Sun Java™ System

Message Queue 3
管理ガイド

2005Q1

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

Part No: 819-2217

目 次

図目次

表目次

手順一覧

はじめに

『Sun Java™ System Message Queue 管理ガイド』は、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。

このマニュアルでは、Sun Java System Message Queue 3 2005Q1 (Message Queue 3.6) について説明します。

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 Message Queue 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

事前に「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

次の表は、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　　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「管理のクイックスタート」	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」	トピックとキューに関する管理タスクを説明します。
第 7 章「セキュリティの管理」	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、送信先管理対象オブジェクトとコネクションファクトリ管理対象オブジェクト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　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する技術と、パフォーマンスを最適化するためのメッセージサーバーの調整方法を説明します。
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を判断する方法、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「コマンドのリファレンス」	Message Queue のコマンドユーティリティの構文と詳細を示します。
第 14 章「ブローカのプロパティのリファレンス」	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のリファレンス」	トピックとキューの設定に使用できるプロパティ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」	送信先管理対象オブジェクトとコネクションファクトリ管理対象オブジェクトの設定に使用できる、プロパティとその説明を一覧表示しています。
第 17 章「JMS リソースアダプタ属性リファレンス」	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「オペレーティングシステムごとの 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 の設定方法を説明します。

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　マニュアルの表記規則

書式	説明
斜体	可変部分に使われます。斜体で表記された項目や値は適宜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	ファイルシステムタイプ (GIF、TXT、HTML など)、環境変数 (IMQ_HOME)、または頭文字 (Message Queue、JSP) を表します。
キー + キー	複数のキーストロークはプラス記号で結合します。Ctrl+A は、両方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™、Windows、および Linux 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。

表 3　Message Queue ディレクトリ変数

変数	説明
IMQ_HOME	この変数は通常、Message Queue マニュアル内で 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) を参照するのに使用されます。 • Solaris および Linux の場合、ルート Message Queue インストールディレクトリは存在しません。そのため、IMQ_HOME は、Solaris 上のファイルの場所を参照するために Message Queue マニュアルで使用されることはありません。 • Solaris と Windows の Sun Java System Application Server の場合、ルート Message Queue インストールディレクトリは Application Server 基本ディレクトリの下の /imq です。 • Windows の場合、ルート Message Queue インストールディレクトリは 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 です。
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ている、/var ディレクトリです。任意のディレクトリを指す環境変数として設定されます。 • Solaris の場合、IMQ_VARHOME のデフォルト値は /var/imq ディレクトリです。 • Solaris の場合、Sun Java System Application Server の Evaluation Edition では、IMQ_VARHOME のデフォルト値は IMQ_HOME/var ディレクトリです。 • Windows の場合、IMQ_VARHOME のデフォルト値は IMQ_HOME/var ディレクトリです。 • Windows の場合、Sun Java System Application Server の IMQ_VARHOME のデフォルト値は IMQ_HOME/var ディレクトリです。 • Linux の場合、IMQ_VARHOME のデフォルト値は /var/opt/sun/mq ディレクトリです。

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) の場所を指す環境変数です。 • Solaris の場合、IMQ_JAVAHOME は次の順序で Java ランタイムを検索しますが、オプションでユーザーが値を必要な JRE が置かれた場所に設定することもできます。 Solaris 8 または 9: /usr/jdk/entsys-j2se /usr/jdk/jdk1.5.* /usr/jdk/j2sdk1.5.* /usr/j2se Solaris 10: /usr/jdk/entsys-j2se /usr/java /usr/j2se • Linux の場合、Message Queue は次の順序で Java ランタイムを検索しますが、オプションでユーザーが IMQ_JAVAHOME の値を必要な JRE が置かれた場所に設定することもできます。 /usr/jdk/entsys-j2se /usr/java/jre1.5.* /usr/java/jdk1.5.* /usr/java/jre1.4.2* /usr/java/j2sdk1.4.2* • Windows の場合、IMQ_JAVAHOME のデフォルト値は IMQ_HOME/jre ですが、ユーザーはオプションで必要な JRE の配置場所を自由に設定できます。

このマニュアルでは、IMQ_HOME、IMQ_VARHOME、および IMQ_JAVAHOME は、プラットフォーム固有の環境変数の表記法や構文 (UNIX 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。

関連マニュアル

このガイド以外にも、Message Queue には追加のマニュアルが用意されています。

Message Queue マニュアルセット

Message Queue マニュアルセットは、次のマニュアルで構成されています。各マニュアルを通常使用する順番で、表 4 に一覧表示します。

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 インストールガイド』	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 仕様および 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方法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 インタフェース (C-API) を使用する、C クライアントプログラムを開発する方法に関する情報を提供します。

オンラインヘルプ

Message Queue には、Message Queue メッセージサービス管理タスクを実行するためのコマンド行ユーティリティが含まれています。各ユーティリティのオンラインヘルプにアクセスするには、第 13 章「コマンドのリファレンス」を参照してください。

また、Message Queue には、グラフィカルユーザーインタフェース (GUI) 管理ツールである管理コンソール (Administration Console) (`imqadmin`) も含まれています。管理コンソールには、操作状況に合わせて表示できるオンラインヘルプが用意されています。

JavaDoc

JavaDoc 形式の JMS と Message Queue API マニュアルは、次の場所に保存されています。

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`

このマニュアルは、Netscape または Internet Explorer などの 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 管理対象オブジェクト用の Message Queue 固有の API が含まれており (『Message Queue Developer's Guide for Java Clients』の第 3 章を参照)、メッセージングアプリケーションの開発者にとって有用です。

クライアントアプリケーション例

クライアントアプリケーションコードのサンプルを示すいくつかのアプリケーション例を、プラットフォーム別のディレクトリに収めています (付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照)。

このディレクトリと各サブディレクトリにある README ファイルを参照してください。

Java Message Service (JMS) 仕様書

JMS 仕様書は、次のサイトにあります。

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

この仕様書には、サンプルのクライアントコードも掲載されています。

関連するサードパーティの Web サイトのリファレンス

このマニュアルでは、サードパーティの URL が参考として示されているほか、追加の関連情報も提供されています。

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、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、http://docs.sun.com にアクセスし、「コメントの送信」をクリックしてください。オンラインフォームで、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。

第 I 部

Message Queue 管理の概要

第 1 章

管理タスクと管理ツール

Sun Java™ System Message Queue の管理には、さまざまなタスクとこれらのタスクを実行するためのさまざまなツールがあります。

この章では、管理タスクの概要を述べてから、コマンド行管理ユーティリティの共通機能に的を絞って、管理ツール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。

開発環境の管理タスク

開発環境では、Message Queue クライアントアプリケーションのプログラミングが作業の中心になり、プログラマは個人用のシステムを管理することが多くなります。Message Queue メッセージサーバーは、主にテストのために必要となります。開発環境では、柔軟性が重視されるため、通常、管理では次の要件が適用されます。

- 開発者がテストで使用するブローカを起動する程度の最小の管理。
- 組み込みの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、およびファイルシステムストアの使用。通常の開発テストでは、これらの簡単な構成で十分です。
- マルチブローカテストで、マスターブローカを使用しない。
- 管理者が作成した送信先ではなく、自動的に作成された送信先を使用。
- 管理対象オブジェクトを管理者ではなく、クライアントコードでインスタンス化。

本稼動環境の管理タスク

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。一般的にこれらのタスクは、セットアップ操作とメンテナンス操作に分類することができます。

セットアップ操作

通常、次のセットアップ操作のうち少なくとも一部を実行する必要があります。

- **管理者のセキュリティ** (保護された管理ツールの使用)
  - 承認 : 特定の個人またはグループが管理接続サービスにアクセスし、デッドメッセージキューからメッセージを消費できます (157 ページの「コネクションサービスのアクセス制御」と 158 ページの「物理的な送信先のアクセス制御」を参照)。
  - デフォルトの管理ユーザー (admin) と、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、ユーザーパスワードを変更します (149 ページの「デフォルトの管理者パスワードの変更」を参照)。
  - グループを認証している場合、各管理者がそのグループに所属していることを確認します。
    - ファイルベースのユーザーリポジトリ

      ファイルベースのユーザーリポジトリには、管理者グループが 1 つ含まれています (admin)。新しく管理ユーザーを作成する場合、新しいユーザーが admin グループに属していることを確認してください。
    - LDAP ユーザーリポジトリ

      LDAP サーバーでグループを作成するか、既存のグループを使用します。管理者権限を付与するユーザーがグループのメンバーであること、またその場合はそのグループのメンバーに管理接続を許可していることを確認します。

      詳細は、150 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。
- **全般的なセキュリティ** (第 7 章「セキュリティの管理」を参照)
  - 認証 : ファイルベースのユーザーリポジトリにエントリを作成するか、あるいはブローカを設定して、既存の LDAP ユーザーリポジトリを使用します。

    少なくとも、管理機能をパスワードで保護する必要があります。
  - 承認 : アクセス制御プロパティファイルのアクセス設定を変更します。

- 暗号化 : SSL ベースのコネクションサービスを設定します (160 ページの「SSL ベースのサービスの操作」を参照)。

- **管理対象オブジェクト** (第 8 章「管理対象オブジェクトの管理」を参照)
  - LDAP オブジェクトストアを設定またはセットアップします。
  - ConnectionFactory と送信先の管理対象オブジェクトを作成します。
- **ブローカのクラスタ** (第 9 章「ブローカクラスタを使用した作業」を参照)
  - 中央設定ファイルを作成します。
  - マスターブローカを使用します。
- **持続** : ブローカによるプラグイン持続または組み込み持続の使用を許可するかを決定し、必要なストアを設定します (99 ページの「持続ストアの設定」を参照)。
- **メモリー管理** : メッセージ数とメッセージに割り当てられるメモリー量が使用可能なブローカのメモリーリソース内に納まるように送信先属性を設定します (339 ページの表 15-1 を参照)。

メンテナンス操作

本稼動環境では、Message Queue 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。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が重視されるため、次に示すさまざまな運用タスクを Message Queue の管理ツールを使用して実行する必要があります。

- **アプリケーション管理**
  - `imq.autocreate.queue` プロパティと `imq.autocreate.topic` プロパティの値を設定して、ブローカの自動作成機能を無効にする (319 ページの「メッセージルーターのプロパティ」を参照)。
  - アプリケーションのために、物理的送信先を作成する (127 ページの第 6 章「物理的送信先の管理」を参照)。
  - 送信先へのユーザーアクセスを設定する (153 ページの「ユーザーの承認 : アクセス制御プロパティファイル」を参照)。
  - 送信先を監視および管理する (122 ページの「永続サブスクリプションの管理」を参照)。
  - 永続サブスクリプションを監視および管理する (122 ページの「永続サブスクリプションの管理」を参照)。
  - トランザクションを監視および管理する (124 ページの「トランザクションの管理」を参照)。

- **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て、ブローカの調整および再設定を行う (221 ページの第 11 章「メッセージサービスの分析と調整」を参照)。
  - ブローカのメモリーリソースを管理する (221 ページの第 11 章「メッセージサービスの分析と調整」を参照)。
  - クラスタにブローカを追加して、ロードバランスを実行する (第 9 章「ブローカクラスタを使用した作業」を参照)。
  - 障害が発生したブローカを復元する (67 ページの「ブローカのインタラクティブな起動」を参照)。
- **アプリケーションの管理**
  - 必要に応じて、追加のコネクションファクトリと送信先の管理対象オブジェクトを作成する (189 ページの「管理対象オブジェクトの追加および削除」を参照)。
  - ConnectionFactory 属性の値を調整して、Java クライアントアプリケーションの正しい動作を確実にする (第 8 章「管理対象オブジェクトの管理」を参照)。

管理ツール

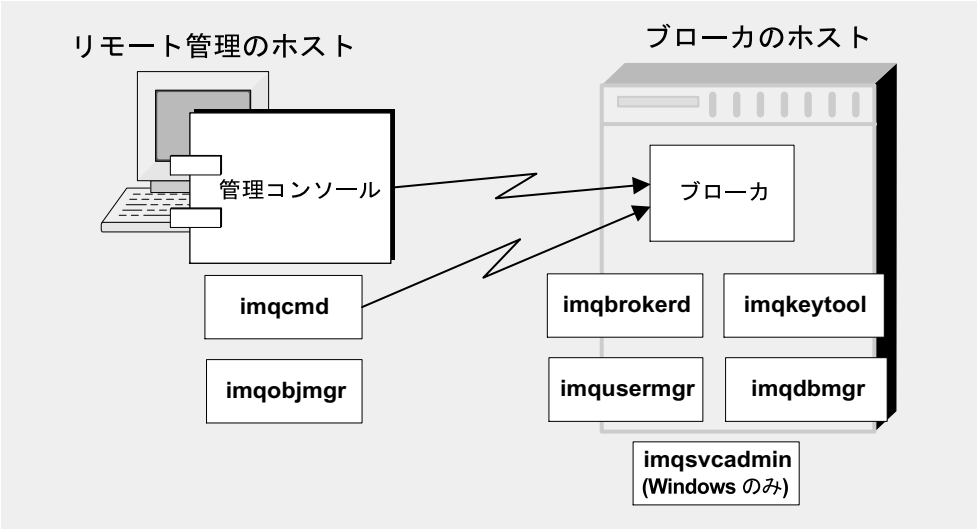
Message Queue の管理ツールは 2 つの種類に分けられます。

- コマンド行ユーティリティ
- グラフィカル管理コンソール (`imqadmin`)

コマンド行ユーティリティ

この節では、Message Queue の管理タスクを実行する際に使用するコマンド行ユーティリティについて説明します。ブローカの起動と管理、その他の特殊な管理タスクを実行する場合に、Message Queue のユーティリティを使用します。

図 1-1　ローカルおよびリモートの管理ユーティリティ



Message Queue のユーティリティはすべて、コマンド行インタフェース (CLI) からアクセスできます。ユーティリティコマンドは、この章の後半で説明するように、共通の形式、構文規則、およびオプションを共有します。コマンド行ユーティリティの使い方の参照情報は、第 13 章「コマンドのリファレンス」で説明しています。

ブローカ (`imqbrokerd`): ブローカを起動するには、ブローカユーティリティを使用します。`imqbrokerd` コマンドのオプションを使用して、クラスタ内でブローカを接続するかどうかを指定したり、起動時にブローカで使用される追加の設定情報を指定したりします。

コマンド (`imqcmd`): ブローカの起動後に、コマンドユーティリティを使用して物理的送信先の作成、更新、および削除、ブローカとブローカのコネクションサービスの管理、およびブローカのリソースの管理を行います。

オブジェクトマネージャ (`imqobjmgr`): オブジェクトマネージャユーティリティを使用して、JNDI を介したアクセスが可能なオブジェクトストア内の管理対象オブジェクトの追加、一覧表示、更新、および削除を行います。管理対象オブジェクトを使用すると、JMS プロバイダ固有の命名および設定形式から独立するため、JMS クライアントはプロバイダに依存しなくなります。

ユーザーマネージャ (`imqusermgr`): ユーザーマネージャユーティリティを使用して、ユーザーの認証および承認に使用するファイルベースのユーザーリポジトリを設定します。

キーツール (`imqkeytool`): キーツールユーティリティを使用して、SSL 認証で使用される自己署名型証明書を生成できます。

データベースマネージャ (`imqdbmgr`): データベースマネージャユーティリティを使用して、持続ストレージ用の JDBC 互換のデータベースを作成および管理します。

サービス管理 (imqsvcadmin) : サービス管理ユーティリティを使用して Windows のサービスとして、ブローカをインストール、クエリー、および削除します。

管理コンソール

管理コンソールは、コマンドユーティリティ (imqcmd) とオブジェクトマネージャユーティリティ (imqobjmgr) の 2 つのコマンド行ユーティリティの機能が組み合わされたものです。

管理コンソールおよび、これらの 2 つのコマンド行ユーティリティを使用すると、ブローカをリモートで管理したり、Message Queue の管理対象オブジェクトを管理したりすることができます。その他のコマンド行ユーティリティ (imqusermgr、imqdbmgr、および imqkeytool) は、37 ページの図 1-1 に示すように、関連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。

管理コンソールに関する情報は、オンラインヘルプから入手できます。特殊なタスクを実行するのに一般的に使用するコマンド行ユーティリティについては、「コマンド行ユーティリティ」で説明します。

管理コンソールを使用すると、次のタスクを実行できます。

- ブローカに接続し、ブローカを管理する。
- ブローカで物理的送信先を作成および管理する。
- オブジェクトストアに接続する。
- オブジェクトストアに管理対象オブジェクトを追加し、それらを管理する。

管理コンソールでは、実行できないタスクがいくつかあります。これには、ブローカの起動、ブローカのクラスタの作成、ブローカと物理的送信先の特殊なプロパティの設定、ユーザーデータベースの管理などが挙げられます。

第 2 章「管理のクイックスタート」では、管理コンソールについて説明し、管理コンソールを使用した基本的なタスクの実行方法を示した簡単で実践的な演習が用意されています。

第 2 章

管理のクイックスタート

このクイックスタートでは、Message Queue のブローカとオブジェクトストアを管理するグラフィカルインタフェースである管理コンソールを使用した、基本的な管理タスクを中心に説明します。この章の手順では、次の作業を実行する方法を説明します。

- ブローカを起動します。
- 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します。
- ブローカに物理的送信先を作成します。
- 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します。
- 送信先オブジェクトをオブジェクトストアに追加し、そのプロパティを表示します。

このクイックスタートでは、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「オペレーティングシステムごとの 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。クイックスタートの最後の部分では、このアプリケーションを実行します。

このクイックスタートでは、主に管理コンソールを使用して、基本的な管理タスクを実行します。クイックスタートとは別に、マニュアルには目を通し、参照するようにしてください。

一部の Message Queue 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を使用します。

- 特定の物理的送信先プロパティを設定する
- ブローカクラスタを作成する
- ユーザーのデータベースを管理する

上記のタスクを実行する方法の詳細は、第 6 章「物理的送信先の管理」、第 9 章「ブローカクラスタを使用した作業」および第 7 章「セキュリティの管理」を参照してください。

準備

開始する前に、Message Queue 製品をインストールする必要があります。詳細は、『Message Queue インストールガイド』を参照してください。この章は Windows を基準にしており、UNIX™ ユーザーへの注意点が追加される形で解説されているので注意してください。

この章では、「アイテム 1」>「アイテム 2」>「アイテム 3」と表記されている場合、「アイテム 1」というメニューをプルダウンし、メニューから「アイテム 2」を選択した後、「アイテム 2」で提供される選択肢から「アイテム 3」を選択することを意味します。

管理コンソールを起動する

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。

- 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。
- Solaris の場合、次のコマンドを入力します。

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

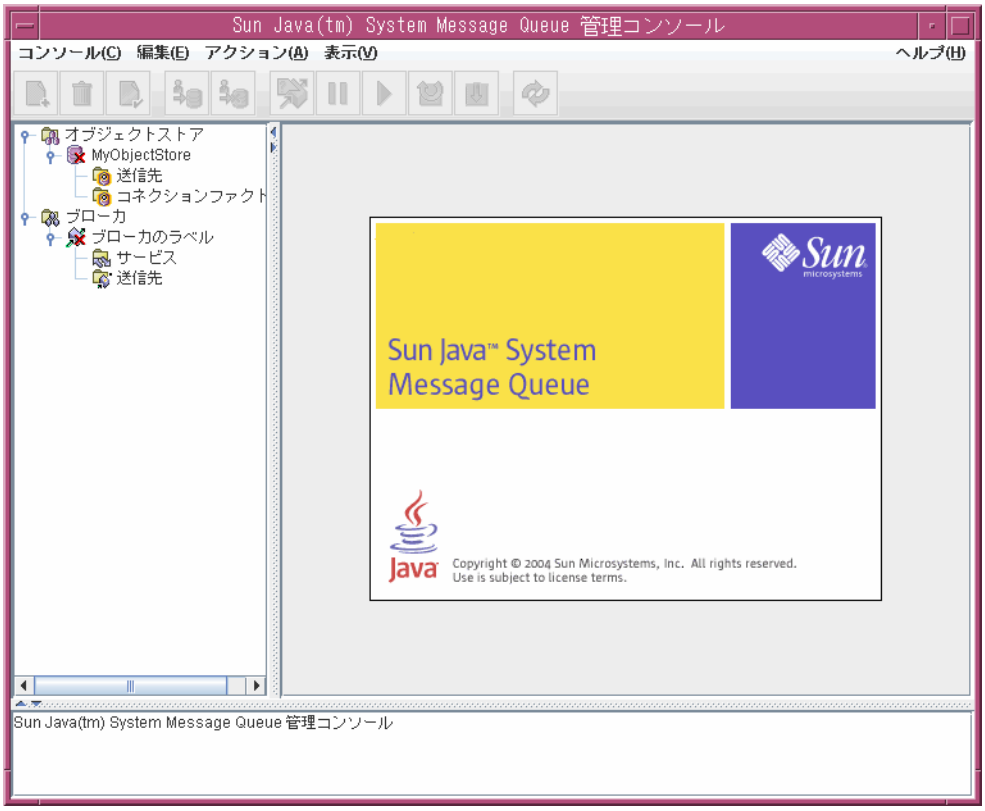
- Linux の場合、次のコマンドを入力します。

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

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。

コンソールのウィンドウの確認に数秒かかります。

コンソールは、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ペイン、右側に結果を表すペイン (この図では Sun Java SystemMessage 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ペインという構成になっています。



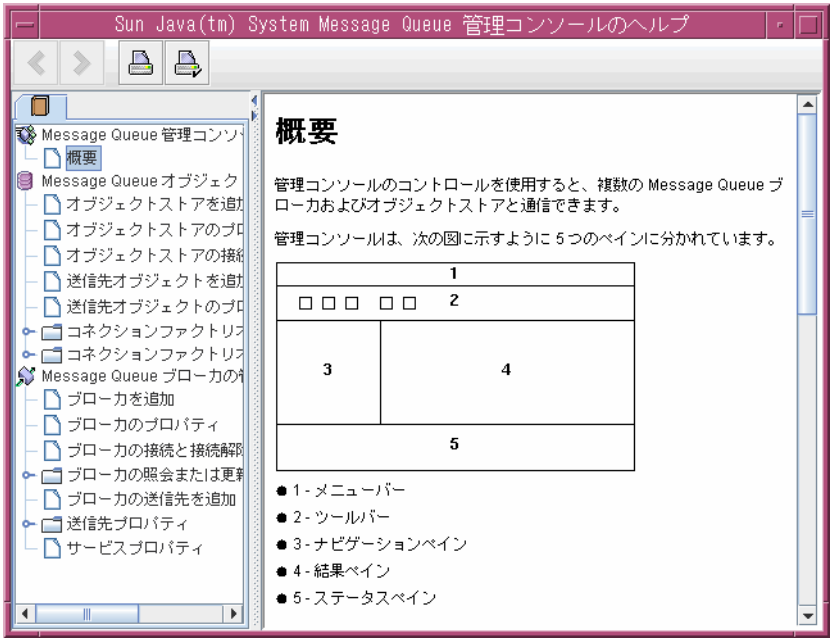
この章には完全な情報が記載されているわけではないので、まず管理コンソールに関するヘルプ情報の使い方を理解しておきましょう。

ヘルプを利用する

ヘルプメニューは、メニューバーの一番右にあります。

➤ 管理コンソールのヘルプ情報を表示する

1. ヘルプメニューをプルダウンして、「Overview (概要)」を選択します。ヘルプウィンドウが表示されます。



ヘルプ情報の構成を確認します。左のナビゲーションペインには目次が表示され、右の結果ペインにはナビゲーションペインで選択したアイテムの内容が表示されます。

ヘルプウィンドウの結果ペインを確認します。これは管理コンソールの構成図で、コンソールの各ペインの使用方法を表しています。

2. ヘルプウィンドウのナビゲーションペインを見ます。トピックが、概要、オブジェクトストアの管理、ブローカの管理の 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。各フォルダには、複数タブのついたダイアログボックスのヘルプがあり、各ファイルはダイアログボックス、またはタブの簡単なヘルプです。

   最初に行うコンソールの管理タスクは、44 ページの「ブローカの追加」に示すように、コンソールを使用して管理する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、オンラインヘルプの情報を確認します。

3. ヘルプウィンドウのナビゲーションペインにある、「Add Broker (ブローカを追加)」アイテムをクリックします。

   結果ペインが変更されています。ここには、ブローカを追加するとはどういうことかを説明するテキストと、「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が表示されます。フィールド名は、太字で表示されます。

4. ヘルプテキストを読みます。

5. ヘルプウィンドウを閉じます。

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。

- 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。
- 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「スタート」メニューを使用している場合、コマンドウィンドウが表示されます。コマンドの応答が表示され、次のような行を表示してブローカが使用できること示します。

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

管理コンソールのウィンドウに戻ります。これで Console 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。

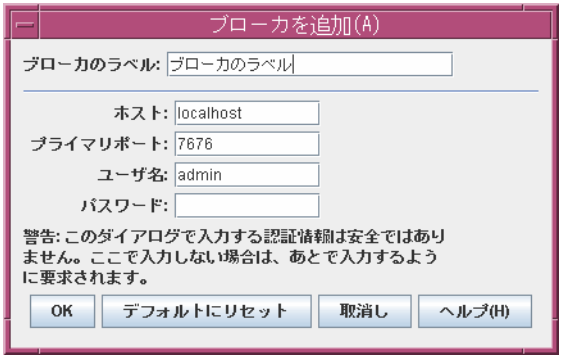
ブローカの追加

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. ナビゲーションペインの「Brokers (ブローカ)」をクリックし、「Add Broker (ブローカを追加)」を選択します。
2. 「Broker Label (ブローカのラベル)」フィールドに `MyBroker` と入力します。

   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。

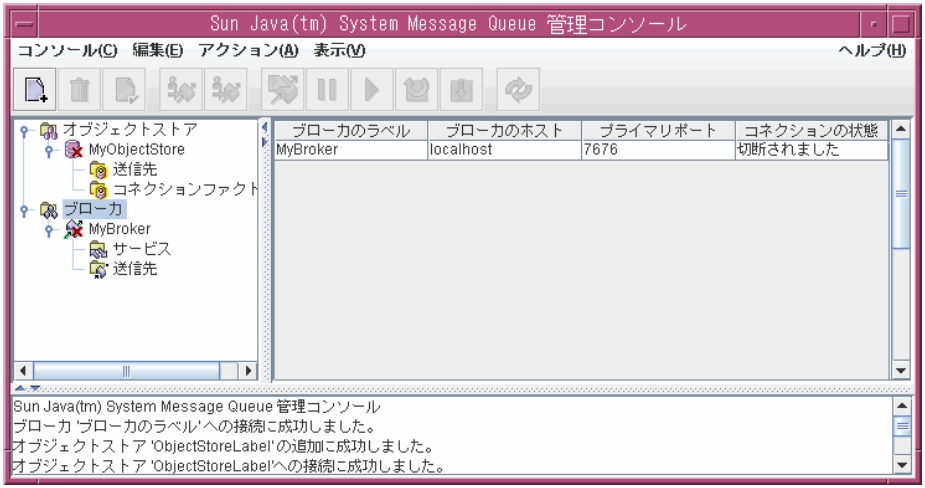


   ダイアログボックスで指定されたデフォルトのホスト名 (`localhost`) と、プライマリポート (`7676`) を控えておきます。これらの値はあとで、クライアントがこのブローカへのコネクションを設定するのに必要なコネクションファクトリを設定するときに、この値を指定する必要があります。

   「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、コネクションを実行するときに指定したほうがより安全です。

3. 「OK」をクリックして、ブローカを追加します。

   ナビゲーションペインを見ます。追加したブローカが、「Brokers (ブローカ)」の下に一覧表示されています。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コンソールに接続されていないことを表しています。



4. 「MyBroker」を右クリックし、ポップアップメニューから「Properties (プロパティ)」を選択します。

   ブローカのプロパティ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、ブローカの追加時に指定したプロパティを更新できます。

5. 「Cancel (取消し)」をクリックしてダイアログを閉じます。

ブローカに接続する

➤ **ブローカに接続する**

1. 「`MyBroker`」を右クリックし、「Connect to Broker」を選択します。

   ダイアログボックスが表示され、ユーザー名とパスワードの入力が要求されます。

   デフォルトでは、管理コンソールは `admin` というパスワードを持つ `admin` というユーザーとしてブローカに接続できます。この演習では、デフォルトの値を使用します。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。詳細は 143 ページの「ユーザーの認証」を参照してください。

2. 「Password (パスワード)」フィールドに、admin と入力します。

   ユーザー名の admin を指定し、正しいパスワードを入力すると、管理者権限でブローカに接続します。

3. 「OK」をクリックして、ブローカに接続します。

ブローカに接続後、「Actions (アクション)」メニューを選択すると、ブローカに関する情報の入手、ブローカの停止、再開、シャットダウン、再起動、およびブローカからの切断を実行できます。

コネクションサービスを表示する

ブローカは、提供するコネクションサービスと、サポートする物理的送信先とで識別されます。

➤ 利用可能なコネクションサービスを表示する

1. ナビゲーションペインで「Services (サービス)」を選択します。

   結果ペインに利用可能なサービスが一覧表示されます。各サービスに対して、名前、ポート番号、状態が表示されます。

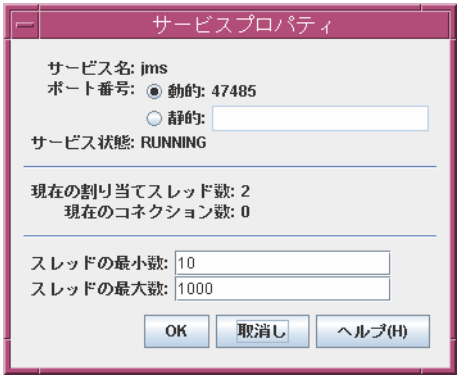
2. 結果ペインにある jms サービスをクリックして選択します。

3. 「Actions (アクション)」メニューをプルダウンして、強調表示されているアイテムを確認します。

   jms サービスを停止したり、プロパティを表示、更新したりするオプションがあります。

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

「Service Properties (サービスプロパティ)」ダイアログを使用して、サービスに静的なポート番号を割り当て、このサービス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。



5. 「OK」をクリックするか、あるいは「Cancel (取消し)」をクリックして、「Service Properties (サービスのプロパティ)」ダイアログボックスを閉じます。
6. 結果ペインで admin サービスを選択します。
7. 「Actions (アクション)」メニューをプルダウンします。

   このサービスを停止することはできませんので注意してください (停止アイテムは無効)。admin サービスとは、ブローカへの管理者のリンクです。これを停止すると、ブローカに接続できなくなります。
8. 「Actions (アクション)」> 「Properties (プロパティ)」の順に選択し、admin サービスのプロパティを表示します。
9. 完了したら、「OK」または「Cancel (取消し)」をクリックします。

ブローカに物理的送信先を追加する

デフォルトでは、ブローカに対して物理的送信先の自動作成が有効になっています。自動作成により、ブローカは物理的送信先を動的に作成できます。

開発環境では、クライアントコードをテストするために物理的送信先を明示的に作成する必要はありません。

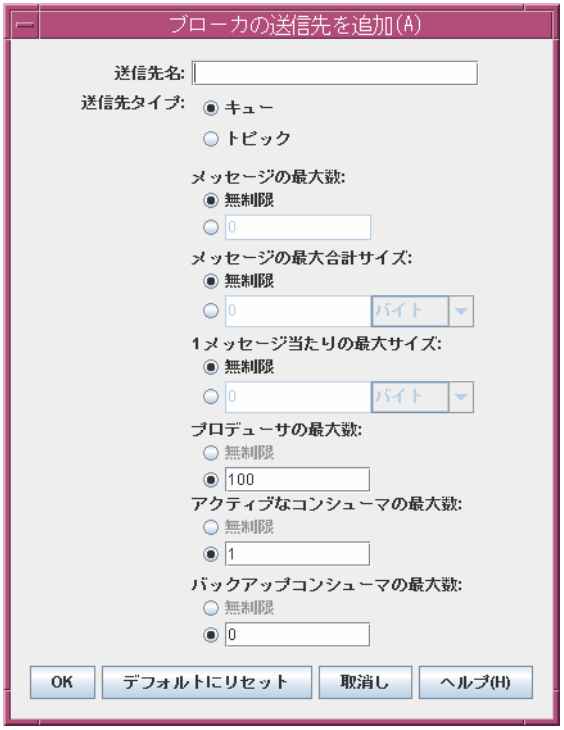
運用時の設定では、物理的送信先を明示的に作成することをお勧めします。そうすることにより、管理者は、ブローカが使用している物理的送信先を十分に認識できます。

次に、ブローカに物理的送信先を追加します。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。

➤ ブローカにキュー送信先を追加する

1. 「MyBroker」の「Destinations (送信先)」ノードを右クリックし、「Add Broker Destination (ブローカの送信先を追加)」を選択します。

   次のダイアログボックスが表示されます。



2. 「Destination Name (送信先名)」フィールドに `MyQueueDest` と入力します。

3. 選択されていない場合は、「Queue (キュー)」ラジオボタンを選択します。
4. 「OK」をクリックして、物理的送信先を追加します。

   物理的送信先が結果ペインに表示されます。

物理的送信先を管理する

ブローカに物理的送信先を追加すると、以下の手順に従い、次のタスクを実行できるようになります。

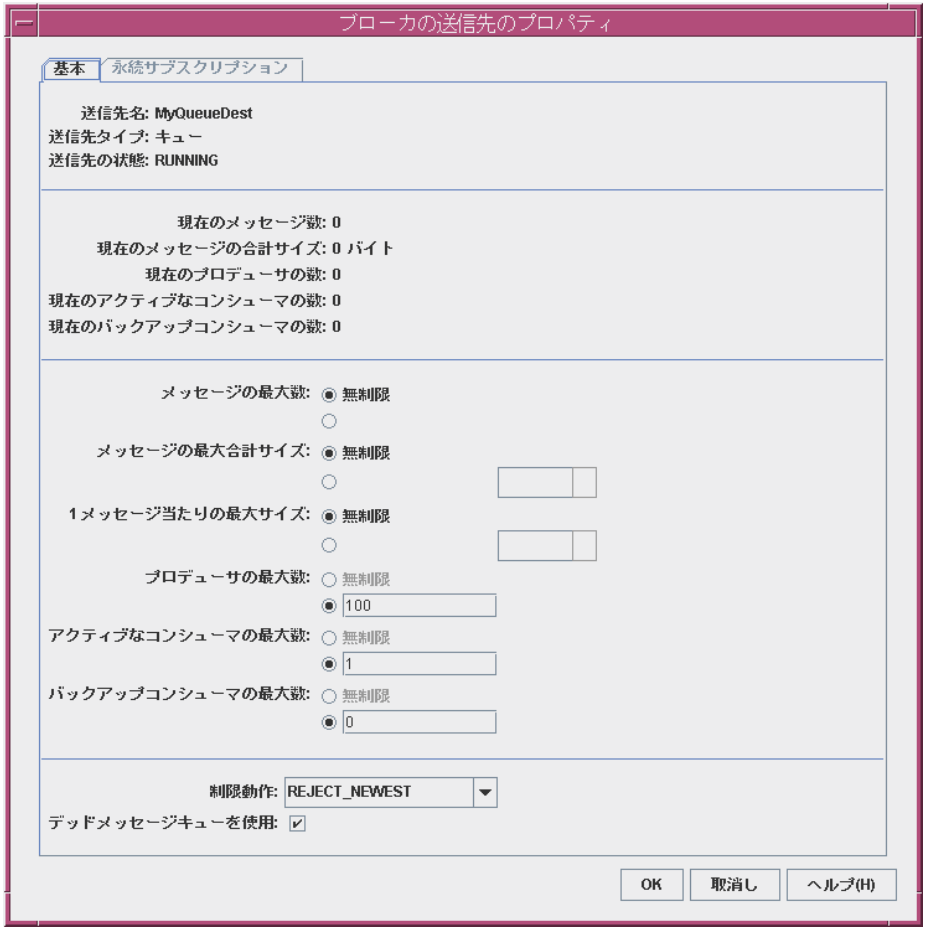
- 物理的送信先のプロパティを表示および更新する
- 物理的送信先でメッセージをパージする
- 物理的送信先を削除する

➤ 物理的送信先のプロパティを表示する

1. 「MyBroker」の「Destination (送信先)」を選択します。

   結果パネルに 2 つの物理的送信先、MyQueueDest と `mq.sys.dmq` が表示されます。送信先 `mq.sys.dmq` は、ブローカの有効期限の切れたメッセージと拒否されたメッセージを格納する、システムで作成されたキューです。現在のところ、このデッドメッセージキューは無視してください。
2. 結果ペインで、「MyQueueDest」を選択します。
3. 「Actions (アクション)」> 「Properties (プロパティ)」の順に選択します。

   次のダイアログボックスが表示されます。



ダイアログボックスには、キューに関する現在の状態情報と変更可能な一部のプロパティが表示されている点に注意してください。

4. 「Cancel (取消し)」をクリックしてダイアログボックスを閉じます。

➤ 物理的送信先からメッセージをパージする

1. 結果ペインで物理的送信先を選択します。
2. 「Actions (アクション)」> 「Purge Messages (メッセージをパージする)」の順に選択します。

   確認ダイアログボックスが表示されます。

メッセージをパージすると、メッセージが削除されたあとに空の送信先が残ります。

➤ 送信先を削除する

1. 結果ペインで物理的送信先を選択します。
2. 「Edit (編集)」> 「Delete (削除)」の順に選択します。

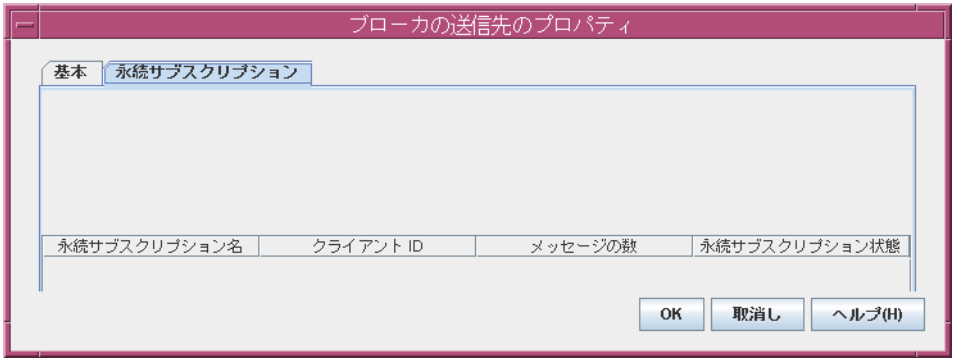
   確認ダイアログボックスが表示されます。

注	MyQueueDest キュー送信先は削除しないでください。

物理的送信先の削除を行うと、その送信先にあるメッセージがパージされたあと、送信先も削除されます。

トピック情報を入手する

ブローカのトピック送信先プロパティのダイアログボックスには、永続サブスクリプションに関する情報を一覧表示する追加タブがあります。このタブは、キューの場合は無効です。



このダイアログボックスを使用して、次のことを実行できます。

- 永続サブスクリプションに関連するすべてのメッセージを削除して、永続サブスクリプションをパージする
- 永続サブスクリプションに関連するすべてのメッセージをパージして、永続サブスクリプションを削除する

オブジェクトストアを操作する

オブジェクトストアは、Message Queue の管理対象オブジェクトの格納に使用されます。この管理対象オブジェクトは、Message Queue 固有の実装と、クライアントアプリケーションで使用されるオブジェクトに関する設定情報をカプセル化します。オブジェクトストアは、LDAP ディレクトリサーバーかファイルシステムストア (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。

管理対象オブジェクトは、クライアントコード内でインスタンス化し、設定できます。ただし、これらのオブジェクトは、JNDI を使用してクライアントアプリケーションからアクセスされるオブジェクトストアで、管理者が作成、設定、保存する方がよいでしょう。これにより、クライアントコードはプロバイダに依存しなくなります。

管理コンソールを使用して、オブジェクトストアを作成することはできません。まず、次の節で説明する操作を行う必要があります。

オブジェクトストアを追加する

オブジェクトストアを追加すると、管理コンソールにある既存のオブジェクトストアへの参照が作成されます。この参照は、コンソールを終了して、再起動しても保持されます。

➤ ファイルシステムオブジェクトストアを追加する

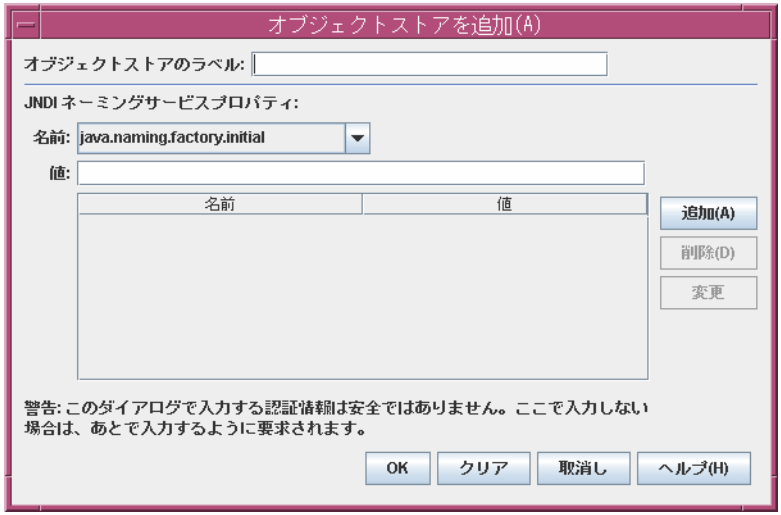
1. C ドライブに `Temp` という名前のフォルダが存在しない場合は、ここで作成します。

   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` というフォルダにあることが前提になっています。一般には、ファイルシステムオブジェクトストアは、任意のドライブの任意のディレクトリに置くことができます。

   Windows 以外 : 既存の `/tmp` ディレクトリを使用します。

2. 「Object Stores (オブジェクトストア)」で右クリックし、「Add Object Store (オブジェクトストアを追加)」を選択します。

   次のダイアログボックスが表示されます。



3. 「ObjectStoreLabel (オブジェクトストアのラベル)」フィールドに「`MyObjectStore`」と入力します。

   ここでは、管理コンソールにあるオブジェクトの表示用のラベルが設定されるだけです。

   次の手順では、JNDI の名前と値の組み合わせを入力する必要があります。これらの組み合わせは、JMS 準拠のアプリケーションが管理対象オブジェクトを検索するときに使用されます。

4. 「Name (名前)」ドロップダウンリストから、`java.naming.factory.initial` を選択します。

   このプロパティで、どの JNDI サービスプロバイダを使用するかを指定できます。たとえば、ファイルシステムサービスプロバイダ、または LDAP サービスプロバイダです。

5. 「Value (値)」フィールドに、次のように入力します。

   `com.sun.jndi.fscontext.RefFSContextFactory`

   つまり、ファイルシステムストアが使用されます。LDAP ストアの場合は、`com.sun.jndi.ldap.LdapCtxFactory` を指定します。

   本稼動環境では、オブジェクトストアとして LDAP ディレクトリサービスを使用する場合があります。サーバーの設定と JNDI 検索については、174 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。

6. 「Add (追加)」ボタンをクリックします。

   プロパティの要約ペインにプロパティとその値が一覧表示されます。

7. 「Name (名前)」ドロップダウンリストから、`java.naming.provider.url` を選択します。

   このプロパティで、オブジェクトストアの正確な場所を指定できます。ファイルシステムのオブジェクトストアでは、これが既存のディレクトリ名になります。

8. 「Value (値)」フィールドに、次のように入力します。

   `file:///C:/Temp`

   (Solaris と Linux 上では、`file:///tmp`)

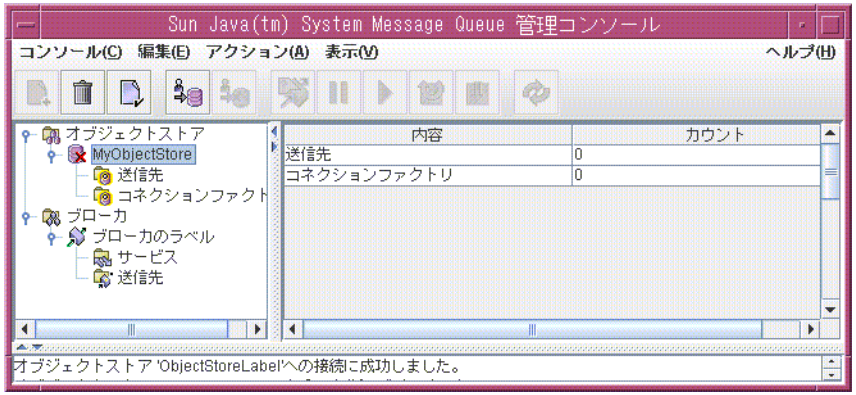
9. 「Add (追加)」ボタンをクリックします。

   プロパティの要約ペインにプロパティとその値がともに一覧表示されます。LDAP サーバーを使用している場合、認証情報も指定する必要があります。ファイルシステムストアでは、認証情報の指定は必要ありません。

10. 「OK」をクリックして、オブジェクトストアを追加します。

11. MyObjectStore ノードがナビゲーションペインで選択されていない場合は、ここで選択します。

    管理コンソールは次のように見えます。



オブジェクトストアはナビゲーションペインに一覧表示され、その内容である送信先とコネクションファクトリは結果ペインに一覧表示されます。まだ管理対象オブジェクトをオブジェクトストアに追加していないため、結果ペインの「Count (カウント)」カラムに表示されます。

ナビゲーションペインにあるオブジェクトストアのアイコンには、X がついています。これは、まだ接続されていないことを表します。オブジェクトストアを使用する前に、接続する必要があります。

オブジェクトストアプロパティを確認する

管理コンソールがオブジェクトストアから切断されている間は、オブジェクトストアのプロパティの一部を確認および変更できます。

➤ オブジェクトストアのプロパティを表示する

1. ナビゲーションペインの「MyObjectStore」を右クリックします。
2. ポップアップメニューから「Properties (プロパティ)」を選択します。

   オブジェクトストアを追加したときに指定した、すべてのプロパティを示すダイアログボックスが表示されます。任意のプロパティを変更し、「OK」をクリックして古い情報を更新します。
3. 「OK」をクリックするか、「Cancel (取消し)」をクリックして、ダイアログボックスを終了します。

オブジェクトストアに接続する

オブジェクトストアにオブジェクトを追加する前に、まず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. ナビゲーションペインの「MyObjectStore」を右クリックします。
2. ポップアップメニューから「Connect to Object Store (オブジェクトストアに接続)」を選択します。

   オブジェクトストアのアイコンに X がついていないことを確認します。これで、コネクションファクトリや送信先のオブジェクトを、オブジェクトストアに追加できます。

コネクションファクトリ管理対象オブジェクトを追加する

管理コンソールを使用して、コネクションファクトリを作成および設定できます。コネクションファクトリは、クライアントコードがブローカに接続するときに使用されます。コネクションファクトリを設定すると、作成するときに使用されるコネクションの動作を制御できます。

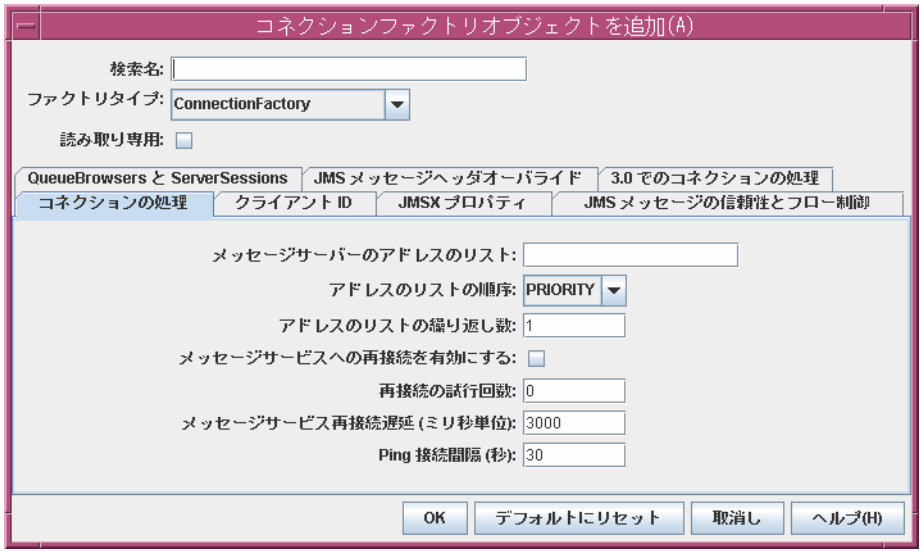
コネクションファクトリの設定については、オンラインヘルプと『Message Queue Developer's Guide for Java Clients』を参照してください。

注	管理コンソールは、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。

➤ コネクションファクトリをオブジェクトストアに追加する

1. まだ接続されていない場合は、MyObjectStore に接続します (55 ページの「オブジェクトストアに接続する」を参照)。
2. 「Connection Factories (コネクションファクトリ)」ノードを右クリックし、「Add Connection Factory Object (コネクションファクトリオブジェクトを追加)」を選択します。

   「Add Connection Factory Object (コネクションファクトリオブジェクトを追加)」ダイアログボックスが表示されます。



3. 「Lookup Name (検索名)」フィールドに「`MyQueueConnectionFactory`」という名前を入力します。

   この名前は、次のコマンド行のように `HelloWorldMessageJNDI.java` からコネクションファクトリを検索するときにクライアントコードが使用する名前です。

   ```
   qcf=(javax.jms.QueueConnectionFactory)
          ctx.lookup("MyQueueConnectionFactory")
   ```

4. プルダウンメニューから「QueueConnectionFactory」を選択し、コネクションファクトリのタイプを指定します。

5. 「コネクションの処理」タブをクリックします。

6. 「Message Server Address List (メッセージサーバーアドレスリスト)」フィールドには、通常、クライアントの接続先となるブローカのアドレスを入力します。このフィールドの例を次に示します。

   ```
   mq://localhost:7676/jms
   ```

   デフォルトでは、コネクションファクトリは、ポート `7676` の localhost で実行しているブローカに接続するように設定されているため、値を入力する必要はありません。クイックスタートの例でも、このデフォルト設定を前提にしています。

7. このダイアログボックスのタブをクリックして、コネクションファクトリの設定可能な情報の種類を確認します。「Add Connection Factory Object (コネクションファクトリオブジェクトを追加)」ダイアログボックスの右側下部にある「Help (ヘルプ)」ボタンを使用して、それぞれのタブの情報を確認します。ここでは、デフォルト値を変更しないでください。

8. 「OK」をクリックして、キューコネクションファクトリを作成します。
9. 結果ペインを確認します。新規作成されたコネクションファクトリの検索名と検索タイプが一覧表示されています。

送信先オブジェクトの追加

送信先管理対象オブジェクトは、ブローカの物理的送信先に関連付けられ、これらの送信先を指します。送信先管理対象オブジェクトにより、クライアントはプロバイダ固有の送信先名と設定とは関係なく、物理的送信先を検索できます。

クライアントはメッセージを送信するときに、送信先管理対象オブジェクトを検索またはインスタンス化し、JMS API の `send()` メソッドの送信先管理対象オブジェクトを参照します。ブローカは、その管理対象オブジェクトに関連する物理的送信先へのメッセージの配信に責任を持ちます。

- その管理対象オブジェクトに関連する物理的送信先を作成した場合、ブローカはその物理的送信先にメッセージを配信します。
- 物理的送信先を作成せず、物理的送信先の自動作成が有効になっている場合、ブローカ自身が物理的送信先を作成し、その送信先にメッセージを配信します。
- 物理的送信先を作成せず、物理的送信先の自動作成が無効になっている場合、ブローカは物理的送信先を作成できず、メッセージも配信できません。

クイックスタートの次の部分では、これまでに追加した物理的送信先に対応する管理対象オブジェクトを追加します。

➤ 送信先をオブジェクトストアに追加する

1. ナビゲーションペインの「MyObjectStore」ノードの下にある「Destinations (送信先)」ノードを右クリックします。
2. 「Add Destination Object (送信先オブジェクトを追加)」を選択します。

   管理コンソールにより、オブジェクトに関する情報を指定する「Add Destination Object (送信先オブジェクトを追加)」ダイアログボックスが表示されます。

3. 「Lookup Name (検索名)」フィールドに `MyQueue` と入力します。

   検索名は、JNDI 検索呼び出しを使用しているオブジェクトを検索するのに使用されます。サンプルアプリケーションでは、呼び出しは次のとおりです。

   ```
   queue=(javax.jms.Queue)ctx.lookup("MyQueue");
   ```

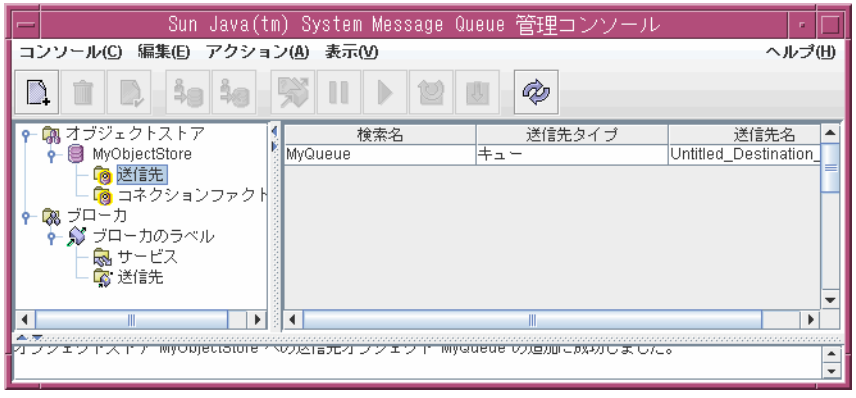
4. 「Destination Type (送信先タイプ)」の「Queue (キュー)」ラジオボタンを選択します。

5. 「Destination Name (送信先名)」フィールドに `MyQueueDest` と入力します。

   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 (48 ページの「ブローカに物理的送信先を追加する」を参照)。

6. 「OK」をクリックします。

7. ナビゲーションペインの「Destinations (送信先)」を選択し、追加したキュー管理対象オブジェクトに関する情報がどのように結果のペインに表示されているかを確認します。



管理対象オブジェクトのプロパティの表示

管理対象オブジェクトのプロパティを表示または更新するには、ナビゲーションペインにある「Destinations (送信先)」または「Connection Factories (コネクションファクトリ)」を選択し、結果ペインで特定のオブジェクトを選択してから、「Actions (アクション)」>「Properties (プロパティ)」の順に選択します。

➤ 送信先オブジェクトのプロパティを表示または更新する

1. ナビゲーションペインで MyObjectStore の送信先ノードを選択します。
2. 結果ペインで、「MyQueue」を選択します。
3. 「Actions (アクション)」>「Properties (プロパティ)」の順に選択し、「Destination Object Properties (送信先オブジェクトのプロパティ)」ダイアログボックスを表示します。

   ここで変更できる値は、送信先名と説明だけです。検索名を変更するには、オブジェクトを削除してから、新しいキュー管理対象オブジェクトを、希望する検索名で追加する必要があります。
4. 「Cancel (取消し)」をクリックしてダイアログを閉じます。

コンソール情報を更新する

オブジェクトストアを操作している場合も、ブローカを操作している場合も、「View (表示)」>「Refresh (更新)」の順に選択すると、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。

サンプルアプリケーションを実行する

このクイックスタートでは、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 の検索名がそれぞれ MyQueueConnectionFactory と MyQueue であるキューのコネクションファクトリ管理対象オブジェクトとキューの管理対象オブジェクト

コードでは 1 つ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。

➤ HelloWorldMessageJNDI アプリケーションを実行する

1. `HelloWorldmessageJNDI` アプリケーションを含むディレクトリを現在のディレクトリにします。たとえば、次のように指定します。

   `cd IMQ_HOMEdemohelloworld¥helloworldmessagejndi` (Windows の場合)

   `cd /usr/demo/imq/helloworld/helloworldmessagejndi` (Solaris の場合)

   `cd /opt/sun/mq/examples/helloworld/helloworldmessagejndi` (Linux の場合)

   HelloWorldMessageJNDI.class ファイルが存在していることを確認します。アプリケーションに変更を加える場合は、『Message Queue Developer's Guide for C Clients』のクイックスタートチュートリアルに記載されたクライアントアプリケーションのコンパイルに関する指示に従って、アプリケーションを再コンパイルする必要があります。`CLASSPATH` 変数に `HelloWorldMessageJNDI.class` ファイルと Message Queue 製品に組み込まれた `jms.jar`、`imq.jar`、および `fscontext.jar` といった jar ファイルを含む現在のディレクトリを追加します。`CLASSPATH` 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。

   JNDI jar ファイル (`jndi.jar`) は JDK 1.4 にバンドルされています。この JDK を使用している場合は、`jndi.jar` を `CLASSPATH` 設定値に追加する必要はありません。それ以前のバージョンの JDK を使用している場合は、`jndi.jar` を `CLASSPATH` に含める必要があります。詳細は、『Message Queue Developer's Guide for Java Clients』を参照してください。

2. 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。このファイルには短いながら詳細な説明が用意されており、作成した管理対象オブジェクトや送信先を使用する方法が明確に理解できます。

3. 次のコマンドのどちらかを実行して、HelloWorldMessageJNDI アプリケーションを稼働します。

   java HelloWorldMessageJNDI (Windows の場合)

   % java HelloWorldMessageJNDI file:///tmp (Solaris と Linux の場合)

   アプリケーションが問題なく実行されると、次の出力があります。

```
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。

この章では、次の節について説明します。

ブローカインスタンスの設定は、設定ファイルセットと、設定ファイル内の対応するプロパティをオーバーライドする `imqbrokerd` コマンドで渡されるオプションによって決まります。ブローカの設定については、73 ページの第 4 章「ブローカの設定」を参照してください。

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 の場合のファイル記述子の制限を実行します。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 (TimeToLive) 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、TimeToLive 機能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 Time 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を設定する (Solaris または Linux)

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 システムでは、クライアントが行うコネクション、あるいはブローカが受け付けるコネクションはすべて、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、これらの要因によってコネクション数が制限されます。ファイル記述子の制限を変更しないかぎり、ブローカまたはクライアントが、Solaris では 256、Linux では 1024 を超えるコネクションを実行することはできません。持続性のためにファイル記述子を使用したことで生じるコネクション数の制限は、実際にはこの値より小さくなります。

ファイル記述子の制限を変更するには、`ulimit` マニュアルページを参照してください。クライアントまたはブローカが実行されるそれぞれのシェルに対して、この制限を変更する必要があります。

ブローカのインタラクティブな起動

imqbrokerd コマンドを使用すると、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 (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 (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が起動します。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 umask を使用して、設定情報と持続データを含むブローカのインスタンスディレクトリに、アクセス権を設定します。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、ブローカのデフォルトのインスタンス (imqbroker) が起動されます。

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 プロパティを設定して、jms コネクション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、第 13 章「コマンドのリファレンス」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 SystemMessage 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。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 `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

`imqsvcadmin` コマンドの構文、サブコマンド、オプションの詳細は、第 13 章「コマンドのリファレンス」を参照してください。

ブローカは Windows のサービスとしてインストールされると、システムの起動時に起動され、システムをシャットダウンするまでバックグラウンドで実行されます。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。

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 67 ページの「ブローカのインタラクティブな起動」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、`imqcmd` コマンドを使用します。

ブローカを Windows のサービスとして実行する場合は、ブローカは 2 つの実行可能プロセスとして、タスクマネージャに表示されます。

- 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

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。

ブローカサービスの再設定

Windows サービスを構成する手順は次のとおりです。

1. サービスを停止します。
2. サービスを削除します。
3. サービスを追加し、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。

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、-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、「Windows サービスコントロールパネルの開始パラメータ」フィールドで指定することも可能です。

「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 d:¥¥jdk1.3
```

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、imqsvcadmin コマンドの query オプションを使用します。

```
imqsvcadmin query

Service iMQ_Broker is installed.
Display Name: iMQ_Broker
Start Type: Manual
Binary location: c:¥Program Files¥Sun Microsystems¥
                    Message Queue 3.5¥bin¥imqbrokersvc
JavaHome: c:¥j2sdk1.4.0
Broker Args: -passfile d:¥imqpassfile
```

サービス開始時の問題のトラブルシューティング

サービスを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. イベントビューアを起動します。
2. 「ログ」>「アプリケーション」を選択します。
3. 「表示」>「最新の情報に更新」を選択して、エラーイベントを確認します。

Windows サービスとして実行中のブローカの削除

サービスとして実行中のブローカを削除するには、次のいずれかを実行します。

- コマンドを使用する。まず imqcmd shutdown bkr コマンドを使用してブローカをシャットダウンし、次に imqsvcadmin remove コマンドでサービスを削除します。

- コントロールパネルの Windows サービスの管理ツールを使用する。この機能は、Windows のバージョンごとに収められている場所が異なります。

完了したら、コンピュータを再起動します。

Message Queue 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。

アプリケーション開発者またはアプリケーションマニュアルは、アプリケーションで設定される属性値に関する情報を提供します。アプリケーションで設定される属性の一部には、オーバーライドできるものがあります。この場合、コマンド行で属性を指定します。

また、JNDI 検索によりコネクション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コネクションファクトリが戻される場合、そのコネクション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。このような場合、Message Queue はその属性をデフォルト値に設定します。コマンド行で属性を指定することで、属性をデフォルト以外の値に設定できます。

コマンド行で属性値を指定するには、Java アプリケーションの次のコマンド行構文を使用します。

```
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、コネクション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、クライアントアプリケーション MyMQClient を起動します。アプリケーションはポート 7677 のホスト OtherHost のブローカに接続し、アプリケーションで設定されたホスト名とポートセットをオーバーライドします。

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

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。

ブローカインスタンスの削除

この節では、Solaris または Linux でのブローカインスタンスの削除に関する詳細を説明します。Windows サービスの削除の詳細は、70 ページの「Windows サービスとして実行中のブローカの削除」を参照してください。

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。ブローカインスタンスを削除するためのコマンド形式は、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。

ブローカの削除に使用できるオプションのリストについては、281 ページの「コマンドのリファレンス」の `imqbrokerd` の詳細を参照してください。

Solaris または Linux では、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` プロパティを `NO` に設定します。

第 4 章

ブローカの設定

ブローカインスタンスが起動すると、その設定は一連の設定ファイルおよび `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、設定ファイルとコマンド行オプションが対話し、ブローカインスタンスを設定する方法と、各ブローカコンポーネントの機能について説明します。また設定プロパティを一覧表示して、設定方法を説明します。

この章では、次の節について説明します。

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

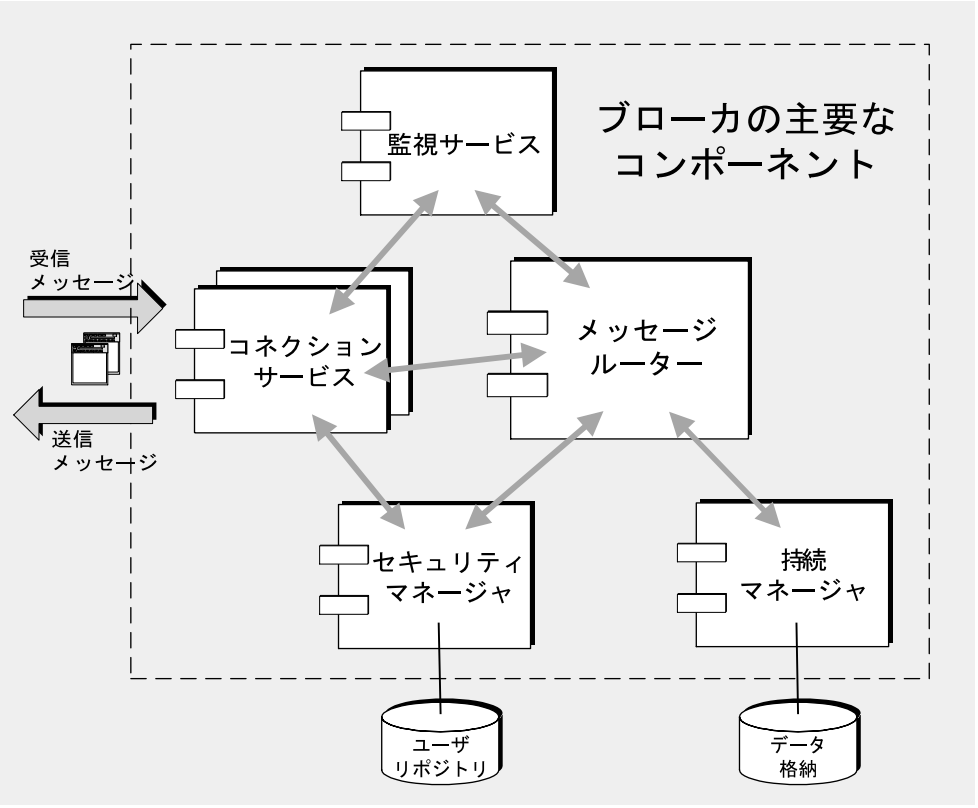
設定可能なブローカコンポーネントの概要

プロデューシングクライアントから送信先へ、さらに送信先から 1 つ以上のコンシューミングクライアントへ配信される Message Queue メッセージングシステムのメッセージ配信は、ブローカまたは、並行して動作するブローカインスタンスのクラスタによって行われます。

メッセージ配信を実行するには、ブローカはクライアントとの通信チャネルの設定、認証と承認、適切なメッセージルーティング、信頼性の高い配信の保証、およびシステムパフォーマンスを監視するためのデータの提供を行う必要があります。

その機能を実行するために、ブローカはさまざまな内部コンポーネントを使用します。各コンポーネントには、配信プロセスにおける特定のロールが割り当てられています。これらのブローカコンポーネントについて、図 4-1 に図示しています。

図 4-1 ブローカサービスのコンポーネント



メッセージルーターコンポーネントが主要なメッセージルーティングと配信サービスを実行し、それ以外のコンポーネントは重要なサポートサービスを提供しています。表 4-1 で各コンポーネントを簡単に説明しています。

表 4-1 主要なブローカサービスコンポーネントと機能

コンポーネント	説明 / 機能	プロパティの説明
コネクションサービス	ブローカと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。	316 ページの「コネクションサービスのプロパティ」

表 4-1 主要なブローカサービスコンポーネントと機能 (*続き*)

コンポーネント	説明 / 機能	プロパティの説明
メッセージルーター	メッセージのルーティングおよび配信を管理します。JMS メッセージと、JMS メッセージの配信をサポートするために Message Queue メッセージングシステムが使用するコントロールメッセージを含みます。	319 ページの「メッセージルーターのプロパティ」
持続マネージャ	データの持続ストレージへの書き込みと、持続ストレージからのデータの取得を管理します。	323 ページの「持続マネージャのプロパティ」
セキュリティマネージャ	ブローカへのコネクションを要求するユーザーに認証サービスを提供し、認証されたユーザーに承認サービス (アクセス制御) を提供します。	328 ページの「セキュリティマネージャのプロパティ」
監視サービス	さまざまな出力チャネルに書き込み可能なメトリックスと診断情報を生成する。管理者はこれらをブローカの監視および管理に使用できます。	333 ページの「監視とロギングのプロパティ」

負荷状態やアプリケーションの複雑さなどに応じて、これらのコンポーネントを設定してブローカのパフォーマンスを最適化することができます。次の節では、各コンポーネントが実行する機能と、コンポーネントの動作を変化させるために設定できるプロパティを説明します。

コネクションサービス

Message Queue ブローカは、Message Queue アプリケーションクライアントと Message Queue 管理クライアントの両方の通信をサポートしています。各コネクションサービスは、次のように、サービスタイプとプロトコルタイプで指定されます。

- *service type* は、JMS メッセージ配信 (`NORMAL`) サービス、または Message Queue 管理 (`ADMIN`) サービスのどちらを提供するのかを指定します。
- *protocol type* は、サービスをサポートする基礎となるトランスポートプロトコルレイヤーを指定します。

表 4-2 に、Message Queue ブローカから利用できるコネクションサービスを一覧にします。

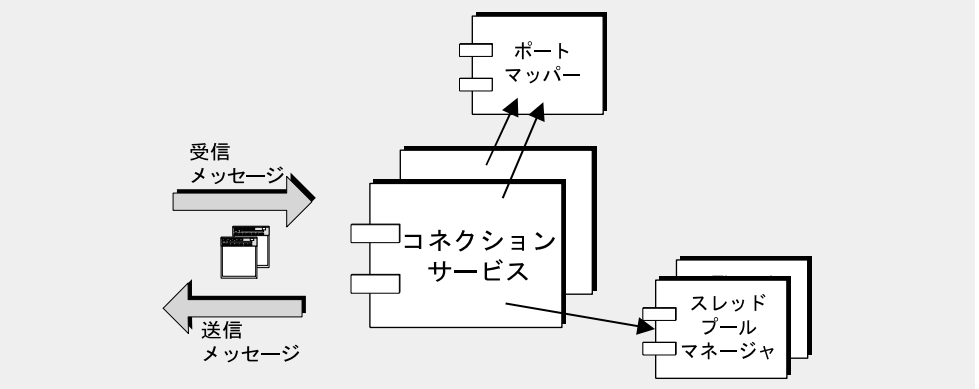
表 4-2　ブローカがサポートするコネクションサービス

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)
httpjms (Enterprise Edition)	NORMAL	http
httpsjms (Enterprise Edition)	NORMAL	https (SSL ベースセキュリティ)
admin	ADMIN	tcp
ssladmin	ADMIN	tls (SSL ベースセキュリティ)

ブローカを設定して、これらのコネクションサービスの一部、またはすべてを実行することができます。各コネクションサービスは、ブローカのホスト名とポート番号で指定した特定のポートで使用できます。`jms` サービスと `admin` サービスは、デフォルトで有効に設定されています。

Message Queue はコネクションサービスを動的にポート番号にマップします。あるいはユーザーがポートを明示的に割り当てられます。図 4-2 に示すように、各サービスはスレッドプールマネージャを保持し、共通のポートマッパーサービスにサービス自体を登録します。

図 4-2　コネクションサービスのサポート



次の節では、コネクションサービスとポートマッパー、およびスレッドプールマネージャとの関連性を説明します。

ポートマッパー

Message Queue には、ポートをコネクションサービスにマップするポートマッパーが用意されています。ポートマッパーは、標準のポート番号 7676 に常駐します。クライアントがブローカとの接続を設定すると、まずポートマッパーに接続し、指定されたコネクションサービスのポート番号を要求します。

jms、ssljms、admin、および ssladmin コネクションサービスのポート番号は、動的にも静的にもなります。デフォルトでは、コネクションサービスは起動時に動的にポートを設定します。また、サービスに静的なポートを指定できますが、静的なポート番号は一般には推奨されません。通常、静的なポート番号は、ファイアウォールを通過するコネクションなど、特別な状況に対してのみ使用されます。

httpjms サービスと httpsjms サービスは、それぞれ付録 C 「HTTP/HTTPS のサポート」の 389 ページの表 C-1 と 401 ページの表 C-3 に示すプロパティを使用して設定します。

スレッドプールマネージャ

各コネクションサービスは、複数のコネクションをサポートするマルチスレッドです。これらのコネクションに必要なスレッドは、スレッドプールマネージャのコンポーネントが管理するスレッドプールに保存されます。

スレッドプールマネージャを設定して、スレッドプールに保持されるスレッドの最小数と最大数を指定することができます。コネクションでスレッドが必要になると、スレッドプールにスレッドが追加されます。スレッドの最小数より少なくなると、システムは最小数のしきい値になるまで、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。新しいスレッドを頻繁に作成する必要がないように、最小数に十分な数を指定します。コネクションの負荷が重い場合、スレッドの数をスレッドプールの最大数まで増やすことができます。それでも足らない場合、スレッドが利用できるようになるまで、コネクションは待機します。

スレッドプール内のスレッドは、1 つのコネクションだけに割り当てる (専用モデル) か、必要に応じて、複数のコネクションに割り当てる (共有モデル) ことができます。

専用モデル : ブローカへのコネクションごとに、コネクションの受信メッセージの処理用と、コネクションの送信メッセージの処理用の 2 つの専用スレッドが必要です。このため、コネクションの数は、スレッドプール内の最大スレッド数の半分に制限されますが、高いパフォーマンスを得られます。

共有モデル (Enterprise Edition): コネクションは、メッセージが送信されるか受信されると必ず、共有スレッドによって処理されます。このモデルでは、コネクションごとの専用スレッドは必要ないため、コネクションサービス、さらにブローカがサポートできるコネクション数が多くなります。ただし、スレッドの共有にはある程度のパフォーマンスオーバーヘッドが伴います。スレッドプールマネージャは、一連のディストリビュータスレッ

ドを使用してコネクションのアクティビティを監視し、必要に応じてスレッドにコネクションを割り当てます。このアクティビティに伴うパフォーマンスオーバーヘッドは、各ディストリビュータスレッドが監視するコネクション数を制限することで最小限に抑えることができます。

セキュリティ

各コネクションサービスは、特定の認証および承認 (アクセス制御) 機能をサポートします (87 ページの「セキュリティマネージャ」を参照)。

コネクションサービスのプロパティ

これはコネクションサービスに関連した設定可能なプロパティです。

- `imq.service.activelist`。ブローカの起動時に起動するコネクションサービスのリスト。
- `imq.hostname`。1 台のコンピュータに、複数のネットワークインタフェースカードがある場合など、複数のホストを使用できる場合には、すべてのコネクションサービスがバインドするホストを指定します。
- `imq.portmapper.port`。ブローカのプライマリポートを指定します。ポートマッパーが常駐するポートです。
- `imq.portmapper.`*hostname*。複数のホストが使用できる場合、ポートマッパーがバインドするホストを指定します。
- `imq.portmapper.backlog`。ポートマッパーが、要求を拒否せずに、同時に処理可能な要求の最大数を指定します。オペレーティングシステムのバックログに格納可能な、ポートマッパーによる処理を待機中の要求の数を、プロパティで設定します。
- `imq.`*service_name*`.`*protocol_type*`.port`。`jms`、`ssljms`、`admin`、および `ssladmin` のサービスの場合のみ、指定したコネクションサービスのポート番号を指定します。
- `imq.`*service_name*`.`*protocol_type*`.hostname`。`jms`、`ssljms`、`admin`、および `ssladmin` のサービスの場合のみ、複数のホストが使用できる場合、指定したコネクションサービスがバインドするホストを指定します。
- `imq.`*service_name*`.min_threads`。指定したコネクション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数を指定します。
- `imq.`*service_name*`. max_threads`。指定したコネクション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数を指定します。新しいスレッドは、それ以上追加されなくなります。
- `imq.`*service_name*`.threadpool_model`。指定したコネクションサービスに対して、スレッドをコネクション専用にするのか、あるいは必要に応じてコネクションで共有するのかどちらかを指定します。

- imq.shared.connectionMonitor_limit。共有スレッドプールモデルの場合のみ、ディストリビュータスレッドで監視できるコネクションの最大数を指定します。

上記のプロパティの詳細は、316 ページの表 14-2 を参照してください。

メッセージルーター

サポートされているコネクションサービスを使用して、クライアントとブローカ間でコネクションが確立されると、メッセージのルーティングおよび配信が処理できます。

基本的な配信メカニズム

通常、ブローカで処理されるメッセージは、2 つのカテゴリに分類されます。

- プロデューサクライアントからコンシューマクライアントに送信される JMS ペイロードメッセージ
- クライアント間で送受信され、JMS メッセージの配信をサポートするコントロールメッセージ

受信メッセージが JMS メッセージの場合、送信先のタイプ (キュー、またはトピック) に基づいて、ブローカはメッセージをコンシューマクライアントにルートします。

- 送信先がトピックの場合、JMS メッセージは、すぐにトピックのすべてのアクティブサブスクライバにルートされます。永続サブスクライバが停止している場合、メッセージルーターは、サブスクライバがアクティブになるまでメッセージを保持し、アクティブになったらメッセージを配信します。
- 送信先がキューの場合、JMS メッセージは、対応するキューに配置され、メッセージがキューの先頭に来たときに、適切なコンシューマに配信されます。メッセージがキューの先頭に来る順番は、メッセージの到着順序と優先度によって変わります。

メッセージルーターは予定のコンシューマすべてにメッセージを配信した後、メモリーからメッセージをクリアします。メッセージが残っている場合、メッセージルーターはブローカの持続データストアからメッセージを削除します。

信頼性の高い配信 : 通知とトランザクション

信頼性の高い配信の要件を追加すると、前述した配信メカニズムはさらに複雑になります。信頼性の高い配信には、2 つの側面が関連します。

- ブローカ間のメッセージ配信の成功を保証する
- ブローカがメッセージの実際の配信前に、メッセージまたは配信情報を損失しないことを保証する

ブローカとのメッセージの配信を正常に行うために、Message Queue は多数の応答コントロールメッセージを使用します。

たとえば、プロデューサが送信先に JMS メッセージ (ペイロードメッセージ) を送信する場合、ブローカは JMS メッセージを受信したという応答を返します (デフォルトでは、プロデューサが JMS メッセージを持続的として指定している場合に限り、Message Queue はこれを実行する)。プロデューシングクライアントは、送信先への配信を保証するために、ブローカの応答を使用します。

同様に、ブローカが JMS メッセージをコンシューマに配信した場合、コンシューミングクライアントは、メッセージを受信および処理したことを示す通知を送り返します。クライアントは、セッションオブジェクトの作成時に、これらの通知を自動的に、またはどのくらいの頻度で送信するのかを指定しますが、メッセージルーターは、メッセージを配信した各メッセージコンシューマ (トピックの複数のサブスクライバなど) から通知を受信するまで、メモリーから JMS メッセージを削除しません。

トピックのサブスクリプションが永続的な場合、メッセージルーターは、各 JMS メッセージをその送信先で保持し、各永続サブスクライバがアクティブなコンシューマになると、そのメッセージを配信します。

メッセージルーターは、クライアントの通知を受信するたびにそれを記録し、すべての通知を受信すると、初めて JMS メッセージを削除します (ただし、この時点で JMS メッセージの有効期限が切れている場合は除く)。

さらに、メッセージルーターは、ブローカの応答をクライアントに送り返して、クライアントの通知を受信したことを確認します。コンシューミングクライアントは、ブローカの応答を使用して、ブローカが JMS メッセージを何度も配信しないようにします。ブローカがクライアントの通知を受信し損なうと、JMS メッセージが繰り返し配信される可能性があります。

ブローカがクライアントの通知を受信しないで、JMS メッセージを再び配信する場合、メッセージに再配信フラグが付けられます。一般にブローカは、次の状況で JMS メッセージを再配信します。

- ブローカがクライアントの通知を受信する前に、クライアントコネクションが終了し、その後、新しいコネクションが確立される。
- クライアントアプリケーションがセッションを復元する。
- クライアントアプリケーションがロールバックされたトランザクションを復元する。

たとえば、メッセージが通知される前に、キューのメッセージコンシューマがオフラインになって、別のコンシューマがキューに登録された場合、ブローカは通知されていないメッセージを新しいコンシューマに再配信します。

前述のクライアントの通知とブローカの応答は、トランザクションに分類される JMS メッセージの配信にも適用されます。この場合、各プロセスは個々の JMS メッセージの送信または受信レベルだけでなく、トランザクションレベルでも動作します。トランザクションがコミットされると、ブローカの応答が自動的に送信されます。

ブローカはトランザクションを追跡し、トランザクションがコミットされるか、あるいは障害が発生した場合にロールバックされるようにします。このトランザクション管理は、大規模な分散トランザクションの一部であるローカルトランザクションもサポートします。ブローカは、これらのトランザクションがコミットされるまで、トランザクションの状態を追跡します。デフォルトでは、ブローカは起動時に、コミットされていないすべてのトランザクションを調べて、PREPARED 状態以外のトランザクションをすべてロールバックします。imq.transaction.autorollback プロパティを設定する場合、ブローカは PREPARED 状態にあるトランザクションもロールバックします。

信頼性の高い配信：持続

信頼性の高い配信のもう 1 つの局面は、メッセージが実際に配信されるまで、ブローカがメッセージ、または配信情報を保持することです。通常、メッセージは配信されたり、有効期限が切れたりするまで、メモリー内に保持されます。ただし、ブローカに障害が発生すると、これらのメッセージは損なわれます。

プロデューサクライアントが、メッセージの持続性を指定している場合、メッセージルーターはこのメッセージを持続マネージャに渡します。持続マネージャは、データベースまたはファイルシステムにメッセージを格納し (83 ページの「持続マネージャ」を参照)、ブローカに障害が発生したときに、メッセージを復元できるようにします。

メモリーリソースとメッセージフローの管理

ブローカのパフォーマンスと安定性は、使用できるシステムリソースとメモリーなどのリソースの使用効率によって異なります。特に、メッセージルーターでは、メッセージの生成量が消費量をかなり上回る場合には、過負荷となりメモリーリソースを使い果たしてしまうことがあります。この問題の発生を避けるために、メッセージルーターは、リソースが不十分なときでもシステムの動作を維持できるように 3 レベルのメモリー保護を使用しています。

個々の送信先のメッセージ制限：メッセージ数およびメッセージで消費される合計メモリーに制限を指定する、物理的送信先プロパティを設定できます (第 15 章「物理的送信先のプロパティのリファレンス」を参照)。また、制限に達したときのメッセージルーターの動作も指定できます。4 種類の制限の動作は次のとおりです。

- メッセージプロデューサを遅くする (FLOW_CONTROL)
- メモリー内のもっとも古いメッセージを廃棄する (REMOVE_OLDEST)
- メッセージの有効期限に従い、メモリー内のもっとも優先度の低いメッセージを廃棄する (REMOVE_LOW_PRIORITY)

- もっとも新しいメッセージを拒否する (REJECT_NEWEST)

システム全体のメッセージ制限：システム全体のメッセージ制限は、二次的な保護を提供します。システム全体の制限を指定すると、システム上のすべての送信先に対して一括して適用され、メッセージの総数とすべてのメッセージによって使用される総メモリー量が制限されます (319 ページの表 14-3 を参照)。どちらかのシステム全体のメッセージ制限に達した場合、メッセージルーターは新しいメッセージを拒否します。

システムメモリーのしきい値：システムメモリーのしきい値は、三次的な保護を提供します。ブローカがさらに深刻な状況に陥ったときに、メモリーの過負荷を避けるためのアクションを実行できるように、使用可能なシステムメモリーのしきい値を指定できます。実行するアクションは、メモリーリソースの状態に応じて、次のように異なります。

- green (使用可能なメモリーが十分にある)
- yellow (ブローカのメモリーが減っている)
- orange (ブローカのメモリーが不十分である)
- red (ブローカのメモリーが不足している)

ブローカのメモリーの状態が green から yellow、orange、red へと進むにつれ、ブローカは次のタイプの本格的なアクションを段階的に実行します。

- アクティブなメモリーのメッセージを持続ストレージにスワップする (83 ページの「持続マネージャ」を参照)。
- 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る。持続メッセージのフローは、メッセージごとにブローカによって通知された要件によって自動的に制限されます。

これらのアクションはどちらもパフォーマンスを低下させます。

システムメモリーのしきい値に達する場合は、送信先メッセージの制限とシステム全体のメッセージ制限が低すぎます。場合によっては、潜在するメモリーの過負荷をしきい値がタイミングよく指摘できないことがあります。したがって、この機能に依存してメモリーリソースを制御するのではなく、メモリーリソースが最適化されるように個々の送信先とシステム全体を設定する必要があります。

メッセージルーターのプロパティ

メモリーリソースの管理には、システム全体の制限とシステムメモリーのしきい値があります。

- imq.destination.DMQ.truncateBody。デッドメッセージキューに、メッセージのヘッダーとプロパティデータのみが残るように指定します。メッセージ本体の内容は破棄されます。
- imq.message.expiration.interval。有効期限が切れたメッセージを再利用させる頻度を秒単位で指定します。
- imq.system.max_count。ブローカが保持するメッセージの最大数を指定します。

- `imq.system.max_size`。ブローカが保持するメッセージの最大合計サイズを指定します。
- `imq.message.max_size`。メッセージ本体の最大サイズを指定します。
- `imq.`*`resource_state`*`.threshold`。指定したメモリーリソースの状態で、そのメモリーリソースがトリガーされるメモリーの利用率を指定します。
- `imq.`*`resource_state`*`.count`。メモリーリソースの各状態がトリガーされたときに、一度に入力可能なメッセージの最大数を指定します。
- `imq.transaction.autorollback`。`PREPARED` 状態の分散トランザクションをブローカの起動時に自動的にロールバックするかどうかを指定します。

上記のプロパティの詳細は、319 ページの表 14-3 を参照してください。

持続マネージャ

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。この場合、ブローカは、すべての持続メッセージを重要な転送情報および配信情報とともにデータストアに保存する必要があります。持続マネージャのコンポーネントは、この情報の書き込みおよび検索を管理します。

障害が発生したブローカを復元する場合、配信されていないメッセージを復元するだけでは不十分です。ブローカは、次のタスクも実行する必要があります。

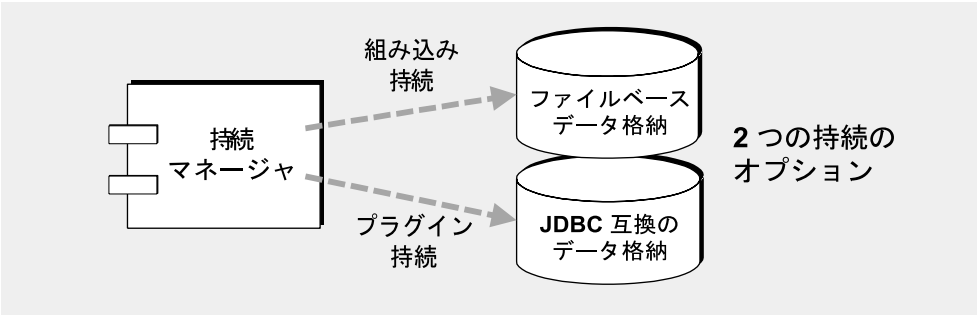
- 送信先の再作成
- 各トピックの永続サブスクリプションのリストの復元
- 各メッセージの通知リストの復元
- コミットされたすべてのトランザクションの状態の復元

持続マネージャは、このすべての状態情報の格納および検索を管理します。

ブローカが再起動すると、送信先と永続サブスクリプションの再作成、持続メッセージの復元、すべてのトランザクションの状態の復元、および配信されていないメッセージのルーティングテーブルの再作成が行われます。この後に、メッセージの配信を再開します。

Message Queue は、組み込み持続モジュールとプラグイン持続モジュールの両方をサポートしています (図 4-3 を参照)。組み込み持続は、ファイルベースのデータストアです。プラグイン持続は、JDBC™ (Java Database Connectivity) インタフェースを使用し、JDBC データストアを必要とします。通常、組み込み持続の方が、プラグイン持続より処理速度が速くなりますが、JDBC 互換のデータベースシステムを使用する冗長性および管理機能を好むユーザーもいます。

図 4-3　持続マネージャのサポート



組み込み持続

デフォルトの Message Queue 持続ストレージソリューションは、ファイルベースのデータストアです。この方法では、メッセージ、送信先、永続サブスクリプション、トランザクションなどの持続データを保存するために、個別のファイルを使用します。

ファイルベースのデータストアは、そのデータストア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配置されます (付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照)。

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

ファイルベースのデータストアは、常駐先の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。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。

メッセージが追加および削除されたときの断片化を減らすために、可変長レコードファイルを圧縮できます (136 ページの「物理的送信先の圧縮」を参照)。さらに、設定可能なしきい値 (`imq.persist.file.message.max_record_size`) より大きいメッセージは、可変長レコードファイルではなく、メッセージに該当するファイルへ組み込み持続マネージャが格納します。これらの個々のファイルでは、ファイルが再利用できるようにファイルプールが維持されます。メッセージファイルが不要になっても削除されません。その代わり、メッセージファイルは送信先ディレクトリの空きファイルのプールに追加され、新しいメッセージの保存に使用されます。

送信先ファイルプールの最大ファイル数を設定できます (`imq.persist.file.destination.message.filepool.limit`)。また、単に再利用のタグを設定するのではなく、ゼロまで切り捨てて削除されるファイルプール内の空きファイルの率も指定できます (`imq.persist.file.message.filepool.cleanratio`)。消去されるファイルの率が増加すると、ディスクスペースは減り、ファイルプールの維持に必要なオーバーヘッドが増加します。

シャットダウン時に、タグの付いたファイルを削除するかどうか (imq.persist.file.message.cleanup) を指定することもできます。ファイルが削除されると、ファイルが使用するディスクスペースが小さくなりますが、ブローカはシャットダウンに時間がかかるようになります。

そのほかの持続データはすべて (送信先、永続サブスクリプション、トランザクション) は、それぞれ個別のファイルに格納されます。つまり、送信先はすべて 1 つのファイルに、永続サブスクリプションはすべて別の 1 つのファイルに、といった具合になります。

信頼性を最大にするため、imq.persist.file.sync.enabled 属性を使用して、持続操作によりメモリー内の状態と物理的なストレージ装置とを同期するように指定できます。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。Sun Cluster 環境で Message Queue を実行している場合、クラスタのすべてのノードに対してこの属性を true に設定する必要があります。

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。保護する方法については、104 ページの「持続データの保護」を参照してください。

プラグイン持続

JDBC 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、ブローカを設定することができます。この作業には、さまざまな JDBC 関連のブローカ設定プロパティの設定、適切なスキーマでデータストアを作成するデータベースマネージャユーティリティ (imqdbmgr) の使用が含まれます。手順および関連する設定プロパティについては、99 ページの「持続ストアの設定」で説明します。

持続マネージャのプロパティ

このプロパティは、使用する持続の種類を指定します。

- imq.persist.store。組み込みのファイルベース (file) の持続、またはプラグインの JDBC 互換 (jdbc) の持続のどちらをブローカが使用するのかを指定します。

次のプロパティは、組み込み持続に関連したものです。

- imq.persist.file.sync.enabled。持続操作でメモリー内の状態を物理的なストレージと同期させるかどうかを指定します。
- imq.persist.file.message.max_record_size。メッセージストレージファイルに追加されるメッセージの最大サイズを指定します。
- imq.persist.file.destination.message.filepool.limit。送信先ファイルプール内の再利用できる空きファイルの最大数を指定します。

- imq.persist.file.message. filepool.cleanratio。クリーン状態 (サイズを 0 にする) で保持される送信先のファイルプールの空きファイルの割合を指定します。
- imq.persist.file.message.cleanup。ブローカがシャットダウンされたとき、送信先のファイルプールの空きファイルを消去するかどうかを指定します。

上記のプロパティの詳細は、323 ページの表 14-6 を参照してください。

次のプロパティは、JDBC ベースの持続に関連したものです。

- imq.persist.jdbc.brokerid。複数のブローカインスタンスで使用される、データベース内のテーブル名に追加するブローカインスタンス識別子を指定します。
- imq.persist.jdbc.driver。データベースに接続する JDBC ドライバの Java クラス名を指定します。
- imq.persist.jdbc.opendburl。既存データベースへのコネクションを開くためのデータベース URL を指定します。
- imq.persist.jdbc.createdburl。データベースを作成するコネクションを開くためのデータベース URL を指定します。
- imq.persist.jdbc.closedburl。ブローカをシャットダウンする場合に、現在のデータベースコネクションをシャットダウンするためのデータベース URL を指定します。
- imq.persist.jdbc.user。必要に応じて、データベースコネクションを開くときに使用するユーザー名を指定します。
- imq.persist.jdbc.needpassword。データベースでブローカのアクセスにパスワードを必要とするかどうかを指定します。
- imq.persist.jdbc.password。データベースコネクションを開くときに使用するパスワードを、必要に応じて指定します。
- imq.persist.jdbc.table.IMQSV35。バージョンテーブルを作成するための SQL コマンドです。
- 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 コマンドです。
- imq.persist.jdbc.table.IMQDEST35。送信先テーブルを作成するための SQL コマンドです。
- imq.persist.jdbc.table.IMQINT35。配信対象テーブルを作成するための SQL コマンドです。
- imq.persist.jdbc.table.IMQMSG35。メッセージテーブルを作成するための SQL コマンドです。
- imq.persist.jdbc.table.IMQPROPS35。プロパティテーブルを作成するための SQL コマンドです。

- `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 コマンドです。
- `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 コマンドです。
- `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 コマンドです。

上記のプロパティの詳細は、324 ページの表 14-7 を参照してください。

セキュリティマネージャ

Message Queue では、認証および承認 (アクセス制御) 機能が用意されており、暗号化機能もサポートされています。

認証機能と承認機能はユーザーリポジトリによって異なります (89 ページの図 4-4 を参照)。ユーザーリポジトリには、ファイル、ディレクトリ、または名前、パスワード、グループメンバーシップなどのメッセージングシステムのユーザーに関する情報を含むデータベースがあります。ユーザー名とパスワードはブローカへのコネクションが要求されたときに、ユーザーを認証するために使用されます。ユーザー名とグループのメンバーシップは、送信先のプロデューシングメッセージ、またはコンシューミングメッセージなどの操作を承認するために、アクセス制御ファイルと一緒に使用されます。

Message Queue の管理者は、Message Queue で提供されるユーザーリポジトリ (144 ページの「単層型ファイルユーザーリポジトリを使用する」を参照) を生成するか、あるいは既存の LDAP ユーザーリポジトリをセキュリティマネージャのコンポーネントに組み込みます (150 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)。

認証

Message Queue のセキュリティは、パスワードベースの認証がサポートしています。クライアントがブローカへのコネクションを要求する場合、ユーザー名とパスワードを提示する必要があります。

セキュリティマネージャは、クライアントから提示されたユーザー名とパスワードをユーザーリポジトリ内に格納されているものと比較します。クライアントからブローカにパスワードが送信される場合、パスワードは、Base-64 か、メッセージダイジェスト (MD) のどちらかを使用して暗号化されます。セキュリティ保護された送信については、89 ページの「暗号化」を参照してください。各コネクションサービスで使用する暗号化のタイプを 1 つずつ設定するか、あるいはブローカ単位で暗号化を設定することができます。

セキュリティマネージャのすべてのプロパティは、90 ページの「セキュリティマネージャのプロパティ」に一覧表示しています。また 90 ページの「セキュリティマネージャのプロパティ」で詳細を説明しています。

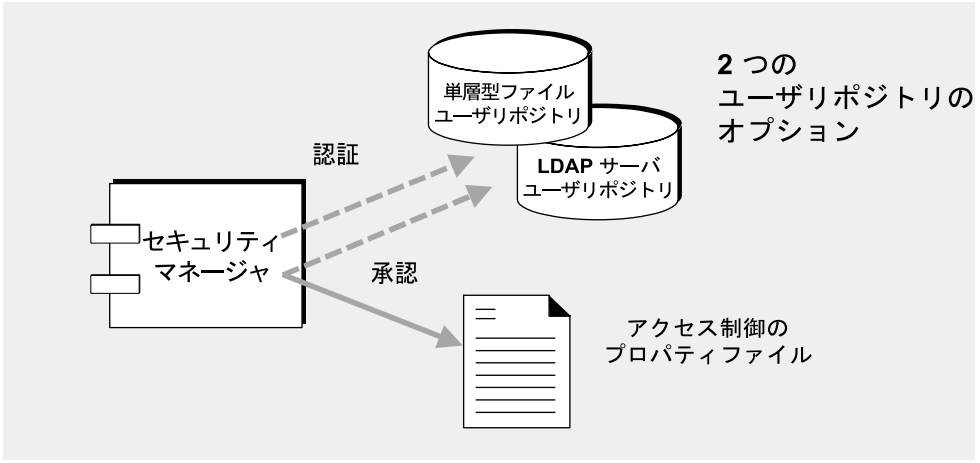
承認

クライアントアプリケーションのユーザーが認証されると、ユーザーはさまざまな 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。セキュリティマネージャは、ユーザーベースとグループベースの両方のアクセス制御をサポートしています。ユーザー名、またはユーザーリポジトリでユーザーに割り当てられているグループに応じて、ユーザーは特定の Message Queue 操作を実行するためのアクセス権を付与されます。これらのアクセス制御は、アクセス制御プロパティファイルで指定します (図 4-4 を参照)。

ユーザーがある操作を実行しようとすると、セキュリティマネージャが、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ファイルでは、次の操作に対するアクセス権を指定します。

- ブローカとのコネクションの確立
- 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

図 4-4　セキュリティマネージャのサポート



デフォルトのアクセス制御プロパティファイルは、*admin* という 1 つのグループだけを明示的に参照します (147 ページの「グループ」を参照)。*admin* グループのユーザーには、admin サービスコネクションのアクセス権が付与されます。admin サービスは、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 サービスコネクションにアクセスできません。

Message Queue の管理者は、グループを定義し、ユーザーリポジトリ内のそれらのグループとユーザーを関連付けることができます。ただし、グループは単層型ファイルのユーザーリポジトリで完全にはサポートされません。

アクセス制御プロパティファイルを編集してユーザー別およびグループ別に目的 (メッセージの生成と消費、またはキューの送信先のメッセージの参照) に応じて、送信先へのアクセスを指定できます。特定のユーザー、またはグループだけがアクセスできる個別の送信先、またはすべての送信先を作成できます。ブローカに送信先の自動作成が設定される場合、アクセス制御プロパティファイルを編集して、ブローカが送信先を自動作成するユーザーとグループを制御することができます。

セキュリティマネージャのすべてのプロパティは、90 ページの「セキュリティマネージャのプロパティ」に一覧表示しています。また 90 ページの「セキュリティマネージャのプロパティ」で詳細を説明しています。

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) 標準に基づいたコネクション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コネクションを確立して、コネクションレベルのセキュリティを提供します。

Message Queue の SSL ベースのコネクションサービスを使用するには、キーツールユーティリティ (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。Message Queue のキーストア自体は、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。160 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。

キーストアのロックが解除されると、ブローカは、コネクション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コネクションを設定します。

セキュリティマネージャのすべてのプロパティは、次の節に一覧表示しています。また 90 ページの「セキュリティマネージャのプロパティ」で詳細を説明しています。

セキュリティマネージャのプロパティ

認証、承認、暗号化、およびその他のセキュリティ保護された通信に関する設定可能なプロパティを以下に示します。

- `imq.authentication.type`。パスワードを Base-64 コーディング (`basic`)、または MD5 ダイジェスト (`digest`) のどちらで送信するのかを指定します。
- `imq.`*service_name*`.authentication.type`。パスワードを Base-64 コーディング (`basic`)、または MD5 ダイジェスト (`digest`) のどちらで送信するのかを指定します。
- `imq.authentication.basic.user_repository`。Base-64 コーディングの場合、認証に使用するユーザーリポジトリの種類 (ファイルベースか LDAP か) を指定します。
- `imq.authentication.client.response.timeout`。ブローカからの認証要求に対するクライアントの応答をシステムが待機する時間を秒単位で指定します。
- `imq.accesscontrol.enabled`。アクセス制御プロパティファイルに指定されているように、認証されたユーザーが、コネクションサービスを使用するためのアクセス権、あるいは特定の送信先に対して特定の Message Queue 操作を実行するためのアクセス権を保持していることをシステムでチェックするかどうかを指定します。
- `imq.`*service_name*`.accesscontrol.enabled`。指定したコネクションサービスに対して、アクセス制御を設定し (`true`/`false`)、ブローカ全体の設定をオーバーライドします。
- `imq.accesscontrol.file.filename`。ブローカインスタンスでサポートされるすべてのコネクションサービスに対して、アクセス制御プロパティファイルの名前を指定します。

- `imq.`*service_name*`.accesscontrol.file.filename`。ブローカインスタンスの指定したコネクションサービスに対して、アクセス制御プロパティファイルの名前を指定します。
- `imq.passfile.enabled`。セキュリティ保護される通信用の (SSL、LDAP、JDBC™ の) ユーザーパスワードをファイルで指定するどうかを指定します。
- `imq.passfile.dirpath`。パスファイルを含むディレクトリのパスを指定します。
- `imq.passfile.name`。パスファイル名を指定します。
- `imq.keystore.`*property_name*。SSL ベースのサービスの場合は、SSL キーストアに関係するセキュリティプロパティを指定します。332 ページの表 14-9 を参照してください。

上記のプロパティの詳細は、328 ページの表 14-8 を参照してください。

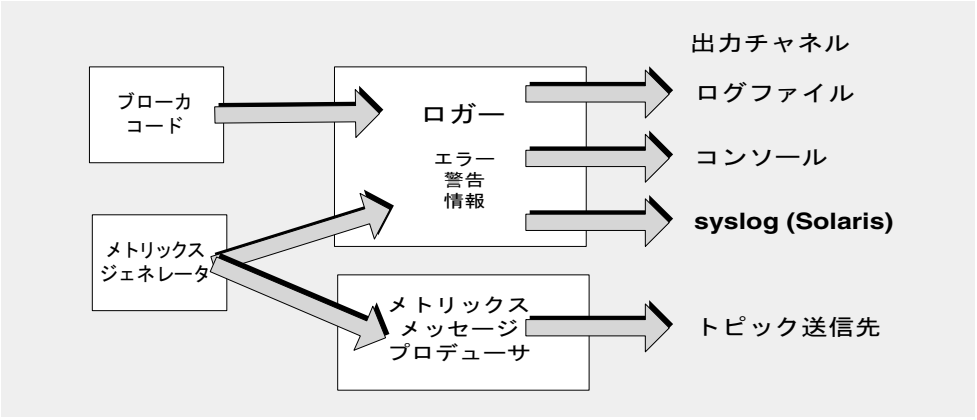
監視サービス

ブローカには、ブローカの動作を監視および診断するためのさまざまなコンポーネントが用意されています。たとえば、次のようなコンポーネントが含まれています。

- データを生成するコンポーネント (イベントを記録するブローカコードとメトリックスジェネレータ)
- 多数の出力チャネルを使用して情報を書き込むロガーコンポーネント (「ロガー」を参照)
- 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ッセージプロデューサ。

この仕組みの概略を、図 4-5 に示します。

図 4-5　監視サービスのサポート



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、開かれているコネクションの数、使用中のスレッドの数など、ブローカの動作に関する情報を提供します。

メトリックスデータの生成をオン、またはオフにすることも、メトリックスレポートを生成する頻度を指定することもできます。

ロガー

Message Queue のロガーは、ブローカコードとメトリックスジェネレータによって生成された情報を取得し、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、syslog デーモンプロセスなどの多数の出力チャネルに書き込みます。

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。

たとえば、ロガーレベルを指定するとロガーが収集する情報の種類、エラー (ERROR)、エラーと警告 (WARNING)、またはエラー、警告、および情報 (INFO) を指定できます。

各出力チャネルに対して、そのチャネルに書き込まれるロガーのカテゴリを設定できます。たとえば、ロガーのレベルが INFO の場合、エラーと警告だけをコンソールに出力し、情報 (メトリックスデータ) だけをログファイルに書き込むように指定することができます。

ログファイルを使用する場合、ログファイルを閉じて新しいファイルに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。新しいロールオーバーログファイルが作成されると、もっとも新しい 9 個のログファイルのアーカイブが保持されます。

ロガーの設定方法については、205 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。Solaris syslog の設定および使用方法については、syslog(1M)、syslog.conf(4)、および syslog(3C) のマニュアルページ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)

メッセージプロデューサのコンポーネントは、メトリックスジェネレータのコンポーネントから定期的に情報を受け取ります。このコンポーネントはメッセージに情報を書き込み、その後メトリックストピック送信先に送信します。メトリックスメッセージを送信する送信先は、メッセージに含まれる情報の種類により異なります。

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 4-3 に示します。

表 4-3　メトリックスのトピック送信先

トピック送信先名	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

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した Message Queue クライアントは、送信先内でメッセージを消費し、メトリックス情報を処理します。たとえば、クライアントは、`mq.metrics.broker` 送信先にサブスクライブし、ブローカ内のメッセージの合計数といった情報を受け取り処理することができます。

メトリックスメッセージプロデューサは、メトリックスデータに相当する、名前と値のペアを含むメッセージを作成する内部 Message Queue クライアントです。メッセージのタイプは `MapMessage` です。これらのメッセージは、該当するメトリックストピック送信先へのサブスクライバが複数存在する場合にだけ生成されます。

メトリックスメッセージプロデューサにより生成されるメッセージのタイプは、`MapMessage` です。メッセージは、メッセージに含まれるメトリックスのタイプに応じて、複数の名前 / 値のペアから構成されます。各名前と値のペアは、メトリックスの数とその値に相当します。

例として、ブローカメトリックスメッセージは、ブローカとの間で送受信されたメッセージの数、これらのメッセージのサイズ、現在メモリー内にあるメッセージの数とサイズなどに関する値を含んでいます。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。このマニュアルでは、メトリックスメッセージの消費に関して Message Queue クライアントを書き込む方法を説明しています。

メトリックスメッセージの本体に含まれるメトリックス情報以外に、各メッセージのヘッダーには次の情報を提供するプロパティがあります。

- メッセージタイプ
- メッセージを送信したブローカのホスト、ポート、アドレス
- 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は、異なる種類または異なるブローカからのメトリックメッセージを処理する Message Queue クライアントアプリケーションに有用です。

監視サービスのプロパティ

次に示すのは、ブローカによる情報の生成、ロギング、およびメトリックスメッセージの生成を設定する設定可能なプロパティです。

- imq.metrics.enabled。メトリックス情報をロガーへ書き込むかどうかを指定します。
- imq.metrics.interval。メトリックスのロギングが有効な場合は、メトリックス情報がロガーへ書き込まれる時間間隔を秒単位で指定します。
- imq.log.level。ロガーレベル、つまり、出力チャネルへ書き込み可能な出力のカテゴリを指定します。
- imq.log.file.output。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。
- imq.log.file.dirpath。ログファイルを含むディレクトリのパスを指定します。
- imq.log.file.filename。ログファイル名を指定します。
- imq.log.file.rolloverbytes。新しいログファイルに出力がロールオーバーされるログファイルのサイズをバイト単位で指定します。
- imq.log.file.rolloversecs。新しいログファイルに出力がロールオーバーされるログファイルの有効期間を秒単位で指定します。
- imq.log.console.output。コンソールへ書き込むロギング情報のカテゴリを指定します。
- imq.log.console.stream。コンソールの出力を標準出力 (OUT)、または標準エラー出力 (ERR) のどちらに書き込むかを指定します。
- imq.log.syslog.facility。(Solaris のみ) Message Queue ブローカが記録する syslog 機能を指定します。
- imq.log.syslog.logpid。(Solaris のみ) メッセージとともにブローカのプロセス ID を記録するかどうかを指定します。
- imq.log.syslog.logconsole。(Solaris のみ) メッセージを syslog に送信できない場合に、システムコンソールにメッセージを書き込むかどうかを指定します。
- imq.log.syslog.identity。(Solaris のみ) syslog に記録される各メッセージの先頭に付加する識別情報文字列を指定します。
- imq.log.syslog.output。(Solaris のみ) syslogd(1M) に書き込むロギング情報のカテゴリを指定します。
- imq.log.timezone。ログのタイムスタンプのタイムゾーンを指定します。

- imq.metrics.topic.enabled。メトリックスメッセージの生成を有効にするかどうかを指定します。
- imq.metrics.topic.interval。メトリックスメッセージを生成する間隔を秒で指定します。
- imq.metrics.topic.persist。メトリックスメッセージが持続メッセージかどうかを指定します。
- imq.metrics.topic.timetolive。メトリックストピック送信先へ送信されるメトリックスメッセージの有効期間を秒単位で指定します。
- imq.destination.logDeadMsgs。ブローカがデッドメッセージを破棄するたびに、またはデッドメッセージキューにデッドメッセージを配置するたびに、ログにメッセージを書き込むかどうかを指定します。

これらのプロパティの詳細は、333 ページの表 14-10 を参照してください。

設定ファイルの概要

ブローカ設定ファイルは、ブローカの設定に使用します。付録 A「オペレーティングシステムごとの Message Queue データの場所」に、オペレーティングシステム別にファイルが置かれたディレクトリを一覧表示しています。

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**。このファイルは、default.properties と呼ばれ、編集はできません。デフォルトの設定を決定したり、変更するプロパティ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を格納する**インストール設定ファイル**。このファイルは、install.properties と呼ばれ、インストール後は編集できません。

インスタンス設定ファイル

最初にブローカを実行したときに、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、ブローカのそのインスタンスに設定プロパティを指定する場合に使用します。

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「オペレーティングシステムごとの Message Queue データの場所」を参照してください。

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ます。このファイルは、管理ツールを使用して設定に変更が加えられ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、設定を変更できます (98 ページの「インスタンス設定ファイルの編集」を参照)。手作業で変更するには、.../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権限を変更する必要があります。

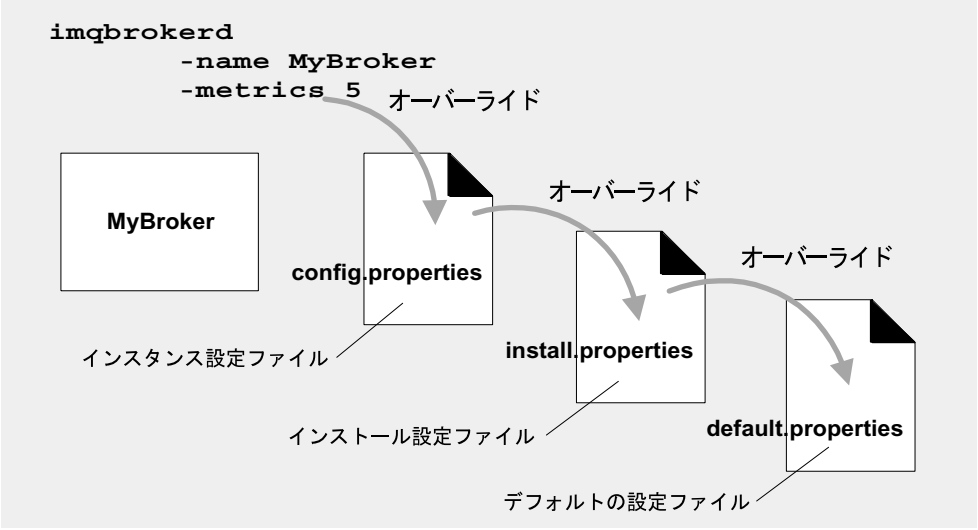
クラスタでブローカインスタンスを接続する場合、クラスタ設定ファイルを使用して、クラスタ設定情報を指定する必要があります。詳細は、336 ページの「クラスタ設定プロパティ」を参照してください。

プロパティ値のマージ

起動時に、ブローカは異なる設定ファイルのプロパティ値をマージします。インストール時の値、およびインスタンス設定ファイルに設定された値が使用され、デフォルトの設定ファイルで指定された値はオーバーライドされます。

imqbrokerd コマンドオプションを使用すると、生成された値をオーバーライドできます。この方式を図 4-6 で図解します。

図 4-6　ブローカ設定ファイル



プロパティ命名構文

設定ファイルにある Message Queue プロパティの定義では、次の命名構文を使用します。

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 86400 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
```

311 ページの第 14 章「ブローカのプロパティのリファレンス」に、ブローカ設定プロパティとそのデフォルト値を一覧表示しています。

インスタンス設定ファイルの編集

初めてブローカインスタンスが実行されると、config.properties ファイルが自動的に作成されます。このインスタンス設定ファイルを編集して、対応するブローカインスタンスの動作やリソースの使用をカスタマイズできます。

ブローカインスタンスが config.properties ファイルを読み込むのは起動時だけです。config.properties ファイルへの変更を確定するには、次の操作のどちらかを行います。

- 管理ツールを使用する。imqcmd を使用して設定できるプロパティについての詳細は、312 ページの表 14-1 を参照してください。
- ブローカインスタンスをシャットダウンしてから config.properties ファイルを編集し、その後インスタンスを再起動します (Solaris および Linux オペレーティングシステムでは、最初にブローカインスタンスを起動したユーザーだけが config.properties ファイルを編集するためのアクセス権を持つ)。

表 14-1 に、ブローカインスタンス設定プロパティとそのデフォルト値をアルファベット順に一覧表示します。各プロパティの意味と使用に関する詳細は、指定された相互参照の節で確認してください。

コマンド行への設定オプションの入力

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

起動時に imqbrokerd コマンド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とその値を指定できます。imqsvcadmin コマンドを使用して、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を指定します。

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を設定できます。実行中のブローカの設定を変更する場合は、imqcmd update bkr コマンドを使用します。

起動時の設定の詳細は、第 3 章「ブローカとクライアントの起動」、特に 67 ページの「ブローカのインタラクティブな起動」の例を参照してください。

実行中のブローカの設定変更の詳細は、第 5 章「ブローカの管理」と第 14 章「ブローカのプロパティのリファレンス」を参照してください。

持続ストアの設定

Message Queue のブローカには、一貫した情報の書き込みおよび取得を管理する持続マネージャのコンポーネントが含まれています。デフォルトでは、持続マネージャは、組み込みのファイルベースのデータストアにアクセスするように設定されていますが、持続マネージャを再設定して、JDBC 互換ドライバを介したアクセスが可能な任意のデータストアに接続できます。

Message Queue データストアには、トランザクション、メッセージ、永続サブスクリプション、物理的送信先に関する情報が収められています。また、メッセージの通知に関する状態の詳細も収められています。

この章では、持続ストアを使用するブローカの設定方法を説明します。次のトピックが含まれます。

- 99 ページの「ファイルシステムストアの設定」
- 100 ページの「JDBC ストアの設定」
- 104 ページの「持続データの保護」

ファイルシステムストアの設定

ファイルシステムデータストアは、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内に置かれます。オペレーティングシステムにより場所が異なります。持続ストアの正確な場所は、付録 A「オペレーティングシステムごとの 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 はディスクへの書き込み操作を非同期的に実行します。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、メッセージは損なわれる可能性があります。信頼性を高めるために、Message Queue がディスクへの書き込みを同時に実行するように設定できますが、このオプションによりパフォーマンスが低下することに注意してください。ディスクへの同時書き込みを指定する場合、ブローカプロパティ `imq.persist.file.sync.` を設定します。このプロパティの詳細は、323 ページの表 14-6 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 -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、284 ページの表 13-2 を参照してください。

JDBC ストアの設定

JDBC ベースの持続を使用する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ユーティリティ (`imqdbmgr`) は、JDBC ドライバとブローカ設定プロパティを使用してデータベースを作成し、管理します。

この章に示す手順は、例として Java 2 プラットフォーム Enterprise Edition (J2EE) SDK にバンドルされる PointBase DBMS を使用して説明しています。バージョン 1.4 は、`java.sun.com` からダウンロードして入手できます。例では、クライアント / サーバーバージョンの代わりに、PointBase の組み込みバージョンを使用します。手順の指示は、PointBase の例のパス名とプロパティ名を使用しています。これらは、「例 :」という言葉で識別されます。

Oracle と PointBase の設定例を参照できます。例を収めたファイルは、付録 A 「オペレーティングシステムごとの Message Queue データの場所」を参照してください。オペレーティングシステム別の情報を示す表で、「アプリケーションと設定の例」の場所を探してください。

また、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 アクセスが可能なデータストアへの接続

JDBC アクセスが可能なデータストアに接続するには、次の手順を実行するだけです。

➤ JDBC アクセスが可能なデータストアに接続する

1. ブローカの設定ファイルに、JDBC 関連のプロパティを設定します。

   324 ページの「JDBC ベースの持続」に示すプロパティを参照してください。

2. 次のパスに、JDBC ドライバの jar ファイルの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)

   コピーの例 (Solaris):

   ```
   % cp j2eeSDK_install_directory/pointbase/lib/pointbase.jar
   /usr/share/lib/imq/ext
   ```

   シンボリックリンクの例 (Solaris):

```
% ln -s j2eeSDK_install_directory/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。103 ページの「データベース管理ユーティリティ (imqdbmgr)」を参照してください。

   例:

   a. `imqdbmgr` がある場所にディレクトリを移動します。

      `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/`*dabatabseName*。

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、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 (103 ページの「データベース管理ユーティリティ (imqdbmgr)」を参照)。

JDBC 関連のブローカのプロパティ

ブローカの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「オペレーティングシステムごとの Message Queue データの場所」を参照)。

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

ファイルが存在しない場合、Message Queue がファイルを作成できるように、`-name` *instanceName* オプションを使用して、ブローカを起動する必要があります。

324 ページの「JDBC ベースの持続」に、JDBC アクセスが可能なデータストアに接続する場合に、設定が必要な設定プロパティを示します。この節の最後に、これらのプロパティをまとめています。プラグイン持続を使用する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`) に、これらのプロパティを設定します。

インスタンス設定プロパティを使用すると、Message Queue 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できます。各データベーステーブルを作成するための SQL コードを指定する設定可能なプロパティがあります。これらのプロパティは、接続されたデータベースが使用するデータタイプを適切に指定するために必要となります。

正確な SQL 構文に関してはデータベースベンダー間で互換性がないため、必ず使用中のデータベースのベンダーが提供している相応するマニュアルを確認し、それに従って 324 ページの表 14-7 のプロパティを調整してください。たとえば、PointBase データベースの場合は、IMQMSG35 テーブルの MSG 列で許容される最大の長さを調整する必要が生じる場合があります (`imq.persist.jdbc.table.IMQMSG35` プロパティを参照)。

すべてのブローカ設定プロパティと同様に、値は `-D` コマンド行オプションを使用して設定できます。データベースで特定のデータベース固有プロパティを設定する必要がある場合、ブローカ (`imqbrokerd`) の起動時に、`-D` コマンド行オプションを使用するか、あるいは Database Manager ユーティリティ (`imqdbmgr`) を使用して設定することができます。

例 :

PointBase の組み込みデータベース例の場合、データベースコネクション URL に、データベースの絶対パスを指定する代わりに、`-D` コマンド行オプションを使用して、PointBase システムディレクトリを定義することができます。

```
-Ddatabase.home=IMQ_VARHOME/instances/instanceName/dbstore
```

その場合、次のように URL を指定してデータベースを作成できます。

```
imq.persist.jdbc.createdburl=jdbc:pointbase:embedded:dbName;new
```

次のように URL を指定してデータベースを開きます。

```
imq.persist.jdbc.opendburl=jdbc:pointbase:embedded:dbName
```

次に示すのは JDBC 関連のプロパティの要約です。

- `imq.persist.store`。ファイルベースまたは JDBC ベースのデータストアを指定します。
- `imq.persist.jdbc.brokerid`。固有の名前にするために、データベーステーブル名に追加されるブローカインスタンス識別子を指定します。
- `imq.persist.jdbc.driver`。データベースに接続する JDBC ドライバの Java クラス名を指定します。
- `imq.persist.jdbc.opendburl`。既存データベースへのコネクションを開くためのデータベース URL を指定します。
- `imq.persist.jdbc.createdburl`。データベースを作成するコネクションを開くためのデータベース URL を指定します。

- imq.persist.jdbc.closedburl。ブローカをシャットダウンする場合に、現在のデータベースコネクションをシャットダウンするためのデータベース URL を指定します。
- imq.persist.jdbc.user。必要に応じて、データベースコネクションを開くときに使用するユーザー名を指定します。
- imq.persist.jdbc.needpassword。データベースでブローカのアクセスにパスワードを必要とするかどうかを指定します。
- imq.persist.jdbc.password。データベースコネクションを開く際に使用するパスワードを、必要に応じて指定します。
- imq.persist.jdbc.table.IMQSV35。バージョンテーブルを作成するための SQL コマンドです。
- 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 コマンドです。
- imq.persist.jdbc.table.IMQDEST35。送信先テーブルを作成するための SQL コマンドです。
- imq.persist.jdbc.table.IMQINT35。配信対象テーブルを作成するための SQL コマンドです。
- imq.persist.jdbc.table.IMQMSG35。メッセージテーブルを作成するための SQL コマンドです。
- imq.persist.jdbc.table.IMQPROPS35。プロパティテーブルを作成するための SQL コマンドです。
- 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 コマンドです。
- 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 コマンドです。
- 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 コマンドです。

これらのプロパティの詳細は、第 14 章「ブローカのプロパティのリファレンス」を参照してください。

データベース管理ユーティリティ (imqdbmgr)

Message Queue には、持続に必要なスキーマをセットアップするためのデータベース管理ユーティリティ (imqdbmgr) が用意されています。テーブルが破損した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、このユーティリティを使用して、Message Queue のデータベーステーブルを削除することもできます。

imqdbmgr コマンドの構文、サブコマンド、オプションの詳細は、第 13 章「コマンドのリファレンス」を参照してください。

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、組み込みファイルストアまたは JDBC ストアでデータを保護する方法を説明します。

組み込み (ファイルベース) 持続ストア

組み込み持続を使用するブローカは、オペレーティングシステ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「オペレーティングシステムごとの 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/filestore/ 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によって異なります。

Solaris および Linux: IMQ_VARHOME/instances/*instanceName*/filestore/ ディレクトリのアクセス権は、そのブローカインスタンスを開始したユーザーの umask によって決まります。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、umask 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、IMQ_VARHOME/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: IMQ_VARHOME/instances/*instanceName*/filestore/ ディレクトリのアクセス権は、使用中の 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。通常は、そのディレクトリ用のプロパティダイアログが開かれます。

プラグイン (JDBC) 持続ストア

プラグインの持続性を使用するブローカは、JDBC に準拠したデータベースに持続データを書き込みます。

Oracle データベースなど、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。このようなデータベーステーブルには、「IMQ」で始まる名前が付いています。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のマニュアルを参照してください。

データベースコネクション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として与えることができます。ただし、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 (『Message Queue 管理ガイド』の付録 A 「プラグイン持続の設定」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、「組み込み (ファイルベース) 持続ストア」で説明するように、通常は持続データが格納されるディレクトリへのファイルアクセス権を設定することでセキュリティがもたらされます。ただし、データベースがブローカと `imqdbmgr` ユーティリティの両方で読み取り可能および書き込み可能になるために、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、ブローカとブローカのサービスの管理に関する基本的なタスクを実行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。主なトピックは、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 コマンドと imqusermgr コマンドを使用します。ブローカを管理する前に、次の作業が必要です。

- imqbrokerd 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 ユーティリティを実行できます。

  運用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コネクションを使用する場合、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、160 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 コマンドユーティリティの使用

imqcmd コマンドユーティリティを使用すると、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、281 ページの第 13 章「コマンドのリファレンス」を参照してください。物理的送信先の管理の詳細は、339 ページの第 15 章「物理的送信先のプロパティのリファレンス」で個別に扱っています。

ユーザー名とパスワードを指定する

それぞれの imqcmd 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。次のような例外があります。

- -h オプションまたは -H オプションを使用してヘルプコマンドを表示するコマンド。
- -v オプションを使用して製品のバージョンを表示するコマンド。

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。

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 passfile の名前を指定します。
- コマンドからパスワードの入力を要求する。ほかのユーザーが入力内容を見ることができない限り、パスワードの指定にはこの方法がもっとも安全です。

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。このオプションは推奨されないため、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、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、myserver のポート 1564 で実行中のブローカのプロパティを一覧表示しています。ユーザー名は aladdin です。このコマンドは、ユーザー名 aladdin が admin グループに割り当てられるようにユーザーリポジトリの更新を要求します。

```
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コネクションを確立するために、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。

ヘルプの表示

imqcmd コマンドユーティリティ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、コマンドユーティリティは -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、コマンドユーティリティは -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。

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:port
```

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の更新

次のブローカのプロパティを更新する場合は、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:port]-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。

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

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
```

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、ブローカのコネクションサービススレッドが中断されるため、ブローカはコネクション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、新しいコネクションの受け入れ、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管理コネクション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。たとえば、特定の物理的送信先にメッセージが集中した場合には、ブローカを停止し、問題の修復に役立つ次のいずれかを実行できます。

- メッセージのソースをトレースする
- 物理的送信先のサイズを制限する
- 物理的送信先を廃棄する

ブローカを停止しても、cluster コネクション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。

次に示すのは、pause bkr サブコマンドの構文です。

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

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
```

個々のコネクションサービス、および個々の物理的送信先も停止できます。詳細は、120 ページの「コネクションサービスの停止および再開」と 134 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:port]
```

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。ブローカは新しいコネクション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:port]
```

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
```

ブローカをシャットダウンし再起動できます。次に示すのは、`restart bkr` サブコマンドの構文です。

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

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。

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起動されます。

```
imqcmd restart bkr -u admin
```

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:port]
        [-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する (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する。
- **cxn**　コネクション、仮想メモリーヒープ、およびスレッドを表示する。

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 (無限) です。

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
```

imqcmd を使用してブローカのメトリックスをレポートする方法の詳細は、364 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。

コネクションサービスの管理

コマンドユーティリティには、次のコネクション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。

- コネクションサービスの一覧表示
- コネクションサービス情報の表示
- コネクションサービスのプロパティの更新
- コネクションサービスのメトリックスの表示
- コネクション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コネクションサービスを、表 5-1 に示します。「サービス名」列の値は、-n オプションでサービス名を指定するのに使用する値になります。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コネクションサービス

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)
httpjms (Enterprise Edition)	NORMAL	http
httpsjms (Enterprise Edition)	NORMAL	https (SSL ベースセキュリティ)

表 5-1　ブローカがサポートするコネクションサービス (続き)

サービス名	サービスタイプ	プロトコルタイプ
admin	ADMIN	tcp
ssladmin (Enterprise Edition)	ADMIN	tls (SSL ベースセキュリティ)

コネクションサービスの一覧表示

ブローカで使用できるコネクション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:port]
```

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コネクションサービスを一覧表示します。

サブコマンドは、コマンド行で次のように使用します。

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

たとえば、次のコマンドでは、`myServer` ホストのポート `6565` で実行しているブローカで使用可能なサービスが一覧表示されます。

```
imqcmd list svc -b MyServer:6565 -u admin
```

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
```

コネクション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:port]
```

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
```

コネクションサービスのプロパティの更新

表 5-2 に示す 1 つ以上のサービスのプロパティ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コネクションサービスプロパティ

プロパティ	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:port]
          -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、316 ページの「コネクションサービスのプロパティ」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
```

コネクション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:port] [-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**　指定したコネクション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する (デフォルトのメトリックスタイプ)。
- **rts**　指定したコネクション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 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する。
- **cxn**　コネクション、仮想メモリーヒープ、およびスレッドを表示する。

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 コネクション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 を使用してコネクションサービスのメトリックスをレポートする方法の詳細は、366 ページの「コネクションサービスのメトリックス」を参照してください。

コネクション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:port]
```

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。admin サービスは停止できません。

次のようなコマンド行を使用します。

```
imqcmd pause svc -n serviceName -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コネクションの受け入れをやめる。Message Queue クライアントが新しいコネクションを開こうとすると、例外が発生する。
- 停止したサービスの既存のコネクションはすべて維持されるが、ブローカはサービスが再開されるまでこれらのコネクションのすべてのメッセージ処理を中断する。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する。
- 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る。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る。

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:port]
```

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。

次のようなコマンド行を使用します。

```
imqcmd resume svc -n serviceName -u admin
```

コネクション情報の入手

コマンドユーティリティには、コネクション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コネクションを一覧表示します。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:port]
```

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コネクションをすべて一覧表示します。サービス名を指定しない場合は、すべてのコネクションが一覧表示されます。

たとえば、次のように指定します。

```
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.
```

シングルコネクション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:port]
```

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コネクション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

ブローカの永続サブスクリプションを管理するには、imqcmd のサブコマンドを使用する必要があります。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パージ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。

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 client_id
```

サブスクリプションを識別するには、サブスクリプションの名前とクライアント ID を使用します。たとえば、次のように指定します。

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

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パージします。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 client_id
```

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を false に設定した場合 (319 ページの表 14-3 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、コネクション識別子、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 transaction_id
```

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
```

これはコマンドにより生成された出力です。

```
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 transaction_id
```

たとえば、次のように指定します。

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

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。

次に示すのは、rollback サブコマンドの構文です。

```
imqcmd rollback txn -n transaction_id
```

詳細は、319 ページの表 14-3 の imq.transaction.autorollback プロパティを参照してください。

第 6 章

物理的送信先の管理

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、次の作業の方法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。

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。

imqcmd コマンドユーティリティの使用

imqcmd コマンドユーティリティを使用すると、物理的送信先を管理できます。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になります。

imqcmd とそのサブコマンド、オプションについての詳細は、281 ページの第 13 章「コマンドのリファレンス」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、293 ページの「物理的送信先管理サブコマンド」を参照してください。

表 6-1　imqcmd コマンドユーティリティ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組み込みの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パージします。
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。
update dst	送信先のプロパティを更新します。

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、プロパティも設定できます。

物理的送信先の多くのプロパティが、ブローカのメモリーリソースおよびメッセージフローを管理します。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が制御するブローカ全体の制限に似ています。

次のプロパティは、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する。
- 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する。
- 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する。
- 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する。
- maxNumProducers。物理的送信先のプロデューサの最大数を指定する。
- 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する。
- isLocalOnly。ブロークラスタに対してのみ適用。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する。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) だけに制限する。

- 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する。

次のプロパティは、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する。
- 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する。
- 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。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する。

物理的送信先のプロパティについての詳細は、339 ページの第 15 章「物理的送信先のプロパティのリファレンス」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値を設定します。自動作成されるプロパティの詳細を、320 ページの表 14-4 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、ブローカのすべての物理的送信先を一覧表示します。これには、`list dst` サブコマンドを使用します。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。

送信先のタイプが指定されない場合は、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、`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値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、次のように指定します。

```
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の値を変更できます (133 ページの「物理的送信先のプロパティの更新」を参照)。

物理的送信先のプロパティの更新

物理的送信先のプロパティを変更するには、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値を更新します。プロパティ名は、表 15-1 で説明しているいずれかのプロパティになります。

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 プロパティが 1000 に、MaxNumMsgs プロパティが 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については、第 15 章「物理的送信先のプロパティのリファレンス」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、プロパティがその他の送信先のプロパティと多少異なります。詳細は、138 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。

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。

物理的送信先のパージ

物理的送信先のキューに現在入っているメッセージは、すべてパージすることが可能です。物理的送信先をパージすると、送信先のキューに入っ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パージ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パージするには、`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パージします。

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、古いメッセージをパージします。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、送信先のパージ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パージ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パージされ、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、プラグインされた JDBC 互換のデータストアではなく、組み込みの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組み込みの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 (339 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 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

imqcmd コマンドユーティリティは、デッドメッセージキューを管理します。デッドメッセージキューは、ほかのキューと同じように管理しますが、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。

デッドメッセージキューのプロパティ

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のデッドメッセージキューの処理

プロパティ	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。
maxNumProducers	デッドメッセージキューに適用されません。

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

プロパティ	デッドメッセージキューによる固有の処理
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。

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、本体の内容を破棄することを検討してください。

本体の内容を破棄し、ヘッダーとプロパティデータのみを残す場合は、次の例に示すように、imq.destination.DMQ.truncateBody ブローカプロパティを true に設定します。

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

デッドメッセージのロギングを有効にする

標準のキューの監視とロギングオプションに加えて、ブローカがデッドメッセージと分類したメッセージのロギングを実行できます。

デッドメッセージのロギングを有効にした場合、ブローカは次のタイプのイベントのロギングを実行します。

- ブローカがメッセージをデッドメッセージキューに移動する。
- 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。

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。次の例では、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

セキュリティの管理

管理者は、ユーザーの認証に使用する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

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。

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、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の名前およびパスワードと一致した場合に、コネクションを許可します。

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ユーティリティ (`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、150 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 Message Queue ユーザーマネージャ (`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 Message Queue ユーザーマネージャユーティリティ (`imqusermgr`) 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「オペレーティングシステムごとの Message Queue データの場所」を参照)。

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

表 7-1 に示すように、リポジトリは 2 つのエントリ (行) から構成されます。

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。Message Queue ブローカを使用するために、ユーザーやパスワードの初期設定は必要ありません。

たとえばテストの目的で、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ユーザー名とパスワード guest 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 (149 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ユーティリティ (imqusermgr)

ユーザーマネージャユーティリティ (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「コマンドのリファレンス」を参照してください。

imqusermgr 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 : 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：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：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：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 *active_state*	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ユーザーリポジトリを指定します。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。

- **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、153 ページの「ユーザーの承認 : アクセス制御プロパティファイル」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、新規コネクション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。この変更を行うには、imqusermgr ツールを使用します。

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の値を設定します。これらのプロパティを使用すると、ブローカインスタンスが LDAP サーバーに対してユーザーおよび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。ブローカは LDAP サーバーに対して、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、特定のメッセージング操作を実行しようとしているかクエリーします。

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「オペレーティングシステムごとの 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 暗号化方式 (`basic`) を使用するか、MD5 ダイジェスト (`digest`) 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を設定する必要もあります。このプロパティは、ブローカインスタンス設定ファイル内にあります。プロパティについては、この節の後半でまとめて説明しています。

   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ファイルの使用に関する詳細は、153 ページの「ユーザーの承認 : アクセス制御プロパティファイル」を参照してください。

5. コネクション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の `imq.user_repository.ldap.ssl.enabled` を `true` に設定します。

次に示すのは LDAP 関連のプロパティです。

- `imq.user_repository.ldap.server`。LDAP サーバーの *host:port* です。
- `imq.user_repository.ldap.principal`。検索時にブローカがディレクトリサーバーにバインドするために使用する識別名です。
- `imq.user_repository.ldap.password`。ブローカが使用する識別名と関連付けられたパスワードです。
- `imq.user_repository.ldap.base`。ユーザーエントリのためのディレクトリベースです。
- `imq.user_repository.ldap.uidattr`。プロバイダ固有の属性識別子。その値はユーザーを一意に識別します。たとえば、次のように指定します。
  `uid, cn`
- `imq.user_repository.ldap.usrfilter`。ユーザーとともに使用する JNDI 検索フィルタです。
- `imq.user_repository.ldap.grpsearch`。グループ検索を有効にするかどうかを指定するブール値です。
- `imq.user_repository.ldap.grpbase`。グループエントリのためのディレクトリベースです。
- `imq.user_repository.ldap.gidattr`。プロバイダ固有の属性識別子。その値はグループ名です。

- imq.user_repository.ldap.memattr。グループエントリにある属性識別子。その値はグループメンバーの識別名です。
- imq.user_repository.ldap.grpfiltler。グループとともに使用する JNDI 検索フィルタです。
- imq.user_repository.ldap.timeout。検索の時間制限を秒単位で指定する整数です。
- imq.user_repository.ldap.ssl.enabled。LDAP サーバーとの通信時にブローカが SSL プロトコルを使用するかどうかを指定するブール値です。

これらのプロパティの詳細は、328 ページの「セキュリティマネージャのプロパティ」を参照してください。

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ファイルで、ADMIN コネクション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 コネクション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 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 「オペレーティングシステムごとの 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。

```
サービスコネクションアクセス制御
##################################
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ファイル

アクセス制御プロパティファイル (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。

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。

- コネクションの作成
- 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。

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 (144 ページの「単層型ファイルユーザーリポジトリを使用する」を参照) LDAP ユーザーリポジトリにある場合でも (150 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)、ACL ファイルが使用されます。

アクセス制御プロパティファイルの作成

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「オペレーティングシステムごとの 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。

- コネクションのアクセス制御
- 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 プロパティでは、ACL プロパティ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コネクション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、すべてのコネクションサービス、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。

表 7-3　アクセス規則の構文要素 (続き)

要素	説明
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、147 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 ql 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 `http://java.sun.com/j2se/1.4/docs/guide/intl/faq.html` を参照してください。

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する。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する。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する。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する。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、暗黙的に拒否される。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る。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る。

コネクションサービスのアクセス制御

ACL プロパティファイルのコネクションアクセス制御のセクションには、ブローカのコネクションサービスのアクセス制御規則が含まれます。コネクション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コネクションサービスです。

デフォルトの ACL プロパティファイルは、すべてのユーザーに NORMAL コネクション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 コネクション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、imqusermgr 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。

コネクション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 プロパティファイルを使用して ADMIN コネクション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 コネクション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 コネクション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 コネクション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のを拒否する。

  ```
  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) 標準に基づくコネクションサービスは、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 ベースのコネクションサービスの設定方法を説明します。

Message Queue は、SSL (Secure Socket Layer) 規格に基づく次のコネクションサービスをサポートしています。

- `ssljms`, `ssladmin` と `cluster` は、TCP/IP で使用される。
- `httpsjms` は HTTP で使用される。

これらのコネクションサービスにより、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 ベースのコネクションサービスを使用するには、キーツールユーティリティ (`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は、ブローカへのコネクション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、コネクションを暗号化します。

Message Queue の SSL ベースのコネクション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 で安全なコネクション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 ベースのコネクションサービス `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コネクションを確立できます。HTTP を使用した安全なコネクションの確立の詳細は、385 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 を介した安全なコネクションサービス

次の SSL ベースのコネクションサービスは、TCP/IP を介した直接的で安全なコネクションを提供します。

- `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コネクション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 (`imqcmd`) とブローカ間に安全な暗号化コネクションを作成します。安全なコネクションは、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コネクションを介したブローカ間通信を確立します (198 ページの「ブローカ間の安全なコネクション」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 166 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 ベースのコネクション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 のいずれかのコネクション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 コネクションサービスだけに適用)。

ssljms および ssladmin コネクション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。

imqkeytool コマンドを実行し、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 imqkeytool を実行する必要があります。

ssljms、ssladmin、cluster のコネクションサービスに対して、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、`imqkeytool` により確認の画面が表示されます。たとえば、次のように指定します。

```
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、`imqkeytool` 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、`imqkeytool` 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。また、キーストアパスワードを passfile (169 ページの「passfile の使用」を参照) に格納できます。

imqkeytool を実行すると、JDK keytool ユーティリティが実行されて、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 「オペレーティングシステムごとの 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 「オペレーティングシステムごとの 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。

- 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 161 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 「オペレーティングシステムごとの 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、161 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 プロパティに追加する必要があります。

注 imq.service.activelist プロパティではなく、imq.cluster.transport プロパティを使用して、SSL ベースの cluster コネクションサービスを有効にします。198 ページの「ブローカ間の安全なコネクション」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「オペレーティングシステムごとの 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、プロパティには jms コネクションサービスと admin コネクション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 コネクションサービスか ssladmin コネクションサービス、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する

  ```
  imqbrokerd
  Please enter Keystore password:mypassword
  ```

- 169 ページの「passfile の使用」の説明に従って、パスワードをパスファイルに格納する。パスワードをパスファイルに格納し、プロパティ `imq.passfile.enabled=true` を設定した後で次のいずれかを実行します。

  - `imqbrokerd` コマンドに passfile の場所を渡す。

    ```
    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を使用する passfile の場所を指定して、ブローカを起動する。

    ```
    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

最後に安全なコネクション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 を使用した安全なコネクションには、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` コネクション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、コネクションに 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` コネクション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、安全な管理コネクションを確立できます。たとえば、次のように指定します。

```
imqcmd list svc -b hostName:port -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、149 ページの「デフォルトの管理者パスワードの変更」を参照)。

コネクション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、安全な管理コネクション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 コネクションを確立するときに、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、161 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 コネクション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. 165 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 コネクションの確立を試みます。

   broker の証明書に著名な CA が署名している場合、コネクション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。コネクション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 truststore に署名 CA の root 証明書をインストールします。

truststore 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する。
- root CA を代替システム jssecacerts ファイルにインストールする。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 truststore として使用するようにクライアントを設定する。

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、truststore 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 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
```

passfile 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	プラグイン持続データストア、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	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	プラグインデータストアを管理	データストアにアクセスします

パスワードファイル (*passfile*) 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。今回のリリースでは、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、パスファイルを使用する必要があります。

パス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。

パスファイルの内容

パスファイルは、一連のプロパティ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。

passfile 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 passfile 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 imdbmgr	必要に応じて、データベースコネクションを開くときに使用するパスワードを指定します。

表 7-7　passfile のパスワード (続き)

パスワード	影響を受けるコマンド	説明
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 「オペレーティングシステムごとの 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、パージ、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。

imq.audit.enabled プロパティの詳細は、328 ページの「セキュリティマネージャのプロパティ」を参照してください。

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を使用すると、ほかの JMS プロバイダへの移植が可能なクライアントアプリケーションを開発できます。管理対象オブジェクトは、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します。

Message Queue の管理者は、通常、クライアントアプリケーションがブローカコネクションの取得に使用するための管理対象オブジェクトを作成します。クライアントアプリケーションは、コネクションを使用して物理的送信先とメッセージの送受信を行います。

この章では、オブジェクトマネージャユーティリティ (`imqobjmgr`) を使用して、これらのタスクを実行する方法について説明します。これらのタスクを実行するには、使用するオブジェクトストアと作成する管理対象オブジェクトの両方の属性を理解する必要があるため、この章では、`imqobjmgr` を使用して管理対象オブジェクトを管理する方法を説明する前に、この 2 つのトピックの背景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。

オブジェクトストアについて

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 検索を行うときに、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。標準 LDAP ディレクトリサーバーまたはファイルシステムのオブジェクトストアの 2 種類のオブジェクトストアが使用できます。

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 実装は、多数のベンダーでサポートされており、分散システムでの使用を考慮した設計になっています。LDAP サーバーは、本稼動環境で役立つセキュリティ機能も備えています。

Message Queue 管理ツールは、LDAP サーバー上のオブジェクトストアを管理できます。ただし、はじめに LDAP サーバーのマニュアルに記載されているとおり、java オブジェクトを格納し JNDI 検索を実行するように LDAP サーバーを設定する必要があります。

また、LDAP サーバーをオブジェクトストアとして使用している場合は、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** : LDAP サーバーオブジェクトストアの場合、この属性は固定です。
- **ロケーション** : LDAP サーバーの設定時に、管理対象オブジェクトの URL とディレクトリパスを指定します。特に、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。
- **セキュリティ情報** : LDAP プロバイダによって異なります。セキュリティ情報をすべての操作で必要とするのか、あるいは格納データを変更する操作にだけ必要とするのか決める場合、使用する LDAP 実装に付属するマニュアルを参照する必要があります。

表 8-1 LDAP オブジェクトストアの属性

属性	説明
`java.naming.factory.initial`	LDAP サーバー上の JNDI 検索用の初期コンテキスト `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	LDAP サーバーの URL とディレクトリパス情報。たとえば、次のように指定します。 ldap://mydomain.com:389/ou=mqobjs,o=myapp 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	LDAP サーバーの呼び出し元を認証するための主体の識別情報。このエントリの形式は、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。 uid=fooUser, ou=People, o=mq このプロパティ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	LDAP サーバーの呼び出し元を認証するための主体の証明書。プロパティの値は、認証スキーマによって異なります。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、証明書などが使用できます。たとえば、次のように指定します。 fooPasswd このプロパティ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	使用するセキュリティのレベル。値は、none、simple、strong のどれかのキーワードになります。 たとえば、simple を指定した場合、値がまだ指定されていない主体または証明書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。 このプロパティ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 は、ファイルシステムのオブジェクトストア実装もサポートしています。ファイルシステムのオブジェクトストアは、まだ十分なテストが行われていないため、運用システムでの使用はお勧めしませんが、開発環境で使いやすいという利点があります。LDAP サーバーをセットアップする必要はなく、ローカルのファイルシステム上にディレクトリを作成するだけで利用できます。

ただし、クライアントが複数のコンピュータノードにまたがって配備されている場合は、これらのクライアントがオブジェクトストアの常駐するディレクトリに対してアクセス権を持つときにだけ、ファイルシステムストアを集中オブジェクトストアとして使用できます。さらに、このディレクトリにアクセス可能なユーザーは、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することができます。

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。これらの属性は、次のように分類されます。

- **初期コンテキスト**: ファイルシステム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**: この属性の値は、使用する管理対象オブジェクトを格納するディレクトリパスを指定します。ディレクトリが存在していて、Message Queue 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は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`	ファイルシステムオブジェクトストアの JNDI 検索の初期コンテキスト。 `com.sun.jndi.fscontext.RefFSContextFactory`
`java.naming.provider.url`	ディレクトリパス情報。たとえば、次のように指定します。 `file:///C:/myapp/mqobjs`

管理対象オブジェクトの属性について

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。

- コネクションファクトリ管理対象オブジェクト。ブローカへのコネクションを作成するために、クライアントアプリケーションが使用する。
- 送信先の管理対象オブジェクト。プロデューサがメッセージを送信する宛先、またはコンシューマがメッセージを受け取る受信先を識別するためにクライアントアプリケーションが使用する。

Message Queue は、管理対象オブジェクトを作成したり管理したりするための 2 つの管理ツールを提供しています。コマンド行オブジェクトマネージャユーティリティ (`imqobjmgr`) と GUI 管理コンソールです。この章では、コマンド行の使い方についてのみ説明します。

管理対象オブジェクトの属性は、属性と値の組み合わせで指定します。

コネクションファクトリの属性

コネクションファクトリの設定は、クライアントアプリケーションに代わってコネクションファクトリが作成したすべてのコネクションに渡されます。コネクションでは、メッセージの送受信に関連したパーティの定義、クライアントランタイムによるメッセージフローの処理方法、コネクションを介して送信されるすべてのメッセージに対する特定の情報の自動設定が設定されます。

コネクションファクトリオブジェクトには 2 つのタイプがあります。

- `ConnectionFactory`。通常のメッセージおよび分散していないトランザクションをサポート。
- `XAConnectionFactory`。分散トランザクションをサポート。

`ConnectionFactory` オブジェクトと `XAConnectionFactory` オブジェクトは、一連の同じ属性値を共有します。

コネクションファクトリオブジェクトは、管理者またはアプリケーション (プロトタイプまたはテストの場合) が作成および設定できます。コネクションファクトリの属性は、`imqobjmgr` ツールまたは管理コンソールを使用して設定します。

この節では、以降の節で記述するコネクションファクトリの属性を説明します。以降の節は属性により影響を受ける動作別に書かれています。

- 178 ページの「コネクションの処理」
- 180 ページの「クライアントの識別」
- 182 ページの「信頼性およびフロー制御」

- 183 ページの「キューブラウザの動作とサーバーセッション」
- 184 ページの「メッセージヘッダーのオーバーライド」
- 183 ページの「JMS の定義済みプロパティのサポート」

主に使用する属性は、imqAddressList です。この属性を使用して、クライアントがコネクションを確立するブローカを指定します。189 ページの「コネクションファクトリの追加」に、コネクションファクトリ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際に、属性を指定する方法を説明しています。

コネクションファクトリの属性の詳細は、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。また Message Queue のクラスの com.sun.messaging.ConnectionConfiguration については JavaDoc API のマニュアルを参照してください。

コネクションの処理

コネクション処理の属性は、接続が必要なメッセージサーバーアドレスの指定、および再接続が必要な場合、再接続を試行する回数と間隔を指定する場合に使用します。

クライアントは、imqAddressList 属性の値として指定するメッセージサーバーアドレスで、メッセージサーバーに接続します。メッセージサーバーアドレスには、ブローカホスト名、ポート番号、コネクションサービスタイプが含まれています。

ポート番号は、ブローカのポートマッパーが常駐するポートか、特定のコネクションサービスが常駐するポートになります。ポートマッパーのポートを指定する場合、ポートマッパーはコネクションのポート番号を動的に割り当てます。メッセージサーバーアドレスの指定の詳細は、345 ページの「imqAddressList 属性値の構文」を参照してください。

自動再接続

シングルブローカ環境またはマルチブローカクラスタ環境では、コネクションが失敗したときにクライアントが自動的にブローカに再接続するコネクション処理の属性を設定します。また再接続プロセスも設定できます。

再接続機能は、コネクションのフェイルオーバーを扱いますが、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損なわれます。

自動再接続が有効に設定されている場合、Message Queue はコネクションが失敗した場合に一時的送信先を維持します。これはクライアントが再び接続し、アクセスする場合があるためです。クライアントに割り当てられた一時的送信先を使用して再接続する時間が経過すると、ブローカは一時的送信先を削除します。

再接続を処理する方法は、クライアントがシングルブローカに接続するのか、クラスタ内の 1 つのブローカに接続するかにより異なります。次の節では、この 2 つの可能性について説明します。

シングルブローカへの再接続：コネクションに失敗したとき、クライアントが自動的にブローカに再接続するように設定するには、次のコネクションファクトリ属性を設定します。

- `imqReconnectEnabled`。自動再接続動作を有効にします。
- `imqReconnectAttempts`。クライアントランタイムがクライアントへの再接続を試みる回数を指定します。
- `imqReconnectInterval`。クライアントランタイムがクライアントへの再接続を試みる間隔を指定します。

これらの属性の詳細は、344 ページの「コネクションの処理」を参照してください。

クラスタ内のブローカへの再接続：マルチブローカクラスタ環境では、`imqAddressList` 属性に複数のアドレスを指定している場合、リスト内の各ブローカについて自動再接続が反復されます。リスト内のすべてのブローカに、Message Queue Enterprise Edition がインストールされている必要があります。

リスト内の最初のアドレスに対するクライアントのコネクションに失敗すると、クライアントランタイムはリストの別のブローカへのクライアントの再接続を試みます。その試行が失敗した場合、クライアントランタイムは、クライアントに再接続できるようになるまで、リスト内で再接続を繰り返します。

どの再接続も失敗する場合、クライアントランタイムは利用可能なブローカが見つかるまで、または利用可能なブローカの検出に失敗するまで、リスト内を指定された回数だけ循環します。`imqAddressListBehavior` 属性の設定により、再接続に選択されたブローカがアドレスリスト内のアドレス順の次に位置するか、そのリストからランダムに選択されたブローカであるか決定します。

クライアントのクラスタ内のブローカへの再接続を有効にするには、次の属性を使用します。

- `imqReconnectEnabled`。自動再接続動作を有効にします。
- `imqReconnectAttempts`。次に移る前に、各ブローカアドレスを試行する回数を指定します。
- `imqReconnectInterval`。次の試行までに待機する時間を指定します。
- `imqAddressListIterations`。リストで反復する回数を指定します。
- `imqAddressListBehavior`。コネクションの試行がアドレスリスト内のアドレス順で行われるか、ランダムに行われるかを指定します。

これらの属性の詳細は、344 ページの「コネクションの処理」を参照してください。

ping コネクション

imqPingInterval 属性は、クライアントランタイムからブローカに対する ping 操作の周期を指定します。コネクションを定期的にテストすることで、クライアントランタイムは障害の発生したコネクションを事前に検出することができます。ping 操作が失敗すると、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の例外リスナーオブジェクトに例外をスローします。アプリケーションに例外リスナーが存在しない場合、アプリケーションが次にコネクションを使用する試みも失敗します。

ping の使用は、メッセージの受信を待機し、メッセージを送信しないコンシューマクライアントアプリケーションに特に重要です。このようなアプリケーションは、この方法以外に、コネクションの失敗を認識する手段がありません。メッセージを時々生成するクライアントにとっても、この機能は有益です。メッセージを送信する前に、障害の発生したコネクションを処理できるためです。

デフォルトでは、ping 間隔は 30 秒に設定されます。-1 の値により ping 操作が無効になります。

破損したコネクションへの応答は、オペレーティングシステムにより異なります。たとえば、一部のオペレーティングシステムでは、ping から障害が迅速に報告されます。また、ブローカへのコネクションの確立を試行し続け、ping が成功するかバッファがオーバーフローになるまで、一連の ping をバッファリングするオペレーティングシステムもあります。

imqPingInterval 属性の詳細は、344 ページの「コネクションの処理」を参照してください。

クライアントの識別

メッセージキューは、クライアント認証と、永続サブスクライバに必要な固有のクライアント ID の設定をサポートする、一連のコネクションファクトリ属性を定義します。

ブローカへの接続を試みるクライアントには、認証が必要です。クライアントがコネクションの作成時にユーザー名またはパスワードを指定しない場合、次のいずれかが起こります。

- コネクションファクトリ属性 imqDefaultUsername と imqDefaultPassword が設定されていない場合、クライアントランタイムは値 guest/guest をブローカに渡し、ブローカはその値を使用してクライアントを認証します。

  ユーザーリポジトリにはエントリ guest/guest が同梱されているため、クライアントはコネクションを取得できます。

- コネクションファクトリ属性 imqDefaultUsername と imqDefaultPassword が設定されている場合、クライアントランタイムはその値をブローカに渡し、ブローカはその値を使用してクライアントを認証します。

ユーザー / パスワードの組み合わせがユーザーリポジトリ内に存在する場合、クライアントはコネクションを確立できます。

このスキーマにより、どのユーザーもコネクションを確立できます。これは開発とテストに好都合です。運用システムでは、コネクションへのアクセスがユーザーリポジトリに追加されたユーザーに制限されます。

コネクションを要求するクライアントのブローカ認証以外に、JMS 仕様では、クライアントに対して状態を維持する必要がある場合に、コネクションが固有のクライアント識別子を割り当てる必要があります。Message Queue はクライアント ID を使用して、そ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停止すると、ブローカはそのサブスクライバのメッセージを保持し、サブスクライバが再びアクティブになった場合にメッセージを配信します。ブローカはクライアント ID を使用して、サブスクライバを識別します。

ClientID は管理者として設定できます。またはクライアントがプログラムで設定することもできます。複数のクライアントが同じコネクションファクトリオブジェクトからコネクションを取得する場合、コネクションファクトリに ClientID を設定します。Message Queue はそのファクトリから取得した各コネクションに対して、固有の ClientID を指定します。

ClientID 値が確実に固有の値になるためには、次の形式を使用して `imqConfiguredClientID` 属性を設定します。

`imqConfiguredClientID=${u}`*string*

属性値の先頭の 4 文字は必ず `${u}` になります。「u」以外の文字が見つかると、コネクションの作成時に JMS 例外が発生します。

string の値は `Xconn` など、このコネクションファクトリで生成されるコネクションに関連付けられる任意の値です。ユーザー認証の段階では、Message Queue は `u:`*userName* を *u* に置き換えます。たとえば、コネクションに関連付けられたユーザーが `Athena`、コネクションに指定された文字列が `${u}Xconn` であれば、ClientID は `u:AthenaXconn` になります。

このスキーマにより、ほかの方法でも同様ですが、コネクションファクトリにより生成された各コネクションには固有の ClientID が格納されます。

このスキーマが機能しないケースがあります。2 つのクライアントが `guest` などのデフォルトのユーザー名を使用してコネクションを取得する場合、各クライアントは同じ `${u}` コンポーネントを含む ClientID を保有します。実行時に、最初にコネクションを要求するクライアントがコネクションを取得し、2 番目にコネクションを要求するクライアントはコネクションを取得できません。これは Message Queue が固有ではない ClientID を使用してコネクションを作成できないためです。

`imqDisableSetClientID` 属性を設定して、コネクションファクトリを使用するクライアントが、設定済みのクライアント ID をプログラムで変更するのを禁止できます。

アプリケーションコードが setClientId() メソッドを使用していなければ、永続サブスクリプションに imqConfiguredClientID 属性を設定する必要があります。

クライアント識別に影響する属性を、次のようにまとめています。

- imqDefaultUsername。クライアントがコネクションの作成時にユーザー名を指定しない場合、ブローカの認証に使用するデフォルトのユーザー名を指定します。
- imqDefaultPassword。クライアントがコネクションの作成時にパスワードを指定しない場合、ブローカの認証に使用するデフォルトのパスワードを指定します。
- imqConfiguredClientID。管理者が設定するクライアント ID の値を指定します。
- imqDisableSetClientID。コネクションファクトリを使用するクライアントが、プログラムでクライアント ID を変更できるかどうかを指定します。

これらの属性の詳細は、348 ページの「クライアントの識別」を参照してください。

信頼性およびフロー制御

クライアントによって送受信されるメッセージと Message Queue が使用する制御メッセージは、同じクライアントとブローカ間のコネクションを使って伝送されます。その結果、ブローカ通知など、制御メッセージの配信時に、JMS メッセージの配信により制御メッセージが保留になると遅延が起こります。

制御メッセージのフローを、クライアントメッセージのフローを基準に管理できる、コネクションファクトリ属性を設定できます。2 種類のメッセージのフローを制御する場合、信頼性とスループットとの妥協が起こります。これらの属性を使用してフロー制御と信頼性を管理する方法の詳細は、246 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。

次の属性は、クライアントのメッセージと制御メッセージのフローに影響します。

- imqAckTimeout。クライアントランタイムがブローカの応答を待機する最長時間をミリ秒で指定します。
- imqConnectionFlowCount。一定量のバッチの JMS メッセージ数を指定します。
- imqConnectionFlowLimitEnabled。コネクションレベルでメッセージフローを制限します。
- imqConnectionFlowLimit。コネクションを介して配信され、消費を待機する間、クライアントランタイムにバッファリングされるメッセージ数の制限を指定します。
- imqConsumerFlowLimit。コネクションを介して配信され、消費を待機する間、クライアントランタイムにバッファリングされるメッセージ数のコンシューマ別の制限を指定します。

- imqConsumerFlowThreshold。コンシューマのメッセージの配信が再開するまで、クライアントランタイムにバッファリングされる各コンシューマのメッセージ数を、imqConsumerFlowLimit のパーセンテージとして指定します。

これらの属性の詳細は、350 ページの「信頼性およびフロー制御」を参照してください。

キューブラウザの動作とサーバーセッション

次の属性はクライアントキューのブラウズに影響します。

- imqQueueBrowserMaxMessagesPerRetrieve。キュー送信先の内容を検索する場合、クライアントが一度に取得できる最大メッセージ数を指定します。
- imqQueueBrowserRetrieveTimeout。キュー送信先の内容を検索する場合、クライアントがメッセージの取得を待機する時間を指定します。
- imqLoadMaxToServerSession。JMS アプリケーションサーバー機能の場合、Message Queue ConnectionConsumer が ServerSession のセッションに対して一度にメッセージをロードする数が maxMessages の数までなのか、あるいは 1 つなのかを指定します。

これらの属性の詳細は、353 ページの「キューブラウザの動作とサーバーセッション」を参照してください。

JMS の定義済みプロパティのサポート

コネクションファクトリ属性を使用して、コネクションが生成するメッセージに、JMS の定義済みプロパティを自動的に設定できます。JMS プロパティは、http://java.sun.com/products/jms/docs.html の JMS 仕様で定義されています。

次の属性を使用して、JMS の定義済みプロパティを設定します。

- imqSetJMSXUserID。生成されたメッセージに対して、Message Queue が JMS の定義済みプロパティ JMSXUserID (メッセージを送信するユーザーの ID) を設定するかどうかを指定します。
- imqSetJMSXAppID。生成されたメッセージに対して、Message Queue が JMS の定義済みプロパティ JMSXAppID (メッセージを送信するアプリケーションの ID) を設定するかどうかを指定します。
- imqSetJMSXProducerTXID。生成されたメッセージに対して、Message Queue が JMS の定義済みプロパティ JMSXProducerTXID (メッセージを生成するトランザクションのトランザクション識別子) を設定するかどうかを指定します。
- imqSetJMSXConsumerTXID。消費されたメッセージに対して、Message Queue が JMS の定義済みプロパティ JMSXConsumerTXID (メッセージを消費するトランザクションのトランザクション識別子) を設定するかどうかを指定します。

- imqSetJMSXRcvTimestamp。消費されたメッセージに対して、Message Queue が JMS の定義済みプロパティ JMSXRcvTimestamp (メッセージをコンシューマに配信する時間) を設定するかどうかを指定します。

これらの属性の詳細は、353 ページの「JMS の定義済みプロパティのサポート」を参照してください。

メッセージヘッダーのオーバーライド

コネクションファクトリの属性を設定すると、持続性、生存期間、メッセージの優先順位を指定する JMS メッセージヘッダーのフィールドをオーバーライドできます。この設定は、コネクションファクトリから取得したコネクションが生成するすべてのメッセージに使用されます。

次の JMS フィールドの値はオーバーライドできます。

- JMSDeliveryMode (メッセージの持続性 / 非持続性)
- JMSExpiration (メッセージの生存期間)
- JMSPriority (メッセージの優先順位、0 ～ 9 の整数)

これらのフィールドの詳細は、http://java.sun.com/products/jms/docs.html の JMS 仕様を参照してください。

メッセージのヘッダーをオーバーライドすると、アプリケーション要件を侵害する場合があるため、この機能はアプリケーションユーザーまたはデザイナーに応じて使用してください。

次のリストは、メッセージのオーバーライドを扱うコネクションファクトリの属性を示したものです。ほとんどの属性は組になっています。各組み合わせについて、最初の属性は指定されたヘッダーフィールドをオーバーライドできるかどうかを指定し、2 番目の属性がオーバーライドの値を指定します。

- imqOverrideJMSDeliveryMode と imqJMSDeliveryMode。最初の属性はクライアントが設定する JMSDeliveryMode フィールドのオーバーライドが可能かどうかを指定し、2 番目の属性はオーバーライドの値を指定します。
- imqOverrideJMSExpiration と imqJMSExpiration。最初の属性はクライアントが設定する JMSExpiration フィールドのオーバーライドが可能かどうかを指定し、2 番目の属性はオーバーライドの値を指定します。
- imqOverrideJMSPriority と imqJMSPriority。最初の属性はクライアントが設定する JMSPriority フィールドのオーバーライドが可能かどうかを指定し、2 番目の属性はオーバーライドの値を指定します。
- imqOverrideJMSHeadersToTemporaryDestinations。オーバーライドを一時的送信先に適用するかどうかを指定します。

これらの属性の詳細は、349 ページの「メッセージヘッダーのオーバーライド」を参照してください。

送信先管理対象オブジェクトの属性

物理的なトピックやキューの送信先を指定する、送信先管理対象オブジェクトには、343 ページの表 16-1 に示すような属性があります。190 ページの「トピックまたはキューの追加」では、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する場合に、これらの属性を指定する方法について説明します。

主に使用する属性は、`imqDestinationName` です。これは、トピックまたはキューの管理対象オブジェクトに対応する物理的送信先に割り当てる名前です。複数のアプリケーションをサポートするために作成するほかの送信先と区別するために、送信先の説明を指定することもできます。

詳細は、Message Queue クラスの `com.sun.messaging.DestinationConfiguration` に関する JavaDoc API ドキュメントを参照してください。

オブジェクトマネージャユーティリティ (imqobjmgr) の使用

オブジェクトマネージャユーティリティを使用すると、Message Queue の管理対象オブジェクトを作成および管理することができます。このユーティリティの使用により、次の作業が実行できます。

- オブジェクトストアへの管理対象オブジェクトの追加やオブジェクトストアからの管理対象オブジェクトの削除。
- 既存の管理対象オブジェクトの一覧表示。
- 管理対象オブジェクトに関する情報のクエリーおよび表示。
- オブジェクトストアにある既存の管理対象オブジェクトの変更。

`imqobjmgr` コマンドの構文、サブコマンド、オプションの詳細は、第 13 章「コマンドのリファレンス」を参照してください。次の節では、`imqobjmgr` のサブコマンドを操作する場合に指定する必要のある情報を説明します。

必要な情報

管理対象オブジェクトに関連する大部分のタスクを実行する場合は、`imqobjmgr` のサブ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タイプ**

  使用可能なタイプを 301 ページの表 13-11 に示します。
- 管理対象オブジェクトの **JNDI 検索名**

これはオブジェクトストア内の管理対象オブジェクトを (JNDI を使用して) 参照する場合に、クライアントコードで使用される論理名です。

- **管理対象オブジェクトの属性** (特に、`add` および `update` サブコマンドで必要)
  - 送信先の場合 : ブローカの物理的送信先の名前。これは `imqcmd create dst` サブコマンドの `-n` オプションで指定した名前となります。この名前を指定しない場合、デフォルト名の `Untitled_Destination_Object` が使用されます。
  - コネクションファクトリの場合 : もっとも一般的に使用される属性は、クライアントの接続先である 1 つ以上のメッセージサーバーアドレスを指定するためのアドレスリスト (`imqAddressList`) です。この情報を指定しないと、ローカルホストとデフォルトのポート番号 (`7676`) が使用されます。つまり、クライアントはローカルホスト上のポート `7676` のブローカに対してコネクションの確立を試みます。189 ページの「コネクションファクトリの追加」では、オブジェクトの属性を指定する方法について説明します。

    その他の属性については、177 ページの「コネクションファクトリの属性」を参照してください。
- **オブジェクトストアの属性**

  この情報は、ファイルシステムストアと LDAP サーバーのどちらを使用するかによって異なりますが、次の属性を設定する必要があります。
  - JNDI 実装のタイプ (初期コンテキスト属性)。たとえば、ファイルシステムまたは LDAP です。
  - オブジェクトストア内の管理対象オブジェクトの場所 (プロバイダの URL 属性)。管理対象オブジェクトが存在する「フォルダ」です。
  - オブジェクトストアへのアクセスに必要なユーザー名、パスワード、および認証タイプ。

  オブジェクトストアの属性については、174 ページの「LDAP サーバーオブジェクトストア」および176 ページの「ファイルシステムオブジェクトストア」を参照してください。

コマンドファイルの使用

`imqobjmgr` コマンドを使用すると、`imqobjmgr` サブコマンド句のすべてまたは一部を表すために java プロパティファイルの構文を使用する、コマンドファイルの名前を指定できます。

オブジェクトマネージャユーティリティ (imqobjmgr) と一緒にコマンドファイルを使用すると、オブジェクトストアの属性を指定する場合に特に便利です。なぜなら、オブジェクトストアの属性は複数の imqobjmgr の呼び出しにおいて、同じ内容になる可能性が高く、通常多くの入力作業が必要になるためです。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入力してしまうのを防ぐことができます。

imqobjmgr コマンドファイルの一般的な構文は、次のとおりです (バージョンプロパティは Message Queue 製品ではなくコマンドファイルのバージョンを示す。これはコマンド行オプションではなく、値は常に 2.0 に設定される)。

```
version=2.0
cmdtype=[ add | delete | list | query | update ]
obj.type=[ q | t | qf | tf | cf | xqf | xtf | xcf | e ]
obj.lookupName=lookup name
obj.attrs.objAttrName1=value1
obj.attrs.objAttrName2=value2
obj.attrs.objAttrNameN=valueN
...
objstore.attrs.objStoreAttrName1=value1
objstore.attrs.objStoreAttrName2=value2
objstore.attrs.objStoreAttrNameN=valueN
...
```

コマンドファイルを使用する方法の例として、次の imqobjmgr コマンドを検討してください。

```
imqobjmgr add
       -t qf
       -l "cn=myQCF"
       -o "imqAddressList=mq://foo:777/jms"
       -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=fooUser, ou=People, o=imq"
       -j "java.naming.security.credentials=fooPasswd"
       -j "java.naming.security.authentication=simple"
```

このコマンドは、次の内容を含む MyCmdFile などのファイルにカプセル化することができます。

```
version=2.0
cmdtype=add
obj.type=qf
obj.lookupName=cn=myQCF
obj.attrs.imqAddressList=mq://foo:777/jms
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=fooUser, ou=People, o=imq
  objstore.attrs.java.naming.security.credentials=fooPasswd
  objstore.attrs.java.naming.security.authentication=simple
```

次に、-i オプションを使用して、このファイルをオブジェクトマネージャユーティリティ (imqobjmgr) に渡すことができます。

```
imqobjmgr -i MyCmdFile
```

コマンドファイルを使用して一部のオプションを指定する一方で、コマンド行を使用してその他のオプションを指定することも可能です。このため、ユーティリティの複数の呼び出しで、同じ内容になるサブコマンド句の一部を、コマンドファイルを使用して指定することができます。たとえば、次のコマンドでは、コネクションファクトリ管理対象オブジェクトを追加する場合に必要なすべてのオプション (ただし、管理対象オブジェクトの保存場所を指定するオプションは除く) が指定されています。

```
imqobjmgr add
       -t qf
      -l "cn=myQCF"
      -o "imqAddressList=mq://foo:777/jms"
       -i MyCmdFile
```

この場合、MyCmdFile ファイルには次の定義が含まれます。

```
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=fooUser, ou=People, o=imq
      objstore.attrs.java.naming.security.credentials=fooPasswd
      objstore.attrs.java.naming.security.authentication=simple
```

コマンドファイルの別の例は、次の場所で参照できます。

`/usr/demo/imq/imqobjmgr` (Solaris)
`/opt/sun/mq/examples/imqobjmgr` (Linux)
`IMQ_HOME/demo/imqobjmgr` (Windows)

管理対象オブジェクトの追加および削除

この節では、コネクションファクトリおよびトピックまたはキューの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する方法について説明します。

注 オブジェクトマネージャユーティリティ (`imqobjmgr`) 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。

コネクションファクトリの追加

クライアントアプリケーションがブローカへのコネクションを取得できるようにするには、クライアントアプリケーションに必要なコネクションタイプを表している管理対象オブジェクトを追加します。このタイプは、トピックコネクションファクトリかキューコネクションファクトリのどちらかになります。

キューのコネクションファクトリを追加するには、次のようなコマンドを使用します。

```
imqobjmgr add
    -t qf
    -l "cn=myQCF"
    -o "imqAddressList=mq://myHost:7272/jms"
    -j "java.naming.factoryinitial=
                com.sun.jndi.ldap.LdapCtxFactory"
    -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
    -j "java.naming.security.principal=
```

```
            uid=fooUser, ou=People, o=imq"
    -j "java.naming.security.credentials=fooPasswd"
    -j "java.naming.security.authentication=simple"
```

先行するコマンドは、検索名が cn=myQCF であり、myHost 上で実行するブローカに接続する管理対象オブジェクトを作成し、ポート 7272 で待機します。この管理対象オブジェクトは、LDAP サーバーに格納されます。引数としてコマンドファイルを imqobjmgr コマンドに指定すると、同じことを実行できます。詳細は、186 ページの「コマンドファイルの使用」を参照してください。

注 **命名規則**：LDAP サーバーを使用して、管理対象オブジェクトを格納する場合、前述の例のように、接頭辞「cn=」が付いた検索名 (cn=myQCF) を割り当てることが重要です。検索名は、-l オプションを使用して指定します。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は、cn 接頭辞を使用する必要はありません。ただし、「/」を含む検索名は使用しないでください。表 8-3 を参照してください。

表 8-3 命名規則の例

オブジェクトストアのタイプ	適した名前	使用が禁止されている名前
LDAP サーバー	cn=myQCF	myQCF
ファイルシステム	myTopic	myObjects/myTopic

トピックまたはキューの追加

クライアントアプリケーションがブローカ上の物理的送信先にアクセスできるようにするには、これらの送信先を指定する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。

該当する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前に、物理的送信先を作成しておくことをお勧めします。コマンドユーティリティ (imqcmd) を使用して、オブジェクトストア内の送信先管理対象オブジェクトによって識別される、ブローカの物理的送信先を作成してください。物理的送信先の作成方法については、121 ページの「コネクション情報の入手」を参照してください。

次のコマンドでは、検索名が myTopic で、物理的送信先名が TestTopic のトピックの送信先を識別する管理対象オブジェクトが追加されます。この管理対象オブジェクトは、LDAP サーバーに格納されます。

```
imqobjmgr add
-t t
-l "cn=myTopic"
-o "imqDestinationName=TestTopic"
-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=fooUser, ou=People, o=imq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

次は同じコマンドです。ただし、管理対象オブジェクトが Solaris のファイルシステムに格納されるという点が異なります。

```
imqobjmgr add
      -t t
      -l "cn=myTopic"
      -o "imqDestinationName=TestTopic"
      -j "java.naming.factory.initial=
            com.sun.jndi.fscontext.RefFSContextFactory"
      -j "java.naming.provider.url=
            file:///home/foo/imq_admin_objects"
```

たとえば、LDAP サーバーの場合、`MyCmdFile` というコマンドファイルを使用して、サブコマンド句を指定できます。ファイルには、次のテキストが含まれます。

```
version=2.0
cmdtype=add
obj.type=t
obj.lookupName=cn=myTopic
obj.attrs.imqDestinationName=TestTopic
objstore.attrs.java.naming.factory.initial=
            com.sun.jndi.fscontext.RefFSContextFactory
objstore.attrs.java.naming.provider.url=
            file:///home/foo/imq_admin_objects
  objstore.attrs.java.naming.security.principal=
            uid=fooUser, ou=People, o=imq
  objstore.attrs.java.naming.security.credentials=fooPasswd
  objstore.attrs.java.naming.security.authentication=simple
```

ファイルを imqobjmgr コマンドに渡す場合は、-i オプションを使用します。

```
imqobjmgr -i MyCmdFile
```

注 LDAP サーバーを使用して、管理対象オブジェクトを格納する場合、前述の例のように、接頭辞「cn=」が付いた検索名を割り当てることが重要です。検索名は、-l オプションを使用して指定します。ファイルシステムのオブジェクトストアを使用する場合は、この接頭辞を使用する必要はありません。

キューオブジェクトを追加する場合は、-t オプションに q を指定することを除いて、まったく同じコマンドを使用します。

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には、delete サブコマンドを使用します。オブジェクトの検索名、タイプ、および場所を指定する必要があります。

次のコマンドでは、検索名が cn=myTopic で、LDAP サーバーに格納される、トピックの管理対象オブジェクトが削除されます。

```
imqobjmgr delete
    -t t
    -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=fooUser, ou=People, o=imq"
    -j "java.naming.security.credentials=fooPasswd"
    -j "java.naming.security.authentication=simple"
```

管理対象オブジェクトの一覧表示

すべての管理対象オブジェクト、または特定タイプのすべての管理対象オブジェクトを一覧表示するには、list サブコマンドを使用します。次のサンプルコードでは、管理対象オブジェクトが LDAP サーバーに格納されることを前提としています。

次のコマンドでは、すべてのオブジェクトが一覧表示されます。

```
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=fooUser, ou=People, o=imq"
    -j "java.naming.security.credentials=fooPasswd"
    -j "java.naming.security.authentication=simple"
```

次のコマンドでは、queue タイプのすべてのオブジェクトが一覧表示されます。

```
imqobjmgr list
    -t q
    -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=fooUser, ou=People, o=imq"
    -j "java.naming.security.credentials=fooPasswd"
    -j "java.naming.security.authentication=simple"
```

単一オブジェクトの情報の取得

管理対象オブジェクトに関する情報を入手するには、query サブコマンドを使用します。オブジェクトの検索名、および管理対象オブジェクト (初期コンテキストおよび場所など) を含むオブジェクトストアの属性を指定する必要があります。

次の例では、query サブコマンドを使用して、cn= myTopic という検索名のオブジェクトに関する情報を表示します。

```
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=fooUser, ou=People, o=imq"
    -j "java.naming.security.credentials=fooPasswd"
  -j "java.naming.security.authentication=simple"
```

管理対象オブジェクトの更新

管理対象オブジェクトの属性を変更するには、update コマンドを使用します。検索名とオブジェクトの場所を指定する必要があります。-o オプションを使用して、属性値を変更します。

このコマンドでは、トピックのコネクションファクトリを表す管理対象オブジェクトの属性が変更されます。

```
imqobjmgr update
 -t tf
 -l "cn=MyTCF"
 -o imqReconnectAttempts=3
 -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=fooUser, ou=People, o=imq"
 -j "java.naming.security.credentials=fooPasswd"
 -j "java.naming.security.authentication=simple"
```

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コネクションを分散し、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、『Sun Java System Message Queue 3 2005Q1 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

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を指定します。このプロパティは、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については、336 ページの表 14-11 で詳しく説明します。クラスタ設定プロパティ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 コネクション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 コネクション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 コネクション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 プロパティと port プロパティ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の設定

ブローカのクラスタ設定プロパティは、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 コネクション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、クライアントコネクションを拒否します。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。

クラスタ設定ファイルを使用する代わりに、imqbrokerd コマンドの -cluster オプションを使用し、それぞれのブローカの起動時に、クラスタのブローカの完全なリストを指定できます。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
```

運用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コネクション

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 コネクションサービスを設定します。160 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 ベースのコネクション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 コネクション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、コマンド行でクラスタを最初に作成したか、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、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て、クラスタに残っているブローカのみを指定し、クラスタに残す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び読み込ませます。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び読み込みます。
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、201 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 プロパティと imq.cluster.masterbroker プロパティ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 という、3 つの監視インタフェースがあります。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 スポット検査に適しています 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 対象となる特定のデータを容易に選択できます わかり易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 データ分析のプログラム化が困難です 履歴レコードを作成しません 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 履歴レコードの作成	ブローカプロパティの設定が必要です。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 ローカル監視のみ 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 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 柔軟にデータを選択できません ブローカメトリックスのみ。送信先とコネクションサービスのメトリックスは含まれていません 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 対象となる特定のデータを容易に選択できます 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 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 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、363 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、関連ブローカのインスタンス名によって識別される、次のディレクトリに配置されます。付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照してください。

```
.../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、imq.log.timezone プロパティに関する情報を参照してください。これについては333 ページの表 14-10 で説明します。

ロガー設定の変更

ログ関連のプロパティについては、333 ページの表 14-10 で説明します。

➤ 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。

ロガープロパティを設定すると、手順は完了します。ロガープロパティの設定は、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 ファイルにロガープロパティ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、または任意のブローカプロパティを変更できます。

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

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とサイズに関連するロールオーバープロパティ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、205 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 ファイルで、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。

以下は、ログファイルに出力されたブローカメトリックスの例です。

```
[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。
- **コネクションサービスのメトリックス** : コネクションとコネクションスレッドのリソースに関する情報、および特定のコネクション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、それぞれのコネクションサービス、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、216 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 [-b *hostName:port*] [-m *metricType*] [-int *interval*] [-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対して、ブローカのメトリックスを表示します。
または	
metrics svc -n *serviceName* [-b *hostName:port*] [-m *metricType*] [-int *interval*] [-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* -n *destName* [-b *hostName:port*] [-m *metricType*] [-int *interval*] [-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:port*	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。 **ttl**: ブローカ、サービス、送信先との間の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、コネクションサービス、送信先との間の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)。 **cxn**: コネクション、仮想メモリーヒープ、およびスレッドを表示します (ブローカとコネクションサービスのみ)。 **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)。 **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コネクションサービスを指定します。デフォルトはありません。
-t *destTyp*	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。

   67 ページの「ブローカのインタラクティブな起動」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、コネクション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 (364 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
```

コネクションサービスのメトリックス

jms コネクション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 (366 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 (369 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 (369 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:port*`]`	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 (111 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:port*`]`	指定したコネクション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コネクション数に関する情報 (118 ページの「コネクションサービス情報の表示」を参照)。
または	
`query dst -t` *destType* `-n` *destName* `[-b` *hostName:port*`]`	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 (132 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、363 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` と設定する

   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* と設定する

   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` を設定する

   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` を設定する

   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。

   設定方法は次の、「セキュリティ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。

153 ページの「ユーザーの承認 : アクセス制御プロパティファイル」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、暗号化されたコネクション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。暗号化されたコネクションの使用方法については、160 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、363 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後も、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。

コネクションの負荷：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コネクションの数です。

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、コネクション、セッション、メッセージプロデューサを使用し、標準サイズの持続的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コネクション、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、コネクションスレッド使用率、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 (226 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、コネクション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、コネクションプロパティ、スレッドプールプロパティ、JVM メモリー制限、制限の動作、組み込み持続とプラグイン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後は、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。

- コネクション数
- 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、コネクション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。

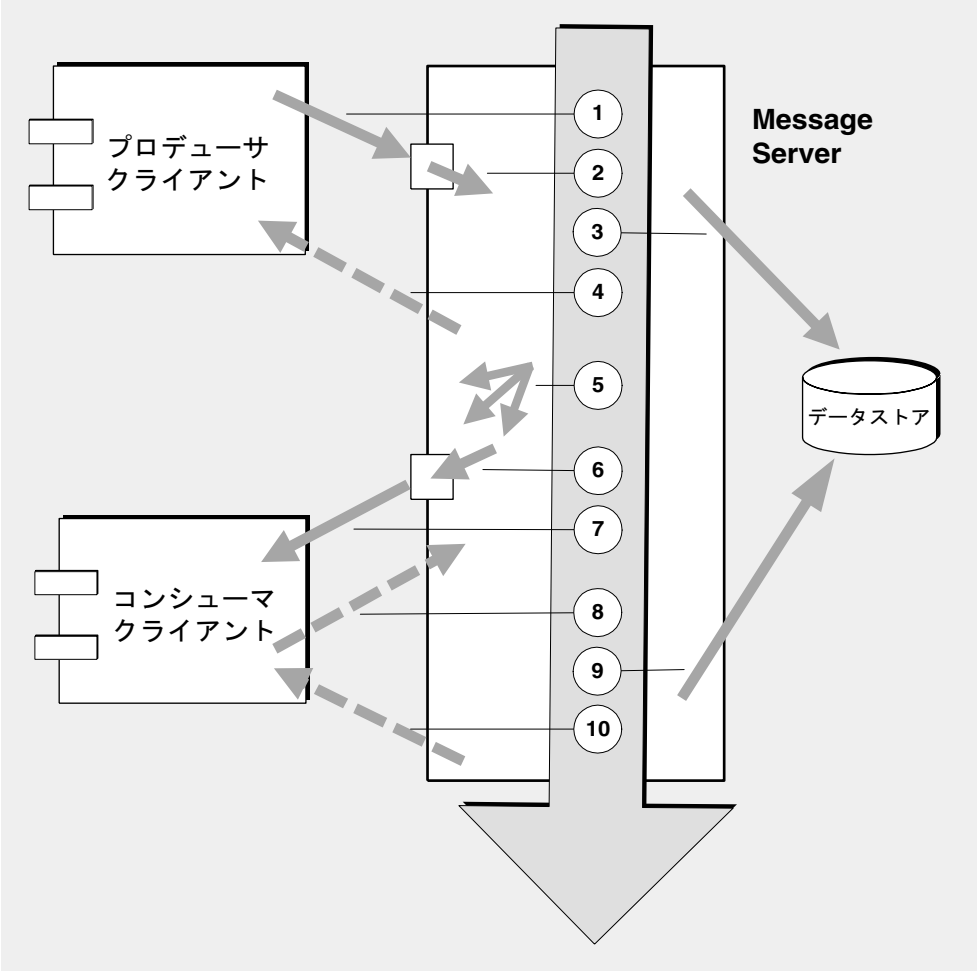
- 1 つの物理的送信先がそのほかの物理的送信先に比べ頻繁に使用されている場合は、その物理的送信先のメッセージメモリー制限をそのほかの送信先より高く設定したり、使用率に応じて制限の動作を調整したりできます。
- 必要なコネクション数が最大スレッドプールサイズによる許容値を大きく上回る場合は、スレッドプールサイズを増やすか、共有スレッドモデルを採用することができます。
- ピーク時のメッセージフローが平均フローより多い場合は、メモリーが不足したときに使用する制限の動作に影響することがあります。

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 低パフォーマンスシナリオ	高パフォーマンス 低信頼性シナリオ
配信モード	持続性メッセージ	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後、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)

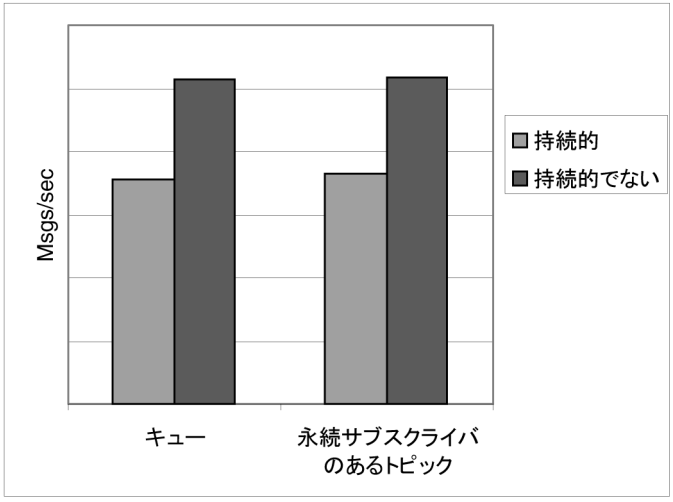
持続性メッセージはメッセージサーバーの障害時にもメッセージの配信を保証します。すべての対象のコンシューマが、メッセージを消費したことを通知するまで、ブローカはメッセージを持続ストアに格納します。

持続性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性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性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性メッセージごとに、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性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、持続性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性メッセージは、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います。
- 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後、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後は、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後でも、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性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。

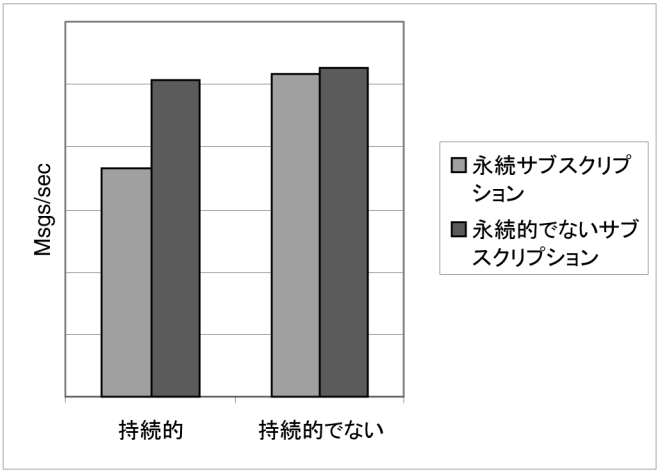
永続サブスクリプションでは、次の理由により、信頼性が高まりますが、スループットが遅くなります。

- 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性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性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コネクションは失われ、メッセージは配信されません。

図 11-3 では、10K バイトの持続性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性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性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

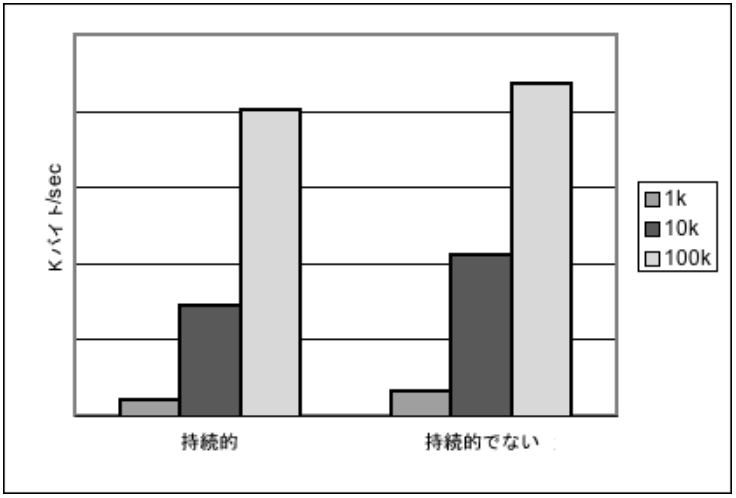
メッセージのサイズはパフォーマンスに影響します。プロデューシングクライアントからブローカへ、さらにブローカからコンシューミングクライアントへは、より多くのデータを渡す必要があり、持続性メッセージの場合はサイズの大きいメッセージを格納する必要があるからです。

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、持続性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、持続性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)
- コネクション
- ブローカの制限と動作
- メッセージサービスのアーキテクチャ
- 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コネクション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。

コネクション

クライアントとブローカ間のコネクションの数と速度は、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コネクションの制限

メッセージサーバーへのアクセスはすべて、コネクション経由で行われます。同時コネクション数の制限によって、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コネクションの数は、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、スレッドプールマネージャを使用します。このマネージャは、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使用するように設定できます (77 ページの「スレッドプールマネージャ」を参照)。

専用スレッドモデルは、各コネクションが専用のスレッドを持つため非常に高速ですが、コネクション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、コネクションごとに、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、コネクション数の制限はありませんが、多数のコネクション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、多くのコネクション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、75 ページの「コネクションサービス」で説明するコネクション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

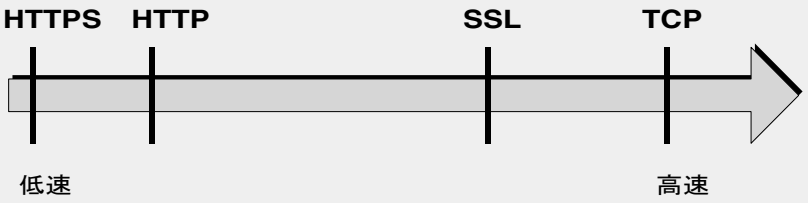


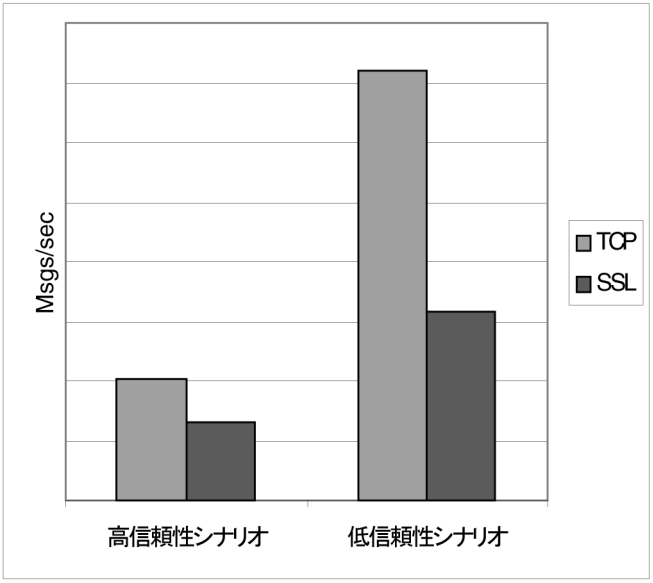
図 11-5 は、さまざまなプロトコルテクノロジのパフォーマンス特性を示しています。

- 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。持続性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 バイトの持続性メッセージを、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性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。

メッセージサービスのアーキテクチャ

Message Queue 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、クライアントのコネクション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、コネクション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、組み込み持続とプラグイン持続の両方をサポートしています。組み込み持続は、ファイルベースのデータストアです。プラグイン持続は、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。

組み込み持続はプラグイン持続より高速です。一方、JDBC 互換のデータベースシステムではアプリケーションに必要な冗長性、セキュリティ、および管理機能を提供できます。

組み込み持続の場合は、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、コネクションファクトリ属性値を使って設定可能で、一般的に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プロパティと動作を設定できます。

たとえば、Message Queue クライアントランタイムでは次の動作を設定できます。

- コネクションフロー測定 (`imqConnectionFlowCount`)。同じコネクションを経由する JMS メッセージと Message Queue 制御メッセージの両方に起因する輻輳を防ぐ上で役立ちます。
- コネクションフローの制限 (`imqConnectionFlowLimit`)。消費されるのを待機するために、クライアントランタイムへのコネクションを介して配信可能なメッセージ数の制限により、クライアントリソースが制限されるのを回避する上で役立ちます。
- コンシューマフローの制限 (`imqConsumerFlowLimit`)。複数のコンシューマがキュー配信を実行する状況で、コンシューマが送信するメッセージの数が不均衡にならないように、コンシューマ間のロードバランスを改善する上で役立ちます。また、コネクション上のあるコンシューマがそのコネクション上の別のコンシューマから過剰な負荷を受けないように防ぎます。このプロパティは、消費されるのを待機するために、コンシューマがクライアントランタイムへのコネクションを介して配信できるコンシューマあたりのメッセージ数を制限します。また、このプロパティはキュー送信先プロパティとして設定することもできます (`consumerFlowLimit`)。

これらの動作とそれを設定するために使用される属性の詳細は、246 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、メモリーを開放し、メッセージの流入を減少させるために、クライアントコネクション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がマシ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「コマンドのリファレンス」の「imqsvcadmin」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を使用して修正できます。

- imq.protocol.*protocol_type*.nodelay
- imq.protocol.*protocol_type*.inbufsz
- imq.protocol.*protocol_type*.outbufsz

TCP と SSL プロトコルの場合、これらのプロパティ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があります (243 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については、次の節で説明します。

nodelay

nodelay プロパティは、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) などの低速コネクション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サイズを満たしていない場合、プロトコルは、バッファが満たされるか、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。

- コネクションの作成
- プロデューサまたはコンシューマの作成
- 持続性メッセージの送信。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、コネクション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) 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、コネクション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

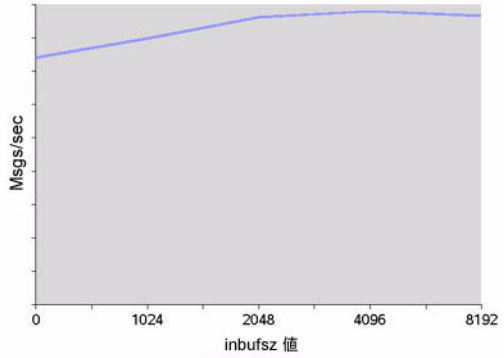
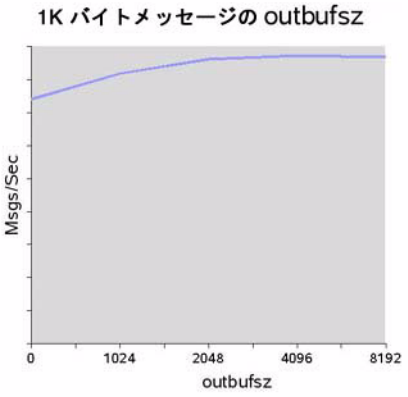


図 11-8 は、1K バイトのパケットの outbufsz を変更した結果を示しています。

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 コネクション (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とそれに対応する `imq.httpsjms.https.pullPeriod` プロパティは、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、各クライアントコネクションが 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、83 ページの「持続マネージャ」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、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と `imq.system.max_size` ブローカプロパティを設定して、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。

持続性のないメッセージも持続性メッセージと同様に例外を戻すようにするには、クライアントが使用しているコネクションファクトリオブジェクトに次のプロパティを設定します。

```
imqAckOnProduce = true
```

上記の設定では、クライアントは次のメッセージを送信する前に応答を待機するため、持続性のないメッセージをブローカへ送信するときのパフォーマンスを低下させることがあります。しかし、通常、ブローカでのメッセージの受信はシステムのボトルネックにならないため、この低下は許容範囲内です。

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。

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 (239 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、コネクション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。コネクション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 (JMS 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コネクション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、JMS 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に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 はコネクション全体の JMS メッセージのフローを測定します。

JMS メッセージは、imqConnectionFlowCount プロパティの指定に従い、設定した数のみが配信されるようにバッチされます。バッチが配信されると、JMS 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。JMS メッセージのそのほか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、持続性メッセージ、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する場合などには、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、コネクション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 JMS 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コネクションあたりのメッセージ数を制限できます。この制限は、コネクションを介して配信し、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。

コンシューマベースの制限

クライアントランタイムへ配信された JMS メッセージの数が、ど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ばあいにだけ、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がブローカに次のバッチを送信するように要求するまでの間、アプリケーションは、少なくともメッセージバッファ容量の 50% (imqConsumerFlowThreshold) つまり 500 メッセージを消費できます。

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、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限り、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、コネクション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。

コネクションベースの制限

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、コネクション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。

コネクションレベルのフロー制御は、コネクション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 を超えると、合計数がコネクションの制限を下回るまで、コネクション経由のメッセージの配信は停止します。imqConnectionFlowLimit は、imqConnectionFlowLimitEnabled プロパティを 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、またはより多くのコネクション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
```

クライアントがコネクションを確立できない

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コネクションを確立できない。
- クライアントが障害の生じたコネクション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コネクションを閉じていないため、コネクション数がリソース制限を超えてしまった
- 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
- コネクションサービスが非アクティブであるか停止される
- 必要なコネクション数に対して使用可能なスレッドが少なすぎる
- Solaris または Linux オペレーティングシステム上で必要なコネクション数に対してファイル記述子が少なすぎる
- TCP バックログにより、確立可能な新しい同時コネクション要求の数が制限される
- オペレーティングシステムによって同時コネクションの数が制限される
- ユーザーの認証に失敗するか権限が与えられない

クライアントアプリケーションがコネクションを閉じていないため、コネクション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

ブローカへのコネクションをすべて一覧表示します。

```
imqcmd list cxn
```

出力にはすべてのコネクションと各コネクションの確立元のホストが一覧表示されます。異常な数のコネクション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コネクション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。

コネクションサービスが非アクティブであるか停止される

この問題の原因を確認するには

すべてのコネクションサービスのステータスを確認します。

```
imqcmd list svc
```

コネクション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コネクションを確立できません。

問題を解決するには

- コネクション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、160 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。
- コネクション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 (120 ページの「コネクションサービスの停止および再開」を参照)。

必要なコネクション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、コネクションサービスのコネクション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
```

コネクションごとに 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 (77 ページの「スレッドプールマネージャ」を参照)。

問題を解決するには

- 専用のスレッドプールモデル (imq.*service_name*. threadpool_model=dedicated) を使用している場合は、コネクション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、コネクション数を増やすには、スレッドプール (imq.*service_name*.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.*service_name*. threadpool_model=shared) を使用している場合は、コネクションの最大数は次の 2 つのプロパティの結果の半分です。それは、コネクション監視制限 (imq.*service_name*.connectionMonitor_limit) とスレッドの最大数 (imq.*service_name*.max_threads) です。そのため、コネクション数を増やすには、スレッドプールのサイズを拡大するか、コネクション監視制限の値を大きくします。
- 最終的に、サポート可能なコネクションの数またはコネクション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コネクションを分散します。

Solaris または Linux オペレーティングシステム上で必要なコネクション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、66 ページの「ファイル記述子制限を設定する (Solaris または Linux)」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

マニュアルの ulimit 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コネクション要求の数が制限される

TCP バックログが、同時コネクション要求の数を制限します。ポートマッパーが追加要求を拒否しなければ、同時コネクション要求はシステムバックログ (imq.portmapper.backlog) に格納されます。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、非常に多数の同時コネクション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、その他のコネクションを拒否している期間にコネクションを受け入れているかどうかを確認します。次に、拒否されたコネクション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 バックログによるコネクション拒否をログしないからです。

正常コネクションがログされ、コネクション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コネクション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。
- クライアントがコネクションを閉じずに、多くのコネクション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コネクションの数が制限される

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コネクションの数を制限します。

この問題の原因を確認するには

`imqcmd query svc` を使用して、コネクション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コネクションを確立できるが、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

- より多くのコネクション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コネクションを分散します。

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、またはユーザーがコネクション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ファイルで、コネクション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 (148 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ファイルを編集し、コネクションサービスへのアクセス許可を与えます (157 ページの「コネクションサービスのアクセス制御」を参照)。

コネクション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コネクション数は、250 ページの「クライアントがコネクションを確立できない」で説明したように制限されているわけではなく、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。

- ネットワークコネクションまたは WAN が遅すぎる
- コネクションサービスプロトコルが、TCP に比べて本質的に低速
- コネクション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
- コネクション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

ネットワークコネクションまたは 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

コネクションが遅すぎる場合は、ネットワークリンクをアップグレードします。

コネクションサービスプロトコルが、TCP に比べて本質的に低速

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 (236 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、241 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。

コネクション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

241 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。

コネクション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

コネクション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、225 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。

- 258 ページの「メッセージの生成が遅れるまたは低速である」
- 261 ページの「メッセージがバックログされる」
- 265 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 (132 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 (133 ページの「物理的送信先のプロパティの更新」を参照)。

アクセス制御プロパティ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を変更します (158 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。

- 持続性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。
- 持続性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、処理速度が低下したメッセージプロデューサが応答する
- ブローカが持続性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、処理速度が低下したメッセージプロデューサが応答する

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: ブローカが持続性メッセージの受信を即時に通知しないため、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性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の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がメモリーリソースを超えないように注意深く変更します。

  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、244 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 (339 ページの表 15-1 を参照)。

ブローカが持続性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、または持続性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

- 組み込み持続の場合は、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、プラグイン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

低速なコネクションまたは、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、ブローカが持続性メッセージの受信を通知するまでに、コネクションファクトリの 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 コネクションファクトリ属性値を変更します (177 ページの「コネクションファクトリの属性」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 (240 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パージします (122 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 (339 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パージします (135 ページの「物理的送信先のパージ」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` コネクションファクトリ属性を設定することで、コネクション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 (349 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コネクションの確立に失敗した。
- 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 (245 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 (339 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パージします (135 ページの「物理的送信先のパージ」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 コネクションファクトリ属性を設定することで、コネクション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 (349 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 (245 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コネクションを共有します。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。
- コンシューマとコネクションのフロー制御パラメータを調整します (246 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 (240 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コネクションでのメッセージ配信の起動に失敗した

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

270 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271 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。

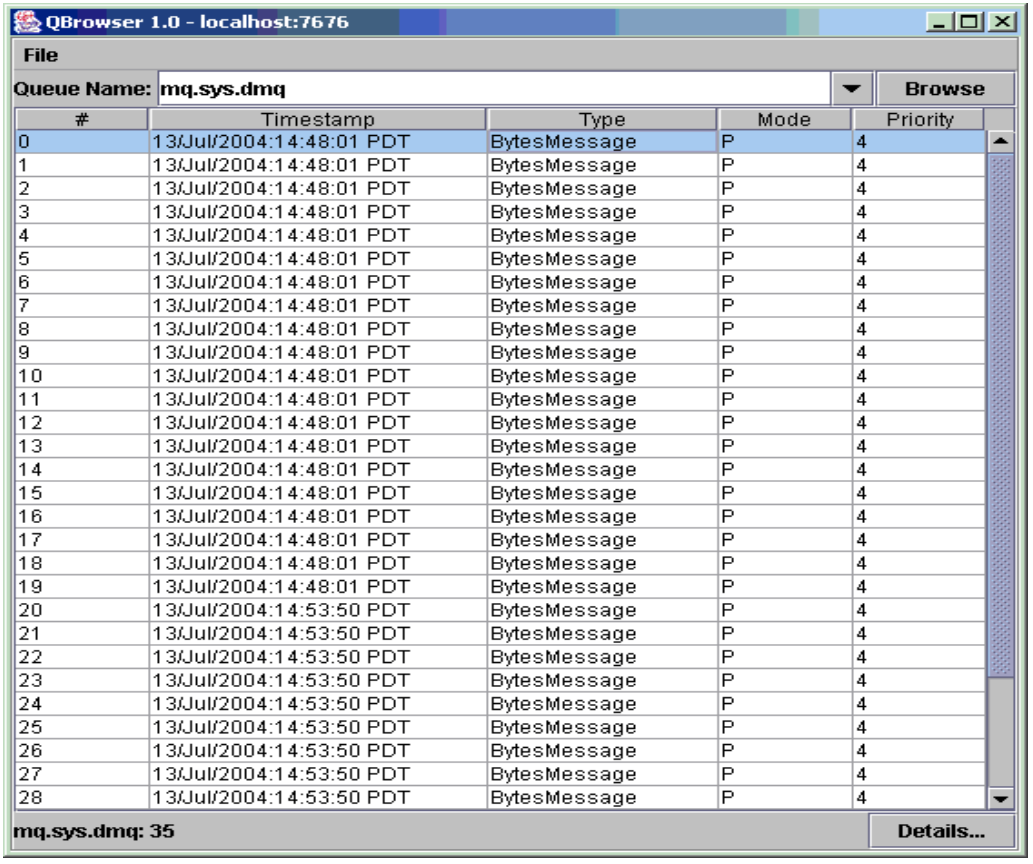
QBrowser デモアプリケーションを使用し、DMQ の内容を調べます。QBrowser デモアプリケーションの場所は、オペレーティングシステムによって異なります。場所については、付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照し、アプリケーション例と場所の表を調べてください。

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。

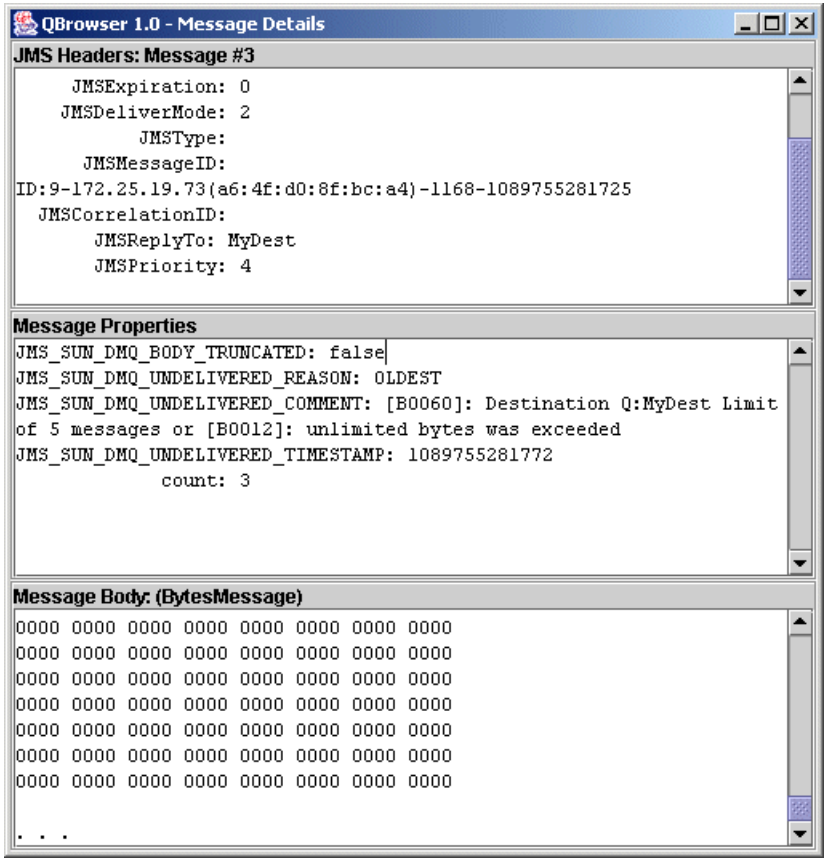
図 12-1　QBrowser のウィンドウ



#	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

65 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コネクションでのメッセージ配信の起動に失敗した

クライアントコードがコネクションを確立し、そのコネクション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コネクション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

コネクション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後で、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「オペレーティングシステムごとの 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。268 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。269 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

65 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、次のように操作します。

```
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、次のうちいずれかが true になります。

- コンシューマのコネクション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。

JMS_SUN_DMQ_UNDELIVERED_REASON 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

273 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後で、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パージします。
- 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

273 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

コマンドのリファレンス

この章では、一般的なコマンド行構文について説明し、それぞれの Message Queue コマンドの参照情報を提供します。この章では、次の節について説明します。

コマンド行の構文

Message Queue のコマンド行ユーティリティはシェルコマンドです。ユーティリティ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。したがって、ユーティリティ自体を起動または終了するコマンドはなく、そのようなコマンドは必要ありません。

コマンド行ユーティリティはすべて、次のコマンド構文を共有します。

Utility_Name [*subcommand*] [*argument*] [[*-option_name* [*-option_argument*]]...]

Utility_Name は、`imqcmd`、`imqobjmgr`、`imqusermgr` などの Message Queue ユーティリティの名前を指定します。

コマンド入力のルール

コマンド入力には、次のように一般的なルールがあります。

- ユーティリティが両方のタイプの引数を受け入れる場合は、サブコマンドと引数の後にオプションを指定します。
- オプションの値にスペースが含まれる場合は、値全体を引用符で囲みます。通常、属性と値の組み合わせは引用符で囲んでおきます。
- コマンド行で -v (バージョン) オプション、または -h/-H (ヘルプ) オプションを指定する場合、そのコマンド行ではほかのオプションを実行できません。
- サブコマンド、引数、オプション、オプションの引数はスペースで区切ります。

コマンド行の例

次に、サブコマンド句のないコマンド行の例を示します。このコマンドでは、デフォルトのブローカが起動します。

```
imqbrokerd
```

次のコマンドは複雑です。このコマンドでは、タイプが queue で名前が myQueue である送信先が破棄されます。認証はユーザー admin に基づいて実行され、ユーザーのパスワードはコマンドによって要求されます。-f オプションは確認がないことを指定し、-s オプションは、コマンドがサイレントモードで実行されることを指定します。

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

共通のコマンドオプション

Message Queue の管理ユーティリティ全体に共通するオプションを表 13-1 に示します。コマンド行のサブオプションの後ろにこのオプションを指定してください。オプションは、任意の順序で入力できます。

表 13-1　共通の Message Queue コマンド行オプション

オプション	説明
-h	特定のユーティリティの使用方法に関するヘルプを表示します。
-H	属性リストや例を含めた詳細な使用方法に関するヘルプを表示します (imqcmd と imqobjmgr だけでサポート)。
-s	サイレントモードに切り替えます。出力は表示されません。imqbrokerd に -silent として指定します。

表 13-1　共通の Message Queue コマンド行オプション (続き)

オプション	説明
-v	バージョン情報を表示します。
-f	ユーザー確認の要求なしで、特定のアクションを実行します。
-pre	imqobjmgr でのみ使用します。コマンドを実際に実行しないで、残りのコマンド行の効果を確認することができるプレビューモードに切り替わります。これは、デフォルトの属性値をチェックする場合に役に立ちます。
-javahome *path*	使用する代替の Java 2 互換のランタイムを指定します。デフォルトではシステム上のランタイムまたは Message Queue にバンドルされたランタイムを使用します。

imqbrokerd

imqbrokerd コマンド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。

構文

```
imqbrokerd [[ -Dproperty=value]...]
     [ -backup fileName]
     [ -cluster "[broker1] [[,broker2]...]"
     [ -dbuser userName]
     [ -force]
     [ -h|-help]
     [ -javahome path]
     [ -license licenseName]
     [ -loglevel level]
     [ -metrics interval]
     [ -name instanceName]
     [ -passfile fileName
     [ -port number]
     [ -remove instance]
     [ -reset data]
     [ -restore fileName]
     [ -shared]
     [ -silent|-s] [ -tty]
     [ -upgrade-store-nobackup]
     [ -version]
     [ -vmargs arg1 [[arg2]...]
```

コマンドオプション

表 13-2 に、imqbrokerd コマンドのオプションと、各オプションから影響を受ける設定プロパティがあれば、そのプロパティを示します。

表 13-2　imqbrokerd オプション

オプション	影響を受けるプロパティ	説明
-backup *fileName*	影響しません。	ブローカクラスタのみに適用されます。指定したファイルに、マスターブローカの設定変更レコードをバックアップします。201 ページの「設定変更レコードの管理」を参照してください。
-cluster"[*broker1*][[,*broker2*]...]" *broker* は次のどちらかになります。 • *host* • :*port* • *host*:*port*	接続先となるブローカのリストを含む imq.cluster.brokerlist をオーバーライドします。	ブローカクラスタのみに適用されます。特定のホストやポートにある全ブローカに接続します。このリストは、imq.cluster.brokerlist プロパティにあるリストとマージされます。*host* に値を指定しない場合は、localhost が使用されます。*port* に値を指定しない場合は、7676 が使用されます。このオプションを使用して複数のブローカに接続する方法については、195 ページの「ブローカクラスタを使用した作業」を参照してください。
-dbpassword *password*	特定のパスワードを含む imq.persist.jdbc.password をオーバーライドします。	プラグインの JDBC 準拠のデータストアに対するパスワードを指定します。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。代わりに次のうちいずれかを使用します。 • コマンド行からパスワードを省略し、コマンドでパスワードを要求。 • -passfile オプションを使用し、データベースパスワードを含むファイルを指定。
-dbuser *userName*	特定のユーザー名を含む imq.persist.jdbc.user をオーバーライドします。	プラグインの JDBC 準拠のデータベースに対するユーザー名を指定します。99 ページの「持続ストアの設定」を参照してください。

表 13-2　imqbrokerd オプション (続き)

オプション	影響を受けるプロパティ	説明
-D *property=value*	システムプロパティを設定します。インスタンス設定ファイル内の対応するプロパティ値をオーバーライドします。	指定したプロパティを指定した値に設定します。ブローカ設定プロパティについては、第 14 章「ブローカのプロパティのリファレンス」を参照してください。 **注 :** -D オプションを使用して設定するプロパティのスペルと形式は、十分に確認してください。誤った値を渡した場合、システムからの警告なしに Message Queue でプロパティを設定できなくなります。
-force	影響しません。	ユーザーの確認なしで、アクションを実行します。このオプションは、-remove instance オプションと -upgrade-store-nobackup オプションのみに適用されます。このオプションでは一般的に確認が要求されます。
-h|-help	影響しません。	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。
-javahome *path*	影響しません。	代替の Java 2 準拠の JDK へのパスを指定します。デフォルトでは、バンドルされたランタイムを使用します。
-ldappassword *password*	指定したパスワードを含む imq.user_repository.ldap.password をオーバーライドします。	LDAP ユーザーリポジトリにアクセスするためのパスワードを指定します。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。代わりに次のうちいずれかを使用します。 • コマンド行からパスワードを省略し、コマンドでパスワードを要求。 • -passfile オプションを使用し、LDAP パスワードを含むファイルを指定。

表 13-2　imqbrokerd オプション (続き)

オプション	影響を受けるプロパティ	説明
-license [*licenseName*]	影響しません。	使用している Message Queue 製品エディションのデフォルトと異なる場合、読み込みのためのライセンスを指定します。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全ライセンスが一覧表示されます。インストールされた Message Queue エディションによって、*licenseName* の値は、pe (Platform Edition - 基本機能の場合)、try (Platform Edition - 90 日間の企業向けトライアル機能の場合)、および unl (Enterprise Edition の場合) のどれかになります。
-loglevel *level*	指定したレベルを含む imq.broker.log.level をオーバーライドします。	ロギングレベルを、NONE、ERROR、WARNING、INFO のどれかに指定します。デフォルト値は INFO です。
-metrics *interval*	指定した秒数を含む imq.metrics.interval をオーバーライドします。	ブローカメトリックスが一定間隔 (秒単位) でロガーに書き込まれるように指定します。
-name *instanceName*	imq.instancename に特定の名前を設定します。	このブローカのインスタンス名を指定し、対応するインスタンス設定ファイルを使用します。ブローカ名を指定しない場合、インスタンス名は imqbroker に設定されます。 **注** : 同一ホスト上で複数のブローカのインスタンスを実行している場合、各インスタンスの名前は一意となる必要があります。
-passfile *fileName*	imq.passfile.enabled をオーバーライドして true に設定します。ファイルを含むパスを伴う imq.passfile.dirpath をオーバーライドします。ファイル名を含む imq.passfile.name をオーバーライドします。	imqcmd コマンドユーティリティ、SSL キーストア、LDAP ユーザーリポジトリ、JDBC 互換データベース、あるいはこの任意の組み合わせのパスワードの読み取り元となるファイル名を指定します。詳細については、169 ページの「passfile の使用」を参照してください。

表 13-2　imqbrokerd オプション (続き)

オプション	影響を受けるプロパティ	説明
-password *keypassword*	指定したパスワードを含む imq.keystore.password をオーバーライドします。	SSL 証明書キーストアのパスワードを指定します。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。代わりに次のうちいずれかを使用します。 • コマンド行からパスワードを省略し、コマンドでパスワードを要求。 • -passfile オプションを使用し、SSL 証明書キーストアパスワードを含むファイルを指定。
-port *number*	指定した番号を含む imq.portmapper.port をオーバーライドします。	ブローカのポートマッパーのポート番号を指定します。デフォルトでは 7676 に設定されています。同一サーバー上でブローカの 2 つのインスタンスを実行するには、各ブローカのポートマッパーが異なるポート番号となる必要があります。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インスタンスに接続します。
-remove instance	影響しません。	ブローカのインスタンスが削除されます。つまり、インスタンス設定ファイル、ログファイル、持続ストア、その他インスタンスに関連するファイルやディレクトリが削除されます。-force オプション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。

表 13-2　imqbrokerd オプション (続き)

オプション	影響を受けるプロパティ	説明
-reset store| messages| durables| props	影響しません。	指定された引数に応じて、データストアまたはデータストアのサブセットをリセットするか、あるいはブローカインスタンスのプロパティをリセットします。 データストアをリセットすると、持続メッセージ、永続サブスクリプション、トランザクションの情報など、すべての持続データが消去されます。このため、データが消去された状態で、ブローカインスタンスを起動できます。また、すべての持続メッセージだけを消去したり、すべての永続サブスクリプションだけを消去したりすることもできます。その後の再起動時に持続ストアをリセットしない場合は、-reset オプションを指定せずにブローカインスタンスを再起動します。 ブローカのプロパティをリセットすると、既存のインスタンス設定ファイル (config.properties) が空のファイルに置き換えられます。プロパティはすべてデフォルト値となります。
-restore *fileName*	影響しません。	ブローカクラスタのみに適用されます。マスターブローカの設定変更レコードを、指定したバックアップファイルに置き換えます。このファイルは、-backup オプションを使用して事前に作成しておく必要があります。201 ページの「設定変更レコードの管理」を参照してください。
-shared	imq.jms.threadpool_model をオーバーライドして shared に設定します。	共通のスレッドプールを使用して、jms コネクションサービスが実装されるように指定します。このスレッドプールでは、スレッドがコネクション間で共有され、ブローカインスタンスにサポートされるコネクション数が増加します。
-silent|-s	imq.log.console.output をオーバーライドして NONE に設定します。	コンソールへのロギングをオフにします。
-tty	imq.log.console.output をオーバーライドして ALL に設定します。	すべてのメッセージがコンソールに表示されるよう指定します。デフォルトでは、WARNING レベルまたは ERROR レベルのメッセージだけが表示されます。

表 13-2　imqbrokerd オプション (続き)

オプション	影響を受けるプロパティ	説明
-upgrade-store-nobackup	影響しません。	非互換のバージョンから Message Queue 3.5 または Message Queue 3.5 SP*x* へアップグレードすると、自動的に古いデータストアが削除されるように指定します。詳細については、『Message Queue インストールガイド』を参照してください。
-version	影響しません。	インストールされた製品のバージョン番号を表示します。
-vmargs *arg1* [[*arg2*] . . .]	影響しません。	Java VM に渡す引数を指定します。引数はスペースで区切ります。複数の引数を渡す場合や、引数にスペースが含まれる場合は、引用符で囲みます。たとえば、次のように指定します。 `imqbrokerd -tty -vmargs "-Xmx128m -Xincgc"` この引数は、コマンド行のみで渡すことができます。config.props ファイルには、関連設定プロパティがありません。

関連項目

imqbrokerd の使用法の詳細およびコマンド例については、67 ページの「ブローカのインタラクティブな起動」を参照してください。

imqcmd

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

構文

```
imqcmd subcommand argument [options]
imqcmd -h|H
imqcmd -v
```

サブコマンド

ヘルプや製品バージョンを表示しない場合は、imqcmd で常にサブコマンドを使用します。表 13-3 では、imqcmd サブコマンドを一覧表示して、サブコマンドの参照情報の場所を掲載します。

表 13-3　imqcmd のサブコマンド

サブコマンドと引数	説明	参照先
commit txn	トランザクションをコミットします。	298 ページの「トランザクション管理サブコマンド」
destroy dur	永続サブスクリプションを破棄します。	297 ページの「永続サブスクリプションのサブコマンド」
list cxn	ブローカのコネクションを一覧表示します。	297 ページの「コネクションのサブコマンド」
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。	297 ページの「永続サブスクリプションのサブコマンド」
list svc	ブローカのサービスを一覧表示します。	295 ページの「コネクションサービス管理サブコマンド」
list txn	ブローカのトランザクションを一覧表示します。	298 ページの「トランザクション管理サブコマンド」
metrics bkr	ブローカのメトリックスを表示します。	292 ページの「ブローカ管理サブコマンド」
metrics svc	サービスのメトリックスを表示します。	295 ページの「コネクションサービス管理サブコマンド」
pause bkr	ブローカのすべてのサービスを停止します。	292 ページの「ブローカ管理サブコマンド」

表 13-3　imqcmd のサブコマンド (続き)

サブコマンドと引数	説明	参照先
pause svc	ブローカのシングルサービスを停止します。	295 ページの「コネクションサービス管理サブコマンド」
purge dur	永続サブスクリプションを破棄しないで、永続サブスクリプションのすべてのメッセージをパージします。	297 ページの「永続サブスクリプションのサブコマンド」
query bkr	ブローカの情報をクエリーおよび表示します。	292 ページの「ブローカ管理サブコマンド」
query cxn	コネクションの情報をクエリーおよび表示します。	297 ページの「コネクションのサブコマンド」
query svc	サービスの情報をクエリーおよび表示します。	295 ページの「コネクションサービス管理サブコマンド」
query txn	トランザクションの情報をクエリーおよび表示します。	298 ページの「トランザクション管理サブコマンド」
reload cls	ブローカクラスタ設定の再読み込みを行います。	292 ページの「ブローカ管理サブコマンド」
restart bkr	現在実行中のブローカインスタンスを再起動します。	292 ページの「ブローカ管理サブコマンド」
resume bkr	ブローカのすべてのサービスを再開します。	292 ページの「ブローカ管理サブコマンド」
resume svc	1 つのサービスを再開します。	295 ページの「コネクションサービス管理サブコマンド」
rollback txn	トランザクションをロールバックします。	298 ページの「トランザクション管理サブコマンド」
shutdown bkr	ブローカインスタンスをシャットダウンします。	292 ページの「ブローカ管理サブコマンド」
update bkr	ブローカの属性を更新します。	292 ページの「ブローカ管理サブコマンド」
update svc	サービスの属性を更新します。	295 ページの「コネクションサービス管理サブコマンド」

Imqcmd コマンドユーティリティには、ブローカの物理的送信先で使用するサブコマンドもあります。送信先サブコマンドについては、第 6 章「物理的送信先の管理」で説明します。

次の節では、imqcmd サブコマンドを機能ごとに一覧にします。

ブローカ管理サブコマンド

表 13-4 は、ブローカを管理するのに使用する imqcmd のサブコマンドについてまとめたものです。ホスト名またはポートを指定しない場合は、デフォルトの localhost:7676 が使用されます。

表 13-4 ブローカを管理する imqcmd のサブコマンド

サブコマンドの構文	説明
metrics bkr [-b *hostName:port*] [-m *metricType*] [-int *interval*] [-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**: ブローカとの間の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**: ブローカとの間の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)。 **cxn**: コネクション、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)。
pause bkr [-b *hostName:port*]	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。113 ページの「ブローカの停止および再開」を参照してください。
query bkr -b *hostName:port*	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) のリストも表示されます。

表 13-4　ブローカを管理する imqcmd のサブコマンド (続き)

サブコマンドの構文	説明
`reload cls`	ブローカクラスタのみに適用されます。クラスタ内のすべてのブローカで、`imq.cluster.brokerlist` プロパティの再読み込みとクラスタ情報の更新を実行します。詳細については、198 ページの「クラスタへのブローカの追加」を参照してください。
`restart bkr [-b` *hostName:port*`]`	ブローカを起動したときに指定したオプションを使用し、デフォルトブローカ、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て再起動します。
`resume bkr [-b` *hostName:port*`]`	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。
`shutdown bkr [-b` *hostName:port*`]`	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。
`update bkr [-b` *hostName:port*`]` `-o` *attribute=value* `[-o` *attribute=value1*`] . . .`	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。

物理的送信先管理サブコマンド

表 13-5 は、物理的送信先の管理に使用する imqcmd のサブコマンドについてまとめたものです。ホスト名またはポートを指定しない場合は、デフォルトの `localhost:7676` が使用されます。

表 13-5　送信先の管理に使用される imqcmd のサブコマンド

サブコマンドの構文	説明
`compact dst [-t` *destType* `-n` *destName*`]`	特定のタイプと名前の送信先に対応する組み込みの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。圧縮する前に送信先を停止する必要があります。
`create dst -t` *destType* `-n` *destName* `[-o` *attribute=value*`] [-o` *attribute=value1*`] . . .`	特定のタイプの送信先を指定した名前と属性で作成します。送信先名には、英数字 (空白文字は含まない) だけを使用する必要があります。送信先名は、英字や "_" および "$" で開始できます。文字列「`mq.`」で開始することはできません。 マスターブローカが一時的に使用できないクラスタでは、この操作を実行できません。

表 13-5　送信先の管理に使用される imqcmd のサブコマンド (続き)

サブコマンドの構文	説明
`destroy dst -t` *destType* `-n` *destName*	特定のタイプと名前の送信先を破棄します。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は破棄できません。 マスターブローカが一時的に使用できないクラスタでは、この操作を実行できません。
`list dst [-t` *destType*`] [-tmp]`	指定したタイプのすべての送信先を一覧表示します。同様に、一時的送信先についても一覧表示するオプションがあります。 type 引数には次の 2 つの値があります。 *destType* = `q` (キュー) *destType* = `t` (トピック) タイプが指定されない場合は、すべてのタイプの送信先すべてが一覧表示されます。
`metrics dst -t` *destType* `-n` *destName* `[-m` *metricType*`] [-int` *interval*`] [-msp` *numSamples*`]`	特定のタイプと名前の送信先に関するメトリックス情報を表示します。 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`**: 送信先との間でやり取りされメモリーに常駐しているメッセージとパケット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`**: 送信先との間のメッセージとパケットのフローレート (秒単位) に関するメトリックスと、その他のレート情報を表示します。 **`con`**: コンシューマ関連のメトリックスを表示します。 **`dsk`**: ディスク使用量のメトリックスを表示します。 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)。

表 13-5　送信先の管理に使用される imqcmd のサブコマンド (続き)

サブコマンドの構文	説明
`pause dst [-t` *destType* `-n` *destName*`] [-pst` *pauseType*`]`	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ていない場合は、すべての送信先が停止されます。デフォルト値は ALL です。
`purge dst -t` *destType* `-n` *destName*	特定のタイプと名前の送信先のメッセージをパージします。
`query dst -t` *destType* `-n` *destName*	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。
`resume dst [-t` *destType* `-n` *destName*`]`	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。
`update dst -t` *destType* `-n` *destName* `-o` *attribute=value* `[-o` *attribute=value1*`] . . .`	特定の送信先で特定の属性値を更新します。 送信先がデッドメッセージキュー mq.sys.dmq でない場合、属性名は表 15-1 で説明した属性のものにすることができます。

コネクションサービス管理サブコマンド

表 13-6 は、コネクションサービスの管理に使用する imqcmd のサブコマンドについてまとめたものです。ホスト名またはポートを指定しない場合は、デフォルトの localhost:7676 が使用されます。

表 13-6　コネクションサービスを管理する imqcmd のサブコマンド

サブコマンドの構文	説明
`list svc [-b` *hostName:port*`]`	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コネクションサービスを一覧表示します。

表 13-6　コネクションサービスを管理する imqcmd のサブコマンド (続き)

サブコマンドの構文	説明

`metrics svc -n` *serviceName*
`[-b` *hostName:port*`]`
`[-m` *metricType*`]`
`[-int` *interval*`]`
`[-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。
	
	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`: 指定したコネクション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`: コネクション、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)。

`pause svc -n` *serviceName* `[-b` *hostName:port*`]`	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。admin サービスは停止できません。
`query svc -n` *serviceName* `[-b` *hostName:port*`]`	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。
`resume svc -n` *serviceName* `[-b` *hostName:port*`]`	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た特定のサービスを再開します。

`update svc -n` *serviceName*
`[-b` *hostName:port*`]`
`-o` *attribute=value*
`[-o` *attribute=value1*`] . . .`
	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、316 ページの「コネクションサービスのプロパティ」を参照してください。

コネクションのサブコマンド

表 13-7 に、コネクションに適用する imqcmd のサブコマンドを一覧表示します。ホスト名、またはポートを指定しない場合は、localhost、7676 を使用するものと仮定します。

表 13-7 コネクションサービスを管理する imqcmd のサブコマンド

サブコマンドの構文	説明
`list cxn [-svn` *serviceName*`] [-b` *hostName:port*`]`	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指定したサービス名のコネクション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、すべてのコネクションが一覧表示されます。
`query cxn -n` *connectionID* `[-b` *hostName:port*`]`	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指定したコネクションに関する情報を表示します。

永続サブスクリプションのサブコマンド

表 13-8 は、imqcmd の永続サブスクリプションのサブコマンドについてまとめたものです。ホスト名、またはポートを指定しない場合は、デフォルトの localhost:7676 が使用されます。

表 13-8 永続サブスクリプションを管理する imqcmd のサブコマンド

サブコマンド	説明
`list dur -d` *destName*	特定の送信先の永続サブスクリプションをすべて一覧表示します。
`destroy dur -n` *subscrName* `-c` *client_id*	特定の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。 マスターブローカが一時的に使用できないクラスタでは、この操作を実行できません。
`purge dur -n` *subscrName* `-c` *client_id*	特定の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パージします。

トランザクション管理サブコマンド

表 13-9 は、imqcmd のトランザクションのサブコマンドについてまとめたものです。ホスト名、またはポートを指定しない場合は、デフォルトの localhost:7676 が使用されます。

表 13-9　トランザクションを管理する imqcmd のサブコマンド

サブコマンド	説明
`list txn`	ブローカによって記録されたトランザクションがすべて一覧表示されます。
`query txn -n` *transaction_id*	特定のトランザクションに関する情報が表示されます。
`commit txn -n` *transaction_id*	特定のトランザクションをコミットします。
`rollback txn -n` *transaction_id*	特定のトランザクションをロールバックします。

コマンドオプション

表 13-10 に、imqcmd コマンドのオプションを一覧表示します。

表 13-10　imqcmd のオプション

オプション	説明
`-b` *hostName:port*	ブローカのホスト名とポート番号を指定します。デフォルト値は localhost:7676 です。 ポートだけを指定する場合：`-b :7878` 名前だけを指定する場合：`-b somehost`
`-c` *clientID*	トピックの永続サブスクライバの ID を指定します。122 ページの「永続サブスクリプションの管理」を参照してください。
`-d` *destinationName*	トピック名を指定します。`list dur` サブコマンドや `destroy dur` サブコマンドと一緒に使用します。122 ページの「永続サブスクリプションの管理」を参照してください。
`-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。 このオプションでは、ユーザー名とパスワードが必要ありません。

表 13-10　imqcmd のオプション (続き)

オプション	説明
-H	使用法のヘルプ、属性リスト、例を表示します。コマンド行ではそれ以外のことは実行されません。 このオプションでは、ユーザー名とパスワードが必要ありません。
-int *interval*	metrics bkr、metrics dst、および metrics svc サブコマンドがメトリックス出力を表示する間隔を秒単位で指定します。
-javahome *path*	使用する代替の Java 2 互換のランタイムを指定します。デフォルトではシステム上のランタイムまたは Message Queue にバンドルされたランタイムを使用します。
-m *metricType*	表示するメトリックス情報のタイプを指定します。このオプションは、metrics dst、metrics svc、または metrics bkr サブコマンドと同時に使用します。*metricType* の値は、メトリックスが送信先、サービス、ブローカのどれに対して生成されたかによって異なります。
-msp *numSamples*	metrics bkr、metrics dst、および metrics svc サブコマンドがメトリックス出力で表示するメトリックスのサンプル数を指定します。
-n *argumentName*	サブコマンドの引数の名前を指定します。これはサブコマンドに応じて、物理的な送信先、永続サブスクリプション、コネクション ID、またはトランザクション ID の名前になります。
-o *attribute=value*	属性の値を指定します。これはサブコマンドの引数に応じて、ブローカ (108 ページの「imqcmd コマンドユーティリティの使用」を参照)、サービス (116 ページの「コネクションサービスの管理」を参照)、または送信先 (122 ページの「永続サブスクリプションの管理」を参照) の属性になります。
-p *password*	管理者パスワードを指定します。このオプションは異論が多く、今後のリリースではサポートされません。代わりに次のうちいずれかを使用します。 • コマンド行からパスワードを省略し、コマンドでパスワードを要求。 • -passfile オプションを使用し、管理パスワードを含むファイルを指定。
-passfile *path*	コマンドを実行しているユーザーのパスワードを含むファイルのパスを指定します。詳細については、169 ページの「passfile の使用」を参照してください。

表 13-10　imqcmd のオプション (続き)

オプション	説明
-pst *pauseType*	送信先を停止したときに、プロデューサ、コンシューマ、または両方を停止させるどうかを指定します。122 ページの「永続サブスクリプションの管理」を参照してください。
-rtm *timeout*	imqcmd のサブコマンドの初期 (再試行) タイムアウト期間を秒単位で指定します。タイムアウトとは、imqcmd のサブコマンドがブローカへの要求を作成した後、待機している時間の長さです。それ以降、サブコマンドが再試行されるたびに、タイムアウト値として初期タイムアウト値の倍数が使用されます。デフォルト値 : 10
-rtr *numRetries*	imqcmd のサブコマンドが最初にタイムアウトになった後の再試行回数を指定します。デフォルト値 : 5
-s	サイレントモード。出力は表示されません。
-secure	ssladmin コネクションサービス (165 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」を参照) を使用して、セキュリティ保護されたブローカへの管理コネクションを指定します。このオプションを省略すると、コネクションは安全でなくなります。
-svn *serviceName*	どのコネクションのサービスを一覧表示するかを指定します。121 ページの「コネクション情報の入手」を参照してください。
-t *destType*	送信先のタイプを指定します。t (トピック)、または q (キュー) のどちらかとなります。122 ページの「永続サブスクリプションの管理」を参照してください。
-tmp	一時的送信先を表示します。293 ページの表 13-5 を参照してください。
-u *userName*	管理者名を指定します。この値を省略すると、管理者名の入力を要求されます。
-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。 このオプションでは、ユーザー名とパスワードが必要ありません。

関連項目

imqcmd の使用法の詳細およびコマンド例については、第 5 章「ブローカの管理」と第 6 章「物理的送信先の管理」を参照してください。

imqobjmgr

オブジェクトマネージャユーティリティ imqobjmgr では、Message Queue 管理対象オブジェクトの作成と管理を行います。

構文

```
imqobjmgr subcommand [options]
imqobjmgr -h|H
imqobjmgr -v
```

サブコマンド

オブジェクトマネージャユーティリティ (imqobjmgr) には、次の表 13-3 に示すようなサブコマンドが含まれています。

表 13-11 imqobjmgr 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	オブジェクトストアから管理対象オブジェクトを削除します。
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	指定された管理対象オブジェクトに関する情報を表示します。
update	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変更します。

コマンドオプション

表 13-12 に、imqobjmgr コマンドのオプションを示します。これらの使用方法については、タスクごとに説明した後続の節を参照してください。

表 13-12　imqobjmgr のオプション

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。
-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。
-H	使用法のヘルプ、属性リスト、例を表示します。コマンド行ではそれ以外のことは実行されません。
-i *fileName*	オブジェクトタイプ、検索名、オブジェクト属性、オブジェクトストア属性などのオプションを指定するサブコマンド句の一部またはすべてを含むコマンドファイルの名前を指定します。通常、オブジェクトストア属性などの反復の多い情報に使用されます。
-j *attribute=value*	JNDI オブジェクトストアを識別しアクセスするために必要な属性を指定します。174 ページの「オブジェクトストアについて」を参照してください。
-javahome *path*	使用する代替の Java 2 互換のランタイムを指定します。デフォルトではシステム上のランタイムまたは Message Queue にバンドルされたランタイムを使用します。
-l *lookupName*	管理対象オブジェクトの JNDI 検索名を指定します。この名前は、オブジェクトストアのコンテキスト内で一意であることが必要です。
-o *attribute=value*	管理対象オブジェクトの属性を指定します。343 ページの第 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。
-pre	プレビューモード。コマンドを実行せずに、実行される内容を示します。
-r *read-only_state*	管理対象オブジェクトが読み取り専用オブジェクトかどうかを指定します。値 true は、管理対象オブジェクトが読み取り専用オブジェクトであることを示します。クライアントは読み取り専用管理対象オブジェクトの属性は変更できません。デフォルトでは、読み取り専用の状態は false に設定されています。
-s	サイレントモード。出力は表示されません。

表 13-12　imqobjmgr のオプション (続き)

オプション	説明
-t *objectType*	Message Queue の管理対象オブジェクトのタイプを指定します。 q = queue t = topic cf = connection factory qf = queue connection factory tf = topic connection factory xcf = XA connection factory (分散トランザクション) xqf = XA queue connection factory (分散トランザクション) xtf = XA topic connection factory (分散トランザクション) e = SOAP endpoint (この管理対象オブジェクトタイプは、『Message Queue Developer's Guide for Java Clients』の説明のとおり、SOAP メッセージのサポートに使用される)
-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。

関連項目

imqobjmgr の詳細およびコマンド例については、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。

imqdbmgr

データベース管理ユーティリティ (imqdbmgr) では、持続に必要となるスキーマを設定します。imqdbmgr コマンドを使用して、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。

構文

```
imqdbmgr subcommand argument [options]
imqdbmgr -h|-help
imqdbmgr -v|-version
```

サブコマンド

データベース管理ユーティリティ (imqdbmgr) には、次の表 13-13 に示すようなサブコマンドが含まれています。

表 13-13　imqdbmgr のサブコマンド

サブコマンドと引数	説明
create all	新しいデータベースと Message Queue の持続ストアのスキーマを作成します。このコマンドは、組み込みデータベースシステムで使用し、プロパティの 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システムに、Message Queue の持続ストアのスキーマを作成します。このコマンドは、外部データベースシステムで使用します。
delete tbl	現在の持続ストアのデータベース内に存在する Message Queue のデータベーステーブルを削除します。
delete oldtbl	旧バージョンの持続ストアのデータベース内に存在するすべての Message Queue データベーステーブルを削除します。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後に使用されます。
recreate tbl	現在の持続ストアのデータベース内に存在する Message Queue のデータベーステーブルを削除した後、Message Queue の持続ストアのスキーマを作成し直します。
reset lck	その他のプロセスが持続ストアの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。

コマンドオプション

表 13-14 に imqdbmgr コマンドのオプションを一覧表示します。

表 13-14 imqdbmgr のオプション

オプション	説明
-D *property=value*	指定したプロパティを指定した値に設定します。
-b *instanceName*	ブローカインスタンス名を指定し、対応するインスタンス設定ファイルを使用します。
-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。
-p *password*	データベースのパスワードを指定します。このオプションは異論が多く、今後のリリースではサポートされません。代わりに次のうちいずれかを使用します。 • コマンド行からパスワードを省略し、コマンドでパスワードを要求。 • -passfile オプションを使用し、データベースパスワードを含むファイルを指定。
-passfile *path*	データベースパスワードを含むファイルのパスを指定します。詳細については、169 ページの「passfile の使用」を参照してください。
-u *name*	データベースの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。

関連項目

持続ストアの設定の詳細については、99 ページの「持続ストアの設定」を参照してください。

imqusermgr

ユーザーマネージャユーティリティ (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したりできます。imqusermgr 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。
- 管理者には、リポジトリに書き込むための適切なアクセス権が必要です。すなわち、Solaris と Linux では、root ユーザーまたは最初にブローカインスタンスを作成したユーザーでなければなりません。

構文

```
imqusermgr subcommand [options]
imqusermgr -h
imqusermgr -v
```

サブコマンド

表 13-15 には imqusermgr 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。

表 13-15 imqusermgr サブコマンド

サブコマンド	説明
add [-i *instanceName*] -u *userName* -p *passwd* [-g *group*] [-s]	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。
delete [-i *instanceName*] -u *userName* [-s] [-f]	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。
list [-i *instanceName*] [-u *userName*]	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。

表 13-15 imqusermgr サブコマンド (続き)

サブコマンド	説明
update [-i *instanceName*] -u *userName* -p *passwd* [-a *state*] [-s] [-f] update [-i *instanceName*] -u *userName* -a *state* [-p *passwd*] [-s] [-f]	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。

コマンドオプション

表 13-16 に imqusermgr コマンドのオプションを一覧表示します。

表 13-16 imqusermgr オプション

オプション	説明
-a *active_state*	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ユーザーリポジトリを指定します。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。

関連項目

単層型ファイルのユーザーリポジトリの設定と管理の詳細、および imqusermgr コマンドの例については、144 ページの「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。

imqsvcadmin

サービス管理ユーティリティ (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。

構文

```
imqsvcadmin subcommand [options]
imqsvcadmin -h
```

サブコマンド

Message Queue サービス管理ユーティリティ (imqsvcadmin) には、次の表 13-17 に示すようなサブコマンドが含まれています。

表 13-17 imqsvcadmin 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、スタートアップのオプションを指定します。
query	imqsvcadmin コマンドのスタートアップのオプションを表示します。これには、サービスを手動または自動のどちらで起動するのか、サービスの場所、Java ランタイムの場所、起動時にブローカに渡される引数の値などが含まれます。
remove	サービスを削除します。

コマンドオプション

表 13-18 に、imqsvcadmin コマンドのオプションを一覧表示します。

表 13-18 imqsvcadmin のオプション

オプション	説明
-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。

表 13-18　imqsvcadmin のオプション (続き)

オプション	説明
-javahome *path*	使用する代替の Java 2 互換のランタイムへのパスを指定します。デフォルトではシステム上のランタイムまたは Message Queue にバンドルされたランタイムを使用します。 例 : imqsvcadmin -install -javahome d:¥jdk1.4
-jrehome *path*	Java 2 互換の JRE へのパスを指定します。 例 : imqsvcadmin -install -jrehome d:¥jre¥1.4
-vmargs *arg* [[*arg*] . . .]	ブローカサービスを実行する Java VM に渡す追加の引数を指定します。これらの引数は、「Windows サービスコントロールパネルの開始パラメータ」フィールドで指定することも可能です。 例 : -vmargs "-Xms16m -Xmx128m"
-args *arg* [[*arg*] . . .]	ブローカサービスに渡す追加のコマンド行引数を指定します。imqbrokerd オプションについては、283 ページの「imqbrokerd」を参照してください。 これらの引数は、「Windows サービスコントロールパネルの開始パラメータ」フィールドで指定することも可能です。たとえば、次のように指定します。 imqsvcadmin -install -args "-passfile d:¥imqpassfile"

-javahome オプション、-vmargs オプション、および -args オプションを使用して指定した情報は、Windows のレジストリで次のパスの JREHome キー、JVMArgs キー、および ServiceArgs キーの下に保存されます。

```
HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet
    ¥Services¥iMQ_Broker¥Parameters
```

関連項目

Message Queue を Windows サービスとして実行することの詳細は、69 ページの「Windows での自動起動」を参照してください。

imqkeytool

imqusermgr コマンドでは、ブローカの自己署名型証明書を生成します。ssljms、ssladmin、cluster のコネクションサービスに対して、同じ証明書を使用できます。UNIX システムでは、superuser (root) アカウントから imqkeytool を実行しなければならないことがあります。

構文

```
imqkeytool -broker
```

関連項目

安全な接続の設定については、160 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。

第 14 章

ブローカのプロパティのリファレンス

この章では、ブローカの設定プロパティを一覧表示して説明します。最初の節では、すべてのブローカプロパティをアルファベット順に一覧表示して、詳しい説明を含む節への参照情報を掲載します。以降すべての節では、機能ごとにブローカプロパティを分類し、プロパティについて詳しく説明します。

この章では、次の節について説明します。

説明の表では、`imqcmd update bkr` コマンドを使用して設定できる場合、そのプロパティにマークが付いています。

プロパティのアルファベット順の一覧

表 14-1 は、ブローカインスタンスのプロパティをアルファベット順に並べた一覧です。プロパティのカテゴリの判断に使用し、カテゴリの説明を使用して、プロパティの詳細な説明をこの章の中で探してください。

表の左側の列では、それぞれのプロパティがアルファベット順に並んでいます。右側の列では、プロパティが属すカテゴリを掲載し、適切な節への相互参照を提供します。

表 14-1　ブローカインスタンス設定プロパティ

プロパティ名	参照先
imq.accesscontrol.enabled	328 ページの「セキュリティマネージャのプロパティ」
imq.accesscontrol.file.filename	328 ページの「セキュリティマネージャのプロパティ」
imq.audit.enabled	328 ページの「セキュリティマネージャのプロパティ」
imq.authentication.basic.user_repository	328 ページの「セキュリティマネージャのプロパティ」
imq.authentication.client.response.timeout	328 ページの「セキュリティマネージャのプロパティ」
imq.authentication.type	328 ページの「セキュリティマネージャのプロパティ」
imq.autocreate.destination.isLocalOnly	320 ページの「自動作成の設定プロパティ」
imq.autocreate.destination.limitBehavior	320 ページの「自動作成の設定プロパティ」
imq.autocreate.destination.maxBytesPerMsg	320 ページの「自動作成の設定プロパティ」
imq.autocreate.destination.maxNumMsgs	320 ページの「自動作成の設定プロパティ」
imq.autocreate.destination.maxNumProducers	320 ページの「自動作成の設定プロパティ」
imq.autocreate.destination.maxTotalMsgBytes	320 ページの「自動作成の設定プロパティ」
imq.autocreate.destination.useDMQ	320 ページの「自動作成の設定プロパティ」
imq.autocreate.queue	320 ページの「自動作成の設定プロパティ」
imq.autocreate.queue.consumerFlowLimit	320 ページの「自動作成の設定プロパティ」
imq.autocreate.queue.localDeliveryPreferred	320 ページの「自動作成の設定プロパティ」
imq.autocreate.queue.maxNumActiveConsumers	320 ページの「自動作成の設定プロパティ」
imq.autocreate.queue.maxNumBackupConsumers	320 ページの「自動作成の設定プロパティ」
imq.autocreate.topic	320 ページの「自動作成の設定プロパティ」
imq.autocreate.topic.consumerFlowLimit	320 ページの「自動作成の設定プロパティ」
imq.cluster.*property_name*	336 ページの「クラスタ設定プロパティ」
imq.destination.DMQ.truncateBody	319 ページの「メッセージルーターのプロパティ」
imq.destination.logDeadMsgs	333 ページの「監視とロギングのプロパティ」

表 14-1　ブローカインスタンス設定プロパティ (続き)

プロパティ名	参照先
imq.hostname	316 ページの「コネクションサービスのプロパティ」
imq.httpjms.http.*property_name*	401 ページの表 C-3
imq.httpsjms.https.*property_name*	401 ページの表 C-3
imq.imqcmd.password	328 ページの「セキュリティマネージャのプロパティ」
imq.keystore.*property_name*	328 ページの「セキュリティマネージャのプロパティ」
imq.log.console.output	333 ページの「監視サービスのプロパティ」
imq.log.console.stream	333 ページの「監視サービスのプロパティ」
imq.log.file.dirpath	333 ページの「監視サービスのプロパティ」
imq.log.file.filename	333 ページの「監視サービスのプロパティ」
imq.log.file.output	333 ページの「監視サービスのプロパティ」
imq.log.file.rolloverbytes	333 ページの「監視サービスのプロパティ」
imq.log.file.rolloversecs	333 ページの「監視サービスのプロパティ」
imq.log.level	333 ページの「監視サービスのプロパティ」
imq.log.syslog.facility	333 ページの「監視サービスのプロパティ」
imq.log.syslog.identity	333 ページの「監視サービスのプロパティ」
imq.log.syslog.logconsole	333 ページの「監視サービスのプロパティ」
imq.log.syslog.logpid	333 ページの「監視サービスのプロパティ」
imq.log.syslog.output	333 ページの「監視サービスのプロパティ」
imq.log.timezone	333 ページの「監視サービスのプロパティ」
imq.message.expiration.interval	319 ページの「メッセージルーターのプロパティ」
imq.message.max_size	319 ページの「メッセージルーターのプロパティ」
imq.metrics.enabled	319 ページの「メッセージルーターのプロパティ」
imq.metrics.interval	319 ページの「メッセージルーターのプロパティ」

表 14-1 ブローカインスタンス設定プロパティ (続き)

プロパティ名	参照先
imq.metrics.topic.enabled	319 ページの「メッセージルーターのプロパティ」
imq.metrics.topic.interval	319 ページの「メッセージルーターのプロパティ」
imq.metrics.topic.persist	319 ページの「メッセージルーターのプロパティ」
imq.metrics.topic.timetolive	333 ページの「監視サービスのプロパティ」
imq.passfile.dirpath	328 ページの「セキュリティマネージャのプロパティ」
imq.passfile.enabled	328 ページの「セキュリティマネージャのプロパティ」
imq.passfile.name	328 ページの「セキュリティマネージャのプロパティ」
imq.persist.file.destination.message.filepool.limit	323 ページの「ファイルベースの持続のプロパティ」
imq.persist.file.message.cleanup	319 ページの「メッセージルーターのプロパティ」
imq.persist.file.message.filepool.cleanratio	319 ページの「メッセージルーターのプロパティ」
imq.persist.file.message.max_record_size	319 ページの「メッセージルーターのプロパティ」
imq.persist.file.sync.enabled	323 ページの「ファイルベースの持続のプロパティ」
imq.persist.jdbc.*property_name*	323 ページの「持続マネージャのプロパティ」
imq.persist.store	319 ページの「メッセージルーターのプロパティ」
imq.ping.interval	316 ページの「コネクションサービスのプロパティ」
imq.portmapper.backlog	316 ページの「コネクションサービスのプロパティ」
imq.portmapper.hostname	316 ページの「コネクションサービスのプロパティ」
imq.portmapper.port	316 ページの「コネクションサービスのプロパティ」

表 14-1　ブローカインスタンス設定プロパティ (続き)

プロパティ名	参照先
imq.*resource_state*.count	319 ページの「メッセージルーターのプロパティ」
imq.*resource_state*.threshold	319 ページの「メッセージルーターのプロパティ」
imq.service.activelist	316 ページの「コネクションサービスのプロパティ」
imq.*service_name*.accesscontrol.enabled	328 ページの「セキュリティマネージャのプロパティ」
imq.*service_name*.accesscontrol.file.filename	328 ページの「セキュリティマネージャのプロパティ」
imq.*service_name*.authentication.type	328 ページの「セキュリティマネージャのプロパティ」
imq.*service_name*.max_threads	316 ページの「コネクションサービスのプロパティ」
imq.*service_name*.min_threads	316 ページの「コネクションサービスのプロパティ」
imq.*service_name*.*protocol_type*.hostname	316 ページの「コネクションサービスのプロパティ」
imq.*service_name*.*protocol_type*.port	316 ページの「コネクションサービスのプロパティ」
imq.*service_name*.threadpool_model	316 ページの「コネクションサービスのプロパティ」
imq.shared.connectionMonitor_limit	316 ページの「コネクションサービスのプロパティ」
imq.system.max_count	319 ページの「メッセージルーターのプロパティ」
imq.system.max_size	319 ページの「メッセージルーターのプロパティ」
imq.transaction.autorollback	319 ページの「メッセージルーターのプロパティ」
imq.user_repository.ldap.*property_name*	328 ページの「セキュリティマネージャのプロパティ」

コネクションサービスのプロパティ

表 14-2 には、コネクションサービスのプロパティを一覧表示します。最初の列はプロパティ名です。プロパティ名ごとに、第 2 列ではプロパティについて説明し、第 3 列ではデータ型を指定し、第 4 列ではデフォルト値を示します。

表 14-2 コネクションサービスのプロパティ

プロパティ名	説明	データ型	デフォルト値
imq.service.activelist	ブローカの起動時にアクティブになる、コンマで区切られた、名前別のコネクションサービスのリスト。サポートされるサービスは、jms、ssljms、httpjms、httpsjms、admin、ssladmin です。	list	jms、admin
imq.ping.interval	ブローカがコネクションを介して Message Queue クライアントランタイムへ継続的に ping を試行する間隔 (秒単位)。	整数	120
imq.hostname	1 台のコンピュータに、複数のネットワークインタフェースカードがある場合など、複数のホストを使用できる場合には、すべてのコネクションサービスがバインドするホストを、ホスト名、または IP アドレスで指定します。	文字列	使用可能な IP アドレスすべて
imq.portmapper.port[1]	ブローカのプライマリポート。ポートマッパーが常駐するポートです。ホストで複数のブローカインスタンスを実行する場合、各ブローカインスタンスに、固有のポートマッパーポートを割り当てる必要があります。	整数	7676
imq.portmapper.hostname	1 台のコンピュータに、複数のネットワークインタフェースカードがある場合など、複数のホストを使用できる場合には、ポートマッパーがバインドするホストを、ホスト名、または IP アドレスで指定します。	文字列	imq.hostname から継承。
imq.portmapper.backlog	ポートマッパーが、要求を拒否せずに、同時に処理可能な要求の最大数。ポートマッパーによる処理を待機する、オペレーティングシステムのバックログに格納可能な要求の数を設定します。	整数	50

表 14-2　コネクションサービスのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.*service_name*.*protocol_type*[2] .port	jms、ssljms、admin、および ssladmin のサービスの場合のみ、指定したコネクションサービスのポート番号を指定します。 httpjms と httpsjms コネクションサービスを設定する場合は、付録 C「HTTP/HTTPS のサポート」を参照してください。	整数	0 (ゼロ) ポートは、ポートマッパーによってダイナミックに割り当てられます。
imq.*service_name*.*protocol_type*[2].hostname	jms、ssljms、admin、および ssladmin のサービスの場合のみ、複数のホストを使用できる際 (1 台のコンピュータに、複数のネットワークインタフェースカードがある場合など) に、指定したコネクションサービスが接続するホスト (ホスト名、または IP アドレス) を指定します。	文字列	imq.hostname から継承。
imq.*service_name*.min_threads	指定したコネクション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数。 デフォルト値は、コネクションサービスによって異なります。	整数	10 (jms) 10 (ssljms) 10 (httpjms) 10 (httpsjms) 4 (admin) 4 (ssladmin)
imq.*service_name*.max_threads	指定したコネクション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数。新しいスレッドは、それ以上追加されなくなります。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。 デフォルト値は、コネクションサービスによって異なります。	整数	1000 (jms) 500 (ssljms) 500 (httpjms) 500 (httpsjms) 10 (admin) 10 (ssladmin)

表 14-2 コネクションサービスのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.*service_name*. threadpool_model	指定したコネクションサービスに対して、スレッドをコネクション専用 (dedicated) にするのか、あるいは必要に応じてコネクションで共有 (shared) するのかを指定する文字列。共有モデル (スレッドプール管理) の場合、ブローカがサポートするコネクションの数が増えますが、jms コネクションサービスと admin コネクションサービスでしか実装されません。 デフォルト値は、コネクションサービスによって異なります。	文字列	dedicated (jms) dedicated (ssljms) dedicated (httpjms) dedicated (httpsjms) dedicated (admin) dedicated (ssladmin)
imq.shared. connectionMonitor_limit	共有スレッドプールモデルの場合のみ、ディストリビュータスレッドで監視できるコネクションの最大数を指定します。システムでは、すべてのコネクションの監視に十分なディストリビュータスレッドが割り当てられます。この値を小さくすると、システムがアクティブなコネクションをスレッドに割り当てる速度が上がります。値を -1 に設定した場合は、無制限になります。 デフォルト値は、オペレーティングシステムによって異なります。	整数	512 (Solaris および Linux の場合) 64 (Windows)

1. このプロパティは imqcmd update bkr コマンドで使用できます。
2. *protocol_type* は表 4-2 に記載されています。

メッセージルーターのプロパティ

表 14-3 には、メッセージルーターのプロパティを一覧表示します。最初の列はプロパティ名です。プロパティ名ごとに、第 2 列ではプロパティについて説明し、第 3 列ではデータ型を指定し、第 4 列ではデフォルト値を示します。

メッセージサーバーが送信先を自動的に作成する機能を設定する自動作成プロパティについては、320 ページの表 14-4 で説明します。

表 14-3 メッセージルーターのプロパティ

プロパティ名	説明	データ型	デフォルト値
`imq.destination.DMQ.truncateBody`[1]	ブローカがメッセージ本文を削除してから、デッドメッセージキューにメッセージを保存するかどうかを指定するブール値。値を `true` にすると、ブローカはメッセージヘッダーとプロパティデータのみを保存します。値を `false` に設定すると、ブローカはヘッダーと本文を保存します。	ブール	`false`
`imq.message.expiration.interval`	期限切れメッセージの再利用が発生する、秒単位の間隔。	整数	`60`
`imq.system.max_count`[1]	ブローカが保持するメッセージの最大数。この値を超えるメッセージは拒否されます。値を -1 に設定した場合は無制限になります。	整数	`-1`
`imq.system.max_size`[1]	ブローカが保持するメッセージの最大のサイズ (バイト、K バイト、または M バイト単位)。この値を超えるメッセージは拒否されます。値を -1 に設定した場合は無制限になります。	バイト文字列[2]	`-1`
`imq.message.max_size`[1]	メッセージの本体の最大許容サイズ (バイト、K バイト、または M バイト単位)。このサイズを超えるメッセージは拒否されます。値を -1 に設定した場合は無制限になります。	バイト文字列[2]	`70m`
`imq.`*resource_state*`.threshold`	メモリーリソースがトリガーされるメモリーの利用率。リソースの状態を表す値は、`green`、`yellow`、`orange`、および `red` です。	整数 (パーセント)	`0` (green) `80` (yellow) `90`(orange) `98` (red)

表 14-3 メッセージルーターのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.*resource_state*.count	システムメモリーをチェックして新しいメモリーしきい値に達したかどうかを確認する前に、バッチで許容される受信メッセージの最大数。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。	整数 (パーセント)	5000 (green) 500 (yellow) 50 (orange) 0 (red)
imq.transaction.autorollback	PREPARED 状態の分散トランザクションをブローカの起動時に自動的にロールバックするかどうかを指定するブール値。false の場合は、imqcmd を使用して、手動でトランザクションをコミット、またはロールバックする必要があります (124 ページの「トランザクションの管理」を参照)。	ブール	false

1. このプロパティは imqcmd update bkr コマンドで使用できます。
2. データ型がバイト文字列となっている値は、バイト、K バイト、M バイト単位で表現できます。たとえば、次のように表現できます。1000 は 1000 バイト、7500b は 7500 バイト、77k は 77K バイト (77 x 1024 = 78848 バイト)、17m は 17M バイト (17 x 1024 x 1024 = 17825792 バイト) をそれぞれ表します。

表 14-4 には、ブローカが送信先を自動的に作成するときに使用するプロパティを掲載します。

表 14-4 自動作成の設定プロパティ

プロパティ名	説明	データ型	デフォルト値
imq.autocreate.destination.isLocalOnly	ブローカクラスタのみに適用されます。送信先がそのほかのブローカに複製されるかどうか、つまりメッセージの配信をローカルコンシューマ (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) だけに制限するかどうかを指定するブール値。いったん送信先が作成されると、この属性は更新できません。	ブール	false

表 14-4 自動作成の設定プロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.autocreate.destination.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する文字列。値は、次のどれかになります。 • FLOW_CONTROL - プロデューサを低速化します。 • REMOVE_OLDEST - もっとも古いメッセージを廃棄します。 • REMOVE_LOW_PRIORITY - メッセージの有効期限に従い優先度が最低のメッセージを破棄します。 • REJECT_NEWEST - 最新のメッセージを拒否します。プロデューシングクライアントは、持続メッセージのみの拒否の例外を受けます。持続性がないメッセージでこの制限動作を使用するには、imqAckOnProduce コネクションファクトリ属性を設定します。 このプロパティを REMOVE_OLDEST か REMOVE_LOW_PRIORITY に、imq.autocreate.destination.useDMQ を true に設定すると、ブローカは余分なメッセージをデッドメッセージキューに移動します。	文字列	REJECT NEWEST
imq.autocreate.destination.maxBytesPerMsg	自動作成された送信先で許容されるシングルメッセージの最大サイズ (バイト単位)。値を -1 にすると、メッセージサイズは無制限になります。	バイト文字列[2]	10k
imq.autocreate.destination.maxNumMsgs	自動作成された送信先で許容される、消費されないメッセージの最大数。値を -1 にすると、メッセージ数は無制限になります。	整数	100,000
imq.autocreate.destination.maxNumProducers	送信先で許容されるプロデューサの最大数。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。値を -1 にすると、プロデューサ数は無制限になります。	整数	100
imq.autocreate.destination.maxTotalMsgBytes	送信先で消費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 (バイト単位)。値を -1 にすると、メモリーは無制限になります。	バイト文字列[2]	10m
imq.autocreate.destination.useDMQ	ブローカが自動作成の送信先のデッドメッセージをデッドメッセージキューに移動するかどうかを指定するブール値。	ブール	true

表 14-4 自動作成の設定プロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.autocreate.queue[1]	ブローカでキューの送信先の自動作成を許可するかどうかを指定するブール値。	ブール	true
imq.autocreate.queue.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。ロードバランスされたキュー配信では、ロードバランスされる前に、最初にキューに入っていてアクティブコンシューマにルートされるメッセージの数となります。それぞれのコネクションで送信先のコンシューマに低い値を設定すると、この制限をオーバーライドできます。値を -1 に設定した場合は無制限になります。	整数	1000
imq.autocreate.queue.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だけ適用されます。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するブール値。自動作成された送信先をローカルだけの配信に制限することはできません (isLocalOnly = false)。	ブール	false
imq.autocreate.queue.maxNumActiveConsumers	自動作成された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できるコンシューマの最大数。値を -1 に設定した場合は無制限になります。	整数	1
imq.autocreate.queue.maxNumBackupConsumers	自動作成された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取って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。値を -1 に設定した場合は無制限になります。	整数	0 (ゼロ)
imq.autocreate.topic	ブローカでトピックの送信先の自動作成を許可するかどうかを指定するブール値。	ブール	true
imq.autocreate.topic.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。値を -1 に設定した場合は無制限になります。	整数	1000

1. imqcmd update bkr で使用できます。

2. データ型がバイト文字列となっている値は、バイト、K バイト、M バイト単位で表現できます。たとえば、次のように表現できます。1000 は 1000 バイト、7500b は 7500 バイト、77k は 77K バイト (77 x 1024 = 78848 バイト)、17m は 17M バイト (17 x 1024 x 1024 = 17825792 バイト) をそれぞれ表します。

持続マネージャのプロパティ

ブローカの持続性機能を設定するには、imq.persist.store に値を指定するか、デフォルト値を受け入れます。

表 14-5　必須持続マネージャプロパティ

プロパティ名	説明	データ型	デフォルト値
imq.persist.store	ブローカが、組み込みのファイルベースの持続を使用するか、プラグインの JDBC 互換の持続を使用するかを指定する文字列。 値は、file か jdbc にする必要があります。	文字列	file

ファイルベースの持続および JDBC ベースの持続をサポートするプロパティについては、次の節で説明します。

ファイルベースの持続

表 14-6 では、ファイルベースの持続をサポートするプロパティについて説明します。最初の列はプロパティ名です。プロパティ名ごとに、第 2 列ではプロパティについて説明し、第 3 列ではデータ型を指定し、第 4 列ではデフォルト値を示します。

表 14-6　ファイルベースの持続のプロパティ

プロパティ名	説明	データ型	デフォルト値
imq.persist.file.sync.enabled	持続操作で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を指定するブール値。このプロパティを true に設定した場合、システムクラッシュによるデータ損失は回避されますが、持続操作のパフォーマンスに負荷がかかります。 Sun クラスタと Sun クラスタデータサービスを Message Queue で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を true に設定してください。	ブール	false
imq.persist.file.message.max_record_size	組み込みのファイルベースの持続では、個別のファイルに格納されるメッセージではなく、メッセージストレージファイルに追加されるメッセージの最大サイズ。	バイト文字列[1]	1m

表 14-6　ファイルベースの持続のプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.persist.file.destination.message.filepool.limit	組み込みのファイルベースの持続では、送信先のファイルプールで再利用できる空きファイルの最大数。この値が大きいほど、ブローカが持続データを処理する速度が速くなります。この値を超える空きファイルは削除されます。この制限を超えると、ブローカは必要に応じて追加ファイルを作成および削除します。	整数	100
imq.persist.file.message.filepool.cleanratio	組み込みのファイルベースの持続では、クリーン状態 (サイズを 0 にする) で保持される送信先のファイルプールの空きファイルの割合。この値が大きいほど、作業中にファイルを削除するのに必要なオーバーヘッドが増えますが、ファイルプールに必要なディスク容量は小さくなります。	整数	0 (ゼロ)
imq.persist.file.message.cleanup	組み込みのファイルベースの持続では、ブローカのシャットダウン時に、送信先のファイルプール内の空きファイルを削除するかどうかを指定するブール値。値を false に設定すると、ブローカのシャットダウンが速まりますが、ファイルを格納するためのディスク容量がさらに必要になります。	ブール	false

1. データ型がバイト文字列となっている値は、バイト、K バイト、M バイト単位で表現できます。たとえば、1000 は 1000 バイト、7500b は 7500 バイト、77k は 77K バイト (77 x 1024 = 78848 バイト)、17m は 17M バイト (17 x 1024 x 1024 = 17825792 バイト) をそれぞれ表します。

JDBC ベースの持続

表 14-7 では、JDBC ベースの持続をサポートするプロパティについて説明します。この表では、プロパティを一覧表示して説明し、PointBase 製品での設定方法の例を挙げます。

表 14-7　JDBC ベースの持続のプロパティ

プロパティ名	説明	例
imq.persist.store	ファイルベースまたは JDBC ベースのデータストアを指定する文字列。	jdbc

表 14-7　JDBC ベースの持続のプロパティ (続き)

プロパティ名	説明	例
imq.persist.jdbc.brokerid	(任意指定) 複数のブローカインスタンスが、持続データストアとして、同じデータベースを使用する場合、データベーステーブル名を一意にするために、データベーステーブル名に追加されるブローカインスタンス識別子。 この属性は、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、一般的に必要ありません。 識別子には、英数字を使用し、データベースで許可されているテーブル名の最大数 12 を超えないようにする必要があります。	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 JDBC ドライバの Java クラス名。	com.pointbase.jdbc. jdbcUniversalDriver
imq.persist.jdbc.opendburl	既存データベースへのコネクションを開くためのデータベース URL。	jdbc:pointbase:embedded: *dbName*;database.home= .../instances/*instanceName*/ dbstore
imq.persist.jdbc.createdburl	(任意指定) データベースを作成するコネクションを開くためのデータベース URL。 この属性は、imqdbmgr を使用してデータベースを作成する場合にのみ指定します。	jdbc:pointbase: embedded:*dbName*;new, database.home= /instances/*instanceName*/ dbstore
imq.persist.jdbc.closedburl	(任意指定) ブローカをシャットダウンする場合に、現在のデータベースコネクションをシャットダウンするためのデータベース URL。	PointBase の場合は不要です
imq.persist.jdbc.user	(任意指定) 必要に応じて、データベースコネクションを開くときに使用するユーザー名。セキュリティ上の理由から、代わりに、次のコマンド行オプションを使用して値を指定できます。imqbrokerd -dbuser および imqdbmgr -u	

表 14-7　JDBC ベースの持続のプロパティ (続き)

プロパティ名	説明	例
imq.persist.jdbc.needpassword	(任意指定) データベースでブローカのアクセスにパスワードを必要とするかどうかを指定するブール値。値を true にした場合は、パスワードが必要になります。 このオプションを設定した場合は、-passfile オプションを使用してパスワードを含むファイルを指定しないかぎり、imqbrokerd コマンドと imqdbmgr コマンドでパスワードが要求されます。	
imq.persist.jdbc.password	(任意指定) 必要に応じて、データベースコネクションを開くときに使用するパスワード。 passfile 内だけでこのプロパティを指定してください。	
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} (STOREVERSION INTEGER NOT NULL, 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} (RECORDTIME BIGINT NOT NULL, 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} (DID VARCHAR(100) NOT NULL, DEST BLOB(10k), primary key(DID))
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} (CUID BIGINT NOT NULL, INTEREST BLOB(10k), primary key(CUID))

表 14-7　JDBC ベースの持続のプロパティ (続き)

プロパティ名	説明	例
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。 MSG 列のデフォルトの最大長は、1M バイト (1m) です。メッセージがこの長さより長くなると予想される場合は、それ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は、変更を加えるためにテーブルを作成し直す必要があります。	CREATE TABLE ${name} (MID VARCHAR(100) NOT NULL, DID VARCHAR(100), MSGSIZE BIGINT, MSG BLOB(1m), primary key(MID))
imq.persist.jdbc.table.IMQPROPS35	プロパティテーブルを作成するための SQL コマンド。	CREATE TABLE ${name} (PROPNAME VARCHAR(100) NOT NULL, PROPVALUE BLOB(10k), primary key(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} (MID VARCHAR(100) NOT NULL, CUID BIGINT, DID VARCHAR(100), STATE INTEGER, primary key(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} (TUID BIGINT NOT NULL, STATE INTEGER, TSTATEOBJ BLOB(10K), primary key(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} (TUID BIGINT NOT NULL, TXNACK BLOB(10k))

セキュリティマネージャのプロパティ

表 14-8 では、セキュリティマネージャのプロパティについて説明します。最初の列はプロパティ名です。プロパティ名ごとに、第 2 列ではプロパティについて説明し、第 3 列ではデータ型を指定し、第 4 列ではデフォルト値を示します。

SSL を使用している場合は、次の表 14-9 に一覧表示するキーストア設定プロパティを参照してください。

表 14-8 セキュリティマネージャのプロパティ

プロパティ名	説明	データ型	デフォルト値
imq.accesscontrol.enabled	ブローカによってサポートされるすべてのコネクションサービスのアクセス制御を設定するかどうかを指定するブール値。アクセス制御プロパティファイルに指定されているように、認証されたユーザーが、コネクションサービスを使用するためのアクセス権、あるいは特定の送信先に対して特定の Message Queue 操作を実行するためのアクセス権を保持していることをシステムでチェックするかどうかを指定します。	ブール	true
imq.accesscontrol.file.filename	ブローカインスタンスでサポートされるすべてのコネクションサービスのアクセス制御プロパティファイルの名前。ファイル名には、アクセス制御ディレクトリへの相対ファイルパスを指定します (付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照)。	文字列	accesscontrol.properties
imq.audit.enabled	ブローカログファイルの監査ロギング (Enterprise Edition のみ) を起動するかどうかを指定するブール値。	ブール	false
imq.authentication.basic.user_repository	認証に使用される (Base-64 コーディング用の) ユーザーリポジトリのタイプとして、ファイルベース (file)、または LDAP (ldap) のどちらかを指定する文字列。	文字列	file
imq.authentication.client.response.timeout	ブローカからの認証要求に対するクライアントの応答をシステムが待機する間隔 (秒単位)。	整数	180
imq.authentication.type	パスワードを Base-64 コーディング (basic)、または MD5 ダイジェスト (digest) のどちらで送信するのかを指定する文字列。ブローカでサポートされるすべてのコネクションサービスに対して、符号化を設定します。	文字列	digest

表 14-8　セキュリティマネージャのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.imqcmd.password	管理ユーザーのパスワード。imqcmd コマンドユーティリティでは、このパスワードが使用され、コマンドのユーザーが操作する前に認証されます。	文字列	なし
imq.keystore.*property_name*	SSL ベースのサービスの場合は、SSL キーストアに関係するセキュリティプロパティを指定する文字列。332 ページの表 14-9 を参照してください。	文字列	なし
imq.passfile.dirpath	パスファイルが配置されているディレクトリへのパス。オペレーティングシステムによって異なります。	文字列	付録 A を参照
imq.passfile.enabled	セキュリティ保護される通信用の (SSL、LDAP、JDBC™ の) ユーザーパスワードをパスファイルで指定するどうかを指定するブール値。	ブール	false
imq.passfile.name	パスファイル名。	文字列	passfile
imq.*service_name*.accesscontrol.enabled	指定したコネクションサービスにアクセス制御を設定し、ブローカ全体の設定をオーバーライドするかどうかを指定するブール値。アクセス制御プロパティファイルに指定されているように、認証されたユーザーが、指定したコネクションサービスを使用するためのアクセス権、あるいは特定の送信先に対して特定の Message Queue 操作を実行するためのアクセス権を保持していることをシステムでチェックするかどうかを指定します。	ブール	システム全体のプロパティ imq.accesscontrol.enabled から継承。
imq.*service_name*.accesscontrol.file.filename	ブローカインスタンスの指定したコネクションサービスに対する、アクセス制御プロパティファイルの名前。ファイル名には、アクセス制御ディレクトリへの相対ファイルパスを指定します (付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照)。 デフォルト値は、システム全体のプロパティ imq.accesscontrol.file.filename から継承されます。	文字列	説明を参照

表 14-8　セキュリティマネージャのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.*service_name*.authentication.type	パスワードを Base-64 コーディング (basic)、または MD5 ダイジェスト (digest) のどちらで送信するのかを指定する文字列。指定したコネクションサービスの符号化を設定して、ブローカ全体の設定をオーバーライドします。 デフォルト値は、システム全体のプロパティ imq.authentication.type から継承されます。	文字列	説明を参照
imq.user_repository.ldap.base	ユーザーエントリのためのディレクトリベース。	文字列	なし
imq.user_repository.ldap.gidattr	プロバイダ固有の属性識別子で、その値はグループ名。	文字列	なし
imq.user_repository.ldap.grpbase	グループエントリのためのディレクトリベース。	文字列	なし
imq.user_repository.ldap.grpfiltler	JNDI 検索フィルタ (論理式で表現される検索クエリー)。グループに検索フィルタを指定すると、ブローカが検索範囲を絞り込めるため検索効率が上がります。詳細は、次の場所にある JNDI チュートリアルを参照してください。 http://java.sun.com/products/jndi/tutorial このプロパティの設定は任意です。	文字列	なし
imq.user_repository.ldap.grpsearch	グループ検索を有効にするかどうかを指定するブール値。ユーザーをグループに関連付けるかどうかを決定するには、LDAP プロバイダから提供されているマニュアルを参照してください。 入れ子にされたグループは、Message Queue ではサポートされていないので注意してください。	ブール	false
imq.user_repository.ldap.memattr	グループエントリにある属性識別子。その値はグループメンバーの識別名です。	文字列	なし
imq.user_repository.ldap.password	ブローカが使用する識別名と関連付けられたパスワード。 passfile 内だけでこのプロパティを指定してください。 ディレクトリサーバーで匿名の検索が許可されている場合、パスワードは不要です。	文字列	なし

表 14-8 セキュリティマネージャのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.user_repository.ldap.principal	検索時にブローカがディレクトリサーバーにバインドするために使用する識別名。ディレクトリサーバーで匿名検索が可能な場合、このプロパティに値を設定する必要はありません。	文字列	なし
imq.user_repository.ldap.*property_name*	今後使用する予定	今後使用する予定	今後使用する予定
imq.user_repository.ldap.server	LDAP サーバーの場合、*host:port* の *host* にはディレクトリサーバーを実行しているホストの完全指定 DNS 名を指定し、*port* にはディレクトリサーバーが通信に使用しているポート番号を指定します。 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。 *host1:port1* `ldap://`*host2:port2* `ldap://`*host3:port3*`...` リスト内のエントリはスペースで区切ります。それぞれのフェイルオーバーサーバーアドレスが `ldap://` で始まることに注意してください。 SSL を使用し、プロパティ `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。アドレスに `"ldaps"` を指定する必要はありません。	文字列	なし
imq.user_repository.ldap.ssl.enabled	LDAP サーバーとの通信時にブローカが SSL プロトコルを使用するかどうかを指定するブール値。	ブール	`false`
imq.user_repository.ldap.timeout	検索の時間制限 (秒単位)。	整数	`280`
imq.user_repository.ldap.uidattr	プロバイダ固有の属性識別子。その値はユーザーを一意に識別します。たとえば、次のように指定します。`uid`、`cn`	文字列	なし

表 14-8　セキュリティマネージャのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.user_repository.ldap.usrfilter	JNDI 検索フィルタ (論理式で表現される検索クエリー)。ユーザーに検索フィルタを指定すると、ブローカが検索範囲を絞り込めるので、検索効率が上がります。詳細については、次の場所にある JNDI チュートリアルを参照してください。 http://java.sun.com/products/jndi/tutorial このプロパティの設定は任意です。	文字列	なし

Message Queue キーストアの設定可能プロパティを表 14-9 に示します。このプロパティは SSL と併用してください。

表 14-9　キーストアのプロパティ

プロパティ名	説明	データ型	デフォルト値
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。デフォルト値については、付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照してください。	文字列	なし
imq.keystore.file.name	SSL ベースサービスの場合は、キーストアファイルの名前。	文字列	keystore
imq.keystore.password	SSL ベースサービスの場合は、キーストアのパスワード。 passfile 内だけでこのプロパティを指定してください。	文字列	なし

監視とロギングのプロパティ

表 14-10 では、監視とロギングに関連するプロパティについて一覧表示します。最初の列はプロパティ名です。プロパティ名ごとに、第 2 列ではプロパティについて説明し、第 3 列ではデータ型を指定し、第 4 列ではデフォルト値を示します。

表 14-10 監視サービスのプロパティ

プロパティ名	説明	データ型	デフォルト値
imq.destination.logDeadMsgs[1]	ブローカが次のタイプのイベントをログするかどうかを指定するブール値。 • 送信先がいっぱいであるか、最大サイズまたは最大メッセージ数に達した。 • 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。 • 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。	ブール	false
imq.log.console.output	コンソールへ書き込むロギング情報のカテゴリを指定する文字列。値は次のうちいずれかになります。 • ALL • NONE • 縦線 (|) で区切った、ERROR、WARNING、INFO の値のうち 1 つ以上。それぞれのカテゴリのログメッセージを個別に指定します。どのメッセージカテゴリにも、その他のカテゴリを含めることはできません。	文字列	ERROR|WARNING
imq.log.console.stream	コンソールの出力を標準出力 (OUT)、または標準エラー出力 (ERR) のどちらに書き込むかを指定する文字列。	文字列	ERR
imq.log.file.dirpath	ログファイルが格納されているディレクトリへのパス。オペレーティングシステムによって異なります。	文字列	付録 A を参照
imq.log.file.filename	ログファイルの名前。	文字列	log.txt

表 14-10　監視サービスのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.log.file.output	コンソールに書き込むロギング情報のカテゴリ。値は次のうちいずれかになります。 • ALL • NONE • 縦線 (|) で区切った、ERROR、WARNING、INFO の値のうち 1 つ以上。それぞれのカテゴリのログメッセージを個別に指定します。どのメッセージカテゴリにも、その他のカテゴリを含めることはできません。	文字列	ALL
imq.log.file.rolloverbytes[1]	新しいログファイルに出力がロールオーバーされるログファイルのサイズ (バイト単位)。値を -1 に設定した場合、ファイルサイズに基づいたロールオーバーは無効になります。	整数	-1
imq.log.file.rolloversecs[1]	新しいログファイルに出力がロールオーバーされるログファイルの有効期間 (秒単位)。値を -1 に設定した場合、ファイルの有効期間に基づいたロールオーバーは無効になります。	整数	604800 (1 週間)
imq.log.level[1]	ロガーレベルを指定する文字列。出力チャネルに書き込むことができる出力のカテゴリです。指定したカテゴリとそれより上のレベルのカテゴリが含まれます。値は、上から下に、ERROR、WARNING、INFO となります。	文字列	INFO
imq.log.syslog.facility	(Solaris のみ) Message Queue ブローカが記録する syslog 機能を指定する文字列。値は、syslog(3C) のマニュアルページに示される値をミラー化します。Message Queue で使用できる適切な値は、LOG_USER、LOG_DAEMON、LOG_LOCAL0 から LOG_LOCAL7 です。	文字列	LOG_DAEMON
imq.log.syslog.identity	(Solaris のみ) syslog に記録される各メッセージの先頭に追加する識別情報文字列。 デフォルト値は imqbrokerd_${imq.*instanceName*} です。	文字列	説明を参照

表 14-10　監視サービスのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.log.syslog.logconsole	(Solaris のみ) メッセージを syslog に送信できない場合に、システムコンソールにメッセージを出力するかどうかを指定するブール値。	ブール	false
imq.log.syslog.logpid	(Solaris のみ) メッセージとともにブローカのプロセス ID を記録するかどうかを指定するブール値 (true/false)。	ブール	true
imq.log.syslog.output	(Solaris のみ) syslogd(1M) に書き込むロギング情報のカテゴリを指定する文字列。値は次のうちいずれかになります。 • ALL • NONE • 縦線 (|) で区切った、ERROR、WARNING、INFO の値のうち 1 つ以上。それぞれのカテゴリのログメッセージを個別に指定します。どのメッセージカテゴリにも、その他のカテゴリを含めることはできません。	文字列	ERROR
imq.log.timezone	ログのタイムスタンプのタイムゾーンを表す文字列。識別子は、java.util.TimeZone.getTimeZone() が使用しているものと同じです。たとえば、GMT、America/LosAngeles、Europe/Rome、Asia/Tokyo のように指定します。	文字列	該当地域のタイムゾーン
imq.metrics.enabled	メトリックス情報をロガーへ書き込むかどうかを指定するブール値。メトリックスメッセージの生成には影響しません (imq.metrics.topic.enabled を参照)。	ブール	true
imq.metrics.interval	メトリックスのロギングが有効な場合 (imq.metrics.enabled=true) は、メトリックス情報がロガーへ書き込まれる時間間隔 (秒単位)。メトリックスメッセージの生成の時間間隔には影響しません (imq.metrics.topic.interval を参照)。 値を -1 に設定した場合は、情報が報告されません。	整数	-1

表 14-10　監視サービスのプロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.metrics.topic.enabled	メトリックスメッセージ生成を有効にするかどうかを指定するブール値。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。	ブール	true
imq.metrics.topic.interval	メトリックトピック送信先へ送信するメトリックメッセージを生成する時間間隔 (秒単位)。	整数	60
imq.metrics.topic.persist	メトリックスメッセージを持続性にするかどうかを指定するブール値。	ブール	false
imq.metrics.topic.timetolive	メトリックストピック送信先へ送信されるメトリックスメッセージの有効期間 (秒単位)。	整数	300

1. imqcmd update bkr で使用できます。

クラスタ設定プロパティ

表 14-11 では、ブローカクラスタに関連する設定プロパティについて説明します。

表 14-11　クラスタ設定プロパティ

プロパティ名	説明	データ型	デフォルト値
imq.cluster.brokerlist	クラスタのすべてのブローカを識別する、*host:port* という形式のエントリをコンマで区切ったリスト。*host* はブローカのホスト名、*port* はポートマッパーのポート番号です。 例 : host1:3000,host2:8000,ctrlhost クラスタのすべてのブローカで同じ値にする必要があります。	文字列	なし

表 14-11 クラスタ設定プロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.cluster.masterbroker	クラスタのマスターブローカがある場合は、そのホスト名とポート番号。 値の形式は、*host*:*port* です。*host* はマスターブローカのホスト名、*port* はポートマッパーのポート番号です。 例 : ctrlhost:7676 クラスタのすべてのブローカで同じ値にする必要があります。	文字列	なし
imq.cluster.url[1]	クラスタ設定ファイルがある場合は、その URL。 例 : http://webserver/imq/cluster.properties (Web サーバー上のファイルの場合) file:/net/mfsserver/imq/cluster.properties (共有ドライブ上のファイルの場合) クラスタのすべてのブローカで同じ値にする必要があります。	文字列	なし
imq.cluster.hostname	複数のホストが使用可能である場合、クラスタのブローカ間の内部通信に使用する cluster コネクションサービスがバインドするホスト名か IP アドレス。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。 クラスタのブローカごとに個別に指定できます。	文字列	imq.hostname の値を継承 (316 ページの表 14-2 を参照)。
imq.cluster.port	cluster コネクションサービスのポート番号。 クラスタのブローカごとに個別に指定できます。	整数	0(動的に割り当てられる)

表 14-11 クラスタ設定プロパティ (続き)

プロパティ名	説明	データ型	デフォルト値
imq.cluster.transport	cluster コネクションサービスによって使用されるネットワークトランスポートプロトコル。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、このプロパティを ssl に設定します。 クラスタのすべてのブローカで同じ値にする必要があります。	文字列	tcp

1. imqcmd update bkr で使用できます。

第 15 章

物理的送信先のプロパティのリファレンス

この章では、物理的送信先の各タイプに設定できるプロパティについて説明します。物理的送信先の作成または更新時にプロパティ値を設定できます。

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 (320 ページの表 14-4 を参照)。

表 15-1　物理的送信先のプロパティ

プロパティ	送信先のタイプ	デフォルト値	説明
`maxNumMsgs`[1]	Queue Topic	`-1` (無制限)	送信先で許容される消費されないメッセージの最大数。 デッドメッセージキューの場合、デフォルト値は 1000 です。
`maxTotalMsgBytes`[1]	Queue Topic	`-1` (無制限)	送信先で消費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 (バイト単位)。 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。

表 15-1 物理的送信先のプロパティ (続き)

プロパティ	送信先のタイプ	デフォルト値	説明
limitBehavior	Queue Topic	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応答方法を指定する文字列。値は、次のどれかになります。 FLOW_CONTROL - プロデューサを低速化します。 REMOVE_OLDEST - もっとも古いメッセージを廃棄します。 REMOVE_LOW_PRIORITY - メッセージの有効期間に従い、優先度のもっとも低いメッセージを破棄します (生成元のクライアントはメッセージの削除に関する通知は受信しない)。 REJECT_NEWEST - 最新のメッセージを拒否します。プロデューシングクライアントは、持続メッセージのみの拒否の例外を受けます。持続性がないメッセージでこの制限動作を使用するには、imqAckOnProduce コネクションファクトリ属性を設定します。 このプロパティを REMOVE_OLDEST か REMOVE_LOW_PRIORITY に、送信先プロパティ useDMQ を true に設定すると、ブローカは余分なメッセージをデッドメッセージキューに移動します。 その他の送信先とは異なり、デッドメッセージキュー自体では、デフォルトの制限動作が REMOVE_OLDEST になっており、FLOW_CONTROL 動作には設定できません。
maxBytesPerMsg	Queue Topic	-1 (無制限)	送信先で許容されるシングルメッセージの最大サイズ (バイト単位)。ackOnProduce プロパティを設定していない場合、プロデューシングクライアントは、持続メッセージの拒否で例外を受けますが、持続性のないメッセージの拒否で通知を受けません。

表 15-1　物理的送信先のプロパティ (続き)

プロパティ	送信先のタイプ	デフォルト値	説明
maxNumProducers	Queue Topic	-1 (無制限)	送信先で許容されるプロデューサの最大数。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。 デッドメッセージキューでは、このプロパティを設定できません。
maxNumActiveConsumers	キューのみ	1	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。値を -1 に設定した場合は無制限になります。 Platform Edition では、この値が 2 に制限されます。
maxNumBackupConsumers	キューのみ	0	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。値を -1 に設定した場合は無制限になります。 Platform Edition では、この値が 1 に制限されます。
consumerFlowLimit	Queue Topic	トピック：1000 キュー：1000	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。ロードバランスされたキュー配信では、ロードバランスされる前に、最初にキューに入っていてアクティブコンシューマにルートされるメッセージの数となります。 送信先コンシューマでは、接続で低い値を指定すると、この制限をオーバーライドできます。値を -1 に設定した場合は無制限になります。
localDeliveryPreferred	キューのみ	false	ブローカクラスタでロードバランスしたキュー配信において、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するブール値。送信先をローカルだけの配信に制限することはできません (isLocalOnly = false)。 このプロパティは、デッドメッセージキューに適用されません。

表 15-1　物理的送信先のプロパティ (続き)

プロパティ	送信先のタイプ	デフォルト値	説明
isLocalOnly	Queue Topic	false	ブローカクラスタの送信先において、送信先がローカルのみの配信に制限されるかどうかを指定するブール値。true に設定すると、送信先はそのほかのブローカに複製されず、メッセージの配信がローカルコンシューマ (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) だけに制限されます。送信先が作成されると、このプロパティは変更できません。 このプロパティは、デッドメッセージキューに適用されません。
useDMQ	Queue Topic	true	デッドメッセージを破棄せずに、デッドメッセージキューに送信するかどうかを指定するブール値。 このプロパティは、デッドメッセージキューに適用されません。

1. クラスタ環境では、このプロパティはクラスタ内のすべてのインスタンスに一括して適用されるのではなく、クラスタ内の送信先の各インスタンスに適用されます。

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。

送信先のプロパティ

表 16-1 では、送信先管理対象オブジェクトを設定する属性について一覧表示します。

表 16-1 送信先管理対象オブジェクトの属性

属性名	説明	データ型	デフォルト値
imqDestinationDescription	送信先オブジェクトの説明。	文字列	なし
imqDestinationName	物理的送信先の名前。	文字列[1]	Untitled_Destination_Object

1. 送信先名には、英数字 (空白文字は含まない) だけを使用できます。送信先名は、英字や、「_」または「$」で開始する必要があります。

コネクションファクトリの属性

この節では、コネクションファクトリ管理対象オブジェクトを設定する属性の参照情報を提供します。属性は次のセクションに分類できます。

表 16-2 は、コネクションファクトリ管理対象オブジェクトの属性のインデックスです。最初の列では、それぞれの属性がアルファベット順に並んでいます。第 2 列はカテゴリ、第 3 列は属性について説明する表への相互参照です。

コネクションの処理

表 16-2 では、コネクション処理のコネクションファクトリの属性について一覧表示します。

表 16-2 コネクションファクトリの属性 : コネクションの処理

属性名	説明	データ型	デフォルト値
imqAddressList	1 つ以上のメッセージサーバーアドレスをコンマで区切ったリスト。使用するコネクションサービスとポート割り当て手法により、アドレススキーマは異なります。 アドレスリストの指定方法、およびリストエントリについて説明する例については、345 ページの「imqAddressList 属性値の構文」を参照してください。	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 346 ページの表 16-3 の最初のエントリ。
imqAddressListBehavior	imqAddressList 属性のアドレスの順序で接続を試すか (PRIORITY)、ランダムに接続を試すか (RANDOM) を指定する文字列。多くのクライアントが同じコネクションファクトリを使用してコネクションを試す場合は、ランダムな順序を使用し、すべてのクライアントが同じアドレスに接続することを防ぐことができます。	文字列	PRIORITY

表 16-2　コネクションファクトリの属性 : コネクションの処理 (続き)

属性名	説明	データ型	デフォルト値
imqAddressListIterations	コネクションの確立か再確立で、クライアントランタイムが imqAddressList を繰り返す回数。値を -1 にすると、試行回数は無制限になります。	整数	5
imqPingInterval	アプリケーションとブローカ間でクライアントランタイムが接続をテストする、秒単位の頻度。 値を -1 か 0 (ゼロ) にすると、クライアントランタイムはコネクションを定期的にテストしません。	整数	30
imqReconnectEnabled	コネクションが失われたとき、クライアントランタイムがメッセージサーバー (または imqAddressList のアドレスリスト) への再接続を試すかどうかを指定するブール値。	ブール	false
imqReconnectAttempts	クライアントランタイムが、imqAddressList の次のアドレスを試す前に、リストのアドレスごとに接続か再接続を試す回数。値を -1 にすると、再接続試行回数は無限になり、クライアントランタイムは、正常に接続されるまで、最初のアドレスへの接続を試します。	整数	0
imqReconnectInterval	再接続を試す間隔 (ミリ秒単位)。この値は、imqAddressList のそれぞれのアドレスにおける試行、およびリストの後続アドレスに適用されます。値が小さすぎる場合、ブローカでは復元時間が不十分になります。値が大きすぎる場合、再接続により、許容できない遅延が発生することがあります。	long	3000
imqSSLIsHostTrusted	クライアントがブローカの自己署名型証明書を受け入れるかどうかを指定するブール値。認証局からの署名付き証明書を使用するには、この値を false に設定します。	ブール	true

imqAddressList 属性値の構文

imqbrokerlist 値のそれぞれのアドレスは、クライアントランタイムが接続できるブローカインスタンスに対応します。

コネクションサービスごとに別々の方法でブローカアドレスを指定します。一般的な構文は次のとおりです。

scheme://*address_syntax*

リストにアドレスを追加するには、コンマに続けて別のアドレスを追加します。リストには任意の数のエントリを含めることができます。形式は次のとおりです。

scheme://*address_syntax*,*scheme*://*address_syntax*...

scheme 変数では、表 16-3 で説明するように、mq、mqtcp、mqssl、http、https など、使用するアドレスタイプを指定します。*address_syntax* 変数は、スキーマ固有のブローカアドレスを表します。表 16-3 では、アドレススキーマについて説明します。最初の列は、アドレススキーマの名前です。第 2 列は、その名前に関連するコネクションサービス、第 3 列は説明、第 4 列は使用する構文です。

表 16-3 imqAddressList 属性のアドレススキーマ

スキーマ	コネクションサービス	説明	構文
mq	ms ssljms	jms サービスか ssljms サービスで使用するダイナミックポート割り当てを提供します。 ポートマッパーのホストとポートを指定します。ポートマッパーにより、接続に使用するポートがダイナミックに割り当てられます。	[*hostName*][:*port*][/*serviceName*] jms コネクションサービスの場合は、次のデフォルト値が適用されます。 *hostName* = `localhost` *port* = `7676` *serviceName* = `jms` ssljms コネクションサービスの場合、デフォルト値はありません。すべての変数を指定する必要があります。
mqtcp	jms	ポート番号を指定し、jms コネクションサービスを使用します。 Message Queue クライアントランタイムにより、指定したホストとポートに TCP 接続が行なわれ、コネクションが確立されます。	*hostName*:*port*/`jms`
mqssl	ssljms	ポート番号を指定し、ssljms コネクションサービスを使用します。 Message Queue クライアントランタイムにより、指定したホストとポートに安全な SSL 接続が行なわれ、コネクションが確立されます。	*port*/`ssljms`

表 16-3　imqAddressList 属性のアドレススキーマ (続き)

スキーマ	コネクションサービス	説明	構文
http	httpjms	httpjms コネクションサービスを使用します。 クライアントランタイムにより、指定した URL で Message Queue トンネルサーブレットへの HTTP 接続が行なわれます。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。	`http://`*hostName:port*`/`*contextRoot*`/tunnel` 複数のブローカインスタンスで同じトンネルサーブレットを使用する場合、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する構文は次のとおりです。 `http://`*hostName:port*`/`*contextRoot*`/tunnel?ServerName=`*hostName:instanceName*
https	httpsjms	httpsjms コネクションサービスを使用します。 Message Queue クライアントランタイムにより、指定した Message Queue トンネルサーブレット URL への安全な HTTPS 接続が行なわれます。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。	`https://`*hostName:port*`/`*contextRoot*`/tunnel` 複数のブローカインスタンスで同じトンネルサーブレットを使用する場合、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する構文は次のとおりです。 `https://`*hostName:port*`/`*contextRoot*`/tunnel?ServerName=`*hostName:instanceName*

表 16-4 はアドレス形式の例です。最初の列はコネクションサービスの名前です。第 2 列は、例のホストがローカルホストであるか、未指定ホストであるか、指定済みホストであるか、該当なしであるかを示します。第 3 列は、例のポートが指定済みであるか、未指定であるか、該当なしであるかを示します。第 4 列は例です。

表 16-4　メッセージサーバーアドレスの例

コネクションサービス	ブローカホスト	ポート	アドレス例
未指定	ローカルホスト	未指定	デフォルト値 (`mq://localHost:7676/jms`)
未指定	指定済みホスト	未指定	`myBkrHost` (`mq://myBkrHost:7676/jms`)
未指定	未指定	ポートマッパーのポート指定	`1012` (`mq://localHost:1012/jms`)
ssljms	ローカルホスト	ポートマッパーのポート未指定	`mq://localHost:7676/ssljms`

表 16-4　メッセージサーバーアドレスの例 (続き)

コネクションサービス	ブローカホスト	ポート	アドレス例
ssljms	指定済みホスト	ポートマッパーポート	mq://myBkrHost:7676/ssljms
ssljms	指定済みホスト	ポートマッパーのポート指定	mq://myBkrHost:1012/ssljms
jms	ローカルホスト	サービスポート指定	mqtcp://localhost:1032/jms
ssljms	指定済みホスト	サービスポート指定	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 16-5 では、クライアント識別のコネクションファクトリの属性について一覧表示します。

表 16-5　コネクションファクトリの属性 : クライアントの識別

属性名	説明	データ型	デフォルト値
imqDefaultUsername	ブローカで認証するためのデフォルトユーザー名。	文字列	guest
imqDefaultPassword	ブローカで認証するためのデフォルトパスワード。	文字列	guest
imqConfiguredClientID	管理用に設定するクライアント ID。	文字列	null
imqDisableSetClientID	JMS API の setClientID() メソッドを使用してクライアント ID を変更することをできないようにするかどうかを指定するブール値。	ブール	false

メッセージヘッダーのオーバーライド

表 16-6 では、JMS メッセージヘッダーフィールドをオーバーライドするコネクションファクトリ属性について一覧表示します。

表 16-6 コネクションファクトリの属性 : メッセージヘッダーのオーバーライド

属性名	説明	データ型	デフォルト値
imqOverrideJMSDeliveryMode	クライアントが設定した JMSDeliveryMode フィールドをオーバーライドできるかどうかを指定するブール値。	ブール	false
imqJMSDeliveryMode	JMSDeliveryMode のオーバーライド値。値は 1 (持続性なし) と 2 (持続性あり) です。	整数	2
imqOverrideJMSExpiration	クライアントが設定した JMSExpiration フィールドをオーバーライドできるかどうかを指定するブール値。	ブール	false
imqJMSExpiration	JMSExpiration のオーバーライド値 (ミリ秒単位)。	long	0 (失効なし)
imqOverrideJMSPriority	クライアントが設定した JMSPriority フィールドをオーバーライドできるかどうかを指定するブール値。	ブール	false
imqJMSPriority	JMSPriority のオーバーライド値 (0 から 9 までの整数)。	整数	4 (標準)
imqOverrideJMSHeadersToTemporaryDestinations	オーバーライドを一時的送信先に適用するかどうかを指定するブール値。	ブール	false

信頼性およびフロー制御

表 16-7 では、信頼性とフロー制御を設定するコネクションファクトリ属性について一覧表示します。

表 16-7 コネクションファクトリの属性 : 信頼性およびフロー制御

属性名	説明	データ型	デフォルト値
imqAckTimeout	クライアントランタイムが例外をスローする前にブローカの応答を待機する最大時間 (ミリ秒単位)。値を 0 にすると、タイムアウトはなくなり、クライアントランタイムは永久に待機します。 一部の状況では、この値が低すぎるため、クライアントランタイムがタイムアウトになることがあります。たとえば、ブローカが LDAP ユーザーリポジトリと安全な (SSL) コネクションを使用して認証するそれぞれのユーザーでは、最初の認証にかかる時間が 30 秒を超えることがあります。	文字列	0
imqAckOnProduce	ブローカがプロデューシングクライアントからのメッセージに応答する方法を指定する文字列。 この属性を true に設定すると、ブローカは、プロデューシングクライアントから受信したすべての JMS メッセージ (持続性ありと持続性なし) に応答します。プロデューシングクライアントスレッドは、この応答の待機中にブロックされます。 この属性を指定しない場合、ブローカは持続メッセージのみに応答します。プロデューシングクライアントスレッドは、この応答の待機中にブロックされます。	文字列	指定なし
imqConnectionFlowCount	測定済みバッチの JMS メッセージ数。この数の JMS メッセージがクライアントランタイムに配信されると、配信は一時的に中断され、保留されていた制御メッセージが配信されます。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による通知時に再開され、カウントに再び達するまで継続されます。 カウントを 0 に設定すると、測定済みバッチの JMS メッセージ数の制限はなくなります。ゼロ以外に設定すると、大量の JMS メッセージ配信によって Message Queue 制御メッセージがブロックされないように、クライアントランタイムによってメッセージフローが測定されます。	整数	100

表 16-7 コネクションファクトリの属性 : 信頼性およびフロー制御 (続き)

属性名	説明	データ型	デフォルト値
imqConnectionFlowLimitEnabled	imqConnectionFlowLimit の値を使用して、コネクションレベルでメッセージフローを制限するかどうかを指定するブール値。	ブール	false
imqConnectionFlowLimit	コネクションを介して配信され、消費を待機する間、クライアントランタイムにバッファリングされるメッセージ数の最大数。 imqConnectionFlowIsLimited を有効にしないと、この制限がチェックされないことに注意してください。 imqConnectionFlowCount によって制御されるフロー測定に従い、クライアントランタイムに配信される JMS メッセージの数がこの制限を超えると、メッセージ配信は停止します。消費されていないメッセージ数が、この属性で設定した値を下回った場合にかぎり、配信は再開されます。 この制限により、メッセージ処理に時間がかかるコンシューミングクライアントが保留メッセージに圧倒されて、メモリー不足になることが防止されます。	整数	1000

表 16-7　コネクションファクトリの属性 : 信頼性およびフロー制御 (続き)

属性名	説明	データ型	デフォルト値
imqConsumerFlowLimit	コネクションを介して配信され、消費を待機する間、クライアントランタイムにバッファリングされる、コンシューマごとのメッセージ数の最大数。複数のコンシューマがキュー配信を実行する状況で、不均衡な数のメッセージが 1 つのコンシューマに送信されないように、コンシューマ間のロードバランスを改善するには、この制限を使用します。キューの consumerFlowLimit 属性でブローカサイドにおいて低い値を設定すると、この制限をオーバーライドできます。『Message Queue 管理ガイド』の送信先属性を参照してください。 この制限では、接続の 1 つのコンシューマのために、コネクションのその他のコンシューマにメッセージが送信されなくなることも防止できます。 クライアントランタイムへ配信された JMS メッセージの数が、いずれかのコンシューマのこの制限を超えた場合、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 で設定した値を下回った場合にだけ、配信処理が再開されます。 コネクションのすべてのコンシューマでバッファリングされるメッセージ総数が imqConnectionFlowLimit を超えると、その総数がコネクションの制限を下回るまで、そのコネクションによるメッセージ配信は停止します。	整数	100
imqConsumerFlow Threshold	クライアントランタイムでバッファリングされる、コンシューマごとのメッセージ数を、imqConsumerFlowLimit の率として指定します。これを下回ると、コンシューマへのメッセージ配信が再開されます。	整数	50

キューブラウザの動作とサーバーセッション

表 16-8 では、クライアントのキューブラウズに影響する属性について説明します。

表 16-8　コネクションファクトリの属性 : キューブラウザの動作

属性名	説明	データ型	デフォルト値
imqQueueBrowserMax MessagesPerRetrieve	キュー送信先の内容を検索する場合、クライアントが一度に取得できる最大メッセージ数。	整数	1000
imqQueueBrowserRetrieveTimeout	キュー送信先の内容を検索するとき、クライアントランタイムが例外をスローする前に、メッセージの取得を待機する、最大時間 (ミリ秒単位)。	long	60000
imqLoadMaxToServerSession	JMS アプリケーションサーバー機能の場合、Message Queue の接続コンシューマが、maxMessages に指定した数までのメッセージを ServerSession のセッションにロードするかどうかを指定するブール値。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ロードします。		true

JMS の定義済みプロパティのサポート

JMS によって定義されるプロパティは、JMS によって予約される名前であり、JMS プロバイダは、クライアントのプログラミング機能を向上するためにサポートすることを選択できます。表 16-9 では、Message Queue によってサポートされる、JMS 定義のプロパティについて説明します。

表 16-9　コネクションファクトリの属性 : JMS の定義済みプロパティのサポート

プロパティ名	説明	データ型	デフォルト値
imqSetJMSXUserID	JMS 定義プロパティ JMSXUserID (メッセージを送信するユーザーの ID) を生成済みメッセージで設定するかどうかを指定するブール値。	ブール	false
imqSetJMSXAppID	JMS 定義プロパティ JMSXAppID (メッセージを送信するアプリケーションの ID) を生成済みメッセージで設定するかどうかを指定するブール値。	ブール	false
imqSetJMSXProducerTXID	JMS 定義プロパティ JMSXProducerTXID (この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) を生成済みメッセージで設定するかどうかを指定するブール値。	ブール	false

表 16-9　コネクションファクトリの属性 : JMS の定義済みプロパティのサポート (続き)

プロパティ名	説明	データ型	デフォルト値
imqSetJMSXConsumerTXID	JMS 定義プロパティ JMSXConsumerTXID (この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) を消費済みメッセージで設定するかどうかを指定するブール値。	ブール	false
imqSetJMSXRcvTimestamp	消費されたメッセージに対して、JMS 定義プロパティ JMSXRcvTimestamp (メッセージをコンシューマに配信する時間) を設定するかどうかを指定するブール値。	ブール	false

SOAP の端点 (endpoint) の属性

表 16-10 では、SOAP を使用するアプリケーションの端点の URL を設定する属性について説明します。SOAP を使用するアプリケーションについては、『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

属性名	説明	データ型	デフォルト値
imqSOAPEndpointList	メッセージ送信先の SOAP の端点を表す、スペースで区切った 1 つ以上の URL のリスト。 複数の URL を指定すると、メッセージはリストのすべての URL にブロードキャストされます。それぞれの URL は、SOAP メッセージの受信と処理を実行できるサーブレットに関連している必要があります。 例 : `http://www.myServlet/ http://www.myServlet2/`	文字列	
imqEndpointName	SOAP の端点の名前。 例 : MyTopicEndpoint	文字列	Untitled_ Endpoint_ Object
imqEndpointDescription	SOAP の端点の属性の説明。 例 : `"imqEndpointDescription=my endpoints for broadcast"`	文字列	A description for the endpoint object

第 17 章

JMS リソースアダプタ属性リファレンス

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 (JCA) により、Sun Java System Message Queue を 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 メッセージの送受信が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 コンポーネントで設定属性が公開されます。

- `ResourceAdapter` 設定は、リソースアダプタ全体の動作に影響します。
- `ManagedConnectionFactory` 設定は、メッセージ駆動型 Beans (MDB) で使用するようにリソースアダプタが作成したコネクションに影響します。
- `ActivationSpec` 設定は、メッセージングシステムとのやり取りにおけるメッセージ駆動型 Beans MDB を表すメッセージの終端に影響します。

このエンティティの属性値を設定するには、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属性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 では、この JavaBean を設定できる属性を一覧表示して説明します。必須プロパティ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属性

名前	説明	デフォルト値
addressList[1]	リソースアダプタが Message Queue サービスに作成するコネクションであり、メッセージサービスアドレス形式で指定します。 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。 この属性名 addressList は Sun Java System Message Queue に固有ですが、標準属性 connectionURL と同じ意味です。Sun Java System Message Queue では、両方の属性名が提供されます。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。	mq://localhost:7676/jms
addressListBehavior	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。 PRIORITY コネクションでは、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。 RANDOM コネクションでは、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。 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。その再接続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。	PRIORITY
addressListIterations	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。	1
connectionURL	リソースアダプタが Message Queue サービスに作成するコネクションであり、メッセージサービスアドレス形式で指定します。 addressList 属性と同じです。詳細については、上の説明を参照してください。	mq://localhost:7676/jms

表 17-1 リソースアダプタの属性 (続き)

名前	説明	デフォルト値
userName[1]	リソースアダプタが Message Queue サービスに接続するデフォルトユーザー名。 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。	guest
password[1]	リソースアダプタが Message Queue サービスに接続するデフォルトパスワード。 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。	guest
reconnectAttempts	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 に設定した場合は、この属性を使用します。	6
reconnectEnabled	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。 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。	false
reconnectInterval	再接続を試す間隔 (ミリ秒単位)。reconnectEnabled を true に設定した場合は、この属性を使用します。	30000

1. このプロパティは必須です。

ManagedConnectionFactory JavaBean

管理対象コネクションファクトリでは、リソースアダプタがメッセージ駆動型 Bean に提供するコネクションの提供と定義を行います。ResourceAdapter JavaBean に類似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 では、Message Queue のリソースアダプタによって提供される管理対象コネクションファクトリの設定可能属性を一覧表示して説明します。

表 17-2 管理対象コネクションファクトリの属性

名前	説明	デフォルト値
addressList	この管理対象コネクションファクトリから派生した接続のリスト。 このプロパティの形式は、356 ページの表 17-1 で説明した Message Service 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 の値がコネクションで使用されます。	なし
addressListBehavior	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。 PRIORITY コネクションでは、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。 RANDOM コネクションでは、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。 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。その再接続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。	PRIORITY
addressListIterations	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。	1
clientID	この管理対象コネクションファクトリから派生したコネクションに使用するクライアント識別子。	なし
password	(任意指定) コネクションのパスワード。 この値を設定しない場合は、356 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コネクションで使用されます。	guest
reconnectAttempts	アドレスリストの 1 つのエントリに再接続を試す回数。	6

表 17-2　管理対象コネクションファクトリの属性 (続き)

名前	説明	デフォルト値
reconnectEnabled	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。 再接続の試行は、reconnectInterval プロパティと reconnectAttempts プロパティによって制御されます。	false
reconnectInterval	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。	30000
userName	(任意指定) 接続のユーザー名。 この値を設定しない場合は、356 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コネクションで使用されます。	guest

ActivationSpec JavaBean

アプリケーションサーバーは、リソースアダプタに命令して、メッセージ終端をアクティブにし、メッセージ終端とメッセージ駆動型 Bean を関連付けるとき、ActivationSpec JavaBean のプロパティを使用します。

表 17-3 では、メッセージ終端アクティブ化仕様の設定可能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、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について説明します。

表 17-3　アクティブ化仕様の属性

名前	説明	デフォルト値
acknowledgeMode	(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。 これは、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティです。 値は、Auto-acknowledge か Dups-ok-acknowledge にすることができます。	Auto-acknowledge

表 17-3 アクティブ化仕様の属性 (続き)

名前	説明	デフォルト値
addressList	(任意指定) メッセージ終端のためにリソースアダプタが作成するコネクションの仕様。 この属性は Message Queue JMS リソースアダプタに固有です。 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。	ResourceAdapter JavaBean 設定の addressList から継承
clientId	このコンシューマ用に作成された JMS コネクションによって使用される JMS クライアント ID。 subscriptionDurability 属性を Durable に設定した場合は、この属性を設定する必要があります。 これは、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティです。	なし
customAcknowledgeMode	MDB メッセージの消費のモードを指定する文字列。 この属性の有効な値は、No_acknowledge か NULL です。 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。	なし
destination	この MDB がメッセージを消費する送信先の名前。 必須属性であり、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティです。 Message Queue の送信先管理対象オブジェクトの destinationName プロパティの値に設定する必要があります。	なし
destinationType	destination 属性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。 必須属性であり、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティです。	なし
endpointExceptionRedeliveryAttempts	メッセージ配信中に MDB で例外をスローしたとき、MDB にメッセージを再配信する回数。	6

表 17-3 アクティブ化仕様の属性 (続き)

名前	説明	デフォルト値
messageSelector	(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。 これは、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティです。	なし
sendUndeliverableMsgsToDMQ	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。 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。	true
subscriptionDurability	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。 永続的でないサブスクリプションの属性は任意指定であり、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、属性 clientID と subscriptionName も設定する必要があります。 標準的な EJB 2.1 と J2EE CA1.5 のプロパティであり、destinationType 属性を avax.jms.Topic に設定した場合にかぎって有効です。	NonDurable
subscriptionName	永続サブスクリプションの指定に使用する文字列。 subscriptionDurability 属性を Durable に設定した場合は、この属性を設定する必要があります。 これは、標準的な EJB 2.1 と J2EE CA 1.5 のプロパティです。	なし

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、JVM ヒープを処理するためにブローカによって生成されるメトリックスデータを一覧表示して説明します。この表では、提供されているメトリックス監視ツールをメトリックスごとに示します。

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。	必須 (cxn)	必須	必須 (...jvm)
JVM heap: total memory	現在の JVM ヒープサイズ。	必須 (cxn)	必須	必須 (...jvm)
JVM heap: max memory	JVM ヒープサイズを拡大可能な上限。	オプション	必須[1]	必須 (...jvm)

1. ブローカの起動時にだけ表示されます。
2. メトリックスのトピック送信先名については、217 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 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とその説明を一覧表示しています。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]
コネクションデータ				
Num connections	現在開いているブローカへのコネクションの数	必須 (cxn)	必須	必須 (...broker)
Num threads	すべてのコネクションサービスで現在使用されているスレッドの合計数	必須 (cxn)	必須	オプション
Min threads	コネクション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	必須 (cxn)	必須	オプション
Max threads	コネクションサービスが使用するために、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	必須 (cxn)	必須	オプション
格納済みのメッセージデータ				
Num messages	現在、ブローカメモリーと持続ストアに格納されている JMS メッセージの数	オプション `query bkr` を使用	オプション	必須 (...broker)
Total message bytes	現在、ブローカメモリーと持続ストアに格納されている JMS メッセージバイトの数	オプション `query bkr` を使用	オプション	必須 (...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が受信した JMS メッセージの数	必須 (ttl)	必須	必須 (...broker)
Message bytes in	ブローカが最後に起動された以降にブローカが受信した JMS メッセージのバイト数	必須 (ttl)	必須	必須 (...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が受信したパケットの数。JMS メッセージと制御メッセージの両方が含まれます	必須 (ttl)	必須	必須 (...broker)
Packet bytes in	ブローカが最後に起動された以降にブローカが受信したパケットのバイト数。JMS メッセージと制御メッセージの両方が含まれます	必須 (ttl)	必須	必須 (...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 JMS メッセージの数	必須 (ttl)	必須	必須 (...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 JMS メッセージのバイト数	必須 (ttl)	必須	必須 (...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。JMS メッセージと制御メッセージの両方が含まれます	必須 (ttl)	必須	必須 (...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のバイト数。JMS メッセージと制御メッセージの両方が含まれます	必須 (ttl)	必須	必須 (...broker)
Rate messages in	ブローカへの JMS メッセージの現在のフローレート	必須 (rts)	必須	オプション
Rate message bytes in	ブローカへの JMS メッセージバイトの現在のフローレート	必須 (rts)	必須	オプション
Rate packets in	ブローカへのパケットの現在のフローレート。JMS メッセージと制御メッセージの両方を含みます	必須 (rts)	必須	オプション
Rate packet bytes in	ブローカへのパケットバイトの現在のフローレート。JMS メッセージと制御メッセージの両方を含みます	必須 (rts)	必須	オプション
Rate messages out	ブローカからの JMS メッセージの現在のフローレート	必須 (rts)	必須	オプション

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 message bytes out	ブローカからの JMS メッセージのバイトの現在のフローレート	必須 (rts)	必須	オプション
Rate packets out	ブローカからのパケットの現在のフローレート。JMS メッセージと制御メッセージの両方を含みます	必須 (rts)	必須	オプション
Rate packet bytes out	ブローカからのパケットバイトの現在のフローレート。JMS メッセージと制御メッセージの両方を含みます	必須 (rts)	必須	オプション
送信先データ				
Num destinations	ブローカ内の物理的な送信先の数	オプション	オプション	必須 (...broker)

1. メトリックスのトピック送信先名については、217 ページの表 10-7 を参照してください。

コネクションサービスのメトリックス

表 18-3 は、個々のコネクション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とその説明を一覧表示しています。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　コネクション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*)
コネクションデータ				
Num connections	現在開かれているコネクション数	必須 (cxn) query svc も使用	オプション	オプション
Num threads	現在使用中のスレッドの数	必須 (cxn) query svc も使用	オプション	オプション

表 18-3　コネクション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*)
Min threads	すべてのコネクションサービスで、コネクション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	必須 (cxn)	オプション	オプション
Max threads	すべてのコネクションサービスで、コネクション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	必須 (cxn)	オプション	オプション
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コネクションサービスが受信した JMS メッセージの数	必須 (ttl)	オプション	オプション
Message bytes in	ブローカが最後に起動された以降にコネクションサービスが受信した JMS メッセージのバイト数	必須 (ttl)	オプション	オプション
Num packets in	ブローカが最後に起動された以降にコネクションサービスが受信したパケットの数。JMS メッセージと制御メッセージの両方が含まれます	必須 (ttl)	オプション	オプション
Packet bytes in	ブローカが最後に起動された以降にコネクションサービスが受信したパケットのバイト数。JMS メッセージと制御メッセージの両方が含まれます	必須 (ttl)	オプション	オプション
Num messages out	ブローカが最後に起動された以降にコネクションサービスから送信した JMS メッセージの数	必須 (ttl)	オプション	オプション
Message bytes out	ブローカが最後に起動された以降にコネクションサービスから送信した JMS メッセージのバイト数	必須 (ttl)	オプション	オプション
Num packets out	ブローカが最後に起動された以降にコネクションサービスから送信したパケットの数。JMS メッセージと制御メッセージの両方が含まれます	必須 (ttl)	オプション	オプション

表 18-3　コネクション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*)
Packet bytes out	ブローカが最後に起動された以降にコネクションサービスから送信したパケットのバイト数。JMS メッセージと制御メッセージの両方が含まれます	必須 (ttl)	オプション	オプション
Rate messages in	コネクションサービス経由でブローカが受信する JMS メッセージの現在のフローレート	必須 (rts)	オプション	オプション
Rate message bytes in	コネクションサービスへの JMS メッセージバイトの現在のフローレート	必須 (rts)	オプション	オプション
Rate packets in	コネクションサービスへのパケットの現在のフローレート。JMS メッセージと制御メッセージの両方を含みます	必須 (rts)	オプション	オプション
Rate packet bytes in	コネクションサービスへのパケットの現在のフローレート。JMS メッセージと制御メッセージの両方を含みます	必須 (rts)	オプション	オプション
Rate messages out	コネクションサービスからの JMS メッセージの現在のフローレート	必須 (rts)	オプション	オプション
Rate message bytes out	コネクションサービスからの JMS メッセージバイトの現在のフローレート	必須 (rts)	オプション	オプション
Rate packets out	コネクションサービスからのパケットの現在のフローレート。JMS メッセージと制御メッセージの両方を含みます	必須 (rts)	オプション	オプション
Rate packet bytes out	コネクションサービスからのパケットバイトの現在のフローレート。JMS メッセージと制御メッセージの両方を含みます	必須 (rts)	オプション	オプション

送信先メトリックス

表 18-4 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とその説明を一覧表示しています。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。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。	必須 (con)	オプション	必須 (...*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	必須 (con)	オプション	必須 (...*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	必須 (con)	オプション	必須 (...*destName*)
Num active consumers	現在のアクティブなコンシューマの数	必須 (con)	オプション	必須 (...*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	必須 (con)	オプション	必須 (...*destName*)
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	必須 (con)	オプション	必須 (...*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	必須 (con)	オプション	必須 (...*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	必須 (con)	オプション	必須 (...*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	必須 (con)	オプション	必須 (...*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]
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 JMS メッセージの数	必須 (con) (ttl) (rts) query dst も使用	オプション	必須 (...*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 JMS メッセージ数の平均	必須 (con) (ttl) (rts)	オプション	必須 (...*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 JMS メッセージの最大時の数	必須 (con) (ttl) (rts)	オプション	必須 (...*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 JMS メッセージのバイト数	必須 (ttl) (rts) query dst も使用	オプション	必須 (...*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 JMS メッセージのバイト数の平均	必須 (ttl) (rts)	オプション	必須 (...*destName*)
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 JMS メッセージの最大時のバイト数	必須 (ttl) (rts)	オプション	必須 (...*destName*)
Peak message bytes	ブローカが最後に起動された以降に、送信先が受信したシングルメッセージ内の JMS メッセージの最大時のバイト数	必須 (ttl) (rts)	オプション	必須 (...*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、この送信先が受信した JMS メッセージの数	必須 (ttl)	オプション	必須 (...*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]
Msg bytes in	ブローカが最後に起動された以降に、この送信先が受信した JMS メッセージのバイト数	必須 (ttl)	オプション	必須 (...*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 JMS メッセージの数	必須 (ttl)	オプション	必須 (...*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 JMS メッセージのバイト数	必須 (ttl)	オプション	必須 (...*destName*)
Rate num messages in	送信先への JMS メッセージの現在のフローレート	必須 (rts)	オプション	オプション
Rate num messages out	送信先からの JMS メッセージの現在のフローレート	必須 (rts)	オプション	オプション
Rate msg bytes in	送信先への JMS メッセージバイトの現在のフローレート	必須 (rts)	オプション	オプション
Rate Msg bytes out	送信先からの JMS メッセージのバイトの現在のフローレート	必須 (rts)	オプション	オプション
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)	必須 (dsk)	オプション	必須 (...*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)	必須 (dsk)	オプション	必須 (...*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	必須 (dsk)	オプション	必須 (...*destName*)

1. メトリックスのトピック送信先名については、217 ページの表 10-7 を参照してください。

第 IV 部

付録

オペレーティングシステムごとの Message Queue データの場所

Sun Java System Message Queue データは、次の節で説明するように、オペレーティングシステムごとに異なる場所に保存されます。

この付録では、次のオペレーティングシステムにおける、さまざまなタイプの Message Queue データの場所について説明します。

- 375 ページの「Solaris」
- 377 ページの「Linux」
- 378 ページの「Windows」

次の表の *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ています。

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。

Sun Java System Application Server のスタンドアロンバージョンを含む Solaris で Message Queue を使用している場合、ディレクトリ構造は、378 ページの「Windows」で説明する構造のようになります。

表 A-1　Solaris 上での Message Queue データの場所

データのカテゴリ	Solaris 上での場所
ブローカインスタンスの設定プロパティ	`/var/imq/instances/`*instanceName*`/props/config.properties`

表 A-1　Solaris 上での Message Queue データの場所 (続き)

データのカテゴリ	Solaris 上での場所
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
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 : 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 : 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 : passfile 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 : passfile の例	/etc/imq/passfile.sample
セキュリティ : 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 上での Message Queue データの場所

データのカテゴリ	Windows 上での場所
ブローカインスタンスの設定プロパティ	/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
セキュリティ : 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 : 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 : passfile 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 : passfile の例	/etc/opt/sun/mq/passfile.sample
セキュリティ : 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。

Message Queue がスタンドアロンバージョンの Sun Java System Application Server とバンドルされている場合は、Solaris での Message Queue データの場所についても説明します。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 のパス名を使用しますが、Windows の円記号 (¥) を Solaris のスラッシュ (/) に変更してください。詳細については、25 ページの表 3 の `IMQ_HOME` と `IMQ_VARHOME` の定義を参照してください。

表 A-3 Windows 上での Message Queue データの場所

データのカテゴリ	Windows 上での場所
ブローカインスタンスの設定プロパティ	`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
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 : 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 : アクセス制御ファイル (デフォルト)	`IMQ_VARHOME¥instances¥`*instanceName*`¥etc¥accesscontrol.properties`
セキュリティ : passfile のディレクトリ (デフォルトの場所)	`IMQ_HOME¥etc¥`
セキュリティ : passfile の例	`IMQ_HOME¥etc¥passfile.sample`
セキュリティ : ブローカのキーストアファイルの場所	`IMQ_HOME¥etc¥`

表 A-3　Windows 上での Message Queue データの場所 (続き)

データのカテゴリ	Windows 上での場所
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リストです。

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 ファイルと 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	
Message Queue JMS リソースアダプタの設定プロパティ	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	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、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、JMS_SUN_*	発展中
Message Queue クライアントメッセージプロパティ : 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 クライアントがサポートされ、直接 TCP コネクションではなく、HTTP またはセキュリティ保護された HTTP (HTTPS) 転送でブローカとやり取りします。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 ベースのコネクション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

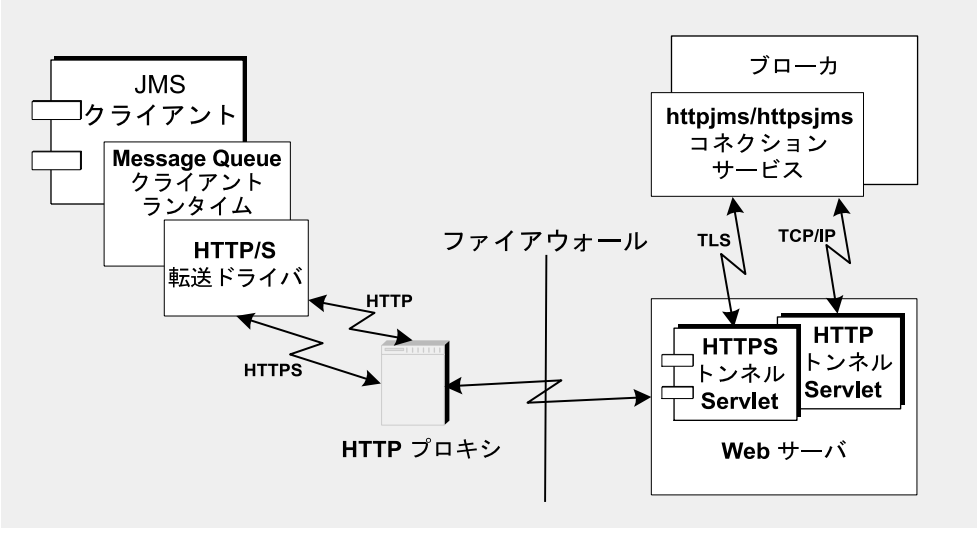
Message Queue メッセージングは、HTTP/HTTPS コネクションで実行できます。HTTP/HTTPS コネクションは、通常ファイアウォールを通して許可されるため、ファイアウォールによってブローカからクライアントアプリケーションを分離できます。

386 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 の転送ドライバが Message Queue メッセージを HTTP 要求にカプセル化し、これらの要求を正しい手順で Web サーバーに確実に送信します。

- クライアントは、HTTP プロキシサーバーを使用して、必要に応じ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。詳細については、392 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) は、Web サーバーに読み込まれ、JMS メッセージがブローカに転送される前に、その JMS メッセージをクライアント HTTP 要求から取り出します。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が複数のブローカへのアクセスに使用されます。

図 C-1　　HTTP/HTTPS サポートのアーキテクチャ



- ブローカ側では、httpjms または httpsjms コネクションサービスが、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非多重化します。
- Web サーバーで障害が生じても再起動すれば、すべてのコネクションが復元され、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コネクション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かった場合は、Web サーバーはクライアントコネクションを復元し、引き続きブローカコネクションを待機しますが、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は非常に良く似ています。主な相違点は、HTTPS (httpsjms コネクションサービス) 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コネクションがあることです。

ブローカへの安全なコネクションは、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、コネクションを要求している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コネクションを設定します。このコネクションが確立されると、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コネクションについて、クライアントアプリケーションと Web サーバーがネゴシエーションを行います。

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 Web サーバーに配置する。
2. ブローカの httpjms コネクションサービスを設定し、ブローカを起動する。
3. HTTP コネクションを設定する。

手順 1: HTTP トンネルサーブレットを Web サーバーに配置する

HTTP トンネルサーブレットを Web サーバーに配置するには、通常次の 2 つの方法があります。

- jar ファイルとして配置します (Servlet 2.1 以前をサポートする Web サーバーの場合)。
- Web アーカイブ (WAR) として配置します (Servlet 2.2 以降をサポートする Web サーバーの場合)。

jar ファイルとして配置する

Message Queue トンネルサーブレットを配置するには、ホスト Web サーバーへアクセス可能な適切な jar ファイルを作成し、起動時にサーブレットを読み込むように Web サーバーを設定して、サーブレットの URL のコンテキストルート部分を指定します。

トンネルサーブレットの jar ファイル (imqservlet.jar) には、HTTP トンネルサーブレットが必要とするすべてのクラスが含まれます。このファイルは、オペレーティングシステムに応じて該当するディレクトリに格納されています (付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照)。

Servlet 2.x をサポートする Web サーバーは、このサーブレットの読み込みに使用できます。サーブレットのクラス名は次のとおりです。

```
com.sun.messaging.jmq.transport.
httptunnel.servlet.HttpTunnelServlet
```

Web サーバーは、imqservlet.jar ファイルを参照する必要があります。Web サーバーとブローカを異なるホストで実行する場合は、Web サーバーがアクセスできる場所に、imqservlet.jar ファイルのコピーを置く必要があります。

また、起動時にこのサーブレットを読み込むように Web サーバーを設定する必要があります。サーブレットの URL のコンテキストルート部分を指定する必要が生じる場合もあります (392 ページの「例 1: HTTP トンネルサーブレットを Sun Java System Web サーバーに配置する」を参照)。

パフォーマンスを向上させるために、Web サーバーのアクセスロギング機能を無効にしておくことをお勧めします。

Web アーカイブファイルとして配置する

HTTP トンネルサーブレットを WAR ファイルとして配置する作業は、Web サーバーから提供されている配置メカニズムを使用することで成り立っています。HTTP トンネルサーブレットの WAR ファイル (imqhttp.war) は、オペレーティングシステムに応じて、.jar、.war、.rar の各ファイルを含むディレクトリに配置されています (付録 A「オペレーティングシステムごとの 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設定情報などの配置記述子が含まれています。Web 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 (395 ページの「例 2: HTTP トンネルサーブレットを Sun Java System Application Server 7.0 に配置する」を参照)。

手順 2: httpjms コネクション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 コネクション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。設定し直すと、67 ページの「ブローカのインタラクティブな起動」で説明されているように、ブローカを起動できます。

➤ httpjms コネクション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*) によって識別されたディレクトリに書き込まれます (付録 A「オペレーティングシステムごとの 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。ただし、リモートトンネルサーブレットにアクセスするには、servletHost と servletPort コネクションサービスプロパティを設定し直します。

パフォーマンスを向上させるために、pullPeriod プロパティも設定し直します。httpjms コネクションサービス設定プロパティについては、389 ページの表 C-1 を参照してください。

表 C-1　　httpjms コネクションサービスのプロパティ

プロパティ名	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 : `localhost`
`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を変更します。デフォルト値 : `7675`

表 C-1　　httpjms コネクションサービスのプロパティ (続き)

プロパティ名	説明
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、可能な限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 転送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 : -1
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。このプロパティは、ブローカが HTTP トンネルサーブレットと通信した後、コネクションを開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 : 60

手順 3: HTTP コネクションを設定する

クライアントアプリケーションは、設定済みのコネクション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 コネクションを確立する必要があります。この節では、HTTP コネクション設定の問題点を説明します。

コネクションファクトリを設定する

HTTP サポートを有効にするには、コネクションファクトリの 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:port/contextRoot*/tunnel

hostName:port は、HTTP トンネルサーブレットをホスティングする 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。

コネクション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。

コネクション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。

- コネクション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 (189 ページの「コネクションファクトリの追加」を参照)、管理コンソール (`imqadmin`) を使用してコネクションファクトリ管理対象オブジェクト作成時に属性を設定します。
- クライアント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)。
- プログラム的にクライアントコードに API 呼び出しを作成したあと、これを使用してコネクション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。現在実行中の複数のブローカで、1 つの Web サーバーや HTTP トンネルサーブレット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` コネクションファクトリ属性を設定する必要があります。

`http://`*hostName:port/contextRoot*`/tunnel?ServerName=`*bkrHostName*`:`*instanceName*

bkrHostName の部分にはブローカインスタンスのホスト名が入り、*instanceName* の部分に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名が入ります。

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 2002
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を設定できます。

例 1: HTTP トンネルサーブレットを Sun Java System Web サーバーに配置する

ここでは、HTTP トンネルサーブレットを Sun Java System Web Server に jar ファイルおよび WAR ファイルとして配置する両方の方法を説明します。どちらを使用するかは、Sun Java System Web Server のバージョンによって決まります。Servlet 2.2 またはそれ以降がサポートされていない場合は、WAR ファイルの配置を行えません。

jar ファイルとして配置する

次の手順は、ブラウザベースの管理 GUI を使用した Sun Java System Web Server 6.1 への配置を説明しています。この方法では、通常次の手順を実行します。

1. サーブレットを追加する
2. サーブレットの仮想パスを設定する
3. サーブレットを読み込む
4. サーブレットアクセスログを無効にする

次の項で、これらの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。

サーブレットを追加する

➤ **トンネルサーブレットを追加する**

1. 「Servlet」タブを選択します。
2. 「Configure Servlet Attributes」を選択します。
3. 「Servlet Name」フィールドに、トンネルサーブレットの名前を指定します。

4. 「Servlet Code (class name)」フィールドに次の値を設定します。

   `com.sun.messaging.jmq.transport.httptunnel.servlet.HttpTunnelServlet`

5. 「Servlet Classpath」フィールドに `imqservlet.jar` への絶対パスを入力します。たとえば、次のように指定します。

   `/usr/share/lib/imq/imqservlet.jar` (Solaris)
   `/opt/sun/mq/share/lib/imqservlet.jar` (Linux)
   `IMQ_HOME/lib/imqservlet.jar` (Windows)

6. 「Servlet args」フィールドに、表 C-2 に示すオプションの引数を入力します。

表 C-2 HTTP トンネルサーブレット jar ファイルの配置に使用するサーブレット引数

引数	デフォルト値	参照先
servletHost	all hosts	389 ページの表 C-1 を参照してください。
servletPort	`7675`	389 ページの表 C-1 を参照してください。

両方の引数を使用する場合は、次のように引数をコンマで区切ります。

`servletPort=`*`portNumber`*`, servletHost=...`

`serverHost` 引数と `serverPort` 引数は、Web サーバーとブローカ間の通信にだけ適用され、またデフォルト値に問題があるときにだけ設定します。ただしその場合、ブローカ設定プロパティを設定する必要があります (389 ページの表 C-1 を参照)。たとえば、次のように設定します。

`imq.httpjms.http.servletPort`

サーブレット仮想パス (サーブレット URL) を設定する

➤ **トンネルサーブレットの仮想パス (サーブレット URL) を設定する**

1. 「Servlet」タブを選択します。
2. 「Configure Servlet Virtual Path Translation」を選択します。
3. 「Virtual Path」フィールドを設定します。

   仮想パスは、トンネルサーブレット URL の */contextRoot*`/tunnel` 部分です。

   `http://`*`hostName`*`:`*`port`*`/`*`contextRoot`*`/tunnel`

   たとえば、*contextRoot* を `imq` に設定すると、「Virtual Path」フィールドは次のようになります。

   `/imq/tunnel`

4. 「Servlet Name」フィールドに 392 ページの「サーブレットを追加する」の手順 3 と同じ値を設定します。

サーブレットを読み込む

➤ **Web サーバーの起動時にトンネルサーブレットを読み込む**

1. 「Servlet」タブを選択します。
2. 「Configure Global Attributes」を選択します。
3. 「Startup Servlets」フィールドに、392 ページの「サーブレットを追加する」の手順 3 と同じサーブレット名の値を入力します。

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。
3. 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。

WAR ファイルとして配置する

次の手順では、Sun Java System Web Server 6.0 Service Pack 2 での配置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」フィールドでは、`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応じて異なるディレクトリに格納されています (付録 A「オペレーティングシステムごとの 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*`:`*port*`/`*contextRoot*`/tunnel`

   たとえば、*contextRoot* を `imq` に設定すると、「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*`:`*port*`/`*contextRoot*`/tunnel`

クライアントはこの URL を使用して、HTTP コネクションを使用しているメッセージサービスに接続できます。

例 2: HTTP トンネルサーブレットを Sun Java System Application Server 7.0 に配置する

この節では、HTTP トンネルサーブレットを WAR ファイルとして Sun Java System Application Server 7.0 に配置する方法を説明します。

2 段階の手順が必要です。

- Application Server 7.0 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 7.0 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`) の場所を入力します。

   `imqhttp.war` ファイルの場所は、使用中のオペレーティングシステムによって異なります (付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照)。

4. 「OK」をクリックします。

5. 次の画面で、「Context Root」テキストフィールドの値を設定します。

   「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*`:`*port*`/`*contextRoot*`/tunnel`

   たとえば、「Context Root」フィールドは `/imq` に設定できます。

6. 「OK」をクリックします。

   次の画面は、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を示しています。

   ```
   /var/opt/SUNWappserver7/domains/domain1/server1/applications/
   j2ee-modules/imqhttp_1
   ```

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*`:`*port*`/`*contextRoot*`/tunnel`

クライアントはこの URL を使用して、HTTP コネクションを使用している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 7.0 は、変更されないかぎりは、強制的にデフォルトのセキュリティ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 ブローカからのコネクション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/SUNWappserver7/domains/domain1/server1/config/
server.policy
```

トンネルサーブレットに Message Queue ブローカからのコネクション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/SUNWappserver7/domains/domain1/server1/
             applications/j2ee-modules/imqhttp_1/-"
{
    permission java.net.SocketPermission "*",
             "connect,accept,resolve";
};
```

HTTPS サポートの有効化

次に、HTTPS サポートを有効化するのに必要な手順を説明します。この手順は、387 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する。
2. HTTPS トンネルサーブレットを Web サーバーに配置する。
3. ブローカの httpsjms コネクションサービスを設定し、ブローカを起動する。
4. HTTPS コネクションを設定する。

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 コネクションサービスのアーキテクチャでは、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。

`imqkeytool` ユーティリティを実行し、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。

```
imqkeytool -servlet keystore_location
```

ユーティリティが、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 `imqkeytool` を実行する必要があります。

`imqkeytool` 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ために、そのパスワードを入力する必要があります。

`imqkeytool` を実行すると、JDK `keytool` ユーティリティが実行されて、自己署名型証明書が生成されます。生成された証明書は、*keystore_location* 引数で指定される場所にある、Message Queue のキーストアファイルに配置されます。キーストアは、JDK1.2 `keytool` でサポートされているのと同じキーストアの形式になっています。

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。*keystore_location* 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確実に移動またはコピーしてください (399 ページの「手順 2: HTTPS トンネルサーブレットを Web サーバーに配置する」を参照)。

手順 2: HTTPS トンネルサーブレットを Web サーバーに配置する

HTTPS トンネルサーブレットを Web サーバーに配置するには、通常、次の 2 つの方法があります。

- jar ファイルとして配置します (Servlet 2.1 以前をサポートする Web サーバーの場合)。
- Web アーカイブ (WAR) として配置します (Servlet 2.2 以降をサポートする Web サーバーの場合)。

どちらの場合も、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ます。

jar ファイルとして配置する

Message Queue トンネルサーブレットを配置するには、ホスト Web サーバーへアクセス可能な適切な jar ファイルを作成し、起動時にサーブレットを読み込むように Web サーバーを設定して、サーブレットの URL のコンテキストルート部分を指定します。

トンネルサーブレットの jar ファイル (`imqservlet.jar`) には、HTTPS トンネルサーブレットが必要とするすべてのクラスが含まれます。このファイルは、オペレーティングシステムに応じて該当するディレクトリに格納されています (付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照)。

Servlet 2.x をサポートする Web サーバーは、このサーブレットの読み込みに使用できます。サーブレットのクラス名は次のとおりです。

```
com.sun.messaging.jmq.transport.
httptunnel.servlet.HttpsTunnelServlet
```

Web サーバーは、`imqservlet.jar` ファイルを参照する必要があります。Web サーバーとブローカを異なるホストで実行する場合は、Web サーバーがアクセスできる場所に、`imqservlet.jar` ファイルのコピーを置く必要があります。

また、起動時にこのサーブレットを読み込むように Web サーバーを設定する必要があります。サーブレットの URL のコンテキストルート部分を指定する必要が生じる場合もあります (405 ページの「例 3: HTTPS トンネルサーブレットを Sun Java System Web サーバーに配置する」を参照)。

サーブレットを Web サーバーで実行するために、JSSE jar ファイルがクラスパスにあることを確認します。確認方法については、Web サーバーのマニュアルを参照してください。

Web サーバーの設定で重要な点は、自己署名型証明書の場所とパスワードを HTTPS トンネルサーブレットが使用するように指定し、ブローカとの安全なコネクションを確立することです。398 ページの「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」で作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置く必要があります。

パフォーマンスを向上させるために、Web サーバーのアクセスロギング機能を無効にしておくことをお勧めします。

Web アーカイブファイルとして配置する

HTTPS トンネルサーブレットを WAR ファイルとして配置する作業は、Web サーバーから提供されている配置メカニズムを使用することで成り立っています。HTTPS トンネルサーブレットの WAR ファイル (`imqhttps.war`) は使用中のオペレーティングシステムに応じて異なるディレクトリに格納されています (付録 A「オペレーティングシステムごとの 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設定情報などの配置記述子が含まれています。Web 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 (409 ページの「例 4: HTTPS トンネルサーブレットを Sun Java System Application Server 7.0 に配置する」を参照)。

ただし、`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 (398 ページの「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」を参照)。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、キーストアの場所を指定する必要があります。

手順 3: httpsjms コネクション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 コネクション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。設定し直すと、67 ページの「ブローカのインタラクティブな起動」で説明されているように、ブローカを起動できます。

➤ httpsjms コネクション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*) によって識別されたディレクトリに書き込まれます (付録 A「オペレーティングシステムごとの 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。ただし、リモートトンネルサーブレットにアクセスするには、servletHost と servletPort コネクションサービスプロパティを設定し直します。

パフォーマンスを向上させるために、pullPeriod プロパティも設定し直します。httpsjms コネクションサービス設定プロパティについては、表 C-3 を参照してください。

表 C-3　　httpsjms コネクションサービスのプロパティ

プロパティ名	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 : `localhost`
`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を変更します。デフォルト値 : `7674`

表 C-3　　httpsjms コネクションサービスのプロパティ (続き)

プロパティ名	説明
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、可能な限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 転送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 : -1
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。このプロパティは、ブローカが HTTPS トンネルサーブレットと通信した後、コネクションを開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 : 60

手順 4: HTTPS コネクションを設定する

クライアントアプリケーションは、適切に設定されたコネクション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 コネクション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」を参照するか、あるいは「root 証明書をインポートする」に進みます。

これらの問題点が解決すると、HTTPS コネクションの設定に進みます。

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 セキュリティプロバイダを静的に JRE_HOME/lib/security/java.security ファイルに追加します。次を追加します。

   security.provider.*n*=com.sun.net.ssl.internal.ssl.Provider

   ここで、*n* には、セキュリティプロバイダパッケージが次に利用可能な優先順位を指定します。

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、または専用の Web サーバー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、次の手順に従うか、あるいは「コネクションファクトリを設定する」を参照します。

証明書が *cert_file* に保存され、*trust_store_file* 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 alias_for_certificate -file cert_file
-keystore trust_store_file
```

次の質問に YES と答えます。Trust this certificate?

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 プロパティを指定する必要もあります。

javax.net.ssl.trustStore=*trust_store_file*

javax.net.ssl.trustStorePassword=*trust_store_passwd*

コネクションファクトリを設定する

HTTPS サポートを有効にするには、コネクションファクトリの 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:port/contextRoot*/tunnel

hostName:port は、HTTPS トンネルサーブレットをホスティングする 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。

コネクション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。

コネクション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。

- コネクション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 (189 ページの「コネクションファクトリの追加」を参照)、管理コンソール (`imqadmin`) を使用してコネクションファクトリ管理対象オブジェクト作成時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コネクションファクトリを作成してから、API 呼び出しを使用してコネクション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` コネクションファクトリ属性を設定する必要があります。

`https://`*hostName:port*`/`*contextRoot*`/tunnel?ServerName=`*bkrHostName*`:`*instanceName*

bkrHostName の部分にはブローカインスタンスのホスト名が入り、*instanceName* の部分に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名が入ります。

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を設定できます。

例 3: HTTPS トンネルサーブレットを Sun Java System Web サーバーに配置する

ここでは、HTTPS トンネルサーブレットを Sun Java System Web Server に jar ファイルおよび WAR ファイルとして配置する両方の方法を説明します。どちらを使用するかは、Sun Java System Web Server のバージョンによって決まります。Servlet 2.2 またはそれ以降がサポートされていない場合は、WAR ファイルの配置を行えません。

jar ファイルとして配置する

次の手順は、ブラウザベースの管理 GUI を使用した Sun Java System Web Server 6.1 への配置を説明しています。この方法では、通常次の手順を実行します。

1. サーブレットを追加する
2. サーブレットの仮想パスを設定する
3. サーブレットを読み込む
4. サーブレットアクセスログを無効にする

次の項で、これらの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、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。

サーブレットを追加する

➤ **トンネルサーブレットを追加する**

1. 「Servlet」タブを選択します。
2. 「Configure Servlet Attributes」を選択します。
3. 「Servlet Name」フィールドに、トンネルサーブレットの名前を指定します。

4. 「Servlet Code (class name)」フィールドに次の値を設定します。

   ```
   com.sun.messaging.jmq.transport.
   httptunnel.servlet.HttpsTunnelServlet
   ```

5. 「Servlet Classpath」フィールドに `imqservlet.jar` への絶対パスを入力します。たとえば、次のように指定します。

   `/usr/share/lib/imq/imqservlet.jar` (Solaris)

   `/opt/sun/mq/share/lib/imqservlet.jar` (Linux)

   `IMQ_HOME/lib/imqservlet.jar` (Windows)

6. 「Servlet args」フィールドに、表 C-4 に示す必要なオプションの引数を入力します。

表 C-4 HTTPS トンネルサーブレット jar ファイルの配置に使用するサーブレット引数

引数	デフォルト値	必須 / オプション
`keystoreLocation`	なし	必須
`keystorePassword`	なし	必須
`servletHost`	all hosts	オプション
`servletPort`	`7674`	オプション

引数はコンマで区切ります。たとえば、次のように指定します。

`keystoreLocation=`*keystore_location*`,keystorePassword=`*keystore_password*`, servletPort=`*portnumber*

`serverHost` 引数と `serverPort` 引数は、Web サーバーとブローカ間の通信にだけ適用され、またデフォルト値に問題があるときにだけ設定します。ただしその場合、ブローカ設定プロパティを設定する必要があります (401 ページの表 C-3 を参照)。たとえば、次のように指定します。

```
imq.httpsjms.https.servletPort
```

サーブレット仮想パス (サーブレット URL) を設定する

➤ **トンネルサーブレットの仮想パス (サーブレット URL) を設定する**

1. 「Servlet」タブを選択します。
2. 「Configure Servlet Virtual Path Translation」を選択します。

3. 「Virtual Path」フィールドを設定します。

   仮想パスは、トンネルサーブレット URL の */contextRoot*/tunnel 部分です。

   ```
   https://hostName:port/contextRoot/tunnel
   ```

   たとえば、*contextRoot* を imq に設定すると、「Virtual Path」フィールドは次のようになります。

   ```
   /imq/tunnel
   ```

4. 「Servlet Name」フィールドに 405 ページの「サーブレットを追加する」の手順 3 と同じ値を設定します。

サーブレットを読み込む

➤ **Web サーバーの起動時にトンネルサーブレットを読み込む**

1. 「Servlet」タブを選択します。
2. 「Configure Global Attributes」を選択します。
3. 「Startup Servlets」フィールドに、405 ページの「サーブレットを追加する」の手順 3 と同じサーブレット名の値を入力します。

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。
3. 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。

WAR ファイルとして配置する

次の手順では、Sun Java System Web Server 6.0 Service Pack 2 での配置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。これを確実にする一番簡単な方法は、jsse.jar、jnet.jar、および jcert.jar を IWS60_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配置する前に、配置記述子がキーストアファイルの配置場所を指し、キーストアパスワードを指定するように変更する必要もあります。

➤ HTTPS トンネルサーブレット WAR ファイルを修正する

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。

➤ 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. 修正済みの `imqhttps.war` ファイルを指すように、「WAR File On and WAR File Path」フィールドに適切な値を選択します (408 ページの「HTTPS トンネルサーブレット WAR ファイルを修正する」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*`:`*port*`/`*contextRoot*`/tunnel`

   たとえば、*contextRoot* を `imq` に設定すると、「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。

`https://`*hostName*`:`*port*`/imq/tunnel`

クライアントはこの URL を使用して、安全な HTTPS コネクションを使用しているメッセージサービスに接続できます。

例 4: HTTPS トンネルサーブレットを Sun Java System Application Server 7.0 に配置する

この節では、HTTPS トンネルサーブレットを WAR ファイルとして Sun Java System Application Server 7.0 に配置する方法を説明します。

2 段階の手順が必要です。

- Application Server 7.0 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 7.0 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) の場所を入力します。

   imqhttps.war ファイルの場所は、使用中のオペレーティングシステムによって異なります (付録 A「オペレーティングシステムごとの Message Queue データの場所」を参照)。

4. 「OK」をクリックします。

5. 次の画面で、「Context Root」テキストフィールドの値を設定します。

   「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:port/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
   ```

6. 「OK」をクリックします。

   次の画面は、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を示しています。

   ```
   /var/opt/SUNWappserver7/domains/domain1/server1/applications/
   j2ee-modules/imqhttps_1
   ```

サーブレットは次のアドレスで利用可能となります。

```
https://hostName:port/contextRoot/tunnel
```

クライアントはこの URL を使用して、HTTPS コネクションを使用している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 7.0 は、変更されないかぎりは、強制的にデフォルトのセキュリティ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 ブローカからのコネクション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/SUNWappserver7/domains/domain1/server1/config/
server.policy
```

トンネルサーブレットに Message Queue ブローカからのコネクション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/SUNWappserver7/domains/domain1/server1/
             applications/j2ee-modules/imqhttps_1/-"
{
    permission java.net.SocketPermission "*",
             "connect,accept,resolve";
};
```

トラブルシューティング

この節では、HTTP や HTTPS コネクション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、すべてのコネクションが復元され、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コネクション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コネクションを復元し、ブローカコネクションを待機しますが、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、コネクション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。

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/doc/819-1933?l=ja) 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

ま

め

ゆ

ら

り

る

ろ