを示します。

表 13-15　サービス管理のサブコマンド

| サブコマンド | 説明 |
|---|---|
| install | サービスをインストールします |
| remove | サービスを削除します |
| query | 起動オプションを表示します<br>起動オプションでは、サービスを手動または自動のどちらで起動するか、サービスの場所、Java ランタイムの場所、および起動時にブローカに渡す引数の値を指定できます。 |

表 13-16 に、imqsvcadmin コマンドのオプションを示します。

表 13-16　サービス管理のオプション

| オプション | 説明 |
|---|---|
| -javahome *path* | 代替 Java ランタイムの場所<br>デフォルトでは、システムにインストールされているランタイム、または Message Queue にバンドルされているランタイムが使用されます。 |
| -jrehome *path* | 代替 Java Runtime Environment (JRE) の場所 |
| -vmargs *arg1* [ [*arg2*] ... ] | ブローカサービスを実行する Java 仮想マシンに渡す追加の引数[1]<br>例 :<br>`imqsvcadmin install -vmargs "-Xms16m -Xmx128m"` |
| -args *arg1* [ [*arg2*] ... ] | ブローカサービスに渡す追加のコマンド行引数[1]<br>例 :<br>`imqsvcadmin install -args "-passfile d:¥imqpassfile"`<br>ブローカのコマンド行引数については、272 ページの「ブローカユーティリティー」を参照してください。 |
| -h | 使用方法に関するヘルプを表示します[2] |

1. これらの引数は、Windows の「管理ツール」コントロールパネルの「サービス」ツールで、サービスの「プロパティ」ウィンドウの「全般」タブにある「開始パラメータ」フィールドを使用して指定することもできます。
2. コマンド行に指定したその他のオプションはすべて無視されます。

-javahome、-vmargs、および -args オプションを使用して指定した情報は、Windows レジストリの次のパスにある JREHome、JVMArgs、および ServiceArgs キーに保存されます。

```
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥iMQ_Broker¥Parameters
```

# キーツールユーティリティー

キーツールユーティリティー (imqkeytool) では、ssljms、ssladmin、または cluster 接続サービスで使用できる、ブローカの自己署名型証明書を生成します。構文は次のとおりです。

```
imqkeytool -broker
```

UNIX システムでは、superuser (root) アカウントからこのユーティリティーを実行する必要がある場合があります。

第 14 章

# ブローカのプロパティーのリファレンス

この章では、メッセージブローカの設定プロパティーに関する参照情報を提供します。この章は、次の節から構成されています。



## 接続のプロパティー

表 14-1 に、接続サービスに関するブローカプロパティーを示します。

**表 14-1** 接続に関するブローカプロパティー

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| `imq.service.activelist` | 文字列 | `jms,admin` | ブローカの起動時にアクティブにする接続サービスのコンマ区切りのリスト |
| `imq.hostname` | 文字列 | 使用可能なすべての IP アドレス | すべての接続サービスのデフォルトのホスト名または IP アドレス |
| `imq.portmapper.hostname` | 文字列 | なし | ポートマッパーのホスト名または IP アドレス<br><br>指定すると、`imq.hostname` がオーバーライドされます |

**表 14-1** 接続に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.portmapper.port[1] | 整数 | 7676 | ポートマッパーのポート番号<br><br>**注**：同じホスト上で複数のブローカインスタンスが実行されている場合は、それぞれに一意のポートマッパーポートを割り当てる必要があります。 |
| imq.*serviceName*.*protocolType*.hostname | 文字列 | なし | 接続サービスのホスト名または IP アドレス[2]<br><br>指定すると、指定した接続サービスについて imq.hostname がオーバーライドされます |
| imq.*serviceName*.*protocolType*.port | 整数 | 0 | 接続サービスのポート番号[2]<br><br>値 0 は、ポート番号がポートマッパーによって動的に割り当てられることを指定します。 |
| imq.portmapper.backlog | 整数 | 50 | オペレーティングシステムのバックログに保留するポートマッパー要求の最大数 |
| imq.*serviceName*.threadpool_model | 文字列 | dedicated | スレッドプール管理のスレッドモデル<br><br>dedicated　接続ごとに、受信メッセージ用と送信メッセージ用の 2 つの専用スレッドを使用します<br><br>shared[3]　メッセージの送受信時に共有スレッドによって接続を処理します<br><br>専用モデルでは、サポートできる接続数が制限されますが、プロバイダのパフォーマンスが向上します。共有モデルでは、サポートできる接続数が多くなりますが、スレッド管理のためのオーバーヘッドが加わるため、パフォーマンスが低下します。 |

**表 14-1** 接続に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.*serviceName*.min_threads | 整数 | jms: 10<br>ssljms: 10<br>httpjms: 10<br>httpsjms: 10<br>admin: 4<br>ssladmin: 4 | 接続サービスのスレッドプールに保持するスレッドの最小数<br>使用可能なスレッドの数がこのしきい値を超えると、スレッドは、最小数に達するまで、解放と同時にとシャットダウンされます。<br>デフォルト値は、接続サービスによって異なります。 |
| imq.*serviceName*.max_threads | 整数 | jms: 1000<br>ssljms: 500<br>httpjms: 500<br>httpsjms: 500<br>admin: 10<br>ssladmin: 10 | 指定された接続サービスのスレッドプールに保持するスレッドの最大数。使用可能な新しいスレッドはこれ以上追加されません。この数は、0 より大きく、min_threads の値よりも大きくする必要があります。<br>デフォルト値は、接続サービスによって異なります。 |
| imq.shared.connectionMonitor_limit | 整数 | Solaris: 512<br>Linux: 512<br>Windows: 64 | 1 つのディストリビュータスレッドが監視する接続の最大数[4]<br>システムによって、すべての接続の監視に十分な数のディストリビュータスレッドが割り当てられます。このプロパティーの値が小さいほど、アクティブな接続へのスレッドの割り当てが高速になります。値 -1 は、スレッドあたりの接続数が無制限であることを示します。<br>デフォルト値は、オペレーティングシステムのプラットフォームによって異なります。 |
| imq.ping.interval | 整数 | 120 | クライアントとブローカ間のテスト接続の間隔 ( 秒単位 )<br>値 0 または -1 を指定すると、接続の定期的なテストが無効になります。 |

1. imqcmd update bkr コマンドで使用できます
2. jms、ssljms、admin、および ssladmin サービスのみ。httpjms および httpsjms サービスの設定については、付録 C「HTTP/HTTPS のサポート」を参照してください
3. jms および admin サービスのみ
4. 共有スレッドモデルのみ

# ルーティングのプロパティー

表 14-2 に、ルーティングサービスに関するブローカプロパティーを示します。送信先の自動作成を設定するプロパティーについては、表 14-3 を参照してください。

**表 14-2** ルーティングに関するブローカプロパティー

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.system.max_count[1] | 整数 | -1 | ブローカが保持するメッセージの最大数<br>値 -1 は、メッセージ数が無制限であることを示します。 |
| imq.system.max_size[1] | 文字列 | -1 | ブローカが保持するメッセージの最大合計サイズ<br>次の接尾辞を使用して、バイト、K バイト、または M バイトの単位で値を表すことができます。<br>b バイト<br>k K バイト (1024 バイト )<br>m M バイト (1024 x 1024 = 1,048,576 バイト )<br>接尾辞を付けない値は、バイト単位になります。値 -1 は、メッセージ容量が無制限であることを示します。<br>例 :<br>1600 1600 バイト<br>1600b 1600 バイト<br>16k 16K バイト (= 16,384 バイト )<br>16m 16M バイト (= 16,777,216 バイト )<br>-1 無制限 |
| imq.message.max_size[1] | 文字列 | 70m | 単一メッセージの本文の最大サイズ<br>構文は imq.system.max_size と同じです ( 前述の説明を参照 )。 |
| imq.message.expiration.interval | 整数 | 60 | 期限切れのメッセージを再利用する間隔 ( 秒単位 ) |

**表 14-2** ルーティングに関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.*resourceState*.threshold | 整数 | green:<br>0<br>yellow:<br>80<br>orange:<br>90<br>red:<br>98 | メモリーリソースの状態をトリガーするしきい値となる使用率 (*resourceState* は green、yellow、orange、または red) |
| imq.*resourceState*.count | 整数 | green:<br>5000<br>yellow:<br>500<br>orange:<br>50<br>red:<br>0 | メモリーリソースの状態のしきい値に達したかどうかを確認するまでの、バッチで許容する受信メッセージの最大数 (*resourceState* は green、yellow、orange、または red)<br>この制限は、システムメモリーがさらに不十分になると、メッセージプロデューサの処理速度を低下させます。 |
| imq.destination.DMQ.truncateBody[1] | ブール | false | デッドメッセージキューに保存する前にメッセージ本文を削除するかどうか<br>true の場合は、メッセージヘッダーとプロパティーデータのみが保存されます。 |
| imq.transaction.autorollback | ブール | false | ブローカの起動時に PREPARED 状態のままになっている分散トランザクションを自動的にロールバックするかどうか<br>false の場合は、コマンドユーティリティー (imqcmd) を使用して、トランザクションを手動でコミットまたはロールバックする必要があります。 |

1. imqcmd update bkr コマンドで使用できます

**表 14-3** 自動作成された送信先に関するブローカプロパティー

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.autocreate.queue[1,2] | ブール | true | キュー送信先の自動作成を許可するかどうか |

**表 14-3** 自動作成された送信先に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| `imq.autocreate.topic`[3] | ブール | `true` | トピック送信先の自動作成を許可するかどうか |
| `imq.autocreate.destination.maxNumMsgs` | 整数 | `100000` | 消費されていないメッセージの最大数<br><br>値 `-1` は、メッセージ数が無制限であることを示します。 |
| `imq.autocreate.destination.maxBytesPerMsg` | 文字列 | `10k` | 単一メッセージの最大サイズ ( バイト単位 )<br><br>次の接尾辞を使用して、バイト、K バイト、または M バイトの単位で値を表すことができます。<br><br>`b` バイト<br>`k` K バイト (1024 バイト )<br>`m` M バイト (1024 x 1024 = 1,048,576 バイト )<br><br>接尾辞を付けない値は、バイト単位になります。値 `-1` は、メッセージサイズが無制限であることを示します。<br><br>例 :<br><br>`1600` 1600 バイト<br>`1600b` 1600 バイト<br>`16k` 16K バイト (= 16,384 バイト )<br>`16m` 16M バイト (= 16,777,216 バイト )<br>`-1` 無制限 |
| `imq.autocreate.destination.maxTotalMsgBytes` | 文字列 | `10m` | 消費されていないメッセージの最大合計メモリー ( バイト単位 )<br><br>構文は `imq.autocreate.destination.maxBytesPerMsg` と同じです ( 前述の説明を参照 )。 |

**表 14-3** 自動作成された送信先に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.autocreate.destination.limitBehavior | 文字列 | REJECT_NEWEST | メモリー制限のしきい値に達したときのブローカの動作<br>FLOW_CONTROL<br>プロデューサの処理速度を下げます<br>REMOVE_OLDEST<br>もっとも古いメッセージを破棄します<br>REMOVE_LOW_PRIORITY<br>メッセージの有効期間に従ってもっとも優先度の低いメッセージを破棄します。プロデューシングクライアントには通知しません<br>REJECT_NEWEST<br>もっとも新しいメッセージを拒否します。持続メッセージの場合のみ、プロデューシングクライアントに例外を通知します<br>値が REMOVE_OLDEST または REMOVE_LOW_PRIORITY で、imq.autocreate.destination.useDMQ プロパティーが true の場合、超過したメッセージはデッドメッセージキューに移動されます。 |
| imq.autocreate.destination.maxNumProducers | 整数 | 100 | 送信先のメッセージプロデューサの最大数<br>この制限に達すると、新しいプロデューサを作成できません。値 -1 は、プロデューサ数が無制限であることを示します。 |
| imq.autocreate.queue.maxNumActiveConsumers[2] | 整数 | 1 | キュー送信先からのロードバランスされた配信でアクティブにするメッセージコンシューマの最大数<br>値 -1 は、コンシューマ数が無制限であることを示します。 |

**表 14-3** 自動作成された送信先に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.autocreate.queue.maxNumBackupConsumers[2] | 整数 | 0 | キュー送信先からのロードバランスされた配信でバックアップにするメッセージコンシューマの最大数<br><br>値 -1 は、コンシューマ数が無制限であることを示します。 |
| imq.autocreate.queue.consumerFlowLimit[2] | 整数 | 1000 | キューコンシューマに単一のバッチで配信するメッセージの最大数<br><br>ロードバランスされたキュー配信では、ロードバランスが開始されるまでの、アクティブコンシューマにルーティングされるキュー内のメッセージの初期数になります。送信先コンシューマは、接続で低い値を指定することで、この制限をオーバーライドできます。<br><br>値 -1 は、コンシューマ数が無制限であることを示します。 |
| imq.autocreate.topic.consumerFlowLimit[3] | 整数 | 1000 | トピックコンシューマに単一のバッチで配信するメッセージの最大数<br><br>値 -1 は、コンシューマ数が無制限であることを示します。 |
| imq.autocreate.destination.isLocalOnly | ブール | false | ローカル配信のみかどうか<br><br>このプロパティーは、ブローカクラスタ内の送信先のみに適用されます。送信先の作成後は変更できません。true の場合、送信先はほかのブローカに複製されず、ローカルコンシューマ ( 送信先が作成されたブローカに接続しているコンシューマ ) だけにメッセージを配信するように制限されます。 |

表 14-3　自動作成された送信先に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.autocreate.queue.localDeliveryPreferred[2] | ブール | false | ローカル配信優先かどうか<br>このプロパティーは、ブローカクラスタ内のロードバランスされたキュー配信のみに適用されます。true の場合、メッセージは、ローカルブローカにコンシューマがない場合にのみリモートコンシューマに配信されます。送信先をローカルのみの配信に制限しないでください。つまり、imq.autocreate.destination.isLocalOnly を false にする必要があります。 |
| imq.autocreate.destination.useDMQ | ブール | true | デッドメッセージをデッドメッセージキューに送信するかどうか<br>false の場合、デッドメッセージは単に破棄されます。 |

1. imqcmd update bkr コマンドで使用できます
2. キュー送信先のみ
3. トピック送信先のみ

# 持続のプロパティー

Message Queue は、持続データストレージにファイルベースモデルと JDBC ベースモデルの両方をサポートしています。ブローカプロパティー imq.persist.store ( 表 14-4) で、使用するモデルを指定します。以降の節では、2 つのモデルのブローカ設定プロパティーについて説明します。

表 14-4　持続に関するグローバルなブローカプロパティー

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.persist.store | 文字列 | file | 持続データストレージのモデル<br>file　ファイルベースの持続<br>jdbc　JDBC ベースの持続 |

# ファイルベースの持続

表 14-5 に、ファイルベースの持続に関するブローカプロパティーを示します。

**表 14-5** ファイルベースの持続に関するブローカプロパティー

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| `imq.persist.file.message.max_record_size` | 文字列 | `1m` | メッセージストレージファイルに追加するメッセージの最大サイズ<br><br>このサイズを超えるメッセージは、個別の専用ファイルに格納されます。<br><br>次の接尾辞を使用して、バイト、K バイト、または M バイトの単位で値を表すことができます。<br><br>`b` バイト<br>`k` K バイト (1024 バイト )<br>`m` M バイト (1024 x 1024 = 1,048,576 バイト )<br><br>接尾辞を付けない値は、バイト単位になります。<br><br>例 :<br><br>`1600` 1600 バイト<br>`1600b` 1600 バイト<br>`16k` 16K バイト (= 16,384 バイト )<br>`16m` 16M バイト (= 16,777,216 バイト ) |
| `imq.persist.file.destination.message.filepool.limit` | 整数 | `100` | 送信先ファイルプール内の再利用可能な空きファイルの最大数<br><br>この制限を超える空きファイルは削除されます。ブローカは、必要に応じて、制限を超える追加ファイルを作成および削除します。<br><br>この制限値が高いほど、ブローカが持続データを処理する速度が速くなります。 |

**表 14-5** ファイルベースの持続に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.persist.file.message.filepool.cleanratio | 整数 | 0 | 空きファイルのプールにクリーン ( 空 ) の状態で保持するファイルの割合<br><br>この値が高いほど、ファイルプールに必要なディスクスペースが少なくなりますが、処理中にファイルを消去するためのオーバーヘッドが大きくなります。 |
| imq.persist.file.message.cleanup | ブール | false | シャットダウン時に空きファイルのプール内のファイルを消去するかどうか<br><br>このプロパティーを true に設定すると、ファイルストア用のディスクスペースを節約できますが、ブローカのシャットダウンに時間がかかります。 |
| imq.persist.file.sync.enabled | ブール | false | メモリー内の状態を物理的なストレージデバイスと同期させるかどうか<br><br>このプロパティーを true に設定すると、システムクラッシュによるデータの損失は回避できますが、パフォーマンスが下がります。<br><br>**注 :** Sun Cluster と Message Queue の Sun Cluster データサービスを実行している場合は、すべてのクラスタノードのブローカでこのプロパティーを true に設定してください。 |

# JDBC ベースの持続

表 14-6 に、JDBC ベースの持続に関するブローカプロパティーを示します。示す例は、DataMirror Mobile Solutions, Inc の PointBase® ファミリーのデータベース製品の例です。

表 14-6　JDBC ベースの持続に関するブローカプロパティー

| プロパティー | 説明 | 例 |
|---|---|---|
| `imq.persist.jdbc.brokerid` | ( 任意指定 ) ブローカインスタンスの識別子<br><br>識別子には英数字を使用し、長さは *n* - 12 (*n* はデータベースで許可されるテーブル名の最大長 ) を超えないようにする必要があります。<br><br>複数のブローカインスタンスが同じデータベースを持続データストアとして使用する場合、データベーステーブル名を一意にするために、この識別子がデータベーステーブル名に追加されます。この識別子は、通常、1 つのブローカインスタンスのみのデータを保存する組み込みデータベースでは必要ありません。 | PointBase 組み込みバージョンの場合は不要です |
| `imq.persist.jdbc.driver` | データベースに接続するための JDBC ドライバの Java クラス名 | `com.pointbase.jdbc.jdbcUniversalDriver` |
| `imq.persist.jdbc.opendburl` | 既存のデータベースへの接続を開くための URL | `jdbc:pointbase:embedded:`*dbName*`;`<br>`database.home=`<br>`.../instances/`*instanceName*`/dbstore` |
| `imq.persist.jdbc.createdburl` | ( 任意指定 ) 新しいデータベースを作成するための URL<br><br>このプロパティーは、Message Queue データベースマネージャーユーティリティー (`imqdbmgr`) を使用してデータベースを作成する場合にのみ必要です。 | `jdbc:pointbase:embedded:`*dbName*`;`<br>`new,database.home=`<br>`.../instances/`*instanceName*`/dbstore` |

表 14-6　JDBC ベースの持続に関するブローカプロパティー ( 続き )

| プロパティー | 説明 | 例 |
|---|---|---|
| imq.persist.jdbc.closedburl | ( 任意指定 ) データベース接続を閉じるための URL | PointBase の場合は不要です |
| imq.persist.jdbc.user | ( 任意指定 ) 必要に応じて、データベース接続を開くためのユーザー名。<br>セキュリティーを考慮する場合、代わりにコマンド行オプション imqbrokerd -dbuser と imqdbmgr -u を使用してこの値を指定できます。 | |
| imq.persist.jdbc.needpassword | ( 任意指定 ) データベースでブローカアクセス用のパスワードが必要かどうか<br>true の場合、imqbrokerd および imqdbmgr コマンドでは、-passfile オプションを使用してパスワードを含むパスワードファイルを指定しないかぎり、パスワードが要求されます。 | |
| imq.persist.jdbc.password | ( 任意指定 ) データベース接続を開くためのパスワード<br>このプロパティーは、パスワードファイルでのみ指定するようにしてください。 | |
| imq.persist.jdbc.table.IMQSV35 | バージョンテーブルを作成するための SQL コマンド | CREATE TABLE ${*name*}<br>(STOREVERSION INTEGER NOT NULL,<br>BROKERID VARCHAR(100)) |
| imq.persist.jdbc.table.IMQCCREC35 | 設定変更レコードテーブルを作成するための SQL コマンド | CREATE TABLE ${*name*}<br>(RECORDTIME BIGINT NOT NULL,<br>RECORD BLOB(10k)) |
| imq.persist.jdbc.table.IMQDEST35 | 送信先テーブルを作成するための SQL コマンド | CREATE TABLE ${*name*}<br>(DID VARCHAR(100) NOT NULL,<br>DEST BLOB(10k),<br>*primaryKey*(DID)) |

表 14-6　JDBC ベースの持続に関するブローカプロパティー ( 続き )

| プロパティー | 説明 | 例 |
|---|---|---|
| imq.persist.jdbc.table.IMQINT35 | 配信対象テーブルを作成するための SQL コマンド | CREATE TABLE ${name}<br>(CUID BIGINT NOT NULL,<br>INTEREST BLOB(10k),<br>*primaryKey*(CUID)) |
| imq.persist.jdbc.table.IMQMSG35 | メッセージテーブルを作成するための SQL コマンド<br><br>MSG 列のデフォルトの最大長は、1M バイト (1m) です。メッセージがこれより長くなると予想される場合は、必要に応じて長さを設定します。テーブルがすでに作成されている場合、メッセージの最大長を変更するには、テーブルを作成し直す必要があります。 | CREATE TABLE ${*name*}<br>(MID VARCHAR(100) NOT NULL,<br>DID VARCHAR(100),<br>MSGSIZE BIGINT,<br>MSG BLOB(1m),<br>*primaryKey*(MID)) |
| imq.persist.jdbc.table.IMQPROPS35 | プロパティーテーブルを作成するための SQL コマンド | CREATE TABLE ${*name*}<br>(PROPNAME VARCHAR(100) NOT NULL,<br>PROPVALUE BLOB(10k),<br>*primaryKey*(PROPNAME)) |
| imq.persist.jdbc.table.IMQILIST35 | 配信対象の状態テーブルを作成するための SQL コマンド | CREATE TABLE ${*name*}<br>(MID VARCHAR(100) NOT NULL,<br>CUID BIGINT,<br>DID VARCHAR(100),<br>STATE INTEGER,<br>*primaryKey*(MID, CUID)) |
| imq.persist.jdbc.table.IMQTXN35 | トランザクションテーブルを作成するための SQL コマンド | CREATE TABLE ${*name*}<br>(TUID BIGINT NOT NULL,<br>STATE INTEGER,<br>TSTATEOBJ BLOB(10K),<br>*primaryKey*(TUID)) |
| imq.persist.jdbc.table.IMQTACK35 | トランザクション通知テーブルを作成するための SQL コマンド | CREATE TABLE ${*name*}<br>(TUID BIGINT NOT NULL,<br>TXNACK BLOB(10k)) |

# セキュリティーのプロパティー

表 14-7 に、セキュリティーサービスに関するブローカプロパティーを示します。

**表 14-7　セキュリティーに関するブローカプロパティー**

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.accesscontrol.enabled | ブール | true | アクセス制御を使用するかどうか<br><br>true の場合、アクセス制御プロパティーファイルが確認され、認証されたユーザーについて、接続サービスの使用または特定の送信先に対する特定の操作の実行が承認されているかどうかが検証されます。 |
| imq.*serviceName*.accesscontrol.enabled | ブール | なし | 接続サービスのアクセス制御を使用するかどうか<br><br>指定すると、指定した接続サービスの imq.accesscontrol.enabled がオーバーライドされます。<br><br>true の場合、アクセス制御プロパティーファイルが確認され、認証されたユーザーについて、指定した接続サービスの使用または特定の送信先に対する特定の操作の実行が承認されているかどうかが検証されます。 |
| imq.accesscontrol.file.filename | 文字列 | accesscontrol.properties | アクセス制御プロパティーファイルの名前<br><br>ファイル名は、アクセス制御ディレクトリ (付録 A を参照) への相対パスで指定します。 |

**表 14-7** セキュリティーに関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.*serviceName*.accesscontrol.file.filename | 文字列 | なし | 接続サービスのアクセス制御プロパティーファイルの名前<br>指定すると、指定した接続サービスの imq.accesscontrol.file.filename がオーバーライドされます。<br>ファイル名は、アクセス制御ディレクトリ ( 付録 A を参照 ) への相対パスで指定します。 |
| imq.authentication.type | 文字列 | digest | パスワードの符号化方法<br>basic Base-64<br>digest MD5 |
| imq.*serviceName*.authentication.type | 文字列 | なし | 接続サービスのパスワードの符号化方法<br>basic Base-64<br>digest MD5<br>指定すると、指定した接続サービスの imq.authentication.type がオーバーライドされます。 |
| imq.authentication.basic.user_repository | 文字列 | file | Base-64 認証のユーザーリポジトリのタイプ<br>file ファイルベース<br>ldap LDAP |
| imq.authentication.client.response.timeout | 整数 | 180 | 認証要求に対するクライアントの応答を待機する間隔 ( 秒単位 ) |
| imq.passfile.enabled | ブール | false | パスワードをパスワードファイルから取得するかどうか |
| imq.passfile.dirpath | 文字列 | 付録 A を参照 | パスワードファイルを含むディレクトリへのパス |
| imq.passfile.name | 文字列 | passfile | パスワードファイルの名前 |

**表 14-7** セキュリティーに関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.imqcmd.password | 文字列 | なし | 管理ユーザーのパスワード<br><br>コマンドユーティリティー (imqcmd) では、コマンドの実行前に、このパスワードを使用してユーザーが認証されます。 |
| imq.user_repository.ldap.server | 文字列 | なし | LDAP サーバーのホスト名とポート番号<br><br>値は次の形式になります。<br><br>*hostName*:*port*<br><br>*hostName* は、LDAP サーバーを実行するホストの完全修飾 DNS 名で、*port* は、サーバーが使用するポート番号です。<br><br>フェイルオーバーサーバーのリストを指定するには、次の構文を使用します。<br><br>*host1*:*port1*<br>ldap://*host2*:*port2*<br>ldap://*host3*:*port3*<br>...<br><br>リスト内のエントリはスペースで区切ります。各フェールオーバーサーバーのアドレスの先頭には ldap:// を付けます。SSL を使用し、プロパティー imq.user_repository.ldap.ssl.enabled を true に設定している場合でも、この形式を使用します。このアドレスでは ldaps を指定する必要はありません。 |
| imq.user_repository.ldap.principal | 文字列 | なし | LDAP ユーザーリポジトリにバインドするための識別名<br><br>LDAP サーバーで匿名検索が許可されている場合は必要ありません。 |

**表 14-7** セキュリティーに関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.user_repository.ldap.password | 文字列 | なし | LDAP ユーザーリポジトリにバインドするためのパスワード<br><br>LDAP サーバーで匿名検索が許可されている場合は必要ありません。<br><br>このプロパティーは、パスワードファイルでのみ指定するようにしてください。 |
| imq.user_repository.ldap.base | 文字列 | なし | LDAP ユーザーエントリのディレクトリベース |
| imq.user_repository.ldap.uidattr | 文字列 | なし | LDAP ユーザー名のプロバイダ固有の属性識別子 |
| imq.user_repository.ldap.usrfilter | 文字列 | なし | ( 任意指定 ) LDAP ユーザー検索の JNDI フィルタ |
| imq.user_repository.ldap.grpsearch | ブール | false | LDAP グループ検索を有効にするかどうか<br><br>**注** : Message Queue は、入れ子にされたグループをサポートしていません。 |
| imq.user_repository.ldap.grpbase | 文字列 | なし | LDAP グループエントリのディレクトリベース |
| imq.user_repository.ldap.gidattr | 文字列 | なし | LDAP グループ名のプロバイダ固有の属性識別子 |
| imq.user_repository.ldap.memattr | 文字列 | なし | LDAP グループ内のユーザー名のプロバイダ固有の属性識別子 |
| imq.user_repository.ldap.grpfilter | 文字列 | なし | ( 任意指定 ) LDAP グループ検索の JNDI フィルタ |
| imq.user_repository.ldap.timeout | 整数 | 280 | LDAP 検索の制限時間 ( 秒単位 ) |
| imq.user_repository.ldap.ssl.enabled | ブール | false | LDAP サーバーとの通信に SSL を使用するかどうか |
| imq.keystore.file.dirpath | 文字列 | 付録 A を参照 | キーストアファイルを含むディレクトリへのパス |
| imq.keystore.file.name | 文字列 | keystore | キーストアファイルの名前 |

表 14-7　セキュリティーに関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.keystore.password | 文字列 | なし | キーストアファイルのパスワード<br><br>このプロパティーは、パスワードファイルでのみ指定するようにしてください。 |
| imq.audit.enabled | ブール | false | ブローカログファイルへの監査ロギングを開始するかどうか<br><br>このオプションは、Message Queue Enterprise Edition のみに適用されます。 |

# 監視のプロパティー

表 14-8 に、監視サービスに関するブローカプロパティーを示します。

表 14-8　監視に関するブローカプロパティー

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.log.level[1] | 文字列 | INFO | ロギングレベル<br><br>出力チャネルに書き込むことのできるロギング情報のカテゴリを指定します。使用可能な値には、レベルの高い順に次のものがあります。<br><br>ERROR<br>WARNING<br>INFO<br><br>各レベルには、その上位のレベルが含まれます。たとえば、WARNING には ERROR が含まれます。 |

表 14-8　監視に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.destination.logDeadMsgs[1] | ブール | false | デッドメッセージに関する情報をログに書き込むかどうか<br>true の場合、次のイベントがログに書き込まれます。<br>• 送信先が最大サイズまたは最大メッセージ数に達していっぱいになった。<br>• 管理コマンドか配信通知以外の理由でブローカがメッセージを破棄した。<br>• ブローカがデッドメッセージキューにメッセージを移動した。 |
| imq.log.console.stream | 文字列 | ERR | コンソール出力の送信先<br>OUT stdout<br>ERR stderr |
| imq.log.console.output | 文字列 | ERROR\|WARNING | コンソールに書き込むロギング情報のカテゴリ<br>NONE<br>ERROR<br>WARNING<br>INFO<br>ALL<br>ERROR、WARNING、および INFO カテゴリには、それぞれの上位のカテゴリは含まれません。したがって、必要に応じて各カテゴリを明示的に指定する必要があります。カテゴリの組み合わせは、縦線 (\|) で区切って指定できます。 |
| imq.log.file.dirpath | 文字列 | 付録 A を参照 | ログファイルを含むディレクトリへのパス |
| imq.log.file.filename | 文字列 | log.txt | ログファイルの名前 |

**表 14-8** 監視に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.log.file.output | 文字列 | ALL | ログファイルに書き込むロギング情報のカテゴリ<br><br>NONE<br>ERROR<br>WARNING<br>INFO<br>ALL<br><br>ERROR、WARNING、および INFO カテゴリには、それぞれの上位のカテゴリは含まれません。したがって、必要に応じて各カテゴリを明示的に指定する必要があります。カテゴリの組み合わせは、縦線 (\|) で区切って指定できます。 |
| imq.log.file.rolloverbytes[1] | 整数 | -1 | 出力を新しいログファイルにロールオーバーするファイル長 ( バイト単位 )<br><br>値 -1 は、バイト数が無制限である、つまりファイル長に基づくロールオーバーは行わないことを示します。 |
| imq.log.file.rolloversecs[1] | 整数 | 604800 (1 週間 ) | 出力を新しいログファイルにロールオーバーするファイルの有効期間 ( 秒単位 )<br><br>値 -1 は、秒数が無制限である、つまりファイルの有効期間に基づくロールオーバーは行わないことを示します。 |
| imq.log.syslog.output[2] | 文字列 | ERROR | syslogd(1M) に書き込むロギング情報のカテゴリ<br><br>NONE<br>ERROR<br>WARNING<br>INFO<br>ALL<br><br>ERROR、WARNING、および INFO カテゴリには、それぞれの上位のカテゴリは含まれません。したがって、必要に応じて各カテゴリを明示的に指定する必要があります。カテゴリの組み合わせは、縦線 (\|) で区切って指定できます。 |

表 14-8　監視に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.log.syslog.facility[2] | 文字列 | LOG_DAEMON | メッセージのロギングのための syslog 機能<br>使用可能な値は、syslog(3C) のマニュアルページに示される値を反映しています。Message Queue で使用するために適切な値は次のとおりです。<br>LOG_USER<br>LOG_DAEMON<br>LOG_LOCAL0<br>LOG_LOCAL1<br>LOG_LOCAL2<br>LOG_LOCAL3<br>LOG_LOCAL4<br>LOG_LOCAL5<br>LOG_LOCAL6<br>LOG_LOCAL7 |
| imq.log.syslog.identity[2] | 文字列 | imqbrokerd_${imq.*instanceName*} | syslog にログとして書き込まれるすべてのメッセージの先頭に付ける識別文字列 |
| imq.log.syslog.logpid[2] | ブール | true | メッセージとともにブローカのプロセス ID をログに書き込むかどうか |
| imq.log.syslog.logconsole[2] | ブール | false | メッセージを syslog に送信できなかった場合にシステムコンソールに書き込むかどうか |
| imq.log.timezone | 文字列 | 該当地域のタイムゾーン | ログのタイムスタンプのタイムゾーン。<br>使用可能な値は、java.util.TimeZone.getTimeZone で使用される値と同じです。次に例を示します。<br>GMT<br>GMT-8:00<br>America/LosAngeles<br>Europe/Rome<br>Asia/Tokyo |
| imq.metrics.enabled | ブール | true | ロガーへのメトリックス情報の書き込みを有効にするかどうか<br>imq.metrics.topic.enabled で制御するメトリックスメッセージの生成には影響しません。 |

**表 14-8** 監視に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.metrics.interval | 整数 | -1 | ロガーにメトリックス情報を書き込む間隔 ( 秒単位 )<br>imq.metrics.topic.interval で制御するメトリックスメッセージの生成間隔には影響しません。<br>値 -1 は、無期限の間隔、つまりロガーにメトリックス情報を書き込まないことを示します。 |
| imq.metrics.topic.enabled | ブール | true | メトリックストピック送信先へのメトリックスメッセージの生成を有効にするかどうか<br>false の場合、メトリックストピック送信先へサブスクライブしようとすると、クライアント側の例外がスローされます。 |
| imq.metrics.topic.interval | 整数 | 60 | メトリックストピック送信先へのメトリックスメッセージを生成する間隔 ( 秒単位 ) |
| imq.metrics.topic.persist | ブール | false | メトリックストピック送信先に送信されるメトリックスメッセージが持続的かどうか |
| imq.metrics.topic.timetolive | 整数 | 300 | メトリックストピック送信先に送信されるメトリックスメッセージの有効期間 ( 秒単位 ) |

1. imqcmd update bkr コマンドで使用できます
2. Solaris プラットフォームのみ

# クラスタ設定プロパティー

表 14-9 に、ブローカクラスタに関する設定プロパティーを示します。

**表 14-9** クラスタ設定に関するブローカプロパティー

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.cluster.brokerlist[1] | 文字列 | なし | ブローカのアドレスのリスト<br>リストは、コンマ区切りの複数のアドレスで構成します。各アドレスでは、クラスタ内のブローカのホスト名とポートマッパーのポート番号を、*hostName*:*portNumber* の形式で指定します。次に例を示します。<br>`host1:3000,host2:8000,ctrlhost` |
| imq.cluster.hostname[2] | 文字列 | なし | cluster 接続サービスのホスト名または IP アドレス<br>指定すると、cluster 接続サービスについて imq.hostname (295 ページの表 14-1 を参照) がオーバーライドされます |
| imq.cluster.port[2] | 整数 | 0 | cluster 接続サービスのポート番号<br>値 0 は、ポート番号がポートマッパーによって動的に割り当てられることを示します。 |
| imq.cluster.transport[1] | 文字列 | tcp | cluster 接続サービスのネットワークトランスポートプロトコル<br>ブローカ間の安全で暗号化されたメッセージ配信を実現するためには、このプロパティーを ssl に設定します。 |
| imq.cluster.url[1]、[3] | 文字列 | なし | cluster 設定ファイルがある場合のファイルの URL<br>例:<br>`http://webserver/imq/cluster.properties`<br>(Web サーバー上のファイルの場合)<br>`file:/net/mfsserver/imq/cluster.properties`<br>( 共有ドライブ上のファイルの場合 ) |

**表 14-9** クラスタ設定に関するブローカプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imq.cluster.masterbroker[1] | 文字列 | なし | クラスタのマスターブローカがある場合の、そのホスト名とポート番号<br><br>値は *hostName*:*portNumber* の形式になります。*hostName* はマスターブローカのホスト名で、*portNumber* はマスターブローカのポートマッパーのポート番号です。次に例を示します。<br><br>`ctrlhost:7676` |

1. クラスタ内のすべてのブローカで同じ値にする必要があります
2. クラスタ内のブローカごとに個別に指定できます。
3. `imqcmd update bkr` コマンドで使用できます

# ブローカプロパティーのアルファベット順の一覧

表 14-10 に、ブローカ設定プロパティーのアルファベット順の一覧と、この章内の関連する表への相互参照を示します。

**表 14-10** ブローカプロパティーのアルファベット順の一覧

| プロパティー | 表 |
|---|---|
| imq.accesscontrol.enabled | 309 ページの表 14-7 |
| imq.accesscontrol.file.filename | 309 ページの表 14-7 |
| imq.audit.enabled | 309 ページの表 14-7 |
| imq.authentication.basic.user_repository | 309 ページの表 14-7 |
| imq.authentication.client.response.timeout | 309 ページの表 14-7 |
| imq.authentication.type | 309 ページの表 14-7 |
| imq.autocreate.destination.isLocalOnly | 299 ページの表 14-3 |
| imq.autocreate.destination.limitBehavior | 299 ページの表 14-3 |
| imq.autocreate.destination.maxBytesPerMsg | 299 ページの表 14-3 |
| imq.autocreate.destination.maxNumMsgs | 299 ページの表 14-3 |
| imq.autocreate.destination.maxNumProducers | 299 ページの表 14-3 |
| imq.autocreate.destination.maxTotalMsgBytes | 299 ページの表 14-3 |
| imq.autocreate.destination.useDMQ | 299 ページの表 14-3 |

表 14-10　ブローカプロパティーのアルファベット順の一覧（続き）

| プロパティー | 表 |
|---|---|
| imq.autocreate.queue | 299 ページの表 14-3 |
| imq.autocreate.queue.consumerFlowLimit | 299 ページの表 14-3 |
| imq.autocreate.queue.localDeliveryPreferred | 299 ページの表 14-3 |
| imq.autocreate.queue.maxNumActiveConsumers | 299 ページの表 14-3 |
| imq.autocreate.queue.maxNumBackupConsumers | 299 ページの表 14-3 |
| imq.autocreate.topic | 299 ページの表 14-3 |
| imq.autocreate.topic.consumerFlowLimit | 299 ページの表 14-3 |
| imq.cluster.brokerlist | 318 ページの表 14-9 |
| imq.cluster.hostname | 318 ページの表 14-9 |
| imq.cluster.masterbroker | 318 ページの表 14-9 |
| imq.cluster.port | 318 ページの表 14-9 |
| imq.cluster.transport | 318 ページの表 14-9 |
| imq.cluster.url | 318 ページの表 14-9 |
| imq.destination.DMQ.truncateBody | 298 ページの表 14-2 |
| imq.destination.logDeadMsgs | 313 ページの表 14-8 |
| imq.hostname | 295 ページの表 14-1 |
| imq.imqcmd.password | 309 ページの表 14-7 |
| imq.keystore.file.dirpath | 309 ページの表 14-7 |
| imq.keystore.file.name | 309 ページの表 14-7 |
| imq.keystore.password | 309 ページの表 14-7 |
| imq.keystore.*propertyName* | 309 ページの表 14-7 |
| imq.log.console.output | 313 ページの表 14-8 |
| imq.log.console.stream | 313 ページの表 14-8 |
| imq.log.file.dirpath | 313 ページの表 14-8 |
| imq.log.file.filename | 313 ページの表 14-8 |
| imq.log.file.output | 313 ページの表 14-8 |
| imq.log.file.rolloverbytes | 313 ページの表 14-8 |
| imq.log.file.rolloversecs | 313 ページの表 14-8 |
| imq.log.level | 313 ページの表 14-8 |

表 14-10　ブローカプロパティーのアルファベット順の一覧 ( 続き )

| プロパティー | 表 |
|---|---|
| imq.log.syslog.facility | 313 ページの表 14-8 |
| imq.log.syslog.identity | 313 ページの表 14-8 |
| imq.log.syslog.logconsole | 313 ページの表 14-8 |
| imq.log.syslog.logpid | 313 ページの表 14-8 |
| imq.log.syslog.output | 313 ページの表 14-8 |
| imq.log.timezone | 313 ページの表 14-8 |
| imq.message.expiration.interval | 298 ページの表 14-2 |
| imq.message.max_size | 298 ページの表 14-2 |
| imq.metrics.enabled | 313 ページの表 14-8 |
| imq.metrics.interval | 313 ページの表 14-8 |
| imq.metrics.topic.enabled | 313 ページの表 14-8 |
| imq.metrics.topic.interval | 313 ページの表 14-8 |
| imq.metrics.topic.persist | 313 ページの表 14-8 |
| imq.metrics.topic.timetolive | 313 ページの表 14-8 |
| imq.passfile.dirpath | 309 ページの表 14-7 |
| imq.passfile.enabled | 309 ページの表 14-7 |
| imq.passfile.name | 309 ページの表 14-7 |
| imq.persist.file.destination.message.filepool.limit | 304 ページの表 14-5 |
| imq.persist.file.message.cleanup | 304 ページの表 14-5 |
| imq.persist.file.message.filepool.cleanratio | 304 ページの表 14-5 |
| imq.persist.file.message.max_record_size | 304 ページの表 14-5 |
| imq.persist.file.sync.enabled | 304 ページの表 14-5 |
| imq.persist.jdbc.brokerid | 306 ページの表 14-6 |
| imq.persist.jdbc.closedburl | 306 ページの表 14-6 |
| imq.persist.jdbc.createdburl | 306 ページの表 14-6 |
| imq.persist.jdbc.driver | 306 ページの表 14-6 |
| imq.persist.jdbc.needpassword | 306 ページの表 14-6 |
| imq.persist.jdbc.opendburl | 306 ページの表 14-6 |

表 14-10　ブローカプロパティーのアルファベット順の一覧（続き）

| プロパティー | 表 |
|---|---|
| imq.persist.jdbc.password | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQCCREC35 | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQDEST35 | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQILIST35 | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQINT35 | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQMSG35 | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQPROPS35 | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQSV35 | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQTACK35 | 306 ページの表 14-6 |
| imq.persist.jdbc.table.IMQTXN35 | 306 ページの表 14-6 |
| imq.persist.jdbc.user | 306 ページの表 14-6 |
| imq.persist.store | 303 ページの表 14-4 |
| imq.ping.interval | 295 ページの表 14-1 |
| imq.portmapper.backlog | 295 ページの表 14-1 |
| imq.portmapper.hostname | 295 ページの表 14-1 |
| imq.portmapper.port | 295 ページの表 14-1 |
| imq.*resourceState*.count | 298 ページの表 14-2 |
| imq.*resourceState*.threshold | 298 ページの表 14-2 |
| imq.service.activelist | 295 ページの表 14-1 |
| imq.*serviceName*.accesscontrol.enabled | 309 ページの表 14-7 |
| imq.*serviceName*.accesscontrol.file.filename | 309 ページの表 14-7 |
| imq.*serviceName*.authentication.type | 309 ページの表 14-7 |
| imq.*serviceName*.max_threads | 295 ページの表 14-1 |
| imq.*serviceName*.min_threads | 295 ページの表 14-1 |
| imq.*serviceName*.*protocolType*.hostname | 295 ページの表 14-1 |
| imq.*serviceName*.*protocolType*.port | 295 ページの表 14-1 |
| imq.*serviceName*.threadpool_model | 295 ページの表 14-1 |
| imq.shared.connectionMonitor_limit | 295 ページの表 14-1 |
| imq.system.max_count | 298 ページの表 14-2 |

表 14-10　ブローカプロパティーのアルファベット順の一覧 ( 続き )

| プロパティー | 表 |
|---|---|
| imq.system.max_size | 298 ページの表 14-2 |
| imq.transaction.autorollback | 298 ページの表 14-2 |
| imq.user_repository.ldap.base | 309 ページの表 14-7 |
| imq.user_repository.ldap.gidattr | 309 ページの表 14-7 |
| imq.user_repository.ldap.grpbase | 309 ページの表 14-7 |
| imq.user_repository.ldap.grpfilter | 309 ページの表 14-7 |
| imq.user_repository.ldap.grpsearch | 309 ページの表 14-7 |
| imq.user_repository.ldap.memattr | 309 ページの表 14-7 |
| imq.user_repository.ldap.password | 309 ページの表 14-7 |
| imq.user_repository.ldap.principal | 309 ページの表 14-7 |
| imq.user_repository.ldap.*propertyName* | 309 ページの表 14-7 |
| imq.user_repository.ldap.server | 309 ページの表 14-7 |
| imq.user_repository.ldap.ssl.enabled | 309 ページの表 14-7 |
| imq.user_repository.ldap.timeout | 309 ページの表 14-7 |
| imq.user_repository.ldap.uidattr | 309 ページの表 14-7 |
| imq.user_repository.ldap.usrfilter | 309 ページの表 14-7 |

第 15 章

# 物理的送信先のプロパティーのリファレンス

この章では、物理的送信先の設定プロパティーに関する参照情報を提供します。これらのプロパティーは、物理的送信先の作成時または更新時に設定できます。自動作成される送信先の場合は、ブローカのインスタンス設定ファイルにデフォルト値を設定します (299 ページの表 14-3 を参照 )。

**表 15-1**　物理的送信先のプロパティー

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| maxNumMsgs[1] | 整数 | -1 | 消費されていないメッセージの最大数<br>値 -1 は、メッセージ数が無制限であることを示します。<br>デッドメッセージキューの場合、デフォルト値は 1000 です。 |

**表 15-1** 物理的送信先のプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| maxBytesPerMsg | 文字列 | -1 | 単一メッセージの最大サイズ ( バイト単位 )<br><br>持続メッセージの拒否は、プロデューシングクライアントに例外として報告されます。持続性のないメッセージの場合は、通知は送信されません。<br><br>次の接尾辞を使用して、バイト、K バイト、または M バイトの単位で値を表すことができます。<br><br>b バイト<br>k K バイト (1024 バイト )<br>m M バイト (1024 x 1024 = 1,048,576 バイト )<br><br>接尾辞を付けない値は、バイト単位になります。値 -1 は、メッセージサイズが無制限であることを示します。<br><br>例 :<br><br>1600 1600 バイト<br>1600b 1600 バイト<br>16k 16K バイト (= 16,384 バイト )<br>16m 16M バイト (= 16,777,216 バイト )<br>-1 無制限 |
| maxTotalMsgBytes[1] | 文字列 | -1 | 消費されていないメッセージの最大合計メモリー ( バイト単位 )<br><br>構文は maxBytesPerMsg と同じです ( 前述の説明を参照 )。<br><br>デッドメッセージキューの場合、デフォルト値は 10m です。 |

**表 15-1**　物理的送信先のプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| limitBehavior | 文字列 | REJECT_NEWEST | メモリー制限のしきい値に達したときのブローカの動作<br>FLOW_CONTROL：プロデューサの処理速度を下げます<br>REMOVE_OLDEST：もっとも古いメッセージを破棄します<br>REMOVE_LOW_PRIORITY：有効期限に従ってもっとも優先度の低いメッセージを破棄します。プロデューシングクライアントには通知しません<br>REJECT_NEWEST：もっとも新しいメッセージを拒否します。持続メッセージの場合のみ、プロデューシングクライアントに例外を通知します<br>値が REMOVE_OLDEST または REMOVE_LOW_PRIORITY で、useDMQ プロパティーが true の場合、超過したメッセージはデッドメッセージキューに移動されます。デッドメッセージキューは、デフォルトの制限動作が REMOVE_OLDEST であり、FLOW_CONTROL には設定できません。 |
| maxNumProducers[2] | 整数 | -1 | 送信先のメッセージプロデューサの最大数<br>この制限に達すると、新しいプロデューサを作成できません。値 -1 は、プロデューサ数が無制限であることを示します。 |
| maxNumActiveConsumers[3] | 整数 | 1 | キュー送信先からのロードバランスされた配信でアクティブにするメッセージコンシューマの最大数<br>値 -1 は、コンシューマ数が無制限であることを示します。Sun Java System Message Queue Platform Edition では、この値は 2 に制限されます。 |

表 15-1　物理的送信先のプロパティー ( 続き )

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| maxNumBackupConsumers[3] | 整数 | 0 | キュー送信先からのロードバランスされた配信でバックアップにするメッセージコンシューマの最大数<br><br>値 -1 は、コンシューマ数が無制限であることを示します。Sun Java System Message Queue Platform Edition では、この値は 1 に制限されます。 |
| consumerFlowLimit | 整数 | 1000 | コンシューマに単一のバッチで配信するメッセージの最大数<br><br>ロードバランスされたキュー配信では、ロードバランスが開始されるまでの、アクティブコンシューマにルーティングされるキュー内のメッセージの初期数になります。送信先コンシューマは、接続で低い値を指定することで、この制限をオーバーライドできます。<br><br>値 -1 は、コンシューマ数が無制限であることを示します。 |
| isLocalOnly[2] | ブール | false | ローカル配信のみかどうか<br><br>このプロパティーは、ブローカクラスタ内の送信先のみに適用されます。送信先の作成後は変更できません。true の場合、送信先はほかのブローカに複製されず、ローカルコンシューマ ( 送信先が作成されたブローカに接続しているコンシューマ ) だけにメッセージを配信するように制限されます。 |
| localDeliveryPreferred[2、3] | ブール | false | ローカル配信優先かどうか<br><br>このプロパティーは、ブローカクラスタ内のロードバランスされたキュー配信のみに適用されます。true の場合、メッセージは、ローカルブローカにコンシューマがない場合にのみリモートコンシューマに配信されます。送信先をローカルのみの配信に制限しないでください。つまり、isLocalOnly を false にする必要があります。 |
| useDMQ[2] | ブール | true | デッドメッセージをデッドメッセージキューに送信するかどうか<br><br>false の場合、デッドメッセージは単に破棄されます。 |

1. クラスタ環境では、クラスタ内のすべての送信先のインスタンスに一括ではなく、各インスタンスに個別に適用されます
2. デッドメッセージキューには適用されません
3. キュー送信先のみ

第 16 章

# 管理対象オブジェクト属性のリファレンス

この章では、管理対象オブジェクトの属性に関する参照情報を提供します。この章は、次の節から構成されています。



## 接続ファクトリの属性

接続ファクトリオブジェクトの属性は、以下の節で説明する次のカテゴリに分けられます。

# 接続の処理

表 16-1 に、接続処理に関する接続ファクトリ属性を示します。

**表 16-1** 接続の処理に関する接続ファクトリ属性

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqAddressList | 文字列 | 既存の Message Queue 3.0 アドレス、またはそれがない場合は 332 ページの表 16-2 の最初のエントリ | ブローカのアドレスのリスト<br>リストは、コンマ区切りの 1 つ以上のメッセージサーバーアドレスから構成されます。各アドレスでは、クライアントが接続できるブローカインスタンスのホスト名、ポート番号、および接続サービスを明示的または暗黙的に指定します。アドレスの構文は、接続サービスとポートの割り当て方法によって異なります。詳細については、後述の説明を参照してください。 |
| imqAddressListBehavior | 文字列 | PRIORITY | サーバーアドレスに接続を試行する順序<br>PRIORITY アドレスリストに指定されている順序<br>RANDOM ランダムな順序<br>**注 :** 多数のクライアントが同じ接続ファクトリを共有する場合は、すべてのクライアントが同じアドレスに接続を試行するのを避けるために、ランダムな接続順序を指定してください。 |
| imqAddressListIterations | 整数 | 5 | 接続の確立または再確立を試行する際にアドレスリストを繰り返す回数<br>値 -1 は、繰り返し回数が無制限であることを示します。 |
| imqPingInterval | 整数 | 30 | クライアントとブローカ間のテスト接続の間隔 ( 秒単位 )<br>値 0 または -1 を指定すると、接続の定期的なテストが無効になります。 |
| imqReconnectEnabled | ブール | false | 失われた接続の再確立を試行するかどうか |
| imqReconnectAttempts | 整数 | 0 | アドレスリスト内の次のアドレスに移る前に、各アドレスに接続または再接続を試行する回数<br>値 -1 は、接続の試行回数が無制限であることを示します。この場合、成功するまで、最初のアドレスに繰り返し接続が試行されます。 |

**表 16-1** 接続の処理に関する接続ファクトリ属性 ( 続き )

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqReconnectInterval | 倍長整数 | 3000 | 再接続を試行する間隔 ( ミリ秒単位 )<br><br>この値は、特定のアドレスに対する連続する試行と、リスト内の連続するアドレスに対する試行の両方に適用されます。<br><br>**注 :** 値が小さすぎると、ブローカが回復するのに十分な時間が与えられない可能性があります。値が大きすぎると、許容できない接続遅延が発生する可能性があります。 |
| imqSSLIsHostTrusted | ブール | true | ブローカの自己署名型の認証証明書を受け入れるかどうか<br><br>**注 :** 認証局からの署名付き証明書を使用するには、この属性を false に設定します。 |

imqAddressList 属性の値は、接続先のメッセージサーバーのアドレスを 1 つ以上指定するコンマ区切りの文字列です。各アドレスの一般的な構文は、次のようになります。

*scheme* : / /*address*

*scheme* には、表 16-2 に示すアドレススキーマのいずれかを指定し、*address* には、サーバーのアドレス自体を指定します。アドレスを指定するために正確な構文は、表の最後の列に示すように、アドレススキーマによって異なります。

表 16-2　メッセージサーバーのアドレススキーマ

| スキーマ | サービス | 構文 | 説明 |
|---|---|---|---|
| mq | jms または ssljms | [*hostName*][ :*portNumber*][/*serviceName*] | jms または ssljms 接続サービスで、ポートを動的に割り当てます。<br><br>アドレスリストのエントリには、Message Queue ポートマッパーのホスト名とポート番号を指定します。ポートマッパー自体が、接続に使用するポートを動的に割り当てます。<br><br>デフォルト値 :<br><br>*hostName* = localhost<br>*portNumber* = 7676<br>*serviceName* = jms<br><br>ssljms 接続サービスでは、すべての変数を明示的に指定する必要があります。 |
| mqtcp | jms | *hostName*:*portNumber*/jms | jms 接続サービスを使用して、指定されたポートに接続します。<br><br>ポートマッパーをバイパスし、指定されたホスト名とポート番号に対して直接、TCP 接続を作成します。 |
| mqssl | ssljms | *hostName*:*portNumber*/ssljms | ssljms 接続サービスを使用して、指定されたポートに接続します。<br><br>ポートマッパーをバイパスし、指定されたホスト名とポート番号に対して直接、安全な SSL 接続を作成します。 |

表 16-2　メッセージサーバーのアドレススキーマ ( 続き )

| スキーマ | サービス | 構文 | 説明 |
|---|---|---|---|
| http | httpjms | http://*hostName*:*portNumber*/*contextRoot*/tunnel<br><br>複数のブローカインスタンスが同じトンネルサーブレットを使用する場合、次の構文では、ランダムに選択されたブローカインスタンスではなく、特定のブローカインスタンスに接続します。<br><br>http://*hostName*:*portNumber*/*contextRoot*/tunnel?<br>ServerName=*hostName*:*instanceName* | httpjms 接続サービスを使用して、指定されたポートに接続します。<br><br>指定された URL の Message Queue トンネルサーブレットに対して、HTTP 接続を作成します。HTTP トンネルサーブレットにアクセスするように、ブローカを設定する必要があります。 |
| https | httpsjms | https://*hostName*:*portNumber*/*contextRoot*/tunnel<br><br>複数のブローカインスタンスが同じトンネルサーブレットを使用する場合、次の構文では、ランダムに選択されたブローカインスタンスではなく、特定のブローカインスタンスに接続します。<br><br>https://*hostName*:*portNumber*/*contextRoot*/tunnel?<br>ServerName=*hostName*:*instanceName* | httpsjms 接続サービスを使用して、指定されたポートに接続します。<br><br>指定された URL の Message Queue トンネルサーブレットに対して、安全な HTTPS 接続を作成します。HTTPS トンネルサーブレットにアクセスするように、ブローカを設定する必要があります。 |

表 16-3 に、さまざまなアドレス形式の例を示します。

表 16-3　メッセージサーバーアドレスの例

| サービス | ブローカホスト | ポート | アドレス例 |
|---|---|---|---|
| 未指定 | 未指定 | 未指定 | アドレスなし<br>(mq://localHost:7676/jms) |
| 未指定 | 指定したホスト | 未指定 | myBkrHost<br>(mq://myBkrHost:7676/jms) |
| 未指定 | 未指定 | 指定したポートマッパーポート | 1012<br>(mq://localHost:1012/jms) |
| ssljms | ローカルホスト | 標準のポートマッパーポート | mq://localHost:7676/ssljms |
| ssljms | 指定したホスト | 標準のポートマッパーポート | mq://myBkrHost:7676/ssljms |

表 16-3　メッセージサーバーアドレスの例 ( 続き )

| サービス | ブローカホスト | ポート | アドレス例 |
|---|---|---|---|
| ssljms | 指定したホスト | 指定したポートマッパーポート | mq://myBkrHost:1012/ssljms |
| jms | ローカルホスト | 指定したサービスポート | mqtcp://localhost:1032/jms |
| ssljms | 指定したホスト | 指定したサービスポート | mqssl://myBkrHost:1034/ssljms |
| httpjms | 該当なし | 該当なし | http://websrvr1:8085/imq/tunnel |
| httpsjms | 該当なし | 該当なし | https://websrvr2:8090/imq/tunnel |

# クライアントの識別

表 16-4 に、クライアント識別に関する接続ファクトリ属性を示します。

表 16-4　クライアントの識別に関する接続ファクトリ属性

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqDefaultUsername | 文字列 | guest | ブローカで認証するデフォルトユーザー名 |
| imqDefaultPassword | 文字列 | guest | ブローカで認証するデフォルトパスワード |
| imqConfiguredClientID | 文字列 | null | 管理用に設定したクライアント識別子 |
| imqDisableSetClientID | ブール | false | クライアントが setClientID メソッドを使用してクライアント識別子を変更できないようにするかどうか |

# 信頼性およびフロー制御

表 16-5 に、信頼性とフロー制御に関する接続ファクトリ属性を示します。

**表 16-5** 信頼性とフロー制御に関する接続ファクトリ属性

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqAckTimeout | 文字列 | 0 | 例外をスローするまでのブローカ通知の最大待ち時間 ( ミリ秒単位 )<br><br>値 0 は、タイムアウトがない、つまり無期限に待つことを示します。<br><br>**注 :** 状況によっては、値が低すぎると、タイムアウトが必要以上に早くなる可能性があります。たとえば、安全な (SSL) 接続を使用するときの LDAP ユーザーリポジトリに対する最初のユーザー認証には、30 秒以上かかることがあります。 |
| imqConnectionFlowCount | 整数 | 100 | 測定対象のバッチのペイロードメッセージ数<br><br>クライアントにこの数のペイロードメッセージを配信したあと、配信を一時停止します。これにより、累積された制御メッセージがある場合は、それを配信できます。ペイロードメッセージの配信は、クライアントランタイムから通知があったときに再開され、ふたたびこの数に達するまで続けられます。<br><br>値 0 では、メッセージ配信の測定が無効になります。この場合、ペイロードメッセージのトラフィックが多いと、Message Queue 制御メッセージがブロックされる可能性があります。 |
| imqConnectionFlowLimitEnabled | ブール | false | 接続レベルでメッセージフローを制限するかどうか |
| imqConnectionFlowLimit | 整数 | 1000 | 接続単位での、消費のために配信およびバッファリングするメッセージの最大数<br><br>imqConnectionFlowCount で制御されるフロー測定によって保留中の消費されていないペイロードメッセージ数がこの制限を越えたときに、特定の接続のメッセージ配信を停止します。保留中のメッセージの数が制限を下回ったときにのみ、配信を再開します。これにより、クライアントが、保留中のメッセージの処理で過負荷になってメモリー不足になるのを回避できます。<br><br>imqConnectionFlowLimitEnabled が false の場合、この属性は無視されます。 |

表 16-5　信頼性とフロー制御に関する接続ファクトリ属性 ( 続き )

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqConsumerFlowLimit | 整数 | 100 | コンシューマ単位での、消費のために配信およびバッファリングするメッセージの最大数<br><br>特定のコンシューマに対して保留中の消費されていないペイロードメッセージ数がこの制限を超えたときに、そのコンシューマへのメッセージ配信を停止します。そのコンシューマに対する保留中のメッセージの数が、imqConsumerFlowThreshold で指定される割合を下回ったときにのみ、配信を再開します。これを使用して、複数のコンシューマ間のロードバランスを向上させ、同じ接続上で 1 つのコンシューマがほかのコンシューマの通信を妨げるのを回避できます。<br><br>キュー独自の consumerFlowLimit 属性 ( 第 15 章「物理的送信先のプロパティーのリファレンス」を参照 ) に低い値を設定することで、この制限をオーバーライドできます。また、特定の接続上のすべてのコンシューマへのメッセージ配信は、imqConnectionFlowLimit で指定する全体の制限に従います。 |
| imqConsumerFlowThreshold | 整数 | 50 | コンシューマ単位での、クライアントランタイムでバッファリングされるメッセージ数。imqConsumerFlowLimit に対する割合で指定し、その数を下回るとメッセージ配信を再開します |

# キューブラウザとサーバーセッション

表 16-6 に、キュー検索とサーバーセッションに関する接続ファクトリ属性を示します。

表 16-6　キューブラウザとサーバーセッションに関する接続ファクトリ属性

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqQueueBrowserMaxMessagesPerRetrieve | 整数 | 1000 | キュー送信先の内容を検索するときに一度に取得できるメッセージの最大数 |
| imqQueueBrowserRetrieveTimeout | 倍長整数 | 60000 | キュー送信先の内容を検索するときに例外をスローするまでのメッセージ取得の最大待ち時間 ( ミリ秒単位 ) |

表 16-6　キューブラウザとサーバーセッションに関する接続ファクトリ属性 ( 続き )

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqLoadMaxToServerSession | ブール | true | サーバーセッションに対して最大数までのメッセージを読み込むかどうか<br>false に設定すると、クライアントは、1 回に 1 つのメッセージのみを読み込みます。<br>この属性は、JMS アプリケーションサーバー機能のみに適用されます。 |

# 標準メッセージプロパティーの設定

表 16-7 に示す接続ファクトリ属性は、『Java Message Service Specification』で定義されているいくつかの標準メッセージプロパティーを、Message Queue クライアントランタイムが設定するかどうかを制御します。

表 16-7　標準メッセージプロパティーに関する接続ファクトリ属性

| プロパティー | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqSetJMSXUserID | ブール | false | 生成されたメッセージに、メッセージを送信するユーザーの識別情報を示す JMSXUserID プロパティーを設定するかどうか |
| imqSetJMSXAppID | ブール | false | 生成されたメッセージに、メッセージを送信するアプリケーションの識別情報を示す JMSXAppID プロパティーを設定するかどうか |
| imqSetJMSXProducerTXID | ブール | false | 生成されたメッセージに、メッセージが生成されたトランザクションのトランザクション識別子を示す JMSXProducerTXID プロパティーを設定するかどうか |
| imqSetJMSXConsumerTXID | ブール | false | 消費されたメッセージに、メッセージが消費されたトランザクションのトランザクション識別子を示す JMSXConsumerTXID プロパティーを設定するかどうか |
| imqSetJMSXRcvTimestamp | ブール | false | 消費されたメッセージに、メッセージがコンシューマに配信された時刻を示す JMSXRcvTimestamp プロパティーを設定するかどうか |

# メッセージヘッダーのオーバーライド

表 16-8 に、JMS メッセージヘッダーフィールドのオーバーライドに関する接続ファクトリ属性を示します。

**表 16-8** メッセージヘッダーのオーバーライドに関する接続ファクトリ属性

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqOverrideJMSDeliveryMode | ブール | false | クライアントが設定した配信モードのオーバーライドを許可するかどうか |
| imqJMSDeliveryMode | 整数 | 2 | 配信モードのオーバーライドの値<br>1 持続性なし<br>2 持続的 |
| imqOverrideJMSExpiration | ブール | false | クライアントが設定した有効期限のオーバーライドを許可するかどうか |
| imqJMSExpiration | 倍長整数 | 0 | 有効期限のオーバーライドの値 ( ミリ秒単位 )<br>値 0 は、有効期限が無制限である、つまりメッセージが期限切れにならないことを示します。 |
| imqOverrideJMSPriority | ブール | false | クライアントが設定した優先度のレベルのオーバーライドを許可するかどうか |
| imqJMSPriority | 整数 | 4 ( 標準 ) | 優先度のレベルのオーバーライドの値 (0 ～ 9) |
| imqOverrideJMSHeadersToTemporaryDestinations | ブール | false | オーバーライドを一時的送信先に適用するかどうか |

# 送信先の属性

表 16-9 に、送信先管理対象オブジェクトに設定できる属性を示します。

**表 16-9　送信先の属性**

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqDestinationName | 文字列 | Untitled_Destination_Object | 物理的送信先の名前<br>送信先名は、スペースを含まない英数字だけを使用でき、英字、下線 (_)、またはドル記号 ($) 文字で始める必要があります。文字列 mq で始めることはできません。 |
| imqDestinationDescription | 文字列 | なし | 送信先の説明文字列 |

# SOAP の端点 (endpoint) の属性

表 16-10 に、SOAP (Simple Object Access Protocol) を使用するアプリケーションの端点の URL を設定するために使用する属性を示します。詳細については、『Message Queue Developer's Guide for Java Clients』を参照してください。

**表 16-10　SOAP の端点の属性**

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqSOAPEndpointList | 文字列 | なし | メッセージ送信先の SOAP の端点を表す 1 つ以上の URL のスペース区切りのリスト<br>それぞれの URL は、SOAP メッセージを受信および処理できるサーブレットに関連付けられている必要があります。<br>例 :<br>`http://www.serv1/ http://www.serv2/`<br>リストで複数の URL を指定すると、メッセージはそれらすべてにブロードキャストされます。 |
| imqEndpointName | 文字列 | Untitled_Endpoint_Object | SOAP の端点の名前 |

表 16-10　SOAP の端点の属性 ( 続き )

| 属性 | データ型 | デフォルト値 | 説明 |
|---|---|---|---|
| imqEndpointDescription | 文字列 | なし | SOAP の端点の説明文字列<br>例 :<br>`My endpoints for broadcast` |

第 17 章

# JMS リソースアダプタのプロパティーのリファレンス

Message Queue の JMS リソースアダプタ (JMS RA) では、標準的な J2EE コネクタアーキテクチャー (JCA) により、Sun Java System Message Queue を任意の J2EE 1.4 アプリケーションサーバーと統合できます。Message Queue の JMS リソースアダプタをアプリケーションサーバーに組み込むと、そのアプリケーションサーバーに配置したアプリケーションでは、Message Queue を使用して JMS メッセージを送受信できるようになります。

Message Queue の JMS リソースアダプタでは、次の 3 つの JavaBean コンポーネントを介して設定プロパティーが公開されます。

- `ResourceAdapter` 設定 : リソースアダプタ全体の動作に影響します。
- `ManagedConnectionFactory` 設定 : メッセージ駆動型 Bean (MDB) が使用するためにリソースアダプタが作成した接続に影響します。
- `ActivationSpec` 設定 : メッセージングシステムとのやり取りの中でメッセージ駆動型 Bean (MDB) を表すメッセージの終端に影響します。

これらのエンティティーのプロパティー値を設定するには、リソースアダプタの設定用と配置用、および MDB の配置用にアプリケーションサーバーによって提供されるツールを使用します。

この章では、Message Queue の JMS リソースアダプタの設定プロパティーを一覧表示して説明します。この章は、次の節から構成されています。

# ResourceAdapter JavaBean

ResourceAdapter 設定では、JMS リソースアダプタのデフォルト動作を設定します。表 17-1 で、この JavaBean を設定するためのプロパティーを一覧表示して説明します。必須プロパティーには脚注マークが付いています。

**表 17-1** リソースアダプタのプロパティー

| プロパティー | デフォルト値 | 説明 |
|---|---|---|
| addressList[1] | mq://localhost:7676/jms | リソースアダプタが Message Queue サービスに対して作成する接続。メッセージサービスのアドレス形式で指定します。<br><br>リソースアダプタによってデフォルト値が提供されます。<br><br>このプロパティー名 addressList は、Sun Java System Message Queue に固有ですが、標準プロパティー connectionURL と同じ意味です。Sun Java System Message Queue では、両方のプロパティー名が提供されます。connectionURL か addressList のどちらかを設定してください。この 2 つは同じものです。 |
| addressListBehavior | PRIORITY | リソースアダプタが Message Queue サービスに接続する方法を指定する文字列。値は、PRIORITY か RANDOM です。<br><br>PRIORITY 接続では、アドレスリスト addressList に指定した最初のものが選択されて Message Queue ブローカが選択されます。<br><br>RANDOM 接続では、アドレスリストから Message Queue ブローカがランダムに選択されます。<br><br>接続障害後の再接続は、PRIORITY と RANDOM で同じです。再接続の試行は、接続がエラーになったブローカから始まります。その試行がエラーになった場合、リソースアダプタはアクティブなアドレスリストを順番に処理します。 |
| addressListIterations | 1 | アドレスリストを繰り返す回数。この値は、最初の接続、およびその後の再接続の試行に適用されます。 |

**表 17-1** リソースアダプタのプロパティー ( 続き )

| プロパティー | デフォルト値 | 説明 |
|---|---|---|
| connectionURL | mq://localhost:7676/jms | リソースアダプタが Message Queue サービスに対して作成する接続。メッセージサービスのアドレス形式で指定します。<br><br>addressList プロパティーと同じです。詳細については、前述の説明を参照してください。 |
| userName[1] | guest | リソースアダプタが Message Queue サービスに接続するときに使用するデフォルトユーザー名。<br><br>リソースアダプタによってデフォルト値が提供されます。 |
| password[1] | guest | リソースアダプタが Message Queue サービスに接続するときに使用するデフォルトパスワード。<br><br>リソースアダプタによってデフォルト値が提供されます。 |
| reconnectAttempts | 6 | アドレスリストの 1 つのエントリに再接続を試す回数。reconnectEnabled を true に設定すると、このプロパティーが使用されます。 |
| reconnectEnabled | false | 接続障害後に再接続を試すかどうかを指定するブール値。<br><br>再接続の動作の試行は、reconnectInterval と reconnectAttempts の値によって制御されます。 |
| reconnectInterval | 30000 | 再接続を試す間隔 ( ミリ秒単位 )。reconnectEnabled を true に設定すると、このプロパティーが使用されます。 |

1. このプロパティーは必須です。

# ManagedConnectionFactory JavaBean

管理対象接続ファクトリでは、リソースアダプタがメッセージ駆動型 Bean に提供する接続の提供と定義を行います。ResourceAdapter JavaBean に類似の属性がある属性を設定した場合、その設定は、ResourceAdapter Bean に指定した類似値より優先されます。

表 17-2 で、Message Queue リソースアダプタによって提供される管理対象接続ファクトリについて設定可能な属性を一覧表示して説明します。

**表 17-2** 管理対象接続ファクトリの属性

| 属性 | デフォルト値 | 説明 |
|---|---|---|
| addressList | なし | この管理対象接続ファクトリから派生した接続のリスト。<br><br>この属性の形式は、342 ページの表 17-1 で説明したメッセージサービスの addressList と同じです。この値を設定しない場合は、前の表で説明した ResourceAdapter JavaBean に指定した addressList の値が接続で使用されます。 |
| addressListBehavior | PRIORITY | リソースアダプタが Message Queue サービスに接続する方法を指定する文字列。値は、PRIORITY か RANDOM です。<br><br>PRIORITY 接続では、アドレスリスト addressList に指定した最初のものが選択されて Message Queue ブローカが選択されます。<br><br>RANDOM 接続では、アドレスリストから Message Queue ブローカがランダムに選択されます。<br><br>接続障害後の再接続は、PRIORITY と RANDOM で同じです。再接続の試行は、接続がエラーになったブローカから始まります。その試行がエラーになった場合、その接続ではアクティブなアドレスリストが順番に処理されます。 |
| addressListIterations | 1 | アドレスリストを繰り返す回数。この値は、最初の接続、およびその後の再接続の試行に適用されます。 |
| clientID | なし | この管理対象接続ファクトリから派生した接続に使用するクライアント識別子。 |
| password | guest | ( 任意指定 ) 接続のパスワード。<br><br>この値を設定しない場合は、342 ページの表 17-1 で説明した ResourceAdapter JavaBean に指定したパスワードが接続で使用されます。 |
| reconnectAttempts | 6 | アドレスリストの 1 つのエントリに再接続を試す回数。 |

表 17-2　管理対象接続ファクトリの属性 ( 続き )

| 属性 | デフォルト値 | 説明 |
|---|---|---|
| reconnectEnabled | false | 接続の障害後に再接続を試すか新しい接続を試すかを指定するブール値。<br>再接続の試行は、reconnectInterval 属性と reconnectAttempts 属性によって制御されます。 |
| reconnectInterval | 30000 | Message Queue サービスへの再接続を試すまでに待機する最小ミリ秒数。 |
| userName | guest | ( 任意指定 ) 接続のユーザー名。<br>この値を設定しない場合は、342 ページの表 17-1 で説明した ResourceAdapter JavaBean に指定したユーザー名が接続で使用されます。 |

# ActivationSpec JavaBean

アプリケーションサーバーは、ActivationSpec JavaBean のプロパティーを使用して、メッセージ終端をアクティブにしてメッセージ終端とメッセージ駆動型 Bean を関連付けるようにリソースアダプタに命令します。

表 17-3 で、メッセージ終端のアクティブ化仕様について設定可能な属性を一覧表示して説明します。この表では、Message Queue のリソースアダプタに固有のプロパティー、および Enterprise JavaBean 2.1 標準か J2EE Connector Architecture (J2EE CA) 1.5 標準に固有のプロパティーについて説明します。

表 17-3　アクティブ化仕様のプロパティー

| プロパティー | デフォルト値 | 説明 |
|---|---|---|
| acknowledgeMode | Auto-acknowledge | ( 任意指定 ) コンシューマに使用する JMS セッション通知モード。<br>これは、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティーです。<br>値は、Auto-acknowledge か Dups-ok-acknowledge にすることができます。 |

**表 17-3** アクティブ化仕様のプロパティー ( 続き )

| プロパティー | デフォルト値 | 説明 |
|---|---|---|
| addressList | ResourceAdapter JavaBean 設定の addressList から継承 | ( 任意指定 ) メッセージ終端に代わってリソースアダプタが作成する接続の仕様。<br>このプロパティーは Message Queue JMS リソースアダプタに固有です。<br>有効な値は、メッセージサービスの接続アドレス構文に従う必要があります。 |
| clientId | なし | このコンシューマ用に作成された JMS 接続によって使用される JMS クライアント ID。<br>subscriptionDurability プロパティーを Durable に設定した場合は、このプロパティーを設定する必要があります。<br>これは、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティーです。 |
| customAcknowledgeMode | なし | MDB メッセージの消費のモードを指定する文字列。<br>このプロパティーの有効な値は、No_acknowledge か NULL です。<br>No_acknowledge モードは、処理済みでも永続的でもないトピックサブスクリプションのみに使用できます。処理済みサブスクリプションか永続サブスクリプションでこの設定を使用すると、サブスクリプションのアクティブ化はエラーになります。 |
| destination | なし | この MDB がメッセージを消費する送信先の名前。<br>これは必須のプロパティーです。標準的な EJB 2.1 と J2EE CA 1.5 のプロパティーです。<br>Message Queue の送信先管理対象オブジェクトの destinationName プロパティーの値に設定する必要があります。 |
| destinationType | なし | destination プロパティーで指定した送信先のタイプ。有効な値は、javax.jms.Queue か javax.jms.Topic です。<br>これは必須のプロパティーです。標準的な EJB 2.1 と J2EE CA 1.5 のプロパティーです。 |
| endpointExceptionRedelivery Attempts | 6 | メッセージ配信中に MDB で例外がスローされたとき、MDB にメッセージを再配信する回数。 |

**表 17-3** アクティブ化仕様のプロパティー ( 続き )

| プロパティー | デフォルト値 | 説明 |
|---|---|---|
| messageSelector | なし | ( 任意指定 ) コンシューマに配信されるメッセージのフィルタリングに使用する JMS メッセージセレクタ。値のタイプは String です。<br><br>これは、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティーです。 |
| sendUndeliverableMsgsToDMQ | true | MDB で実行時例外がスローされ、再配信回数が endpointExceptionRedeliveryAttempts の値を超えたとき、デッドメッセージキューにメッセージを配置するかどうかを指定するブール値。<br><br>false に設定した場合、Message Queue ブローカは、同一 MDB も含めた有効なコンシューマにメッセージを再配信しようとします。 |
| subscriptionDurability | NonDurable | トピック送信先のコンシューマが永続的であるかどうかを指定する文字列。値は、NonDurable か Durable にすることができます。<br><br>このプロパティーは、永続的でないサブスクリプションでは任意指定であり、永続サブスクリプションでは必須です。この値を Durable に設定した場合は、clientID プロパティーと subscriptionName プロパティーも設定する必要があります。<br><br>標準的な EJB 2.1 と J2EE CA1.5 のプロパティーであり、destinationType プロパティーを avax.jms.Topic に設定した場合にのみ有効です。 |
| subscriptionName | なし | 永続サブスクリプションの指定に使用する文字列。<br><br>subscriptionDurability プロパティーを Durable に設定した場合は、このプロパティーを設定する必要があります。<br><br>これは、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティーです。 |

第 18 章

# メトリックスのリファレンス

この章では、Message Queue 製品によって生成されるメトリックスを一覧表示して説明します。この章では、次の節について説明します。



## JVM メトリックス

表 18-1 では、ブローカプロセスの JVM ヒープについてブローカが生成するメトリックスデータを一覧表示して説明します。各メトリックスについて、どのメトリックス監視ツールでそれが提供されるかも示します。

表 18-1　JVM メトリックス

| メトリックス量 | 説明 | imqcmd **metrics bkr** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*)[2] |
|---|---|---|---|---|
| JVM heap: free memory | JVM ヒープに使用可能な空きメモリー量 | あり (cxn) | あり | あり (...jvm) |
| JVM heap: total memory | 現在の JVM ヒープサイズ | あり (cxn) | あり | あり (...jvm) |
| JVM heap: max memory | JVM ヒープサイズを拡大可能な上限 | なし | あり[1] | あり (...jvm) |

1. ブローカの起動時にだけ表示されます。
2. メトリックストピック送信先名については、207 ページの表 10-7 を参照してください。

# ブローカ全体のメトリックス

表 18-2 では、ブローカ全体のメトリックス情報についてブローカが報告するデータを一覧表示して説明します。また、各種メトリックス監視ツールを使用してどのデータを取得できるかも示しています。

表 18-2　ブローカ全体のメトリックス

| メトリックス量 | 説明 | imqcmd **metrics bkr** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*)[1] |
|---|---|---|---|---|
| **接続データ** | | | | |
| Num connections | 現在開いているブローカへの接続の数 | あり (cxn) | あり | あり (...broker) |
| Num threads | すべての接続サービスで現在使用されているスレッドの合計数 | あり (cxn) | あり | なし |
| Min threads | 接続サービスで使用するために、スレッドプールに保持している作成済みのスレッド数 | あり (cxn) | あり | なし |
| Max threads | 接続サービスで使用するために、スレッドプールに保持するスレッドの最大数。新しいスレッドは、それ以上追加されなくなります | あり (cxn) | あり | なし |
| **格納済みのメッセージデータ** | | | | |
| Num messages | 現在、ブローカのメモリーと持続ストアに格納されているペイロードメッセージの数 | なし query bkr を使用 | なし | あり (...broker) |
| Total message bytes | 現在、ブローカのメモリーと持続ストアに格納されているペイロードメッセージのバイト数 | なし query bkr を使用 | なし | あり (...broker) |
| **メッセージフローデータ** | | | | |
| Num messages in | ブローカが最後に起動された以降にブローカで受信したペイロードメッセージの数 | あり (ttl) | あり | あり (...broker) |
| Message bytes in | ブローカが最後に起動された以降にブローカで受信したペイロードメッセージバイト数 | あり (ttl) | あり | あり (...broker) |

表 18-2　ブローカ全体のメトリックス ( 続き )

| メトリックス量 | 説明 | imqcmd **metrics bkr** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*)[1] |
|---|---|---|---|---|
| Num packets in | ブローカが最後に起動された以降にブローカで受信したパケットの数。ペイロードメッセージと制御メッセージの両方が含まれます | あり (ttl) | あり | あり (...broker) |
| Packet bytes in | ブローカが最後に起動された以降にブローカで受信したパケットバイト数。ペイロードメッセージと制御メッセージの両方が含まれます | あり (ttl) | あり | あり (...broker) |
| Num messages out | ブローカが最後に起動された以降にブローカから送信したペイロードメッセージの数 | あり (ttl) | あり | あり (...broker) |
| Message bytes out | ブローカが最後に起動された以降にブローカから送信したペイロードメッセージバイト数 | あり (ttl) | あり | あり (...broker) |
| Num packets out | ブローカが最後に起動された以降にブローカから送信したパケットの数。ペイロードメッセージと制御メッセージの両方が含まれます | あり (ttl) | あり | あり (...broker) |
| Packet bytes out | ブローカが最後に起動された以降にブローカから送信したパケットバイト数。ペイロードメッセージと制御メッセージの両方が含まれます | あり (ttl) | あり | あり (...broker) |
| Rate messages in | ブローカで受信するペイロードメッセージの現在のフローレート | あり (rts) | あり | なし |
| Rate message bytes in | ブローカで受信するペイロードメッセージバイト数の現在のフローレート | あり (rts) | あり | なし |
| Rate packets in | ブローカで受信するパケットの現在のフローレート。ペイロードメッセージと制御メッセージの両方を含みます | あり (rts) | あり | なし |

表 18-2　ブローカ全体のメトリックス ( 続き )

| メトリックス量 | 説明 | imqcmd **metrics bkr** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*)[1] |
|---|---|---|---|---|
| Rate packet bytes in | ブローカで受信するパケットバイト数の現在のフローレート。ペイロードメッセージと制御メッセージの両方を含みます | あり (rts) | あり | なし |
| Rate messages out | ブローカから送信するペイロードメッセージの現在のフローレート | あり (rts) | あり | なし |
| Rate message bytes out | ブローカから送信するペイロードメッセージバイト数の現在のフローレート | あり (rts) | あり | なし |
| Rate packets out | ブローカから送信するパケットの現在のフローレート。ペイロードメッセージと制御メッセージの両方を含みます | あり (rts) | あり | なし |
| Rate packet bytes out | ブローカから送信するパケットバイト数の現在のフローレート。ペイロードメッセージと制御メッセージの両方を含みます | あり (rts) | あり | なし |
| **送信先データ** | | | | |
| Num destinations | ブローカ内の物理的な送信先の数 | なし | なし | あり (...broker) |

1. メトリックストピック送信先名については、207 ページの表 10-7 を参照してください。

# 接続サービスのメトリックス

表 18-3 では、個々の接続サービスについてブローカが報告するメトリックスデータを一覧表示して説明します。また、各種メトリックス監視ツールを使用してどのデータを取得できるかも示しています。

表 18-3　接続サービスのメトリックス

| メトリックス量 | 説明 | imqcmd **metrics svc** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*) |
|---|---|---|---|---|
| **接続データ** | | | | |

表 18-3　接続サービスのメトリックス ( 続き )

| メトリックス量 | 説明 | imqcmd **metrics svc** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*) |
|---|---|---|---|---|
| Num connections | 現在開かれている接続数 | あり (cxn) query svc でも可 | なし | なし |
| Num threads | 現在使用中のスレッドの数 | あり (cxn) query svc でも可 | なし | なし |
| Min threads | すべての接続サービスで、接続サービスが使用するスレッドプールに初めに保持されるスレッドの合計数 | あり (cxn) | なし | なし |
| Max threads | すべての接続サービスで、接続サービスが使用するスレッドプールに保持されるスレッドの最大合計数。新しいスレッドは、それ以上追加されなくなります | あり (cxn) | なし | なし |
| **メッセージフローデータ** | | | | |
| Num messages in | ブローカが最後に起動された以降に接続サービスで受信したペイロードメッセージの数 | あり (ttl) | なし | なし |
| Message bytes in | ブローカが最後に起動された以降に接続サービスで受信したペイロードメッセージバイト数 | あり (ttl) | なし | なし |
| Num packets in | ブローカが最後に起動された以降に接続サービスで受信したパケットの数。ペイロードメッセージと制御メッセージの両方が含まれます | あり (ttl) | なし | なし |
| Packet bytes in | ブローカが最後に起動された以降に接続サービスで受信したパケットバイト数。ペイロードメッセージと制御メッセージの両方が含まれます | あり (ttl) | なし | なし |
| Num messages out | ブローカが最後に起動された以降に接続サービスから送信したペイロードメッセージの数 | あり (ttl) | なし | なし |

表 18-3　接続サービスのメトリックス ( 続き )

| メトリックス量 | 説明 | imqcmd **metrics svc** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*) |
|---|---|---|---|---|
| Message bytes out | ブローカが最後に起動された以降に接続サービスから送信したペイロードメッセージバイト数 | あり (ttl) | なし | なし |
| Num packets out | ブローカが最後に起動された以降に接続サービスから送信したパケットの数。ペイロードメッセージと制御メッセージの両方が含まれます | あり (ttl) | なし | なし |
| Packet bytes out | ブローカが最後に起動された以降に接続サービスから送信したパケットバイト数。ペイロードメッセージと制御メッセージの両方が含まれます | あり (ttl) | なし | なし |
| Rate messages in | 接続サービス経由でブローカが受信するペイロードメッセージの現在のフローレート | あり (rts) | なし | なし |
| Rate message bytes in | 接続サービスで受信するペイロードメッセージバイト数の現在のフローレート | あり (rts) | なし | なし |
| Rate packets in | 接続サービスで受信するパケットの現在のフローレート。ペイロードメッセージと制御メッセージの両方を含みます | あり (rts) | なし | なし |
| Rate packet bytes in | 接続サービスで受信するパケットバイト数の現在のフローレート。ペイロードメッセージと制御メッセージの両方を含みます | あり (rts) | なし | なし |
| Rate messages out | 接続サービスから送信するペイロードメッセージの現在のフローレート | あり (rts) | なし | なし |
| Rate message bytes out | 接続サービスから送信するペイロードメッセージバイト数の現在のフローレート | あり (rts) | なし | なし |
| Rate packets out | 接続サービスから送信するパケットの現在のフローレート。ペイロードメッセージと制御メッセージの両方を含みます | あり (rts) | なし | なし |

表 18-3　接続サービスのメトリックス ( 続き )

| メトリックス量 | 説明 | imqcmd **metrics svc** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*) |
|---|---|---|---|---|
| Rate packet bytes out | 接続サービスから送信するパケットバイト数の現在のフローレート。ペイロードメッセージと制御メッセージの両方を含みます | あり (`rts`) | なし | なし |

# 送信先メトリックス

表 18-4 では、個々の送信先についてブローカが報告するメトリックスデータを一覧表示して説明します。また、各種メトリックス監視ツールを使用してどのデータを取得できるかも示しています。

表 18-4　送信先メトリックス

| メトリックス量 | 説明 | imqcmd **metrics dst** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*)[1] |
|---|---|---|---|---|
| **コンシューマのデータ** | | | | |
| Num consumers | 現在のコンシューマの数<br>トピックの場合、この値には、永続的でないサブスクリプション、アクティブな永続サブスクリプション、アクティブでない永続サブスクリプションが含まれます。キューの場合、この値には、アクティブなコンシューマとバックアップコンシューマが含まれます。 | あり (`con`) | なし | あり (*...destName*) |
| Avg num consumers | ブローカが最後に起動された以降のコンシューマの数の平均 | あり (`con`) | なし | あり (*...destName*) |
| Peak num consumers | ブローカが最後に起動された以降のコンシューマの数のピーク | あり (`con`) | なし | あり (*...destName*) |
| Num active consumers | 現在のアクティブなコンシューマの数 | あり (`con`) | なし | あり (*...destName*) |
| Avg num active consumers | ブローカが最後に起動された以降のアクティブなコンシューマの数の平均 | あり (`con`) | なし | あり (*...destName*) |

表 18-4　送信先メトリックス ( 続き )

| メトリックス量 | 説明 | imqcmd **metrics dst** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*)[1] |
|---|---|---|---|---|
| Peak num active consumers | ブローカが最後に起動された以降のアクティブなコンシューマの最大時の数 | あり<br>(con) | なし | あり<br>(...*destName*) |
| Num backup consumers | 現在のバックアップコンシューマの数。キューにだけ適用されます | あり<br>(con) | なし | あり<br>(...*destName*) |
| Avg num backup consumers | ブローカが最後に起動された以降のバックアップコンシューマ数の平均。キューにだけ適用されます | あり<br>(con) | なし | あり<br>(...*destName*) |
| Peak num backup consumers | ブローカが最後に起動された以降のバックアップコンシューマの最大時の数。キューにだけ適用されます | あり<br>(con) | なし | あり<br>(...*destName*) |
| **格納済みのメッセージデータ** | | | | |
| Num messages | 現在、送信先メモリーと持続ストアに格納されているペイロードメッセージの数 | あり<br>(con)<br>(ttl)<br>(rts)<br>query dst でも可 | なし | あり<br>(...*destName*) |
| Avg num messages | ブローカが最後に起動された以降に送信先メモリーと持続ストアに格納されたペイロードメッセージ数の平均 | あり<br>(con)<br>(ttl)<br>(rts) | なし | あり<br>(...*destName*) |
| Peak num messages | ブローカが最後に起動された以降に送信先メモリーと持続ストアに格納されたペイロードメッセージの最大時の数 | あり<br>(con)<br>(ttl)<br>(rts) | なし | あり<br>(...*destName*) |
| Total message bytes | 現在、送信先メモリーと持続ストアに格納されているペイロードメッセージのバイト数 | あり<br>(ttl)<br>(rts)<br>query dst でも可 | なし | あり<br>(...*destName*) |
| Avg total message bytes | ブローカが最後に起動された以降に送信先メモリーと持続ストアに格納されたペイロードメッセージバイト数の平均 | あり<br>(ttl)<br>(rts) | なし | あり<br>(...*destName*) |

表 18-4　送信先メトリックス ( 続き )

| メトリックス量 | 説明 | imqcmd **metrics dst** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*)[1] |
|---|---|---|---|---|
| Peak total message bytes | ブローカが最後に起動された以降に送信先メモリーと持続ストアに格納されたペイロードメッセージの最大時のバイト数 | あり (ttl) (rts) | なし | あり (...*destName*) |
| Peak message bytes | ブローカが最後に起動された以降に送信先が受信した、単一メッセージ内のペイロードメッセージの最大時のバイト数 | あり (ttl) (rts) | なし | あり (...*destName*) |
| **メッセージフローデータ** | | | | |
| Num messages in | ブローカが最後に起動された以降にこの送信先で受信したペイロードメッセージの数 | あり (ttl) | なし | あり (...*destName*) |
| Msg bytes in | ブローカが最後に起動された以降にこの送信先で受信したペイロードメッセージバイト数 | あり (ttl) | なし | あり (...*destName*) |
| Num messages out | ブローカが最後に起動された以降にこの送信先から送信したペイロードメッセージの数 | あり (ttl) | なし | あり (...*destName*) |
| Msg bytes out | ブローカが最後に起動された以降にこの送信先から送信したペイロードメッセージバイト数 | あり (ttl) | なし | あり (...*destName*) |
| Rate num messages in | 送信先で受信するペイロードメッセージの現在のフローレート | あり (rts) | なし | なし |
| Rate num messages out | 送信先から送信するペイロードメッセージの現在のフローレート | あり (rts) | なし | なし |
| Rate msg bytes in | 送信先で受信するペイロードメッセージバイト数の現在のフローレート | あり (rts) | なし | なし |
| Rate Msg bytes out | 送信先から送信するペイロードメッセージバイト数の現在のフローレート | あり (rts) | なし | なし |

表 18-4　送信先メトリックス ( 続き )

| メトリックス量 | 説明 | imqcmd **metrics dst** (*metricType*) | ログファイル | メトリックスメッセージ (*metrics topic*)[1] |
|---|---|---|---|---|
| **ディスク利用率データ** | | | | |
| Disk reserved | メッセージレコードが使用する、送信先のファイルベースのストアにある、アクティブレコードと空きレコードを含むすべてのメッセージレコードによって使用されているディスクスペース ( バイト単位 ) | あり (`dsk`) | なし | あり (...*destName*) |
| Disk used | 送信先のファイルベースのストアにあるアクティブメッセージレコードによって使用されているディスクスペース ( バイト単位 ) | あり (`dsk`) | なし | あり (...*destName*) |
| Disk utilization ratio | 予約済みのディスクスペースに対する、使用されているディスクスペースの割合。割合が高いほど、アクティブメッセージを保持するためにより多くのディスクスペースが使用されています | あり (`dsk`) | なし | あり (...*destName*) |

1. メトリックストピック送信先名については、207 ページの表 10-7 を参照してください。

第 IV 部

# 付録

# プラットフォームごとの Message Queue データの場所

Sun Java System Message Queue データは、オペレーティングシステムのプラットフォームによって異なる場所に格納されます。以降の表に、次のプラットフォームでの各種の Message Queue データの場所を示します。


- 361 ページの「Solaris」
- 362 ページの「Linux」
- 363 ページの「Windows」


表の中の *instanceName* は、データが関連付けられているブローカインスタンスの名前を示します。

## Solaris

表 A-1 は、Solaris オペレーティングシステム上での Message Queue データの場所を示しています。Solaris でスタンドアロンバージョンの Sun Java System Application Server とともに Message Queue を使用している場合は、363 ページの「Windows」で説明するディレクトリ構造と同様になります。

**表 A-1** Solaris プラットフォームでの Message Queue データの場所

| データのカテゴリ | 場所 |
|---|---|
| ブローカインスタンスの設定プロパティー | `/var/imq/instances/`*instanceName*`/props/config.properties` |
| ブローカ設定ファイルのテンプレート | `/usr/share/lib/imq/props/broker/` |
| 持続ストア ( メッセージ、送信先、永続サブスクリプション、トランザクション ) | `/var/imq/instances/`*instanceName*`/fs350` または JDBC のアクセス可能なデータストア |

表 A-1　Solaris プラットフォームでの Message Queue データの場所 ( 続き )

| データのカテゴリ | 場所 |
|---|---|
| ブローカインスタンスのログファイルのディレクトリ ( デフォルトの場所 ) | /var/imq/instances/*instanceName*/log/ |
| 管理対象オブジェクト ( オブジェクトストア ) | 任意のローカルディレクトリ、または LDAP サーバー |
| セキュリティー : ユーザーリポジトリ | /var/imq/instances/*instanceName*/etc/passwd または LDAP サーバー |
| セキュリティー : アクセス制御ファイル ( デフォルトの場所 ) | /var/imq/instances/*instanceName*/etc/accesscontrol.properties |
| セキュリティー : パスワードファイルのディレクトリ ( デフォルトの場所 ) | /var/imq/instances/*instanceName*/etc/ |
| セキュリティー : パスワードファイルの例 | /etc/imq/passfile.sample |
| セキュリティー : ブローカのキーストアファイルの場所 | /etc/imq/ |
| JavaDoc API のマニュアル | /usr/share/javadoc/imq/index.html |
| アプリケーションと設定の例 | /usr/demo/imq/ |
| Java アーカイブ (.jar)、Web アーカイブ (.war)、およびリソースアダプタアーカイブ (.rar) の各ファイル | /usr/share/lib/ |

# Linux

表 A-2 は、Linux オペレーティングシステム上での Message Queue データの場所を示しています。

表 A-2　Linux プラットフォームでの Message Queue データの場所

| データのカテゴリ | 場所 |
|---|---|
| ブローカインスタンスの設定プロパティー | /var/opt/sun/mq/instances/*instanceName*/props/config.properties |
| ブローカ設定ファイルのテンプレート | /opt/sun/mq/private/share/lib/props/ |
| 持続ストア ( メッセージ、送信先、永続サブスクリプション、トランザクション ) | /var/opt/sun/mq/instances/*instanceName*/fs350/ または JDBC のアクセス可能なデータストア |
| ブローカインスタンスのログファイルのディレクトリ ( デフォルトの場所 ) | /var/opt/sun/mq/instances/*instanceName*/log/ |
| 管理対象オブジェクト ( オブジェクトストア ) | 任意のローカルディレクトリ、または LDAP サーバー |

表 A-2　Linux プラットフォームでの Message Queue データの場所 ( 続き )

| データのカテゴリ | 場所 |
|---|---|
| セキュリティー : ユーザーリポジトリ | /var/opt/sun/mq/instances/*instanceName*/etc/passwd または LDAP サーバー |
| セキュリティー : アクセス制御ファイル ( デフォルトの場所 ) | /var/opt/sun/mq/instances/*instanceName*/etc/accesscontrol.properties |
| セキュリティー : パスワードファイルのディレクトリ ( デフォルトの場所 ) | /var/opt/sun/mq/instances/*instanceName*/etc/ |
| セキュリティー : パスワードファイルの例 | /etc/opt/sun/mq/passfile.sample |
| セキュリティー : ブローカのキーストアファイルの場所 | /etc/opt/sun/mq/ |
| JavaDoc API のマニュアル | /opt/sun/mq/javadoc/index.html |
| アプリケーションと設定の例 | /opt/sun/mq/examples/ |
| Java アーカイブ (.jar)、Web アーカイブ (.war)、およびリソースアダプタアーカイブ (.rar) の各ファイル | /opt/sun/mq/share/lib/ |
| 共有ライブラリ (.so) ファイル | /opt/sun/mq/lib/ |

# Windows

表 A-3 は、Windows オペレーティングシステム上での Message Queue データの場所を示しています。Solaris プラットフォームでも、Message Queue がスタンドアロンバージョンの Sun Java System Application Server にバンドルされている場合は、この表が適用されます。スタンドアロンバージョンの Application Server は、Solaris と Sun Java Enterprise System にはバンドルされません。表 A-3 のパス名では、Windows の円記号 (¥) を Solaris のスラッシュ (/) に変更してください。IMQ_HOME および IMQ_VARHOME ディレクトリ変数の定義については、22 ページの「ディレクトリ変数の表記規則」を参照してください。

表 A-3　Windows プラットフォームでの Message Queue データの場所

| データのカテゴリ | 場所 |
|---|---|
| ブローカインスタンスの設定プロパティー | IMQ_VARHOME¥instances¥*instanceName*¥props¥config.properties |
| ブローカ設定ファイルのテンプレート | IMQ_HOME¥lib¥props¥broker¥ |
| 持続ストア ( メッセージ、送信先、永続サブスクリプション、トランザクション ) | IMQ_VARHOME¥instances¥*instanceName*¥fs350¥ または JDBC のアクセス可能なデータストア |

表 A-3　Windows プラットフォームでの Message Queue データの場所 ( 続き )

| データのカテゴリ | 場所 |
|---|---|
| ブローカインスタンスのログファイルのディレクトリ ( デフォルトの場所 ) | IMQ_VARHOME¥instances¥*instanceName*¥log¥ |
| 管理対象オブジェクト ( オブジェクトストア ) | 任意のローカルディレクトリ、または LDAP サーバー |
| セキュリティー : ユーザーリポジトリ | IMQ_VARHOME¥instances¥*instanceName*¥etc¥passwd または LDAP サーバー |
| セキュリティー : アクセス制御ファイル ( デフォルトの場所 ) | IMQ_VARHOME¥instances¥*instanceName*¥etc¥accesscontrol.properties |
| セキュリティー : パスワードファイルのディレクトリ ( デフォルトの場所 ) | IMQ_HOME¥etc¥ |
| セキュリティー : パスワードファイルの例 | IMQ_HOME¥etc¥passfile.sample |
| セキュリティー : ブローカのキーストアファイルの場所 | IMQ_HOME¥etc¥ |
| JavaDoc API のマニュアル | IMQ_HOME¥javadoc¥index.html |
| アプリケーションと設定の例 | IMQ_HOME¥demo¥ |
| Java アーカイブ (.jar)、Web アーカイブ (.war)、およびリソースアダプタアーカイブ (.rar) の各ファイル | IMQ_HOME¥lib¥ |

# Message Queue インタフェースの安定度

Sun Java System Message Queue では多くのインタフェースが使用されるので、管理者はタスクを自動化できます。この付録では、安定度によってインタフェースを分類します。インタフェースの安定度が高くなるほど、製品の今後のバージョンで変更される可能性が低くなります。

この付録に掲載されないインタフェースは非公開であり、お客様は使用できません。

表 B-1 では、安定度分類方式について説明します。

**表 B-1** インタフェースの安定度の分類方式

| 分類 | 説明 |
|---|---|
| 非公開 | ユーザーは直接使用しません。リリースによって変更される、あるいは削除される可能性があります。 |
| 発展中 | ユーザーが使用します。メジャーリリース (3.0 や 4.0 など )、またはマイナーリリース (3.1 や 3.2 など ) で、互換性のない変更が生じる可能性があります。変更は慎重に、かつ徐々に行われます。すべての変更について、互換性が保てるように十分な努力が払われますが、保証はされていません。 |
| 安定 | ユーザーが使用します。互換性のない変更は、メジャーリリース (3.0 や 4.0 など ) でしか生じません。 |
| 標準 | ユーザーが使用します。これらのインタフェースは、形式標準によって定義され、標準組織によって制御されます。これらのインタフェースでは、互換性のない変更はめったにありません。 |
| 不安定 | ユーザーが使用します。メジャーリリース (3.0 や 4.0 など )、またはマイナーリリース (3.1 や 3.2 など ) で、互換性のない変更が生じる可能性があります。これらのインタフェースは、今後のリリースで、互換性のない方法でかなりの削除や変更が行われる可能性があることに注意してください。不安定なインタフェースでは、明示的な依存関係を作成しないでください。 |

表 B-2 は、インタフェースとその分類の一覧です。

表 B-2　Message Queue インタフェースの安定度

| インタフェース | 分類 |
|---|---|
| **コマンド行インタフェース** | |
| imqbrokerd コマンド行インタフェース | 発展中 |
| imqadmin コマンド行インタフェース | 不安定 |
| imqcmd コマンド行インタフェース | 発展中 |
| imqdbmgr コマンド行インタフェース | 不安定 |
| imqkeytool コマンド行インタフェース | 発展中 |
| imqobjmgr コマンド行インタフェース | 発展中 |
| imqusermgr コマンド行インタフェース | 不安定 |
| imqbrokerd、imqadmin、imqcmd、imqdbmgr、imqkeytool、imqobjmgr、imqusermgr からの出力 | 不安定 |
| **コマンド** | |
| imqobjmgr コマンドファイル | 発展中 |
| imqbrokerd コマンド | 安定 |
| imqadmin コマンド | 不安定 |
| imqcmd コマンド | 安定 |
| imqdbmgr コマンド | 不安定 |
| imqkeytool コマンド | 安定 |
| imqobjmgr コマンド | 安定 |
| imqusermgr コマンド | 不安定 |
| **API** | |
| JMS API (javax.jms) | 標準 |
| JAXM API (javax.xml) | 標準 |
| C-API | 発展中 |
| C-API 環境変数 | 不安定 |
| メッセージベースの監視 API | 発展中 |
| 管理対象オブジェクト API (com.sun.messaging) | 発展中 |
| **.jar および .war ファイル** | |

表 B-2　Message Queue インタフェースの安定度 ( 続き )

| インタフェース | 分類 |
|---|---|
| imq.jar の格納場所および名前 | 安定 |
| jms.jar の格納場所および名前 | 発展中 |
| imqbroker.jar の格納場所および名前 | 非公開 |
| imqutil.jar の格納場所および名前 | 非公開 |
| imqadmin.jar の格納場所および名前 | 非公開 |
| imqservlet.jar の格納場所および名前 | 発展中 |
| imqhttp.war の格納場所および名前 | 発展中 |
| imqhttps.war の格納場所および名前 | 発展中 |
| imqjmsra.rar の格納場所および名前 | 発展中 |
| imqxm.jar の格納場所および名前 | 発展中 |
| jaxm-api.jar の格納場所および名前 | 発展中 |
| saaj-api.jar の格納場所および名前 | 発展中 |
| saaj-impl.jar の格納場所および名前 | 発展中 |
| activation.jar の格納場所および名前 | 発展中 |
| mail.jar の格納場所および名前 | 発展中 |
| dom4j.jar の格納場所および名前 | 非公開 |
| fscontext.jar の格納場所および名前 | 不安定 |
| **ファイル** | |
| ブローカログファイルの格納場所および内容形式 | 不安定 |
| パスワードファイル | 不安定 |
| accesscontrol.properties ファイル | 不安定 |
| **システム送信先** | |
| mq.sys.dmq 送信先 | 安定 |
| mq.metrics.* 送信先 | 発展中 |
| **設定プロパティー** | |
| Message Queue JMS リソースアダプタの設定プロパティー | 発展中 |
| Message Queue JMS リソースアダプタの JavaBean と ActivationSpec の設定プロパティー | 発展中 |

表 B-2　Message Queue インタフェースの安定度 ( 続き )

| インタフェース | 分類 |
|---|---|
| **メッセージのプロパティーと形式** | |
| デッドメッセージキューのメッセージプロパティー、JMSXDeliveryCount | 標準 |
| デッドメッセージキューのメッセージプロパティー、JMS_SUN_* | 発展中 |
| Message Queue クライアントメッセージプロパティー : JMS_SUN_* | 発展中 |
| メトリックスまたは監視メッセージの JMS メッセージ形式 | 発展中 |
| **その他** | |
| Message Queue JMS リソースアダプタパッケージ、com.sun.messaging.jms.ra | 発展中 |
| 持続メッセージの保存の JDBC スキーマ | 発展中 |

# HTTP/HTTPS のサポート

Message Queue の Enterprise Edition では、Java クライアントが、TCP 接続を直接使用するのではなく、HTTP またはセキュリティー保護された HTTP (HTTPS) 転送を使用してブローカと通信するようにサポートされています。C クライアントでは、HTTP/HTTPS がサポートされません。

この付録では、このようなサポートを有効にするために使用されるアーキテクチャーについて説明し、クライアントが Message Queue メッセージングに HTTP ベースの接続を使用するために必要となる設定の手順を示します。この付録は、次の節から構成されています。



## HTTP/HTTPS サポートのアーキテクチャー

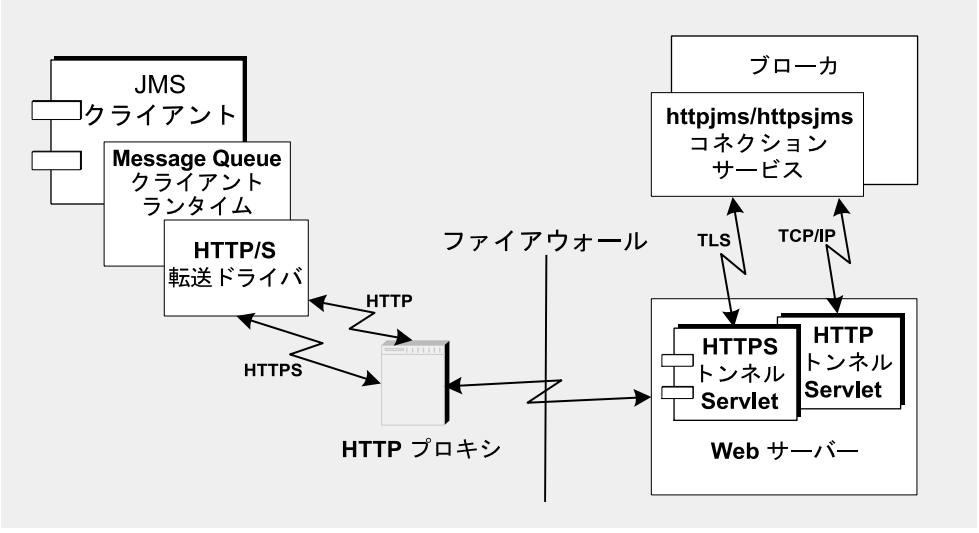
Message Queue メッセージングは、HTTP/HTTPS 接続で実行できます。HTTP/HTTPS 接続は、通常ファイアウォールの通過が許可されるため、ファイアウォールによってブローカからクライアントアプリケーションを分離できます。

370 ページの図 C-1 に、HTTP/HTTPS サポートの提供に関連する主なコンポーネントを示します。

- クライアント側では、HTTP または HTTPS トランスポートドライバが、Message Queue メッセージを HTTP 要求にカプセル化し、これらの要求が正しい手順で Web サーバーまたはアプリケーションサーバーに送信されるようにします。

- クライアントは、必要に応じて、HTTP プロキシサーバーを使用してブローカと通信できます。プロキシのアドレスは、クライアントの起動時に、コマンド行オプションを使用して指定します。詳細は、378 ページの「HTTP プロキシを使用する」を参照してください。
- HTTP または HTTPS トンネルサーブレット ( どちらも Message Queue にバンドルされている ) が、Web サーバーまたはアプリケーションサーバーに読み込まれます。サーブレットは、ペイロードメッセージをクライアント HTTP 要求から取り出してから、ブローカに転送します。また HTTP/HTTPS トンネルサーブレットは、クライアントが作成した HTTP 要求に応じて、ブローカのメッセージをクライアントに返送します。1 つの HTTP/HTTPS トンネルサーブレットで複数のブローカにアクセスできます。

図 C-1　　HTTP/HTTPS サポートのアーキテクチャー




- ブローカ側では、httpjms または httpsjms 接続サービスが、対応するトンネルサーブレットから送られてくるメッセージを開いて逆多重化します。
- Web サーバーまたはアプリケーションサーバーに障害が生じ再起動された場合は、すべての接続が復元されるため、クライアントへの影響はありません。ブローカに障害が生じ再起動された場合は、例外がスローされ、クライアントはそれぞれの接続を再確立する必要があります。発生するのはまれですが、Web サーバーまたはアプリケーションサーバーとブローカの両方に障害が生じ、ブローカが再起動しなかった場合、Web サーバーまたはアプリケーションサーバーはクライアント接続を復元し、引き続きブローカ接続を待機しますが、クライアントには通知しません。この状況を避けるために、常に、ブローカを再起動してください。

図 C-1 からわかるとおり、HTTP と HTTPS サポートのアーキテクチャーは非常に良く似ています。主な相違点は、HTTPS (httpsjms 接続サービス ) の場合、トンネルサーブレットにクライアントアプリケーションとブローカの両方への安全な接続があることです。

ブローカへの安全な接続は、SSL に対応したトンネルサーブレット、つまり Message Queue の HTTPS トンネルサーブレットを通して提供されます。このトンネルサーブレットが、接続の要求時にブローカに自己署名型証明書を渡します。ブローカは証明書を使用して、HTTPS トンネルサーブレットへの暗号化された接続を設定します。この接続が確立されると、クライアントアプリケーションと Web サーバーまたはアプリケーションサーバーは、クライアントアプリケーションとトンネルサーブレット間の安全な接続のネゴシエーションを行うことができます。

# HTTP サポートの有効化

次に、HTTP サポートを有効化するのに必要な手順を説明します。

➤ **HTTP サポートを有効にする**

1. HTTP トンネルサーブレットを配置します。HTTP トンネルサーブレットは次の場所に配置できます。
   - Sun Java System Web Server
   - Sun Java System Application Server
2. ブローカの httpjms 接続サービスを設定し、ブローカを起動します。
3. HTTP 接続を設定します。

## 手順 1: HTTP トンネルサーブレットを配置する

HTTP トンネルサーブレットは、Sun Java System Web Server または Sun Java System Application Server に、Web アーカイブ (`.war`) ファイルとして配置できます。

HTTP トンネルサーブレットを `.war` ファイルとして配置するときは、Web サーバーまたはアプリケーションサーバーで提供される配置メカニズムを使用します。HTTP トンネルサーブレットの `.war` ファイル (`imqhttp.war`) は、`.jar`、`.war`、および `.rar` ファイルを含むディレクトリに配置します。これはオペレーティングシステムによって異なります ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

.war ファイルには、Web サーバーまたはアプリケーションサーバーがサーブレットを読み込んで実行するときに必要となる基本的な設定情報が示された配置記述子が含まれています。Web サーバーまたはアプリケーションサーバーによっては、サーブレットの URL のコンテキストルート部分を指定しなければならない場合もあります。

Web アーカイブファイルとして配置する

Sun Java System Web Server への配置については、372 ページの「HTTP トンネルサーブレットを Sun Java System Web Server に配置する」を参照してください。

Sun Java System Application Server への配置については、373 ページの「HTTP トンネルサーブレットを Sun Java System Application Server に配置する」を参照してください。

## HTTP トンネルサーブレットを Sun Java System Web Server に配置する

ここでは、Sun Java System Web Server への配置手順について説明します。Web ブラウザを使用してサーブレットの URL にアクセスすると、HTTP トンネルサーブレットが問題なく配置されたことが確認できます。ステータス情報も表示されます。

### ➤ HTTP トンネルサーブレットを .war ファイルとして配置する

1. ブラウザベースの管理 GUI で、「Virtual Server Class」タブを選択してから、「Manage Classes」を選択します。
2. 適切な仮想サーバークラス名 (`defaultClass` など ) を選択して、「Manage」ボタンをクリックします。
3. 「Manage Virtual Servers」を選択します。
4. 適切な仮想サーバー名を選択し、「Manage」ボタンをクリックします。
5. 「Web Applications」タブを選択します。
6. 「Deploy Web Application」をクリックします。
7. 「WAR File On and WAR File Path」フィールドで、`imqhttp.war` ファイルを指す適切な値を選択します。このファイルはオペレーティングシステムによって異なるディレクトリに格納されています ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

8. 「Application URI」フィールドにパスを入力します。

   「Application URI」フィールドの値は、トンネルサーブレット URL の */contextRoot* 部分です。

   `http://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

   たとえば、*contextRoot* を `imq` に設定した場合、「Application URI」フィールドは次のようになります。

   ```
   /imq
   ```

9. サーブレットを配置するインストールディレクトリのパス ( 通常は、Sun Java System Web Server インストールルートの内の場所 ) を入力します。
10. 「OK」をクリックします。
11. Web サーバーインスタンスを再起動します。

サーブレットは次のアドレスで利用可能となります。

`http://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

クライアントはこの URL を使用することで、HTTP 接続を使ってメッセージサービスに接続できます。

### *サーバーのアクセスログを無効にする*

必ずしもサーバーのアクセスログを無効にする必要はありませんが、無効にしたほうがより良いパフォーマンスを得ることができます。

#### ➤ サーバーのアクセスログを無効にする

1. 「Status」タブを選択します。
2. 「Log Preferences Page」を選択します。

Log クライアントアクセス制御を使用して、ロギングを無効にします。

## HTTP トンネルサーブレットを Sun Java System Application Server に配置する

この節では、HTTP トンネルサーブレットを Sun Java System Application Server に `.war` ファイルとして配置し、Message Queue ブローカからの接続を受け付けるようにトンネルサーブレットを設定する方法について説明します。

2 段階の手順が必要です。

- Application Server の配置ツールを使用して HTTP トンネルサーブレットを配置します。
- アプリケーションサーバーインスタンスの `server.policy` ファイルを変更します。

### 配置ツールを使用する

➤ **HTTP トンネルサーブレットを Application Server 環境に配置する**

1. Web ベースの管理 GUI で、次を選択します。

   「App Server」> 「Instances」> 「server1」> 「Applications」> 「Web Applications」

2. 「Deploy」ボタンをクリックします。

3. 「File Path:」テキストフィールドに、HTTP トンネルサーブレットの `.war` ファイル (`imqhttp.war`) の場所を入力し、「OK」をクリックします。

   `imqhttp.war` ファイルの場所は、オペレーティングシステムによって異なります ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

4. 「Context Root」テキストフィールドの値を設定し、「OK」をクリックします。

   「Context Root」フィールドの値は、トンネルサーブレット URL の */contextRoot* 部分です。

   `http://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

   たとえば、「Context Root」フィールドは `/imq` に設定できます。

   表示される確認画面で、トンネルサーブレットが正常に配置され、デフォルトで有効になっており、この場合は次の場所に格納されていることが確認されます。

   ```
   /var/opt/SUNWappserver8/domains/domain1/server1/applications/
   j2ee-modules/imqhttp_1
   ```

サーブレットは次の URL で利用可能となります。

`http://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

クライアントはこの URL を使用することで、HTTP 接続を使ってメッセージサービスに接続できます。

## server.policy ファイルを変更する

Application Server は、セキュリティーポリシーが変更されていないかぎり、強制的にデフォルトのセキュリティーポリシーセットを適用し、HTTP トンネルサーブレットが Message Queue ブローカからの接続を受け入れるのを阻止します。

各アプリケーションサーバーインスタンスには、セキュリティーポリシーまたはルールを含むファイルがあります。たとえば、Solaris 上の server1 インスタンスのこのファイルは次の場所にあります。

```
/var/opt/SUNWappserver8/domains/domain1/server1/config/
server.policy
```

トンネルサーブレットに Message Queue ブローカからの接続を受け入れさせるように設定するには、このファイルにエントリを追加する必要があります。

### ➤ アプリケーションサーバーの server.policy ファイルを変更する

1. `server.policy` ファイルを開きます。
2. 次のエントリを追加します。

```
grant codeBase
"file:/var/opt/SUNWappserver8/domains/domain1/server1/
              applications/j2ee-modules/imqhttp_1/-"
{
    permission java.net.SocketPermission "*",
              "connect,accept,resolve";
};
```

## 手順 2: httpjms 接続サービスを設定する

デフォルトでは、HTTP サポートはブローカに対してアクティブになっていないため、httpjms 接続サービスをアクティブにするようブローカを再設定する必要があります。再設定したあとは、68 ページの「ブローカの起動」で説明されているようにブローカを起動できます。

### ➤ httpjms 接続サービスをアクティブにする

1. ブローカのインスタンス設定ファイルを開きます。

   インスタンス設定ファイルは、その設定ファイルが関連付けられているブローカインスタンスの名前 (*instanceName*) によって識別されるディレクトリに格納されています ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

   `.../instances/`*instanceName*`/props/config.properties`

2. `httpjms` の値を `imq.service.activelist` プロパティーに追加します。

   ```
   imq.service.activelist=jms,admin,httpjms
   ```

ブローカは、起動時に、Web サーバーまたはアプリケーションサーバーとそのホストマシン上で実行している HTTP トンネルサーブレットを探します。リモートトンネルサーブレットにアクセスするには、接続サービスの `servletHost` および `servletPort` プロパティーを設定し直します。

`pullPeriod` プロパティーを設定し直して、パフォーマンスを向上させることもできます。表 C-1 では、httpjms 接続サービスの設定プロパティーについて説明します。

表 C-1　httpjms 接続サービスのプロパティー

| プロパティー | 説明 |
|---|---|
| imq.httpjms.http.servletHost | 必要に応じてこの値を変更し、HTTP トンネルサーブレットを実行するホストの名前 ( ホスト名または IP アドレス ) を指定します。リモートホストか、またはローカルホストの特定のホスト名のどちらかになります。デフォルト値は、localhost です。 |
| imq.httpjms.http.servletPort | この値を変更して、ブローカが HTTP トンネルサーブレットにアクセスするために使用するポート番号を指定します。Web サーバー上でデフォルトのポート番号が変更されている場合は、それに合わせてこのプロパティーを変更します。デフォルト値は、7675 です。 |
| imq.httpjms.http.pullPeriod | メッセージをブローカから取り出すために各クライアントランタイムが出す HTTP 要求の間隔を秒単位で指定します。このプロパティーはブローカで設定され、クライアントランタイムに伝達される点に注意してください。値がゼロまたは負の場合、クライアントは常に HTTP 要求の 1 つを保留にして、可能なかぎり迅速なメッセージの取り出しに備えます。クライアント数が多いと、この動作によって Web サーバーまたはアプリケーションサーバーのリソースが消耗し、サーバーの応答が遅くなる場合があります。そのような場合には、pullPeriod プロパティーを正の秒数に設定する必要があります。これにより、後続の取り出し要求が出される前の、クライアントの HTTP トランスポートドライバの待機時間が設定されます。値を正の数に設定すると、クライアントにより監視される応答時間を犠牲にして、Web サーバーまたはアプリケーションサーバーのリソースが維持されます。デフォルト値は、-1 です。 |
| imq.httpjms.http.connectionTimeout | クライアントランタイムが例外をスローする前に HTTP トンネルサーブレットからの応答を待機する時間を秒単位で指定します。このプロパティーはブローカで設定され、クライアントランタイムに伝達される点に注意してください。このプロパティーは、ブローカが HTTP トンネルサーブレットと通信したあと、接続を解放するまでの時間も指定します。ブローカとトンネルサーブレットは、HTTP サーブレットへアクセス中のクライアントが異常終了したかどうかを確認する手段を持っていないため、この場合はタイムアウトが必須となります。デフォルト値は、60 です。 |

# 手順 3: HTTP 接続を設定する

クライアントアプリケーションは、設定済みの接続ファクトリ管理対象オブジェクトを適切に使用して、ブローカへの HTTP 接続を確立する必要があります。この節では、HTTP 接続設定について説明します。

## 接続ファクトリを設定する

HTTP サポートを有効にするには、接続ファクトリの `imqAddressList` 属性を HTTP トンネルサーブレット URL に設定する必要があります。HTTP トンネルサーブレット URL の一般的な構文は次のとおりです。

`http://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

*hostName*`:`*portNumber* は、HTTP トンネルサーブレットをホストする Web サーバーまたはアプリケーションサーバーの名前とポートです。*contextRoot* は、Web サーバーまたはアプリケーションサーバーにトンネルサーブレットを配置したときに設定したパスです。

接続ファクトリ属性の全般と `imqAddressList` 属性の詳細については、『Message Queue Developer's Guide for Java Clients』を参照してください。

接続ファクトリ属性の設定は、次のいずれかの方法で行います。

- 接続ファクトリ管理対象オブジェクトを作成する `imqobjmgr` コマンドで `-o` オプションを使用するか (178 ページの「接続ファクトリの追加」を参照 )、管理コンソール (`imqadmin`) を使用して接続ファクトリ管理対象オブジェクトを作成するときに属性を設定します。
- クライアントを起動するコマンドで `-D` オプションを使用します (『Message Queue Developer's Guide for Java Clients』を参照 )。
- クライアントのプログラムで接続ファクトリを作成してから、API 呼び出しを使用して接続ファクトリの属性を設定します (『Message Queue Developer's Guide for Java Clients』を参照 )。

## 1 つのサーブレットを使用して複数のブローカにアクセスする

複数のブローカを実行している場合、複数の Web サーバーまたはアプリケーションサーバーとサーブレットインスタンスを設定する必要はありません。現在実行中のブローカ間で、1 つの Web サーバーまたはアプリケーションサーバーと HTTP トンネルサーブレットのインスタンスを共有できます。複数のブローカインスタンスが 1 つのトンネルサーブレットを共有している場合は、次に示すとおり、`imqAddressList` 接続ファクトリ属性を設定する必要があります。

`http://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel?ServerName=`*bkrHostName*`:`*instanceName*

*bkrHostName* はブローカインスタンスのホスト名で、*instanceName* はクライアントにアクセスさせる特定のブローカインスタンスの名前です。

*bkrHostName* と *instanceName* に正しい文字列を入力したことを確認するには、ブラウザからサーブレット URL にアクセスして、HTTP トンネルサーブレットの状態レポートを生成します。レポートでは、サーブレットがアクセスしているすべてのブローカが次のように一覧表示されます。

```
HTTP tunnel servlet ready.
Servlet Start Time : Thu May 30 01:08:18 PDT 2005
Accepting TCP connections from brokers on port : 7675
Total available brokers = 2
Broker List :
   jpgserv:broker2
   cochin:broker1
```

## HTTP プロキシを使用する

HTTP プロキシを使用して HTTP トンネルサーブレットにアクセスする場合、次の設定を行います。

- `http.proxyHost` システムプロパティーをプロキシサーバーのホスト名に設定します。
- `http.proxyPort` システムプロパティーをプロキシサーバーのポート番号に設定します。

クライアントアプリケーションを起動するコマンドに `-D` オプションを使用して、これらのプロパティーを設定できます。

# HTTPS サポートの有効化

以降の節では、HTTPS サポートを有効化する手順について説明します。この手順は、371 ページの「HTTP サポートの有効化」の手順とほとんど同じですが、さらに SSL 証明書の生成とアクセスに必要となる手順も追加されています。

➤ **HTTPS サポートを有効にする**

1. HTTPS トンネルサーブレットの自己署名型証明書を生成します。
2. HTTP トンネルサーブレットの `.war` ファイルの配置記述子を次のように変更します。
   - 証明書キーストアを配置した場所を指すようにします
   - 証明書キーストアのパスワードを指定します
3. HTTP トンネルサーブレットを配置します。HTTP トンネルサーブレットは次の場所に配置できます。
   - Sun Java System Web Server
   - Sun Java System Application Server
4. ブローカの httpsjms 接続サービスを設定し、ブローカを起動します。
5. HTTPS 接続を設定します。

それぞれの手順については、順次、詳しく説明します。

## 手順 1: HTTPS トンネルサーブレットの自己署名型証明書を生成する

Message Queue の SSL Support は、クライアントが既知の信頼されたサーバーと通信することを前提に、ネットワーク上のデータを保護することを目的としています。したがって、自己署名型のサーバー証明書だけを使用して SSL が実装されます。
httpsjms 接続サービスのアーキテクチャーでは、HTTPS トンネルサーブレットが、ブローカに対してもアプリケーションクライアントに対してもサーバーの役割をします。

`keytool` ユーティリティーを実行し、トンネルサーブレットの自己署名型証明書を生成します。コマンドプロンプトで次のとおり入力します。

```
JRE_HOME/bin/keytool -servlet keyStoreLocation
```

ユーティリティーが、必要な情報を要求します。UNIX システムでは、キーストアを作成するアクセス権を取得するためにスーパーユーザー (root) として `keytool` を実行する必要があります。

keytool は、まず、キーストアに対するパスワードの入力を要求します。次に一部の組織情報の入力、続いて確認を要求します。確認が取れると、キーの組み合わせを生成している間、このコマンドは停止します。その後、特定のキーの組み合わせをロックするためのパスワード ( キーパスワード ) の入力を要求してくるので、Return キーを押します。これで、キーパスワードに、キーストアと同じパスワードが設定されます。

| 注 | 設定したパスワードを忘れないでください。あとでトンネルサーブレットがキーストアを開くことができるように、そのパスワードを入力する必要があります。 |
|---|---|

JDK keytool ユーティリティーは、自己署名型証明書を生成し、*keyStoreLocation* 引数で指定された Message Queue のキーストアファイル内に証明書を配置します。

| 注 | HTTPS トンネルサーブレットは、キーストアを参照する必要があります。必ず、*keyStoreLocation* にある生成されたキーストアを、HTTPS トンネルサーブレットがアクセスできる場所に移動またはコピーしてください (381 ページの「手順 3: HTTPS トンネルサーブレットを配置する」を参照 )。 |
|---|---|

# 手順 2: HTTP トンネルサーブレットの .war ファイルの記述子ファイルを変更する

HTTP トンネルサーブレットの .war ファイルには、Web サーバーまたはアプリケーションサーバーがサーブレットを読み込んで実行するときに必要となる基本的な設定情報が示された配置記述子が含まれています。

imqhttps.war ファイルの配置記述子は、トンネルサーブレットが必要とするキーストアファイルが配置された場所を認識できません。そのため、imqhttps.war ファイルを配置する前に、トンネルサーブレットの配置記述子 (XML ファイル ) を編集し、キーストアの場所とパスワードを指定する必要があります。

### ➤ HTTPS トンネルサーブレット .war ファイルを変更する

1. .war ファイルを一時ディレクトリにコピーします。

   cp /usr/share/lib/imq/imqhttps.war /tmp (Solaris)

   cp /opt/sun/mq/share/lib/imqhttps.war /tmp (Linux)

   cp IMQ_HOME/lib/imqhttps.war /tmp (Windows)

2. 一時ディレクトリを現在のディレクトリにします。

```
$ cd /tmp
```

3. .war ファイルの内容を抽出します。

```
$ jar xvf imqhttps.war
```

4. .war ファイルの配置記述子を一覧表示します。

```
$ ls -l WEB-INF/web.xml
```

5. web.xml ファイルを編集して、keystoreLocation と keystorePassword という引数に正しい値を設定します。必要に応じて serverPort と serverHost の引数も設定します。

6. .war ファイルの内容を設定し直します。

```
$ jar uvf imqhttps.war WEB-INF/web.xml
```

これで修正済みの imqhttps.war ファイルを使用して、HTTPS トンネルサーブレットを配置できるようになりました。キーストアパスワードの漏洩が心配な場合は、ファイルシステムアクセス権を使用して、imqhttps.war ファイルへのアクセスを制限できます。

# 手順 3: HTTPS トンネルサーブレットを配置する

HTTP トンネルサーブレットは、Sun Java System Web Server または Sun Java System Application Server に、Web アーカイブ (WAR) ファイルとして配置できます。

HTTPS トンネルサーブレットを .war ファイルとして配置するときは、Web サーバーまたはアプリケーションサーバーで提供される配置メカニズムを使用します。HTTPS トンネルサーブレットの .war ファイル (imqhttps.war) は、オペレーティングシステムによって異なるディレクトリに格納されています ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

Web サーバーで暗号化がアクティブであり、クライアントとブローカの間で終端間の安全な通信が有効であることを確認してください。

## Web アーカイブファイルとして配置する

Sun Java System Web Server への配置については、382 ページの「HTTPS トンネルサーブレットを Sun Java System Web Server に配置する」を参照してください。

Sun Java System Application Server への配置については、383 ページの「HTTPS トンネルサーブレットを Sun Java System Application Server に配置する」を参照してください。

## HTTPS トンネルサーブレットを Sun Java System Web Server に配置する

この節では、HTTPS トンネルサーブレットを Sun Java System Web Server に .war ファイルとして配置する方法について説明します。Web ブラウザを使用してサーブレットの URL にアクセスすると、HTTPS トンネルサーブレットが問題なく配置されたことが確認できます。ステータス情報も表示されます。

HTTPS トンネルサーブレットを配置する前に、JSSE .jar ファイルが Web サーバーのクラスパスに含まれていることを確認します。これを確実にするもっとも簡単な方法は、jsse.jar、jnet.jar、および jcert.jar を WebServer_TOPDIR/bin/https/jre/lib/ext にコピーすることです。

### ➤ HTTPS トンネルサーブレットを .war ファイルとして配置する

1. ブラウザベースの管理 GUI で、「Virtual Server Class」タブを選択します。「Manage Classes」をクリックします。
2. 適切な仮想サーバークラス名 (defaultClass など ) を選択して、「Manage」ボタンをクリックします。
3. 「Manage Virtual Servers」を選択します。
4. 適切な仮想サーバー名を選択し、「Manage」ボタンをクリックします。
5. 「Web Applications」タブを選択します。
6. 「Deploy Web Application」をクリックします。
7. 「WAR File On and WAR File Path」フィールドで、修正済みの imqhttps.war ファイルを指す適切な値を選択します (380 ページの「HTTPS トンネルサーブレット .war ファイルを変更する」を参照 )。
8. 「Application URI」フィールドにパスを入力します。

   「Application URI」フィールドの値は、トンネルサーブレット URL の */contextRoot* 部分です。

   ```
   https://hostName:portNumber/contextRoot/tunnel
   ```

   たとえば、*contextRoot* を imq に設定した場合、「Application URI」フィールドは次のようになります。

   ```
   /imq
   ```
9. サーブレットを配置するインストールディレクトリのパス ( 通常は、Sun Java System Web Server インストールルートの内の場所 ) を入力します。
10. 「OK」をクリックします。
11. Web サーバーインスタンスを再起動します。

サーブレットは次の URL で利用可能となります。

```
https://hostName:portNumber/imq/tunnel
```

クライアントはこの URL を使用することで、安全な HTTPS 接続を使ってメッセージサービスに接続できます。

### *サーバーのアクセスログを無効にする*

必ずしもサーバーのアクセスログを無効にする必要はありませんが、無効にしたほうがより良いパフォーマンスを得ることができます。

➤ **サーバーのアクセスログを無効にする**

1. 「Status」タブを選択します。
2. 「Log Preferences Page」を選択します。

Log クライアントアクセス制御を使用して、ロギングを無効にします。

## HTTPS トンネルサーブレットを Sun Java System Application Server に配置する

この節では、HTTPS トンネルサーブレットを Sun Java System Application Server に `.war` ファイルとして配置する方法について説明します。

2 段階の手順が必要です。

- Application Server の配置ツールを使用して HTTPS トンネルサーブレットを配置します。
- アプリケーションサーバーインスタンスの `server.policy` ファイルを変更します。

### *配置ツールを使用する*

➤ **HTTPS トンネルサーブレットを Application Server 環境に配置する**

1. Web ベースの管理 GUI で、次を選択します。

   「App Server」> 「Instances」> 「server1」> 「Applications」> 「Web Applications」

2. 「Deploy」ボタンをクリックします。
3. 「File Path:」テキストフィールドに、HTTPS トンネルサーブレットの `.war` ファイル (`imqhttps.war`) の場所を入力し、「OK」をクリックします。

   `imqhttps.war` ファイルの場所は、オペレーティングシステムによって異なります ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

4. 「Context Root」テキストフィールドの値を設定し、「OK」をクリックします。

   「Context Root」フィールドの値は、トンネルサーブレット URL の */contextRoot* 部分です。

   ```
   https://hostName:portNumber/contextRoot/tunnel
   ```

   たとえば、「Context Root」フィールドは次のように設定できます。

   ```
   /imq
   ```

   次の画面で、トンネルサーブレットが正常に配置され、デフォルトで有効になっており、この場合は次の場所に格納されていることが表示されます。

   ```
   /var/opt/SUNWappserver8/domains/domain1/server1/applications/
   j2ee-modules/imqhttps_1
   ```

サーブレットは次の URL で利用可能となります。

```
https://hostName:portNumber/contextRoot/tunnel
```

クライアントはこの URL を使用することで、HTTPS 接続を使ってメッセージサービスに接続できます。

### *server.policy ファイルを変更する*

Application Server は、セキュリティーポリシーが変更されていないかぎり、強制的にデフォルトのセキュリティーポリシーセットを適用し、HTTPS トンネルサーブレットが Message Queue ブローカからの接続を受け入れるのを阻止します。

各アプリケーションサーバーインスタンスには、セキュリティーポリシーまたはルールを含むファイルがあります。たとえば、Solaris 上の server1 インスタンスのこのファイルは次の場所にあります。

```
/var/opt/SUNWappserver8/domains/domain1/server1/config/
server.policy
```

トンネルサーブレットに Message Queue ブローカからの接続を受け入れさせるには、このファイルにエントリを追加する必要があります。

#### ➤ アプリケーションサーバーの server.policy ファイルを変更する

1. server.policy ファイルを開きます。
2. 次のエントリを追加します。

```
grant codeBase
"file:/var/opt/SUNWappserver8/domains/domain1/server1/
             applications/j2ee-modules/imqhttps_1/-"
```

```
{
    permission java.net.SocketPermission "*",
            "connect,accept,resolve";
};
```

# 手順 4: httpsjms 接続サービスを設定する

デフォルトでは、HTTPS サポートはブローカに対してアクティブになっていないため、httpsjms 接続サービスをアクティブにするようブローカを再設定する必要があります。再設定したあとは、68 ページの「ブローカの起動」で説明されているようにブローカを起動できます。

### ➤ httpsjms 接続サービスをアクティブにする

1. ブローカのインスタンス設定ファイルを開きます。

   インスタンス設定ファイルは、その設定ファイルが関連付けられているブローカインスタンスの名前 (*instanceName*) によって識別されるディレクトリに格納されています ( 付録 A「プラットフォームごとの Message Queue データの場所」を参照 )。

   ```
   .../instances/instanceName/props/config.properties
   ```

2. `httpsjms` の値を `imq.service.activelist` プロパティーに追加します。

   ```
   imq.service.activelist=jms,admin,httpsjms
   ```

ブローカは、起動時に Web サーバーとそのホストマシン上で実行している HTTPS トンネルサーブレットを探します。リモートトンネルサーブレットにアクセスするには、接続サービスの `servletHost` および `servletPort` プロパティーを設定し直します。

`pullPeriod` プロパティーを設定し直して、パフォーマンスを向上させることもできます。表 C-2 では、httpsjms 接続サービスの設定プロパティーについて説明します。

**表 C-2** `httpsjms` 接続サービスのプロパティー

| プロパティー | 説明 |
|---|---|
| `imq.httpsjms.https.servletHost` | 必要に応じて、この値を変更し、HTTPS トンネルサーブレットを実行するホストの名前 ( ホスト名または IP アドレス ) を指定します。リモートホストか、またはローカルホストの特定のホスト名のどちらかになります。デフォルト値は、`localhost` です。 |

表 C-2　　httpsjms 接続サービスのプロパティー ( 続き )

| プロパティー | 説明 |
|---|---|
| imq.httpsjms.https.servletPort | この値を変更して、ブローカが HTTPS トンネルサーブレットにアクセスするために使用するポート番号を指定します。Web サーバー上でデフォルトのポート番号が変更されている場合は、それに合わせてこのプロパティーを変更します。デフォルト値は、7674 です。 |
| imq.httpsjms.https.pullPeriod | メッセージをブローカから取り出すために各クライアントが出す HTTP 要求の間隔を秒単位で指定します。このプロパティーはブローカで設定され、クライアントランタイムに伝達される点に注意してください。値がゼロまたは負の場合、クライアントは常に HTTP 要求の 1 つを保留にして、可能なかぎり迅速なメッセージの取り出しに備えます。クライアント数が多いと、この動作によって Web サーバーのリソースが消耗し、サーバーの応答が遅くなる場合があります。そのような場合には、pullPeriod プロパティーを正の秒数に設定する必要があります。これにより、後続の取り出し要求が出される前の、クライアントの HTTP トランスポートドライバの待機時間が設定されます。値を正の数に設定すると、クライアントにより監視される応答時間を犠牲にして、Web サーバーのリソースが維持されます。デフォルト値は、-1 です。 |
| imq.httpsjms.https.connectionTimeout | クライアントランタイムが例外をスローする前に HTTPS トンネルサーブレットからの応答を待機する時間を秒単位で指定します。このプロパティーはブローカで設定され、クライアントランタイムに伝達される点に注意してください。このプロパティーは、ブローカが HTTPS トンネルサーブレットと通信したあと、接続を解放するまでの時間も指定します。ブローカとトンネルサーブレットは、HTTPS サーブレットへアクセス中のクライアントが異常終了したかどうかを確認する手段を持っていないため、この場合はタイムアウトが必要となります。デフォルト値は、60 です。 |

# 手順 5: HTTPS 接続を設定する

クライアントアプリケーションは、適切に設定された接続ファクトリ管理対象オブジェクトを使用して、ブローカへの HTTPS 接続を確立する必要があります。

ただし、クライアントは Java Secure Socket Extension (JSSE) で提供される SSL ライブラリへもアクセスし、root 証明書を持つ必要もあります。SSL ライブラリは、JDK 1.4 に付属しています。それ以前の JDK バージョンを使用している場合は、「JSSE を設定する」を参照してください。それ以外の場合は、「root 証明書をインポートする」に進んでください。

これらの点を解決したあとで、HTTPS 接続の設定に進むことができます。

## JSSE を設定する

➤ **JSSE を設定する**

1. JSSE .jar ファイルを JRE_HOME/lib/ext ディレクトリにコピーします。

   jsse.jar、jnet.jar、jcert.jar

2. JSSE セキュリティープロバイダを静的に追加します。

   security.provider.*n*=com.sun.net.ssl.internal.ssl.Provider

   これを JRE_HOME/lib/security/java.security ファイルに追加します。*n* は、セキュリティープロバイダパッケージに対して次に利用可能な優先順位です。

3. JDK 1.4 を使用していない場合は、-D オプションをクライアントアプリケーションを起動するコマンドに使用して、次の JSSE プロパティーを設定します。

   java.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol

## root 証明書をインポートする

Web サーバーの証明書に署名した認証局 (CA) の root 証明書が、デフォルトで信頼されるデータベースにない場合、または Web サーバーまたはアプリケーションサーバーの専用の証明書を使用している場合、信頼されるデータベースに証明書を追加する必要があります。これに該当する場合は、次の手順に従ってください。それ以外の場合は、「接続ファクトリを設定する」に進んでください。

証明書が *certFile* に保存され、*trustStoreFile* がキーストアであると仮定して、次のコマンドを実行します。

```
JRE_HOME/bin/keytool -import -trustcacerts
-alias aliasForCertificate -file certFile
-keystore trustStoreFile
```

Trust this certificate? という質問に YES と答えます。

クライアントアプリケーションを起動するコマンドに -D オプションを使用して、次の JSSE プロパティーを指定する必要もあります。

```
javax.net.ssl.trustStore=trustStoreFile
javax.net.ssl.trustStorePassword=trustStorePasswd
```

## 接続ファクトリを設定する

HTTPS サポートを有効にするには、接続ファクトリの imqAddressList 属性を HTTPS トンネルサーブレット URL に設定する必要があります。HTTPS トンネルサーブレット URL の一般的な構文は次のとおりです。

```
https://hostName:portNumber/contextRoot/tunnel
```

*hostName*:*portNumber* は、HTTPS トンネルサーブレットをホストする Web サーバーの名前とポートです。*contextRoot* は、Web サーバーにトンネルサーブレットを配置したときに設定したパスです。

接続ファクトリ属性の全般と imqAddressList 属性の詳細については、『Message Queue Developer's Guide for Java Clients』を参照してください。

接続ファクトリ属性の設定は、次のいずれかの方法で行います。

- 接続ファクトリ管理対象オブジェクトを作成する imqobjmgr コマンドで -o オプションを使用するか (178 ページの「接続ファクトリの追加」を参照 )、管理コンソール (imqadmin) を使用して接続ファクトリ管理対象オブジェクトを作成するときに属性を設定します。
- クライアントアプリケーションを起動するコマンドに -D オプションを使用します (『Message Queue Developer's Guide for Java Clients』を参照 )。
- クライアントアプリケーションのプログラムで接続ファクトリを作成してから、API 呼び出しを使用して接続ファクトリの属性を設定します (『Message Queue Developer's Guide for Java Clients』を参照 )。

## 1 つのサーブレットを使用して複数のブローカにアクセスする

複数のブローカを実行している場合、複数の Web サーバーとサーブレットインスタンスを設定する必要はありません。現在実行中のブローカ間で、1 つの Web サーバーと HTTPS トンネルサーブレットのインスタンスを共有できます。複数のブローカインスタンスが 1 つのトンネルサーブレットを共有している場合は、次に示すとおり、imqAddressList 接続ファクトリ属性を設定する必要があります。

```
https://hostName:portNumber/contextRoot/tunnel?ServerName=bkrHostName:instanceName
```

*bkrHostName* はブローカインスタンスのホスト名で、*instanceName* はクライアントにアクセスさせる特定のブローカインスタンスの名前です。

*bkrHostName* と *instanceName* に正しい文字列を入力したことを確認するには、ブラウザからサーブレット URL にアクセスして、HTTPS トンネルサーブレットの状態レポートを生成します。レポートでは、サーブレットがアクセスしているすべてのブローカが次のように一覧表示されます。

```
HTTPS tunnel servlet ready.
Servlet Start Time : Thu May 30 01:08:18 PDT 2002
Accepting TCP connections from brokers on port : 7674
Total available brokers = 2
Broker List :
   jpgserv:broker2
   cochin:broker1
```

## HTTP プロキシを使用する

HTTP プロキシを使用して HTTPS トンネルサーブレットにアクセスする場合、次の設定を行います。

- `http.proxyHost` システムプロパティーをプロキシサーバーのホスト名に設定します。
- `http.proxyPort` システムプロパティーをプロキシサーバーのポート番号に設定します。

クライアントアプリケーションを起動するコマンドに `-D` オプションを使用して、これらのプロパティーを設定できます。

# トラブルシューティング

この節では、HTTP や HTTPS 接続で発生する可能性がある問題、およびその問題の解決方法について説明します。

## サーバーかブローカの障害

Web サーバーで障害が生じても再起動すれば、すべての接続が復元され、クライアントへの影響はありません。しかしブローカに障害が生じて再起動された場合は、例外がスローされ、クライアントはそれぞれの接続を再確立する必要があります。

Web サーバーとブローカの両方に障害が生じ、ブローカが再起動されない場合、Web サーバーはクライアント接続を復元し、ブローカ接続を待機しますが、クライアントには通知しません。この状況を避けるため、ブローカの再起動を必ず確認してください。

## クライアントのトンネルサーブレットによる接続障害

HTTPS クライアントがトンネルサーブレットでブローカに接続できない場合は、次のように操作します。

1. サーブレットとブローカを起動します。
2. ブラウザを使用し、HTTPS トンネルサーブレット URL でサーブレットに手動でアクセスします。
3. 次の管理コマンドを使用し、接続の一時停止と再開を行います。

```
imqcmd pause svc -n httpsjms -u admin
imqcmd resume svc -n httpsjms -u admin
```

サービスが再開する場合、HTTPS クライアントはトンネルサーブレットでブローカに接続できます。

# よく使用するコマンドユーティリティーのコマンド

この付録では、よく使用するいくつかの Message Queue コマンドユーティリティー (imqcmd) のコマンドを示します。コマンド行から利用可能なコマンドオプションおよび属性の包括的なリストについては、第 13 章「コマンド行のリファレンス」の 278 ページの「コマンドユーティリティー」を参照してください。

## 構文

```
imqcmd subcommand argument [options]
imqcmd -h|H
imqcmd -v
```

-H または -h は、包括的なヘルプを提供します。-v サブコマンドは、バージョン情報を提供します。

imqcmd を使用するときは、コマンドユーティリティーによってパスワードを要求されます。要求されないようにしてセキュリティーを高めるには、-passfile *pathToPassfile* オプションを使用して、管理者のユーザー名とパスワードが含まれるパスワードファイルを指定できます。

例 : `imqcmd query bkr -u adminUserName -passfile pathToPassfile -b myServer:7676`

# ブローカとクラスタの管理

```
imqcmd query bkr
imqcmd pause bkr
imqcmd restart bkr
imqcmd resume bkr
imqcmd shutdown bkr -b myBroker:7676
imqcmd update bkr -o "imq.system.max_count=1000"
imqcmd reload cls
```

## ブローカ設定プロパティー (-o オプション)

表 D-1 に、よく使用するブローカ設定プロパティーの一覧を示します。ブローカ設定プロパティーとその説明の完全なリストについては、第 14 章「ブローカのプロパティーのリファレンス」を参照してください。

表 D-1　　ブローカ設定プロパティー (-o オプション)

| プロパティー | メモ |
|---|---|
| imq.autocreate.queue | |
| imq.autocreate.queue.maxNumActiveConsumers | 無制限の場合は -1 を指定します |
| imq.autocreate.queue.maxNumBackupConsumers | 無制限の場合は -1 を指定します |
| imq.autocreate.topic | |
| imq.cluster.url | |
| imq.destination.DMQ.truncateBody | |
| imq.destination.logDeadMessages | |
| imq.log.file.rolloverbytes | 無制限の場合は -1 を指定します |
| imq.log.file.rolloversecs | 無制限の場合は -1 を指定します |
| imq.log.level | NONE<br>ERROR<br>WARNING<br>INFO |

表 D-1　ブローカ設定プロパティー (-o オプション ) ( 続き )

| プロパティー | メモ |
|---|---|
| imq.message.max_size | 無制限の場合は -1 を指定します |
| imq.portmapper.port | |
| imq.system.max_count | 無制限の場合は -1 を指定します |
| imq.system.max_size | 無制限の場合は -1 を指定します |

# サービスと接続の管理

```
imqcmd list svc

imqcmd query svc

imqcmd update svc -n jms -o "minThreads=200" -o "maxThreads=400" -o
"port=8995"

imqcmd pause svc -n jms

imqcmd resume svc -n jms

imqcmd list cxn -svn jms

imqcmd query cxn -n 1234567890
```

# 永続サブスクライバの管理

```
imqcmd list dur -d MyTopic

imqcmd destroy dur -n myDurSub -c "clientID-111.222.333.444"

imqcmd purge dur -n myDurSub -c "clientID-111.222.333.444"
```

# トランザクションの管理

```
imqcmd list txn
imqcmd commit txn -n 1234567890
imqcmd query txn -n 1234567890
imqcmd rollback txn -n 1234567890
```

# 送信先の管理

```
imqcmd create dst -n MyQueue -t q -o "maxNumMsgs=1000" -o
"maxNumProducers=5"
imqcmd update dst -n MyTopic -t t -o "limitBehavior=FLOW_CONTROL|
REMOVE_OLDEST|REJECT_NEWEST|REMOVE_LOW_PRIORITY"
imqcmd compact dst -n MyQueue -t q
imqcmd purge dst -n MyQueue -t q
imqcmd pause dst -n MyQueue -t q -pst PRODUCERS|CONSUMERS|ALL
imqcmd resume dst -n MyQueue -t q
imqcmd destroy dst -n MyQueue -t q
imqcmd query dst -n MyQueue -t q
imqcmd list dst -tmp
```

## 送信先設定プロパティー (-o オプション )

表 D-2 に、よく使用する送信先設定プロパティーの一覧を示します。送信先設定プロパティーとその説明の完全なリストについては、第 15 章「物理的送信先のプロパティーのリファレンス」を参照してください。

表 D-2　送信先設定プロパティー (-o オプション )

| プロパティー | メモ |
|---|---|
| consumerFlowLimit | 無制限の場合は -1 を指定します |
| isLocalOnly ( 作成のみ ) | |

表 D-2　送信先設定プロパティー (-o オプション ) ( 続き )

| プロパティー | メモ |
|---|---|
| limitBehavior | FLOW_CONTROL<br>REMOVE_OLDEST<br>REJECT_NEWEST<br>REMOVE_LOW_PRIORITY |
| localDeliveryPreferred ( キューのみ ) | |
| maxNumActiveConsumers ( キューのみ ) | 無制限の場合は -1 を指定します |
| maxNumBackupConsumers ( キューのみ ) | 無制限の場合は -1 を指定します |
| maxBytesPerMsg | 無制限の場合は -1 を指定します |
| maxNumMsgs | 無制限の場合は -1 を指定します |
| maxNumProducers | 無制限の場合は -1 を指定します |
| maxTotalMsgBytes | 無制限の場合は -1 を指定します |
| useDMQ | |

# メトリックス

```
imqcmd metrics bkr -m cxn|rts|ttl -int 5 -msp 20
imqcmd metrics svc -m cxn|rts|ttl
imqcmd metrics dst -m con|dsk|rts|ttl
```

# 用語集

Message Queue 用語の詳細については、『Message Queue 技術の概要』の用語集を参照してください。Sun Java System 製品群で使用される用語の完全なリストは、『Java Enterprise System 用語集』(http://docs.sun.com/app/docs/doc/819-4629) にあります。

# 索引

## I

## J



## L

## M



## N



## O



## P



## R



## S



## T

## か



## き



## く

## こ



## さ



## し

# す



# せ

## そ



## ち



## つ



## て



## と

## に



## は

# ふ

## へ



## ほ

## ら



## り



## る



## ろ