管理者ガイド

Sun™ ONE Message Queue

Version 3.0

816-6457-10
2002 年 6 月

Copyright © 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

Sun Microsystems, Inc. は、この製品に含まれるテクノロジに関する知的所有権を保持しています。特に限定されることなく、これらの知的所有権は http://www.sun.com/patents に記載されている 1 つ以上の米国特許および米国およびその他の国における 1 つ以上の追加特許または特許出願中のものが含まれている場合があります。

本書で説明されている製品は著作権法により保護されており、その使用、複製、頒布および逆コンパイルを制限するライセンスのもとにおいて頒布されます。Sun および Sun のライセンサーの書面による事前の許可なく、本製品および関連する文書のいかなる部分も、いかなる方法によっても複製することが禁じられます。

フォントテクノロジを含む第三者のソフトウェアの著作権は Sun の提供者により保護されており、ライセンス許諾されています。

Sun、Sun Microsystems、Sun のロゴマーク、Java、Solaris、iPlanet、JDK、Java Naming and Directory Interface、Java Coffee Cup のロゴおよび iPlanet のロゴマークは米国およびその他の国における米国 Sun Microsystems, Inc. (以下、米国 Sun Microsystems 社とします) の商標もしくは登録商標です。

すべての SPARC の商標はライセンスの基づいて使用され、米国およびその他の国における SPARC International, Inc. の商標もしくは登録商標です。SPARC の商標に関連する製品は Sun Microsystems, Inc. によって開発されたアーキテクチャに基づいています。

UNIX は、X/Open Company, Ltd が独占的にライセンスしている米国およびその他の国における登録商標です。

Federal Acquisitions: Commercial Software—Government Users Subject to Standard License Terms and Conditions.

目 次

図目次 **11**

表目次 **13**

手順一覧 **15**

はじめに **17**
マニュアルの対象読者 17
マニュアルの構成 18
マニュアルの表記規則 19
テキストの表記規則 19
環境変数の表記規則 20
関連マニュアル 22
MQ マニュアルセット 22
オンラインヘルプ 22
JavaDoc 23
クライアントアプリケーションの例 23
Java Message Service (JMS) 仕様書 23

第 1 章 概要 **25**
Sun ONE Message Queue とは 25
製品エディション 26
エンタープライズメッセージングシステム 27
エンタープライズメッセージングシステムの要件 27
集中メッセージングとピアツーピアメッセージング 28
メッセージングシステムの概念 30
メッセージ 30
メッセージサービスのアーキテクチャ 30
メッセージ配信モデル 31

JMS 仕様 32
JMS メッセージ構造 32
JMS プログラミングモデル 32
管理対象オブジェクト 34
JMS/J2EE プログラミング : メッセージ駆動型 Beans 35
メッセージ駆動型 Beans 35
アプリケーションサーバのサポート 37
JMS メッセージングの問題点 37
JMS プロバイダへの非依存性 37
プログラミングドメイン 38
クライアント識別子 39
信頼性のあるメッセージング 40
通知およびトランザクション 40
持続ストレージ 42
パフォーマンスのかね合い 42
メッセージの選択 43
メッセージの順番と優先度 43

第 2 章 MQ メッセージングシステム 45
MQ メッセージサーバ 46
ブローカ 46
コネクションサービス 48
メッセージルーター 52
持続マネージャ 56
セキュリティマネージャ 59
ロガー 64
物理的な送信先 67
キューの送信先 68
トピックの送信先 69
自動作成 (および管理者によって作成) された送信先 69
一時送信先 70
複数ブローカの設定 (クラスタ) 71
複数ブローカのアーキテクチャ 71
開発環境でのクラスタの使用 74
クラスタ設定プロパティ 74
MQ クライアントランタイム 76
メッセージのプロデュース 77
メッセージのコンシューム 77
MQ 管理対象オブジェクト 78
コネクションファクトリ管理対象オブジェクト 80
送信先管理対象オブジェクト 81
クライアントの起動時の属性値のオーバーライド 81

第 3 章 MQ の管理 83
MQ の管理タスク 83
開発環境 83
運用環境 84
セットアップ操作 84
メンテナンス操作 84
MQ 管理ツール 85
管理コンソール 86
コマンド行ユーティリティの概要 86
コマンド行の構文 88
共通のコマンド行オプション 88

第 4 章 管理コンソールのチュートリアル 91
準備 92
管理コンソールを起動する 92
ヘルプを利用する 94
ブローカを操作する 95
ブローカの起動 96
ブローカの追加 97
管理者パスワードの変更 98
ブローカに接続する 99
コネクションサービスを表示する 99
ブローカに物理的送信先を追加する 101
物理的送信先を操作する 103
トピック送信先に関する情報を取得する 104
オブジェクトストアを操作する 105
オブジェクトストアを追加する 105
オブジェクトストアプロパティを確認する 108
オブジェクトストアに接続する 108
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 108
送信先管理対象オブジェクトを追加する 110
管理対象オブジェクトのプロパティ 112
コンソール情報を更新する 112
サンプルアプリケーションを実行する 113

第 5 章 ブローカの起動と設定 115
設定ファイル 115
プロパティ値のマージ 116
プロパティ命名構文 117
インスタンス設定ファイルの編集 118
ブローカの起動 122
ブローカクラスタの操作 126
クラスタ設定プロパティ 126

ブローカの接続 128
方法 1: クラスタ設定ファイルを使用しない方法 128
方法 2: クラスタ設定ファイルを使用する方法 129
クラスタへのブローカの追加 129
クラスタ内のブローカの再起動 130
クラスタからのブローカの削除 130
マスターブローカの設定変更レコードのバックアップ 131
マスターブローカの設定変更レコードの復元 131
ログ作成 132
デフォルトのログ作成の設定 132
ログメッセージの書式設定 133
ロガー設定の変更 133
出力チャネルの変更 134
ロールオーバー条件の変更 135
ブローカパフォーマンスメトリックスのログ作成 135

第 6 章 ブローカとアプリケーションの管理 139
コマンドユーティリティ 140
コマンドの構文 140
imqcmd のサブコマンド 140
imqcmd のオプションの概要 142
imqcmd を使用するための前提条件 144
例 144
ブローカの状態の制御 145
ブローカのプロパティのクエリと更新 146
ブローカのクエリ 147
ブローカの更新 148
コネクションサービスの管理 150
コネクションサービスの一覧表示 152
サービスのプロパティのクエリと更新 152
サービスの停止および再開 154
送信先の管理 154
送信先の作成 155
送信先に関する情報の入手 156
送信先の更新 156
送信先のパージ 157
送信先の破棄 157
永続サブスクリプションの管理 158
トランザクションの管理 160

第 7 章 管理対象オブジェクトの管理 163
オブジェクトストアについて 164

管理対象オブジェクト 164
オブジェクトマネージャユーティリティ (imqobjmgr) 165
コマンドの構文 165
imqobjmgr のサブコマンド 165
imqobjmgr コマンドのオプションの概要 166
必要な情報 167
管理対象オブジェクトの属性 168
コネクションファクトリ管理対象オブジェクト 168
送信先管理対象オブジェクト 170
オブジェクトストアの属性 171
初期コンテキストおよび場所の情報 171
セキュリティ情報 (LDAP のみ) 171
入力ファイルの使用 172
管理対象オブジェクトの追加および削除 175
コネクションファクトリ管理対象オブジェクトの追加 175
トピックまたはキューの管理対象オブジェクトの追加 176
管理対象オブジェクトの削除 178
情報の入手 179
管理対象オブジェクトの一覧表示 179
単一オブジェクトの情報 180
管理対象オブジェクトの更新 181

第 8 章 セキュリティ管理 183
ユーザの認証 184
単層型ファイルユーザリポジトリを使用する 184
MQ User Manager サブコマンドとオプション 185
グループ 186
状態 187
ユーザ名とパスワードの形式 187
ユーザリポジトリの設定と管理 188
デフォルトの管理者パスワードの変更 189
ユーザリポジトリに LDAP サーバを使用する 190
ユーザの承認 : アクセス制御プロパティファイル 193
アクセス規則の構文 194
アクセス権の計算 195
コネクションアクセス制御 196
送信先アクセス制御 197
送信先自動作成アクセス制御 198
暗号化 : SSL サービスの操作 199
TCP/IP を介した SSL サービスの設定 199
手順 1. 自己署名型証明書の生成 200
手順 2. ブローカでの SSL ベースのサービスを有効にする 201
手順 3. ブローカを起動する 202

手順 4. SSL ベースのクライアントを設定および実行する 203
HTTP を介した SSL サービスの設定 204
passfile の使用 205

付録 A プラグイン持続の設定 207
概要 207
JDBC アクセスが可能なデータストアへの接続 208
JDBC 関連のブローカの設定プロパティ 209
データベース管理ユーティリティ (imqdbmgr) 212

付録 B HTTP/HTTPS サポート 215
HTTP/HTTPS サポートのアーキテクチャ 215
HTTP サポートの実装 217
手順 1. HTTP トンネルサーブレットを Web サーバに配置する 217
jar ファイルとして配置する 217
Web アーカイブファイルとして配置する 218
手順 2. httpjms コネクションサービスを設定する 218
手順 3. HTTP コネクションを設定する 220
コネクションファクトリの属性の設定 220
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 220
HTTP プロキシを使用する 221
例 : HTTP トンネルサーブレットを配置する 221
jar ファイルとして配置する 221
WAR ファイルとして配置する 223
HTTPS サポートを実装する 225
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 225
手順 2. HTTPS トンネルサーブレットを Web サーバに配置する 226
jar ファイルとして配置する 226
Web アーカイブファイルとして配置する 227
手順 3. httpsjms コネクションサービスを設定する 228
手順 4. HTTPS コネクションを設定する 229
JSSE を設定する 229
root 証明書をインポートする 230
コネクションファクトリの属性を設定する 230
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 231
HTTP プロキシを使用する 231
例 : HTTPS トンネルサーブレットを配置する 232
jar ファイルとして配置する 232
WAR ファイルとして配置する 234

付録 C Windows のサービスとしてのブローカの使用 237
Windows のサービスとしてのブローカの実行 237

サービス管理ユーティリティ (imqsvcadmin) 238
コマンドの構文 238
imqsvcadmin のサブコマンド 238
imqsvcadmin のオプションの概要 239
ブローカサービスの削除 240
ブローカサービスの再設定 240
代替 Java ランタイムの使用 240
ブローカサービスのクエリ 240
トラブルシューティング 241

付録 D MQ データの場所 243

付録 E MQ インタフェースの安定度 245

用語集 249

索引 253

図目次

図 1-1　集中メッセージングとピアツーピアメッセージング 29
図 1-2　メッセージサービスのアーキテクチャ 31
図 1-3　JMS プログラミングオブジェクト 33
図 1-4　MDB を使用したメッセージング 36
図 2-1　MQ システムアーキテクチャ 45
図 2-2　ブローカのコンポーネント 47
図 2-3　コネクションサービスのサポート 49
図 2-4　持続マネージャのサポート 57
図 2-5　セキュリティマネージャのサポート 61
図 2-6　ロギングスキーマ 64
図 2-7　複数ブローカ (クラスタ) のアーキテクチャ 72
図 2-8　メッセージングの処理 76
図 2-9　MQ クライアントランタイムへのメッセージの配信 78
図 3-1　ローカルおよびリモートの管理ユーティリティ 86
図 5-1　ブローカ設定ファイル 117
図 B-1　HTTP/HTTPS サポートのアーキテクチャ 216

表目次

表 1　マニュアルの内容 18
表 2　マニュアルの表記規則 19
表 3　MQ 環境変数 20
表 4　MQ マニュアルセット 22
表 1-1　JMS プログラミングオブジェクト 38
表 2-1　ブローカの主要なコンポーネントと機能 47
表 2-2　ブローカでサポートされているコネクションサービス 48
表 2-3　コネクションサービスのプロパティ 50
表 2-4　メッセージルーターのプロパティ 55
表 2-5　持続のプロパティ 58
表 2-6　セキュリティのプロパティ 62
表 2-7　ロギングのカテゴリ 64
表 2-8　ロガーのプロパティ 66
表 2-9　自動作成の設定プロパティ 70
表 2-10　クラスタ設定プロパティ 74
表 2-11　送信先の属性 81
表 3-1　共通の MQ コマンド行オプション 89
表 5-1　ブローカインスタンス設定プロパティ 118
表 5-2　imqbrokerd オプション 123
表 5-3　クラスタ設定プロパティ 126
表 5-4　imqbrokerd ロガーオプションと対応するプロパティ 133
表 5-5　コネクションサービスに作成されたメトリックス 136
表 5-6　各ブローカに収集されたメトリックス 137
表 6-1　imqcmd のサブコマンドと引数 140
表 6-2　imqcmd のオプション 142
表 6-3　ブローカを制御する imqcmd のサブコマンド 146
表 6-4　情報の入手およびブローカの更新を行う imqcmd のサブコマンド 146

表 6-5　ブローカのプロパティ 148
表 6-6　コネクションサービスを管理する imqcmd のサブコマンド 150
表 6-7　ブローカでサポートされているコネクションサービス 151
表 6-8　コネクションサービスの属性 153
表 6-9　送信先を管理する imqcmd のサブコマンド 154
表 6-10 送信先の属性 155
表 6-11 永続サブスクリプションを管理する imqcmd のサブコマンド 158
表 6-12 トランザクションを管理する imqcmd のサブコマンド 160
表 7-1　imqobjmgr のサブコマンド 165
表 7-2　imqobjmgr のオプション 166
表 7-3　コネクションファクトリ属性 169
表 7-4　送信先の属性 170
表 7-5　オブジェクトストアのセキュリティ属性 172
表 8-1　ユーザリポジトリでの初期エントリ 184
表 8-2　imqusermgr サブコマンド 185
表 8-3　imqusermgr オプション 186
表 8-4　ユーザ名とパスワードに無効な文字 187
表 8-5　LDAP 関連プロパティ 190
表 8-6　アクセス規則の構文要素 194
表 8-7　送信先アクセス制御規則の要素 197
表 8-8　キーストアのプロパティ 201
表 8-9　passfile のパスワード 205
表 A-1　JDBC 関連のプロパティ 209
表 A-2　imqdbmgr のサブコマンド 212
表 A-3　imqdbmgr のオプション 213
表 B-1　httpjms コネクションサービスのプロパティ 219
表 B-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 222
表 B-3　httpsjms コネクションサービスのプロパティ 228
表 B-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 233
表 C-1　imqsvcadmin のサブコマンド 238
表 C-2　imqsvcadmin のオプション 239
表 D-1　MQ 3.0 データの場所 243
表 E-1　MQ 3.0 インタフェースの安定度 245
表 E-2　インタフェースの安定度の分類方式 247

手順一覧

管理コンソールを起動するには 92
管理コンソールのヘルプ情報を表示するには 94
ブローカを起動するには 96
管理コンソールにブローカを追加するには 97
管理者パスワードを変更するには 98
ブローカに接続するには 99
利用可能なコネクションサービスを表示するには 99
ブローカにキュー送信先を追加するには 101
物理的送信先のプロパティを表示するには 103
送信先からメッセージをパージするには 104
送信先を削除するには 104
ファイルシステムオブジェクトストアを追加するには 105
オブジェクトストアのプロパティを表示するには 108
オブジェクトストアに接続するには 108
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には 109
送信先をオブジェクトストアに追加するには 110
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には 112
`SimpleJNDIClient` アプリケーションを実行するには 113
ブローカをクラスタに接続するには 128
クラスタ設定ファイルを使用している場合に、ブローカをクラスタに追加するには 129
障害時にマスターブローカを復元するには 131
ブローカのロガー設定を変更するには 133
設定ファイルを編集して、LDAP サーバを使用するには 190
ssljms コネクションサービスを設定するには 199
キーの組み合わせを生成し直すには 201
JDBC アクセスが可能なデータストアに接続するには 208
HTTP サポートを実装するには 217

httpjms コネクションサービスを有効にするには 218
トンネルサーブレットを追加するには 222
トンネルサーブレットの仮想パス (サーブレット URL) を設定するには 223
Web サーバの起動時にトンネルサーブレットを読み込むには 223
サーバのアクセスログを無効にするには 223
HTTP トンネルサーブレットを WAR ファイルとして配置するには 224
HTTPS サポートを実装するには 225
httpsjms コネクションサービスを有効にするには 228
JSSE を設定するには 229
トンネルサーブレットを追加するには 232
トンネルサーブレットの仮想パス (サーブレット URL) を設定するには 233
Web サーバの起動時にトンネルサーブレットを読み込むには 234
サーバのアクセスログを無効にするには 234
HTTPS トンネルサーブレット WAR ファイルを修正するには 234
HTTPS トンネルサーブレットを WAR ファイルとして配置するには 235
記録されているサービスのエラーイベントを表示するには 241

はじめに

この『Sun™ ONE Message Queue (MQ) 3.0 管理者ガイド』には、MQ メッセージングシステムの管理タスクを実行するために必要となる基本情報が記載されています。

この章には次の節があります。

- マニュアルの対象読者
- マニュアルの構成
- マニュアルの表記規則
- 関連マニュアル

マニュアルの対象読者

このマニュアルは、MQ 管理タスクを実行する必要があるアプリケーション開発者および管理者を対象としています。

MQ 管理者とは、MQ メッセージングシステム、特にシステムの中枢となる MQ メッセージサーバの設定および管理の担当者です。このマニュアルを読むにあたって、メッセージングシステムの知識や理解は必要とされません。

このマニュアルは、アプリケーション開発者がアプリケーションを最適化する方法をより理解し、MQ メッセージングシステムの機能と柔軟性を有効に使用することを目的としています。

マニュアルの構成

このマニュアルは、読者が全章を読むように構成されています。各章の内容を次の表で簡単に説明します。

表 1　　マニュアルの内容

章	説明
第 1 章「概要」	MQ メッセージングシステムについての高レベルの概念的な概要と用語の説明
第 2 章「MQ メッセージングシステム」	連動してメッセージングサービスを提供する MQ ブローカと MQ クライアントランタイムに重点をおいた、MQ メッセージングシステムの説明
第 3 章「MQ の管理」	MQ 管理タスクとツールの説明、および管理に使用する各コマンド行ユーティリティとそれらの一般的な機能の紹介
第 4 章「管理コンソールのチュートリアル」	MQ メッセージサーバのグラフィカルインタフェースである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の提供
第 5 章「ブローカの起動と設定」	MQ ブローカとブローカクラスタの起動および設定方法の説明
第 6 章「ブローカとアプリケーションの管理」	アプリケーションに依存しない、MQ ブローカの管理に関連したタスク、およびメッセージングアプリケーションを管理するためのタスクの実行方法の説明
第 7 章「管理対象オブジェクトの管理」	MQ 管理対象オブジェクトの作成および管理に関連したタスクの実行方法の説明
第 8 章「セキュリティ管理」	認証、承認、暗号化の管理など、アプリケーションに関連したセキュリティタスクの実行方法の説明
付録 A「プラグイン持続の設定」	JDBC と互換性のあるデータベースを使用して持続機能を実行するように、MQ を設定する方法の説明
付録 B「HTTP/HTTPS サポート」	メッセージングクライアントと MQ メッセージサーバとの間に HTTP コネクションサービスを設定する方法の説明
付録 C「Windows のサービスとしてのブローカの使用」	MQ の Service Administration ユーティリティ (`imqsvcadmin`) を使用して、Windows のサービスとして実行するブローカをインストール、クエリ、および削除する方法の説明
付録 D「MQ データの場所」	さまざまなカテゴリの MQ データの場所の説明
付録 E「MQ インタフェースの安定度」	さまざまな MQ インタフェースの安定性の説明

表 1　　マニュアルの内容 (続き)

章	説明
「用語集」	MQ のマニュアルで使用する用語の定義

マニュアルの表記規則

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

書体	説明
斜体	可変部分に使われる。斜体で表記された項目や値は適宜置き換える必要がある。説明の対象となる語句や項目に対しても使われる
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 (MQ§JSP) を表す
キー + キー	複数のキーストロークはプラス記号で結合する。Ctrl+A は、両方のキーを同時に押すことを表す
キー - キー	連続するキーストロークはハイフンで結合する。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す

環境変数の表記規則

MQ では 3 種類の環境変数が使用されますが、その使用方法は、プラットフォームによって異なります。表 3 では、これらの環境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　　MQ 環境変数

環境変数	説明
IMQ_HOME	インストールされるすべてのファイルが配置されるルート MQ インストールディレクトリ • Solaris の場合、ルート MQ インストールディレクトリは存在しない。IMQ_HOME は MQ ソフトウェアで使用されることも、Solaris 上のファイルの場所を参照するために MQ マニュアルで使用されることもない • Solaris の場合、Sun ONE Application Server の Evaluation Edition では、IMQ_HOME が MQ ソフトウェアで使用されることはないが、ルート MQ インストールディレクトリ (Application Server インストールのルートディレクトリの下にある、imq サブディレクトリ) を参照するために MQ マニュアルでは使用される • Windows の場合、IMQ_HOME は MQ ソフトウェアで使用され、またルート MQ インストールディレクトリを参照するために MQ マニュアルでも使用される。IMQ_HOME の値は、MQ インストーラにより設定される。デフォルトでは C:¥Program Files¥Sun Microsystems¥Message Queue 3.0 • Linux の場合、IMQ_HOME は MQ ソフトウェアでは使用されないが、ルート MQ インストールディレクトリを参照するために MQ マニュアルで使用される。デフォルトでは /opt の下の imq サブディレクトリ

表 3　　MQ 環境変数 (続き)

環境変数	説明
IMQ_VARHOME	MQ 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 ディレクトリを参照する • Solaris の場合、IMQ_VARHOME のデフォルト値は /var/imq ディレクトリだが、ユーザはオプションでこの値を任意のディレクトリに設定できる • Solaris の場合、Sun ONE Application Server の Evaluation Edition では、IMQ_VARHOME のデフォルト値は IMQ_HOME/var だが、ユーザはオプションでこの値を任意のディレクトリに設定できる • Windows の場合、IMQ_VARHOME のデフォルト値は IMQ_HOME/var だが、ユーザはオプションでこの値を任意のディレクトリに設定できる • Linux の場合、IMQ_VARHOME のデフォルト値は IMQ_HOME/var だが、ユーザはオプションでこの値を任意のディレクトリに設定できる
IMQ_JAVAHOME	MQ 実行可能ファイルが必要とする Java ランタイム (JRE 1.4) の場所を参照する • Solaris の場合、IMQ_JAVAHOME のデフォルト値は /usr/j2se/jre ディレクトリだが、ユーザはオプションでこの値を JRE 1.4 が配置されている任意の場所に設定できる • Windows の場合、IMQ_JAVAHOME のデフォルト値は IMQ_HOME/jre だが、ユーザはオプションでこの値を JRE 1.4 が配置されている任意の場所に設定できる • Linux の場合、IMQ_JAVAHOME のデフォルト値は /usr/java/j2sdk1.0/jre ディレクトリだが、ユーザはオプションでこの値を JRE 1.4 が配置されている任意の場所に設定できる

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。すべてのパス名には、UNIX のファイル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

このマニュアル以外にも、MQ には追加のマニュアルが用意されています。

MQ マニュアルセット

MQ 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　　MQ マニュアルセット

マニュアル	対象読者	説明
インストールガイド	開発者および管理者	Solaris、Linux、および Windows の各プラットフォームで MQ ソフトウェアをインストールする方法を説明
リリースノート	開発者および管理者	新機能、制限事項、既知のバグ、および技術的な注意点を説明
開発者ガイド	開発者	MQ の JMS 実装に関連するクイックスタートチュートリアルおよびプログラミング情報を提供
管理者ガイド	管理者。開発者にも推奨	MQ 管理ツールを使用して管理タスクを実行するために必要な基本情報を提供

オンラインヘルプ

MQ 3.0 には、MQ メッセージ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、88 ページの「共通のコマンド行オプション」を参照してください。

また、MQ 3.0 には、グラフィカルユーザインタフェース (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 および MQ API マニュアルは、次の場所にあります。

`IMQ_HOME/javadoc/index.html`
(Solaris の場合は、`/usr/share/javadoc/imq/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 MQ 管理対象オブジェクト用の MQ 固有の API マニュアルが含まれており、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の例

いくつかのアプリケーションの例が次の場所に収録されています。この例には、クライアントアプリケーションのサンプルコードが用意されています。

`IMQ_HOME/demo` (Solaris の場合 `/usr/demo/imq`)

このディレクトリとそのすべてのサブディレクトリ内にある 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、次のサイトで見ることができます。

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

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。

第 1 章

概要

この章では、管理者およびプログラマを対象に、Sun ONE Message Queue (MQ) の概要を説明します。

Sun ONE Message Queue とは

MQ 製品には、アプリケーション間の通信や信頼できるメッセージ配信における問題の標準的なソリューションが用意されています。MQ は、企業向けメッセージングシステムで、Java Message Service (JMS) のオープンスタンダード JMS プロバイダを実装します。

JMS 仕様では、プログラミングインタフェースのセット (32 ページの「JMS プログラミングモデル」を参照) を説明します。このインタフェースには、Java アプリケーションが分散環境でメッセージを作成、送信、受信、および読み込みを行う一般的な方法が用意されています。

Sun ONE Message Queue ソフトウェアを使用すると、異なるプラットフォームやオペレーティングシステムで実行されているプロセスが、共通の MQ メッセージサービス (30 ページの「メッセージサービスのアーキテクチャ」を参照) に接続して、情報を送受信することが可能になります。アプリケーションの開発者は、ネットワーク上のアプリケーションの通信方法に関する低レベルな詳細ではなく、アプリケーションのビジネスロジックに集中できます。

MQ には、JMS 仕様の必要条件を上回る機能が用意されています。たとえば、次のような機能が含まれています。

集中管理 : MQ メッセージサービスと、送信先やセキュリティなど、メッセージングアプリケーション固有の機能を管理するために、コマンド行と GUI ツールの両方が用意されています。

スケーラブルなメッセージサービス : 連携して動作する複数の MQ メッセージサービスコンポーネントであるブローカ間 (マルチブローカクラスタ) でロードバランスを実行し、JMS クライアント (コンポーネントまたはアプリケーション) の増加を可能にします。

調整可能なパフォーマンス : 信頼性の低い配信を受信する場合、MQ メッセージサービスのパフォーマンスを向上させます。

マルチトランスポート : JMS クライアントが、TCP や HTTP など、複数の異なるトランスポートを介し、安全な (SSL) コネクションを使用して互いに通信する機能をサポートします。

JNDI のサポート : ファイルベースおよび LDAP ディレクトリの両方のサービスを、オブジェクトストアおよびユーザリポジトリとしてサポートします。

SOAP メッセージングのサポート : JMS メッセージング経由で、SOAP (シンプルオブジェクトアクセスプロトコル) 仕様に準拠したメッセージである SOAP メッセージの作成と配信をサポートします。SOAP により、ピア間の構造化 XML データを分散環境で交換できます。詳細は、『MQ 開発者ガイド』を参照してください。

JMS 準拠に関連した問題については、『MQ3.0 リリースノート』を参照してください。

製品エディション

Sun ONE Message Queue 製品には、Platform および Enterprise の 2 つのエディションが用意されています。各エディションは、後続の節で説明するように、別々のライセンス機能に対応しています。MQ をいずれか一方のエディションから別のエディションにアップグレードするには、『MQ インストールガイド』を参照してください。

Platform Edition: このエディションは、Sun の Web サイトから無料でダウンロードできます。また、最新の Sun ONE Application Server プラットフォームにもバンドルされています。Platform Edition には、次のように 2 つのライセンスが付属しています。

- 基本ライセンス。このライセンスには、基本的な JMS サポート (完全な JMS プロバイダ) が用意されていますが、ロードバランス (マルチブローカのメッセージサービス)、HTTP/HTTPS コネクション、安全なコネクションサービス、スケーラブルなコネクション機能、複数キューの配信ポリシーなど、企業向けの機能は含まれていません。このライセンスには期限が設定されていないため、要求の少ない運用環境で使用できます。

- 90 日間の企業向けトライアルライセンス。このライセンスには、マルチブローカのメッセージサービス、HTTP/HTTPS コネクション、安全なコネクションサービス、スケーラブルなコネクション機能、複数キューの配信ポリシーに対するサポートなど、基本ライセンスには含まれていない企業向けの機能がすべて含まれています。ただし、ソフトウェアのライセンスの有効期間は 90 日間です。そのため、Enterprise Edition の製品 (「Enterprise Edition」を参照) で利用できる企業向けの機能を評価するのに最適です。

Platform Edition では、各 MQ メッセージサービスでサポートされる JMS クライアントのコネクション数に制限はありません。基本ライセンスから企業向けライセンスへの切り替えについては、『MQ 管理者ガイド』の `license` コマンド行オプションの説明を参照してください。

Enterprise Edition: このエディションは、運用環境でメッセージングアプリケーションを配置および実行するために使用されます。また Enterprise Edition は、メッセージングアプリケーションやコンポーネントの開発、デバッグ、および負荷テストにも使用できます。Enterprise Edition のライセンスには有効期限がなく、マルチブローカのメッセージサービスにおけるブローカ数や、各ブローカにサポートされるクライアントコネクション数にも制限がありません。ただし、Enterprise Edition がサポートする CPU の数はライセンスで指定されています。

エンタープライズメッセージングシステム

エンタープライズメッセージングシステムを使用すると、独立した分散コンポーネントやアプリケーション間でメッセージを介して対話できるようになります。これらのコンポーネントは、同一システム、つまり同一ネットワーク上にあっても、インターネットを介して緩やかに接続されていても、メッセージングを使用してデータを渡し、それぞれの機能を調整します。

エンタープライズメッセージングシステムの要件

エンタープライズアプリケーションシステムは通常、24 時間連続した基幹システムで必要な操作で、大量のメッセージを交換する多数の分散コンポーネントで構成されています。これらのシステムをサポートするために、エンタープライズメッセージングシステムは、次の要件を満たす必要があります。

信頼性のある配信 : あるコンポーネントから別のコンポーネントに送信されるメッセージが、ネットワークやシステムの障害で失われないようにする必要があります。つまり、システムで、メッセージが確実に配信されることを保証する必要があります。

非同期の配信 : 複数のコンポーネントでメッセージを同時に交換したり、高密度スループットのサポートを可能にするために、メッセージを即座に受信するコンシューマの準備状態に関係なくメッセージを送信する必要があります。コンシューマが使用中またはオフラインの場合、コンシューマの準備が完了すると、システムはメッセージの送受信を可能にする必要があります。これは非同期メッセージ配信、また一般には蓄積転送メッセージングとして知られています。

セキュリティ : メッセージングシステムは、ユーザの認証、メッセージやリソースへの承認されたアクセス、ネットワーク経由の暗号化といった基本的なセキュリティ機能をサポートする必要があります。

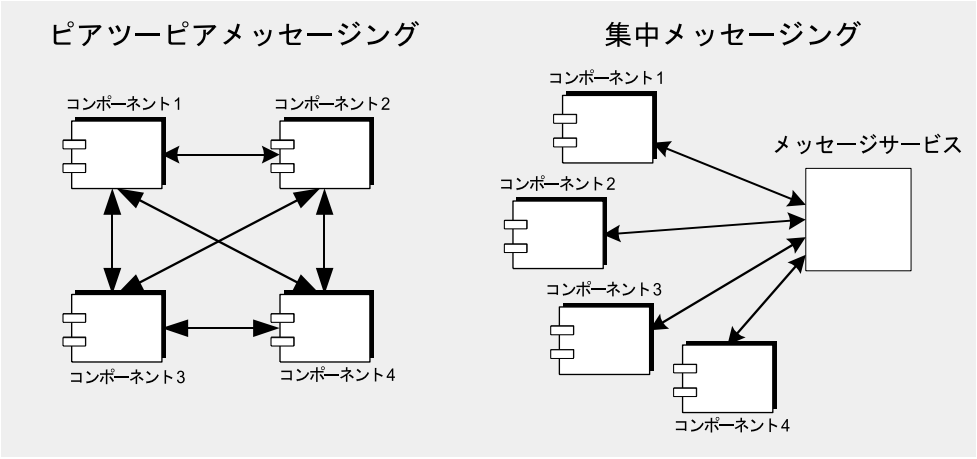
スケーラビリティ : メッセージングシステムは、パフォーマンスやメッセージスループットを実質的に失うことなく、ユーザやメッセージの増加という増大する負荷に対応できる必要があります。ビジネスやアプリケーションが拡大するにつれ、これは非常に重要な要件となります。

管理機能 : メッセージングシステムには、メッセージの配信を監視および管理し、システムリソースを最適化するためのツールが用意されていることが必要です。これらのツールは、信頼性、セキュリティ、およびパフォーマンスの測定と維持に有用です。

集中メッセージングとピアツーピアメッセージング

図 1-1 で示すように、エンタープライズメッセージングシステムの要件を従来のピアツーピアメッセージングシステムで満たすのは困難です。

図 1-1　集中メッセージングとピアツーピアメッセージング



ピアツーピアのようなシステムでは、メッセージングコンポーネントは、ほかのそれぞれのコンポーネントとのコネクションを維持しています。これらのコネクションでは、高速で安全な、信頼性の高い配信が可能ですが、コンポーネントごとに信頼性やセキュリティをサポートするコードが必要となります。コンポーネントがシステムに追加されるに従い、コネクション数は急激に増加します。そのため、非同期メッセージ配信やスケーラビリティを実現するのは難しくなります。また、集中管理メッセージングにも問題があります。

図 1-1 に示すように、エンタープライズメッセージングに適しているのは、集中メッセージングシステムです。このアプローチ方法では、各メッセージングコンポーネントは、1 つの中央メッセージサービスへのコネクションを維持します。このメッセージサービスでは、コンポーネント間のメッセージのルーティングや配信ができ、信頼性の高い配信とセキュリティが実現されます。コンポーネントは、詳細に定義されたプログラミングインタフェースを使用してメッセージサービスと対話します。コンポーネントがシステムに追加されると、コネクション数は一定の割合で増加するため、メッセージサービスを拡張することにより、システムを容易に拡張できます。さらに、集中メッセージサービスを利用することで、システムを一元管理できます。

メッセージングシステムの概念

エンタープライズメッセージングサービスには、いくつかの基本概念があります。その概念には、メッセージ、メッセージサービスアーキテクチャ、およびメッセージ配信モデルが含まれます。

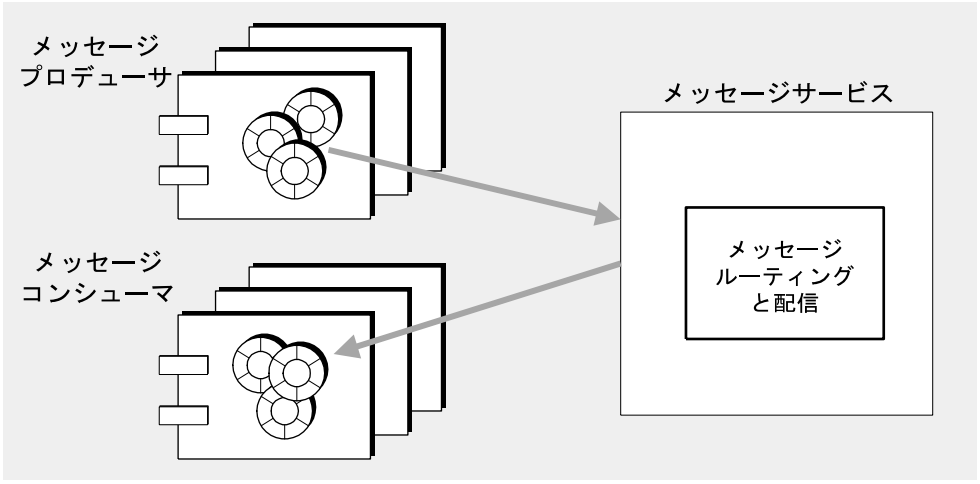
メッセージ

メッセージは、ある形式のデータ (メッセージ本体) と、送信先、生存期間、またはメッセージシステムによって決定されたほかの特性など、メッセージの特性やプロパティを示すメタデータ (メッセージヘッダー) で構成されています。

メッセージサービスのアーキテクチャ

31 ページの図 1-2 に、メッセージングシステムの基本アーキテクチャを示します。この基本アーキテクチャは、共通メッセージサービス経由でメッセージを交換する、メッセージプロデューサとメッセージコンシューマで構成されています。メッセージプロデューサとメッセージコンシューマは、同一メッセージングコンポーネント (またはアプリケーション) 内に複数存在できます。メッセージプロデューサは、メッセージサービスにメッセージを送信します。次に、メッセージサービスはメッセージルーティングと配信コンポーネントを利用して、メッセージに必要事項を登録した 1 つまたはそれ以上のメッセージコンシューマにメッセージを配信します。メッセージルーティングと配信コンポーネントは、すべての適切なコンシューマへのメッセージの配信を保証する役割を担います。

図 1-2　メッセージサービスのアーキテクチャ



メッセージ配信モデル

プロデューサとコンシューマの関係には、いろいろな種類があります。1 対 1、1 対多、および多対多の関係です。たとえば、メッセージの配信は次のように行われます。

- 1 つのプロデューサから 1 つのコンシューマへ
- 1 つのプロデューサから多数のコンシューマへ
- 多数のプロデューサから 1 つのコンシューマへ
- 多数のプロデューサから多数のコンシューマへ

これらの関係は、多くの場合、ポイントツーポイントのメッセージングとパブリッシュ / サブスクライブのメッセージングという 2 つのメッセージ配信モデルにまとめることができます。ポイントツーポイント配信モデルは、特定のプロデューサが発信元となり、特定のコンシューマが受信するメッセージに焦点を当てています。パブリッシュ / サブスクライブ配信モデルは、不特定多数のプロデューサが発信元となり、不特定多数のコンシューマが受信するメッセージに焦点を当てています。この 2 つの配信モデルは、重複する場合もあります。

今まで、メッセージングシステムでは、これら 2 つのメッセージ配信モデルのさまざまな組み合わせをサポートしていました。JMS (Java Message Service) の API は、Java メッセージングの共通プログラミングアプローチの作成を目的としていました。JMS では、ポイントツーポイント、およびパブリッシュ / サブスクライブの両方のメッセージ配信モデルをサポートしています (38 ページの「プログラミングドメイン」を参照)。

JMS 仕様

JMS では、メッセージ構造、プログラミングモデル、およびメッセージングの動作を決定するルールやセマンティックのセットを指定します。MQ は JMS を実装するため、JMS の概念は、MQ メッセージングシステムの動作を理解する基本となります。ここでは、マニュアルを理解するのに必要な概念や用語を説明します。

JMS メッセージ構造

JMS 仕様に従い、メッセージはヘッダー、プロパティ、および本体で構成されています。

ヘッダー : ヘッダーでは、メッセージの送信先、持続的かどうか、生存期間、および優先度といったメッセージの JMS 特性が指定されます。これらの特性によって、メッセージングシステムがどのようにメッセージを配信するかが決定されます。

プロパティ : ヘッダーの拡張部分とも考えられるプロパティは省略可能です。プロパティでは、アプリケーションがさまざまな選択条件に基づいて、メッセージをフィルタするのに使用する値が提供されます。プロパティは省略可能です。

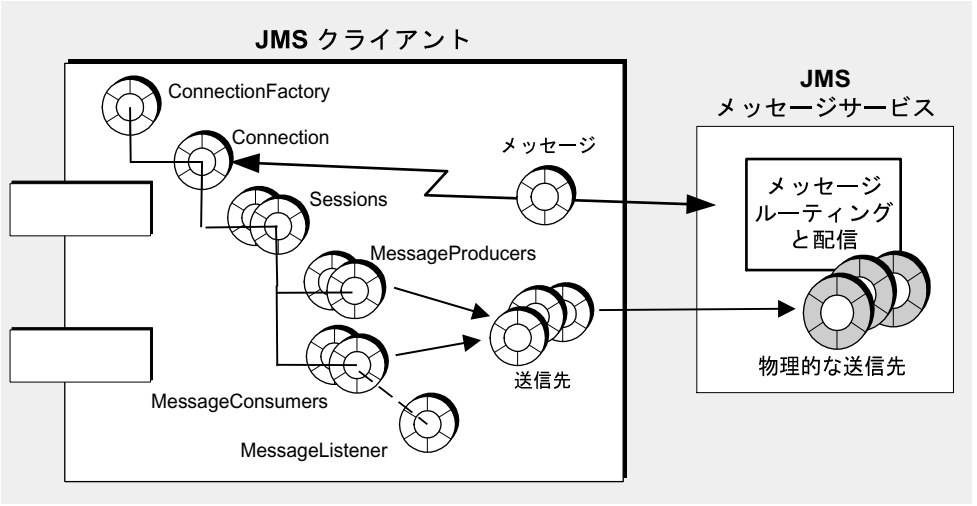
メッセージ本体 : メッセージ本体には、交換される実際のデータが含まれます。JMS では、6 種類のメッセージ本体がサポートされています。

JMS プログラミングモデル

JMS プログラミングモデルでは、JMS クライアント (コンポーネントまたはアプリケーション) が JMS メッセージサービス経由でメッセージを交換します。メッセージプロデューサはメッセージをメッセージサービスに送信し、メッセージコンシューマはこのメッセージサービスからメッセージを受信します。これらのメッセージングは、JMS アプリケーションプログラミングインタフェース (API) を実装するオブジェクトセット (JMS により提供) を使用して動作します。

ここでは、JMS API を実装し、メッセージ配信のための JMS クライアントの設定に使用されるオブジェクトを紹介します (詳細は、『MQ 開発者ガイド』を参照)。33 ページの図 1-3 は、メッセージ配信のプログラム作成に使用される JMS オブジェクトを示します。

図 1-3　JMS プログラミングオブジェクト



JMS プログラミングモデルでは、JMS クライアントが ConnectionFactory オブジェクトを使用して、メッセージを JMS メッセージサービスから受信したり、JMS メッセージサービスへ送信したりするときに使用するコネクションを確立します。Connection とは、メッセージサービスに対する JMS クライアントのアクティブなコネクションです。通信リソースの割り当てとクライアントの認証は、コネクションが確立されたときに行われます。このオブジェクトは比較的重いため、多くのクライアントはメッセージングを 1 つのコネクションだけで行います。

コネクションはセッションの確立に使用されます。Session は、メッセージのプロデュースとコンシュームのためのシングルスレッドコンテキストです。このコンテキストは、メッセージの送受信を行うメッセージプロデューサとメッセージコンシューマの作成に使用され、配信するメッセージの順番を定義します。セッションでは、複数の通知オプションまたはトランザクション (分散トランザクションマネージャで管理されている) を通して、信頼性の高い配信をサポートしています。

JMS クライアントは MessageProducer を使用して、送信先オブジェクトとして API で表される特定の物理的送信先にメッセージを送信します。メッセージプロデューサは、デフォルトの配信モード (持続メッセージと持続的でないメッセージ)、優先度、およびプロデューサが物理的送信先に送信する全メッセージを決定する生存期間を指定します。

同様に、JMS クライアントは MessageProducer を使用して、送信先オブジェクトとして API で表される特定の物理的送信先からメッセージを受信します。メッセージコンシューマは、メッセージセレクタを使用できます。このメッセージセレクタを使用すると、メッセージサービスで選択条件に一致するメッセージコンシューマにだけメッセージを送信できます。

メッセージコンシューマは、同期または非同期のどちらかのメッセージのコンシュームをサポートしています (『MQ 開発者ガイド』を参照)。コンシューマを使用して `MessageListener` を登録すると、非同期のコンシュームが実行されます。セッションスレッドが `MessageListener` オブジェクトの `onMessage()` メソッドを呼び出すと、クライアントはメッセージをコンシュームします。

管理対象オブジェクト

32 ページの「JMS プログラミングモデル」で説明している 2 つのオブジェクトは、JMS プロバイダがどのように JMS メッセージサービスを実装するかに依存します。コネクションファクトリオブジェクトは、基礎となっているプロトコルや、プロバイダがメッセージの送信に使用するメカニズムに依存し、送信先オブジェクトは、特定の命名規則や、プロバイダが使用する物理的送信先に依存します。

通常、これらプロバイダ固有の特性により、JMS クライアントコードは特定の JMS 実装に依存します。ただし、JMS クライアントコードがプロバイダに依存しないようにするには、JMS 仕様ではプロバイダ固有の実装と設定情報が管理対象オブジェクトと呼ばれるものにカプセル化される必要があります。これらのオブジェクトには、プロバイダ固有ではない、標準的な方法でアクセスできます。

管理対象オブジェクトは管理者によって作成、設定されてネーミングサービスに格納されます。JMS クライアントは標準的な Java Naming and Directory Service (JNDI) 検索コードを介して格納された管理対象オブジェクトにアクセスします。この方法で管理対象オブジェクトを使用すると、JMS クライアントコードがプロバイダに依存しなくなります。

JMS には、コネクションファクトリと送信先の 2 種類の一般的な管理対象オブジェクトがあります。どちらもプロバイダ固有情報をカプセル化しますが、JMS クライアント内では、それぞれまったく違った方法で使用されます。コネクションファクトリは、メッセージサーバへのコネクションを確立するのに使用されますが、送信先オブジェクトは、JMS メッセージサービスに使用される物理的送信先を識別するのに使用されます。

管理対象オブジェクトの詳細については、78 ページの「MQ 管理対象オブジェクト」を参照してください。

JMS/J2EE プログラミング : メッセージ駆動型 Beans

32 ページの「JMS プログラミングモデル」で説明した一般的な JMS クライアントプログラミングモデルのほかに、JMS に特化したプログラミングモデルがあり、Java 2 Enterprise Edition (J2EE) アプリケーションのコンテキストで使用されます。この特殊な JMS クライアントは、メッセージ駆動型 Beans と呼ばれ、EJB 2.0 仕様書 (`http://java.sun.com/products/ejb/docs.html`) で指定されている Enterprise JavaBeans (EJB) コンポーネントのファミリーの 1 つです。

ほかの EJB コンポーネント (セッション Beans とエンティティ Beans) はメッセージ駆動型 Beans と同時に呼び出す必要があります。これらの EJB コンポーネントは、標準的な EJB インタフェースを介してしかアクセスできないため、非同期メッセージ受信のメカニズムは備わっていません。

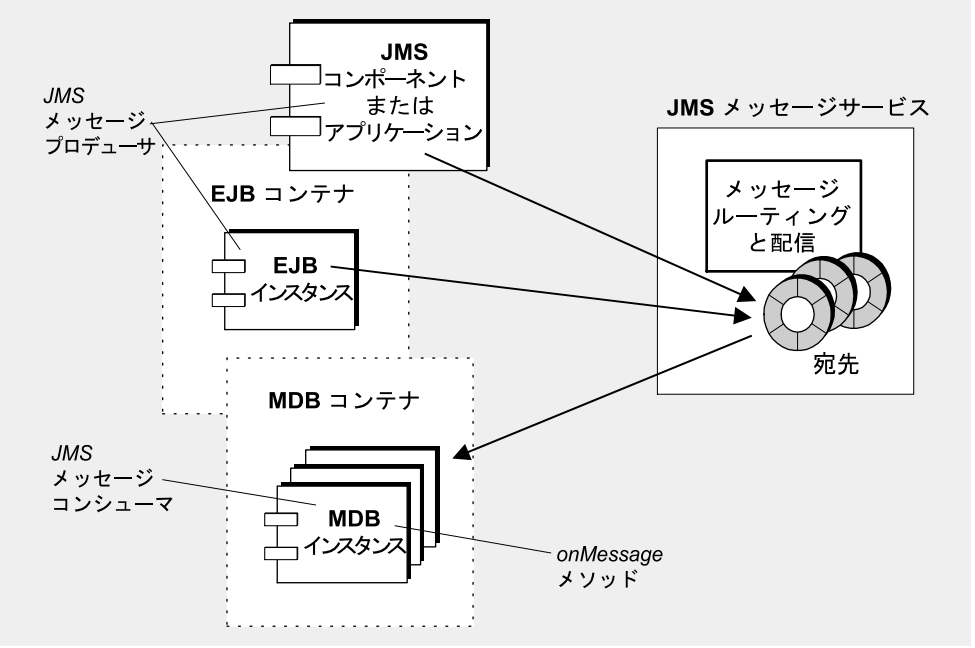
ただし、非同期メッセージングは多くのエンタープライズアプリケーションの要件となっています。これらのアプリケーションの多くは、サーバサイドコンポーネントがサーバリソースと結びつかずに、お互いに通信や応答できることが要件となっています。このため、メッセージのプロデューサにしっかり対応していなくてもメッセージを受信し、コンシュームできる EJB コンポーネントが必要となります。この機能は、サーバサイドコンポーネントがアプリケーションイベントに応答する必要のある、すべてのアプリケーションで必要となります。エンタープライズアプリケーションでは、負荷が増加する場合、この機能を拡張する必要があります。

メッセージ駆動型 Beans

メッセージ駆動型 Beans (MDB) は、特殊な EJB コンテナ (サポートするコンポーネントに分散サービスを提供するソフトウェア環境) がサポートする特殊な EJB コンポーネントです。

メッセージ駆動型 Beans: MDB とは、JMS `MessageListener` インタフェースを実装する JMS メッセージコンシューマです。MDB 開発者により書き込まれた `onMessage` メソッドは、MDB コンテナがメッセージを受信したときに呼び出されます。`onMessage()` メソッドは、標準的な `MessageListener` オブジェクトの `onMessage()` メソッドがコンシュームするのと同じように、メッセージをコンシュームします。ほかの EJB コンポーネントの場合とは異なり、MDB はメソッドをリモートで呼び出さないため、これと関連付けられているホームまたはリモートのインタフェースはありません。MDB は単一の送信先からのメッセージをコンシュームできます。36 ページの図 1-4 にあるように、スタンドアロン JMS アプリケーション、JMS コンポーネント、または EJB コンポーネントでメッセージをプロデュースできます。

図 1-4　MDB を使用したメッセージング



MDB コンテナ : MDB コンテナは、MDB のインスタンスを作成し、メッセージの非同期のコンシュームのためにそのインスタンスを設定する役割を担います。これには、メッセージサービスを使用したコネクションの設定 (認証を含む)、特定の送信先に関するセッションのプールの作成、セッションのプールや関連する MDB インスタンスでメッセージが受信されるときのメッセージ分散の管理が含まれます。コンテナは MDB インスタンスのライフサイクルを制御するので、受信メッセージの読み込みに対応できるように MDB インスタンスのプールを管理します。

コネクションファクトリと送信先のメッセージコンシュームを設定するときに、コンテナに使用される管理対象オブジェクトの JNDI 検索名を指定する配置記述子は、MDB に関連付けられています。また、配置記述子には、MDB コンテナを設定するために配置ツールで使用されるほかの情報も含まれます。各 MDB コンテナは、1 つの MDB のインスタンスだけをサポートします。

アプリケーションサーバのサポート

J2EE アーキテクチャ (`http://java.sun.com/j2ee/download.html#platformspec` にある「J2EE Platform Specification」を参照) では、EJB コンテナ (MDB コンテナを含む) がアプリケーションサーバにホストされます。アプリケーションサーバには、トランザクションマネージャ、持続マネージャ、ネーミングサービス、JMS プロバイダ (メッセージングの場合) などのさまざまなコンテナで必要とされるリソースが用意されています。

Sun ONE Application Server の場合、メッセージングリソースは Sun ONE Message Queue に用意されています。つまり、MQ メッセージングシステム (第 2 章「MQ メッセージングシステム」を参照) は Sun ONE Application Server に統合され、アプリケーションサーバ環境で実行している MDB やほかの JMS メッセージングコンポーネントへ JMS メッセージを送信するのに必要なサポートを提供しています。

JMS メッセージングの問題点

この節では、MQ メッセージサービスの管理に影響を与える、JMS プログラミングの多くの問題点を説明します。その中で MQ 管理者に必要な概念や用語について重点的に解説しています。

JMS プロバイダへの非依存性

JMS は管理対象オブジェクト (34 ページの「管理対象オブジェクト」を参照) の使用方法を指定し、ほかの JMS プロバイダに移植可能なクライアントアプリケーションの開発をサポートします。管理対象オブジェクトにより、JMS クライアントはプロバイダ固有オブジェクトを検索および参照するための論理名を使用できます。この方法では、クライアントコードは特定の命名構文やアドレス指定構文、またはプロバイダによって使用される設定可能なプロパティを知る必要はありません。これにより、コードはプロバイダに依存しなくなります。

管理対象オブジェクトは、MQ 管理者により作成および設定される MQ システムオブジェクトです。これらのオブジェクトは、JNDI ディレクトリサービスに配置され、JMS クライアントは JNDI 検索を使用してアクセスします。

また、MQ 管理対象オブジェクトは、JNDI ディレクトリサービスで検索されず、クライアントによってインスタンス化されます。この管理対象オブジェクトには、アプリケーション開発者がプロバイダ固有の API を使用する必要があるという欠点があります。また、MQ 管理者が MQ メッセージサーバを制御、管理できないこともあります。

管理対象オブジェクトの詳細については、78 ページの「MQ 管理対象オブジェクト」を参照してください。

プログラミングドメイン

JMS では、ポイントツーポイント、およびパブリッシュ / サブスクライブの 2 つのメッセージ配信モデルをサポートしています。

ポイントツーポイント (キュー送信先): メッセージは、1 つのプロデューサから 1 つのコンシューマに配信されます。この配信モデルの場合、送信先はキューです。まず、メッセージはキュー送信先に配信されます。次にキューの配信ポリシー (68 ページの「キューの送信先」を参照) に従って 1 回に 1 つずつ、キューに登録されたコンシューマの 1 つへ配信されます。複数のプロデューサがメッセージをキュー送信先に配信しても、メッセージは 1 つだけのコンシューマに配信されて、正常にコンシュームされます。キュー送信先にコンシューマが 1 つも登録されていない場合、キューは受信したメッセージを保持し、コンシューマがキューに登録されるとすぐに、そのメッセージを配信します。

パブリッシュ / サブスクライブ (トピック送信先): メッセージは、1 つのプロデューサから複数のコンシューマに配信されます。この配信モデルの場合、送信先はトピックです。まず、メッセージはトピック送信先に配信されます。次にトピックに加入されているすべてのアクティブなコンシューマに配信されます。トピック送信先へのメッセージの送信は、複数のプロデューサで実行でき、メッセージごとに加入済みの複数のコンシューマに配信されます。また、トピック送信先は、永続サブスクリプションの概念もサポートしています。永続サブスクリプションは、トピック送信先とともに登録されたコンシューマを表していますが、メッセージ配信時にアクティブでないことがあります。その後、コンシューマがアクティブになると、メッセージを受信します。トピック送信先に登録されたコンシューマがない場合、アクティブでないコンシューマの永続サブスクリプションがない限り、トピックは受信したメッセージを保持しません。

これら 2 つのメッセージ配信モデルは、わずかに異なるセマンティックを持った別々の API オブジェクトを使用して処理され、表 1-1 にあるように異なるプログラミングドメインを表しています。

表 1-1 JMS プログラミングオブジェクト

基本タイプ (統一ドメイン)	ポイントツーポイントドメイン	パブリッシュ / サブスクライブドメイン
Destination (Queue または Topic)*	Queue	Topic

* プログラミングのアプローチによっては、特定の送信先タイプを指定する場合がある

表 1-1　JMS プログラミングオブジェクト (続き)

基本タイプ (統一ドメイン)	ポイントツーポイントドメイン	パブリッシュ / サブスクライブドメイン
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Connection	QueueConnection	TopicConnection
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver	TopicSubscriber

* プログラミングのアプローチによっては、特定の送信先タイプを指定する場合がある

表 1-1 の 1 列目にある統一ドメインオブジェクトを使用して、ポイントツーポイントとパブリッシュ / サブスクライブのどちらのメッセージングもプログラミングできます。このアプローチをお勧めします。ただし、JMS 1.02b 仕様に準拠するには、ポイントツーポイントのメッセージングにポイントツーポイントドメインのオブジェクトを、パブリッシュ / サブスクライブのメッセージングにパブリッシュ / サブスクライブドメインのオブジェクトを使用します。

クライアント識別子

JMS プロバイダは、クライアント識別子の概念をサポートする必要があります。それによりクライアントの代わりにメッセージサービスが保持しているステート情報を使用して、JMS クライアントのコネクションはメッセージサービスに関連付けられます。クライアント識別子は固有のもので、一度に 1 人のユーザだけに適用されます。クライアント識別子は永続サブスクリプション名 (38 ページの「パブリッシュ / サブスクライブ (トピック送信先)」を参照) と組み合わせて使用され、各永続サブスクリプションが 1 人のユーザだけに対応するようにします。

JMS 仕様では、クライアント識別子は、API メソッドの呼び出しを介してクライアントで設定できますが、コネクションファクトリ管理対象オブジェクト (34 ページの「管理対象オブジェクト」を参照) を使用して管理者が設定することをお勧めします。ただし、コネクションファクトリに物理的に組み込まれている場合、各ユーザは個々のコネクションファクトリに固有の ID を持たせる必要があります。

MQ には、ConnectionFactory オブジェクトで設定できる特殊な変数置換構文を使用してクライアント識別子を ConnectionFactory とユーザ固有のものの両方にできる方法が用意されています。この方法を使用すると、命名の重複やセキュリティの問題を気にせずに、永続サブスクリプションを作成する複数のユーザが、単一の ConnectionFactory オブジェクトを使用できます。したがって、ユーザの永続サブスクリプションは、誤って消去されることも、別のユーザが間違ったクライアント識別子を設定したために使用できなくなることもありません。

この MQ の機能の使用に関する詳細は、『MQ 開発者ガイド』にあるコネクションファクトリの属性についての記述を参照してください。

どんな場合でも、永続サブスクリプションを作成するには、クライアントが JMS API を使用してプログラム的にクライアント識別子を設定するか、クライアントが使用する ConnectionFactory オブジェクト内で管理者がクライアント識別子を設定する必要があります。

信頼性のあるメッセージング

JMS は、次の 2 つの配信モードを定義します。

持続メッセージ : 持続メッセージは、必ず 1 回 (1 回のみ) 配信され確実にコンシュームされることが保証されています。持続メッセージにとって、信頼性は大変重要です。

持続的でないメッセージ : 持続的でないメッセージは、1 回の配信だけが保証されています。持続的でないメッセージにとって、信頼性はそれほど重要な問題ではありません。

持続メッセージの場合、信頼性の保証には 2 つの側面があります。1 つは、メッセージサービスへの配信とメッセージサービスからの配信が問題なく行われるようにすることです。もう 1 つは、コンシューマに配信される前にメッセージサービスが持続メッセージを失わないようにすることです。

通知およびトランザクション

信頼性のあるメッセージングでは、持続メッセージの送信先からの配信や送信先への配信が確実に行われることが重要です。MQ セッションは 2 つの一般的なメカニズム (通知およびトランザクション) をサポートしています。このどちらかを使用すると、信頼性のあるメッセージングを実現できます。トランザクションを使用する場合、分散トランザクションマネージャに制御されている状態では、ローカルまたは分散のどちらかになります。

通知

信頼性の高い配信を実現するために、セッションを設定して通知を使用するようにします。

プロデューサの場合、プロデューサの `send()` メソッドが返される前に、メッセージサービスが送信先に持続メッセージの配信を通知します。コンシューマの場合は、メッセージサービスが送信先からメッセージを削除する前に、クライアントが送信先からの持続メッセージの配信とコンシュームを通知します。

ローカルトランザクション

セッションを処理済として設定することもできます。その場合、1 つ以上のメッセージのプロデュースおよびコンシュームが、トランザクションという極小の単位にグループ化されます。JMS API には、トランザクションの起動、コミット、およびロールバックするメソッドが用意されています。

メッセージがトランザクション内でプロデュースまたはコンシュームされると、ブローカはさまざまな送受信を追跡し、クライアントが呼び出しを実行してトランザクションをコミットしたときにだけ、追跡を完了させます。トランザクション内で特定の送信や受信が失敗すると、例外が発生します。クライアントコードは、例外の無視、操作の再試行、あるいはトランザクション全体のロールバックを実行して、例外を処理します。トランザクションがコミットされると、すべての操作が正常に完了したことになります。トランザクションがロールバックされると、正常に行われたすべての操作が取り消されます。

ローカルトランザクションの範囲は、常に単一セッションです。つまり、単一セッションのコンテキストで実行される、1 つ以上のプロデューサまたはコンシューマの操作は、単一のローカルトランザクションにグループ化されます。

トランザクションは単一セッションだけに及ぶため、メッセージのプロデュースとコンシュームの両方を含むエンド・ツー・エンドのトランザクションを持つことはできません。言い換えると、送信先へのメッセージの配信と、それに続くクライアントへのメッセージの配信は、単一トランザクションに置くことはできません。

分散トランザクション

MQ は、分散トランザクションもサポートしています。つまり、メッセージのプロデュースとコンシュームは、データベースシステムなど、ほかのリソースマネージャに関連した操作を含む大容量の分散トランザクションの一部です。分散トランザクションでは、Java Transaction API (JTA)、*XA Resource* API 仕様で定義された 2 階層コミットプロトコルを使用して、メッセージサービスやデータベースマネージャといった複数のリソースマネージャが実行する操作を、分散トランザクションマネージャが追跡および管理します。Java の世界では、リソースマネージャと分散トランザクションマネージャ間の対話は、JTA 仕様で記述されます。

分散トランザクションをサポートするということは、JTA で定義される XAResource インタフェースを介して、メッセージングクライアントが、分散トランザクションに参加できるということです。このインタフェースは、2 階層コミットを実装するために数多くのメソッドを定義します。API の呼び出しがクライアント側で行われている間、MQ ブローカは分散トランザクション内のさまざまな送受信操作やトランザクションの状態を追跡し、Java Transaction Service (JTS) で提供される分散トランザクションマネージャと一致したときにだけ、メッセージング操作を完了します。

ローカルトランザクションで例外を処理する場合、クライアントは例外の無視、操作の再試行、および分散トランザクション全体のロールバックを実行します。

MQ は、XA コネクションファクトリを介した分散トランザクションのサポートを実装しています。この XA コネクションファクトリでは、XA セッション (32 ページの「JMS プログラミングモデル」を参照) を作成する XA コネクションを確立できます。さらに、分散トランザクションをサポートするには、サードパーティの JTS、または JTS を提供する J2EE 準拠の Application Server のいずれかが必要です。

持続ストレージ

信頼性のほかの重要な側面は、1 度持続メッセージが送信先に配信されたら、そのメッセージがコンシューマに配信されるまで、メッセージサービスがメッセージを失わないようにすることです。つまり、持続メッセージが送信先に配信されたら、メッセージサービスは持続メッセージを持続ストレージ (56 ページの「持続マネージャ」を参照) に配置する必要があります。何かの理由でメッセージサービスが停止した場合、持続データストアでメッセージは復元され、適切なコンシューマに配信されます。これにより、メッセージ配信にオーバーヘッドが発生しますが、信頼性も向上します。

メッセージサービスは、永続サブスクリプションも格納する必要があります。これは、トピック送信先への配信を保証するには、持続メッセージの復元だけでは十分ではないからです。メッセージサービスでは、トピックの永続サブスクリプションに関する情報も復元する必要があります。復元しないと、メッセージ到着時にアクティブではなく、あとからアクティブになるサブスクライバにメッセージを配信できません。

保証されたメッセージ配信を行うには、メッセージングアプリケーションは、メッセージを持続的として指定し、キュー送信先、またはトピック送信先のどちらかに対する永続サブスクリプションを使用する必要があります。

パフォーマンスのかね合い

メッセージ配信の信頼性が高くなるにつれて、その信頼性を実現するために、より多くのオーバーヘッドや帯域幅が必要となります。信頼性とパフォーマンスのかね合いは、設計上考慮するべき重要な点です。持続的でないメッセージをプロデュースおよびコンシュームするように設定すると、パフォーマンスを最大に高められます。一方、

持続メッセージをプロデュースおよびコンシュームして、処理済みのセッションを使用すると、信頼性を最大に高められます。この 2 つの要素には複数のオプションがあり、どちらを優先させるかは MQ 固有のコネクションや通知プロパティの使用など、アプリケーションの必要性によって異なります (『MQ 開発者ガイド』を参照)。

メッセージの選択

JMS には、メッセージセレクタに設定された条件に基づいて、メッセージサービスがフィルタやルーティングを実行できるようにするメカニズムが用意されています。プロデューシングクライアントは、アプリケーション固有のプロパティをメッセージに設定します。コンシューミングクライアントは、設定されたプロパティに基づく選択条件を使用して、メッセージの項目を示します。これにより、クライアントの作業が単純になり、不要なクライアントにメッセージを配信するオーバーヘッドがなくなります。ただし、選択条件を処理しているメッセージサービスに、一部のオーバーヘッドが追加されます。メッセージセレクタの構文とセマンティックは、JMS 仕様書で解説されています。

メッセージの順番と優先度

通常、単一のセッションで送信先に送信されたすべてのメッセージは、送信された順にコンシューマへ配信されるようになっています。ただし、異なる優先度が割り当てられている場合、メッセージングサービスは、優先度の高いメッセージを先に配信しようとします。

それ以外の場合、クライアントアプリケーションがメッセージをコンシュームする順番は、プロデュースされた順番と関係しますが、密接な関係はありません。これは、メッセージの送信先への配信と、送信先からの配信が、メッセージの送信順、メッセージの送信元となるセッション (コネクション)、メッセージが持続的かどうか、メッセージの生存期間、メッセージの優先度、キュー送信先 (68 ページの「キューの送信先」を参照) のメッセージ配信ポリシー、およびメッセージサービスの可用性といった、配信のタイミングに影響を与える多くのことに依存するためです。

相互接続された複数のブローカ (71 ページの「複数ブローカの設定 (クラスタ)」を参照) を使用している MQ メッセージサーバの場合、クライアントがメッセージをコンシュームする順序付けはさらに複雑になります。これは、異なるブローカの送信先からの配信順が決定されていないためです。したがって、あるブローカが配信したメッセージが、他のブローカが配信して先にメッセージングサービスに受信されたメッセージよりも優先的に配信されることもあります。

どんな場合でも、特定のコンシューマでは、優先度の低いメッセージよりも優先度の高いメッセージが優先されます。

第 2 章

MQ メッセージングシステム

この章では、MQ メッセージングシステムについて説明します。図 2-1 に示すシステムの主要部分に重点を置き、確実なメッセージ配信を実現するための連動方法について説明します。

図 2-1　MQ システムアーキテクチャ

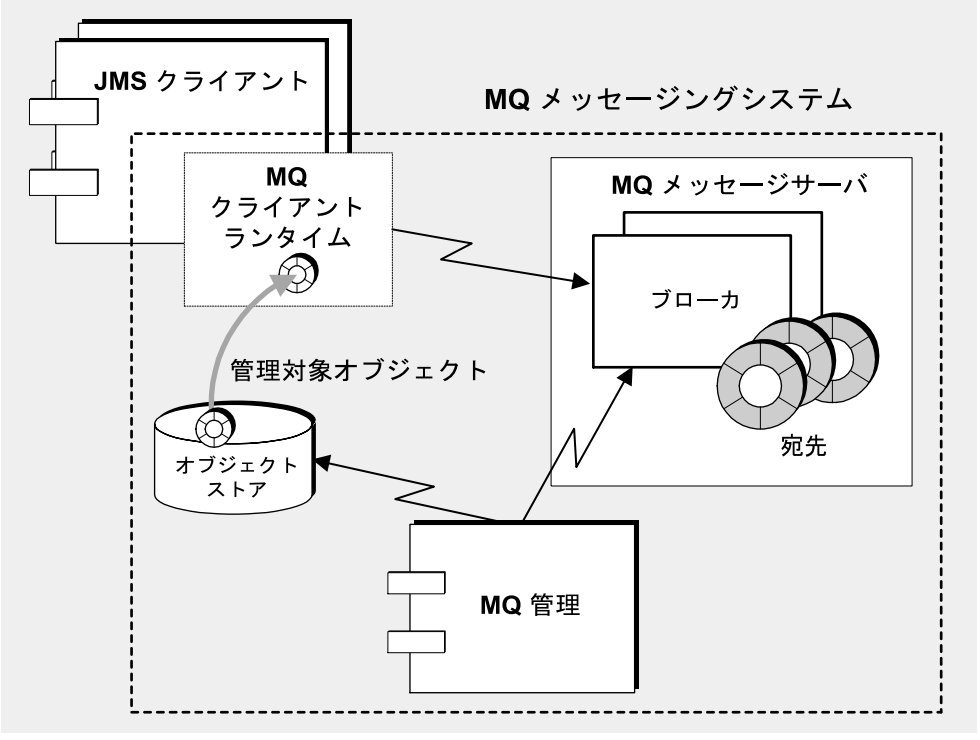


図 2-1 に示す MQ メッセージングシステムの主要部分は、次のとおりです。

- MQ メッセージサーバ
- MQ クライアントランタイム
- MQ 管理対象オブジェクト
- MQ Administration

最初の 3 項目については、次の節で説明します。最後の項目については、第 3 章「MQ の管理」で説明します。

MQ メッセージサーバ

この節では、45 ページの図 2-1 に示される MQ メッセージサーバのさまざまな部分について説明します。次の項目が含まれます。

ブローカ : ブローカには、MQ メッセージングシステムの配信サービスが用意されています。メッセージ配信は、コネクションサービス、メッセージのルーティングと配信、持続性、セキュリティ、およびログ記録を処理するさまざまなサポートコンポーネントに依存します (詳細は、46 ページの「ブローカ」を参照)。メッセージサーバでは、1 つまたは複数のブローカを使用できます (71 ページの「複数ブローカの設定 (クラスタ)」を参照)。

物理的な送信先 : メッセージは 2 段階のプロセスで配信されます。まず、プロデューシングクライアントから、ブローカが管理する物理的な送信先に配信されます。次に、送信先から 1 つまたは複数のコンシューミングクライアントに配信されます。物理的な送信先は、ブローカの物理的なメモリまたは持続ストレージ、あるいはその両方の場所を表します (詳細は、67 ページの「物理的な送信先」を参照)。

ブローカ

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 つまたは複数のコンシューミングクライアントへ配信される MQ 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。

この複雑な機能の組み合わせを実行するために、ブローカはさまざまな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。負荷状態やアプリケーションの複雑さなどに応じて、これらの内部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。図 2-2 にブローカの主要なコンポーネントを示し、表 2-1 でそれらについて簡単に説明します。

図 2-2 ブローカのコンポーネント

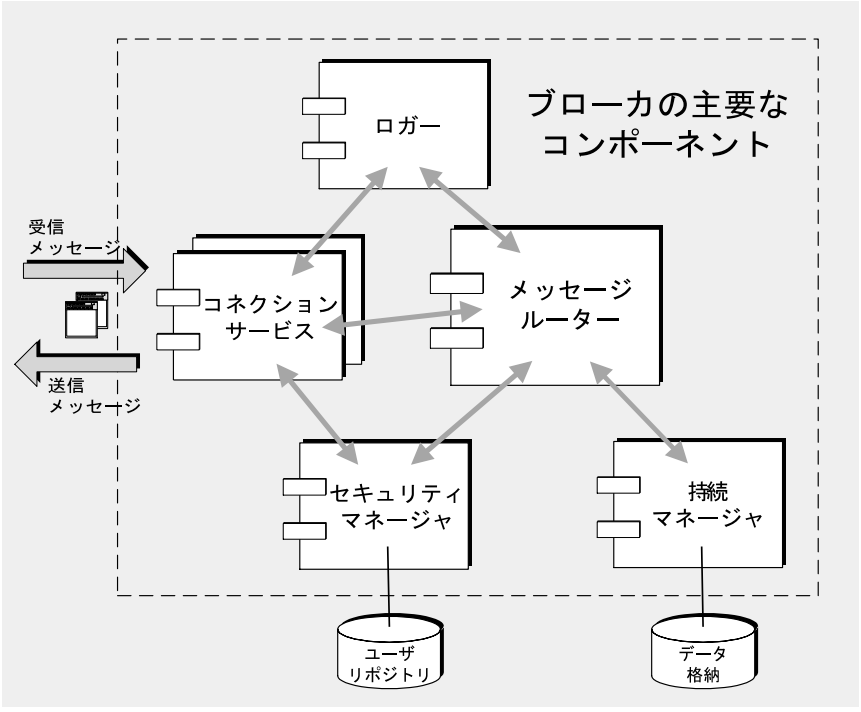


表 2-1 ブローカの主要なコンポーネントと機能

コンポーネント	説明 / 機能
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う
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する。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 MQ 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む

表 2-1　ブローカの主要なコンポーネントと機能 (続き)

コンポーネント	説明 / 機能
持続マネージャ	システム障害が発生しても、JMS メッセージが配信されるように、持続ストレージへのデータの書き込みを管理する
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に認証サービスを提供し、認証されたユーザに承認サービス (アクセス制御) を提供する
ロガー	管理者がブローカを監視および管理できるように、監視情報や診断情報をログファイル、またはコンソールに出力する

次の節では、別のブローカの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に影響を及ぼす設定可能なプロパティについて、詳しく説明します。

コネクションサービス

MQ ブローカは、JMS クライアントと MQ 管理クライアントの両方の通信をサポートしています (85 ページの「MQ 管理ツール」を参照)。各サービスは、サービスタイプとプロトコルタイプで指定されます。

サービスタイプ : JMS メッセージ配信 (NORMAL) サービス、または MQ 管理 (ADMIN) サービスのどちらを提供するのかを指定する

プロトコルタイプ : 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を指定する

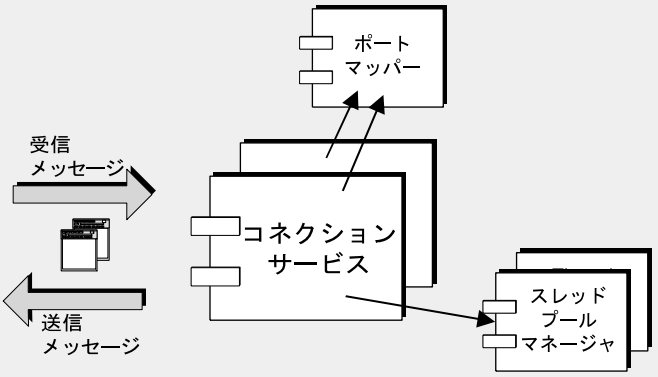
MQ ブローカで現在使用できるコネクションサービスは、表 2-2 に示されています。

表 2-2　ブローカでサポートされてい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 (JMS メッセージ配信)	tcp
ssljms	NORMAL (JMS メッセージ配信)	tls (SSL ベースセキュリティ)
httpjms	NORMAL (JMS メッセージ配信)	http
httpsjms	NORMAL (JMS メッセージ配信)	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。図 2-3 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 2-3 コネクションサービスのサポート



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。ポートは、静的または動的に割り当てることが可能です。MQ には、異なるコネクションサービスに動的に割り当てられたポートをマップするポートマッパーが用意されています。ポートマッパー自体は、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コネクションをする場合、最初に、必要なコネクションサービスのポート番号を要求するポートマッパーに接続します。

jms、ssljms、admin、および ssladmin のコネクションサービスを設定する場合、これらのコネクションサービスに、静的なポート番号を割り当てることも可能ですが、これはお勧めしません。httpjms サービスと httpsjms サービスは、付録 B 「HTTP/HTTPS サポート」の表 B-1 (219 ページ) と表 B-3 (228 ページ) にそれぞれ示すプロパティを使用して設定します。

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 : 専用モデルの場合、ブローカへのコネクションごとに、受信メッセージを処理するスレッドと、送信メッセージを処理するスレッドの 2 つのスレッドが必要になります。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 : 共有スレッドモデルの場合、メッセージを送受信する場合にだけ、スレッドにコネクションを割り当てます。コネクションがスレッドを共有するので、コネクションサービス (および、ブローカ) がサポートできるコネクションの数は増えますが、パフォーマンスのオーバーヘッドが若干生じます。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。このディストリビュータスレッドが監視するコネクションの数を制限することによって、パフォーマンスを改善できます。

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 (59 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に関する設定可能なプロパティを表 2-3 に示します。各プロパティの設定方法については、第 5 章「ブローカの起動と設定」を参照してください。

表 2-3 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.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。サポートされているサービスは、`jms`、`ssljms`、`httpjms`、`httpsjms`、`admin`、`ssladmin`。デフォルトは `jms`、`admin`
`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する。デフォルトは、コネクションサービスによって異なる (118 ページの表 5-1 を参照)
`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する。新しいスレッドは、それ以上追加されなくなる。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る。デフォルトは、コネクションサービスによって異なる (118 ページの表 5-1 を参照)

表 2-3　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.*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) するのかどちらかを指定する。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るが、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ない。デフォルトは、コネクションサービスによって異なる (118 ページの表 5-1 を参照)
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する。(すべてのコネクションを監視するのに十分なディストリビュータスレッドをシステムが割り当てる。) この値が小さいほど、システムはより早くスレッドにアクティブコネクションを割り当てることができる。値を 0 に設定した場合、無制限になる。デフォルトは、オペレーティングシステムによって異なる (118 ページの表 5-1 を参照)
imq.portmapper.port	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。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る。デフォルトは 7676
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する。デフォルトは 0 (ポートは、ポートマッパーによって動的に割り当てられる)
	httpjms コネクションサービスと httpsjms コネクションサービスを設定する場合、付録 B「HTTP/HTTPS サポート」を参照
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) を指定する。デフォルトは null (任意のホスト)

* protocol_type は 表 2-2 に記載されています。

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。1 つ目は、プロデューサクライアントによって送信されるコンシューマクライアント向けの JMS メッセージ、すなわちペイロードメッセージです。2 つ目は、JMS メッセージの配信をサポートするために、クライアント間で送受信される多数のコントロールメッセージです。

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。ルート先がアクティブになっていない永続サブスクライバの場合、サブスクライバがアクティブになるまで、メッセージルーターはメッセージを保持し、後でそのサブスクライバに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、意図したすべてのコンシューマにメッセージを配信すると、メモリからそのメッセージを消去します。また、メッセージが持続的 (40 ページの「信頼性のあるメッセージング」を参照) である場合、ブローカの持続データストアから、そのメッセージを削除します。

信頼性の高い配信 : 通知およびトランザクション

信頼性の高い配信 (40 ページの「信頼性のあるメッセージング」を参照) 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。信頼性の高い配信では、ブローカとのメッセージの配信が正常に終了することと、メッセージが実際に配信される前に、ブローカがメッセージや配信情報を失うことがないようにする 2 つの局面があります。

ブローカとのメッセージの配信を正常に行うために、MQ は通知と呼ばれる多数の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) を送信した場合、ブローカは JMS メッセージを受信したことを示すコントロールメッセージ (ブローカの通知) を送り返します。実際には、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、MQ はこれを実行します。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、ブローカの通知を使用します (77 ページの「メッセージのプロデュース」を参照)。

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。原則として、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) から通知を受信していない場合、メモリから JMS メッセージを削除しません。

トピックのサブスクライバが永続的な場合、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、ブローカの通知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、ブローカの通知を使用して、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 (77 ページの「メッセージのコンシューム」を参照)。何らかの理由で、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。一般的に、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のコネクションが閉じられて、新しいコネクションが開かれると、ブローカは JMS メッセージを再配信します。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とブローカの通知プロセスは、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、クライアントとブローカの通知は、各 JMS メッセージレベルだけでなく、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、ブローカの通知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 (41 ページの「分散トランザクション」を参照)。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。

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、メモリ内に保持されます。ただし、ブローカで障害が発生すると、これらのメッセージは失われる可能性があります。

プロデューサクライアントは、メッセージを持続に指定することができます。そのように指定すると、メッセージルーターは持続マネージャ (56 ページの「持続マネージャ」を参照) にメッセージを渡します。持続マネージャはそのメッセージをデータベースまたはファイルシステムに格納するので、ブローカで障害が発生しても、メッセージを復元することができます。

システムリソースの管理

ブローカのパフォーマンスは、使用できるシステムリソースとメモリなどのリソースの使用効率によって異なります。たとえば、メッセージルーターには、システムのメモリを監視するメモリ管理方式が用意されています。メモリリソースが不足すると、メモリを回復するメカニズム、つまり受信メッセージのフローを遅らせるメカニズムがアクティブになります。

メモリ管理メカニズムは、green (使用可能なメモリが十分にある)、yellow (ブローカのメモリが減っている)、orange (ブローカのメモリが不十分である)、red (ブローカのメモリが不足している) といったメモリリソースの状態によって異なります。メモリリソースの状態が、green から yellow や orange を経て red になると、ブローカはメモリを回復するための重大なアクション、つまり、メッセージプロデューサを徐々に減らし、最終的にブローカへのメッセージのフローを停止するアクションをとります。

プロパティを使用して、ブローカのメモリ管理機能を設定できます。プロパティでは、メモリ内のメッセージの総数やサイズを制限したり、メモリリソースの状態を更新する頻度を調整できます。

プロパティを表 2-4 に示します。これらのプロパティの設定については、第 5 章「ブローカの起動と設定」を参照してください。

表 2-4　メッセージルーターのプロパティ

プロパティ名	説明
imq.message.expiration.interval	有効期限が切れたメッセージを回復させる頻度を秒単位で指定する。デフォルトは 60
imq.system.max_count	メモリ内とディスク内の両方 (スワッピングのため) のメッセージの最大数を指定する。この値を超えるメッセージは拒否される。値を 0 に設定した場合、無制限になる。デフォルトは 0
imq.system.max_size	メモリ内とディスク内の両方 (スワッピングのため) のメッセージの最大合計サイズ (バイト単位で、K バイトまたは M バイト) を指定する。この値を超えるメッセージは拒否される。値を 0 に設定した場合、無制限になる。デフォルトは 0
imq.message.max_size	メッセージの本体の最大許容サイズ (バイト単位で、K バイトまたは M バイト) を指定する。このサイズを超えるメッセージは拒否される。値を 0 に設定した場合、無制限になる。デフォルトは 70m (M バイト)
imq.*resource_state*.threshold	指定したメモリリソースの状態で、そのメモリリソースがトリガーされるメモリの利用率を指定する。リソースの状態を表す値は、green、yellow、orange、および red。デフォルトは、それぞれ 0、60、75、90 になる
imq.redelivered.optimization	再配信フラグを設定して、メッセージルーターがパフォーマンスを最適化するかどうか (true/false) を指定する。true の場合はメッセージが再配信されるたびに実行 、false の場合は論理的に必要がある場合にのみ実行。デフォルトは true
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 (true/false)。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る (160 ページの「トランザクションの管理」を参照)。デフォルトは false

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。

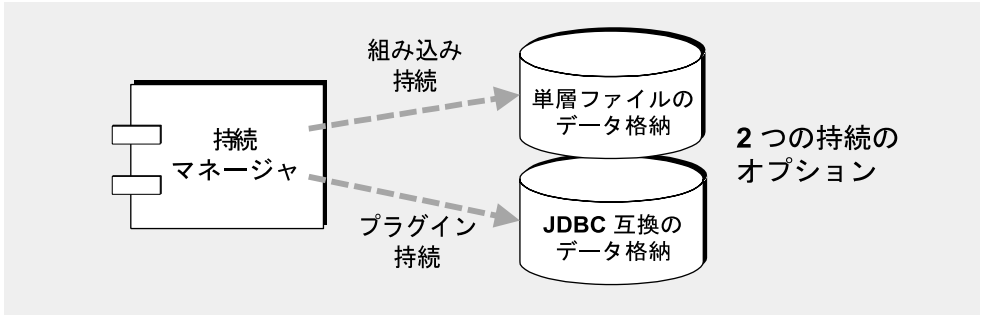
- 送信先の再作成
- 各トピックの永続サブスクリプションのリストの復元
- 各メッセージの通知リストの復元
- コミットされたすべてのトランザクションの状態の復元

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。

MQ 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がサポートしています (図 2-4 を参照)。組み込み持続は、単層ファイルのデータストアに基づいています。プラグイン持続は、JDBC 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。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もいます。

図 2-4　持続マネージャのサポート



組み込み持続

デフォルトの MQ 持続ストレージソリューションは、単層ファイルストアです。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。

単層ファイルのデータストアは、次の場所にあります。

IMQ_VARHOME/instances/*brokerName*/filestore/
(Solaris では、/var/imq/instances/*brokerName*/filestore/)

brokerName には、ブローカインスタンスを識別する名前が入ります。

ファイルベースのデータストアでは、持続メッセージは対応するファイルに 1 つずつ保存されます。ただし、送信先、永続サブスクリプション、およびトランザクションは、1 つのファイルにすべての送信先、別のファイルにすべての永続サブスクリプションといった具合に、それぞれのファイルにすべてが保存されます。

メッセージをデータストアに追加したり、データストアから削除したりするたびに、ファイルの作成または削除されると、ファイルシステムに負荷がかかります。このため、MQ 実装では、不必要になったファイルを削除する代わりに、再利用が可能な空きファイルのプールに追加して、これらのメッセージファイルを再利用します。このファイルプールのサイズは、設定することができます。また、単に再利用のためのタグを付ける (切り捨てられない) のに対して、ファイルプール内で削除する (サイズを 0 にする) 空きファイルの割合を指定することもできます。削除するファイルの割合が大きいほど、ディスク容量は小さくなりますが、ファイルプールを保持するためのオーバーヘッドが増えます。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容量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。

単層ファイルストアにメッセージを格納する速度は、データストアで使用できるファイル記述子の数に影響されます。記述子の数が多いと、システムはより多くの持続メッセージをより迅速に処理することができます。ファイル記述子の増加については、『MQ リリースノート』の「技術的な注意点」の節を参照してください。

送信先ファイルストアの場合、送信先が追加されるたびに、ファイルのサイズを大きくするのではなく、固定サイズのファイルに送信先を追加する方がより効率的です。このため、最終的に送信先ファイルに保存されると思われる送信先の数に応じて、送信先ファイルのサイズを設定することでパフォーマンスを向上できます (各送信先は約 500 バイトをコンシュームします)。

データストアでは、機密情報を保持するメッセージが含まれるため、承認されていないアクセスから、*brokerName*/filestore/ ディレクトリを保護することをお勧めします。手順については、『MQ リリースノート』の「技術的な注意点」の節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 Database Manager 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、付録 A 「プラグイン持続の設定」で説明します。

持続関連の設定プロパティを、58 ページの表 2-5 に示します。これらのプロパティの設定については、第 5 章「ブローカの起動と設定」を参照してください。

表 2-5　持続のプロパティ

プロパティ名	説明
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する。デフォルトは file
imq.persist.file.destination.file.size	組み込みのファイルベースの持続の場合、送信先を格納するファイルの初期サイズを指定する。デフォルトは 1m (M バイト)
imq.persist.file.message.filepool.limit	組み込みのファイルベースの持続の場合、ファイルプールで再利用できる空きファイルの最大数を指定する。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る。この値を超える空きファイルは削除される。この制限を越えると、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する。デフォルトは 10000

表 2-5　持続のプロパティ (続き)

プロパティ名	説明
imq.persist.file.message.filepool.cleanratio	組み込みのファイルベースの持続の場合、クリーン状態 (サイズを 0 にする) で保持されるファイルプールの空きファイルの割合を指定する。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るが、ファイルプールに必要なディスク容量は小さくなる。デフォルトは 0
imq.persist.file.message.cleanup	組み込みのファイルベースの持続の場合、ブローカのシャットダウン時に、ファイルストア内の空きファイルを削除するかどうかを指定する。値を false に設定すると、ブローカのシャットダウンが迅速化するが、ファイルストアのためのディスク容量がさらに必要になる。デフォルトは false
imq.persist.file.message.fdpool.limit	組み込みのファイルベースの持続の場合、開いたままにするデータファイルの最大数 (ファイル記述子プールのサイズ) を指定する。値が大きくなると、持続操作のパフォーマンスは向上するが、クライアントコネクションの作成のように、ファイル記述子を必要とするブローカ操作に負荷がかかる。デフォルトは Solaris および Linux の場合 25、Windows の場合 1024
imq.persist.file.sync.enabled	持続操作でメモリ内の状態を物理的なストレージと同期化させるかどうかを指定する。true の場合、システムクラッシュによるデータ損失は回避されるが、持続操作のパフォーマンスに負荷がかかる。デフォルトは false

セキュリティマネージャ

MQ 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、ファイル、ディレクトリ、またはメッセージングシステムのユーザに関する情報 (ユーザ名、パスワード、およびグループのメンバーシップ) が保存されるデータベースなどのユーザリポジトリによって異なります (61 ページの図 2-5 を参照)。ユーザ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、ユーザを認証するために使用されます。ユーザ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。

MQ の管理者は、MQ で提供されるユーザリポジトリ (184 ページの「単層型ファイルユーザ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接続します。

認証

MQ 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名とパスワードをユーザ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 暗号化か、メッセージダイジェストの MD5 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、61 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。

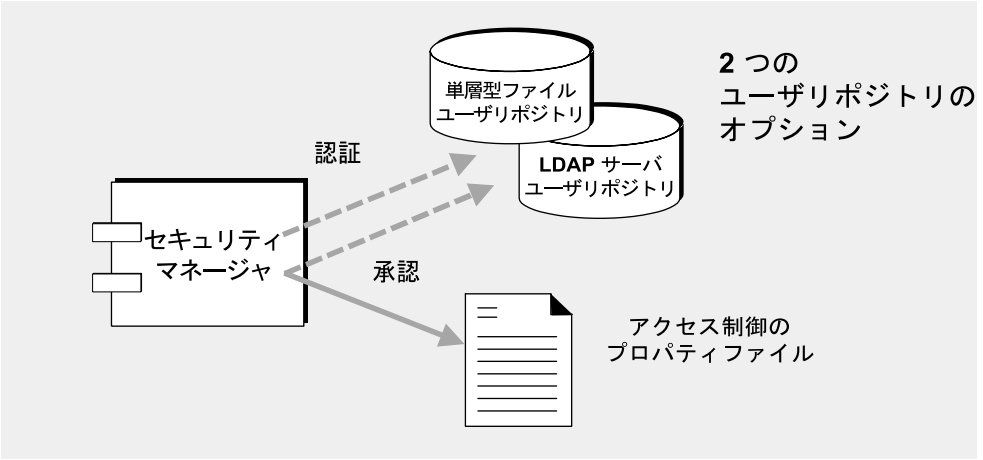
承認

クライアントアプリケーションのユーザが認証されると、ユーザはさまざまな MQ 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。ユーザ名、またはユーザリポジトリでユーザに割り当てられているグループに応じて、ユーザは特定の MQ 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 2-5 を参照)。

ユーザ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、ユーザリポジトリ内のユーザ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ユーザ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には、操作に対するユーザのアクセス権が指定されています。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 2-5　セキュリティマネージャのサポート



MQ 3.0 の場合、デフォルトのアクセス制御プロパティファイルは、1 つのグループ *admin* だけを明示的に参照します (186 ページの「グループ」を参照)。*admin* グループのユーザには、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。

MQ の管理者は、グループを定義し、ユーザリポジトリ内のそれらのグループとユーザを関連付けることができます (ただし、グループは単層ファイルのユーザリポジトリで完全にはサポートされません)。そうすると、アクセス制御プロパティファイルを編集してユーザ別およびグループ別に目的 (メッセージのプロデュースとコンシューム、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、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。

さらに、ブローカを設定して、送信先の自動作成 (69 ページの「自動作成 (および管理者によって作成) された送信先」を参照) を許可する場合、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できるユーザを制御することができます。

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。

MQ の SSL ベースのコネクションサービスを使用するには、Key Tool 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、自己署名付き証明書に公開キーを組み込んで、それを MQ のキーストアに配置します。MQ 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。詳細は、199 ページの「暗号化 : SSL 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。

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表 2-6 に示します。各プロパティの設定方法については、第 5 章「ブローカの起動と設定」を参照してください。

表 2-6 セキュリティのプロパティ

プロパティ名	説明
`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。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、符号化を設定する。デフォルトは `digest`
`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。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する。デフォルトは、`imq.authentication.type` に設定される値から継承される
`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。その他の LDAP のプロパティについては、190 ページの表 8-5 を参照。デフォルトは `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時間を秒単位で指定する。デフォルトは `180` (秒)

表 2-6　セキュリティのプロパティ (続き)

プロパティ名	説明
imq.accesscontrol.enabled	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、アクセス制御を設定する (true/false)。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 MQ 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する。デフォルトは true
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する。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 MQ 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する。デフォルトは、imq.accesscontrol.enabled プロパティの値を継承する
imq.accesscontrol.file.filename	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、アクセス制御プロパティファイルの名前を指定する。ファイル名には、IMQ_HOME/etc ディレクトリの相対ファイルパスを指定する。デフォルトは accesscontrol.properties
imq.*service_name*.accesscontrol.file.filename	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する。ファイル名には、IMQ_HOME/etc ディレクトリの相対ファイルパスを指定する。デフォルトは、imq.accesscontrol.file.filename の値を継承する
imq.passfile.enabled	セキュリティ保護される通信用の (SSL、LDAP、JDBC の) ユーザパスワードをパスファイルで指定するどうか (true/false) を指定する。デフォルトは false
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を指定する。デフォルトは、IMQ_HOME/etc (Solaris では /etc/imq)
imq.passfile.name	パスファイル名を指定する。デフォルトは passfile

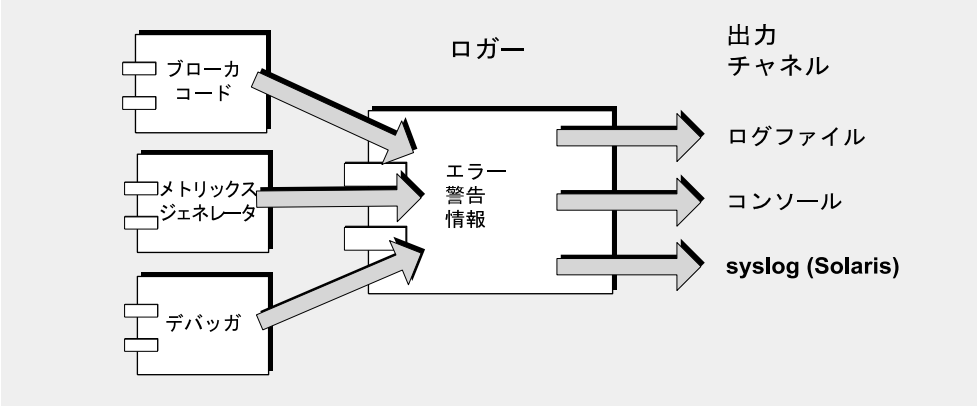
表 2-6　セキュリティのプロパティ (続き)

プロパティ名	説明
imq.keystore.*property_name*	SSL ベースのサービスの場合、SSL キーストアに関するセキュリティのプロパティを指定する。201 ページの表 8-8 を参照

ロガー

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。これらのコンポーネントの中には、データ (ブローカコード、メトリックスジェネレータ、およびデバッガ) を生成するコンポーネントや、多数の出力チャネル (ログファイル、コンソール、および Solaris syslog) を介して、情報を書き出すロガーコンポーネントがあります。このスキーマを、図 2-6 に示します。

図 2-6　ロギングスキーマ



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。

もっとも深刻で重要な情報 (エラー) から、重要度の低い情報 (メトリックスデータ) まで、ロガーのレベルを指定することもできます。情報のカテゴリを重要度の高い順に、表 2-7 に示します。

表 2-7　ロギングのカテゴリ

カテゴリ	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
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

表 2-7　ロギングのカテゴリ (続き)

カテゴリ	説明
INFO	メトリックスおよびその他の情報メッセージの報告

ロガーのレベルを設定するには、これらのカテゴリのどれかを指定します。ロガーは、指定されたカテゴリとそれより上のレベルのカテゴリのデータをすべて出力します。たとえば、ロギングに WARNING レベルを指定すると、ロガーは、WARNING 情報と ERROR 情報の両方を出力します。

ロガーは、標準出力 (コンソール)、ログファイル、syslog デーモンプロセス (Solaris プラットフォームの場合) などのさまざまな出力チャネルに、データを書き出すことができます。

各出力チャネルに対して、ロガーに設定されているカテゴリのうちどれを書き込むかを指定できます。たとえば、ロガーのレベルが ERROR 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。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 Man ページを参照してください。

ログファイルに出力する場合、ログファイルを閉じて、新しいファイルに出力がロールオーバーされる時点を指定できます。ログファイルが指定したサイズや有効期間に達すると、そのファイルは保存されて、新しいログファイルが作成されます。ログファイルは、次の場所に保存されます。

```
IMQ_VARHOME/instances/brokerName/log/
```

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。ログファイルは、次のように順番に名前が付けられるテキストファイルです。

```
log.txt
log_1.txt
log_2.txt
...
log_9.txt
```

log.txt がもっとも新しいファイルで、数字が大きくなるほど古いファイルになります。

ブローカによる情報の生成およびロギングを設定する設定可能なプロパティを表 2-8 に示します。各プロパティの設定方法については、第 5 章「ブローカの起動と設定」を参照してください。

表 2-8　ロガーのプロパティ

プロパティ名	説明
imq.metrics.enabled	メトリックス情報を収集するかどうかを指定する (true/false)。デフォルトは true
imq.metrics.interval	メトリックス情報が報告される間隔を秒単位で指定する。値を 0 に設定した場合は、情報が報告されない。デフォルトは 0
imq.log.level	ロガーのレベル (出力チャネルに書き込むことができる出力のカテゴリ) を指定する。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る。指定できる値は、高いレベルから順に ERROR、WARNING、INFO。デフォルトは INFO
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する。指定できる値は、縦線 (|) で区切ったロギングのカテゴリのセット、ALL または NONE。デフォルトは ALL
imq.log.file.dirpath	ログ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する。デフォルトは IMQ_VARHOME/instances/brokerName/log/
imq.log.file.filename	ログファイル名を指定する。デフォルトは log.txt
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する。値を 0 に設定した場合、ファイルのサイズに基づいたロールオーバーは行われない。デフォルトは 0
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する。値を 0 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行われない。デフォルトは 604800
imq.log.console.output	ログファイルに出力するロギング情報のカテゴリを指定する。指定できる値は、縦線 (|) で区切ったロギングのカテゴリのセット、ALL または NONE。デフォルトは 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。デフォルトは ERR

表 2-8　ロガーのプロパティ (続き)

プロパティ名	説明
imq.log.syslog.facility	(Solaris のみ) MQ ブローカが記録する syslog 機能を指定する。値は、syslog(3C) の Man ページに示される値をミラー化する。MQ で使用する適切な値は、LOG_USER、LOG_DAEMON、および LOG_LOCAL0 から LOG_LOCAL7 まで。デフォルトは、LOG_DAEMON
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 (true/false)。デフォルトは true
imq.log.syslog.console	(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 (true/false)。デフォルトは false
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する。デフォルトは、imqbrokerd_ の後にブローカインスタンス名
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。指定できる値は、縦線 (|) で区切ったロギングのカテゴリのセット、ALL または NONE。デフォルトは、ERROR

物理的な送信先

MQ のメッセージングは、メッセージの 2 段階配信を前提としています。メッセージは最初にプロデューサクライアントから、ブローカの送信先に配信され、次にブローカの送信先から 1 つまたは複数のコンシューマクライアントに配信されます。送信先にはキュー (ポイントツーポイント配信モード) とトピック (パブリッシュ / サブスクライブ配信モデル) の 2 つのタイプがあります (38 ページの「プログラミングドメイン」を参照)。これらの送信先は、受信メッセージがコンシューマクライアントにルーティングされる前に整列化するブローカの物理的なメモリの場所を表しています。

物理的な送信先を作成するには、MQ 管理ツールを使用します (154 ページの「送信先の管理」を参照)。送信先は、69 ページの「自動作成 (および管理者によって作成) された送信先」に示すように、自動的に作成することも可能です。

この節では、キューとトピックの 2 つのタイプの物理的な送信先のプロパティと動作について説明します。

キューの送信先

キューの送信先は、ポイントツーポイントメッセージングで使用されます。メッセージは、送信先に配信対象を登録している多数のコンシューマのうちの 1 つだけに最終的に配信されるようになっています。メッセージはプロデューサクライアントから到着すると、キューに入って、コンシューマクライアントに配信されます。

キューに入っているメッセージのルーティングは、キューの配信ポリシーによって異なります。MQ は、次の 3 つのキュー配信ポリシーを実装します。

- **シングル**　このキューでは、メッセージを 1 つのメッセージコンシューマだけにルートします。2 つ目のメッセージコンシューマが、キューに登録しようとすると拒否されます。登録されたメッセージコンシューマが接続を解除した場合、メッセージはルートされずに、新しいコンシューマが登録されるまで保存されます。
- **フェイルオーバー**　このキューでは、メッセージを複数のメッセージコンシューマにルートしますが、これはプライマリメッセージコンシューマ (ブローカに登録した最初のメッセージコンシューマ) が接続を解除したときだけ行われます。メッセージは、次に登録するメッセージコンシューマに移動し、そのコンシューマに障害などが発生しないかぎり、そのコンシューマにルートされ続けます。メッセージコンシューマが登録されていない場合は、コンシューマが登録されるまでメッセージは保存されます。
- **ラウンドロビン**　このキューでは、メッセージを複数のメッセージコンシューマにルートします。多数のコンシューマがキューに登録されている場合、キューの 1 番目のメッセージは、1 番目に登録されているメッセージコンシューマにルートされ、2 番目のメッセージは、2 番目に登録されているコンシューマに順番にルートされます。追加メッセージは、一連の同様のコンシューマに、同様の順序でルートされます。コンシューマがキューに登録する前に、多数のメッセージがキューに入っている場合、1 つのコンシューマにメッセージが集中するのを避けるため、メッセージはバッチにルートされます。メッセージコンシューマが接続を解除すると、そのコンシューマにルートされたメッセージは、残りのアクティブコンシューマに配信し直されます。このような再配信が発生するため、コンシューマへのメッセージの配信順序は、メッセージがキューに受信された順序と同じになるとはかぎりません。

メッセージは、長期間キューに存在することができるため、メモリリソースが問題になる可能性があります。キューに割り当てるメモリが、多すぎると、メモリが十分に利用されず、少なすぎると、メッセージが拒否されます。各キューのロード要求に基づいて柔軟性をもたせるために、キューの作成時に、キュー内のメッセージの最大数、キュー内のメッセージに割り当てられる最大メモリ、およびキュー内の任意のメッセージの最大サイズといった物理的なプロパティを設定できます (155 ページの表 6-10 を参照)。

トピックの送信先

トピックの送信先は、パブリッシュ / サブスクライバメッセージングで使用されます。メッセージは、送信先に配信対象を登録しているすべてのコンシューマに最終的に配信されるようになっています。メッセージはプロデューサから送信されてくると、トピックをサブスクライブするすべてのコンシューマにルートされます。コンシューマがトピックの永続サブスクリプションを登録している場合、コンシューマは、トピックにメッセージが配信されるときに、アクティブになっている必要はありません。コンシューマが再びアクティブになるまで、ブローカがメッセージを保存し、配信するからです。

通常、メッセージはトピックの送信先に長期間保存されることがないため、メモリリソースは大きな問題にはなりません。しかし、送信先で受信されるメッセージの最大許容サイズを設定することは可能です (155 ページの表 6-10 を参照)。

自動作成 (および管理者によって作成) された送信先

JMS メッセージサーバは、メッセージングシステムの中心的なハブであるため、そのパフォーマンスと信頼性が、エンタープライズアプリケーションの成功に向けての重要な鍵となります。送信先は、送信先が処理するメッセージの数とサイズ、および登録するメッセージコンシューマの数と永続性によっては、リソースを著しくコンシュームする可能性があるので、メッセージサーバのパフォーマンスと信頼性を確保するために、送信先を綿密に管理する必要があります。したがって、アプリケーションのための送信先の作成、送信先の監視、必要に応じたリソース要件の再構成が、MQ の管理者の基本的な業務になります。

しかし、送信先を動的に作成するのが望ましい場合もあります。たとえば、開発およびテストサイクル中に、管理者の介入なしで、必要に応じてブローカに送信先を自動的に作成させる必要がでてくる場合があります。

MQ は、この自動作成機能がサポートしています。自動作成を有効にすると、実在しない送信先に、MessageConsumer や MessageProducer がアクセスしようとしたときに、ブローカが送信先を自動的に作成します。ただし、クライアントアプリケーションのユーザは、自動作成の権限を保持する必要があります (198 ページの「送信先自動作成アクセス制御」を参照)

送信先が明示的ではなく自動的に作成される場合、同じ送信先名を使用する異なるクライアントアプリケーション間でクラッシュが発生したり、送信先をサポートするのに必要なリソースによって、システムのパフォーマンスが低下したりする可能性があります。このため、MQ の自動作成された送信先は、使用されなくなると (すなわち、メッセージコンシューマクライアントがなくなり、メッセージが含まれていない場合)、ブローカによって自動的に破棄されます。ブローカが再起動すると、持続メッセージが含まれている場合にだけ、自動作成された送信先が再び作成されます。

表 2-9 に示すプロパティを使用して、自動作成機能を有効、または無効にするように、MQ メッセージサーバを設定することができます (各プロパティの設定方法については、第 5 章「ブローカの起動と設定」を参照)。

表 2-9　自動作成の設定プロパティ

プロパティ名	説明
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 (true/false)。デフォルトは true
imq.autocreate.queue	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 (true/false)。デフォルトは true
imq.queue.deliverypolicy	自動作成されたキューのデフォルトの配信ポリシーを指定する。指定できる値は single、round-robin、または failover。デフォルトは single

一時送信先

一時送信先は、ほかのクライアントに送信したメッセージに対する応答を受信するために送信先が必要な場合に、クライアントアプリケーションによって、(JMS API を使用して) 明示的に作成および破棄されます。これらの送信先は、送信先が作成されたコネクションの存続期間中にだけ、ブローカによって保持されます。管理者が一時送信先を破棄することはできません。また、一時送信先が使用されている間、すなわち、アクティブなメッセージコンシューマがいる間は、クライアントアプリケーションが一時送信先を破棄することもできません。管理者が作成した送信先や自動作成された送信先 (持続メッセージを保持する) と違って、一時送信先は、継続的には格納されず、ブローカの再起動時に作成し直されることもありません。また、一時送信先は、MQ の管理ツールで認識できません。

複数ブローカの設定 (クラスタ)

MQ Enterprise Edition (26 ページの「製品エディション」を参照) は、複数の連結したブローカ、すなわち、ブローカのクラスタを使用したメッセージサーバの実装がサポートしています。クラスタのサポートによって、メッセージサーバのスケーラビリティが提供されます。

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、ファイル記述子やメモリ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、MQ メッセージサーバにブローカを追加して、クライアントのコネクションとメッセージの配信を複数のブローカに分散することです。

また、複数のブローカ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、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、個々のブローカへのクライアントの接続に、速度の速いリンクを使用することができます。

異なるユーザリポジトリを保持するワークグループに対応したり、ファイアウォールの制限に対処したりする場合など、ブローカのクラスタを使用する理由はほかにも存在しますが、フェイルオーバーはこれに含まれません。クラスタ内の 1 つのブローカを、障害が発生した別のブローカの自動バックアップとして使用することはできません。ブローカの自動フェイルオーバー保護は、MQ 3.0 ではサポートされていません (ただし、カスタマイズしたフェイルオーバー方式を実装するために、複数のブローカを使用するようにアプリケーションを設計することは可能)。

ブローカのクラスタの設定および管理については、126 ページの「ブローカクラスタの操作」で説明します。次の節では、MQ のブローカのクラスタのアーキテクチャと内部機能について説明します。

複数ブローカのアーキテクチャ

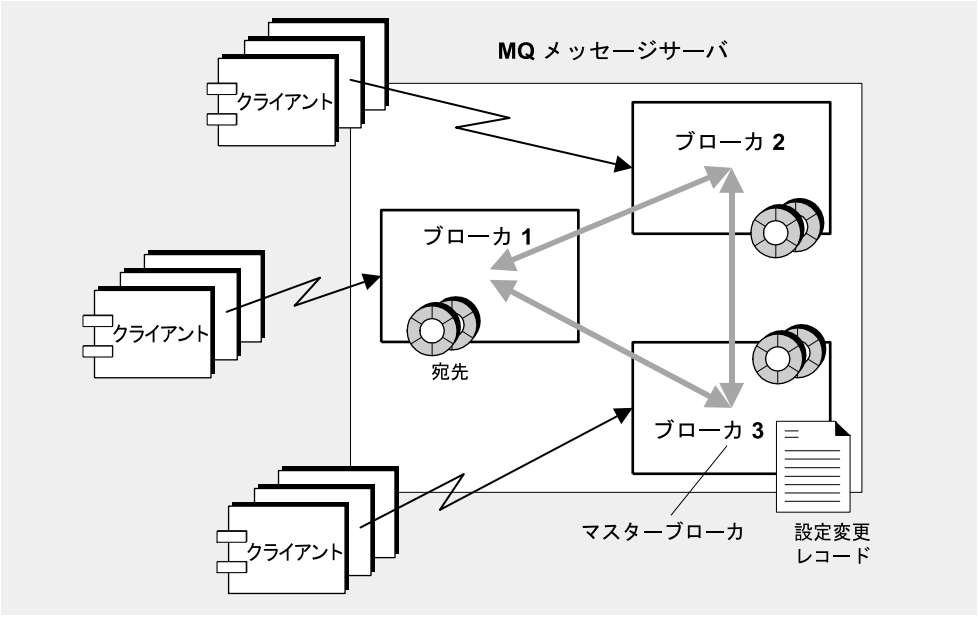
図 2-7 に示すように、複数ブローカのメッセージサーバを使用すると、クライアントのコネクションを多数のブローカに分散することができます。クライアント側から見た場合、各クライアントは個々のブローカ (クライアントの ホームブローカ) に接続し、ホームブローカがクラスタ内の唯一のブローカであるかのように、メッセージを送受信します。しかし、サーバ側から見た場合、ホームブローカは、クラスタ内のほかのブローカと並行して動作し、直接接続しているメッセージプロデューサとコンシューマに配信サービスを提供します。

通常、クラスタ内のブローカには、任意のトポロジで接続することが可能です。ただし、72 ページの図 2-7 に示すように、MQ 3.0 は、完全に接続されたクラスタ、すなわち、各ブローカがクラスタ内のほかのすべてのブローカに直接接続するトポロジだけをサポートしています。

複数ブローカの設定では、各送信先のインスタンスは、クラスタ内のすべてのブローカに常駐します。また、各ブローカは、ほかのすべてのブローカに登録されたメッセージコンシューマを認識します。このため、各ブローカは、ブローカに直接接続しているメッセージプロデューサから、リモートのメッセージコンシューマにメッセージをルートし、リモートのプロデューサから、ブローカに直接接続しているコンシューマにメッセージを配信することができます。

クラスタ設定では、各メッセージプロデューサが直接接続しているブローカが、そのプロデューサによって送信されたメッセージのルートを行います。したがって、持続メッセージの格納とルートは、両方ともメッセージのホームブローカによって行われます。

図 2-7　複数ブローカ (クラスタ) のアーキテクチャ



管理者がブローカの送信先を作成、または破棄すると、この情報がクラスタ内のほかのすべてのブローカに自動的に伝えられます。同様に、メッセージコンシューマがホームブローカに登録された場合や、明示的、またはクライアントやネットワークの障害やホームブローカのダウンが原因で、コンシューマがホームブローカから接続を解除された場合に、コンシューマについての関連情報がクラスタ全体に伝えられます。永続サブスクリプションに関する情報も同じように、クラスタ内のすべてのブローカに伝えられます。

通常、送信先やメッセージコンシューマに関する情報を特定のブローカに伝える場合、共有リソースで変更が行われたときに、ブローカがオンラインになっている必要があります。ブローカがクラッシュして再起動していたり、新しいブローカをクラスタに動的に追加していたりして、ブローカがオフラインになっているときにこのような変更が行われると、どうなるかについて、次に説明します。

オフラインになっているブローカ (または、新規追加されたブローカ) に対応するために、MQ は、クラスタ内のすべての持続エンティティの変更記録、すなわち、作成または破棄されたすべての送信先と永続サブスクリプションの記録が保管されます。ブローカはクラスタに動的に追加されると、最初に、この 設定変更記録から送信先および永続サブスクライバの情報を読み取ります。ブローカはオンラインになったときに、現在のアクティブコンシューマに関する情報をほかのブローカと交換します。この情報を使用して、新しいブローカはクラスタに完全に統合化されます。

設定変更記録は、クラスタ内の 1 つのブローカ、すなわち、マスターブローカとして設定されているブローカで管理されます。クラスタにブローカを動的に追加する場合、マスターブローカがキーポイントになるため、常にこのブローカを最初に起動しておく必要があります。マスターブローカがオンラインになっていないと、クラスタ内のほかのブローカが初期化を実行できません。

マスターブローカがオフラインになると、ほかのブローカが設定変更記録にアクセスできなくなり、MQ は送信先と永続サブスクリプションをクラスタ全体に伝えられません。このような状況では、送信先または永続サブスクリプションを作成または破棄しようとすると (あるいは、永続サブスクリプションの再有効化のようなさまざまな関連操作を実行しようとすると)、例外が生じます。

基幹アプリケーション環境の場合、設定変更記録の偶発的な破損や、マスターブローカの障害に対応するために、設定変更記録を定期的にバックアップすることをお勧めします。これは `imqbrokerd` コマンドの `-backup` オプション (123 ページの表 5-2 を参照) を使用すると実行できます。このオプションでは、設定変更記録を含んだバックアップファイルを作成する方法が提供されます。その後で、`-restore` オプションを使用して、設定変更記録を復元することができます。

必要に応じて、マスターブローカの役割を果たすブローカを変更することができます。これを実行するには、設定変更記録をバックアップし、適切なクラスタ設定プロパティ (74 ページの表 2-10 を参照) を変更して新しいマスターブローカを指定した後、`-restore` オプションを使用して新しいマスターブローカを再起動します。

開発環境でのクラスタの使用

クラスタをテストに使用し、スケーラビリティやブローカの復元が重視されない開発環境では、マスターブローカはさほど必要ありません。マスターブローカなしで設定した環境の場合、MQ はほかのブローカを起動するために、マスターブローカを実行する要件を緩和し、送信先や永続サブスクリプションの変更を行って、この変更をクラスタ内で実行中のすべてのブローカに伝えることを許可します。ただし、ブローカがオフラインになって、その後で復元された場合、オフライン中に行われた変更は同期化されません。

通常、テスト環境の場合、送信先は自動作成 (69 ページの「自動作成 (および管理者によって作成) された送信先」を参照) され、これらの送信先の永続サブスクリプションは、テストしているアプリケーションによって作成および破棄されます。送信先と永続サブスクリプションにおけるこれらの変更は、クラスタ全体に伝えられます。ただし、マスターブローカを使用する環境を再設定すると、MQ は送信先および永続サブスクリプションを変更し、クラスタ全体にこれらの変更を伝えるために、マスターブローカを起動しておく要件を再び強要するようになります。

クラスタ設定プロパティ

クラスタ内の各ブローカは、起動時に、クラスタ内のほかのブローカに関する情報 (ホスト名とポート番号) を受け取る必要があります。この情報は、クラスタ内のブローカ間で、コネクションを確立する際に使用されます。各ブローカは、マスターブローカ (使用している場合) のホスト名とポート番号についても把握する必要があります。

クラスタ内のすべてのブローカは、同じクラスタ設定プロパティを使用する必要があります。これは、各ブローカが起動時に参照する中央の 1 つの クラスタ設定ファイルに、クラスタ設定プロパティを配置することで実現できます。

これらの設定プロパティを複製して、各ブローカに配置することも可能ですが、クラスタ設定で不一致が生じる可能性があるため、お勧めしません。クラスタ設定プロパティを 1 つのファイルだけにすることで、すべてのブローカが同じ情報を読み取るようになります。

MQ のクラスタ設定プロパティを、表 2-10 に示します。これらのプロパティの設定については、126 ページの「ブローカクラスタの操作」を参照してください。

表 2-10 クラスタ設定プロパティ

プロパティ名	説明
`imq.cluster.brokerlist`	クラスタ内のすべてのブローカを指定する。*host:port* エントリのコンマで区切られたリストで構成される。*host* は、各ブローカのホスト名で、*port* はそのポートマッパーのポート番号

表 2-10　クラスタ設定プロパティ (続き)

プロパティ名	説明
imq.cluster.masterbroker	クラスタ内のどのブローカをマスターブローカにするのかを指定する。マスターブローカは状態の変化を追跡する。プロパティは、*host:port* で構成される。*host* は、マスターブローカのホスト名で、*port* はそのポートマッパーのポート番号
imq.cluster.url	クラスタ設定ファイルの場所を指定する。クラスタのプロパティ値を個別に提供するのではなく、ブローカが中央の 1 つの設定ファイルを参照する場合に使用する。URL の文字列で設定される。Web サーバ上にある場合は、通常の `http:`*URL* を使用して、アクセスできる。共有ドライブ上にある場合は、`file:`*URL* を使用して、アクセスできる
imq.cluster.port	クラスタ内にある各ブローカに対して、クラスタコネクションサービスのポート番号の指定に使用される。クラスタコネクションサービスは、クラスタにあるブローカ間での内部通信に使用される。デフォルトは、`0` (ポートは動的に割り当てられる)
imq.cluster.hostname	クラスタ内の各ブローカに対して、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、複数の使用可能なホストがある場合に、クラスタコネクションサービスのバインド先となるホスト (ホスト名または IP アドレス) の指定に使用される。クラスタコネクションサービスは、クラスタにあるブローカ間での内部通信に使用される。デフォルトは、`null` (すべての使用可能なホスト)

クラスタ設定ファイルは、一連のブローカに共通するすべてのブローカ設定プロパティを格納できます。本来、クラスタ設定ファイルは、クラスタを設定するためのものですが、クラスタ内のすべてのブローカに共通するほかのブローカのプロパティを格納するのにも使用できます。

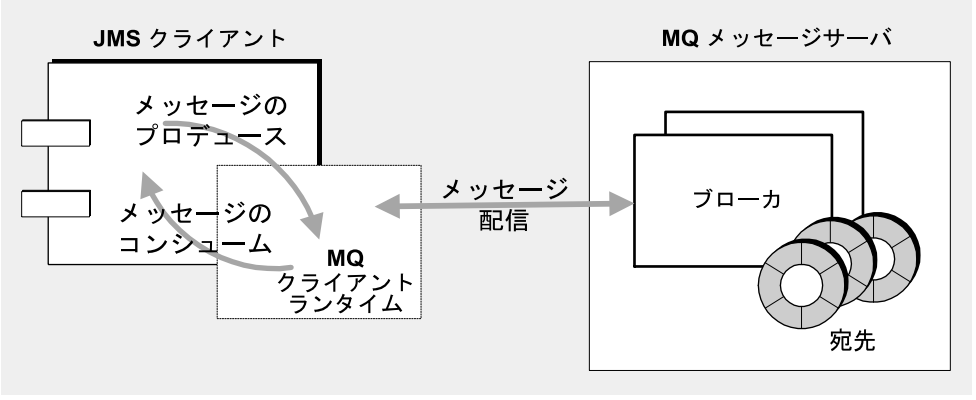
MQ クライアントランタイム

MQ 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 MQ メッセージサーバへのインタフェースを提供します。つまり、クライアントランタイムによって、32 ページの「JMS プログラミングモデル」に記載されるすべての JMS プログラミングオブジェクトがクライアントアプリケーションに提供されます。クライアントランタイムでは、クライアントが送信先にメッセージを送信し、送信先からメッセージを受信するために必要なすべての処理がサポートされます。

この節では、MQ クライアントランタイムの動作方法について詳しく説明します。クライアントランタイムのパフォーマンスに影響を与える要因については、クライアントアプリケーションの設計やパフォーマンスに影響するため、『MQ 開発者ガイド』で説明します。

76 ページの図 2-8 に、クライアントアプリケーションと MQ クライアントランタイム間の対話におけるメッセージのプロデュースとコンシューム、および MQ クライアントランタイムと MQ メッセージサーバ間の対話におけるメッセージの配信について示します。

図 2-8　メッセージングの処理



メッセージのプロデュース

メッセージのプロデュースでは、メッセージはクライアントによって作成され、ブローカの送信先へのコネクションを介して送信されます。MessageProducer オブジェクトのメッセージ配信モードが、持続 (配信を 1 回だけ保証する) に設定されている場合、メッセージが送信先に配信されて、ブローカの持続データストアに格納されたことをブローカが通知するまで、クライアントスレッドはブロックされます。メッセージが持続的でない場合、ブローカの通知メッセージ (「ACK」というプロパティ名で呼ばれる) は返されず、クライアントスレッドはブロックされません。

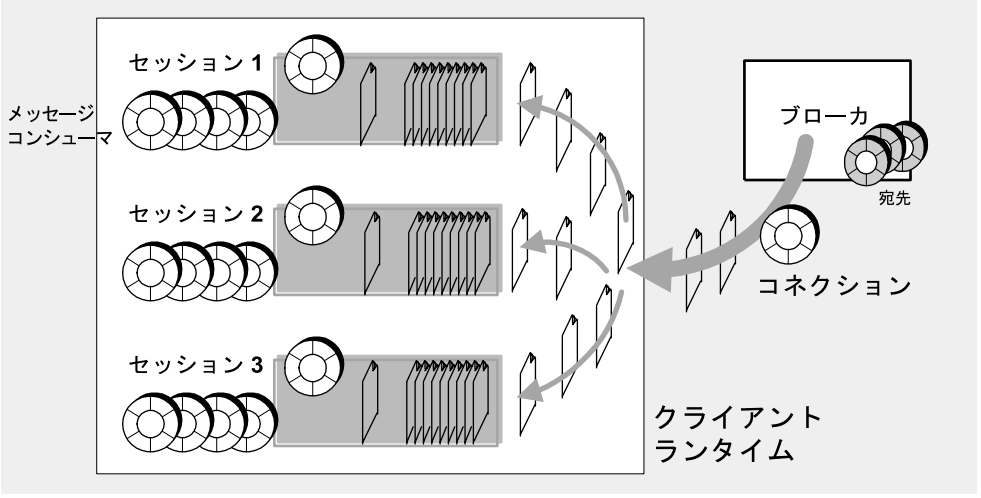
メッセージのコンシューム

メッセージのコンシュームは、プロデュースより複雑です。ブローカの送信先に到着したメッセージは、次の条件に基づいて MQ 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されます。

- クライアントが、特定の送信先のコンシューマを設定している
- コンシューマの選択基準が、特定の送信先に到着したメッセージの基準と一致している
- メッセージの配信を開始するように、コネクションが指示されている

図 2-9 に示すように、コネクションを介して配信されるメッセージは、適切な MQ セッションに分散されます。ここで、メッセージはキューに入れられて適切な MessageConsumer オブジェクトによってコンシュームされます。メッセージは、1 つずつ各セッションキューからフェッチされ (セッションはシングルスレッドになる)、`receive` メソッドを呼び出すクライアントスレッドによって同期的、または MessageListener オブジェクトの `onMessage` メソッドを呼び出すセッションスレッドによって非同期的にコンシュームされます。

図 2-9　MQ クライアントランタイムへのメッセージの配信



ブローカはクライアントランタイムにメッセージを配信する際、メッセージにマークを付けますが、メッセージが受信、またはコンシュームされたかどうかは実際には把握していません。このため、ブローカは、ブローカの送信先からメッセージを削除する前に、クライアントがメッセージの受信を通知するのを待ちます。

MQ 管理対象オブジェクト

管理対象オブジェクトを使用すると、クライアントアプリケーションコードがプロバイダに依存しなくなります。これはクライアントアプリケーションが使用するオブジェクトに、プロバイダに依存しない方法で、プロバイダ固有の実装と設定情報をカプセル化することによって実現できます。管理対象オブジェクトは、管理者が作成および設定し、ネームサービスに格納されます。クライアントアプリケーションは、標準 JNDI 検索コードを介して管理対象オブジェクトにアクセスします。

MQ には、`ConnectionFactory` と `Destination` の 2 つのタイプの管理対象オブジェクトが用意されています。どちらもプロバイダの固有情報をカプセル化し、クライアントアプリケーション内でいろいろな方法で使われます。`ConnectionFactory` オブジェクトは、メッセージサーバへのコネクションを作成する際に使用され、`Destination` オブジェクトは、物理的な送信先を識別する際に使用されます。

管理対象オブジェクトで、MQ メッセージサーバの制御および管理を非常に簡単に行うことができます。

- クライアントアプリケーションをあらかじめ設定した `ConnectionFactory` オブジェクト (168 ページの「管理対象オブジェクトの属性」を参照) にアクセスさせることによって、コネクションの動作を制御できる

- 既存の物理的な送信先に対応するあらかじめ設定した Destination オブジェクトに、クライアントアプリケーションをアクセスさせることによって、物理的な送信先の拡散を制御できる。ただし、ブローカの自動作成機能を無効にしておく必要がある (69 ページの「自動作成 (および管理者によって作成) された送信先」を参照)
- クライアントアプリケーションが設定したメッセージヘッダーの値をオーバーライド (168 ページの「管理対象オブジェクトの属性」を参照) することによって、MQ メッセージサーバのリソースを制御できる

この仕組みによって、MQ の管理者はメッセージサーバの設定詳細を管理することが可能となり、同時に、クライアントアプリケーションがプロバイダに依存しなくなります。このため、クライアントアプリケーションは、プロバイダ固有の構文およびオブジェクトの命名規則 (37 ページの「JMS プロバイダへの非依存性」を参照)、またはプロバイダ固有の設定プロパティを把握する必要がありません。

管理対象オブジェクトは、第 7 章「管理対象オブジェクトの管理」に示すように、MQ の管理ツールを使用して作成します。管理対象オブジェクトを作成する場合、オブジェクトを作成するときに設定した MQ 固有の設定値が、クライアントアプリケーションによって変更されないように、オブジェクトを読み取り専用に設定することができます。つまり、クライアントコードが読み取り専用の管理対象オブジェクトに、属性値を設定することはできませんし、81 ページの「クライアントの起動時の属性値のオーバーライド」に示すように、管理者がクライアントアプリケーションの起動オプションを使用して、これらの値をオーバーライドすることもできません。

クライアントアプリケーションは、ConnectionFactory と Destination の両方の管理対象オブジェクトをインスタンス化することができますが、それは、管理対象オブジェクトの基本目的 (MQ の管理者が、アプリケーションに必要なブローカリソースの制御とそのパフォーマンスの調整を行えるようにする) が徐々に失われます。また、管理対象オブジェクトを直接インスタンス化すると、クライアントアプリケーションは、プロバイダに依存しないものではなく、プロバイダに依存したものになります。

コネクションファクトリ管理対象オブジェクト

ConnectionFactory オブジェクトは、クライアントアプリケーションと MQ メッセージサーバ間の物理的なコネクションを確立する場合に使用します。また、コネクションの動作や、ブローカにアクセスするためにコネクションを使用するクライアントランタイムの動作を指定する場合にも使用します。

分散トランザクション (41 ページの「ローカルトランザクション」を参照) をサポートする必要がある場合、分散トランザクションをサポートする特殊な XAConnectionFactory オブジェクトを使用する必要があります。

ConnectionFactory 管理対象オブジェクトを作成する場合は、175 ページの「コネクションファクトリ管理対象オブジェクトの追加」を参照してください。

ConnectionFactory 管理対象オブジェクトを設定して、オブジェクトがプロデュースするすべてのコネクションに共通する属性値 (プロパティ) を指定します。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。これらの属性値は、影響を及ぼす動作に基づいて、いくつかのカテゴリに分類されます。

- コネクションの指定
- 自動再接続の動作
- クライアントの識別
- メッセージヘッダーのオーバーライド
- 信頼性およびフローの制御
- キューブラウザの動作
- アプリケーションサーバのサポート
- JMS 定義されたプロパティのサポート

各カテゴリと対応する属性については、『MQ 開発者ガイド』で詳しく説明します。MQ の管理者は、これらの属性値の調整を行わなければならない場合もありますが、クライアントアプリケーションのパフォーマンスをチューニングするために、調整が必要な属性を判断するのは、通常、アプリケーション開発者です。169 ページの表 7-3 に、属性の概要をアルファベット順に示します。

送信先管理対象オブジェクト

Destination 管理対象オブジェクトは、公に名前の付けられた Destination オブジェクトに対応するブローカの物理的な送信先 (キュー、またはトピック) を表しています。この 2 つの属性を表 2-11 に示します。Destination オブジェクトを作成すると、クライアントアプリケーションの MessageConsumer オブジェクトまたは MessageProducer オブジェクト、あるいはその両方が、対応する物理的な送信先にアクセスできるようになります。

Destintion 管理対象オブジェクトを作成する場合は、176 ページの「トピックまたはキューの管理対象オブジェクトの追加」を参照してください。

表 2-11 送信先の属性

属性 / プロパティ名	説明
imqDestinationName	物理的な送信先のプロバイダ固有名を指定する。物理的な送信先を作成したときに、この名前を指定する。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る。送信先名は、英字や "_" および "$" で開始できる。 デフォルトは Untitled_Destination_Object
imqDestinationDescription	オブジェクトの管理に役立つ情報を指定する。デフォルトは A Description for the Destination Object

クライアントの起動時の属性値のオーバーライド

Java アプリケーションの場合と同様に、コマンド行でシステムのプロパティを指定して、メッセージングアプリケーションを起動できます。この方法は、クライアントアプリケーションコードで使用されている管理対象オブジェクトの属性値をオーバーライドすることもできます。たとえば、JNDI 検索を介してアクセスするクライアントアプリケーションコードの管理対象オブジェクトの設定をオーバーライドすることが可能です。

クライアントアプリケーションの起動時に、管理対象オブジェクトの設定をオーバーライドするには、次のようなコマンド行の構文を使用します。

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

attribute は、168 ページの「コネクションファクトリ管理対象オブジェクト」に示す ConnectionFactory 管理対象オブジェクトの属性と一致します。

たとえば、クライアントアプリケーションをクライアントコードでアクセスされる ConnectionFactory 管理対象オブジェクトに指定されたブローカとは異なるブローカに接続する場合、別のブローカの `imqBrokerHostName` と `imqBrokerHostPort` を設定するために、オーバーライドのコマンド行を使用して、クライアントアプリケーションを起動することができます。

ただし、管理対象オブジェクトが読み取り専用に設定されている場合、オーバーライドのコマンド行を使用して、その属性値を変更することはできません。そのようなオーバーライドは、無視されます。

第 3 章

MQ の管理

MQ 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。

MQ の管理タスク

開発環境、または運用環境のどちらにいるかによって、実行が必要な特定のタスクが変わります。

開発環境

開発環境では、作業は MQ のクライアントアプリケーションのプログラミングに重点が置かれます。MQ メッセージサーバは、主にテストのために必要となります。開発環境では、テストで使用するための開発者用のブローカの起動が大部分を占めているため、柔軟性に重点が置かれ、管理は最小限になります。一般的に、デフォルトで実装されているデータストア、ユーザリポジトリ、アクセス制御プロパティファイル、およびオブジェクトストアは、開発テストに適しています。複数のブローカのテストを実行する場合、マスターブローカは使用しません。また、テストを行うアプリケーションでは、一般的に自動作成された送信先を使用するので、集中管理される管理対象オブジェクトは使用しません。

運用環境

運用環境では、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。実行が必要な管理タスクは、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。しかし、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。

- セキュリティ (第 8 章「セキュリティ管理」を参照)
  - ファイルベースのユーザ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 ユーザリポジトリを使用する

    (少なくとも、管理機能をパスワードで保護する必要がある)
  - 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する
  - SSL ベースのコネクションサービスを設定する
- 管理対象オブジェクト (第 7 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する
  - ConnectionFactory と送信先の管理対象オブジェクトを作成する
- ブローカのクラスタ (126 ページの「ブローカクラスタの操作」を参照)
  - 中央設定ファイルを作成する
  - マスターブローカを使用する
- 持続 : ブローカを構成して、組み込み持続ではなく、プラグイン持続を使用する (付録 A「プラグイン持続の設定」を参照)

メンテナンス操作

さらに、運用環境では、MQ メッセージサーバ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 MQ 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
  - ブローカの自動作成機能を無効にする (70 ページの表 2-9 を参照)
  - アプリケーションのために、物理的な送信先を作成する (155 ページの「送信先の作成」を参照)

- 送信先へのユーザアクセスを設定する (193 ページの「ユーザの承認 : 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 (154 ページの「送信先の管理」を参照)
- 永続サブスクリプションを監視および管理する (158 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 (160 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
  - ブローカのメモリリソースを管理する
  - 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
  - 障害が発生したブローカを復元する
- 管理対象オブジェクトの管理
  - 必要に応じて、追加の ConnectionFactory と送信先の管理対象オブジェクトを作成する
  - ConnectionFactory の属性値を調整して、パフォーマンスとスループットを改善する (80 ページの「コネクションファクトリ管理対象オブジェクト」を参照)

MQ 管理ツール

MQ の管理ツールは、コマンド行ユーティリティとグラフィカルユーザインタフェース (GUI) 管理コンソールの 2 つのカテゴリに分類されます。コンソールは、ユーティリティ (`imqcmd`) と Object Manager 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を兼ね備えています。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、MQ 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 (`imqbrokerd`、`imqusermgr`、`imqdbmgr`、および `imqkeytool`) は、図 3-1 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。通常、特殊なタスクを実行するの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の概要」で説明します。

管理コンソール

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
- ブローカに物理的送信先を作成する
- 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

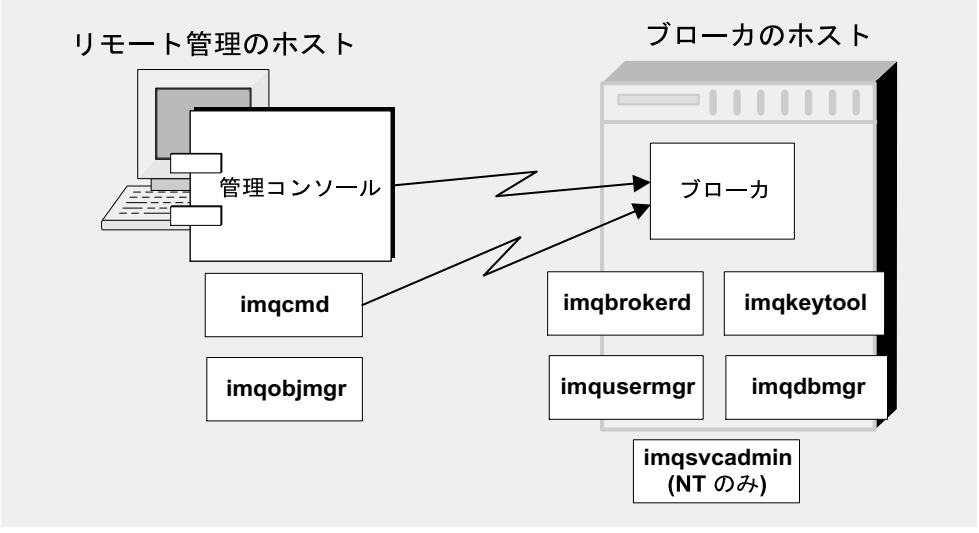
管理コンソールでは、実行できないタスクがいくつかあります。これには、ブローカの起動、ブローカのクラスタの作成、ブローカの特殊なプロパティの設定、ユーザデータベースの管理などが挙げられます。

第 4 章「管理コンソールのチュートリアル」では、コンソールについて説明し、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チュートリアルが用意されています。

コマンド行ユーティリティの概要

この節では、MQ 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、MQ のユーティリティを使用します。

図 3-1　ローカルおよびリモートの管理ユーティリティ



MQ 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、この章の後続の節で説明するように、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用方法については、後続の章でさらに詳しく説明します。

ブローカ (imqbrokerd): ブローカを起動するには、ブローカー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、追加の設定情報を指定したりします。このユーティリティについては、第 5 章「ブローカの起動と設定」で説明します。

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な送信先の作成、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制御、およびブローカのリソースの管理を行います。このユーティリティを実行するには、imqcmd コマンドを使用します。このユーティリティについては、第 6 章「ブローカとアプリケーションの管理」で説明します。

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、クライアントアプリケーションは移植可能になります。このユーティリティを実行するには、imqobjmgr コマンドを使用します。このユーティリティについては、第 7 章「管理対象オブジェクトの管理」で説明します。

ユーザマネージャ (imqusermgr): ユーザマネージャユーティリティを使用して、ユーザの認証および承認に使用するファイルベースのユーザリポジトリを設定します。このユーティリティを実行するには、imqusermgr コマンドを使用します。このユーティリティについては、第 8 章「セキュリティ管理」で説明します。

キーツール (imqkeytool): キーツールユーティリティを使用して、SSL 認証用の自己署名付き証明書を生成して、MQ のキーストアファイルに配置します。このユーティリティを実行するには、imqkeytool コマンドを使用します。このユーティリティについては、第 8 章「セキュリティ管理」で説明します。

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。このユーティリティを実行するには、imqdbmgr コマンドを使用します。詳細は、付録 A 「プラグイン持続の設定」を参照してください。

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、および削除します。詳細は、付録 C 「Windows のサービスとしてのブローカの使用」を参照してください。

コマンド行の構文

MQ のコマンド行インタフェースユーティリティは、単純なシェルコマンドです。つまり、それらのシェルコマンドが入力される Windows、Linux、または Solaris のコマンドシェルの観点から、ユーティリティ自体の名前がコマンドとそのサブコマンドになるか、あるいはオプションが単純にそのコマンドに渡される引数になります。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_Clause*]

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

一部のコマンドのオプションである *Subcommand_Clause* は、次のような構文になります。

[*Subcommand*] [*target*] [[*-option_name* [*operand*]]...]

次の 4 つの点に注意してください。

- 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、サブコマンド (およびターゲット) の後にオプションを指定する
- 引数に空白文字が入る場合、引数全体を引用符で囲む。通常、属性と値の組み合わせは引用符で囲んでおくことが望ましい
- 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ない。共通オプションについては、89 ページの表 3-1 を参照
- サブコマンドの引数 (ターゲット、オプション名、オペランド) は空白文字で区切る

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、少し複雑になります。管理者 (ユーザ) 名が `admin` で、対応するパスワードも `admin` の `myQueue` という名前が付いた `queue` タイプの送信先が破棄されます。この場合、確認は行われず、コンソールには何も出力されません。

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

共通のコマンド行オプション

MQ の管理ユーティリティ全体に共通するオプションを表 3-1 に示します。コマンド行でサブコマンドを指定した後に、これらのオプションを指定するという要件は別として、次に示すオプション (または、ユーティリティに渡されるその他のオプション) を特別な順序で入力する必要はありません。

表 3-1　共通の MQ 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する
-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する (imqcmd と imqobjmgr だけでサポートされる)
-s	画面に何も出力されないサイレントモードに切り替わる。imqbrokerd に -silent として指定する
-v	バージョン情報を表示する
-f	ユーザ確認の要求なしで、特定のアクションを実行する
-pre	(imqobjmgr でのみ使用) 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る。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つ
-javahome *path*	使用する代替の Java 2 ランタイムの場所を指定する

第 4 章

管理コンソールのチュートリアル

この章では、MQ メッセージサーバのグラフィカルインタフェースである、管理コンソールの使用方法を説明します。用意されているチュートリアルでは、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、コンソールを使用してブローカに接続し、管理する
- ブローカに物理的送信先を作成する
- オブジェクトストアを作成し、コンソールを使用して接続する
- オブジェクトストアに管理対象オブジェクトを追加する

チュートリアルは、送信先や、単純 JMS 準拠のアプリケーションである `SimpleJNDIClient` の実行に必要な管理対象オブジェクトを設定するように作成されています。チュートリアルの最後の部分では、このアプリケーションを実行します。

このチュートリアルでは、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。このチュートリアルは、『MQ 開発者ガイド』や、『管理者ガイド』のほかの章とは内容が異なります。

一部の MQ 管理タスクは、グラフィカルツールを使用しても実行できない場合があります。この場合、コマンド行ユーティリティを使用して、次のように実行する必要があります。

- 特定のブローカプロパティを設定する

  一部のブローカプロパティは、管理コンソールを使用しても設定できません。これらのプロパティは、第 5 章「ブローカの起動と設定」の 115 ページや 148 ページの「ブローカの更新」の手順を実行すると、設定されます。

- ブローカクラスタを作成する

  詳細は、126 ページの「ブローカクラスタの操作」を参照してください。

- ユーザのデータベースを管理する

  詳細は、184 ページの「ユーザの認証」を参照してください。

準備

このチュートリアルを開始する前に、MQ 製品をインストールする必要があります。詳細は、『MQ インストールガイド』を参照してください。このチュートリアルは Windows を基準にしており、Unix ユーザ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。

このチュートリアルでは、「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

管理コンソールは、次の作業を行うときに使用するグラフィカルツールです。

- ブローカに対する参照やコネクションを作成する
- ブローカを管理する
- ブローカがメッセージの配信に使用する、物理的送信先をブローカに作成する
- MQ 管理対象オブジェクトを配置したオブジェクトストアに接続する

  管理対象オブジェクトにより、JMS 準拠アプリケーションのメッセージングの要求を管理できます。詳細は、78 ページの「MQ 管理対象オブジェクト」を参照してください。

➤ 管理コンソールを起動するには

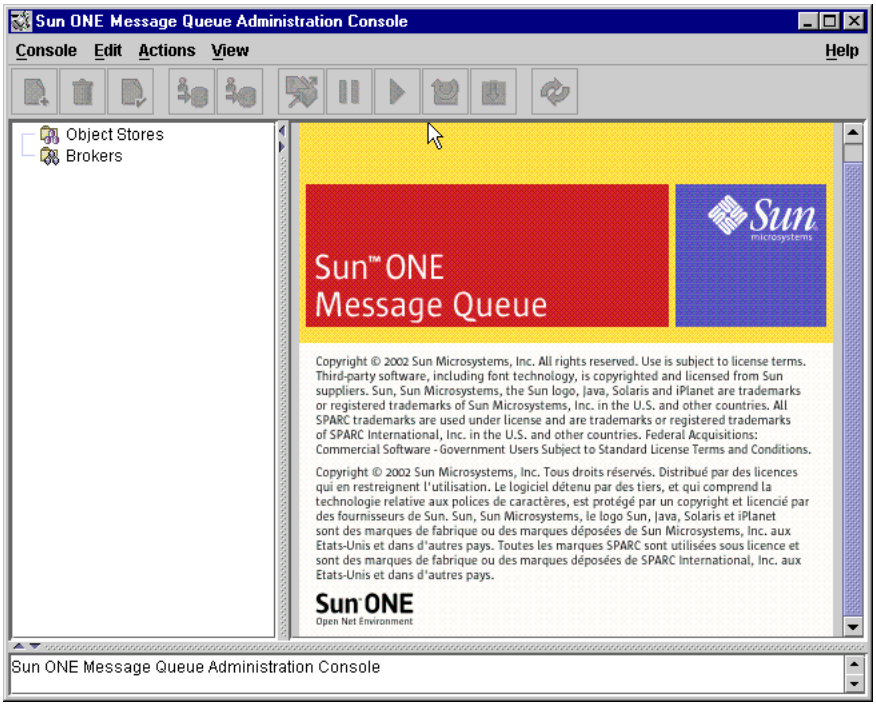
1. 「スタート」>「プログラム」>「Message Queue 3.0」>「Administration」の順に選択します。

   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。

   Windows 以外のユーザは、コマンドプロンプトで次のコマンドを入力します。

   `$IMQ_HOME/bin/imqadmin (Solaris では、/usr/bin/imqadmin)`

2. 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、メニュ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、右側にそれより大きいペイン (この図では Sun ONE Message Queue 製品を表すグラフィックが表示されている)、および一番下に状態のペインという構成になっています。

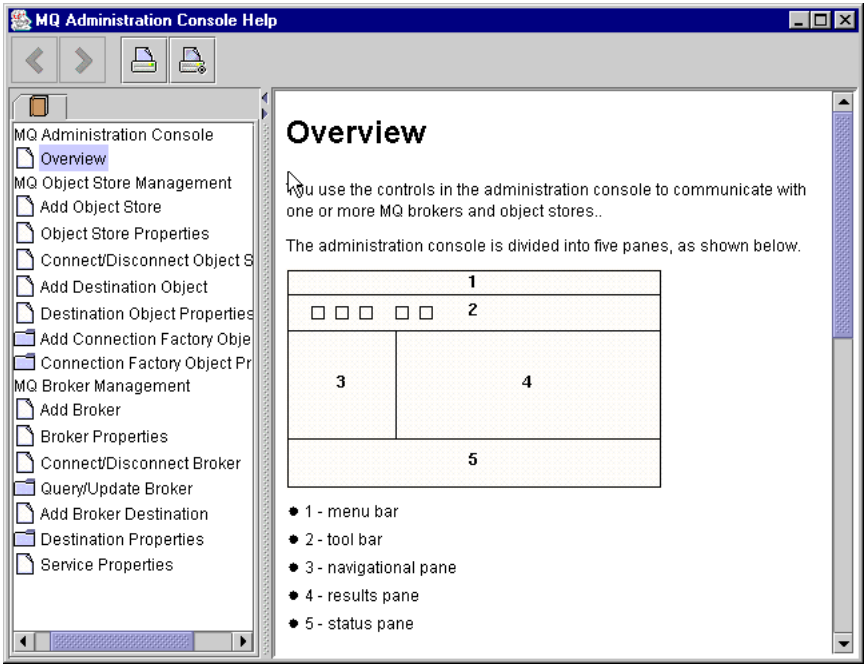
チュートリアルには完全な情報が記載されているわけではないので、まず管理コンソールに関するヘルプ情報の使い方を理解しておきましょう。

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には

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。左のペインには目次が表示され、右のペインには左の目次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右の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内容ペインを見ます。概要、オブジェクトストアの管理、ブローカの管理という 3 つの領域でトピックが構成されています。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のヘルプがあり、各ファイルにはダイアログ、またはタブが 1 つだけのヘルプがあります。

   最初に行うコンソールの管理タスクは、97 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目次のペインにある、「Add Broker (ブローカを追加)」アイテムをクリックします。

   目次の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。

ブローカを操作する

ブローカには、MQ メッセージングシステムの配信サービスが用意されています。メッセージ配信は 2 階層のプロセスで、まずメッセージはブローカの物理的送信先に配信され、次に 1 つまたは複数のコンシュームするクライアントに配信されます。

ブローカの操作は、次のタスクで構成されます。

- ブローカの起動と設定

  ブローカの起動は、Windows の「スタート」>「プログラムメニュー」または `imqbrokerd` コマンドを使用または行います。`imqbrokerd` コマンドを使用する場合、コマンド行オプションを利用してブローカの設定情報を指定できます。「プログラムメニュー」を使用する場合、コンソールまたは第 5 章「ブローカの起動と設定」にある別の方法を使用して、設定情報を指定できます。

- 管理コンソールまたは `imqcmd` コマンドを使用して、ブローカとそのサービスを管理する

- クライアントアプリケーションに必要な物理的送信先を作成する

- リソースの使用を監視し、スループットと信頼性を向上させる

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との通信をサポートしています。このサポートはさまざまなコネクションサービスを使用して行われ、任意のコネクションサービスまたはすべてのコネクションサービスを実行するようにブローカを設定できます。コネクションサービスについての詳細は、48 ページの「コネクションサービス」を参照してください。

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。ブローカは、次の方法で起動します (第 5 章「ブローカの起動と設定」を参照)。

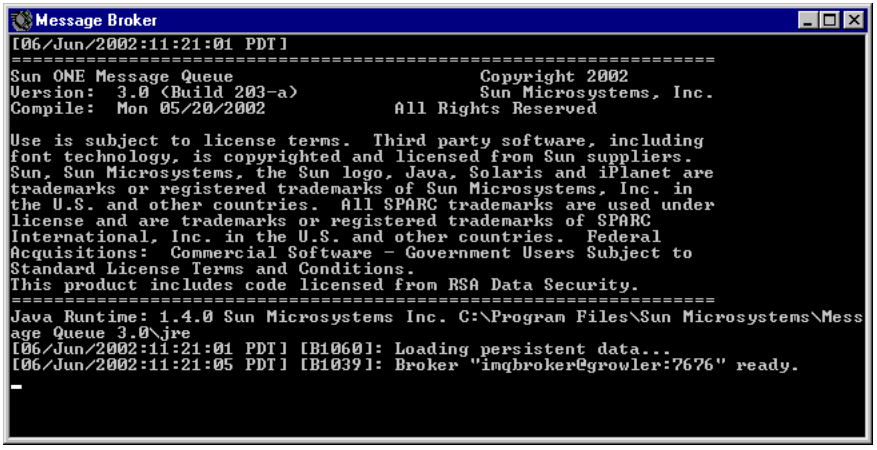
➤ ブローカを起動するには

1. 「スタート」>「プログラム」>「Message Queue 3.0」>「Broker (ブローカ)」の順に選択します。

   Windows 以外では、次のコマンドを入力してブローカを起動します。

   `%$IMQ_HOME/bin/imqbrokerd` (Solaris では、`/usr/bin/imqbrokerd`)

   ブローカプロセスウィンドウが表示されます。ブローカにはあらかじめ準備された名前が指定されます。



2. 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には

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。

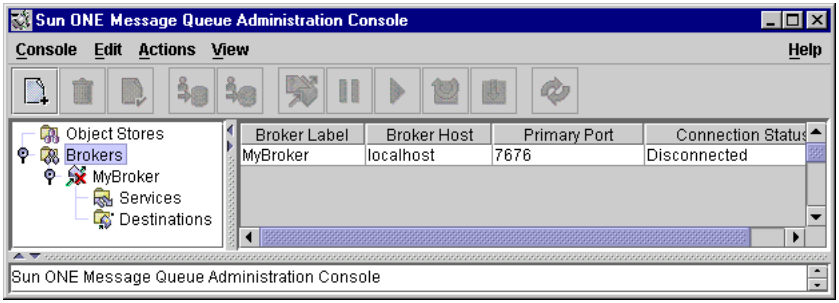
   これにより、管理コンソールでブローカを識別するラベルが作成されます。

   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。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が表示されます。このダイアログを使用して、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。

管理者パスワードの変更

ブローカを追加した際、パスワードを指定しなかった場合は、ブローカに接続するときにパスワードが要求されます。セキュリティを高めるために、接続前にデフォルトの管理者パスワード (admin) を変更することをお勧めします。

➤ 管理者パスワードを変更するには

1. コマンドプロンプトウィンドウを開くか、あるいはすでに開かれている場合は、最前面に置きます。
2. 次のようにコマンドを入力します。ただし、abracadabra には自分のパスワードを入力します。ここで指定したパスワードが、デフォルトのパスワード admin に置き換わります。

   ```
   imqusermgr update -u admin -p abracadabra
   ```

   (Solaris および Linux では、この操作を行うには root である必要があります。)

変更は直ちに反映されます。MQ コマンド行ユーティリティや管理コンソールを使用するときは必ず、この新しいパスワードを指定する必要があります。

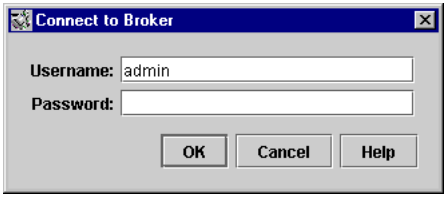
クライアントは、管理者と異なるコネクションサービスを使用しますが、デフォルトのユーザ名とパスワードが割り当てられているため、詳細な管理の設定を行わなくても MQ をテストできます。デフォルトでは、クライアントは guest というパスワードを持つ guest というユーザとしてブローカに接続できます。ただし、できるだけ早くクライアントに安全なユーザ名とパスワードを設定してください。詳細は、184 ページの「ユーザの認証」を参照してください。

ブローカに接続する

➤ ブローカに接続するには

1. 「MyBroker」を右クリックし、「Connect to Broker (ブローカに接続)」を選択します。

   ユーザ名とパスワードを指定するダイアログが表示されます。



2. 「Password (パスワード)」フィールドに admin または 98 ページの「管理者パスワードの変更」でパスワードに指定した値を入力します。

   ユーザ名の 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。

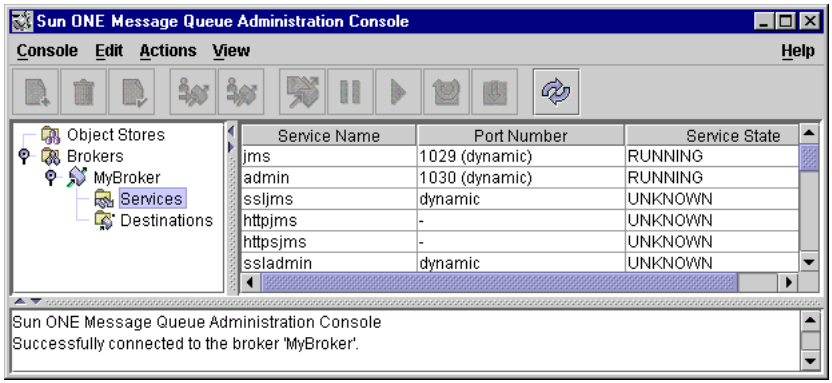
コネクションサービスを表示する

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には

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

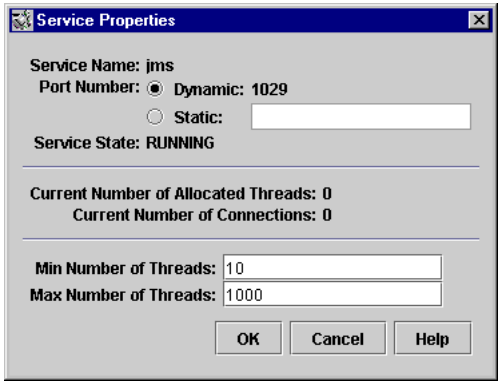
   結果ペインに利用可能なサービスが一覧表示されます。各サービスに対して、名前、ポート番号、状態が表示されます。



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 (サービスのプロパティ)」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

ブローカに物理的送信先を明示的に作成して、JMS 準拠のアプリケーションが正常に実行されるようにする必要があります。ブローカの送信先自動作成が有効になっていて、物理的送信先を動的に作成できる場合は、これを作成する必要はありません。

送信先の自動作成は、開発環境で利用できます。ただし、運用設定では、送信先の自動作成を無効にして、明示的に作成した物理的送信先をブローカが使用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送信先を十分に認識できます。

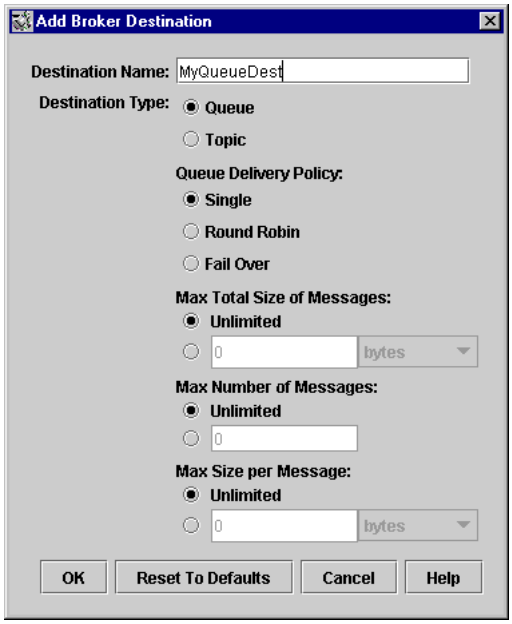
`imq.autocreate.topic` または `imq.autocreate.queue` プロパティを設定して、ブローカの自動作成送信先の追加を制御できます。詳細は、69 ページの「自動作成 (および管理者によって作成) された送信先」を参照してください。

ここでは、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には

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. 「Queue Delivery Policy (キュー配信ポリシー)」で「Single (シングル)」が選択されていることを確認します。
5. 「OK」をクリックして、物理的送信先を追加します。

   送信先が結果ペインに表示されます。

物理的送信先を操作する

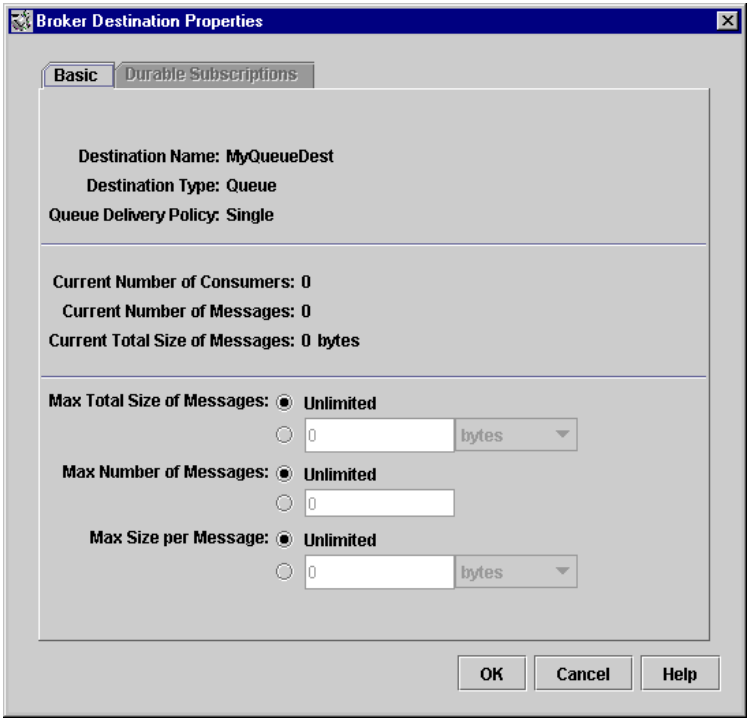
ブローカに物理的送信先を追加すると、次のタスク実行できるようになります。

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

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

   次のダイアログが表示されます。



   変更できるプロパティは、そのキューに配信されるメッセージのサイズと数に関連するプロパティだけです。

4. 「Cancel (取消し)」をクリックしてダイアログを閉じます。

➤ 送信先からメッセージをパージするには

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。

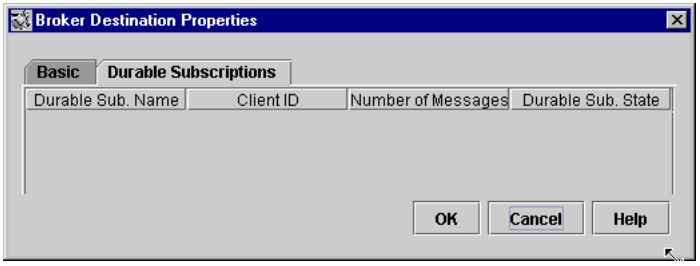
➤ 送信先を削除するには

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。

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。

トピック送信先に関する情報を取得する

トピック送信先に関するダイアログには、永続サブスクリプションに関する情報が一覧になった追加タブがあります。



このダイアログを使用して、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、LDAP ディレクトリサーバまたはファイルシステムストア (ファイルシステムにあるディレクトリ) であっても、MQ 管理対象オブジェクトを格納するために使用されます。MQ 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が使用するオブジェクトに関する MQ 固有の実装と設定情報をカプセル化します。

クライアントコード内で管理対象オブジェクトをインスタンス化および設定することは可能ですが、管理者がこれらのオブジェクトを作成、および設定して、クライアントアプリケーションが標準 JMDI 検索コードを介してアクセスできるオブジェクトストアに格納することをお勧めします。これにより、クライアントコードはプロバイダに依存しなくなります。

管理対象オブジェクトについては、78 ページの「MQ 管理対象オブジェクト」を参照してください。

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には

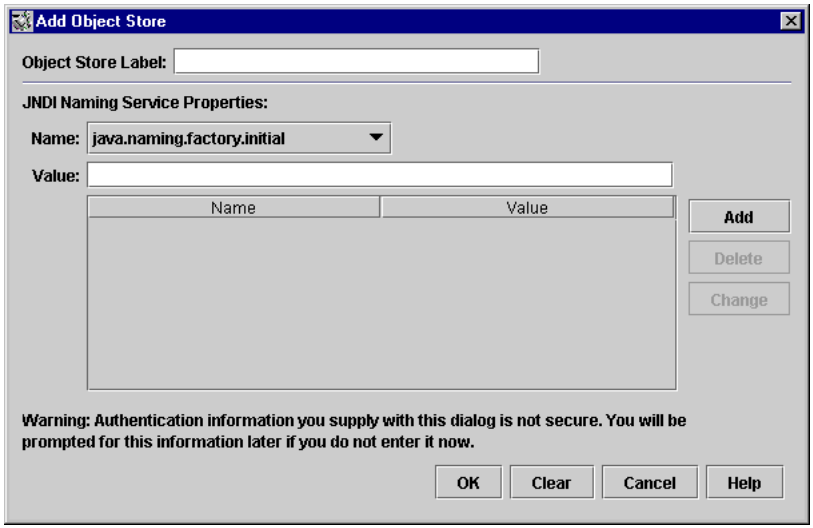
1. C ドライブに `Temp` という名前のフォルダが存在しない場合は、ここで作成します。

   このチュートリアル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 以外では、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)」プルダウンメニューから、`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`

   これは、ファイルシステムストアを使用することを表しています (LDAP ストアの場合は、`com.sun.jndi.ldap.LdapCtxFactory` と指定する)

   運用環境では、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。サーバの設定と JNDI 検索については、171 ページの「オブジェクトストアの属性」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)」プルダウンメニューから、`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
   ```

   Windows 以外では、`file:///tmp` を指定します。

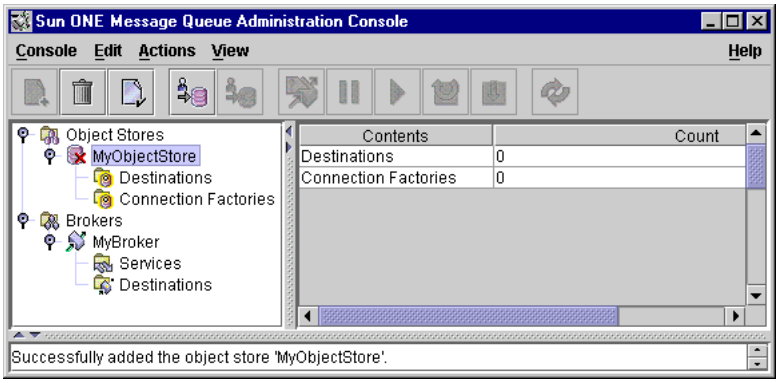
9. 「Add (追加)」ボタンをクリックします。

   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には

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には

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

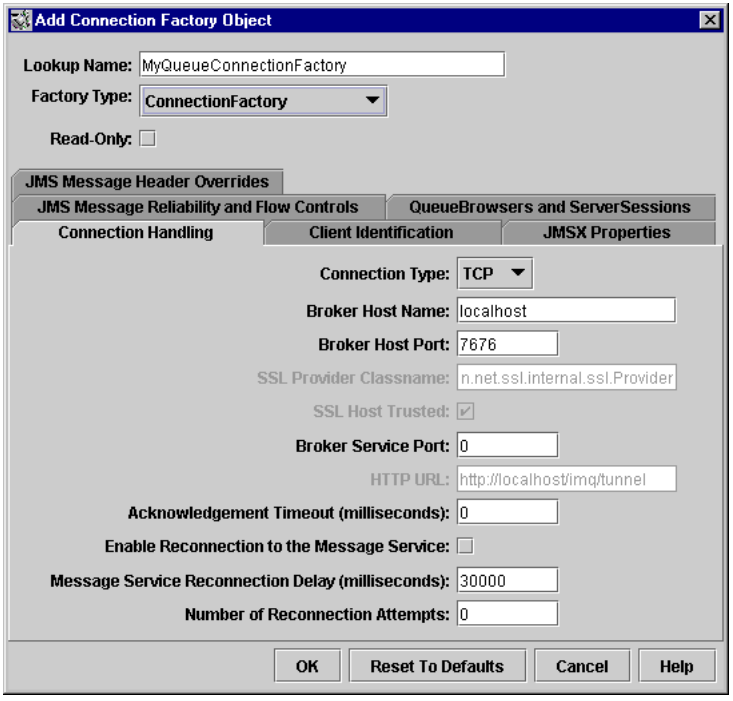
管理コンソールを使用して、コネクションファクトリを作成および設定できます。コネクションファクトリは、クライアントコードがブローカに接続するときに使用されます。コネクションファクトリを設定すると、作成するときに使用されるコネクションの動作を制御できます。

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『MQ 開発者ガイド』を参照してください。

➤ 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には

1. まだ接続されていない場合は、MyObjectStore に接続します (108 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)」ダイアログが表示されます。



3. 「LookupName (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 `SimpleAdmin.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. 「Broker Host Name (ブローカのホスト名)」フィールドと「Broker Host Port (ブローカのホストのポート)」フィールドに、クライアントが接続しようとしているブローカのホスト名とポートを入力します。

   このチュートリアルでは、クライアントはデフォルトのブローカ、つまりポート `7676` の `localhost` にあるブローカに接続するため、これらのフィールドを変更する必要はありません。

6. このダイアログ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。

7. 「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。

8. 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。

送信先管理対象オブジェクトを追加する

送信先管理対象オブジェクトは、ブローカーの物理的送信先に関係しています。送信先管理対象オブジェクトは送信先を指すので、プロバイダ固有の送信先の命名方法および設定方法に関係なく、クライアントは物理的送信先を検索できます。

JMS 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。

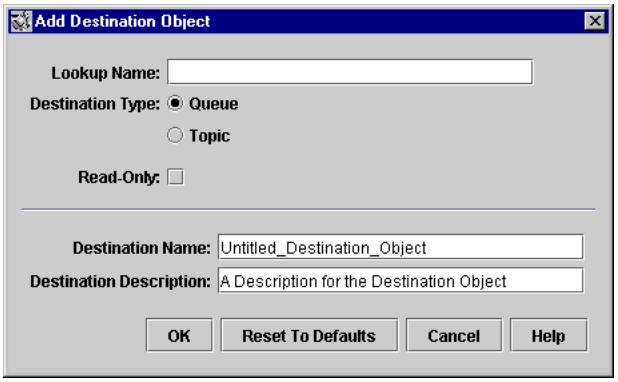
- その管理対象オブジェクトに関連する物理的送信先を作成した場合、ブローカはその物理的送信先にメッセージを配信する
- 物理的送信先を作成せず、物理的送信先の自動作成が有効になっている場合、ブローカ自身が物理的送信先を作成し、その送信先にメッセージを配信する
- 物理的送信先を作成せず、物理的送信先の自動作成が無効になっている場合、ブローカは物理的送信先を作成できず、メッセージも配信できない

チュートリアルの次の部分では、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には

1. ナビゲーションペインの「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");
   ```

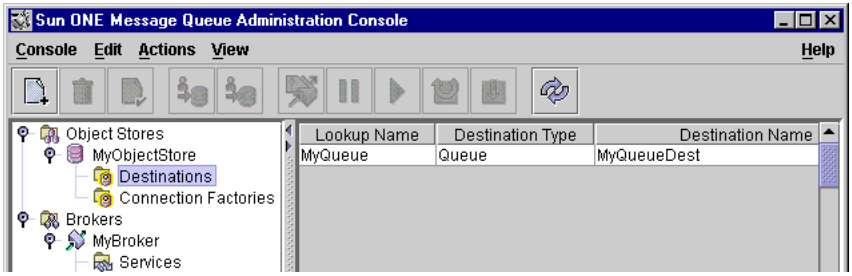
4. 「Destination Type (送信先タイプ)」の「Queue (キュー)」ラジオボタンを選択します。

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。

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。



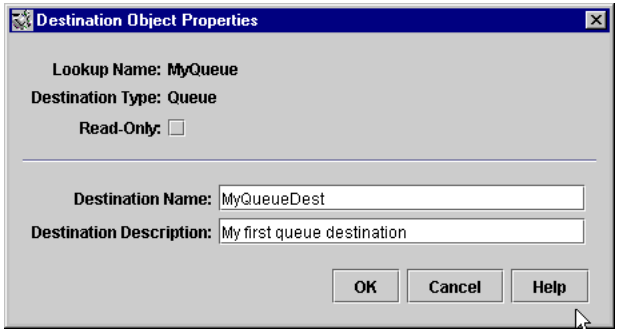
管理対象オブジェクトのプロパティ

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には

1. ナビゲーションペインで「Destinations (送信先)」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。



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

このチュートリアルには、SimpleJNDIClient というサンプルアプリケーションが用意されています。このサンプルアプリケーションでは、チュートリアルのこれまでの部分で作成した、MyQueueDest という名前のキュー送信先、MyQueueConnectionFactory という名前のキューコネクションファクトリ管理対象オブジェクト、および MyQueue という名前のキュー管理対象オブジェクトが使用されます。

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。

➤ SimpleJNDIClient アプリケーションを実行するには

1. JAVA_HOME 環境変数が、JDK のインストールされたディレクトリを指していることを確認します。
2. SimpleJNDIClient 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_HOME/demo/jms (Solaris では /usr/demo/imq/jms)

   SimpleJNDIClient.class ファイルがない場合、または変更後にアプリケーションをコンパイルする必要がある場合は、『MQ 開発者ガイド』の「クイックスタートチュートリアル」にある手順に従ってください。
3. CLASSPATH 変数を設定して、jar ファイル (jms.jar、imq.jar、fscontext.jar、および providerutil.jar) と、SimpleJNDIClient.java が存在する現在のディレクトリを含めます。

   JDK 1.3 またはそれ以前のものを使用している場合、jndi.jar も含める必要があります。
4. アプリケーションを実行する前に、ソースファイルの SimpleJNDIClient.java を開き、一通り読んでおきます。

   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、チュートリアルを使用して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。
5. SimpleJNDIClient アプリケーションを実行します

   C:> java SimpleJNDIClient (Windows の場合)

   % java SimpleJNDIClient file:///tmp (Windows 以外の場合)

   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 SimpleJDNIClient file:///tmp
Usage: java SimpleJDNIClinet [Context.PROVIDER_URL]
On Unix:
        java SimpleJDNIClient file:///tmp
On Windows:
        java SimpleJDNIClient file:///C:Temp
Publishing a message to Queue: MyQueueDest
Received the following message: Hello World.
================================================
```

第 5 章

ブローカの起動と設定

MQ をインストールしたあとで、imqbrokerd コマンドを使用してブローカを起動します。ブローカの設定は、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。

この章では、imqbrokerd コマンドの構文と、コマンド行オプションや設定ファイルを使用してブローカを設定する方法について説明します。さらに、次の方法についても説明します。

- ブローカのインスタンス設定ファイルの編集
- ブローカクラスタの操作
- ブローカのログ作成の制御

Windows のサービスとしてブローカを起動し、使用する方法については、237 ページの「Windows のサービスとしてのブローカの使用」を参照してください。

設定ファイル

ブローカの設定に使用する、インストール済みの設定ファイルは、次のディレクトリにあります。

IMQ_HOME/lib/props/broker
(Solaris では /usr/share/lib/imq/props/broker)

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、編集はできない。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必要がある

- MQ のインストール時に指定されたプロパティを含む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ない

また、初めてブローカを実行すると、インスタンス設定ファイルが作成されます。このファイルを使用して、ブローカのインスタンスの設定プロパティを指定できます。このファイルは、管理コマンドに応答してブローカに管理されます。また、直接編集することも可能ですが慎重に行う必要があります。インスタンス設定ファイルは、次の場所に格納されています。

`IMQ_VARHOME/instances/`*brokerName*`/props/config.properties`
(Solaris では、`/var/imq/instances/`*brokerName*`/props/config.properties`)

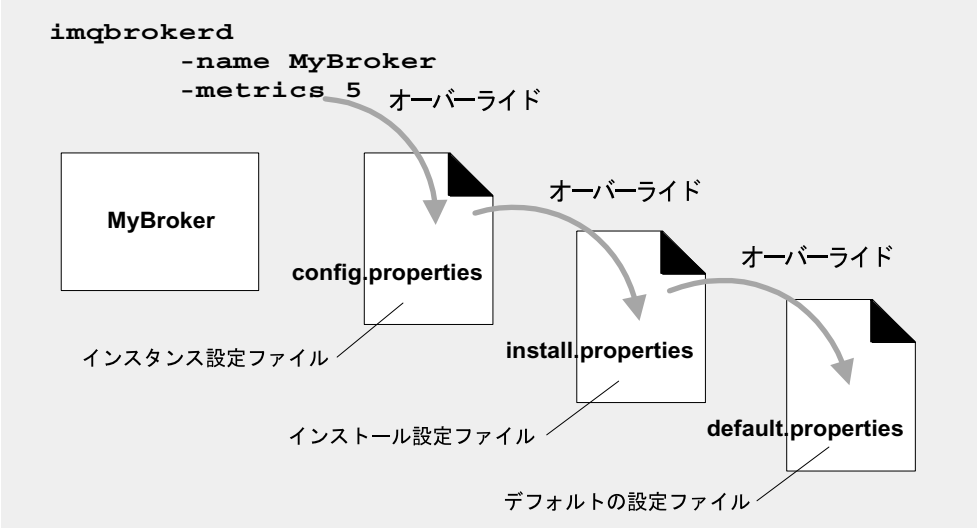
brokerName には、ブローカ名が表示されます (デフォルトでは `imqbrokerd`)。インスタンス設定ファイルを編集して、設定を変更できます (118 ページの「インスタンス設定ファイルの編集」を参照)。

クラスタでブローカを接続する場合 (71 ページの「複数ブローカの設定 (クラスタ)」を参照)、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、126 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、システムは異なる設定ファイルのプロパティ値をマージします。インストール時に設定された値、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 117 ページの図 5-1 で図解します。

図 5-1　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 MQ 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、次のエントリでは、自動作成キューのキュータイプを定義します。

```
imq.queue.default=single
```

次のエントリでは、メッセージの有効期限切れタイムアウト値を定義します。

```
imq.message.expiration.timeout=90
```

118 ページの表 5-1 に、ブローカ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。

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。

ブローカは起動時にだけ config.properties ファイルを読み込みます。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、148 ページの表 6-5 を参照
- ブローカインスタンスの停止中に config.properties ファイルを編集し、その後インスタンスを再起動する (Solaris および Linux プラットフォームでは、最初にブローカインスタンスを起動したユーザだけが config.properties ファイルを編集するアクセス権を持っている)

表 5-1 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。

表 5-1 ブローカインスタンス設定プロパティ

プロパティ名	データ型	デフォルト値	参照先
imq.accesscontrol.enabled	ブール	true	62 ページの表 2-6
imq.accesscontrol.file.filename	文字列	accesscontrol.properties	62 ページの表 2-6
imq.authentication.basic.user_repository	文字列	file	62 ページの表 2-6
imq.authentication.client.response.timeout	整数 (秒)	180	62 ページの表 2-6
imq.authentication.type	文字列	digest	62 ページの表 2-6
imq.autocreate.queue	ブール	true	70 ページの表 2-9
imq.autocreate.topic	ブール	true	70 ページの表 2-9
imq.cluster.url	文字列	null	74 ページの表 2-10
imq.keystore.*property_name*			201 ページの表 8-8
imq.log.console.output	文字列	ERROR|WARNING	66 ページの表 2-8
imq.log.console.stream	文字列	ERR	66 ページの表 2-8

* 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 バイト単位で表されます。例 : 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 5-1 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	データ型	デフォルト値	参照先
imq.log.file.dirpath	文字列	IMQ_VARHOME/instances/*brokerName*/log	66 ページの表 2-8
imq.log.file.name	文字列	log.txt	66 ページの表 2-8
imq.log.file.output	文字列	ALL	66 ページの表 2-8
imq.log.file.rolloverbytes	整数 (バイト)	0	66 ページの表 2-8
imq.log.file.rolloversecs	整数 (秒)	604800	66 ページの表 2-8
imq.log.level	文字列	INFO	66 ページの表 2-8
imq.log.syslog.facility	文字列	LOG_DAEMON	66 ページの表 2-8
imq.log.syslog.logpid	ブール	true	66 ページの表 2-8
imq.log.syslog.logconsole	ブール	false	66 ページの表 2-8
imq.log.syslog.identity	文字列	imqbrokerd_${imq.instancename}	66 ページの表 2-8
imq.log.syslog.output	文字列	ERROR	66 ページの表 2-8
imq.message.expiration.interval	整数 (秒)	60	55 ページの表 2-4
imq.message.max_size	バイト文字列 *、0 (制限なし)	70m	55 ページの表 2-4
imq.metrics.enabled	ブール	true	66 ページの表 2-8
imq.metrics.interval	整数 (秒)	0	66 ページの表 2-8
imq.passfile.enabled	ブール	false	62 ページの表 2-6
imq.passfile.dirpath	文字列	IMQ_HOME/etc/etc/imq (Solaris の場合)	62 ページの表 2-6
imq.passfile.name	文字列	passfile	62 ページの表 2-6
imq.persist.file.destination.file.size	バイト文字列 *	1m	58 ページの表 2-5
imq.persist.file.message.cleanup	ブール	false	58 ページの表 2-5

* 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 バイト単位で表されます。例 : 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 5-1 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	データ型	デフォルト値	参照先
`imq.persist.file.message.fdpool.limit`	整数	`25` (Solaris および Linux の場合) `1024` (Windows の場合)	58 ページの表 2-5
`imq.persist.file.message.filepool.cleanratio`	整数	`0`	58 ページの表 2-5
`imq.persist.file.message.filepool.limit`	整数	`10000`	58 ページの表 2-5
`imq.persist.file.sync.enabled`	ブール	`false`	58 ページの表 2-5
`imq.persist.jdbc.`*property_name*			209 ページの表 A-1
`imq.persist.store`	文字列	`file`	58 ページの表 2-5
`imq.portmapper.port`	整数	`7676`	50 ページの表 2-3
`imq.queue.deliverypolicy`	文字列	`single`	70 ページの表 2-9
`imq.redelivered.optimization`	ブール	`true`	55 ページの表 2-4
`imq.`*resource_state*`.threshold`	整数 (パーセント)	`0` (green) `60` (yellow) `75` (orange) `90` (red)	55 ページの表 2-4
`imq.service.activelist`	リスト	`jms`、`admin`	50 ページの表 2-3
`imq.`*service_name*`.accesscontrol.enabled`	ブール	システム全体のプロパティから値を継承	62 ページの表 2-6
`imq.`*service_name*`.accesscontrol.file.filename`	文字列	システム全体のプロパティから値を継承	62 ページの表 2-6
`imq.`*service_name*`.authentication.type`	文字列	システム全体のプロパティから値を継承	62 ページの表 2-6
`imq.`*service_name*`.max_threads`	整数	`1000` (jms) `500` (ssljms) `500` (httpjms) `500` (httpsjms) `50` (admin)	50 ページの表 2-3

* 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 バイト単位で表されます。例 : 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 5-1 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	データ型	デフォルト値	参照先
imq.*service_name*.min_threads	整数	10 (jms) 10 (ssljms) 10 (httpjms) 10 (httpsjms) 4 (admin)	50 ページの表 2-3
imq.*service_name*.*protocol type*.hostname	文字列	null	50 ページの表 2-3
imq.*service_name*.*protocol type*.port	整数	0	50 ページの表 2-3
imq.*service_name*.threadpool_model	文字列	dedicated (jms) dedicated (ssljms) dedicated (httpjms) dedicated (httpsjms) dedicated (admin)	50 ページの表 2-3
imq.shared.connectionMonitor_limit	整数	512 (Solaris および Linux の場合) 64 (Windows の場合)	50 ページの表 2-3
imq.system.max_count	整数、 0 (制限なし)	0	55 ページの表 2-4
imq.system.max_size	バイト文字列 *、 0 (制限なし)	0	55 ページの表 2-4
imq.transaction.autorollback	ブール	false	55 ページの表 2-4
imq.user_repository.ldap.*property_name*			190 ページの表 8-5

* 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 バイト単位で表されます。例 : 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。

ブローカの起動

ブローカを起動し、1 つまたは複数のプロパティ値をオーバーライドするには、imqbrokerd コマンドを使用し、有効なオプションを指定します。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。コマンド行オプションは、設定プロパティファイルに書き込まれません。

imqbrokerd コマンドの構文は、次のとおりです (引数はスペースで区切る)。

```
imqbrokerd [[ -Dproperty=value]...]
     [ -backup filename]
     [ -cluster "[broker] [[,broker]...]"
     [ -dbuser username] [ -dbpassword password]
     [ -javahome path | -jrehome path]
     [ -ldappassword password]
     [ -license name]
     [ -loglevel level]
     [ -metrics number]
     [ -name brokername ] [ -port number]
     [ -shared]
     [ -password keypassword] [ -passfile filename]
     [ -remove instance]
     [ -reset data]
     [ -restore filename]
     [ -shared]
     [ -silent] [ -tty]
     [ -version] [ -vmargs arg [[arg]...]
```

たとえば、デフォルトのブローカ名および設定を使用する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、ポート 7676 のポートマッパ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。

表 5-2 に、imqbrokerd コマンドのオプションと、存在する場合は各オプションから影響を受ける設定プロパティを示します。

注 Solaris および Linux プラットフォームでは、設定情報と持続データを保持するディレクトリに対するアクセス権は、最初にブローカインスタンスを起動するユーザの umask コマンドによる設定に依存します。したがって、ブローカインスタンスを正常に動作させるためには、それ以降も所定のユーザだけが起動する必要があります。

表 5-2　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*	影響なし	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する。ブローカクラスタに対してのみ適用可能。131 ページの「マスターブローカの設定変更レコードのバックアップ」を参照
-cluster"[*broker*][[,*broker*]...]" ブローカは次のどちらか • *host*[:*port*] • [*host*]:*port*	imq.cluster.brokerlist に接続先となるブローカのリストを設定する	特定のホストやポートにある全ブローカに接続する。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る。ホストに値を指定しない場合、localhost が使用される。ポートに値を指定しない場合、7676 が使用される。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、126 ページの「ブローカクラスタの操作」を参照
-dbpassword *password*	imq.persist.jdbc.password に特定のパスワードを設定する	プラグインの JDBC 準拠のデータベースに対するパスワードを指定する。付録 A 「プラグイン持続の設定」を参照
-dbuser *userName*	imq.persist.jdbc.user に特定のユーザ名を設定する	プラグインの JDBC 準拠のデータベースに対するユーザ名を指定する。付録 A 「プラグイン持続の設定」を参照
-D*property=value*	システムプロパティを設定する。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する	指定したプロパティを指定した値に設定する。ブローカ設定プロパティについては、118 ページの表 5-1 を参照 **注意**：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、十分に確認すること。誤った値を渡した場合、システムからの警告なしに MQ でプロパティを設定できなくなる
-javahome *path*	影響なし	代替 Java 2 互換 JDK へのパスを指定する。デフォルトでは、バンドルされたランタイムを使用する
-jrehome *path*	影響なし	Java 2 JRE へのパスを指定する
-ldappassword *password*	imq.user_repository.ldap.password に特定のパスワードを設定する	LDAP ユーザリポジトリにアクセスするためのパスワードを指定する。190 ページの「ユーザリポジトリに LDAP サーバを使用する」を参照

表 5-2　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 [*name*]	影響なし	使用している MQ 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する。ライセンス名を指定しない場合、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る。インストールされた MQ エディションによって、*name* 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) のどれかになる。26 ページの「製品エディション」を参照
-loglevel *level*	imq.broker.log.level に特定のレベルを設定する	ログ作成レベルを、NONE、ERROR、WARNING、INFO のどれかに指定する。デフォルト値は INFO。詳細は、64 ページの「ロガー」を参照
-metrics *int*	imq.metrics.report.interval に特定の秒数を設定する	メトリックスが特定の秒間隔で報告されるように指定する
-name *brokerName*	imq.instancename に特定の名前を設定する	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する。ブローカ名を指定しない場合、ファイル名は imqbrokerd に設定される **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る
-passfile *filename*	imq.passfile.enabled に true を設定する。jmq.passfile.dirpath にファイルが含まれるパスを設定する。imq.passfile.name にファイル名を設定する	SSL キーストア、LDAP ユーザリポジトリ、および JDBC 準拠のデータベースのパスワードの読み込み元となるファイル名を指定する。詳細は、205 ページの「passfile の使用」を参照してください。
-password *keypassword*	imq.keystore.password に特定のパスワードを設定する	SSL 証明書のキーストアに対するパスワードを指定する。詳細は、59 ページの「セキュリティマネージャ」を参照
-port *number*	imq.portmapper.port に特定の数を設定する	ブローカのポートマッパのポート番号を指定する。デフォルトでは 7676 に設定されている。同一サーバ上でブローカの 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が異なるポート番号となる必要がある

表 5-2　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
-remove instance	影響なし	ブローカのインスタンスが削除される。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る
-reset store| messages| durables| props	影響なし	指定された引数に応じて、ブローカの持続ストアまたはストアのサブセットをリセットするか、あるいはブローカのプロパティをリセットする ブローカの持続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、および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る。このため、データが消去された状態で、ブローカを起動できる。すべての持続メッセージだけ、またはすべての永続サブスクリプションだけを消去することも可能 (次回起動時に持続ストアをリセットしたくない場合は、-reset オプションを使用せずにブローカを再起動する)。詳細は、56 ページの「持続マネージャ」を参照。 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、プロパティはすべてデフォルト値となる
-restore *filename*	影響なし	マスターブローカの設定変更レコードを、指定したファイルに置き換える。131 ページの「マスターブローカの設定変更レコードの復元」を参照
-shared	imq.jms.threadpool_model に shared を設定する	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する。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、ブローカにサポートされるコネクション数が増加する。詳細は、48 ページの「コネクションサービス」を参照
-silent	imq.log.console.output に NONE を設定する	コンソールへのログ作成をオフにする
-tty	imq.log.console.output に ALL を設定する	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する。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る

表 5-2　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
-version	影響なし	インストールされた製品のバージョン番号を表示する
-vmargs arg [[arg]...]	影響なし	Java VM に渡す引数を指定する。引数はスペースで区切られる。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、引用符で囲む。例 : imqbrokerd -tty -vmargs "-Xmx128m -Xincgc"

ブローカクラスタの操作

この節では、ブローカクラスタを設定するために使用するプロパティ、ブローカを接続する 2 つの方法、およびクラスタを管理する方法について説明します。この機能は、MQ Enterprise Edition でのみ利用できます (26 ページの「製品エディション」を参照)。

クラスタの概要については、71 ページの「複数ブローカの設定 (クラスタ)」を参照してください。

クラスタを操作するときは、クラスタにあるすべてのブローカのホスト間でクロックの同期が取れていることを確認してください。

クラスタ設定プロパティ

ブローカをクラスタに接続する場合、接続されたすべてのブローカで、クラスタ設定プロパティに同じ値を指定する必要があります。これらのプロパティにより、クラスタにあるブローカの関係が記述されます。表 5-3 で、クラスタ関連の設定プロパティの概要を説明します。

表 5-3　クラスタ設定プロパティ

プロパティ	説明
imq.cluster.brokerlist	カンマで区切られたリスト内のクラスタの全ブローカを指定する。この場合、各アイテムではブローカのホストやポートが指定される。 次に例を示す。host1:3000, host2:8000, ctrhost
imq.cluster.masterbroker	マスターブローカのホストとポートを指定する。 この値は運用環境の場合に設定する。 たとえば、ctrhost:7676

表 5-3　クラスタ設定プロパティ (続き)

プロパティ	説明
imq.cluster.url	クラスタ設定ファイルの場所を指定する。 次に例を示す。 http://webserver/imq/cluster.properties file:/net/mfsserver/imq/cluster.properies
imq.cluster.port	クラスタ内にある各ブローカに対して、クラスタコネクションサービスのポート番号を指定する。クラスタコネクションサービスは、クラスタにあるブローカ間での内部通信に使用される。 デフォルトは、0 (ポートは動的に割り当てられる)
imq.cluster.hostname	クラスタ内の各ブローカに対して、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など、使用可能な複数のホストがある場合に、クラスタコネクションサービスのバインド先となるホスト (ホスト名または IP アドレス) を指定する。クラスタコネクションサービスは、クラスタにあるブローカ間での内部通信に使用される。 デフォルトは、null (すべての使用可能なホスト)

次の 2 つの方法のうちどちらかを使用して、クラスタプロパティを設定できます。

- クラスタ関連の設定プロパティを、各ブローカのインスタンス設定ファイル (または各ブローカを起動するコマンド行) で設定します。たとえば、ブローカ A (host1 上、ポート 7676)、ブローカ B (host2 上、ポート 5000)、およびブローカ C (ctrlhost 上、ポート 7676) を接続する場合、ブローカ A、B、および C のインスタンス設定ファイルに、次のプロパティを設定する必要があります。

  ```
  imq.cluster.brokerlist=host1, host2:5000, ctrlhost
  ```

  クラスタ設定を変更する場合、この方法では、すべてのブローカにあるクラスタ関連のプロパティを更新する必要があります。

- クラスタ設定プロパティを、1 つの中心的なクラスタ設定ファイルで設定します。これらのプロパティには、接続されるブローカのリスト (imq.cluster.brokerlist) が含まれます。またオプションで、マスターブローカのアドレス (imq.cluster.masterbroker) も含まれます。

  この方法を使用する場合、クラスタ内の各ブローカに対して imq.cluster.url を設定してクラスタ設定ファイルの場所を指すようにする必要があります。管理を簡単にするという点から、クラスタの設定にはこの方法をお勧めします。

次のサンプルコードは、クラスタ設定ファイルの内容を表示しています。host1 と ctrlhost は両方とも、デフォルトのポートで動作しています。これらのプロパティは、host1 と ctrlhost がクラスタで接続され、ctrlhost がマスターブローカであることを指定しています。

```
imq.cluster.brokerlist=host1,host2:5000,ctrlhost
imq.cluster.masterbroker=ctrlhost
```

このクラスタで接続されている各ブローカのインスタンス設定ファイルは、クラスタ設定ファイルの URL を含んでいる必要があります。次に例を示します。

```
imq.cluster.url=file:/home/cluster.properties
```

ブローカの接続

この節では、ブローカをクラスタに接続する 2 つの方法を説明します。どちらの方法を使用する場合でも、起動するブローカはすべて 5 秒間隔でほかのブローカへの接続を試みますが、クラスタ内のほかのブローカが起動されるまでは、接続できません。

ブローカをクラスタに接続する場合、先にマスターブローカを起動する必要はありません。マスターブローカより先にクラスタのブローカを起動した場合、クラスタのブローカはクライアントのコネクションを拒否する中断状態になります。マスターブローカを起動すると、自動的に中断状態だったブローカが完全に機能するようになります。

方法 1: クラスタ設定ファイルを使用しない方法

➤ ブローカをクラスタに接続するには

1. ブローカを起動する imqbrokerd コマンドの -cluster オプションを使用し、-cluster オプションに、接続先のブローカの完全なリストを引数として指定します。
2. クラスタに接続するブローカを起動するときに、各ブローカに対してこの手順を実行します。

   たとえば、次のコマンドは新しいブローカを起動して、host1 のデフォルトのポートで実行中のブローカ、host2 のポート 7677 で実行中のブローカ、およびローカルホストのポート 7678 で実行中のブローカに接続します。

```
imqbrokerd -cluster host1,host2:7677,:7678
```

方法 2: クラスタ設定ファイルを使用する方法

接続されるブローカのリストを指定するクラスタ設定ファイルを作成することも可能です。この場合、オプションでマスターブローカのアドレスも指定できます。クラスタを定義するこの方法は、運用システムに適しています。クラスタのブローカごとに、imq.cluster.url プロパティの値がクラスタ設定ファイルを指すように設定する必要があります。

クラスタへのブローカの追加

ブローカクラスタを設定すると、新しいブローカを追加するか、あるいはすでにクラスタの一部となっているブローカを再起動する必要があります。

新しいブローカを既存のクラスタに追加するには、次のいずれかの手順を実行します。

クラスタ設定ファイルを使用していない場合、新しいブローカを起動するときに、imq.cluster.brokerlist と、必要な場合は imq.cluster.masterbroker プロパティを -D オプションを使用してコマンド行で指定します。

➤ クラスタ設定ファイルを使用している場合に、ブローカをクラスタに追加するには

1. 新しいブローカを、クラスタ設定ファイルにある imq.cluster.brokerlist プロパティに追加します。
2. クラスタ内の任意のブローカに次のコマンドを実行します。

   ```
   imqcmd reload cls
   ```

   このコマンドを実行すると、すべてのブローカが imq.cluster.brokerlist プロパティを再読み込みし、クラスタ内のブローカの持続情報がすべて最新になります。

クラスタ内のブローカの再起動

すでにクラスタのメンバーであるブローカを再起動するには、次のどちらかの手順を実行します。

- クラスタ設定ファイルを使用してクラスタを定義する場合、-D オプションを使用して、ブローカの起動に使用されるコマンド行で `imq.cluster.url` プロパティを指定します。
- クラスタ設定ファイルを使用してクラスタを定義しない場合、新しいブローカを起動するときに、-D オプションを使用してコマンド行で `imq.cluster.brokerlist` を指定します。必要な場合は `imq.cluster.masterbroker` プロパティも指定します。クラスタにマスターブローカが含まれていない場合は、新しいブローカの起動時に、`-cluster` オプションを使用するだけで、クラスタにあるブローカのリストを指定できます。

クラスタからのブローカの削除

ブローカをクラスタから削除するときには、次の点に注意します。

- ブローカ A、B、および C が次のコマンド行を使用して起動された場合、A だけを再起動しても、ブローカはクラスタから削除されません。

  ```
  imqbrokerd -cluster A,B,C
  ```

  A だけを再起動するのではなく、次のコマンド行を使用してほかの 2 つのブローカを再起動する必要があります。

  ```
  imqbrokerd -cluster B,C
  ```

  次に、`-cluster` オプションを指定せずに、ブローカ A を起動する必要があります。
- クラスタ設定ファイルを使用してブローカのリストが指定されている場合、次の手順を実行する必要があります。
  - 設定ファイルからブローカの詳細を削除します。
  - 今後共通プロパティを使用しないように、削除するブローカの `imq.cluster.url` プロパティを変更または削除します。
  - `imqcmd reload cls` コマンドを使用して、ブローカすべてにクラスタ設定を再読み込みさせ、クラスタを設定し直します。

マスターブローカの設定変更レコードのバックアップ

クラスタごとに 1 つのマスターブローカがあり、クラスタの持続的な状態に対する変更がすべて追跡されます。たとえば、永続サブスクリプションの変更や管理者が作成した物理送信先の変更が記録されます。すべてのブローカは、これら持続オブジェクトに関する情報の同期を取るため、起動時にマスターブローカを参照します。その結果、マスターブローカに障害が発生すると、クラスタ全体が機能しなくなる場合があります。このため、`imqbrokerd` コマンドの `-backup` オプションを使用して、定期的にマスターブローカの変更レコードのバックアップを取っておくことが重要です。たとえば、次のように指定します。

```
imqbrokerd -backup mybackuplog
```

これを適宜行うことが重要です。きわめて古いバックアップを復元すると、情報が失われることがあります。最後のバックアップ以降に作成された持続オブジェクトはすべて失われます。

マスターブローカの設定変更レコードの復元

➤ 障害時にマスターブローカを復元するには

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て、マスターブローカの設定変更レコードを復元します。

   ```
   imqbrokerd  -restore mybackuplog
   ```

3. マスターブローカに新しい名前やポート番号を割り当てる場合、プロパティ `imq.cluster.masterbroker` を使用して、クラスタ設定ファイルを更新し、マスターブローカがクラスタの一部であること、およびマスターブローカの新しい名前を指定する必要があります。
4. すべてのブローカを再起動します。

ブローカを復元しても、必ずブローカの設定変更レコードに古いデータが再読み込みされます。ただし、前述したように、定期的なバックアップを頻繁に実行することで、古いデータの再読込みが最小限に抑えられます。

マスターブローカは、持続オブジェクトへの変更履歴全体を記録するため、一定期間を超えるとデータベースが非常に大きくなることがあります。バックアップや復元を実行することにより、大きくなったデータベースが圧縮されたり最適化されるという効果が得られます。

ログ作成

この節では、ブローカのデフォルトログ作成の設定を説明し、代替出力チャネルへのログ情報のリダイレクト、ロールオーバー条件の変更、およびブローカメトリックスの報告を行うために設定を変更する方法を解説します。ログ作成の説明については、64 ページの「ロガー」を参照してください。

デフォルトのログ作成の設定

ブローカの起動時に、ブローカは自動的に設定され、次の場所にあるローリングログファイルのセットにログの出力が保存されます。

```
IMQ_VARHOME/instances/BrokerName/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、プロパティ `imq.log.file.dirpath` 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。

ブローカは、`ERROR`、`WARNING`、および `INFO` の 3 つのログカテゴリをサポートしています (64 ページの表 2-7 を参照)。ログ作成レベルを設定すると、設定したレベルおよびそれよりも低いレベルのメッセージがすべて収集されます。デフォルトのログレベルは `INFO` です。この場合、`ERROR`、`WARNING`、および `INFO` レベルのメッセージが記録されることになります。

ログメッセージの書式設定

記録されたメッセージは、タイムスタンプ、メッセージコード、およびメッセージの本文で構成されています。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
```

ロガー設定の変更

66 ページの表 2-8 にすべてのロガープロパティを示します。

➤ ブローカのロガー設定を変更するには

1. ログレベルを設定します。
2. 1 つまたはそれ以上のログ作成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条件を設定します。

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 5-4 にログ作成に影響する imqbrokerd オプションを一覧表示します。

表 5-4　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 *number*	メトリックス情報が収集される間隔を、秒単位で指定する
-loglevel *level*	ログレベルを、ERROR、WARNING、INFO のいずれかに設定する
-silent	コンソールへのログ作成をオフにする

表 5-4　imqbrokerd ロガーオプションと対応するプロパティ (続き)

imqbrokerd オプション	説明
-tty	すべてのメッセージをコンソールに送信する。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る

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
- ロールオーバー条件の変更
- ブローカメトリックス情報の記録

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。

- 指定したすべてのレベルのログカテゴリが画面に表示するには、imqbrokerd コマンドの -tty オプションを使用する
- 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 オプションを使用する
- 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グ作成情報のカテゴリを指定する。たとえば、次のように指定する

  ```
  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グ作成情報のカテゴリを指定する。たとえば、次のように指定する

  ```
  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 に書き込むログ作成情報のカテゴリを指定する。たとえば、次のように指定する

  ```
  imq.log.syslog.output=NONE
  ```

注 ログメッセージの送信先を変更する前に、出力チャネルにマッピングしているログカテゴリに対応するレベルに、ログ作成が設定されていることを確認します。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、これらのレベルのメッセージのログ作成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。

ロールオーバー条件の変更

ログファイルのロールオーバーには、時間とサイズの 2 つの条件があります。デフォルトでは時間の条件が使用され、7 日ごとにファイルがロールオーバーされます。

- 間隔を変更するには、プロパティ imq.log.file.rolloversecs を変更する必要がある。たとえば、次のようにプロパティを定義すると、間隔が 10 日に変更される

  ```
  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条件を変更するには、imq.log.file.rolloverbytes プロパティを設定する必要がある。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る

  ```
  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。

ブローカパフォーマンスメトリックスのログ作成

ブローカのデフォルトの設定には、次の設定が含まれます。

- imq.metrics.enabled=true
- imq.metrics.interval=0
- imq.log.level=INFO

これらの設定により、ブローカはアクティブコネクションサービスおよびブローカのパフォーマンスメトリックスを収集しますが、メトリックスレポートは作成されません。

次の 2 つのどちらかの方法で、ブローカがメトリックスレポートを作成するように設定できます。

- imqbrokerd コマンドの -metrics オプションを使用し、ブローカがレポートを作成する間隔を秒単位で指定する

- ブローカにレポートを作成させる秒単位の間隔を imq.metrics.interval プロパティに設定する

メトリックスレポートは、INFO カテゴリに含まれるため、デフォルトではログファイル出力チャネルに書き込まれます。

次に、メトリックス情報の例を示します。

```
[31/Jan/2001:15:00:50 PST]
Connections:0 JVM Heap:6291456 bytes (5186320 free)
      In: 0 mesgs (0bytes) 0 pkts (0 bytes)
     Out: 0 mesgs (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)
```

表 5-5 にコネクションサービスごとに作成されたメトリックスの説明を示します。

表 5-5 コネクションサービスに作成されたメトリックス

メトリックス	説明
Pkts in (total)	最後にリセットされてからブローカが読み込んだパケット数の合計。パケット数の合計には、JMS メッセージだけでなく MQ プロトコルパケットも含まれる
Pkts out (total)	最後にリセットされてからブローカが書き込んだパケット数の合計
JMS Messages in (total)	最後にリセットされてからブローカが読み込んだ JMS メッセージ数の合計
JMS Messages out (total)	最後にリセットされてからブローカが書き込んだ JMS メッセージ数の合計
Message Bytes in (total)	最後にリセットされてからブローカが読み込んだメッセージバイト数の合計
Message Bytes out (total)	書き込まれたメッセージバイト数の合計
Current # connections	現在開かれているコネクション数

表 5-6 にブローカごとに収集および報告されたメトリックスの説明を示します。

表 5-6　各ブローカに収集されたメトリックス

メトリックス	説明
VM heap size (bytes)	Java VM ヒープの最大サイズ
VM heap free space (bytes)	Java VM ヒープに残っている空き容量

注　この情報は `imqcmd metrics` コマンドからでも使用できます。

第 6 章

ブローカとアプリケーションの管理

この章では、ブローカとブローカが提供するサービスの管理に関するタスクを実行する方法について説明します。これらのタスクの中には、特定のクライアントアプリケーションに依存していないものがあります。これには、次のようなものが含まれます。

- ブローカの状態の制御。ブローカを停止、再開、シャットダウン、および再起動することができる
- ブローカのプロパティのクエリおよび更新
- コネクションサービスのクエリおよび更新
- リソースの割り当ておよび管理
- コネクションサービスの管理

その他のブローカのタスクは、特定のアプリケーションのために実行されます。これらのタスクには、物理的な送信先、永続サブスクリプション、トランザクションなどの管理が含まれます。

- MQ のメッセージは、ブローカの送信先を経由して、受信側またはサブスクライバにルートされる。管理者は、ブローカ上でこれらの送信先を作成する必要がある
- 永続サブスクリプションを保持するクライアントが、アクティブになっていない場合でも、MQ によって、永続サブスクライバのリソースの割り当ておよび管理が行われる。MQ のリソースを節約するために、MQ のコマンドツールを使用して、永続サブスクリプションに関する情報を入手し、永続サブスクリプションを破棄、あるいはそれらのメッセージをパージすることができる
- MQ のトランザクションおよび分散トランザクションは、ブローカによって記録される。障害が発生した場合、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る

この章では、コマンドユーティリティ (imqcmd) を使用して、これらのすべてのタスクを実行する方法について説明します。MQ メッセージサーバの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て、これらと同じタスクの多くを実行することができます。詳細は、第 4 章「管理コンソールのチュートリアル」を参照してください。

コマンドユーティリティ

コマンドユーティリティを使用すると、ブローカおよびブローカが提供するサービスを管理することができます。この節では、imqcmd コマンドの基本構文、サブコマンドのリスト、imqcmd オプションの概要について説明します。後続の節では、これらの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。

コマンドの構文

imqcmd コマンドの一般的な構文は、次のとおりです。

```
imqcmd [subcommand argument ] options
```

または

```
imqcmd subcommand argument [options ]
```

-v、-h、および -H オプションを指定する場合、コマンド行で指定するその他のサブコマンドは実行できません。たとえば、次のコマンドを入力すると、バージョン情報が表示されますが、restart サブコマンドは実行されません。

```
imqcmd restart bkr -v
```

imqcmd のサブコマンド

コマンドユーティリティ (imqcmd) には、次の表 6-1 に示すようなサブコマンドが含まれています。

表 6-1　imqcmd のサブコマンドと引数

サブコマンド	説明
create dst	送信先を作成する
commit txn	トランザクションをコミットする
destroy dst	送信先を破棄する

表 6-1　imqcmd のサブコマンドと引数 (続き)

サブコマンド	説明
destroy dur	永続サブスクリプションを破棄する
list dst	ブローカの送信先を一覧表示する
list dur	トピックの永続サブスクリプションを一覧表示する
list svc	ブローカのサービスを一覧表示する
list txn	ブローカのトランザクションを一覧表示する
metrics bkr	ブローカのメトリックスを表示する
metrics svc	サービスのメトリックスを表示する
pause bkr	ブローカのすべてのサービスを停止する
pause svc	1 つのサービスを停止する
purge dst	送信先を破棄しないで、送信先のすべてのメッセージをパージする
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する
query bkr	ブローカの情報をクエリおよび表示する
query dst	送信先の情報をクエリおよび表示する
query svc	サービスの情報をクエリおよび表示する
query txn	トランザクションの情報をクエリおよび表示する
reload cls	ブローカクラスタ設定の再読み込みを行う
restart bkr	ブローカを再起動する
resume bkr	ブローカのすべてのサービスを再開する
resume svc	1 つのサービスを再開する
rollback txn	トランザクションをロールバックする
shutdown bkr	ブローカをシャットダウンする
update bkr	ブローカの属性を更新する
update dst	送信先の属性を更新する
update svc	サービスの属性を更新する

imqcmd のオプションの概要

表 6-2 に、imqcmd コマンドのオプションを一覧表示します。これらの使用方法については、後続のタスクごとの節を参照してください。

表 6-2　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する。デフォルト値は localhost:7676 ポートだけを指定する場合は -b :7878 ホスト名だけを指定する場合は -b somehost
-c *"clientID"*	トピックの永続サブスクライバの ID を指定する。詳細は、158 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 *topicName*	トピック名を指定する。list dur サブコマンドや destroy dur サブコマンドと一緒に使用する。158 ページの「永続サブスクリプションの管理」を参照
-f	ユーザの確認なしで、アクションを実行する
-h	使用方法に関するヘルプを表示する
-H	使用方法に関するヘルプ、属性リスト、および例を表示する
-int *interval*	imqcmd がブローカのメトリックスを表示する間隔を秒単位で指定する (metrics サブコマンドと一緒に使用する)
-javahome	使用する代替の Java 2 互換のランタイムを指定する (デフォルトでは、製品にバンドルされているランタイムが使用される)
-m *metricType*	表示するメトリックス情報のタイプを指定する。タイプには、次のどれかを指定する ttl　ブローカの送受信メッセージの総数 (デフォルト) rts　ttl と同じ情報を提供するが、1 秒あたりのメッセージ数を指定する cxn　コネクション、仮想メモリヒープ、スレッド このオプションは、metrics bkr サブコマンド、または metrics svc サブコマンドと一緒に使用する。次のコマンドでは、デフォルトブローカの cxn タイプのメトリックスが 5 秒ごとに表示される imqcmd metrics bkr -m cxn -int 5

表 6-2　imqcmd のオプション (続き)

オプション	説明
-n *targetName*	ターゲット名を指定する。これはサブコマンドに応じて、物理的な送信先、サービス、トランザクション ID、または永続サブスクリプションの名前になる。送信先の管理での使用方法については、154 ページの「送信先の管理」を参照
-o *attribute=value*	属性の値を指定する。これはサブコマンドの引数に応じて、ブローカの属性 (146 ページの「ブローカのプロパティのクエリと更新」を参照)、サービス (150 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 (154 ページの「送信先の管理」を参照) になる
-secure	ssladmin コネクションサービス (「手順 4. SSL ベースのクライアントを設定および実行する」、203 ページの「コマンドユーティリティ (imqcmd)」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する
-p *password*	管理者パスワードを指定する。この値を省略すると、管理者名の入力を要求される
-s	サイレントモード。出力が表示されない
-t *destinationType*	t (トピック) または q (キュー) のどちらかの送信先のタイプを指定する
-tmp	一時送信先を表示する
-u *name*	管理者名を指定する。この値を省略すると、管理者名の入力を要求される
-v	バージョン情報を表示する

imqcmd のサブコマンドを実行するたびに、ホスト名とポート番号 (-b)、ユーザ名 (-u)、パスワード (-p)、および安全なコネクション (-secure) のオプションを指定する必要があります。ホスト名とポート番号を指定しない場合、デフォルト値が使用されます。ユーザ名とパスワードの情報を指定しないと、これらの情報の入力を要求されます。-secure オプションを指定しない場合は、コネクションがセキュリティ保護されません。

imqcmd を使用するための前提条件

imqcmd コマンドを使用してブローカを管理する場合、次のタスクを実行する必要があります。

- imqbrokerd コマンドを使用して、ブローカを起動する

  コマンドユーティリティは、すでに起動しているブローカを管理する場合にのみ使用できる。コマンドユーティリティでは、ブローカを起動することはできない。122 ページの「ブローカの起動」を参照

- ブローカをポート 7676 のローカルホストで実行しない場合、-b オプションを使用して、ターゲットブローカを指定する

- 適切な管理者名とパスワードを指定する。この値を指定しないと、値の入力を要求される。imqcmd を使用する操作はすべてユーザリポジトリに照らし合わされて認証される

  MQ 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リポジトリがインストールされます。このファイルには、IMQ_HOME/etc/passwd (Solaris では /etc/imq/passwd) という名前が付けられています。リポジトリには、管理ユーザ用とゲストユーザ用の 2 つのエントリがデフォルトで用意されています。これらのエントリを使用すれば、追加作業を行わずにブローカに接続することができます。たとえば、MQ をテストするだけなら、デフォルトのユーザ名とパスワード (admin/admin) を使用して、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、ユーザを認証および認可するための追加作業を行う必要があります。また、ユーザリポジトリについて、既存の LDAP ディレクトリサーバを使用するオプションも用意されています。詳細は、184 ページの「ユーザの認証」を参照してください。

例

次のコマンドでは、ポート 7676 の localhost で実行しているブローカのプロパティが一覧表示されます。

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

次のコマンドでは、ユーザ名が alladin で、ユーザパスワードが abracadabra のポート 1564 の myserver で実行しているブローカのプロパティが一覧表示されます。

```
imqcmd query bkr -b myserver:1564 -u alladin -p abracadabra
```

ユーザ名の alladin が admin グループに割り当てられている場合、指定したブローカに、admin クライアントとして接続されます。

ブローカの状態の制御

ブローカの起動後に、次の 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。

- ブローカの停止

  ブローカを停止すると、ブローカのサービススレッドが中断されるため、ブローカはポートでの待機をやめます。その後で、ブローカへのメッセージフローを調整するのに必要な管理タスクを実行できます。たとえば、特定の送信先にメッセージが集中している場合、ブローカを停止して、メッセージソースの追跡、送信先のサイズの制限、送信先の破棄など、問題を解決するためのアクションを実行できます。

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

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

- ブローカの再開

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

  ```
  imqcmd resume bkr
  ```

- ブローカのシャットダウン

  ブローカをシャットダウンすると、ブローカプロセスが終了します。これは正規の終了であるため、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。次のコマンドでは、ポート 1572 の ctrlsrv 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
  ```

- ブローカの再起動

  ブローカをシャットダウンして、再起動します。次のコマンドでは、ポート 7676 の localhost で実行していたブローカが再起動されます。

  ```
  imqcmd restart bkr
  ```

表 6-3 は、ブローカを制御するのに使用する imqcmd のサブコマンドについてまとめたものです。ポート 7676 の localhost で実行するブローカを対象にしない場合は、ブローカのホスト名とポート番号を指定する必要があります。

表 6-3　ブローカを制御する imqcmd のサブコマンド

サブコマンド	説明
`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する
`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する
`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する
`restart 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て、再起動する このコマンドは、ブローカを最初に起動したときに指定したオプションを使用して、ブローカを再起動する。別のオプションを有効にする必要がある場合、ブローカをシャットダウンしてから、必要なオプションを指定して、ブローカを起動し直す必要がある

ブローカのプロパティのクエリと更新

Command ユーティリティには、ブローカに関する情報を入手したり、ブローカのプロパティを更新したりすることができるサブコマンドが含まれています。表 6-4 にこれらのサブコマンドを一覧表示します。

表 6-4　情報の入手およびブローカの更新を行う imqcmd のサブコマンド

サブコマンドの構文	説明
`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する。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る
`reload cls`	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する。詳細は、129 ページの「クラスタへのブローカの追加」を参照

表 6-4　情報の入手およびブローカの更新を行う imqcmd のサブコマンド

サブコマンドの構文	説明
`update bkr [-b` *hostName:port*`] -o` *attr=val*	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する
`metrics bkr [-b` *hostName:port*`] [-m metricType] [-int interval]`	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する 表示するメトリックスのタイプを指定するには、-m オプションを使用する `ttl` ブローカの送受信メッセージの総数 (デフォルト) `rts` ttl と同じ情報を提供するが、1 秒あたりのメッセージ数を指定する `cxn` コネクション、仮想メモリヒープ、スレッド 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 オプションを使用する。デフォルトは 5 秒

ポート 7676 の `localhost` で実行するブローカを対象にしない場合、表 6-4 に示されるサブコマンドを使用するときに、ブローカのホスト名とポート番号を指定する必要があります。

ブローカのクエリ

シングルブローカに関する情報のクエリと表示を行うには、query サブコマンドを使用します。たとえば、次のように指定します。

```
imqcmd query bkr
```

このサブコマンドでは、次のような情報が出力されます。

```
%imqcmd query bkr

Querying the broker specified by:

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

Auto Create Queues                      true
Auto Create Topics                      true
Auto Created Queue Delivery Policy      Round Robin
Cluster Broker List (active)            myhost/192.18.116.221:7676
Cluster Broker List (configured)
Cluster Master Broker
Cluster URL
Current Number of Messages in System    0
Current Size of Messages in System      0
Instance Name                           imqbroker
Log Level                               INFO
Log Rollover Interval (seconds)         604800
Log Rollover Size (bytes)               0 (unlimited)
Max Message Size                        70m
Max Number of Messages in System        0 (unlimited)
Max Size of Messages in System          0 (unlimited)
Primary Port                            7676
Version                                 3.0

Successfully queried the broker.
```

ブローカの更新

表 6-5 に示すブローカのプロパティを更新するには、update サブコマンドを使用します。ブローカの更新情報は、ブローカのインスタンス設定ファイルに自動的に書き込まれます。

表 6-5　ブローカのプロパティ

プロパティ	説明
imq.autocreate.queue	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。デフォルトは True

表 6-5　ブローカのプロパティ (続き)

プロパティ	説明
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。デフォルトは True
imq.queue.deliverypolicy	自動作成されたキューのデフォルトの配信ポリシーを指定する。指定できる値は s (シングル)、r (ラウンドロビン)、または f (フェイルオーバー)。デフォルトは s
imq.cluster.url	クラスタ設定ファイルの場所を指定する。詳細は、126 ページの「クラスタ設定プロパティ」を参照
imq.log.level	ログレベルを指定する。指定できる値は NONE、ERROR、WARNING、INFO 。デフォルトは INFO
imq.log.file.rolloversecs	ログファイルがロールオーバーされるまでの有効期間 (秒単位)。値を 0 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行われない。デフォルトは 604800 (7 日間)
imq.log.file.rolloverbytes	ログファイルがロールオーバーされるまでのファイルの最大サイズを指定する。値を 0 に設定した場合、ファイルのサイズに基づいたロールオーバーは行われない。デフォルトは 0
imq.message.max_size	1 つのメッセージの最大サイズをバイト単位で指定する。デフォルトは 70m
imq.system.max_count	メモリおよびディスク内のメッセージの最大数を指定する。値を 0 に設定した場合、無制限になる。デフォルトは 0
imq.system.max_size	メモリおよびディスク内のメッセージの最大合計サイズを指定する。値を 0 に設定した場合、無制限になる。デフォルトは 0
imq.portmapper.port	ポートマッパーポートの数を指定する。デフォルトは 7676

たとえば、次のコマンドでは、キューのデフォルト配信ポリシーがシングルからラウンドロビンに変更されます。

```
imqcmd update bkr -o "imq.queue.deliverypolicy=r"
```

コネクションサービスの管理

Command ユーティリティには、次のタスクを実行できるさまざまなサブコマンドが含まれています。

- 使用可能なコネクションサービスを一覧表示する
- 特定のサービスに関する情報を表示する
- サービスの属性を更新する
- サービスを停止および再開する

MQ のコネクションサービスの概要については、48 ページの「コネクションサービス」を参照してください。

表 6-6 に、コネクションサービスを制御する `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 6-6 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する
`metrics svc -n` *serviceName* [`-b` *hostName:port*] [`-m` *metricType*] [`-int` *interval*]	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一覧表示する 表示するメトリックスのタイプを指定するには、-m オプションを使用する `ttl` ブローカの送受信メッセージの総数 (デフォルト) `rts` ttl と同じ情報を提供するが、1 秒あたりのメッセージ数を指定する `cxn` コネクション、仮想メモリヒープ、スレッド 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 オプションを使用する。デフォルトは 5 秒
`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する

表 6-6　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
`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する。admin サービスは停止できない
`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する
`update svc -n` *serviceName* [`-b` *hostName:port*] `-o` *attr=val*	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する。サービスの属性については、153 ページの表 6-8 を参照

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の通信をサポートしています。MQ ブローカで現在使用できるサービスを、151 ページの表 6-7 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 (NORMAL または ADMIN) と基礎となるトランスポートレイヤによって指定されます。

表 6-7　ブローカでサポートされてい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 (JMS メッセージ配信)	tcp
ssljms	NORMAL (JMS メッセージ配信)	tls (SSL ベースセキュリティ)
httpjms	NORMAL (JMS メッセージ配信)	http
httpsjms	NORMAL (JMS メッセージ配信)	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)

コネクションサービスの一覧表示

ブローカの使用可能なコネクションサービスを一覧表示するには、次のようなコマンドを使用します。

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

たとえば、次のコマンドでは、ポート 6565 の MyServer ホスト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
```

次のコマンドでは、ポート 7676 の `localhost` 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
```

このコマンドでは、次のような情報が出力されます。

```
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          dynamic            UNKNOWN
ssljms            dynamic            UNKNOWN

Successfully listed services.
```

サービスのプロパティのクエリと更新

シングルサービスに関する情報のクエリと表示を行うには、query サブコマンドを使用します。たとえば、次のように指定します。

```
imqcmd query svc -n jms
```

このサブコマンドでは、次のような情報が出力されます。

```
Querying the service where:

Service Name
jms

On the broker specified by:

Host                          Primary Port
localhost                     7676


Current Number of Allocated Threads    120
Current Number of Connections          20
Max Number of Threads                  1000
Min Number of Threads                  50
Port Number                            42019 (dynamic)
Service Name                           jms
Service State                          RUNNING

Successfully queried the service.
```

表 6-8 に示す 1 つ以上のサービスの属性値を変更するには、update サブコマンドを使用します。

表 6-8 コネクションサービスの属性

属性	説明
port	更新するサービスに割り当てられるポート (httpjms または httpsjms には適用しない)
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"
```

サービスの停止および再開

admin サービス (停止できない) 以外のサービスを停止するには、次のようなコマンドを使用します。

```
imqcmd pause svc -n serviceName
```

サービスを再開するには、次のようなコマンドを使用します。

```
imqcmd resume svc -n serviceName
```

送信先の管理

MQ のメッセージはすべて、特定のブローカで作成された送信先、キュー、およびトピックを経由して、コンシューマクライアントにルートされます。管理者は、ブローカのこれらの送信先を管理する必要があります。送信先の管理には、コマンドユーティリティを使用して送信先の作成と破棄、送信先の一覧表示、送信先に関する情報の表示、およびメッセージのパージを行うことが含まれます。送信先の概要については、67 ページの「物理的な送信先」を参照してください。

表 6-9 は、imqcmd の送信先のサブコマンドについてまとめたものです。デフォルトのブローカ (localhost:7676) を使用しない場合、ブローカのホスト名とポート番号を指定する必要があります。

表 6-9 送信先を管理する imqcmd のサブコマンド

サブコマンド	説明
`list dst [-tmp]`	すべての送信先を一覧表示する。同様に、一時送信先についても一覧表示するオプションがある (70 ページの「一時送信先」を参照)
`create dst -t type -n name [-o att=val] [-o att=val1]...]`	特定のタイプの送信先を指定した名前と属性で作成する。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る。送信先名は、英字や「 _ 」で開始できる
`destroy dst -t type -n name`	特定のタイプと名前の送信先を破棄する
`purge dst -t type -n name`	特定のタイプと名前の送信先のメッセージをパージする
`query dst -t type -n name`	特定のタイプと名前の送信先に関する情報を一覧表示する

表 6-9　(続き) 送信先を管理する imqcmd のサブコマンド (続き)

サブコマンド	説明
update dst -t *type* -n *name* -o *att=val* [[-o *att=val1*] . . .]	特定の送信先で特定の属性値を更新する 属性名は、表 6-10 に示す属性のどれかになる

送信先の作成

送信先を作成する場合、タイプ (トピックまたはキュー) と送信先の属性値 (必要に応じて) を指定する必要があります。これらの属性のデフォルト値は、ブローカの設定ファイル (115 ページの「設定ファイル」を参照) に設定されます。

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。

表 6-10 に送信先を作成する際、各タイプの送信先に設定できる属性を示します。

表 6-10　送信先の属性

送信先タイプ	属性	デフォルト値	説明
キュー	queueDeliveryPolicy	Single	メッセージを転送するのに使用するアルゴリズムを指定する 値は、次のどれかになる f = フェイルオーバー r = ラウンドロビン s = シングル
キュー	maxTotalMsgBytes	0 (無制限)	キューで許可されるメッセージの最大合計サイズ (バイト単位)
キュー	maxNumMsgs	0 (無制限)	キューで許可されるメッセージの最大数
キュー	maxBytesPerMsg	0 (無制限)	キューで許可されるシングルメッセージの最大サイズ
トピック	maxBytesPerMsg	0 (無制限)	トピックに送信されるシングルメッセージの最大サイズ

- 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する

```
imqcmd create dst -n myQueue -t q -o "queueDeliveryPolicy=f"
```

送信先名は、有効な Java 識別子にする必要がある

- 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する

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

送信先に関する情報の入手

送信先の現在の属性値に関する情報を入手するには、次のようなコマンドを使用します。

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

さらに、update imqcmd サブコマンドを使用して、1 つ以上の属性の値を変更することができます。

ポート 4545 の myHost などのように、特定のブローカの送信先をすべて一覧表示するには、次のようなコマンドを使用します。

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

list コマンドでは、オプションで一時送信先 (-tmp オプションを使用) を指定することが可能です。一時送信先とは、ほかのクライアンに送信したメッセージに対する応答を受け取るための送信先が必要なクライアントアプリケーションによって作成される送信先のことです (70 ページの「一時送信先」を参照)。これらの送信先は、アクティブなメッセージコンシューマが存在しない場合に、クライアントアプリケーションが作成する API コールによってのみ破棄されます。

送信先の更新

送信先の属性を変更するには、update dst サブコマンドと -o オプションを使用して、更新する属性を指定します。複数の属性を更新する必要がある場合、-o オプションを使用できます。たとえば、次のコマンドでは maxBytesPerMsg 属性が 1000 に、MaxNumMsgs 属性が 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
```

更新が可能な属性については、155 ページの表 6-10 を参照してください。

送信先のタイプやキューのキュー配信ポリシーを更新する場合、update dst サブコマンドは使用できません。

送信先のパージ

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。送信先をパージすると、物理的な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。送信先に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。

送信先のメッセージをパージするには、次のようなコマンドを使用します。

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

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

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、`imqbrokerd` コマンドの `reset` サブコマンドを使用して、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
```

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。

送信先の破棄

送信先を破棄するには、次のようなコマンドを使用します。

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

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。

表 6-12 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。デフォルトのブローカ (localhost:7676) を使用しない場合、ブローカのホスト名とポート番号を指定する必要があります。

表 6-11 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` *destination*	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する
`destroy dur -n` *name* `-c` *client_id*	特定のクライアント識別子の特定の永続サブスクリプションを破棄する (39 ページの「クライアント識別子」を参照)
`purge dur -n` *name* `-c` *client_id*	特定のクライアント識別子のメッセージをすべてパージする (39 ページの「クライアント識別子」を参照)

たとえば、次のコマンドでは、トピック SPQuotes の永続サブスクリプションがすべて一覧表示されます。

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

トピックの永続サブスクリプションごとに、list サブコマンドは、永続サブスクリプションの名前、ユーザのクライアント 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 / 非アクティブ) を返します。次に例を示します。

```
Listing all the durable subscriptions on the topic myTopic
on the broker specified by:
-------------------------
Host          Primary Port
-------------------------
localhost    7676

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

Successfully listed durable subscriptions.
```

list 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。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
```

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、XA リソースマネージャによって管理される MQ 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 (41 ページの「ローカルトランザクション」を参照)。トランザクションにはすべて、MQ 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。MQ は、MQ 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。

表 6-12 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。デフォルトのブローカ (localhost:7676) を使用しない場合、ブローカのホスト名とポート番号を指定する必要があります。

表 6-12 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る
`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る
`commit txn -n` *transaction_id*	特定のトランザクションをコミットする
`rollback txn -n` *transaction_id*	特定のトランザクションをロールバックする

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。次に例を示します。

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

-----------------------------------------------------------------
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

Successfully listed transactions.
```

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 (55 ページの表 2-4 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プロデュース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、list サブコマンドで表示される情報のほかに、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) などの追加情報が表示されます。たとえば、次のように指定します。

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

このサブコマンドによって、次のような情報が出力されます。

```
Querying the transaction where:
-------------------------
Transaction ID
-------------------------
64248349708800

On the broker specified by:

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


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

Successfully queried the transaction.
```

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。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。詳細は、55 ページの表 2-4 の imq.transaction.autorollback プロパティを参照してください。

第 7 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とは、プロバイダ固有の設定および命名情報をカプセル化するオブジェクトのことです。通常、これらのオブジェクトは MQ の管理者によって作成され、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する際に使用されます。さらに、これらのオブジェクトは、物理的な送信先にメッセージを送信したり、物理的な送信先からメッセージを受信したりする場合にも使用されます。

管理対象オブジェクトの概要については、78 ページの「MQ 管理対象オブジェクト」を参照してください。

MQ には、管理対象オブジェクトを作成および管理するために、オブジェクトマネージャコマンド行ユーティリティ (`imqobjmgr`) と GUI の管理コンソールの 2 つの管理ツールが用意されています。これらのツールでは、次のようなタスクを実行できます。

- 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

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。管理コンソールについては、第 4 章「管理コンソールのチュートリアル」を参照してください。

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 : LDAP サーバは、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 サーバには、運用環境で有効なセキュリティ機能も用意されています。MQ の管理ツールは、LDAP サーバでの使用を考慮して設計されています。

ファイルシステムストア : MQ 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。このディレクトリにアクセス可能なユーザは、MQ 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。

管理対象オブジェクト

管理対象オブジェクトの概要については、78 ページの「MQ 管理対象オブジェクト」を参照してください。

MQ の管理対象オブジェクトには、コネクションファクトリと送信先の 基本的な 2 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は、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します。送信先管理対象オブジェクトは、送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します。その送信先にプロデューサはメッセージを送信し、その送信先からコンシューマはメッセージを受信します。特殊な SOAP 終端管理対象オブジェクトは、SOAP メッセージングに使用されます。詳細は、『MQ 開発者ガイド』を参照してください。

メッセージ配信モデル (ポイントツーポイント、またはパブリッシュ / サブスクライブ) に応じて、特定タイプのコネクションファクトリおよび送信先を使用できます。たとえば、ポイントツーポイントプログラミングの場合、queueConnectionFactory とキューの送信先を使用できます。同様に、パブリッシュおよびサブスクライブプログ

ラミングの場合、topicConnectionFactory とトピックの送信先を使用できます。また、分散トランザクションをサポートするコネクションファクトリのタイプとして、不特定のコネクションファクトリと送信先管理対象オブジェクトのタイプも使用できます (すべてのサポートタイプについては、38 ページの表 1-1 を参照)。

オブジェクトマネージャユーティリティ (imqobjmgr)

オブジェクトマネージャユーティリティを使用すると、MQ 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。この節では、imqobjmgr コマンドの基本構文、サブコマンドのリスト、imqobjmgr コマンドのオプションの概要について説明します。後続の節では、これらの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。

コマンドの構文

imqcmd コマンドの一般的な構文は、次のとおりです。

```
imqobjmgr subcommand [options ]
```

-v、-h、および -H オプションを指定する場合、そのコマンド行で指定するその他のサブコマンドは実行できません。たとえば、次のコマンドを入力すると、バージョン情報が表示されますが、list サブコマンドは実行されません。

```
imqcmd list -v
```

imqobjmgr のサブコマンド

オブジェクトマネージャユーティリティ (imqobjmgr) には、次の表 7-1 に示すようなサブコマンドが含まれています。

表 7-1 imqobjmgr のサブコマンド

サブコマンド	説明
add	管理対象オブジェクトをオブジェクトストアに追加する
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する
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する
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する

表 7-1　imqobjmgr のサブコマンド (続き)

サブコマンド	説明
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する

imqobjmgr コマンドのオプションの概要

表 7-2 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 7-2　imqobjmgr のオプション

オプション	説明
-f	ユーザの確認なしで、アクションを実行する
-h	使用方法に関するヘルプを表示する
-H	使用方法に関するヘルプ、属性リスト、および例を表示する
-i *fileName*	実行するコマンドとそのコマンドに必要なその他の情報 (検索名、オブジェクトストアの属性、オブジェクトタイプ、およびオブジェクト属性) が入っている java プロパティファイルの名前を指定する
-j *attribute=value*	JNDI オブジェクトストアの属性を指定する
-javahome	使用する代替の Java 2 互換のランタイムを指定する (デフォルトでは、製品にバンドルされているランタイムが使用される)
-l *name*	管理対象オブジェクトの JNDI 検索名を指定する。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
-o *attribute=value*	管理対象ブジェクトの属性を指定する
-r *read-only*	管理対象オブジェクトを読み取り専用オブジェクトとして作成するかどうかを指定する。値を true にすると、読み取り専用オブジェクトとして管理対象オブジェクトが作成される クライアントアプリケーションは、読み取り専用になっている管理対象オブジェクトの属性を変更することはできない デフォルトでは、false が設定される

表 7-2　imqobjmgr のオプション (続き)

オプション	説明
-t *type*	MQ の管理対象オブジェクトのタイプを指定する q = queue t = topic cf = ConnectionFactory qf = queueConnectionFactory tf = topicConnectionFactory xcf = XA ConnectionFactory (分散トランザクション) xqf = XA queueConnectionFactory (分散トランザクション) xtf = XA topicConnectionFactory (分散トランザクション) e = SOAP 終端*
-v	バージョン情報を表示する

* この管理対象オブジェクトのタイプは、SOAP メッセージをサポートするために使用されます (『MQ 開発者ガイド』を参照)。

次に、imqobjmgr のサブコマンドを使用する場合に、設定する必要がある情報について説明します。

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、管理者は imqobjmgr 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のタイプ

  使用可能なタイプを表 7-2 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

  - キューおよびトピックの場合 : ブローカの物理的な送信先の名前。これは imqcmd create サブコマンドの -n オプションで指定した名前になります。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 : クライアントが接続するブローカのホスト名とポート番号。この情報を指定しない場合、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。168 ページの「管理対象オブジェクトの属性」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、168 ページの「管理対象オブジェクトの属性」を参照してください。

- JNDI 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
- 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」
- オブジェクトストアへのアクセスに必要なユーザ名、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、171 ページの「オブジェクトストアの属性」を参照してください。

管理対象オブジェクトの属性

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。次に、これらの属性について説明します。

コネクションファクトリ管理対象オブジェクト

コネクションファクトリ (および XA コネクションファクトリ) の管理対象オブジェクトには、表 7-3 に示すような属性があります。主に使用する 2 つの属性は、`imqBrokerHostPort` と `imqBrokerHostName` です。これらの属性を使用して、クライアントアプリケーションがコネクションを確立するブローカを指定します。175 ページの「コネクションファクトリ管理対象オブジェクトの追加」では、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。

コネクションファクトリ属性の詳細およびこれらの属性の使用方法については、『MQ 開発者ガイド』と MQ クラスの `com.sun.messaging.ConnectionConfiguration` に関する『JavaDoc API Documentation』を参照してください。

表 7-3　コネクションファクトリ属性

属性 / プロパティ名	データ型	デフォルト値
imqAckOnAcknowledge	文字列	未設定
imqAckOnProduce	文字列	未設定
imqAckTimeout	文字列	0 ミリ秒
imqBrokerHostName	文字列	localhost
imqBrokerHostPort	文字列	7676
imqBrokerServicePort	文字列	0
imqConfiguredClientID	文字列	未設定
imqConnectionType	文字列	TCP
imqConnectionURL	文字列	http://localhost/imq/tunnel
imqDefaultPassword	文字列	guest
imqDefaultUsername	文字列	guest
imqDisableSetClientID	文字列	false
imqFlowControlCount	文字列	100
imqFlowControlIsLimited	文字列	false
imqFlowControlLimit	文字列	1000
imqLoadMaxToServerSession	文字列	false
imqQueueBrowserMax MessagesPerRetrieve	文字列	1000
imqQueueBrowserRetrieve Timeout	文字列	60,000 ミリ秒
imqReconnect	ブール	false
imqReconnectDelay	文字列	30,000 ミリ秒
imqReconnectRetries	文字列	0
imqSetJMSXAppID	文字列	false
imqSetJMSXConsumerTXID	文字列	false
imqSetJMSXProducerTXID	文字列	false
imqSetJMSXRcvTimestamp	文字列	false
imqSetJMSXUserID	文字列	false
imqSSLIsHostTrusted	文字列	true

表 7-3　コネクションファクトリ属性 (続き)

属性 / プロパティ名	データ型	デフォルト値
imqJMSDeliveryMode	整数	2 （持続）
imqJMSExpiration	整数	0 (有効期限なし)
imqJMSPriority	整数	4 (通常の優先度)
imqOverrideJMSDeliveryMode	ブール	false
imqOverrideJMSExpiration	ブール	false
imqOverrideJMSPriority	ブール	false
imqOverrideJMSHeadersTo TemporaryDestinations	ブール	false

送信先管理対象オブジェクト

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、表 7-4 に示すような属性があります。176 ページの「トピックまたはキューの管理対象オブジェクトの追加」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な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、MQ クラスの com.sun.messaging.DestinationConfiguration に関する『JavaDoc API Documentation』を参照してください。

表 7-4　送信先の属性

属性 / プロパティ名	データ型	デフォルト値
imqDestinationDescription	文字列	送信先オブジェクトの説明
imqDestinationName	文字列*	Untitled_Destination_Object

* 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「$」、あるいはその両方の文字で開始する必要があります。

オブジェクトストアの属性

オブジェクトストアの属性は、-j オプションを使用して指定し、属性と値の組み合わせで構成します。一般的に、次の属性を指定する必要があります。

初期コンテキストおよび場所の情報

これらのエントリの形式は、ファイルシステムストアと LDAP サーバのどちらを使用するかによって異なります。

ファイルシステムストア : ファイルシステムストアを使用する例として、C ドライブに *MyObjstore* という名前のフォルダを作成し、初期コンテキストと場所の属性について、それぞれ次の値を指定してください。

```
-j "java.naming.factory.initial=
          com.sun.jndi.fscontext.RefFSContextFactory"
-j "java.naming.provider.url=file:///C:/MyObjStore"
```

LDAP サーバ : LDAP サーバを使用する例として、初期コンテキストと場所の属性について、それぞれ次の値を指定してください。

```
-j "java.naming.factory.initial=
          com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
```

セキュリティ情報 (LDAP のみ)

これらのエントリの形式は、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も参照する必要があります。

セキュリティ属性は、次のようになります。

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

表 7-5 にこれらのエントリの説明を示します。

表 7-5　オブジェクトストアのセキュリティ属性

属性	説明
...principal	サービスの呼び出し元を認証するためのプリンシパルの識別情報。このエントリの形式は、認証スキーマによって異なる。このプロパティを指定しない場合は、サービスプロバイダによって動作が決定される
...credentials	サービスの呼び出し元を認証するためのプリンシパルの証明書。プロパティの値は、認証スキーマによって異なる。たとえば、ハッシュ値のパスワード、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る。このプロパティを指定しない場合は、サービスプロバイダによって動作が決定される
...authentication	使用するセキュリティのレベル。値は none、simple、strong のどれかになる。このプロパティを指定しない場合は、サービスプロバイダによって動作が決定される simple を指定した場合、imqobjmgr によって、値がまだ指定されていないプリンシパルまたは証明書の値を入力するよう要求される。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る

入力ファイルの使用

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、入力ファイルの名前を指定できます。

Object Manager ユーティリティ (imqobjmgr) と一緒に入力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、入力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 入力ファイルの一般的な構文は、次のとおりです。なお、version プロパティは、入力ファイルにのみ適用されます。これはコマンド行オプションではありません。この値には 2.0 を設定する必要があります。

```
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
...
```

入力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 "imqBrokerHostName=foo"
       -o "imqBrokerHostPort=777"
       -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.imqBrokerHostName=foo
obj.attrs.imqBrokerHostPort=777
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
```

入力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入力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 "imqBrokerHostName=foo"
        -o "imqBrokerHostPort=777"
        -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
```

入力ファイルの別の例は、次の場所で参照できます。

```
IMQ_HOME/demo/imqobjmgr
```

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。

コネクションファクトリ管理対象オブジェクトの追加

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のタイプ (トピックのコネクションファクトリ、またはキュー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を追加します。

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 "imqBrokerHostName=myHost"
  -o "imqBrokerHostPort=7272"
  -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"
```

上記のコマンドでは、検索名が cn=myQCF で、myHost 上で実行しているブローカに接続し、ポート 7272 で待機する管理対象オブジェクトが作成されます。この管理対象オブジェクトは、LDAP サーバに格納されます。

注 LDAP サーバを使用して、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。

引数として入力ファイルに imqobjmgr コマンドを指定すると、同じことを実行できます。詳細は、172 ページの「入力ファイルの使用」を参照してください。

トピックまたはキューの管理対象オブジェクトの追加

クライアントアプリケーションがブローカ上の物理的な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、物理的な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、ブローカの物理的な送信先を作成してください。物理的な送信先の作成方法については、154 ページの「送信先の管理」を参照してください。

次のコマンドでは、検索名が myTopic で、物理的な送信先名が TestTopic のトピック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 サーバの場合、MyCmdFile という入力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"
```

情報の入手

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、個々のオブジェクトに関する情報を表示するには、`list` サブコマンドと `query` サブコマンドを使用します。

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

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、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 imqReconnectRetries=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 8 章

セキュリティ管理

この章では、認証、承認、暗号化といったセキュリティ関連のタスクを実行する方法を説明します。

ユーザの認証 : 管理者は、ユーザ、ユーザグループ、およびパスワードのリストをユーザリポジトリに保持しておく責任があります。この章の最初の節では、ユーザリポジトリの作成、設定、および管理の方法を説明します。MQ セキュリティについては、59 ページの「セキュリティマネージャ」を参照してください。

ユーザの承認 : 管理者は、プロパティファイルを編集する責任があります。プロパティファイルはブローカ操作へのユーザのアクセスをユーザ名またはユーザグループにマッピングします。この章の 2 番目の節では、このプロパティファイルのカスタマイズ方法を説明します。

暗号化 - SSL サービスの設定 : SSL (Secure Socket Layer) 標準に基づくコネクションサービスを使用すると、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できます。MQ による暗号化の処理方法については、61 ページの「暗号化」を参照してください。この章の最後の節では、SSL ベースのコネクションサービスを設定する方法を説明し、SSL 使用に関する追加情報を記載します。

ブローカが SSL キーストア、LDAP ユーザリポジトリ、または JDBC 準拠の持続ストアへ安全にアクセスするのにパスワードが必要となる状況では、次の 3 つの手段でパスワードが要求されます。

- ブローカの起動時に、システムがユーザに要求する
- ブローカの起動時に、コマンド行オプションとしてパスワードを渡す (122 ページの「ブローカの起動」および 123 ページの表 5-2 を参照)
- ブローカの起動時にシステムがアクセスする passfile にパスワードを格納する (205 ページの「passfile の使用」を参照)

ユーザの認証

ユーザ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、参照するように設定されているユーザリポジトリの名前と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。このリポジトリには、次の 2 つのタイプがあります。

- MQ 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リポジトリの使用は簡単です。ユーザ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、ユーザリポジトリにユーザ名、パスワード、およびユーザグループの名前を設定します。

  ユーザリポジトリの設定と管理についての詳細は、184 ページの「単層型ファイルユーザリポジトリを使用する」を参照してください。

- LDAP サーバ

  このリポジトリは、既存または新規の LDAP ディレクトリサーバで、ユーザリポジトリに LDAP v2 または v3 プロトコルを使用します。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、より安全であるため、運用環境に適しています。

  LDAP ユーザ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、ユーザリポジトリを設定、管理する必要があります。詳細は、190 ページの「ユーザリポジトリに LDAP サーバを使用する」を参照してください。

単層型ファイルユーザリポジトリを使用する

MQ には、単層型ファイルユーザリポジトリ、コマンド行ツール、および単層型ファイルユーザリポジトリの設定と管理ができる MQ ユーザマネージャ (`imqusermgr`) が用意されています。次に、単層型ファイルユーザリポジトリ、その初期エントリ、およびリポジトリの設定や管理方法を説明します。

デフォルトの単層型ファイルリポジトリは、次の場所にあります。

`IMQ_HOME/etc/passwd` (Solaris では `/etc/imq/passwd`)

表 8-1 に示すように、リポジトリには定義済みの 2 つのエントリ (行) が付属しています。

表 8-1 ユーザリポジトリでの初期エントリ

ユーザ名	パスワード	グループ	状態
`admin`	`admin`	`admin`	アクティブ

表 8-1　ユーザリポジトリでの初期エントリ (続き)

ユーザ名	パスワード	グループ	状態
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 MQ ブローカを使用できます。言い換えれば、MQ ブローカを使用する上で、ユーザ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、初期の guest ユーザエントリを使って、JMS クライアントはデフォルトの guest ユーザ名とパスワードでブローカに接続できます。

初期の admin ユーザエントリでは、デフォルトの admin ユーザ名とパスワードで、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ことをお勧めします。

- Solaris の場合：インストール後、スーパーユーザ権限を持ったユーザだけがユーザリポジトリファイルに書き込むことができます。スーパーユーザ権限は、ファイルのアクセス権属性を使用するファイルへのアクセスを制御するオペレーティングシステムのポリシーと一貫性があります。
- Windows の場合：インストール後、オペレーティングシステムがユーザ名ベースのアクセス権属性を使用するファイルへのアクセスを制御しないため、あらゆるユーザがユーザリポジトリファイルに書き込むことができます。

ブローカをインストールしたあと、ユーザマネージャユーティリティを使用してユーザリポジトリを設定できます。この設定が完了するまで、ブローカを設定または起動する必要はありません。ユーザマネージャユーティリティを使用する上での唯一の要件は、ブローカがインストールされたホスト上で実行するということです。ブローカが使用するリポジトリの設定、および管理方法については後で説明します。

MQ User Manager サブコマンドとオプション

表 8-2 に imqusermgr サブコマンドを一覧表示します。

表 8-2　imqusermgr サブコマンド

サブコマンド	説明
add -u *name* -p *passwd* [-g *group*] [-s]	ユーザと関連パスワードをリポジトリに追加し、オプションでユーザグループを指定する
delete -u *name* [-s] [-f]	ユーザをリポジトリから削除する
list [-u *name*]	1 人または複数のユーザに関する情報を一覧表示する

表 8-2　imqusermgr サブコマンド (続き)

サブコマンド	説明
`update -u` *name* `-p` *passwd* `[-a` *state*`] [-s] [-f]` `update -u` *name* `-a` *state* `[-p` *passwd*`] [-s] [-f]`	ユーザのパスワードおよび状態を更新する

表 8-3 に imqusermgr コマンドのオプションを一覧表示します。

表 8-3　imqusermgr オプション

オプション	説明
`-a true | false`	ユーザの状態をアクティブにするかどうかを指定する。値が true の場合、状態はアクティブである。デフォルト値 は true
`-f`	ユーザの確認なしにアクションを実行する
`-h`	ヘルプを表示する
`-p` *passwd*	ユーザのパスワードを指定する
`-g admin | user | anonymous`	ユーザグループを指定する
`-s`	サイレントモードを設定する
`-u` *name*	ユーザ名を指定する
`-v`	バージョン情報を表示する

グループ

ユーザエントリをリポジトリに追加するとき、管理者はあらかじめ定義された admin、user、または anonymous のうち、どれかのグループをユーザに指定できます。グループが指定されない場合、デフォルトの user が割り当てられます。

- *admin* グループは、ブローカの管理者です。このグループに割り当てられたユーザは、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、複数のユーザを admin グループに割り当てることができます。
- *user* グループは、管理者ではない通常の JMS クライアントアプリケーションのためのグループです。ほとんどの MQ クライアントアプリケーションは、ユーザグループで認証されたブローカにアクセスします。このように、クライアントアプリケーションでは、デフォルトで、あらゆるトピックやキューに対するメッセージのプロデュース、あらゆるトピックやキューからのメッセージのコンシューム、あるいは任意のキューのメッセージの検索が実行できます。

- *anonymous* グループは、ブローカが認識している名前を使用しない JMS クライアントアプリケーションのためのグループです。それには、アプリケーションが実際に使用するユーザ名を認識していないなどの理由があります。このグループは、多くの FTP サーバにある匿名アカウントに似ています。管理者が一度に anonymous グループに割り当てられるのは、1 人のユーザだけです。管理者は、ユーザグループと比べてこのグループのアクセス権限をアクセス制御を使用して制限したり、あるいは配置時にユーザを削除することが考えられます。

ユーザグループを変更するには、管理者はユーザエントリを削除し、それからユーザの別のエントリを追加して新しいグループを指定します。

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、193 ページの「ユーザの承認 : アクセス制御プロパティファイル」を参照してください。

状態

管理者がユーザをリポジトリに追加したとき、デフォルトのユーザの状態はアクティブです。ユーザを非アクティブに変更するには、管理者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は、新規コネクションを開くことはできません。ユーザが非アクティブのとき、管理者が同じ名前を持つ別のユーザを追加すると、操作に障害が発生します。管理者は非アクティブなユーザのエントリを削除するか、ユーザの名前を変更するか、あるいは新しいユーザに別の名前を使用する必要があります。これは、管理者が重複する名前やパスワードを追加しないようにするためです。

ユーザ名とパスワードの形式

ユーザ名とパスワードは、次の規則に従う必要があります。

- ユーザ名とパスワードには、表 8-4 に一覧表示されている文字を使用できない

表 8-4 ユーザ名とパスワードに無効な文字

文字	説明
*	アスタリスク
,	カンマ
:	コロン

- ユーザ名とパスワードには、改行やキャリッジリターンを文字として使用できない
- ユーザ名やパスワードに空白を入れる場合は、ユーザ名やパスワード全体を引用符で囲む
- ユーザ名やパスワードは、1 文字以上であることが必要
- 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強制されない限り、パスワードやユーザ名の長さに制限はない

ユーザ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 というユーザが追加されます。

```
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 alladin
```

1 人またはそれ以上のユーザ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に関する情報が表示されます。

```
imqusermgr list -u isa
```

```
-----------------------------------
User Name    Group     Active State
-----------------------------------
isa          admin    true
```

次のコマンドでは、すべてのユーザに関する情報が表示されます。

```
imqusermgr list
```

```
--------------------------------------
User Name         Group     Active State
--------------------------------------
testuser3     user          true
testuser2     user          true
testuser1     user          true
isa           admin         true
admin         admin         true
guest         anonymous     true
testuser5     user          false
testuser4     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認識でき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。

次のコマンドでは、デフォルトのパスワードが grandpoobah に変更されます。

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

ブローカ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、次のコマンドを実行します。

```
imqcmd list svc -u admin -p grandpoobah
```

ここでは変更前のパスワードは使用できません。

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。

ユーザリポジトリに LDAP サーバを使用する

ユーザリポジトリに LDAP サーバを使用する場合は、インスタンス設定ファイルで特定のブローカプロパティを設定する必要があります。このプロパティでは、ユーザがブローカへ接続しようとしたり、特定の操作を実行したりすると、ブローカがユーザとグループに関する情報について LDAP にクエリを実行できるようになります。インスタンス設定ファイルは、次の場所にあります。

`IMQ_VARHOME/instances/`*`brokerName`*`/props/config.properties`
(Solaris では、`/var/imq/instances/`*`brokerName`*`/props/config.properties`)

➤ 設定ファイルを編集して、LDAP サーバを使用するには

1. 次のプロパティを設定して、LDAP ユーザ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をユーザ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。これらのプロパティは、ブローカのインスタンス設定ファイルにあります。表 8-5 の説明を参照してください。MQ は、JNDI API を使用して、LDAP ディレクトリサーバと通信します。構文と、これらのプロパティが参照する用語についての詳細は、JNDI のマニュアルを確認してください。MQ 3.0 では、Sun JNDI LDAP プロバイダと、単一の認証が使用されます。

表 8-5　LDAP 関連プロパティ

プロパティ	説明
`imq.user_repository.ldap.server`	LDAP サーバのホスト、およびポート。ホストでは、ディレクトリサーバを実行中の、完全修飾 DNS 名が指定される。ポートでは、ディレクトリサーバが通信に利用しているポート番号が指定される
`imq.user_repository.ldap.principal`	検索時にブローカがディレクトリサーバ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ない

表 8-5　LDAP 関連プロパティ (続き)

プロパティ	説明
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。このパスワードは passfile 内だけで指定可能 (205 ページの「passfile の使用」を参照)。さらにセキュリティを強化するには、ブローカがパスワードを求めるように設定するか、あるいは imqbrokerd -ldappassword というコマンド行オプションを使用してパスワードを指定する ディレクトリサーバで匿名の検索が許可されている場合、パスワードは不要
imq.user_repository.ldap.base	ユーザエントリのためのディレクトリベース
imq.user_repository.ldap.uidattr	プロバイダ固有の属性識別子で、その値はユーザを一意に識別する。たとえば、uid、cn など
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)。ユーザ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る。詳細は、次の場所にある JNDI チュートリアルを参照 http://java.sun.com/products/jndi/tutorial このプロパティの設定は任意
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 ネストされたグループは、MQ 3.0 ではサポートされていないので注意する デフォルトは false
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
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
imq.user_repository.ldap.memattr	グループエントリにある属性識別子で、その値はグループメンバーの識別名

表 8-5　LDAP 関連プロパティ (続き)

プロパティ	説明
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)。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る。詳細は、次の場所にある JNDI チュートリアルを参照 http://java.sun.com/products/jndi/tutorial このプロパティの設定は任意
imq.user_repository.ldap.timeout	検索の時間制限を秒単位で指定する整数。デフォルトでは 180 に設定されている
imq.user_repository.ldap.ssl.enabled	LDAP サーバ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。デフォルトでは false に設定されている

サンプル (デフォルト) の LDAP ユーザリポジトリ関連プロパティ設定については、ブローカの default.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、193 ページの「ユーザの承認 : 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 ユーザリポジトリプロパティに安全な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
     ```

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

ユーザの承認 : アクセス制御プロパティファイル

ブローカを接続したあと、ユーザがメッセージをプロデュースしたり、送信先でメッセージをコンシュームしたり、あるいはキュー送信先でメッセージを検索したりする場合があります。ユーザがこういった操作を試行すると、ブローカはアクセス制御プロパティファイル (ACL ファイル) をチェックし、ユーザが操作を実行する承認を受けていることを確認します。ACL ファイルには、特定のユーザ (またはユーザのグループ) がどの操作の実行を承認されているかを指定する規則が含まれています。デフォルトでは、すべての承認されたユーザは、任意の送信先でメッセージをプロデュースおよびコンシュームできます。アクセス制御プロパティファイルを編集して、特定のユーザやグループによるこれらの操作を制限できます。

ユーザ情報が単層型ファイルリポジトリにあっても、LDAP リポジトリにあっても、ACL ファイルが使用されます。デフォルトの ACL プロパティファイルは、ブローカとともにインストールされます。ACL プロパティファイルの名前は、`accesscontrol.properties` で、インストーラにより次の場所に配置されます。

`IMQ_HOME/etc` (Solaris では `/etc/imq`)

ACL ファイルは、Java プロパティファイルのようにフォーマットされています。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。

- コネクションアクセス制御
- 送信先アクセス制御
- 送信先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、特定のユーザやグループが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 8-6 に構文規則の各要素を示します。

表 8-6 アクセス規則の構文要素

要素	説明
resourceType	`connection`、`queue`、または `topic` のどれか
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、またはすべての送信先を表すのに使用できる
operation	公式化されているアクセス規則の種類に依存する値
access	`allow` または `deny` のどちらか
principalType	`user` または `group` のどちらか。詳細は、186 ページの「グループ」を参照
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を示す。ここでは、`principalType` が `user` の場合は個々のユーザまたはカンマで区切られたユーザのリストとなり、`principalType` が `group` の場合は 1 つのグループまたはカンマで区切られたグループのリストとなる。ワイルドカードの文字 (*) を、すべてのユーザまたはすべてのグループを表すのに使用できる

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、あらゆるユーザがメッセージを q1 という名前のキューに送信する

  ```
  queue.q1.produce.allow.user=*
  ```

- 次の規則では、あらゆるユーザがあらゆるキューにメッセージを送信する

  ```
  queue.*.produce.allow.user=*
  ```

注 ASCII でないユーザ、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) の表記法を使用する必要があります。ASCII でない暗号化方式でこれらの名前の 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/1.3/docs/guide/intl/faq.html` を参照してください。

アクセス権の計算

次の原理は、一連の規則が示すアクセス権を計算するときに適用されます。

- 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、ユーザはだれでもすべてのキューに送信できるが、Bob は tq1 に送信できない

  ```
  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。次の規則では、Bob は tq1 にメッセージをプロデュースできないが、その他のユーザは tq1 にメッセージをプロデュースできる

  ```
  queue.tq1.produce.allow.user=*

  queue.tq1.produce.deny.user=Bob
  ```

- ユーザの * *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がメッセージを tq1 に送信できる

  ```
  queue.tq1.produce.allow.user=*

  queue.tq1.produce.deny.group=*
  ```

- ユーザに許可されたアクセスは、ユーザ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、tq1 にメッセージをプロデュースできないが、User のほかのメンバーは tq1 にメッセージをプロデュースできる

  ```
  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、すべてのユーザはすべての操作を実行できない

- 同じユーザ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 つの規則では、結果として Bob が t1 を検索できなくなる

  ```
  queue.q1.browse.allow.user=Bob
  ```

  ```
  queue.q1.browse.deny.user=Bob
  ```

  次の 2 つの規則では、User というグループが q5 にあるメッセージをコンシュームできなくなる

  ```
  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
```

NORMAL と ADMIN の 2 つの値が、*resourceVariant* に定義されます。デフォルトでは、すべてのユーザが NORMAL タイプにアクセスできますが、admin グループのユーザは、ADMIN タイプのコネクションサービスにアクセスできます。

コネクションアクセス制御規則を編集して、ユーザの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またはグループを指定できます。

自分自身のサービスタイプを作成することは許可されていません。NORMAL と ADMIN の定数で指定された定義済みのタイプに対して自分自身を制限する必要があります。

送信先アクセス制御

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、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 (ユーザまたはグループ) が何 (操作) をどこ (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、あらゆるユーザまたはグループが、任意の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 8-7 にこれらの要素の説明を示します。

表 8-7 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` または `topic` のどちらか
resourceVariant	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全送信先 (*)
operation	`produce`、`consume`、または `browse` のどれか
access	`allow` または `deny` のどちらか
principalType	`user` または `group` のどちらか

アクセスは、1 人以上のユーザまたは 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、さまざまな種類の送信先アクセス制御規則を示します。

- すべてのユーザが、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
  ```

送信先自動作成アクセス制御

ACL プロパティファイルの最後のセクションには、どのユーザおよびグループに対してブローカが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。

ユーザがまだ存在していない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て、物理的送信先がまだ存在していない場合、ブローカは送信先を作成します。

デフォルトでは、任意のユーザやグループは、ブローカに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。

送信先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
```

送信先自動作成規則の結果は、送信先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、ブローカは送信先が存在しない場合、送信先を作成しますが、メッセージの配信は行いません。

暗号化 : SSL サービスの操作

MQ Enterprise Edition (26 ページの「製品エディション」を参照) は、TCP/IP (ssljms および ssladmin) や HTTP (httpsjms) を介した SSL (Secure Socket Layer) 標準に基づくコネクションサービスがサポートしています。これらの SSL ベースのコネクションサービスで、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できます。現在の MQ リリースでは、自己署名型サーバ証明書に基づく 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利用するには、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組み合わせを生成する必要があります。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。

MQ 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。したがって、TCP/IP および HTTP を介した安全なコネクションについては、次に別々に説明します。

TCP/IP を介した SSL サービスの設定

TCP/IP を使用してダイレクトで安全なコネクションを提供する SSL ベースのコネクションサービスには、2 種類あります。

ssljms: このコネクションサービスは、クライアントとブローカ間の安全で暗号化されたコネクションを経由した JMS メッセージの配信に使用されます。

ssladmin: このコネクションサービスは、コマンド行管理ツールであるコネクションユーティリティ (`imqcmd`) とブローカ間の、安全で暗号化されたコネクションの確立に使用されます。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。

➤ ssljms コネクションサービスを設定するには

1. 自己署名型証明書を生成します。
2. ブローカでの ssljms コネクションサービスを有効にします。
3. ブローカを起動します。
4. クライアントを設定、実行します。

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

MQ 3.0 の SSL Support は、クライアントが既知の信頼されたサーバ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 MQ 3.0 では、自己署名型証明書だけを使用して SSL が実装されます。

`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。ssljms と ssladmin の両方のコネクションサービスには、同じ証明書が使用されます。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、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 (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。あとでブローカを起動したときにキーストアを開くため、そのパスワードを入力する必要があります。また、キーストアパスワードを passfile (205 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、次の場所にある MQ のキーストアに配置されます。

`IMQ_HOME/etc/keystore` (Solaris では `/etc/imq/keystore`)

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フォーマットになっています。

MQ キーストアの設定可能なプロパティを表 8-8 に示します。各プロパティの説明については、第 5 章「ブローカの起動と設定」を参照してください。

表 8-8　キーストアのプロパティ

プロパティ名	説明
imq.keystore.file.dirpath	SSL ベースのサービスの場合、キーストアファイルを含むディレクトリへのパスを指定する デフォルトは、IMQ_HOME/etc (Solaris では /etc/imq)
imq.keystore.file.name	SSL ベースのサービスの場合、キーストアファイルの名前を指定する デフォルトは、keystore
imq.keystore.password	SSL ベースのサービスの場合、キーストアパスワードを指定する。このパスワードは passfile 内だけで指定可能 (205 ページの「passfile の使用」を参照)。さらにセキュリティを強化するには、ブローカがパスワードを求めるように設定するか、あるいは imqbrokerd -password というコマンド行オプションを使用してパスワードを指定する

たとえば次のような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
- ブローカを起動して、次のような例外を受け取ったときに、SSL サービスに障害が発生して初期化されてしまった
  java.security.UnrecoverableKeyException: Cannot recover key

  この例外は、自己署名型証明書を 200 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には

1. ブローカのキーストアを、次の場所から削除します。

   IMQ_HOME/etc/keystore (Solaris では /etc/imq/keystore)

2. imqkeytool をもう 1 度実行し、200 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 サービスを有効にするには、ssljms (ssladmin) を imq.service.activelist プロパティに追加する必要があります。

1. ブローカのインスタンス設定ファイルを開きます。ブローカのインスタンス設定ファイルは、次の場所にあります。

   IMQ_VARHOME/instances/*brokerName*/props/config.properties

   brokerName には、ブローカインスタンスの名前が表示されます。

2. ssljms か ssladmin、またはその両方 (希望するサービスによる) を、imq.service.activelist プロパティに追加します。

   imq.service.activelist=jms,admin,httpjms,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*

- imqbrokerd コマンドに -password オプションを使用する

  imqbrokerd -password *mypassword*

- ブローカの起動時にアクセスする passfile ファイル (205 ページの「passfile の使用」を参照) にパスワードを入力する。最初に、ブローカ設定プロパティ (118 ページの「インスタンス設定ファイルの編集」を参照) を次のように設定しておく必要がある

  imq.passfile.enabled=true

  このプロパティが設定されると、次のどちらかの方法で passfile にアクセスできる

  - 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 つのブローカ設定ファイルを使用する 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*

passfile 関連ブローカプロパティの一覧については、62 ページの表 2-6 を参照してください。

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、MQ 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

最後にクライアントを設定して、安全なコネクションサービスを使用するようにする必要があります。使用しているコネクションサービスによって、次の 2 つのタイプのクライアントがあります。ssljms を使用する JMS クライアントと、ssladmin を使用する MQ 管理 Command ユーティリティ (`imqcmd`) です。後続の節では、これらについて次で別個に説明します。

JMS クライアント

クライアントが必要な Secure Socket Extension (JSSE) 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 MQ 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、『MQ 開発者ガイド』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。

コマンドユーティリティ (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 (142 ページの表 6-2 を参照)。たとえば、次のように指定します。

```
imqcmd list svc -b hostName:port -u adminName -p adminPassword -secure
```

adminName と *adminPassword* には、MQ ユーザリポジトリにある有効なエントリが表示されます (単層型ファイルリポジトリを使用している場合は、189 ページの「デフォルトの管理者パスワードの変更」を参照)。

例のように、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を示されます。

```
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.
```

HTTP を介した SSL サービスの設定

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は、HTTPS トンネルサーブレット経由で安全なコネクションを確立します。HTTPS サポートのアーキテクチャと実装は、215 ページの付録 B「HTTP/HTTPS サポート」で説明されています。

passfile の使用

ブローカに必要なパスワードを要求させずに起動する場合や、`imqbrokerd` コマンドのオプションとしてこれらのパスワードの入力を要求させずに起動する場合、必要なパスワードを *passfile* に置くことができます。

passfile は、パスワードを含む簡単なテキストファイルです。ファイルは暗号化されないため、手動でパスワードを入力するより安全性は低くなります。ただし、このファイルにアクセス権を設定して、ファイルにアクセスして表示できるユーザを制限できます。passfile のアクセス権では、ブローカを起動したユーザに読み込みのアクセス権を与える必要があります。

passfile には、表 8-9 に示すパスワードを含めることができます。

表 8-9 passfile のパスワード

パスワード	説明
`imq.keystore.password`	SSL ベースサービスにキーストアパスワードを指定する
`imq.user_repository.ldap.password`	設定された LDAP ユーザ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する
`imq.persist.jdbc.password`	必要に応じて、データベースコネクションを開くためのパスワードを指定する

サンプル passfile は、次の場所にあります。

`IMQ_HOME/etc/`passfile.sample (Solaris では、`/etc/imq/passfile.sample`)

付録 A

プラグイン持続の設定

この付録では、JDBC アクセスが可能なデータストアにアクセスするために、プラグイン持続を使用するブローカの設定方法について説明します。

概要

MQ のブローカには、持続情報の書き込みおよび検索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 (56 ページの「持続マネージャ」を参照)。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。

プラグイン持続を使用するようにブローカを設定するには、ブローカインスタンスの設定ファイルに、多数の JDBC 関連のプロパティを設定する必要があります。また、MQ の持続処理を実行するために、適切なデータベーススキーマを作成する必要があります。MQ には、JDBC ドライバおよびブローカの設定プロパティを使用して、プラグインデータベースの作成および管理を行う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。

この付録に示す手順は、例として Java 2 SDK Enterprise Edition (J2EE SDK は java.sun.com からダウンロード可能) にバンドルされる Cloudscape DBMS を使用して説明しています。例では、クライアント / サーババージョンの代わりに、Cloudscape の組み込みバージョンを使用します。手順の指示は、Cloudscape 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。

その他の例は、次の場所で参照できます。

`IMQ_HOME/demo/jdbc` (Solaris では `/usr/demo/imq/jdbc`)

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には

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。

   209 ページの表 A-1 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。

   `IMQ_VARHOME/lib/ext` (Solaris では `/usr/share/lib/imq/ext/`)

   コピーの例 (Solaris) :

   ```
   % cp j2sdk_install_directory/lib/cloudscape/cloudscape.jar
   IMQ_VARHOME/lib/ext
   ```

   シンボリックリンクの例 (Solaris) :

   ```
   % ln -s j2sdk_install_directory/lib/cloudscape/cloudscape.jar
   IMQ_VARHOME/lib/ext
   ```

3. MQ 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。212 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 :

   `% cd IMQ_HOME/bin` (Solaris では `/usr/bin`)

   ```
   % imqdbmgr create all
   ```

JDBC 関連のブローカの設定プロパティ

ブローカのインスタンス設定ファイルは、次の場所に配置されます。

```
IMQ_VARHOME/instances/brokerName/props/config.properties
```

ファイルが存在しない場合、ファイルを作成する MQ に対して、`-name brokerName` 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。

表 A-1 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。プラグイン持続を使用して、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。この表には、Cloudscape DBMS の例で指定する値が含まれています。

表 A-1 JDBC 関連のプロパティ

プロパティ名	説明
`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 例 : `jdbc`
`imq.persist.jdbc.brokerid` (省略可能)	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を指定する。通常、1 つのブローカインスタンスだけのデータを格納する組み込みデータベースでは不要。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る 例 : Cloudscape では不要
`imq.persist.jdbc.driver`	データベースに接続する JDBC ドライバの Java クラス名を指定する 例 : `COM.cloudscape.core.JDBCDriver`
`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 を指定する 例 : `jdbc:cloudscape:IMQ_VARHOME/instances/brokerName/dbstore/imqdb`

表 A-1　JDBC 関連のプロパティ (続き)

プロパティ名	説明
imq.persist.jdbc.createdburl (省略可能)	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 を指定する。imqdbmgr を使用して、データベースを作成する場合にだけ指定する 例 : jdbc:cloudscape:IMQ_VARHOME/instances/*brokerName*/dbstore/imqdb;create=true
imq.persist.jdbc.closedburl (省略可能)	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 を指定する 例 (Cloudscape で必要) : jdbc:cloudscape:;shutdown=true
imq.persist.jdbc.user (省略可能)	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名を指定する。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る imqbrokerd -dbuser および imqdbmgr -u
imq.persist.jdbc.needpassword (省略可能)	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。値を true にすると、パスワードが必要になる。パスワードは、次のコマンド行オプションを使用して指定できる imqbrokerd -dbpassword imqdbmgr -p コマンド行オプション、またはパスファイル (205 ページの「passfile の使用」を参照) のどちらかを使用してパスワードを指定しないと、ブローカによってパスワードの入力が要求される
imq.persist.jdbc.password (省略可能)	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する。このパスワードは、パスファイル内だけで指定可能 (205 ページの「passfile の使用」を参照)。セキュリティを向上させるために、ブローカにパスワードの入力を要求させるか、あるいは次のコマンド行オプションを使用してパスワードを指定する imqbrokerd -dbpassword imqdbmgr -p

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 :

Cloudscape 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 (表 A-1 の例を参照) 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、Cloudscape システムディレクトリを定義することができます。

```
-Dcloudscape.system.home=IMQ_VARHOME/instances/brokerName/dbstore
```

この場合、データベースを作成したり、開いたりするための URL は、それぞれ次のように指定できます。

```
imq.persist.jdbc.createdburl=jdbc:cloudscape:imqdb;create=true
```

および

```
imq.persist.jdbc.opendburl=jdbc:cloudscape:imqdb
```

データベース管理ユーティリティ (imqdbmgr)

MQ 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。データベース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、MQ のデータベーステーブルを削除することもできます。

組み込みデータベースを使用し、IMQ_VARHOME/instances/*brokerName*/ ディレクトリの下に、データベースを作成する場合は、次のディレクトリの下に作成することをお勧めします。

IMQ_VARHOME/instances/*brokerName*/dbstore/*dabatabseName*.

組み込みデータベースは、ユーザ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。ブローカがデータベースを読み取ったり、書き込んだりするためには、ブローカを実行するユーザが、組み込みデータベースを作成するのに使用する imqdbmgr コマンドを実行する必要があります。

表 A-2 に imqdbmgr のサブコマンドを一覧表示します。

表 A-2 imqdbmgr のサブコマンド

サブコマンド	説明
create all	新しいデータベースと MQ の持続データストレージのスキーマを作成する。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る
create tbl	既存のデータベースシステムに、MQ の持続データストレージのスキーマを作成する。このコマンドは、外部データベースシステムで使用する
delete tbl	現在の持続データストレージのデータベース内に存在する MQ のデータベーステーブルを削除する
recreate tbl	現在の持続データストレージのデータベース内に存在する MQ のデータベーステーブルを削除した後、MQ の持続データストレージのスキーマを作成し直す

表 A-3 に imqdbmgr コマンドのオプションを一覧表示します。

表 A-3　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する。
-b *name*	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する
-h	ヘルプを表示する
-p *passwd*	データベースのパスワードを指定する
-u *name*	データベースのユーザ名を指定する
-v	バージョン情報を表示する

付録 B

HTTP/HTTPS サポート

MQ Enterprise Edition (26 ページの「製品エディション」を参照) には、HTTP コネクションと HTTPS コネクションの両方のサポートが含まれています (HTTPS は HTTP を介した安全なコネクションで、Secure Socket Layer 標準を使用)。このため、クライアントアプリケーションは、直接 TCP コネクションを使用せずに HTTP プロトコルを使用しているブローカと通信できます。この付録では、このようなサポートを実装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 MQ 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。

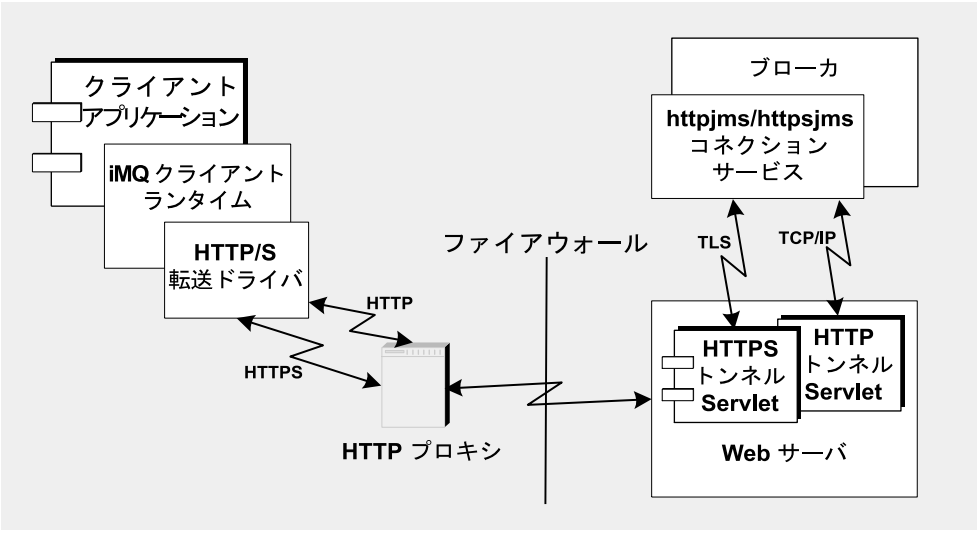
HTTP/HTTPS サポートのアーキテクチャ

MQ 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。

216 ページの図 B-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 転送ドライバが MQ メッセージを HTTP 要求にカプセル化し、これらの要求が正しい順番で Web サーバに送信されることを確認する
- クライアントアプリケーションは、HTTP プロキシサーバを使用して、必要に応じてブローカと通信できる。プロキシのアドレスは、クライアントアプリケーションの起動時に、コマンド行オプションを使用して指定する。詳細は、221 ページの「HTTP プロキシを使用する」を参照
- HTTP または HTTPS トンネルサーブレット (どちらも MQ にバンドルされている) は、Web サーバに読み込まれ、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する。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る。

図 B-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、非多重化する

図 B-1 に示すように、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 MQ 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 サーバがネゴシエーションを行います。

HTTP サポートの実装

次に、HTTP サポートを実装するのに必要な手順を説明します。

➤ HTTP サポートを実装するには

1. HTTP トンネルサーブレットを Web サーバに配置します。
2. ブローカの httpjms コネクションサービスを設定し、ブローカを起動します。
3. HTTP コネクションを設定します。

手順 1. HTTP トンネルサーブレットを Web サーバに配置する

HTTP トンネルサーブレットを Web サーバに配置するには、通常 次の 2 つの方法があります。

- jar ファイルとして配置する (Servlet 2.1 以前をサポートする Web サーバの場合)
- Web アーカイブ (WAR) として配置する (Servlet 2.2 以降をサポートする Web サーバの場合)

jar ファイルとして配置する

MQ トンネルサーブレットを配置する作業は、適切な jar ファイルをホスト Web サーバに対してアクセス可能にすることと、その Web サーバが起動時にサーブレットを読み込むように設定することで成り立っています。

トンネルサーブレット jar ファイル (`imqservlet.jar`) には、HTTP トンネルサーブレットに必要なすべてのクラスが含まれています。このファイルは、次の場所にあります。

`IMQ_HOME/lib` (Solaris では `/usr/share/lib/imq`)

Servlet 2.x をサポートする Web サーバは、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 サーバは、`imqservlet.jar` ファイルを参照する必要があります。Web サーバとブローカを異なるホストで実行する場合は、Web サーバ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。

起動時にこのサーブレットを読み込むように、Web サーバを設定する必要があります (221 ページの「例 : HTTP トンネルサーブレットを配置する」を参照)。

パフォーマンスを向上させるために、Web サーバのアクセスログ作成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 WAR ファイル (`imqhttp.war`) は、次のディレクトリにあります。

`IMQ_HOME/lib` (Solaris では `/usr/share/lib/imq`)

WAR ファイルには、Web サーバ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。

手順 2. httpjms コネクションサービスを設定する

HTTP サポートは、MQ 3.0 のブローカのデフォルトでは有効になっていないため、ブローカを設定して、httpjms コネクションサービスを有効にする必要があります。設定し直すと、122 ページの「ブローカの起動」で説明されているように、ブローカを起動できます。

➤ httpjms コネクションサービスを有効にするには

1. 次の場所にあるブローカのインスタンス設定ファイルを開きます。

   `IMQ_VARHOME/instances/`*`brokerName`*`/props/config.properties`

   brokerName には、ブローカインスタンスの名前が表示されます。

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、219 ページの表 B-1 を参照してください。

表 B-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) を指定する。(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) デフォルトは 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する。Web サーバ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する。デフォルトは 7675
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が出す HTTP 要求の間隔を秒単位で指定する。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る。クライアント数が多いと、この動作によって Web サーバのリソースが消耗し、サーバの応答が遅くなる場合がある。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る。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る。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 サーバのリソースが維持される。デフォルトは -1

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の属性の設定

HTTP サポートを実装するには、次のコネクションファクトリの属性を設定します (168 ページの「コネクションファクトリ管理対象オブジェクト」を参照)。

- `imqConnectionType` 属性を `HTTP` に設定する
- `imqConnectionURL` に HTTP トンネルサーブレットの URL を設定する

  `http://`*hostName*`:`*port*`/imq/tunnel`

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。

- `-o` オプションを、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` コマンドに使用するか、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する (175 ページの「コネクションファクトリ管理対象オブジェクトの追加」を参照)
- 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` オプションを使用する (『MQ 開発者ガイド』を参照)
- プログラム的にクライアントアプリケーションコードに JMS 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する (『MQ 開発者ガイド』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。共有するには、`imqConnectiodnURL` コネクションファクトリの属性を、次のように設定する必要があります。

`http://`*hostName:port/imq/tunnel*`?ServerName=`*hostName:brokerName*

hostName にはブローカのホスト名が入り、*brokerName* 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。

hostName と *brokerName* 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のホスト名に設定する
- `http.proxyPort` システムプロパティをプロキシサーバのポート番号に設定する

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 : HTTP トンネルサーブレットを配置する

ここでは、HTTP トンネルサーブレットを Sun ONE 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 ONE 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 ONE Web Server、FastTrack Edition 4.1 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には

1. 「Servlets」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。

   ```
   IMQ_HOME/lib/imqservlet.jar
   ```
   (Solaris の場合、`/usr/share/lib/imq/imqservlet.jar`)

6. 「Servlet args」フィールドに、表 B-2 に示すオプションの引数を入力します。

表 B-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	219 ページの表 B-1 を参照
servletPort	`7675`	219 ページの表 B-1 を参照

両方の引数を使用する場合は、次のように引数をカンマで区切ります。

`servletPort=`*portnumber*`, servletHost=`*...*

`serverHost` 引数と `serverPort` 引数は、Web サーバ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されます。その場合、状況に応じてブローカ設定プロパティを設定する必要があります (219 ページの表 B-1 を参照)。たとえば、次のように設定されます。

```
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) を設定するには

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。

   たとえば、URL に `http://`*hostName*`:`*port*`/imq/tunnel` を指定する場合、「Virtual Path」フィールドに次の文字列を入力します。

   ```
   /imq/tunnel
   ```

4. 「Servlet Name」フィールドに 222 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 サーバの起動時にトンネルサーブレットを読み込むには

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、222 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。

サーバのアクセスログを無効にする

必ずしもサーバ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。

➤ サーバのアクセスログを無効にするには

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、ログ作成を無効にします。

WAR ファイルとして配置する

次の手順では、Sun ONE Web Server 6.0 Service Pack 2 での配置について説明しています。

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には

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. `imqhttp.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。これは、次のディレクトリにあります。

   `IMQ_HOME/lib` (Solaris では `/usr/share/lib/imq`)
8. 「Application URI」フィールドに、「`/imq`」(入力時には、かぎカッコ不要) を入力します。
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 ONE Web Server インストールルートの任意の場所) を入力します。
10. 「`OK`」をクリックします。
11. Web サーバ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`*`/imq/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。

HTTPS サポートを実装する

次に、HTTPS サポートを実装するのに必要な手順を説明します。この手順は、217 ページの「HTTP サポートの実装」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 サポートを実装するには

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

MQ 3.0 での SSL Support は、クライアントが既知の信頼されたサーバ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 MQ 3.0 では、自己署名型サーバ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 (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、MQ 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フォーマットになっています。

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 (226 ページの「手順 2. HTTPS トンネルサーブレットを Web サーバに配置する」を参照)。

手順 2. HTTPS トンネルサーブレットを Web サーバに配置する

HTTPS トンネルサーブレットを Web サーバに配置するには、通常、次の 2 つの方法があります。

- jar ファイルとして配置する (Servlet 2.1 以前をサポートする Web サーバの場合)
- Web アーカイブ (WAR) として配置する (Servlet 2.2 以降をサポートする Web サーバの場合)

どちらの場合も、Web サーバ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

MQ トンネルサーブレットを配置する作業は、適切な jar ファイルをホスト Web サーバに対してアクセス可能にすることと、その Web サーバが起動時にサーブレットを読み込むように設定することで成り立っています。

トンネルサーブレット jar ファイル (imqservlet.jar) には、HTTPS トンネルサーブレットで必要となるすべてのクラスが含まれています。このファイルは、次の場所にあります。

IMQ_HOME/lib (Solaris では /usr/share/lib/imq)

Servlet 2.x をサポートする Web サーバは、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 サーバは、imqservlet.jar ファイルを参照する必要があります。Web サーバとブローカを異なるホストで実行する場合は、Web サーバ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。

起動時にこのサーブレットを読み込むように、Web サーバを設定する必要があります(232 ページの「例 : HTTPS トンネルサーブレットを配置する」を参照)。

サーブレットを Web サーバで実行するために、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 サーバのマニュアルを参照してください。

Web サーバ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。225 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 サーバのアクセスログ作成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 WAR ファイル (imqhttps.war) は、次のディレクトリにあります。

IMQ_HOME/lib (Solaris では /usr/share/lib/imq)

WAR ファイルには、Web サーバ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。

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(225 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

HTTPS サポートは、MQ 3.0 のブローカのデフォルトでは有効になっていないため、ブローカを設定して、httpsjms コネクションサービスを有効にする必要があります。設定し直すと、122 ページの「ブローカの起動」で説明されているように、ブローカを起動できます。

➤ httpsjms コネクションサービスを有効にするには

1. 次の場所にあるブローカのインスタンス設定ファイルを開きます。

   `IMQ_VARHOME/instances/`*brokerName*`/props/config.properties`

   brokerName には、ブローカインスタンスの名前が表示されます。

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 B-3 を参照してください。

表 B-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) を指定する。(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) デフォルトは `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する。Web サーバ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する。デフォルトは `7674`

表 B-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する。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る。クライアント数が多いと、この動作によって Web サーバのリソースが消耗し、サーバの応答が遅くなる場合がある。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る。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る。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 サーバのリソースが維持される。デフォルトは、-1

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、設定済み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、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 を設定するには

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

   n には、セキュリティプロバイダパッケージの次の利用可能優先順位が表示されます。

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の属性を設定する」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想定して、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の属性を設定する

HTTPS サポートを実装するには、次のコネクションファクトリの属性を設定します (168 ページの「コネクションファクトリ管理対象オブジェクト」を参照)。

- imqConnectionType 属性を HTTP に設定します。

  ブローカへの安全なコネクションは、HTTP トンネルサーブレットではなく、HTTPS トンネルサーブレットを介して配置および接続することで確立されます。ただし、どちらのトンネルサーブレットも同じコネクションタイプを使用します。

- imqConnectionURL に HTTPS トンネルサーブレット URL を設定します。

  ```
  https://hostName:port/imq/tunnel
  ```

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。

- -o オプションを、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 コマンドに使用するか、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する (175 ページの「コネクションファクトリ管理対象オブジェクトの追加」を参照)

- 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 オプションを使用する (『MQ 開発者ガイド』を参照)
- プログラム的にクライアントアプリケーションコードに JMS 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する (『MQ 開発者ガイド』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 サーバと HTTPS トンネルサーブレットを共有できます。共有するには、imqConnectiodnURL コネクションファクトリの属性を、次のように設定する必要があります。

`https://`*hostName:port/imq/tunnel*?ServerName=*hostName*:*brokerName*

hostName の部分にはブローカのホスト名が入り、*brokerName* 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。

hostName と *brokerName* 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のホスト名に設定する
- `http.proxyPort` システムプロパティをプロキシサーバのポート番号に設定する

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：HTTPS トンネルサーブレットを配置する

ここでは、HTTPS トンネルサーブレットを Sun ONE 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 ONE 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 ONE Web Server、FastTrack Edition 4.1 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には

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。

   ```
   IMQ_HOME/lib/imqservlet.jar
   ```
   (Solaris の場合、`/usr/share/lib/imq/imqservlet.jar`)

6. 「Servlet args」フィールドに、表 B-4 に示す必要なオプションの引数を入力します。

表 B-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	関連項目
keystoreLocation	なし	必須	201 ページの表 8-8
keystorePassword	なし	必須	201 ページの表 8-8
serverHost	all hosts	オプション	228 ページの表 B-3
serverPort	7674	オプション	228 ページの表 B-3

引数は、次のようにカンマで区切ります。

keystoreLocation=*keystore_location*,keystorePassword=*keystore_password*,servletPort=*portnumber*

serverHost 引数と serverPort 引数は、Web サーバ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されます。ただしその場合、状況に応じてブローカ設定プロパティを設定する必要があります (228 ページの表 B-3 を参照)。たとえば、次のように設定されます。

```
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) を設定するには

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。

   たとえば、URL に http://*hostName*:*port*/imq/tunnel を指定する場合、「Virtual Path」フィールドに次の文字列を入力します。

   /imq/tunnel

4. 「Servlet Name」フィールドに 232 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 サーバの起動時にトンネルサーブレットを読み込むには

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、232 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。

サーバのアクセスログを無効にする

必ずしもサーバ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。

➤ サーバのアクセスログを無効にするには

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、ログ作成を無効にします。

WAR ファイルとして配置する

次の手順では、Sun ONE Web Server 6.0 Service Pack 2 での配置について説明しています。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。これを確認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には

1. WAR ファイルを一時ディレクトリにコピーします。

   ```
   $ cp IMQ_HOME/lib/imqhttps.war /tmp
   ```
   (Solaris の場合、`$ cp /usr/share/lib/imq/imqhttps.war /tmp`)

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)。

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には

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 (234 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には」を参照)。

8. 「Application URI」フィールドに、「/imq」(入力時には、かぎカッコ不要) を入力します。

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 ONE Web Server インストールルートの任意の場所) を入力します。

10. 「OK」をクリックします。

11. Web サーバ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。

Windows のサービスとしてのブローカの使用

この付録では、Windows のサービスとして実行するブローカのインストール、クエリ、および削除を行うサービス管理ユーティリティ (imqsvcadmin) の使用方法について説明します。

Windows のサービスとしてのブローカの実行

MQ 3.0 をインストールするときに、ブローカを Windows のサービスとしてインストールすることができます。また、MQ 3.0 のインストール後に、imqsvcadmin を使用して、ブローカを Windows のサービスとしてインストールすることも可能です。

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。ブローカにスタートアップのオプションを渡すには、imqsvcadmin コマンド (239 ページの表 C-2 を参照) で -args 引数を使用し、imqbrokerd コマンド (122 ページの「ブローカの起動」を参照) に使用したのと同じオプションを指定します。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。

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。1 つ目は、Windows のネイティブサービスラッパーである imqbrokersvc.exe です。2 つ目は、実際にブローカを実行する Java ランタイムです。

1 回につき 1 つのブローカだけを、Windows のサービスとしてインストールおよび実行できます。

サービス管理ユーティリティ (imqsvcadmin)

サービス管理ユーティリティ (imqsvcadmin) を使用すると、Windows のサービスとして起動するブローカのインストール、クエリ、および削除を行うことができます。この節では、imqsvcadmin コマンドの基本構文、サブコマンドのリスト、imqsvcadmin コマンドのオプションの概要、および特定のタスクを実行するこれらのコマンドの使用方法について説明します。

コマンドの構文

imqsvcadmin コマンドの一般的な構文は、次のとおりです。

```
imqsvcadmin subcommand [options ]
```

-v、-h、および -H オプションを指定する場合、コマンド行で指定するその他のサブコマンドは実行できません。たとえば、次のコマンドを入力すると、ヘルプ情報が表示されますが、query サブコマンドは実行されません。

```
imqcmd query -h
```

imqsvcadmin のサブコマンド

MQ サービス管理ユーティリティ (imqsvcadmin) には、次の 表 C-1 に示すようなサブコマンドが含まれています。

表 C-1　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する
query	imqsvcadmin コマンドのスタートアップのオプションを表示する。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る
remove	サービスを削除する

imqsvcadmin のオプションの概要

表 C-2 に、imqsvcadmin コマンドのオプションを一覧表示します。これらの使用方法については、後続のタスクごとの節を参照してください。

表 C-2　imqsvcadmin のオプション

オプション	説明
-javahome *path*	代替の Java 2 互換の JDK へのパスを指定する。デフォルトでは、バンドルされたランタイムを使用する 例 : imqsvcadmin -install -javahome d:¥jdk1.4
-jrehome *path*	Java 2 互換の JRE へのパスを指定する 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する。(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) 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する。imqbrokerd オプションについては、122 ページの「ブローカの起動」を参照 (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) 例 : 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、次のパスの JavaHome キー、JVMArgs キー、および ServiceArgs キーの下にある Windows のレジストリに保存されます。

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

ブローカサービスの削除

ブローカサービスを削除する前に、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ておく必要があります。次に、imqsvcadmin remove コマンドを使用して、サービスを削除し、コンピュータを再起動します。

ブローカサービスの再設定

サービスを再設定するには、最初にサービスを削除してから、-args 引数を使用して異なるスタートアップのオプションを指定し、サービスをインストールし直します。

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 (¥) は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、-javahome d:¥¥jdk1.3 などのように、円記号を 2 つ入力してください。

ブローカサービスのクエリ

ブローカサービスのスタートアップのオプションを指定するには、imqsvcadmin コマンドの -q 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 3.0¥bin¥imqbrokersvc
JavaHome: c:¥j2sdk1.4.0
Broker Args: -passfile d:¥imqpassfile
```

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、次の操作を実行すると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には

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。

付録 D

MQ データの場所

MQ では、さまざまなカテゴリのデータを使用します。これらのデータは、表 D-1 に示すようにそれぞれ異なる場所に保存されています。

表 D-1　MQ 3.0 データの場所

データのカテゴリ	データの場所
ブローカインスタンスの設定プロパティ	IMQ_VARHOME/instances/*brokerName*/props/config.properties (Solaris では、/var/imq/instances/*brokerName*/props/config.properties)
持続ストア (メッセージ、出力先、永続サブスクリプション、トランザクション)	IMQ_VARHOME/instances/*brokerName*/filestore/ (Solaris では、/var/imq/instances/*brokerName*/filestore/) 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	IMQ_VARHOME/instances/*brokerName*/log/ (Solaris では、/var/imq/instances/*brokerName*/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
セキュリティ : ユーザリポジトリ	IMQ_HOME/etc/passwd (Solaris では、/etc/imq/passwd) または LDAP サーバ
セキュリティ : アクセス制御ファイル	IMQ_HOME/etc/accesscontrol.properties (Solaris では、/etc/imq/accesscontrol.properties)

表 D-1　MQ 3.0 データの場所 (続き)

データのカテゴリ	データの場所
セキュリティ : passfile	`IMQ_HOME/etc/`*passfileName* (Solaris では、`/etc/imq/`*passfileName*)

付録 E

MQ インタフェースの安定度

MQ では、管理タスクを自動化して管理者の役に立つさまざまなインタフェースを使用します。表 E-1 では、これらのインタフェースが安定度によって分類されています。安定度とは、後続バージョンの製品で変更が生じる可能性の低さを意味します。分類方式については、247 ページの表 E-2 で説明しています。

表 E-1　MQ 3.0 インタフェースの安定度

インタフェース	分類
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
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

表 E-1　MQ 3.0 インタフェースの安定度 (続き)

インタフェース	分類
imqobjmgr コマンド	安定
imqusermgr コマンド	不安定
JMS API (javax.jms)	標準
JAXM API (javax.xml)	標準
管理対象オブジェクト API (com.sun.messaging)	発展中
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
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
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	非公開
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
providerutil.jar の格納場所および名前	不安定
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
passfile	不安定
accesscontrol.properties	不安定

表 E-2　インタフェースの安定度の分類方式

分類	説明
非公開	ユーザは直接使用しない。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る
発展中	ユーザが使用する。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る。変更は慎重に、かつ徐々に行われる。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るが、保証はされていない
安定	ユーザが使用する。互換性のない変更は、メジャーリリース (3.0 や 4.0 など) でしか生じない
標準	ユーザが使用する。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る。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ない
不安定	ユーザが使用する。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る。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相当に削除や変更が行われる可能性があることに注意。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ようにする

用語集

この用語集では、MQ の使用時に知っておくと便利な用語や概念について説明します。

JMS (Java Message Service)　メッセージサービス機能へのクライアントアプリケーションのアクセス方法について定義するインタフェースおよびセマンティックの標準セット。これらのインタフェースには、メッセージの作成、送信、受信、および読み取りを行うための Java プログラム用の標準手段が用意されている。

JMS プロバイダ (JMS provider)　メッセージングシステムの JMS インタフェースを実装し、製品全体に必要な管理および制御機能を追加する製品。

MQ クライアントランタイム (MQ client runtime)　クライアントアプリケーションに MQ メッセージサーバへのインタフェースを提供するソフトウェア。クライアントランタイムは、クライアントが送信先にメッセージを送信し、送信先からメッセージを受信するために必要なすべての操作をサポートする。

MQ メッセージサーバ (MQ message server)　クライアントアプリケーションへのコネクション、メッセージのルートおよび配信、持続性、セキュリティ、ログ記録などの MQ メッセージングシステム用の配信サービスを提供するソフトウェア。メッセージサーバは、クライアントアプリケーションのメッセージ送信先であり、コンシューミングクライアントへのメッセージ配信元である物理的な送信先を管理する。

管理対象オブジェクト (administered objects)　コネクションファクトリ、送信先など、あらかじめ設定された MQ のオブジェクト。1 つ以上のクライアントアプリケーションで使用するために、管理者が作成する。

管理対象オブジェクトを使用すると、クライアントアプリケーションがプロバイダ依存しないようにすることができる。つまり、クライアントアプリケーションがプロバイダ固有の問題から切り離される。これらのオブジェクトは、管理者によって JNDI ネームスペースに配置され、JNDI 検索を使用するクライアントアプリケーションからアクセスされる。

キュー (queue)　ポイントツーポイント配信モデルを実装するために、管理者が作成するオブジェクト。メッセージをコンシュームするクライアントがアクティブでない場合でも、メッセージを保持するためにキューは常に使用可能である。キューは、プロデューサとコンシューマの中間段階の待機場所として使用される。

クライアントアプリケーション (client application)　メッセージを交換するメッセージサービスを使用して、ほかのクライアントアプリケーションと対話するアプリケーション (またはソフトウェアコンポーネント)。

クライアント識別子 (client identifier)　コネクションおよびコネクションのオブジェクトをクライアントアプリケーションの代わりに、MQ メッセージサーバが管理する状態と関連付ける識別子。

クライアントランタイム (client runtime)　「MQ クライアントランタイム」を参照。

クラスタ (cluster)　メッセージングサービスを提供するために、並行して処理を行う複数の連結したブローカ。

コネクション (connection)　1) MQ メッセージサーバへのアクティブコネクション。キューコネクション、またはトピックコネクションのどちらかである。2) メッセージをプロデュースおよびコンシュームするために、MQ メッセージサーバの基礎となるコネクションを使用するセッションのファクトリ。

コネクションファクトリ (connection factory)　MQ メッセージサーバへのコネクションを作成するために、クライアントが使用する管理対象オブジェクト。QueueConnectionFactory オブジェクトか、または TopicConnectionFactory オブジェクトのどちらかである。

コンシューマ (consumer)　送信先からメッセージを受信するために使用するセッションによって作成されるオブジェクト (MessageConsumer)。ポイントツーポイント配信モデルの場合、コンシューマは受信側、またはブラウザ (QueueReceiver または QueueBrowser) のどちらかであり、パブリッシュ / サブスクライブ配信モデルの場合、コンシューマはサブスクライバ (TopicSubscriber) である。

コンシューム (consume)　送信先から取得するメッセージをメッセージコンシューマが受信すること。

承認 (authorization)　コネクションサービス、送信先などのメッセージサービスのリソースに、ユーザがアクセスできるかどうかをメッセージサービスが判断するプロセス。

セッション (session)　メッセージを送受信するためのシングルスレッドのコンテキスト。キューセッション、またはトピックセッションのどちらかである。

設定ファイル (configuration file)　ブローカを設定するのに使用する MQ の設定が含まれた 1 つ以上のテキストファイル。プロパティは、インスタンスに固有である場合とクラスタに関連する場合がある。

送信先 (destination)　コンシューマへのルートおよび後続の配信のために、プロデュースされたメッセージが配信される MQ メッセージサーバの物理的な送信先。この物理的な送信先は、プロデュースするメッセージの送信先とコンシュームするメッセージのソースを指定するためにクライアントアプリケーションが使用する管理対象オブジェクトによって、識別およびカプセル化される。

データストア (data store)　ブローカに必要な情報 (永続サブスクリプション、送信先のデータ、持続メッセージ、および監査データ) が恒久的に格納されるデータベース。

トピック (topic)　パブリッシュ / サブスクライブ配信モデルを実装するために、管理者が作成するオブジェクト。トピックは、アドレス指定されたメッセージの収集および配信を担うコンテンツ階層のノードとして表示できる。中間段階としてトピックを使用することにより、メッセージパブリッシャがメッセージサブスクライバから分離される。

ドメイン (domain)　JMS メッセージング処理をプログラミングするために、クライアントアプリケーションが使用するオブジェクトの集まり。ポイントツーポイント配信モデル用とパブリッシュ / サブスクライブ配信モデル用の 2 つのプログラミングドメインがある。

トランザクション (transaction)　不可分な単位の作業。この作業は完了されるか、あるいは完全にロールバックされる必要がある。

配信ポリシー (delivery policy)　複数のメッセージコンシューマを登録した場合に、キューがメッセージをルートする方法の指定。シングル、フェイルオーバー、およびラウンドロビンのポリシーがある。

配信モード (delivery mode)　メッセージングの信頼性のインジケータ。必ず 1 回 (1 回に限って) 配信され確実にコンシュームされることを保証する (持続配信モード) 、あるいはメッセージが 1 回は配信されることを保証する (非持続配信モード) かのどちらかを示す。

配信モデル (delivery model)　メッセージが配信されるモデル。ポイントツーポイント、またはパブリック / サブスクライブのどちらかである。JMS の場合、クライアントランタイムのオブジェクトと送信先タイプ (キューまたはトピック) によって、配信モデルごとにプログラミングドメインがある。

パブリッシュ / サブスクライブ配信モデル (publish/subscribe delivery model)　通常、パブリッシャとサブスクライバは匿名であり、トピックを動的にパブリッシュまたはサブスクライブできる。システムは、トピックの複数のパブリッシャから到着するメッセージを複数のサブスクライバに配信する。

非同期通信 (asynchronous communication)　メッセージの送信側がほかの作業を続ける前に、送信メソッドが戻るのを待つ必要がない通信のモード。

ブローカ (broker)　メッセージのルート、配信、持続性、セキュリティ、およびログ記録を管理し、管理者がパフォーマンスとリソース使用の監視および調整を実行できるインタフェースを提供する MQ のエンティティ。

プロデューサ (producer)　送信先にメッセージを送信するために使用するセッションによって作成されるオブジェクト (MessageProducer)。ポイントツーポイント配信モデルの場合、プロデューサは送信側 (QueueSender) になり、パブリッシュ / サブスクライブ配信モデルの場合、プロデューサはパブリッシャ (TopicPublisher) になる

プロデュース (produce)　メッセージを送信先に配信するために、クライアントランタイムにメッセージを送信すること。

ポイントツーポイント配信モデル (point-to-point delivery model)　プロデューサがメッセージのアドレスを特定のキューに指定し、コンシューマがメッセージを保持するために確立されるキューから、メッセージを取り出す。メッセージは、1 つのメッセージコンシューマにだけ配信される。

メッセージ (messages)　クライアントアプリケーションがコンシュームする非同期要求、レポート、またはイベント。これらのメッセージには、クライアントアプリケーションを調整するために必要な情報が含まれている。メッセージは、ヘッダーと本体から構成されており、ヘッダーにはフィールドを追加できる。メッセージのヘッダーでは、標準フィールドとオプションのプロパティを指定する。送信中のデータはメッセージ本体に含まれる。

メッセージサービス (message service)　「MQ メッセージサーバ」を参照。

メッセージセレクタ (message selector)　JMS メッセージヘッダーのプロパティ値 (セレクタ) に基づいて、メッセージを選択するコンシューマのための手段。メッセージサービスは、メッセージセレクタに配置される条件に基づいて、メッセージのフィルタリングやルートを行う。

メッセージング (messaging)　エンタープライズアプリケーションで使用される非同期要求、レポート、またはイベントのシステム。これにより、弱い連結のアプリケーションが情報を確実かつ安全に送信できる。

ユーザグループ (user group)　コネクションや送信先などの MQ メッセージサーバのリソースへのアクセスを承認するために、クライアントアプリケーションのユーザが属するグループ。

索引

A

admin コネクションサービス , 48, 151

C

config.properties ファイル , 116

E

Enterprise Edition, 27

H

HTTP, 48, 151
httpjms コネクションサービス , 48, 151
httpsjms コネクションサービス , 48, 151
HTTPS コネクション
  サポート , 215
  トンネルサーブレット、「HTTPS トンネルサーブレット」を参照
  複数ブローカ , 231
HTTPS コネクション要求間隔 , 229
HTTPS サポートのアーキテクチャ , 215
HTTPS トンネルサーブレット , 204, 215
HTTP コネクション
  サポート , 215
  トンネルサーブレット、「HTTP トンネルサーブレット」を参照
  複数ブローカ , 220
  要求間隔 , 219
HTTP サポートのアーキテクチャ , 215
HTTP 転送ドライバ , 215
HTTP トンネルサーブレット , 215, 221
HTTP プロキシ , 215

I

imq.accesscontrol.enabled プロパティ , 63, 118
imq.accesscontrol.file.filename プロパティ , 63, 118
imq.authentication.basic.user_repository プロパティ , 62, 118
imq.authentication.client.response.timeout プロパティ , 62, 118
imq.authentication.type プロパティ , 62, 118
imq.autocreate.queue プロパティ , 70, 118, 148
imq.autocreate.topic プロパティ , 70, 118, 149
imq.cluster.brokerlist プロパティ , 74, 126
imq.cluster.hostname プロパティ , 75, 127
imq.cluster.masterbroker プロパティ , 75, 126
imq.cluster.port プロパティ , 75, 127
imq.cluster.url プロパティ , 75, 118, 127, 149
imq.httpjms.http.pullPeriod プロパティ , 219

imq.httpjms.http.servletHost プロパティ，219
imq.httpjms.http.servletPort プロパティ，219
imq.httpsjms.https.pullPeriod プロパティ，229
imq.httpsjms.https.servletHost プロパティ，228
imq.httpsjms.https.servletPort プロパティ，228
imq.keystore.file.dirpath プロパティ，201
imq.keystore.file.name プロパティ，201
imq.keystore.password プロパティ，201, 205
imq.log.console.output.stream プロパティ，66, 118
imq.log.console.output プロパティ，66, 118
imq.log.console.syslog.facility プロパティ，67, 119
imq.log.console.syslog.identity プロパティ，67, 119
imq.log.console.syslog.logconsole プロパティ，67, 119
imq.log.console.syslog.logpid プロパティ，67, 119
imq.log.console.syslog.output プロパティ，67, 119
imq.log.file.dirpath プロパティ，66, 119
imq.log.file.filename プロパティ，66, 119
imq.log.file.output プロパティ，66, 119
imq.log.file.rolloverbytes プロパティ，66, 119, 149
imq.log.file.rolloversecs プロパティ，66, 119, 149
imq.log.level プロパティ，66, 119, 149
imq.message.expiration.interval プロパティ，55, 119
imq.message.max_size プロパティ，55, 119, 149
imq.metrics.enabled プロパティ，66, 119
imq.metrics.interval プロパティ，66, 119
imq.passfile.dirpath プロパティ，63, 119
imq.passfile.enabled プロパティ，63, 119
imq.passfile.name プロパティ，63, 119
imq.persist.file.destination.file.size プロパティ，58, 119
imq.persist.file.message.cleanup プロパティ，59, 119
imq.persist.file.message.fdpool.limit プロパティ，59, 120
imq.persist.file.message.filepool.cleanratio プロパティ，59, 120
imq.persist.file.message.filepool.limit プロパティ，58, 120
imq.persist.file.sync.enabled プロパティ，59, 120
imq.persist.jdbc.brokerid プロパティ，209
imq.persist.jdbc.createdburl プロパティ，210
imq.persist.jdbc.driver プロパティ，209
imq.persist.jdbc.opendburl プロパティ，209
imq.persist.jdbc.password プロパティ，205, 210
imq.persist.store プロパティ，58, 120, 209
imq.portmapper.port プロパティ，51, 120, 149
imq.queue.deliverypolicy プロパティ，70, 120, 149
imq.redelivered.optimization プロパティ，55, 120
imq.*resource_state*.threshold プロパティ，55, 120
imq.service.activelist プロパティ，50, 120
imq.*service_name*.accesscontrol.enabled プロパティ，63, 120
imq.*service_name*.accesscontrol.file.filename プロパティ，63, 120
imq.*service_name*.authentication.type プロパティ，62, 120
imq.*service_name*.max_threads プロパティ，50, 120
imq.*service_name*.min_threads プロパティ，50, 121
imq.*service_name.protocol_type*.hostname プロパティ，51, 121
imq.*service_name.protocol_type*.port プロパティ，51, 121
imq.*service_name*.threadpool_model プロパティ，51, 121
imq.shared.connectionMonitor_limit プロパティ，51, 121
imq.system.max_count プロパティ，55, 121, 149
imq.system.max_size プロパティ，55, 121, 149
imq.transaction.autorollback プロパティ，55, 121, 162
imq.user_repository.ldap.base プロパティ，191
imq.user_repository.ldap.gidattr プロパティ，191
imq.user_repository.ldap.grpbase プロパティ，191
imq.user_repository.ldap.grpfiltler プロパティ，192
imq.user_repository.ldap.grpsearch プロパティ，191

imq.user_repository.ldap.memattr プロパティ , 191
imq.user_repository.ldap.password プロパティ , 191, 205
imq.user_repository.ldap.principal プロパティ , 190
imq.user_repository.ldap.server プロパティ , 190
imq.user_repository.ldap.ssl.enabled プロパティ , 192
imq.user_repository.ldap.timeout プロパティ , 192
imq.user_repository.ldap.uidattr プロパティ , 191
imq.user_repository.ldap.usrfilter プロパティ , 191
IMQ_HOME 環境変数 , 20
IMQ_JAVAHOME 環境変数 , 21
IMQ_VARHOME 環境変数 , 21
imqAckOnAcknowledge 属性 , 169
imqAckOnProduce 属性 , 169
imqAckTimeout 属性 , 169
imqBrokerHostName 属性 , 169
imqBrokerHostPort 属性 , 169
imqBrokerServicePort 属性 , 169
imqbroker コマンド
  オプション , 122
  使用 , 122
imqcmd コマンド
  オプション , 142
  概要 , 87
  使用 , 140, 165
  送信先の管理 , 154
  トランザクション管理 , 160
  フォーマット , 140, 165
  ブローカへの安全なコネクション , 143, 203
  ブローカへの接続 , 144
  要約 , 140
imqConfiguredClientID 属性 , 169
imqConnectionType 属性 , 169
imqConnectionURL 属性 , 169
imqdbmgr コマンド
  オプション , 213
  概要 , 87
imqDefaultPassword 属性 , 169
imqDefaultUsername 属性 , 169
imqDestinationDescription 属性 , 81, 170
imqDestinationName 属性 , 81, 170
imqDisableSetClientID 属性 , 169
imqFlowControlCount 属性 , 169
imqFlowControlIsLimited 属性 , 169
imqFlowControlLimit 属性 , 169
imqJMSDeliveryMode 属性 , 170
imqJMSExpiration 属性 , 170
imqJMSPriority 属性 , 170
imqkeytool コマンド
  概要 , 87
  使用 , 200, 225
imqLoadMaxToServerSession 属性 , 169
imqobjmgr コマンド
  オプション , 166
  概要 , 87
  要約 , 165
imqOverrideJMSDeliveryMode 属性 , 170
imqOverrideJMSExpiration 属性 , 170
imqOverrideJMSHeadersToTemporaryDestinations 属性 , 170
imqOverrideJMSPriority 属性 , 170
imqQueueBrowserMaxMessagesPerRetrieve 属性 , 169
imqQueueBrowserRetrieveTimeout 属性 , 169
imqReconnectDelay 属性 , 169
imqReconnectRetries 属性 , 169
imqReconnect 属性 , 169
imqSetJMSXAppID 属性 , 169
imqSetJMSXConsumerTXID 属性 , 169
imqSetJMSXProducerTXID 属性 , 169
imqSetJMSXRcvTimestamp 属性 , 169
imqSetJMSXUserID 属性 , 169
imqSSLIsHostTrusted 属性 , 169
imqsvcadmin コマンド , 87, 237
  オプション , 239
  使用 , 238
  フォーマット , 238
  要約 , 238

imqusermgr コマンド
オプション , 185
概要 , 87
サブコマンド , 185
パスワード , 187
ユーザ名 , 187

J

J2EE アプリケーション
EJB 仕様 , 35
JMS, 35
メッセージ駆動型 Beans、「メッセージ駆動型 Beans」を参照
JDBC サポート
概要 , 58
設定 , 207
ドライバ , 207, 209
JDK
パスの指定 , 123
パスの指定オプション , 142, 166, 239
JMS
仕様 , 32
プログラミングモデル , 32
メッセージ構造 , 32
jms コネクションサービス , 48, 151
JMS 仕様書 , 25, 23
JNDI
LDAP サーバ , 190
MQ サポート , 26
オブジェクトストア , 87, 164, 168
管理対象オブジェクト , 37
検索 , 78, 81, 105, 167
コネクションファクトリ検索 , 34
メッセージ駆動型 Beans, 36
JRE、パスの指定 , 123

L

LDAP サーバアクセス , 190

M

MDB、「メッセージ駆動型 Beans」を参照

P

passfile
broker configuration プロパティ , 63
コマンド行オプション , 124
使用 , 205
場所 , 205, 244
Platform Edition, 26

S

SOAP, 26
SSL
HTTP 経由 , 204
TCP/IP 経由 , 199
暗号化 , 199
概要 , 61
コネクションサービス , 48, 151
サービス、設定 , 183
ssladmin コネクションサービス , 48, 151
設定 , 199, 202
ssljms コネクションサービス , 48, 151
設定 , 199, 202
SSL 標準、「SSL」を参照
SSL ベースのコネクションサービス
設定 , 199
SSL ベースのサービス
起動 , 202
syslog, 65, 134

T

TCP, 48, 151
TLS, 48, 151

W

Windows のサービス、ブローカの実行 , 237

X

XA コネクションファクトリ
  概要 , 42
  「コネクションファクトリ管理対象オブジェクト」も参照
XA リソースマネージャ、「分散トランザクション」を参照

あ

アクセス規則 , 195
アクセス権
  admin サービス , 61
  MQ 操作 , 60
  passfile, 205
  アクセス制御プロパティファイル , 60, 193
  キーストア , 225
  組み込みデータベース , 212
  計算 , 195
  データストア , 58
  ユーザリポジトリ , 185
アクセス制御ファイル
  アクセス規則 , 195
  使用 , 193
  バージョン , 193
  場所 , 193, 243
  フォーマット , 194
アクセス制御プロパティファイル、「アクセス制御ファイル」を参照
アプリケーションサーバ , 37
暗号化
  SSL ベースのサービス , 199
  概要 , 61
  キーツール , 62

い

移植性、「プロバイダへの非依存性」を参照
一時送信先 , 70, 156

え

永続サブスクライバ、「永続サブスクリプション」を参照
永続サブスクリプション
  一覧表示 , 158
  概要 , 38
  クライアント ID, 39
  破棄 , 158
  メッセージの削除 , 158
エディション、製品
  概要 , 26
  企業向け , 27
  プラットフォーム , 26

お

オブジェクトストア
  LDAP サーバ , 164
  概要 , 164
  属性 , 171
  場所 , 243
  ファイルシステムストア , 164

か

環境変数
IMQ_HOME, 20
IMQ_JAVAHOME, 21
IMQ_VARHOME, 21
管理対象オブジェクト
XA コネクションファクトリ、「コネクションファクトリ管理対象オブジェクト」を参照
一覧表示 , 179
オブジェクトストア、「オブジェクトストア」を参照
概要 , 34, 78
キュー、「キュー」を参照
クエリ , 180
更新 , 181
コネクションファクトリ、「コネクションファクトリ管理対象オブジェクト」を参照
削除 , 178
種類 , 34, 78, 165
送信先、「送信先管理対象オブジェクト」を参照
属性 , 168
トピック、「トピック」を参照
名前の検索 , 166
必要な情報 , 167
プロバイダへの非依存性 , 79
管理タスク
開発環境 , 83
本稼働環境 , 84
管理ツール
概要 , 85
管理コンソール , 85
コマンド行ユーティリティ , 86

き

キーストア
ファイル , 200, 201, 226
プロパティ , 201
キーツール , 62
キーの組み合わせ
再生成 , 201
生成 , 200
キュー , 68
管理対象オブジェクトの追加 , 177
自動作成 , 70, 118
配信ポリシー、「キュー配信ポリシー」を参照
キュー送信先、「キュー」を参照
キュー配信ポリシー
概要 , 68
シングル , 68
属性 , 155
フェイルオーバー , 68
ラウンドロビン , 68

く

組み込み持続 , 57
クライアント
アプリケーション、「クライアントアプリケーション」を参照
識別子 (クライアント ID), 39
プログラミングモデル , 32
ランタイム , 76
クライアントアプリケーション
システムのプロパティ , 81
プロバイダへの非依存性 , 37
クラスタ設定ファイル , 74
クラスタ、「ブローカクラスタ」を参照

こ

コネクション
概要 , 33
再接続 , 169
再接続試行 , 169
再接続遅延 , 169
コネクションサービス
admin, 48, 151
HTTP、「HTTP コネクション」を参照
httpjms, 48, 151
HTTPS、「HTTPS コネクション」を参照
httpsjms, 48, 151

jms, 48, 151
ssladmin, 48, 151, 199
ssljms, 48, 151, 199
SSL ベース , 201
アクセス制御 , 63
概要 , 47
起動時にアクティブ化 , 50
クエリ , 150, 152
更新 , 151, 152
コネクションタイプ , 48
コマンドの影響 , 150
サービスタイプ , 48
再開 , 151, 154
スレッドの割り当て , 153
スレッドプールマネージャ , 49
静的ポート , 51
停止 , 151, 154
プロパティ , 50
ポートマッパー、「ポートマッパー」を参照
コネクションファクトリ管理対象オブジェクト
JNDI 検索 , 34
オーバーライド , 81
概要 , 33, 80
クライアント ID, 39
属性 , 80, 168, 169
追加 , 175
コマンド行の構文 , 88
コマンドギョウユーティリティ
imqdbmgr, 87
imqkeytool, 87
imqsvcadmin, 87
コマンド行ユーティリティ
imqcmd, 87, 140
imqobjmgr, 87, 165
imqsvcadmin, 237, 238
imqusermgr, 87
概要 , 86
基本構文 , 88
共通するオプション , 88
コマンドのオプション , 88
コンシューマ , 33
コンテナ
EJB, 36
MDB, 36
コントロールメッセージ , 52
コンポーネント
EJB, 35
MDB, 35

さ

サービスタイプ
ADMIN, 48
NORMAL, 48
再配信フラグ , 53
サブスクリプション
永続サブスクリプションの管理 , 158
永続サブスクリプションの破棄 , 159
永続的 ID, 142

し

自己署名型証明書 , 200, 225
システムのプロパティ、設定 , 81
持続
JDBC、「JDBC 持続」を参照
組み込み , 57
持続マネージャ、「持続マネージャ」を参照
データストアのリセット , 125
配信モード、「配信モード」を参照
プラグイン、「プラグイン持続」を参照
持続マネージャ
JDBC データストア , 209
概要 , 56
データストア、「データストア」を参照
プラグイン持続 , 207
ブローカのコンポーネント , 48
プロパティ , 58
持続メッセージ , 40
承認
「アクセス制御ファイル」も参照
概要 , 60
管理 , 193

ユーザグループ , 61
証明書 , 200, 225
シンプルオブジェクトアクセスプロトコル、「SOAP」を参照
信頼性の高い配信 , 40

す

スレッドプールマネージャ
概要 , 49
共有スレッド , 50
専用スレッド , 50

せ

セキュリティ
暗号化、「暗号化」を参照
オブジェクトストア , 171
承認、「承認」を参照
認証、「認証」を参照
マネージャ、「セキュリティマネージャ」を参照
セキュリティマネージャ
概要 , 59
ブローカのコンポーネント , 48
プロパティ , 62
セッション
概要 , 33
処理済み , 40
通知オプション , 40
設定ファイル
config.properties, 116
インスタンス , 116, 127, 148, 243
インストール , 116
設定された値のオーバーライド , 116
デフォルト , 116
場所 , 116, 243
編集 , 118
設定変更記録 , 73

そ

送信先
アクセス制御 , 197
一時 , 70, 156
一覧表示 , 154
概要 , 46
管理 , 154
キュー、「キュー」を参照
作成 , 155
自動作成 , 69, 198
情報 , 154, 156
属性 , 155
属性の更新 , 155, 156
トピック、「トピック」を参照
破棄 , 154, 155
物理的 , 67
メッセージの削除 , 154, 157
送信先管理対象オブジェクト , 81
概要 , 33
属性 , 170
送信先の自動作成
概要 , 69
プロパティ , 70

つ

通知
概要 , 40, 52
クライアント , 53
待機時間 , 169
トランザクション , 54
配信 , 53
ブローカ , 53, 169
ツール、管理、「管理ツール」を参照

て

データ、MQ、場所 , 243
データストア

JDBC アクセス可能 , 58
概要 , 56
単層型ファイル , 57
場所 , 243
リセット , 125
転送、「メッセージルーター」を参照

と

トピック
概要 , 38
管理対象オブジェクトの追加 , 177
自動作成 , 70, 118
物理的な送信先 , 69
トピック送信先、「トピック」を参照
ドメイン , 38
トランザクション
概要 , 40
管理 , 160
コミット , 160
情報 , 160
通知 , 54
分散、「分散トランザクション」を参照
ロールバック , 160
トランスポートプロトコル
HTTP, 48, 151
TCP, 48, 151
TLS, 48, 151

に

入力ファイル , 172
認証
概要 , 60
管理 , 184

は

配信、高信頼性 , 40
配信モード
持続 , 40
非持続的 , 40
パスワード
JDBC, 205
LDAP, 205
passfile、「passfile」を参照
SSL キーストア , 124, 201, 205
デフォルト , 169
符号化 , 62
命名規則 , 187
パスワードファイル、「passfile」を参照
パフォーマンス、信頼性 , 42
パブリッシュ / サブスクライブ配信 , 38

ふ

ファイアウォール , 215
プラグイン持続
概要 , 58
設定 , 207
ブローカ
httpjms コネクションサービスのプロパティ , 219
httpsjms コネクションサービスのプロパティ , 228
HTTPS、サポート , 225
HTTP サポート , 217
JDBC サポート、「JDBC サポート」を参照
SSL ベースのサービスの起動 , 202
Windows のサービス、実行 , 237
アクセス制御、「承認」を参照
インスタンス設定プロパティ , 118
インスタンス名 , 124
概要 , 46
起動 , 122
クエリ , 146, 147
クラスタ、「ブローカクラスタ」を参照
更新 , 146

コネクションサービス、「コネクションサービス」を参照
サービスの一覧表示 , 152
再開 , 146
再起動 , 56, 146
システムリソース , 54
持続マネージャ、「持続マネージャ」を参照
シャットダウン , 146
障害からの復元 , 56
制御 , 145
セキュリティマネージャ、「セキュリティマネージャ」を参照
接続 , 128, 144
設定ファイル、「設定ファイル」を参照
送信先の自動作成のプロパティ , 70
タスク , 47
通知 (ACK), 53
停止 , 146
プロパティ , 148
マスターブローカ , 73
マルチブローカクラスタ、「ブローカクラスタ」を参照
メッセージ転送、「メッセージルーター」を参照
メッセージの容量 , 55
メトリックス、「メトリックス」を参照
連結、「ブローカのクラスタ」を参照
ログ作成、「ロガー」を参照
ブローカクラスタ
アーキテクチャ , 71
開発専用環境 , 74
クラスタ設定ファイル , 74, 127
指定するオプション , 123
使用する理由 , 71
情報の伝播 , 72
設定プロパティ , 74, 126
設定変更記録 , 73
ブローカの再起動 , 130
ブローカの接続 , 128
ブローカの追加 , 129
プロパティの設定 , 127
マスターブローカ , 73, 74
プログラミングドメイン , 38
プロデューサ , 33
プロトコルタイプ
HTTP, 48, 151
TCP, 48, 151
TLS, 48, 151
プロトコル、「トランスポートレイヤ」を参照
プロバイダへの非依存性
概要 , 37
管理対象オブジェクト , 79
プロパティ
httpjms コネクションサービス , 219
httpsjms コネクションサービス , 228
JDBC 関連 , 209
LDAP 関連 , 190
キーストア , 201
クラスタ設定 , 74, 126
コネクションサービス , 50
持続 , 58
自動作成 , 70
セキュリティ , 62
ブローカインスタンス設定 , 118
ブローカの更新 , 148
メッセージルーター , 55
ロガー , 66
分散トランザクション
「XA コネクションファクトリ」も参照
XA リソースマネージャ , 41, 160
概要 , 41

ほ

ポイントツーポイント配信 , 38
ポート、動的な割り当て , 49
ポートマッパ
概要 , 49
ポートの割り当て , 51, 124

ま

マスターブローカ , 73, 74

め

メッセージ
  SOAP, 26
  概要 , 32
  構造 , 32
  コンシューム , 77
  コントロール , 52
  再配信 , 53
  持続 , 54, 56
  持続的 , 40
  順序付け , 43
  信頼性の高い配信 , 40
  制限 , 155
  送信先で削除 , 154
  通知 , 53, 169
  転送および配信 , 52
  配信モード、「配信モード」を参照
  配信モデル , 31, 38
  パブリッシュ / サブスクライブ配信 , 38
  フィルタリング , 43
  ブローカの制限 , 55
  プロデュース , 77
  ポイントツーポイント配信 , 38
  有効期限の回復 , 55
  優先度の決定 , 43
  リスナ , 34, 77
メッセージ駆動型 Beans
  MDB コンテナ , 36
  アプリケーションサーバのサポート , 37
  概要 , 35
  配置記述子 , 36
メッセージコンシューマ、「コンシューマ」を参照
メッセージサーバ
  概要 , 46
  複数ブローカ、「ブローカのクラスタ」を参照 , 71
メッセージサービス , 30
メッセージサービスにメッセージを送信するメッセージプロデューサを示す図 , 31
メッセージサービスの , 31
メッセージサービスはメッセージコンシューマにメッセージを中継する , 31
メッセージ配信モデル , 31, 38
メッセージプロデューサ、「プロデューサ」を参照
メッセージリスナ、「リスナ」を参照
メッセージルーター
  概要 , 52
  ブローカのコンポーネント , 47
  プロパティ , 55
メッセージングシステム
  MQ アーキテクチャ , 46
  アーキテクチャ , 30
  メッセージサービス , 30
メトリックス
  概要 , 64
  報告の間隔 , 124
  要約 , 136
メモリ管理 , 54

ゆ

ユーザグループ
  概要 , 60
  定義済み , 186
  デフォルト , 61
  割り当ての削除 , 187
ユーザ名 , 169, 187
ユーザリポジトリ
  LDAP サーバ , 190
  概要 , 59
  管理 , 188
  種類 , 62
  設定 , 188
  単層型ファイル , 184
  場所 , 184, 243
  プラットフォーム依存 , 185
  ユーザグループ , 187
  ユーザの状態 , 187

ら

ライセンス
MQ エディション用 , 26
読み込み , 124

り

リスナ , 34, 35

ろ

ロガー
アーカイブファイル , 65
概要 , 64
カテゴリ , 64
コンソールへの書き込み , 66, 125
出力チャネル , 65, 134
設定の変更 , 133
デフォルトの設定 , 132
ブローカのコンポーネント , 48
プロパティ , 66
メッセージの書式設定 , 133
メトリックス情報 , 66, 135
レベル , 64, 66, 124
ロールオーバー条件 , 135
ログメッセージのリダイレクト , 135
ログ作成、「ロガー」を参照
ログファイル
デフォルトの場所 , 65, 243
ロールオーバー条件 , 66