



Sun Java™ System

Message Queue 3.5

管理ガイド

---

Service Pack 1

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

Part No: 817-7205

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. は、この製品に含まれるテクノロジー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。

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

Sun、Sun Microsystems、Sun のロゴマーク、Java、Solaris、JDK、Java Naming and Directory Interface、Javadoc、JavaMail、JavaHelp、Java Coffee Cup のロゴ、Sun[tm] ONE および Sun[tm] ONE 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 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標が付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャに基づくものです。

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。

この製品は、米国の輸出規制に関する法規の適用および管理下であり、また、米国以外の国の輸出および輸入規制に関する法規の制限を受ける場合があります。核、ミサイル、生物化学兵器もしくは原子力船に関連した使用またはかかる使用者への提供は、直接的にも間接的にも、禁止されています。このソフトウェアを、米国の輸出禁止国へ輸出または再輸出すること、および米国輸出制限対象リスト ( 輸出が禁止されている個人リスト、特別に指定された国籍者リストを含む ) に指定された、法人、または団体に輸出または再輸出することは一切禁止されています。

# 目次

図目次 .....	15
表目次 .....	17
手順一覧 .....	21
はじめに .....	23
マニュアルの対象読者 .....	23
マニュアルの構成 .....	24
マニュアルの表記規則 .....	26
テキストの表記規則 .....	26
ディレクトリ変数の表記規則 .....	26
関連マニュアル .....	29
Message Queue マニュアルセット .....	29
オンラインヘルプ .....	30
JavaDoc .....	30
クライアントアプリケーション例 .....	30
Java Message Service (JMS) 仕様書 .....	30
関連するサードパーティの Web サイトのリファレンス .....	31
 第 1 章 概要 .....	 33
Sun Java System Message Queue とは .....	33
製品エディション .....	35
Platform Edition .....	35
Enterprise Edition .....	36
エンタープライズメッセージングシステム .....	36
エンタープライズメッセージングシステムの要件 .....	36
集中メッセージングとピアツーピアメッセージング .....	37

メッセージングシステムの概念 .....	38
メッセージ .....	38
メッセージサービスのアーキテクチャ .....	38
メッセージ配信モデル .....	39
JMS 仕様 .....	40
JMS メッセージ構造 .....	40
JMS プログラミングモデル .....	40
JMS 管理対象オブジェクト .....	42
JMS/J2EE プログラミング: メッセージ駆動型 Beans .....	43
メッセージ駆動型 Beans .....	43
J2EE アプリケーションサーバのサポート .....	45
JMS メッセージングの問題点 .....	45
JMS プロバイダへの非依存性 .....	45
プログラミングドメイン .....	46
クライアント識別子 .....	47
信頼性のあるメッセージング .....	48
通知 / トランザクション .....	49
持続ストレージ .....	50
パフォーマンスのかね合い .....	51
メッセージの選択 .....	51
メッセージの順番と優先度 .....	52
 <b>第 2 章 Message Queue メッセージングシステム .....</b>	<b>53</b>
Message Queue メッセージサーバ .....	54
ブローカ .....	54
コネクションサービス .....	56
メッセージルーター .....	61
持続マネージャ .....	66
セキュリティマネージャ .....	69
監視サービス .....	74
物理的な送信先 .....	80
キューの送信先 .....	80
トピックの送信先 .....	82
自動作成 ( および管理者によって作成 ) された送信先 .....	82
一時送信先 .....	85
マルチブローカクラスタ (Enterprise Edition) .....	86
マルチブローカのアーキテクチャ .....	86
開発環境でのクラスタの使用 .....	89
クラスタ設定プロパティ .....	90
Message Queue クライアントランタイム .....	90
メッセージのプロデュース .....	91
メッセージのコンシューム .....	92
Message Queue 管理対象オブジェクト .....	93

コネクションファクトリ管理対象オブジェクト .....	94
送信先管理対象オブジェクト .....	95
クライアントの起動時の属性値のオーバーライド .....	96
<b>第3章 Message Queue の管理タスクと管理ツール .....</b>	<b>97</b>
Message Queue の管理タスク .....	97
開発環境 .....	97
運用環境 .....	98
セットアップ操作 .....	98
運用環境を設定するには .....	98
メンテナンス操作 .....	99
運用環境を設定するには .....	99
Message Queue 管理ツール .....	100
管理コンソール .....	100
コマンド行ユーティリティの概要 .....	101
コマンド行の構文 .....	102
共通のコマンド行オプション .....	103
<b>第4章 管理コンソールのチュートリアル .....</b>	<b>105</b>
準備 .....	106
管理コンソールを起動する .....	106
管理コンソールを起動するには .....	106
ヘルプを利用する .....	108
管理コンソールのヘルプ情報を表示するには .....	108
ブローカを操作する .....	109
ブローカの起動 .....	110
ブローカを起動するには .....	110
ブローカの追加 .....	111
管理コンソールにブローカを追加するには .....	111
管理者パスワードの変更 .....	112
管理者パスワードを変更するには .....	112
ブローカに接続する .....	113
ブローカに接続するには .....	113
コネクションサービスを表示する .....	114
利用可能なコネクションサービスを表示するには .....	114
ブローカに物理的送信先を追加する .....	116
ブローカにキュー送信先を追加するには .....	116
物理的送信先を操作する .....	118
物理的送信先のプロパティを表示するには .....	118
送信先からメッセージをページするには .....	119
送信先を削除するには .....	120
トピック送信先に関する情報を取得する .....	120

オブジェクトストアを操作する	121
オブジェクトストアを追加する	121
ファイルシステムオブジェクトストアを追加するには	121
オブジェクトストアプロパティを確認する	124
オブジェクトストアのプロパティを表示するには	124
オブジェクトストアに接続する	124
オブジェクトストアに接続するには	124
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	125
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	125
送信先管理対象オブジェクトを追加する	127
送信先をオブジェクトストアに追加するには	127
管理対象オブジェクトのプロパティ	129
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	129
コンソール情報を更新する	130
サンプルアプリケーションを実行する	130
HelloWorldMessageJNDI アプリケーションを実行するには	130
 <b>第 5 章 ブローカの起動と設定</b>	 <b>133</b>
設定ファイル	133
インスタンス設定ファイル	134
プロパティ値のマージ	134
プロパティ命名構文	135
インスタンス設定ファイルの編集	136
ブローカの起動	141
imqbrokerd コマンドの構文	141
起動の例	142
デフォルトのブローカ名と設定を使用するブローカインスタンスを起動するには	142
Enterprise Edition のトライアルライセンスでブローカインスタンスを起動するには	142
プラグイン持続で名前付きブローカインスタンスを起動するには	142
imqbrokerd オプションの概要	143
クラスタの操作 (Enterprise Edition)	147
クラスタ設定プロパティ	147
ブローカの接続	150
接続方法	150
ブローカをクラスタに接続するには	150
ブローカ間の安全なコネクション	150
クラスタで安全なコネクションを設定するには	151
クラスタ内のブローカの管理	151
クラスタへのブローカの追加	151
新しいブローカを既存のクラスタに追加するには	151
クラスタ内のブローカの再起動	151
既存のクラスタのメンバーであるブローカを再起動するには	152
クラスタからのブローカの削除	152

ブローカを既存のクラスタから削除するには	152
マスターブローカの設定変更レコードの管理	153
設定変更レコードのバックアップ	153
設定変更レコードをバックアップするには	153
設定変更レコードの復元	153
障害時にマスターブローカを復元するには	153
ログ作成	154
デフォルトのログ作成の設定	154
ログメッセージの書式設定	155
ロガー設定の変更	155
ブローカのロガー設定を変更するには	155
出力チャンネルの変更	156
ログファイルのロールオーバー条件の変更	157
<b>第 6 章 ブローカとアプリケーションの管理</b>	<b>159</b>
コマンドユーティリティ	160
imqcmd コマンドの構文	160
imqcmd のサブコマンド	160
imqcmd のオプションの概要	162
imqcmd コマンドの使用	164
imqcmd の使用例	165
ブローカの管理	165
ブローカ情報の表示	167
ブローカのプロパティの更新	168
ブローカの状態の制御	169
ブローカの停止および再開	169
ブローカのシャットダウンと再起動	170
ブローカのメトリックスの表示	170
コネクションサービスの管理	171
コネクションサービスの一覧表示	173
コネクションサービス情報の表示	174
コネクションサービスのプロパティの更新	174
コネクションサービスのメトリックスの表示	175
コネクションサービスの停止および再開	175
コネクション情報の取得	176
送信先の管理	177
送信先の作成	179
定義の一覧表示	182
送信先情報の表示	183
送信先の属性の更新	184
送信先のメトリックスの表示	184
送信先の停止と再開	185
送信先のページ	185

送信先の破棄 .....	186
送信先の圧縮 .....	186
送信先のディスク利用率の監視 .....	186
未使用の送信先ディスクスペースの再利用 .....	187
未使用の送信先ディスクスペースを再利用するには .....	187
永続サブスクリプションの管理 .....	188
トランザクションの管理 .....	189
 <b>第 7 章 管理対象オブジェクトの管理 .....</b>	<b>193</b>
オブジェクトストアについて .....	194
LDAP サーバオブジェクトストア .....	194
ファイルシステムオブジェクトストア .....	195
管理対象オブジェクト .....	197
コネクションファクトリ管理対象オブジェクトの属性 .....	197
送信先管理対象オブジェクトの属性 .....	199
オブジェクトマネージャユーティリティ (imqobjmgr) .....	200
imqobjmgr コマンドの構文 .....	200
imqobjmgr のサブコマンド .....	200
imqobjmgr コマンドのオプションの概要 .....	201
必要な情報 .....	203
コマンドファイルの使用 .....	204
管理対象オブジェクトの追加および削除 .....	206
コネクションファクトリの追加 .....	206
トピックまたはキューの追加 .....	207
管理対象オブジェクトの削除 .....	209
情報の入手 .....	210
管理対象オブジェクトの一覧表示 .....	210
単一オブジェクトの情報 .....	211
管理対象オブジェクトの更新 .....	211
 <b>第 8 章 セキュリティの管理 .....</b>	<b>213</b>
ユーザーの認証 .....	214
単層型ファイルユーザーリポジトリを使用する .....	214
ユーザーリポジトリの作成 .....	214
ユーザーマネージャユーティリティ (imqusermgr) .....	215
グループ .....	217
状態 .....	218
ユーザー名とパスワードの形式 .....	218
ユーザーリポジトリの設定と管理 .....	219
デフォルトの管理者パスワードの変更 .....	220
ユーザーリポジトリに LDAP サーバを使用する .....	221
設定ファイルを編集して、LDAP サーバを使用するには .....	221



ユーザーの承認：アクセス制御プロパティファイル .....	224
アクセス制御プロパティファイルの作成 .....	224
アクセス規則の構文 .....	225
アクセス権の計算 .....	226
コネクションアクセス制御 .....	227
送信先アクセス制御 .....	228
送信先自動作成アクセス制御 .....	229
暗号化：SSL ベースのサービスとの連動 (Enterprise Edition) .....	230
TCP/IP を介した SSL ベースのサービスの設定 .....	230
SSL ベースのコネクションサービスを設定するには .....	230
手順 1: 自己署名型証明書の生成 .....	231
キーの組み合わせを生成し直すには .....	232
手順 2: ブローカでの SSL ベースのサービスを有効にする .....	233
ブローカで SSL ベースのサービスを有効にするには .....	233
手順 3: ブローカを起動する .....	233
手順 4: SSL ベースのクライアントを設定および実行する .....	234
HTTP を介した SSL サービスの設定 .....	236
passfile の使用 .....	236
 <b>第 9 章 メッセージサービスの分析と調整 .....</b>	<b>237</b>
パフォーマンス関連 .....	237
パフォーマンス調整プロセス .....	237
パフォーマンスのさまざまな側面 .....	238
ベンチマーク .....	239
基準になる使用パターン .....	240
パフォーマンスに影響する要因 .....	241
パフォーマンスに影響するアプリケーション設計の要因 .....	242
配信モード ( 持続的 / 持続性のないメッセージ ) .....	244
トランザクションの使用 .....	245
通知モード .....	246
永続的サブスクリプションと永続的でないサブスクリプション .....	247
セレクトアの使用 ( メッセージのフィルタリング ) .....	248
メッセージのサイズ .....	248
メッセージ本体のタイプ .....	249
パフォーマンスに影響するメッセージサービスの要因 .....	250
ハードウェア .....	250
オペレーティングシステム .....	251
Java 仮想マシン (JVM) .....	251
コネクション .....	251
メッセージサービスのアーキテクチャ .....	253
ブローカの制限と動作 .....	254
データストアのパフォーマンス .....	254
クライアントランタイムの設定 .....	255

メッセージサーバの監視 .....	256
監視ツール .....	256
Message Queue コマンドユーティリティ (imgcmd) .....	256
metrics サブコマンドを使用するには .....	259
Message Queue ブローカログファイル .....	261
ログファイルを使用してメトリックス情報を報告するには .....	262
メッセージベースの監視 API .....	262
メッセージベースの監視を設定するには .....	263
適切な監視ツールの選択 .....	265
メトリックスデータの送信先 .....	267
JVM メトリックス .....	267
ブローカ全体のメトリックス .....	268
コネクションサービスのメトリックス .....	270
送信先メトリックス .....	273
パフォーマンス問題のトラブルシューティング .....	276
問題: クライアントがコネクションを確立できない .....	276
現象: .....	276
考えられる原因: .....	276
問題: コネクションスループットが遅過ぎる .....	280
現象: .....	280
考えられる原因: .....	280
問題: クライアントがメッセージプロデューサを作成できない .....	282
現象: .....	282
考えられる原因: .....	282
問題: メッセージのプロデュースが遅れるまたは低速である .....	283
現象: .....	283
考えられる原因: .....	283
問題: メッセージサーバでメッセージがバックログされる .....	285
現象: .....	285
考えられる原因: .....	286
問題: メッセージサーバのスループットが散発的である .....	290
現象: .....	290
考えられる原因: .....	290
問題: メッセージがコンシューマに到達しない .....	291
現象: .....	291
考えられる原因: .....	291
パフォーマンスを改善するための設定の調整 .....	293
システムの調整 .....	293
Solaris での調整: CPU 使用率、ページング / スワッピング / ディスク I/O .....	293
Java 仮想マシン (JVM) の調整 .....	293
トランスポートプロトコルの調整 .....	294
ファイルベースの持続ストアの調整 .....	297
ブローカの調整 .....	297

メモリ管理：負荷のある状態でブローカの安定性を高める .....	297
複数のコンシューマキューのパフォーマンス .....	298
クライアントランタイムのメッセージフローの調整 .....	299
メッセージフロー測定 .....	299
メッセージフロー制限 .....	300
 <b>付録 A Message Queue データの場所 .....</b>	<b>303</b>
Solaris .....	303
Linux .....	304
Windows .....	306
 <b>付録 B プラグイン持続の設定 .....</b>	<b>309</b>
概要 .....	309
JDBC アクセスが可能なデータストアへの接続 .....	310
JDBC アクセスが可能なデータストアに接続するには .....	310
JDBC 関連のブローカの設定プロパティ .....	311
データベース管理ユーティリティ (imqdbmgr) .....	316
imqdbmgr コマンドの構文 .....	316
imqdbmgr のサブコマンド .....	316
imqdbmgr コマンドのオプションの概要 .....	317
 <b>付録 C HTTP/HTTPS サポート (Enterprise Edition) .....</b>	<b>319</b>
HTTP/HTTPS サポートのアーキテクチャ .....	319
HTTP サポートの有効化 .....	321
HTTP サポートを有効にするには .....	321
手順 1: HTTP トンネルサーブレットを Web サーバに配置する .....	321
jar ファイルとして配置する .....	321
Web アーカイブファイルとして配置する .....	322
手順 2: httpjms コネクションサービスを設定する .....	323
httpjms コネクションサービスをアクティブにするには .....	323
手順 3: HTTP コネクションを設定する .....	325
コネクションファクトリを設定する .....	325
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 .....	325
HTTP プロキシを使用する .....	326
例 1: HTTP トンネルサーブレットを Sun Java System Web サーバに配置する .....	326
jar ファイルとして配置する .....	326
トンネルサーブレットを追加するには .....	327
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 .....	328
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 .....	328
サーバのアクセスログを無効にするには .....	328
WAR ファイルとして配置する .....	328
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 .....	329

例 2: HTTP トンネルサーブレットを Sun Java System Application Server 7.0 に配置する	330
配置ツールを使用する	330
HTTP トンネルサーブレットを Application Server 7.0 環境に配置するには	330
server.policy ファイルを変更する	331
アプリケーションサーバの server.policy ファイルを変更するには	331
HTTPS サポートの有効化	332
HTTPS サポートを有効にするには	332
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	332
手順 2: HTTPS トンネルサーブレットを Web サーバに配置する	333
jar ファイルとして配置する	333
Web アーカイブファイルとして配置する	334
手順 3: httpsjms コネクションサービスを設定する	335
httpsjms コネクションサービスをアクティブにするには	335
手順 4: HTTPS コネクションを設定する	337
JSSE を設定する	337
JSSE を設定するには	337
root 証明書をインポートする	337
コネクションファクトリを設定する	338
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	338
HTTP プロキシを使用する	339
例 3: HTTPS トンネルサーブレットを Sun Java System Web サーバに配置する	339
jar ファイルとして配置する	339
トンネルサーブレットを追加するには	340
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	341
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	341
サーバのアクセスログを無効にするには	342
WAR ファイルとして配置する	342
HTTPS トンネルサーブレット WAR ファイルを修正するには	342
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	343
例 4: HTTPS トンネルサーブレットを Sun Java System Application Server 7.0 に配置する	344
配置ツールを使用する	344
HTTPS トンネルサーブレットを Application Server 7.0 環境に配置するには	344
server.policy ファイルを変更する	345
アプリケーションサーバの server.policy ファイルを変更するには	345
<b>付録 D Windows のサービスとしてのブローカの使用</b>	<b>347</b>
Windows のサービスとしてのブローカの実行	347
サービス管理ユーティリティ (imqsvcadmin)	348
imqsvcadmin コマンドの構文	348
imqsvcadmin のサブコマンド	348
imqsvcadmin のオプションの概要	349
ブローカサービスの削除	350
ブローカサービスの再設定	350

代替 Java ランタイムの使用 .....	350
ブローカサービスのクエリー .....	350
トラブルシューティング .....	351
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 .....	351
<b>付録 E 技術的な注意点 .....</b>	<b>353</b>
システムの時計の設定 .....	353
同期の推奨 .....	353
システムの時計を逆戻りさせない .....	354
OS で定義されるファイル記述子の制限事項 .....	354
持続データの保護 .....	355
組み込み持続ストア .....	355
プラグイン持続ストア .....	356
<b>付録 F Message Queue リソースアダプタ .....</b>	<b>357</b>
<b>付録 G Message Queue オプションの JMS 機能の実装 .....</b>	<b>359</b>
<b>付録 H Message Queue インタフェースの安定度 .....</b>	<b>361</b>
<b>用語集 .....</b>	<b>365</b>
<b>索引 .....</b>	<b>369</b>



# 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	37
図 1-2	メッセージサービスのアーキテクチャ	39
図 1-3	JMS プログラミングオブジェクト	41
図 1-4	MDB を使用したメッセージング	44
図 2-1	Message Queue システムアーキテクチャ	53
図 2-2	ブローカサービスのコンポーネント	55
図 2-3	コネクションサービスのサポート	57
図 2-4	持続マネージャのサポート	67
図 2-5	セキュリティマネージャのサポート	71
図 2-6	監視サービスのサポート	74
図 2-7	マルチブローカ ( クラスタ ) のアーキテクチャ	87
図 2-8	メッセージングの処理	91
図 2-9	Message Queue クライアントランタイムへのメッセージの配信	92
図 3-1	ローカルおよびリモートの管理ユーティリティ	101
図 5-1	ブローカ設定ファイル	135
図 9-1	Message Queue サービスを使用したメッセージの配信	241
図 9-2	配信モードによるパフォーマンスへの影響	245
図 9-3	サブスクリプションタイプによるパフォーマンスへの影響	247
図 9-4	メッセージのサイズによるパフォーマンスへの影響	249
図 9-5	トランスポートプロトコルの速度	252
図 9-6	トランスポートプロトコルによるパフォーマンスへの影響	253
図 9-7	1K バイト (1024 バイト) パケットの inbufsz を変更した結果	296
図 9-8	1K バイト (1024 バイト) パケットの outbufsz を変更した結果	296
図 C-1	HTTP/HTTPS サポートのアーキテクチャ	320





# 表目次

表 1	マニュアルの内容	24
表 2	マニュアルの表記規則	26
表 3	Message Queue ディレクトリ変数	27
表 4	Message Queue マニュアルセット	29
表 1-1	JMS プログラミングオブジェクト	47
表 2-1	主要なブローカサービスコンポーネントと機能	55
表 2-2	ブローカがサポートするコネクションサービス	56
表 2-3	コネクションサービスのプロパティ	59
表 2-4	メッセージルーターのプロパティ	65
表 2-5	持続マネージャのプロパティ	68
表 2-6	セキュリティマネージャのプロパティ	72
表 2-7	ロギングのカテゴリ	75
表 2-8	メトリックスのトピック送信先	76
表 2-9	監視サービスのプロパティ	77
表 2-10	自動作成の設定プロパティ	83
表 2-11	送信先の属性	95
表 3-1	共通の Message Queue コマンド行オプション	103
表 5-1	ブローカインスタンス設定プロパティ	136
表 5-2	imqbrokerd オプション	143
表 5-3	クラスタ設定プロパティ	147
表 5-4	imqbrokerd ログオプションと対応するプロパティ	155
表 6-1	imqcmd のサブコマンド	160
表 6-2	imqcmd のオプション	162
表 6-3	ブローカを管理する imqcmd のサブコマンド	166
表 6-4	imqcmd によって更新されるブローカプロパティ	168
表 6-5	コネクションサービスを管理する imqcmd のサブコマンド	171
表 6-6	ブローカでサポートされているコネクションサービス	173

表 6-7	<a href="#">imqcmd</a> によって更新されるコネクションサービスプロパティ	174
表 6-8	<a href="#">コネクションサービスを管理する imqcmd のサブコマンド</a>	176
表 6-9	<a href="#">送信先の管理に使用される imqcmd のサブコマンド</a>	177
表 6-10	<a href="#">送信先の属性</a>	180
表 6-11	<a href="#">送信先ディスク利用率のメトリックス</a>	187
表 6-12	<a href="#">永続サブスクリプションを管理する imqcmd のサブコマンド</a>	188
表 6-13	<a href="#">トランザクションを管理する imqcmd のサブコマンド</a>	189
表 7-1	<a href="#">LDAP オブジェクトストアの属性</a>	194
表 7-2	<a href="#">ファイルシステムオブジェクトストアの属性</a>	196
表 7-3	<a href="#">コネクションファクトリ管理対象オブジェクトの属性</a>	198
表 7-4	<a href="#">送信先管理対象オブジェクトの属性</a>	199
表 7-5	<a href="#">imqobjmgr サブコマンド</a>	200
表 7-6	<a href="#">imqobjmgr のオプション</a>	201
表 7-7	<a href="#">命名規則の例</a>	207
表 8-1	<a href="#">ユーザーリポジトリでの初期エントリ</a>	215
表 8-2	<a href="#">imqusermgr サブコマンド</a>	216
表 8-3	<a href="#">imqusermgr オプション</a>	217
表 8-4	<a href="#">ユーザー名とパスワードに無効な文字</a>	218
表 8-5	<a href="#">LDAP 関連プロパティ</a>	222
表 8-6	<a href="#">アクセス規則の構文要素</a>	225
表 8-7	<a href="#">送信先アクセス制御規則の要素</a>	228
表 8-8	<a href="#">キーストアのプロパティ</a>	232
表 8-9	<a href="#">passfile のパスワード</a>	236
表 9-1	<a href="#">高信頼性シナリオと高パフォーマンスシナリオの比較</a>	243
表 9-2	<a href="#">imqcmd metrics サブコマンドの構文</a>	257
表 9-3	<a href="#">imqcmd metrics サブコマンドのオプション</a>	257
表 9-4	<a href="#">imqcmd query サブコマンドの構文</a>	261
表 9-5	<a href="#">メトリックスのトピック送信先</a>	263
表 9-6	<a href="#">メトリックス監視ツールの長所と短所</a>	266
表 9-7	<a href="#">JVM メトリックス</a>	268
表 9-8	<a href="#">ブローカ全体のメトリックス</a>	268
表 9-9	<a href="#">コネクションサービスのメトリックス</a>	270
表 9-10	<a href="#">送信先メトリックス</a>	273
表 A-1	<a href="#">Solaris 上での Message Queue データの場所</a>	303
表 A-2	<a href="#">Linux 上での Message Queue データの場所</a>	304
表 A-3	<a href="#">Windows 上での Message Queue データの場所</a>	306
表 B-1	<a href="#">JDBC 関連プロパティ</a>	312

表 B-2	<code>imqdbmgr</code> のサブコマンド	316
表 B-3	<code>imqdbmgr</code> のオプション	317
表 C-1	<code>httpjms</code> コネクションサービスのプロパティ	323
表 C-2	HTTP トンネルサーブレット jar ファイルの配置に使用するサーブレット引数	327
表 C-3	<code>httpsjms</code> コネクションサービスのプロパティ	335
表 C-4	HTTPS トンネルサーブレット jar ファイルの配置に使用するサーブレット引数	340
表 D-1	<code>imqsvcadmin</code> のサブコマンド	348
表 D-2	<code>imqsvcadmin</code> のオプション	349
表 G-1	オプションの JMS 機能	359
表 H-1	Message Queue インタフェースの安定度	361
表 H-2	インタフェースの安定度の分類方式	363



# 手順一覧

運用環境を設定するには	98
運用環境を設定するには	99
管理コンソールを起動するには	106
管理コンソールのヘルプ情報を表示するには	108
ブローカを起動するには	110
管理コンソールにブローカを追加するには	111
管理者パスワードを変更するには	112
ブローカに接続するには	113
利用可能なコネクションサービスを表示するには	114
ブローカにキュー送信先を追加するには	116
物理的送信先のプロパティを表示するには	118
送信先からメッセージをパージするには	119
送信先を削除するには	120
ファイルシステムオブジェクトストアを追加するには	121
オブジェクトストアのプロパティを表示するには	124
オブジェクトストアに接続するには	124
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	125
送信先をオブジェクトストアに追加するには	127
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	129
HelloWorldMessageJNDI アプリケーションを実行するには	130
デフォルトのブローカ名と設定を使用するブローカインスタンスを起動するには	142
Enterprise Edition のトライアルライセンスでブローカインスタンスを起動するには	142
プラグイン持続で名前付きブローカインスタンスを起動するには	142
ブローカをクラスタに接続するには	150
クラスタで安全なコネクションを設定するには	151
新しいブローカを既存のクラスタに追加するには	151
既存のクラスタのメンバーであるブローカを再起動するには	152
ブローカを既存のクラスタから削除するには	152
設定変更レコードをバックアップするには	153
障害時にマスターブローカを復元するには	153
ブローカのローガー設定を変更するには	155

未使用の送信先ディスクスペースを再利用するには	187
設定ファイルを編集して、LDAP サーバを使用するには	221
SSL ベースのコネクションサービスを設定するには	230
キーの組み合わせを生成し直すには	232
ブローカで SSL ベースのサービスを有効にするには	233
metrics サブコマンドを使用するには	259
ログファイルを使用してメトリックス情報を報告するには	262
メッセージベースの監視を設定するには	263
JDBC アクセスが可能なデータストアに接続するには	310
HTTP サポートを有効にするには	321
httpjms コネクションサービスをアクティブにするには	323
トンネルサーブレットを追加するには	327
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	328
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	328
サーバのアクセスログを無効にするには	328
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	329
HTTP トンネルサーブレットを Application Server 7.0 環境に配置するには	330
アプリケーションサーバの server.policy ファイルを変更するには	331
HTTPS サポートを有効にするには	332
httpsjms コネクションサービスをアクティブにするには	335
JSSE を設定するには	337
トンネルサーブレットを追加するには	340
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	341
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	341
サーバのアクセスログを無効にするには	342
HTTPS トンネルサーブレット WAR ファイルを修正するには	342
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	343
HTTPS トンネルサーブレットを Application Server 7.0 環境に配置するには	344
アプリケーションサーバの server.policy ファイルを変更するには	345
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	351

# はじめに

この『Sun Java™ System Message Queue 3.5 SP1 管理ガイド』には、Message Queue 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。

ここでは、次の節について説明します。

- [23 ページの「マニュアルの対象読者」](#)
- [24 ページの「マニュアルの構成」](#)
- [26 ページの「マニュアルの表記規則」](#)
- [29 ページの「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、Message Queue 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。

Message Queue 管理者とは、Message Queue メッセージングシステム、特にシステムの中核となる Message Queue 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、Message Queue 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、読者が全編を読み通すように構成されています。次の表は、各章の内容について簡単に説明します。

表 1 マニュアルの内容	
章	説明
第 1 章「概要」	Message Queue 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「Message Queue 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 Message Queue ブローカと Message Queue クライアントランタイムに重点をおいた、Message Queue メッセージングシステムの説明
第 3 章「Message Queue の管理タスクと管理ツール」	Message Queue 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」	Message Queue 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」	Message Queue 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、Message Queue 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」	Message Queue 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「セキュリティの管理」	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
第 9 章「メッセージサービスの分析と調整」	メッセージサーバのパフォーマンスを監視し分析する技術と、パフォーマンスを最適化するためのメッセージサーバの調整方法の説明
付録 A「Message Queue データの場所」	さまざまなカテゴリの Message Queue データの場所の説明
付録 B「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、Message Queue を設定する方法の説明



表 1 マニュアルの内容 ( 続き )

章	説明
付録 C 「HTTP/HTTPS サポート (Enterprise Edition)」	メッセージングクライアントと Message Queue 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 D 「Windows のサービスとしてのブローカの使用」	Message Queue 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、クエリー、および削除する方法の説明
付録 E 「技術的な注意点」	Message Queue 固有の管理には含まれないが、このマニュアルで取り上げたトピックに関連する多数の特殊な技術的注意事項
付録 F 「Message Queue リソースアダプタ」	Message Queue リソースアダプタの説明、その配備方法、設定および使用方法の説明
付録 G 「Message Queue オプションの JMS 機能の実装」	JMS 仕様において、JMS プロバイダの実装で省略可能として一覧されている各アイテムを、Message Queue 製品が処理する方法の説明
付録 H 「Message Queue インタフェースの安定度」	さまざまな Message Queue インタフェースの安定性の説明
用語集	Message Queue 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	
書式	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。強調するマニュアル名や説明の対象となる語句や項目に対しても使用される
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 (Message Queue、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

## ディレクトリ変数の表記規則

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

表 3 Message Queue ディレクトリ変数

変数	説明
<b>IMQ_HOME</b>	<p>この変数は通常、Message Queue マニュアル内で Message Queue 基本ディレクトリ (ルートインストールディレクトリ) を参照するのに使用される</p> <ul style="list-style-type: none"> <li>• Solaris の場合、ルート Message Queue インストールディレクトリは存在しない。そのため、IMQ_HOME は、Solaris 上のファイルの場所を参照するために Message Queue マニュアルで使用されることはない</li> <li>• Solaris の Sun Java System Application Server の場合、ルート Message Queue インストールディレクトリは Application Server 基本ディレクトリの下での /imq である</li> <li>• Windows の場合、ルート Message Queue インストールディレクトリは Message Queue インストーラによって設定される。デフォルトでは、C:\Program Files\Sun\MessageQueue3</li> <li>• Windows の Sun Java System Application Server の場合、ルート Message Queue インストールディレクトリは Application Server 基本ディレクトリの下での /imq である</li> <li>• Linux の場合、ルート Message Queue インストールディレクトリは存在しない。そのため、IMQ_HOME は、Linux 上のファイルの場所を参照するために Message Queue マニュアルで使用されることはない</li> </ul>
<b>IMQ_VARHOME</b>	<p>Message Queue の一時的な、または動的に作成された設定ファイルやデータファイルが格納されている、/var ディレクトリ。任意のディレクトリを指す環境変数として設定される</p> <ul style="list-style-type: none"> <li>• Solaris の場合、IMQ_VARHOME のデフォルト値は /var/imq ディレクトリ</li> <li>• Solaris の場合、Sun Java System Application Server の Evaluation Edition では、IMQ_VARHOME のデフォルト値は IMQ_HOME/var ディレクトリ</li> <li>• Windows の場合、IMQ_VARHOME のデフォルト値は IMQ_HOME/var ディレクトリ</li> <li>• Windows の場合、Sun Java System Application Server の IMQ_VARHOME のデフォルト値は IMQ_HOME/var ディレクトリ</li> <li>• Linux の場合、IMQ_VARHOME のデフォルト値は /var/opt/imq ディレクトリ</li> </ul>

表 3      Message Queue ディレクトリ変数 ( 続き )

変数	説明
IMQ_JAVAHOME	<p>Message Queue 実行可能ファイルに必要な、Java™ ランタイム (JRE) の場所を指す環境変数</p> <ul style="list-style-type: none"><li>• Solaris の場合、IMQ_JAVAHOME のデフォルト値は /usr/j2se/jre だが、ユーザーはオプションで必要な JRE の配置場所を自由に設定できる</li><li>• Windows の場合、IMQ_JAVAHOME のデフォルト値は IMQ_HOME/jre だが、ユーザーはオプションで必要な JRE の配置場所を自由に設定できる</li><li>• Linux の場合、最初に Message Queue は /usr/java/j2sdkVersion ディレクトリ内で Java ランタイムを検索してから /usr/java/j2reVersion ディレクトリ内を検索する。ユーザーはオプションで IMQ_JAVAHOME の値に必要な JRE の配置場所を自由に設定できる</li></ul>

このマニュアルでは、IMQ\_HOME、IMQ\_VARHOME、および IMQ\_JAVAHOME は、プラットフォーム固有の環境変数の表記法や構文 (UNIX の \$IMQ\_HOME など) に関係なく示されています。パス名には、通常、UNIX のディレクトリ区切り文字の表記法 (/) が使用されています。

# 関連マニュアル

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

## Message Queue マニュアルセット

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

表 4 Message Queue マニュアルセット		
マニュアル	対象読者	説明
『Message Queue インストールガイド』	開発者および管理者	Message Queue ソフトウェアの Solaris、Linux、Windows の各プラットフォームへのインストール方法を説明
『Message Queue リリースノート』	開発者および管理者	新機能、制限、既知のバグ、および技術的な注意点を収録
『Message Queue 管理ガイド』	管理者。開発者にも推奨	Message Queue 管理ツールを使用した管理タスクの実行に必要な基本情報を提供
『Message Queue Java Client Developer's Guide』	開発者	JMS 仕様と SOAP/JAXM 仕様の Message Queue 実装を使用する Java クライアントプログラムの開発者向けにクイックスタートチュートリアルとプログラミング情報を提供
『Message Queue C Client Developer's Guide』	開発者	Message Queue メッセージサーバへの C インタフェース (C-API) を使用する C クライアントプログラムの開発者向けにプログラミングマニュアルとリファレンスマニュアルを提供

## オンラインヘルプ

Message Queue には、Message Queue メッセージサービス管理タスクを実行するためのコマンド行ユーティリティが含まれています。各ユーティリティのオンラインヘルプにアクセスするには、[103 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、Message Queue には、グラフィカルユーザーインターフェース (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

Message Queue Java クライアント API (JMS API を含む) の JavaDoc 形式のマニュアルは、オペレーティングシステムに応じて該当するディレクトリに含まれています ([付録 A 「Message Queue データの場所」](#)を参照)。

このマニュアルは、Netscape または Internet Explorer などの HTML ブラウザで表示できます。このマニュアルには、標準の JMS API マニュアルおよび Message Queue 管理対象オブジェクト用の Message Queue 固有の API が含まれており (『Message Queue Java Client Developer's Guide』の第 3 章を参照)、メッセージングアプリケーションの開発者にとって有用です。

## クライアントアプリケーション例

クライアントアプリケーションのサンプルコードを示す多数のアプリケーション例が、オペレーティングシステムに応じて該当するディレクトリに含まれています ([付録 A 「Message Queue データの場所」](#)を参照)。

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

## Java Message Service (JMS) 仕様書

JMS 仕様書は、次のサイトにあります。

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

この仕様書には、サンプルのクライアントコードも掲載されています。

# 関連するサードパーティの Web サイトのリファレンス

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

---

## 注

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

---

関連するサードパーティの Web サイトのリファレンス



## 概要

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

## Sun Java System Message Queue とは

Message Queue 製品は、分散アプリケーションに対応した信頼性の高い非同期メッセージングを実現する標準的なソリューションです。Message Queue は、JMS (Java™ Message Service) オープンスタンダードを実装するエンタープライズメッセージングシステムで、実際に、JMS 参照実装として利用されています。ただし、Message Queue は、企業向けの強力な機能をフルに備えた JMS プロバイダでもあります。

JMS 仕様では、一連のメッセージングのセマンティクスと動作、アプリケーションプログラミングインタフェース (API) が規定されています。この仕様には、Java 言語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 (40 ページの「JMS プログラミングモデル」を参照)。Message Queue は、Java メッセージングアプリケーションをサポートする以外にも、Message Queue サービスへの C 言語インタフェース (Message Queue C-API) を提供しています。

Sun Java System 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 Message Queue メッセージサービス (38 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。アプリケーション開発者は、ネットワークを介したアプリケーションの信頼できる通信手順という低レベルの問題にとらわれることなく、アプリケーションのビジネスロジックの開発に集中できます。

Message Queue には、JMS 仕様で要求されている以上の機能があります。たとえば、次のような機能が含まれています。

**集中管理** : Message Queue サービスや、送信先、トランザクション、永続サブスクリプション、セキュリティなどのアプリケーションによって異なるエンティティを管理するために、コマンド行と GUI ツールの両方を提供します。Message Queue は、Message Queue サービスのリモートでの監視もサポートしています。

**スケーラブルなメッセージサービス** : 増加するコンポーネントやアプリケーションといったクライアントにサービスを提供するため、マルチブローカクラスタでは並行して動作することで、複数の Message Queue メッセージサーバコンポーネント (ブローカ) 間でのロードバランスを最適化します。

**クライアントコネクションフェイルオーバー** : Message Queue メッセージサーバへのクライアントコネクションに障害が生じた場合に、コネクションを自動的に復元します。

**調整可能なパフォーマンス** : 信頼性がそれほど重要ではない配信を行う場合に、Message Queue サービスのパフォーマンスを向上させることができます。

**複数のトランスポート** : 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、Message Queue クライアントが Message Queue メッセージサーバと通信する機能をサポートします。

**JNDI のサポート** : オブジェクトストアおよびユーザーリポジトリとしてファイルベースと Java Naming and Directory Interface (JNDI) による LDAP 実装の両方をサポートします。

**SOAP メッセージングのサポート** : 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、JMS メッセージング経由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、『Message Queue Java Client Developer's Guide』を参照してください。

JMS への準拠に関連する問題については、[付録 G 「Message Queue オプションの JMS 機能の実装」](#) を参照してください。

# 製品エディション

Sun Java System 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、それぞれ機能とライセンス数が異なります。Message Queue のエディションのアップグレード方法については、『Message Queue インストールガイド』を参照してください。

## 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、Sun Java System Application Server プラットフォームにもバンドルされています。Platform Edition では、Message Queue メッセージサーバでサポートされるクライアントコネクションの数に制限はありません。Platform Edition には、次の 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 ) が用意されているが、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、複数コンシューマへのキュー配信、リモートでのメッセージベースの監視、C-API サポートなどの企業向けの機能は含まれていない。このライセンスには期限がないため、要件の緩い運用環境で利用できる
- **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、クライアントコネクションのフェイルオーバー、複数コンシューマへのキュー配信、リモートメッセージベースでの監視、C-API 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る。ただし、ソフトウェアのライセンスの有効期間は 90 日間であるため、Enterprise Edition の製品 ( 「Enterprise Edition」を参照 ) で提供される企業向けの機能を評価するのに適している。

---

注	90 日間のトライアルライセンスは、142 ページの「Enterprise Edition のトライアルライセンスでブローカインスタンスを起動するには」の説明にしたがい、Message Queue メッセージサーバ (Message Queue ブローカインスタンス) を起動することで有効にできます。
---	---

---

## 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備および実行するために使用されます。このエディションには、マルチブローカ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、クライアントコネクションフェイルオーバー、複数コンシューマへのキュー配信、リモートメッセージベースでの監視、C-API サポートが含まれています。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はありません。また、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には制限がありませんが、これは使用される CPU の数によって決まります。

## 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、独立した分散アプリケーションやアプリケーションコンポーネント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、同一ホスト、同一ネットワーク、または無操作に接続されたインターネットであっ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、ユーザーの認証、メッセージとリソースへの承認されたアクセス、ネットワーク上の暗号化など、基本的なセキュリティ機能をサポートしている必要があります。

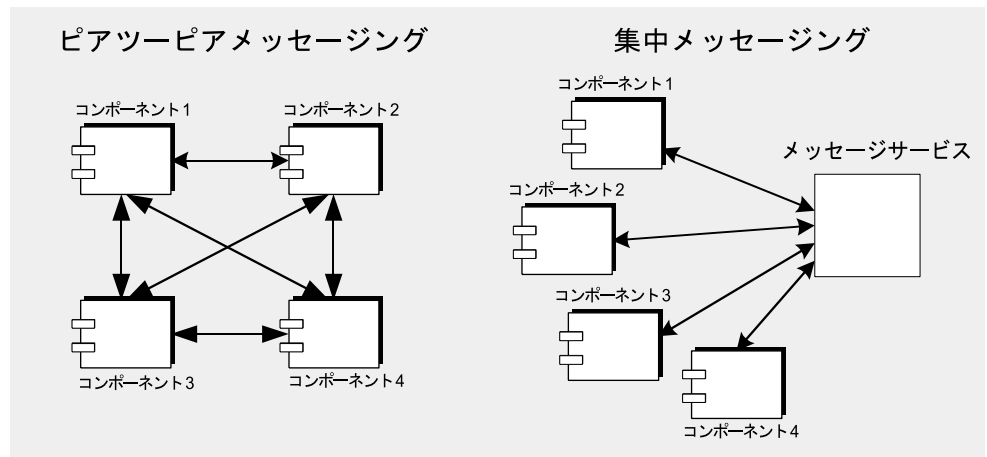
**スケーラビリティ：**メッセージングシステムは、パフォーマンスやメッセージスループットを実質的に失うことなく、ユーザーやメッセージの増加という増大する負荷に対応できる必要があります。ビジネスやアプリケーションが拡大するにつれ、これは非常に重要な要件となります。

**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必要があります。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接続を維持しています。これらの接続では、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、接続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接続を維持します。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。次のものが含まれます。メッセージ、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。

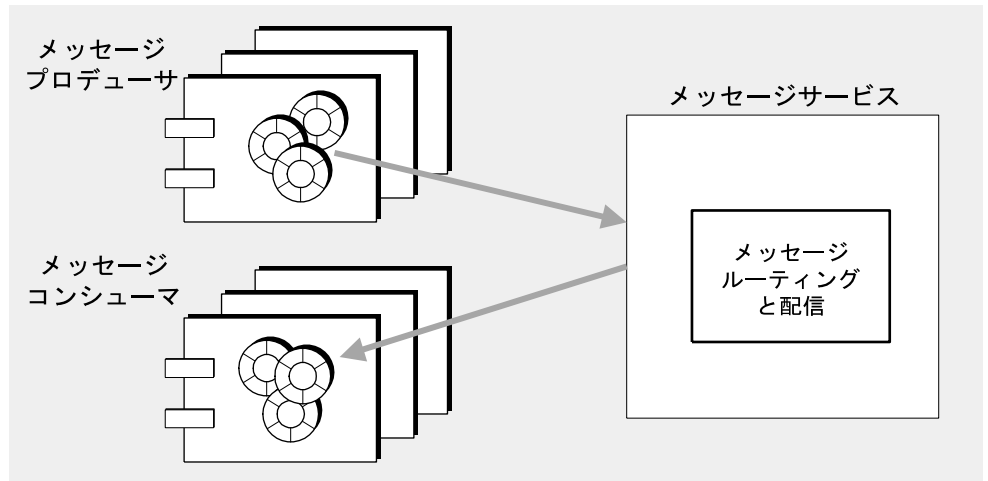
### メッセージ

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

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

通常、これらの関係は、ポイントツーポイントメッセージングとパブリック / サブスクライブメッセージングの 2 つのメッセージ配信モデルに集約されます。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) 仕様では、Java プログラミング用の API とのメッセージングに対応した標準のセマンティクスが作成されています。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 (46 ページの「プログラミングドメイン」を参照)。

# JMS 仕様

JMS 仕様では、プログラミングモデル、メッセージ構造、APIをはじめ、メッセージングを制御する一連のルールとセマンティクスが規定されています。**Message Queue** は JMS を実装するため、JMS の概念は、**Message Queue** 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 メッセージは、ヘッダー、プロパティ、本体の 3 つの部分から構成されています。

**ヘッダー**：ヘッダーは、メッセージの JMS 特性を指定します。この特性には、メッセージの送信先、持続的かどうか、生存期間、優先度が含まれます。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。

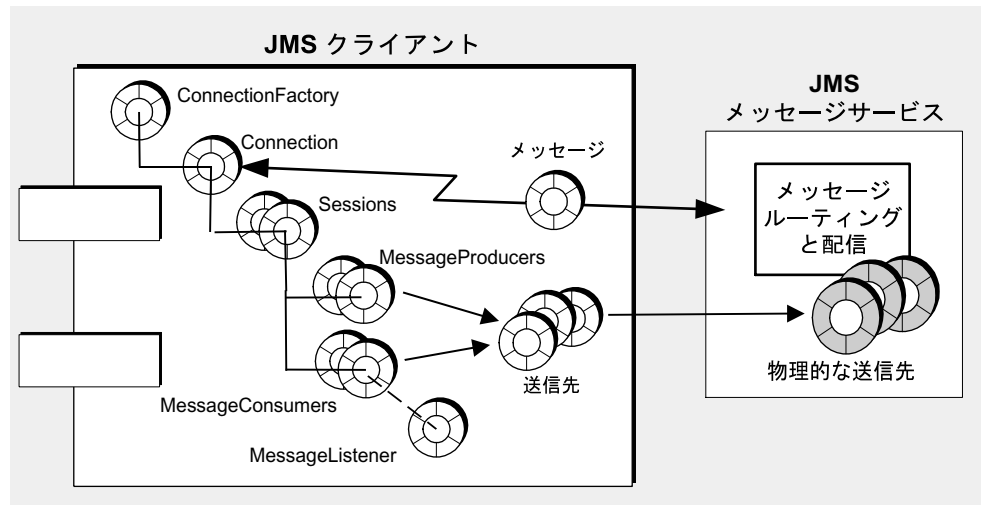
## JMS プログラミングモデル

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、『**Message Queue Java Client Developer's Guide**』を参照)。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、メッセージをメッセージサービスから受信したり、メッセージサービスへ送信したりするときには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ので、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。セッションは、多数の通知オプションまたはトランザクションを使って、信頼性の高い配信処理をサポートします。

クライアントは `MessageProducer` を使用して、API 中の送信先 ID オブジェクトに示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、クライアントは `MessageConsumer` を使用して、API 中の送信先オブジェクトに示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、メッセージセクタを使用できます。このメッセージセクタを使用すると、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。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。

## JMS 管理対象オブジェクト

JMS 仕様では、プロバイダ固有の設定情報をカプセル化する管理対象オブジェクトを規定することで、プロバイダからのクライアントの独立性を高めています。

40 ページの「[JMS プログラミングモデル](#)」で説明している 2 つのオブジェクトでは、JMS プロバイダが JMS メッセージサービスを異なった方法で実装しています。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 仕様では、プロバイダ固有の実装をして、設定情報をコネクションファクトリと送信先オブジェクトにカプセル化する必要があります。その後、プロバイダに依存しない標準的な方法で、これらのオブジェクトにアクセスできます。

管理対象オブジェクトは管理者によって作成、構成され、ネーミングサービスに格納されると、クライアントが標準的な `Java Naming and Directory Service (JNDI)` 検索コードを介してアクセスします。この方法で管理対象オブジェクトを使用すると、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。

コネクションファクトリと送信先の 2 つのタイプの管理対象オブジェクトはどちらもプロバイダ固有の情報をカプセル化しますが、クライアント内での用途は大きく異なっています。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、物理的送信先を識別するのに使用されます。

# JMS/J2EE プログラミング：メッセージ駆動型 Beans

40 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、さらに JMS に特化したプログラミングモデルがあり、Java 2 Platform, Enterprise Edition (J2EE プラットフォーム) 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 (<http://java.sun.com/products/ejb/docs.html>) で指定されている Enterprise JavaBeans (EJB) コンポーネントのシリーズの 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) は同時に呼び出す必要があるため、メッセージ駆動型 Beans が必要となります。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を結びつけずに、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SUME 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。

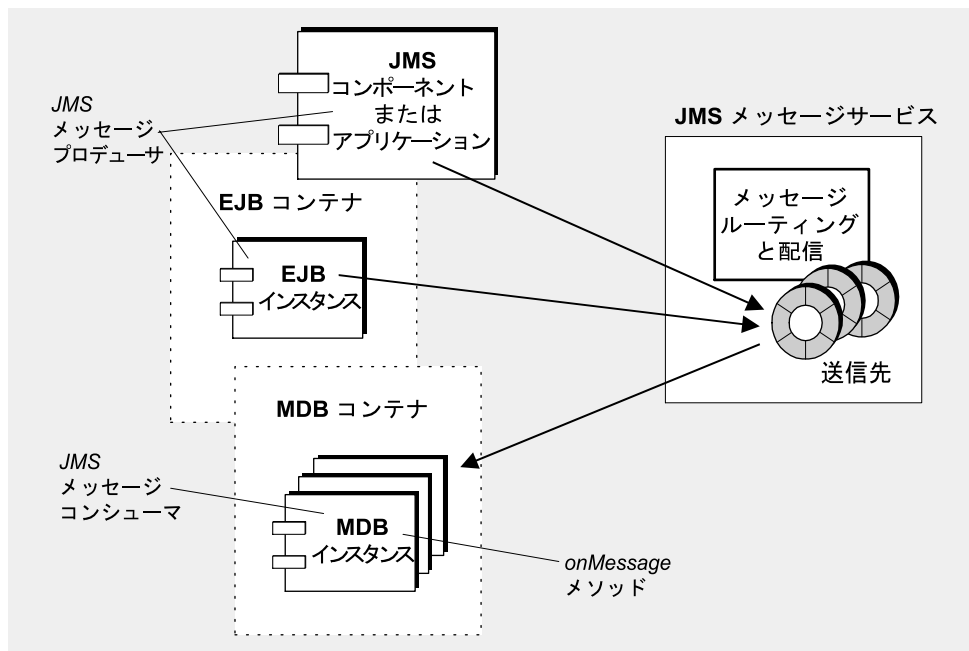
## メッセージ駆動型 Beans

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) のサポート対象となる特殊な 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 開発者がプログラミングした `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SUME するのと同じように、メッセージをコンSUME します。ほかの EJB コンポーネントの場合とは異なり、メソッドを MDB でリモートに呼び出さないため、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SUME できます。[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、EJB コンポーネント、および Web コンポーネントにより、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 は、専用の EJB コンテナによってサポートされます。この EJB 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コンシュームを行うようにインスタンスを設定する役目を持ちます。これには、メッセージサービスを使用した接続の設定 (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に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ため、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、コンテナを設定するために配置ツールによって使用されるほかの情報も含まれます。各コンテナでは、1 つの MDB のインスタンスだけをサポートします。

## J2EE 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 コンテナが J2EE 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は、トランザクションマネージャ、持続マネージャ、ネームサービス、メッセージングや MDB の場合には JMS プロバイダといった、さまざまなコンテナで必要とされるリソースを提供します。

Sun Java System アプリケーションサーバでは、Sun Java System Message Queue によって JMS メッセージングのリソースが提供されます。

- Sun Java System アプリケーションサーバ 7.0 の場合、Message Queue メッセージングシステムはネイティブな JMS プロバイダとしてアプリケーションサーバに統合されます。
- Sun J2EE 1.4 アプリケーションサーバの場合、Message Queue は埋め込みの JMS リソースアダプタとしてアプリケーションサーバにプラグインされます (付録 F「Message Queue リソースアダプタ」を参照)。
- アプリケーションサーバの将来リリースでは、Message Queue はリソースアダプタの標準的な配備方法と設定方法を使用してアプリケーションサーバにプラグインされる予定です。

## JMS メッセージングの問題点

この節では、Message Queue 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 Message Queue 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 (42 ページの「JMS 管理対象オブジェクト」を参照) の使用方法を規定し、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、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は、Message Queue 管理者により作成および設定される Message Queue 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 クライアントが JNDI 検索を使用してアクセスします。

また、Message Queue 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。この欠点は、アプリケーション開発者がプロバイダ固有の API を使用しなければならない点です。また、Message Queue 管理者が Message Queue メッセージサーバを容易に制御し管理するための機能も損なわれてしまいます。

管理対象オブジェクトの詳細については、[93 ページの「Message Queue 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 は、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 2 つの個別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。この配信モデルでは、送信先がキューとなります。メッセージは最初にキュー送信先に配信され、次にキューの配信ポリシー ([80 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 つへキューから配信されます。キュー送信先に複数のプロデューサがメッセージを送信してもかまいません。しかし、個々のメッセージは、単一のコンシューマに向けて配信され、コンシュームされることになっています。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したとき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。この配信モデルでは、トピックが送信先になります。メッセージは最初にトピック送信先に配信され、次にトピックに加入した、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。トピック送信先は、永続的サブスクリプションの概念もサポートします。永続的なサブスクリプションでは、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的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。

この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クスを持った別々の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) <sup>1</sup>	Queue	Topic
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Connection	QueueConnection	TopicConnection
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver	TopicSubscriber

1. プログラミングの手法によっては、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、クライアントの代わりにメッセージサービス保持している状態の情報を使用して、JMS クライアントのコネクションをメッセージサービスに関連付ける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。本来、クライアント識別子は固有のもので、一度に1人のユーザーだけに適用されます。クライアント識別子は永続的サブスクリプション名 (46 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、それぞれの永続的サブスクリプションが1人のユーザー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 メソッド経由でクライアントにより設定されますが、コネクションファクトリ管理対象オブジェクト (42 ページの「[JMS 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、各ユーザー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。

Message Queue 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 とユーザー固有のものの両方の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、永続的サブスクリプションを作成する複数のユーザーが、単一の ConnectionFactory オブジェクトを使用できます。したがって、ユーザーの永続的サブスクリプションは、誤って消去されることも、別のユーザー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 Message Queue の機能の使用に関する詳細は、『[Message Queue Java Client Developer's Guide](#)』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。

どのような場合でも、クライアント識別子は、永続的サブスクリプションを作成するために、JMS API を使用してクライアントがプログラム上設定するか、またはクライアントが使用する ConnectionFactory オブジェクト内に管理上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。

**持続性メッセージ:** 持続性メッセージは、必ず 1 回は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。持続性のメッセージにとって、信頼性は大変重要です。

**持続性のないメッセージ:** 持続性のないメッセージは、1 回の配信だけが保証されています。持続性の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うことがないようにすることです。



## 通知 / トランザクション

信頼性のあるメッセージングは、送信先からの持続メッセージや送信先への持続メッセージが確実に配信されることに依存します。Message Queue 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。トランザクションの場合、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る可能性があります。

### 通知

信頼性の高い配信を確実にするため、通知を使用するようにセッションを構成できます。

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により送信先にコネクションメッセージの配信が通知されることになります。コンシューマの場合は、メッセージサービスがメッセージを削除する前に、送信先からのコネクションメッセージの配信とコンシュームが、クライアントにより通知されることになります。

### 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。ここでは、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、トランザクションを起動、確定、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に従って、ブローカが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確定したときにだけ、送受信の操作を完了させます。トランザクション内での特定の送信や受信の操作が失敗すると、例外が発生します。クライアントコードは、これを無視するか、操作を試行し直すか、またはトランザクション全体をロールバックして、例外を処理できます。トランザクションが確定して、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た、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終端間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は置くことはできません。

## 分散トランザクション

Message Queue では、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となります。分散トランザクションでは、Java Transaction API (JTA)、XA Resource API の仕様に定義された 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によって実行される操作を、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 の仕様に記述されます。

分散トランザクションをサポートするということは、メッセージングクライアントが、JTA で定義される XAResource インタフェースを介して分散トランザクションに加わることができるということです。このインタフェースでは、2 階層コミットを実装するための、数多くのメソッドが定義されます。API の呼び出しがクライアント側で行われている間、Message Queue 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に関しては、無視したり、操作を試行し直したり、分散トランザクション全体をロールバックしたりして、クライアントは例外を処理できます。

Message Queue では、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が実装されています。この XA コネクションファクトリでは、次に XA セッション (40 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へのサポートには、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データストア (66 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、持続データ格納ではメッセージが修復され、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、永続的サブスクリプションも格納する必要があります。これは、トピック送信先の場合の配信を保証するためで、コネクションメッセージを修復する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ればなるほど、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兼ね合いは、設計上考慮すべき重要な点です。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 **Message Queue** 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 (『**Message Queue Java Client Developer's Guide**』を参照)。パフォーマンスと信頼性のかね合いについては、[242 ページの「パフォーマンスに影響するアプリケーション設計の要因」](#)で詳しく説明します。

## メッセージの選択

JMS には、メッセージセレクトに設定された条件に基づくフィルタや転送を、メッセージサービスが実行できるようにするメカニズムが用意されています。プロデュース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できます。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すことができます。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への配信メッセージの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。メッセージセレクトの構文とセマンティクスは、JMS の仕様書で解説されています。

## メッセージの順番と優先度

一般には、単一のセッションにより送信先に設定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、別々のプロパティが割り当てられている場合、メッセージングサービスは、優先度が高いほうのメッセージを先に配信しようとしています。

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 ([80 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 ([86 ページの「マルチブローカクラスタ \(Enterprise Edition\)」](#)を参照) を使用している Message Queue 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。

# Message Queue メッセージングシステム

この章では、Sun Java™ System Message Queue 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携作業について説明します。

図 2-1 Message Queue システムアーキテクチャ

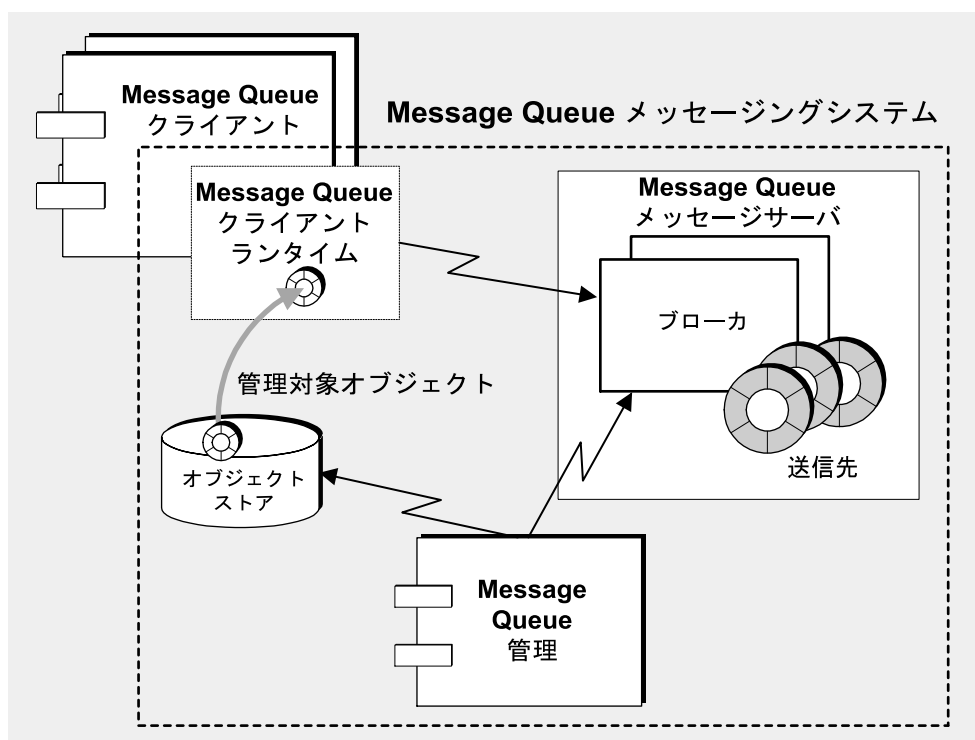


図 2-1 に示す Message Queue メッセージングシステムの主要部分は次のとおりです。

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

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「Message Queue の管理タスクと管理ツール」](#)で説明します。

## Message Queue メッセージサーバ

この節では、[53 ページの図 2-1](#) に示す Message Queue 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。次のものが含まれます。

**ブローカ** : Message Queue ブローカには、Message Queue 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、[「ブローカ」](#)を参照 )。メッセージサーバでは、1 つ以上のブローカインスタンスを使用できます ([86 ページの「マルチブローカクラスタ \(Enterprise Edition\)」](#)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、プロデュース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ー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、[80 ページの「物理的な送信先」](#)を参照 )。

### ブローカ

プロデュースングクライアントから送信先へ、さらに送信先から 1 つ以上のコンシューミングクライアントへ配信される Message Queue メッセージングシステムのメッセージ配信は、ブローカ ( または、並行して動作するブローカインスタンスのクラスタ ) によって行われます。メッセージ配信を実行するには、ブローカはクライアントとの通信チャネルの設定、認証と承認、適切なメッセージルーティング、信頼性の高い配信の保証、およびシステムパフォーマンスを監視するためのデータの提供を行う必要があります。

この複合的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内部コンポーネントを使用します。各コンポーネントには、配信プロセスにおける特定の役割が割り当てられています。図 2-2 にブローカの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。メッセージルーターコンポーネントがメッセージルーティングと配信サービスを主に実行し、それ以外のコンポーネントはメッセージルーターが依存する重要なサポートサービスを提供しています。

図 2-2 ブローカサービスのコンポーネント

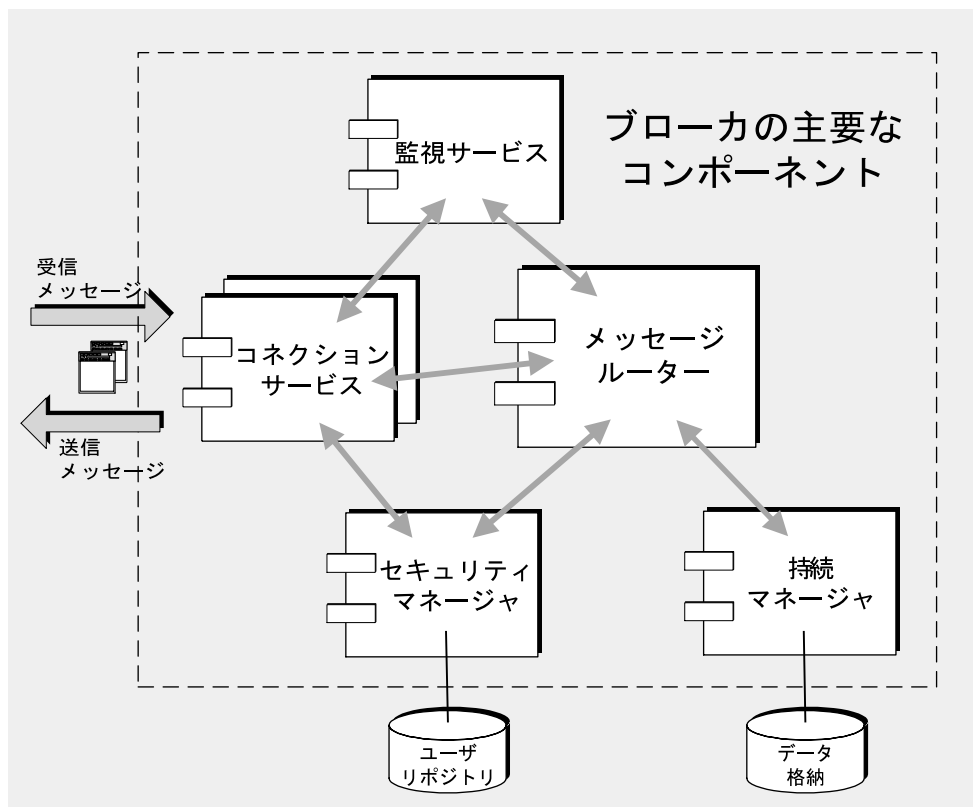


表 2-1 主要なブローカサービスコンポーネントと機能

コンポーネント	説明 / 機能
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 Message Queue メッセージングシステムが使用するコントロールメッセージを含む

表 2-1            主要なブローカサービスコンポーネントと機能 ( 続き )

コンポーネント	説明 / 機能
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
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ーに認証サービスを提供し、認証されたユーザーに承認サービス ( アクセス制御 ) を提供する
監視サービス	さまざまな出力チャネルに書き込み可能なメトリックスと診断情報を生成する。管理者はこれらをブローカの監視および管理に使用できる

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。次の節では、さまざまな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

Message Queue ブローカは、Message Queue アプリケーションクライアントと Message Queue 管理クライアントの両方の通信をサポートしています (100 ページの「Message Queue 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 Message Queue 管理 (ADMIN) サービスのどちらを提供するのかを指定します。

**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ーを指定します。

Message Queue ブローカで現在使用できるコネクションサービスを、表 2-2 に示します。

表 2-2            ブローカがサポートするコネクションサービス

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



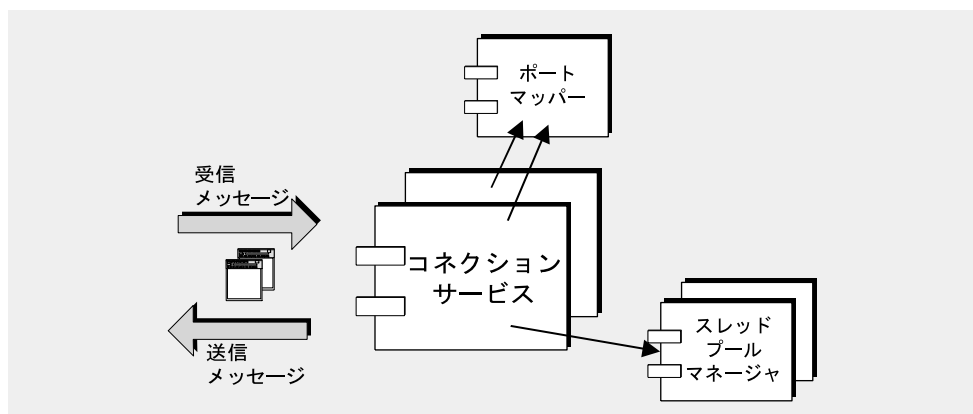
表 2-2 ブローカがサポートするコネクションサービス ( 続き )

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

ブローカを設定して、これらのコネクションサービスの一部、またはすべてを実行することができます。各コネクションサービスは、ブローカのホスト名とポート番号で指定した特定のポートで使用できます。ポートを動的に割り当てることもできれば、コネクションサービスが使用可能なポートを指定することもできます。

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



### ポートマッパー

Message Queue には、ポートをさまざまなコネクションサービスにマップするポートマッパーが用意されています。ポートマッパー自体は、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、たとえばファイアウォール経由のコネクションのような特殊な状況でだけ実行され、一般にはお勧めしません。httpjms サービスと httpsjms サービスは、それぞれ付録 C 「HTTP/HTTPS サポート (Enterprise Edition)」の 323 ページの表 C-1 と 335 ページの表 C-3 に示すプロパティを使用して設定します。

## スレッドプールマネージャ

各コネクションサービスは、複数のコネクションをサポートするマルチスレッドです。これらのコネクションに必要なスレッドは、スレッドプールマネージャのコンポーネントが管理するスレッドプールに保存されます。スレッドプールマネージャを設定して、スレッドプールに保持されるスレッドの最小数と最大数を指定することができます。コネクションでスレッドが必要になると、スレッドプールにスレッドが追加されます。スレッドの最小数より少なくなると、システムは最小数のしきい値になるまで、スレッドをシャットダウンして、スレッドを解放します。これによってメモリ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あるように、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。それでも足りない場合、スレッドが利用できるようになるまで、コネクションは待機します。

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

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

**共有モデル (Enterprise Edition):** コネクションは、メッセージが送信されるか受信されると必ず、共有スレッドによって処理されます。このモデルでは、コネクションごとの専用スレッドは必要ないため、コネクションサービス、さらにブローカがサポートできるコネクション数が多くなります。ただし、スレッドの共有にはある程度のパフォーマンスオーバーヘッドが伴います。スレッドプールマネージャは、一連のディストリビュータスレッドを使用してコネクションのアクティビティを監視し、必要に応じてスレッドにコネクションを割り当てます。このアクティビティに伴うパフォーマンスオーバーヘッドは、各ディストリビュータスレッドが監視するコネクション数を制限することで最小限に抑えることができます。

## セキュリティ

各コネクションサービスは、特定の認証および承認 (アクセス制御) 機能をサポートします (69 ページの「セキュリティマネージャ」を参照)。

## コネクションサービスのプロパティ

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 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
<code>imq.service.activelist</code>	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、 <code>jms</code> 、 <code>ssljms</code> 、 <code>httpjms</code> 、 <code>httpsjms</code> 、 <code>admin</code> 、 <code>ssladmin</code> デフォルト値: <code>jms</code> 、 <code>admin</code>
<code>imq.ping.interval</code>	ブローカがコネクションを介して Message Queue クライアントランタイムへ継続的に <code>ping</code> を試行する間隔 デフォルト値: 120 秒
<code>imq.hostname</code>	1 台のコンピュータに、複数のネットワークインタフェースカードがある場合など、複数のホストを使用できる場合には、すべてのコネクションサービスがバインドするホストを、ホスト名、または IP アドレスで指定する デフォルト値: すべての使用可能な IP アドレス
<code>imq.portmapper.port</code>	ブローカのプライマリポートを指定する。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 デフォルト値: 7676
<code>imq.portmapper.hostname</code>	1 台のコンピュータに、複数のネットワークインタフェースカードがある場合など、複数のホストを使用できる場合には、ポートマッパーがバインドするホストを、ホスト名、または IP アドレスで指定する デフォルト値: <code>imq.hostname</code> の値を継承する
<code>imq.portmapper.backlog</code>	ポートマッパーが、要求を拒否せずに、同時に処理可能な要求の最大数を指定する。オペレーティングシステムのバックログに格納可能な、ポートマッパーによる処理を待機中の要求の数を、プロパティで設定する デフォルト値: 50.

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
<code>imq.service_name. protocol_type<sup>1</sup>.port</code>	<p>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 デフォルト値: 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 )</p> <p>httpjms と httpsjms コネクションサービスを設定する 場合、付録 C 「HTTP/HTTPS サポート (Enterprise Edition)」を参照</p>
<code>imq.service_name. protocol_type<sup>1</sup>.hostname</code>	<p>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 デフォルト値: <code>imq.hostname</code> の値を継承する</p>
<code>imq.service_name. min_threads</code>	<p>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 デフォルト値: コネクションサービスによって異なる (136 ページの表 5-1 を参照)</p>
<code>imq.service_name. max_threads</code>	<p>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、<code>min_threads</code> の値よりも大きくす る必要がある デフォルト値: コネクションサービスによって異なる (136 ページの表 5-1 を参照)</p>
<code>imq.service_name. threadpool_model</code>	<p>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 デフォルト値: コネクションサービスによって異なる (136 ページの表 5-1 を参照)</p>

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.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 -1 に設定した場合、無制限になる デフォルト値: オペレーティングシステムによって異なる (136 ページの表 5-1 を参照)

1. `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 (48 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 (48 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。信頼性の高い配信には、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 つの要素があります。

ブローカとのメッセージの配信を正常に行うために、Message Queue 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。デフォルトでは、プロデューサが JMS メッセージを持続的として指定している場合に限り、Message Queue はこれを実行します。プロデュース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 (91 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、メモリーから JMS メッセージを削除しません。

トピックのサブスクリプションが永続的な場合、メッセージルーターは、各 JMS メッセージをその送信先で保持し、各永続サブスクライバがアクティブなコンシューマになると、そのメッセージを配信します。メッセージルーターは、クライアントの通知を受信するたびにそれを記録し、すべての通知を受信すると、初めて JMS メッセージを削除します (ただし、この時点で JMS メッセージの有効期限が切れている場合は除く)。

さらに、メッセージルーターは、ブローカの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 (92 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 (50 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、メモリー内に保持されます。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 (66 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。

### メモリーリソースとメッセージフローの管理

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

**個々の送信先のメッセージ制限：**物理的な送信先の属性を設定して、メッセージ数とメッセージが使用する総メモリー量の上限を指定できます (180 ページの表 6-10 を参照)。また、この上限に達したときに、メッセージルーターが 4 種類の対応のうちどれを実行するかを指定できます。4 種類の制限の動作は次のとおりです。

- メッセージプロデューサを遅くする
- メモリー内のもっとも古いメッセージを廃棄する
- メッセージの有効期限にしたがい、メモリー内のもっとも優先度の低いメッセージを廃棄する
- 最新のメッセージを拒否する

**システム全体のメッセージ制限：**システム全体のメッセージ制限は、二次的な保護を提供します。システム全体の制限を指定すると、システム上のすべての送信先に対して一括して適用され、メッセージの総数とすべてのメッセージによって使用される総メモリー量が制限されます (65 ページの表 2-4 を参照)。どちらかのシステム全体のメッセージ制限に達した場合、メッセージルーターは新しいメッセージを拒否します。

**システムメモリーのしきい値：**システムメモリーのしきい値は、三次的な保護を提供します。ブローカがさらに深刻な状況に陥ったときに、メモリーの過負荷を避けるためのアクションを実行できるように、使用可能なシステムメモリーのしきい値を指定できます。実行するアクションは、green (使用可能なメモリーが十分にある)、yellow (ブローカのメモリーが減っている)、orange (ブローカのメモリーが不十分である)、red (ブローカのメモリーが不足している) といったメモリーリソースの状態によって異なります。ブローカのメモリーの状態が green から yellow、orange、red へと進むにつれ、ブローカは次のタイプの本格的なアクションを段階的に実行します。

- アクティブメモリーから持続ストレージへメッセージをスワップする (66 ページの「持続マネージャ」を参照)。通常は保存されない持続的でないメッセージも、システムがメモリーを再利用できるようにスワップされることがある
- 持続的でないメッセージのプロデューサの処理速度を低下させ、最終的にブローカへのメッセージフローを止める (持続メッセージのフローは、メッセージごとにブローカによって通知された要件によって自動的に制限される)

これらのアクションはどちらもパフォーマンスを低下させます。

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

### メッセージルーターのプロパティ

メモリーリソースを管理するためのシステム全体の制限とシステムメモリーのしきい値の詳細は、表 2-4 を参照してください。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
<code>imq.message.expiration.interval</code>	有効期限が切れたメッセージを再利用させる頻度を秒単位で指定する デフォルト値: 60
<code>imq.system.max_count</code>	ブローカが保持する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 -1 に設定した場合、無制限になる デフォルト値: -1
<code>imq.system.max_size</code>	ブローカが保持するメッセージの最大のサイズ (バイト、K バイト、または M バイト単位) 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 -1 に設定した場合、無制限になる デフォルト値: -1
<code>imq.message.max_size</code>	メッセージの本体の最大許容サイズ (バイト、K バイト、または M バイト単位) 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 -1 に設定した場合、無制限になる デフォルト値: 70m (M バイト)
<code>imq.resource_state.threshold</code>	指定したメモリーリソースの状態で、そのメモリーリソースがトリガーされるメモリーの利用率を指定する。リソースの状態を表す値は、green、yellow、orange、および red デフォルト値: それぞれ、0、80、90、98
<code>imq.resource_state.count</code>	メモリーリソースの各状態がトリガーされたときに、一度に入力可能なメッセージの最大数を指定する。この制限は、システムメモリーがさらに不十分になると、メッセージプロデューサの処理速度を低下させる デフォルト値: それぞれ、5000、500、50、0
<code>imq.transaction.autorollback</code>	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、 <code>imqcmd</code> 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 ( <a href="#">189 ページの「トランザクションの管理」</a> を参照) デフォルト値: 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。

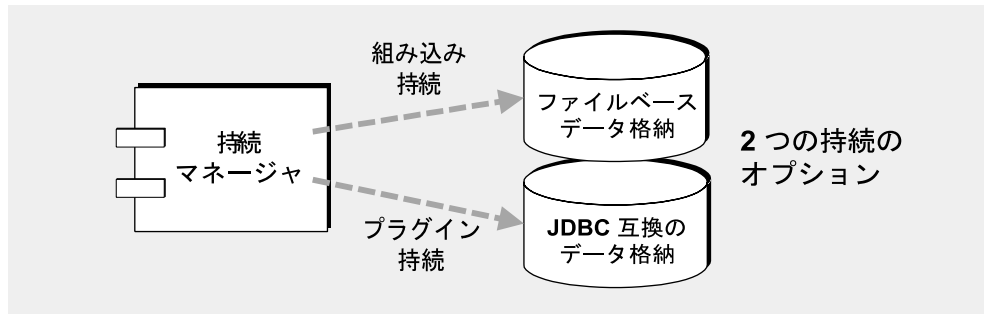
- 送信先の再作成
- 各トピックの永続サブスクリプションのリストの復元
- 各メッセージの通知リストの復元
- コミットされたすべてのトランザクションの状態の復元

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。

Message Queue は、組み込み持続モジュールとプラグイン持続モジュールの両方をサポートしています ( [図 2-4](#) を参照 )。組み込み持続は、ファイルベースのデータストアです。プラグイン持続は、JDBC™ (Java Database Connectivity) インタフェースを使用し、JDBC 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ーもいます。

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

ファイルベースのデータストアは、そのデータストアが関連付けられているブローカインスタンスの名前 (*instanceName*) によって識別されるディレクトリに配置されます (付録 A 「Message Queue データの場所」を参照)。

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

ファイルベースのデータストアは、常駐先の送信先に応じてメッセージがディレクトリに格納されるように構成されています。大半のメッセージは、可変長レコードから構成されるシングルファイルに格納されます。

メッセージが追加および削除されたときの断片化を減らすために、可変長レコードファイルを圧縮できます (186 ページの「送信先の圧縮」を参照)。さらに、設定可能なしきい値 (`imq.persist.file.message.max_record_size`) より大きいメッセージは、可変長レコードファイルではなく、メッセージに該当するファイルへ組み込み持続マネージャが格納します。これらの個々のファイルでは、ファイルが再利用できるようにファイルプールが維持されます。メッセージファイルは不要となった場合でも、削除されず、送信先ディレクトリの空きファイルのプールに追加され、新しいメッセージを格納するために使用されます。

送信先ファイルプールのファイルの最大数

(`imq.persist.file.destination.message.filepool.limit`) を設定できます。また、ファイルプールにおける空きファイルの割合

(`imq.persist.file.message.filepool.cleanratio`) を指定できます。再利用のために単にタグをつける場合と異なり、空きファイルは削除されてサイズが 0 になります。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 (`imq.persist.file.message.cleanup`) を指定することもできます。ファイルが削除されると、ファイルが使用するディスク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。

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

信頼性を最大にするため、持続操作によりメモリー内の状態と物理的なストレージとを同期するように指定できます (`imq.persist.file.sync.enabled`)。この同期化は、システムクラッシュによるデータの損失をなくす上で役立ちますが、パフォーマンスが犠牲になります。

データストアには機密事項を扱うメッセージや財産的価値のあるメッセージが含まれることがあるため、`...instances/instanceName/fs350/` ディレクトリは承認されていないアクセスから保護することをお勧めします。保護する方法については、[355 ページの「持続データの保護」](#)を参照してください。

プラグイン持続

JDBC ドライバを介してアクセスが可能な任意のデータストアにアクセスするように、ブローカを設定することができます。この作業には、さまざまな JDBC 関連のブローカ設定プロパティの設定、適切なスキーマでデータストアを作成する 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 B「プラグイン持続の設定」](#)で説明します。

持続マネージャのプロパティ

持続関連の設定プロパティを、[68 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。

最初のプロパティを除き、[表 2-5](#) に示されたプロパティはすべて、組み込み持続に関係しています。プラグイン持続に関係するプロパティは、[312 ページの表 B-1](#) に示されています。

表 2-5 持続マネージャのプロパティ

プロパティ名	説明
<code>imq.persist.store</code>	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 デフォルト値: file

表 2-5 持続マネージャのプロパティ ( 続き )

プロパティ名	説明
<code>imq.persist.file.sync.enabled</code>	持続操作でメモリー内の状態を物理的なストレージと同期させるかどうかを指定する。 <code>true</code> 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 デフォルト値: <code>false</code>
<code>imq.persist.file.message.max_record_size</code>	組み込みのファイルベースの持続では、個別のファイルに格納されるメッセージではなく、メッセージストレージファイルに追加されるメッセージの最大サイズを指定する デフォルト値: <code>1m</code> (M バイト)
<code>imq.persist.file.destination.message.filepool.limit</code>	組み込みのファイルベースの持続では、送信先のファイルプールで再利用できる空きファイルの最大数を指定する。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 デフォルト値: <code>100</code>
<code>imq.persist.file.message.filepool.cleanratio</code>	組み込みのファイルベースの持続では、クリーン状態 ( サイズを 0 にする ) で保持される送信先のファイルプールの空きファイルの割合を指定する。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 デフォルト値: <code>0</code>
<code>imq.persist.file.message.cleanup</code>	組み込みのファイルベースの持続では、ブローカのシャットダウン時に、送信先のファイルプール内の空きファイルを削除するかどうかを指定する。値を <code>false</code> に設定すると、ブローカのシャットダウンが速まるが、ファイルを格納するためのディスク容量がさらに必要になる デフォルト値: <code>false</code>

## セキュリティマネージャ

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

認証機能と承認機能はユーザーリポジトリによって異なります (71 ページの図 2-5 を参照)。ユーザーリポジトリには、ファイル、ディレクトリ、または名前、パスワード、グループメンバーシップなどのメッセージングシステムのユーザーに関する情報を含むデータベースがあります。ユーザー名とパスワードはブローカへの接続が要求されたときに、ユーザーを認証するために使用されます。ユーザー名とグループのメンバーシップは、送信先のプロデュースメッセージ、またはコンシューミングメッセージなどの操作を承認するために、アクセス制御ファイルと一緒に使用されます。

Message Queue の管理者は、Message Queue で提供されるユーザーリポジトリ (214 ページの「単層型ファイルユーザーリポジトリを使用する」を参照) を生成するか、あるいは既存の LDAP ユーザーリポジトリをセキュリティマネージャのコンポーネントに組み込みます (221 ページの「ユーザーリポジトリに LDAP サーバを使用する」を参照)。単層型ファイルのユーザーリポジトリは簡単に使用できますが、セキュリティ攻撃に対して脆弱なため、評価や開発を目的とする場合にだけ使用してください。一方、LDAP ユーザーリポジトリは安全なため、実稼働を目的とする場合に適しています。

## 認証

Message Queue のセキュリティは、パスワードベースの認証がサポートしています。クライアントがブローカへの接続を要求する場合、ユーザー名とパスワードを提示する必要があります。セキュリティマネージャは、クライアントから提示されたユーザー名とパスワードをユーザーリポジトリ内に格納されているものと比較します。クライアントからブローカにパスワードが送信される場合、パスワードは、Base-64 か、メッセージダイジェスト (MD) のどちらかを使用して暗号化されます。セキュリティ保護された送信については、72 ページの「暗号化 (Enterprise Edition)」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。

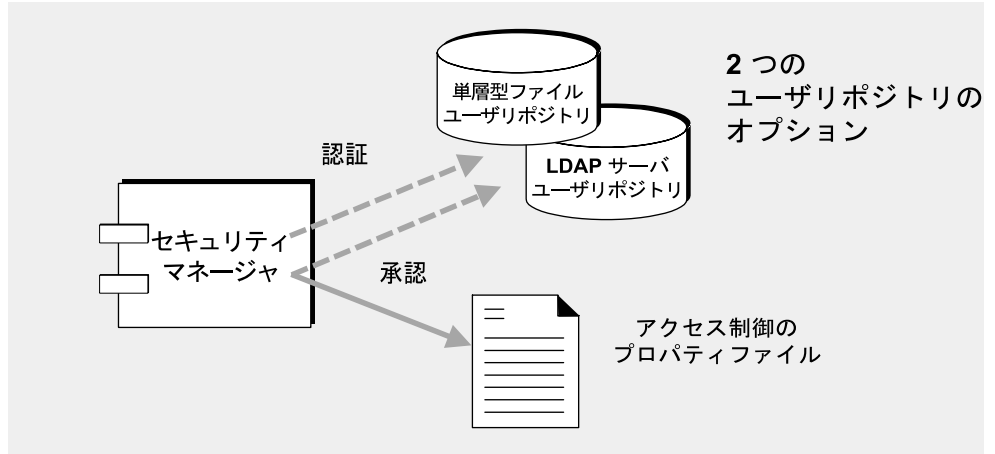
## 承認

クライアントアプリケーションのユーザーが認証されると、ユーザーはさまざまな Message Queue 関連のアクティビティを実行することを承認されます。セキュリティマネージャは、ユーザーベースとグループベースの両方のアクセス制御をサポートしています。ユーザー名、またはユーザーリポジトリでユーザーに割り当てられているグループに応じて、ユーザーは特定の Message Queue 操作を実行するためのアクセス権を付与されます。これらのアクセス制御は、アクセス制御プロパティファイルで指定します (図 2-5 を参照)。

ユーザーがある操作を実行しようとする、セキュリティマネージャが、ユーザーリポジトリ内のユーザー名とグループのメンバーシップを、アクセス制御プロパティファイル内のユーザー名とグループのメンバーシップと照らし合わせます。アクセス制御プロパティには、操作に対するユーザー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。

- ブローカとの接続の確立
- 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



デフォルトのアクセス制御プロパティファイルは、*admin* という 1 つのグループだけを明示的に参照します (217 ページの「グループ」を参照)。 *admin* グループのユーザーには、*admin* サービス接続のアクセス権が付与されます。 *admin* サービスは、送信先の作成、ブローカの監視と制御などの管理機能を実行できます。その他のグループに定義されたユーザーは、デフォルトでは *admin* サービス接続にアクセスできません。

Message Queue の管理者は、グループを定義し、ユーザーリポジトリ内のそれらのグループとユーザーを関連付けることができます。ただし、グループは単層型ファイルのユーザーリポジトリで完全にはサポートされません。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ー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。特定のユーザー、またはグループだけがアクセスできる個別の送信先、またはすべての送信先を作成できます。

さらに、ブローカを設定して、送信先の自動作成 (82 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、ブローカが送信先を自動作成するユーザーを、アクセス制御プロパティファイルを編集して管理することができます。

暗号化 (Enterprise Edition)

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

Message Queue の SSL ベースのコネクションサービスを使用するには、キーツールユーティリティ (imqkeytool) を使用して、非公開キーと公開キーのペアを生成します。このユーティリティは、自己署名型証明書に公開キーを組み込んで、それを Message Queue のキーストアに配置します。Message Queue のキーストア自体は、パスワードによって保護されているため、起動時にキーストアのパスワードを入力して、ロックを解除する必要があります。[230 ページの「暗号化: SSL ベースのサービスとの連動 \(Enterprise Edition\)」](#)を参照

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 2-6           セキュリティマネージャのプロパティ

プロパティ名	説明
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 デフォルト値: 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 デフォルト値: imq.authentication.type の値を継承する
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、 <a href="#">222 ページの表 8-5</a> を参照 デフォルト値: file



表 2-6 セキュリティマネージャのプロパティ ( 続き )

プロパティ名	説明
<code>imq.authentication. client.response.timeout</code>	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 デフォルト値: 180 ( 秒 )
<code>imq.accesscontrol. enabled</code>	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、認証されたユーザーが、コネクションサービスを使用するためのアクセス権、あるいは特定の送信先に対して特定の Message Queue 操作を実行するためのアクセス権を保持していることをシステムでチェックするかどうかを指定する デフォルト値: true
<code>imq.service_name. accesscontrol.enabled</code>	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、認証されたユーザーが、指定したコネクションサービスを使用するためのアクセス権、あるいは特定の送信先に対して特定の Message Queue 操作を実行するためのアクセス権を保持していることをシステムでチェックするかどうかを指定する デフォルト値: <code>imq.accesscontrol.enabled</code> の値を継承
<code>imq.accesscontrol.file. filename</code>	ブローカインスタンスでサポートされるすべてのコネクションサービスに対して、アクセス制御プロパティファイルの名前を指定する。ファイル名には、アクセス制御ディレクトリへの相対ファイルパスを指定する ( 付録 A 「Message Queue データの場所」を参照 ) デフォルト値: <code>accesscontrol.properties</code>
<code>imq.service_name. accesscontrol.file. filename</code>	ブローカインスタンスの指定したコネクションサービスに対して、アクセス制御プロパティファイルの名前を指定する。ファイル名には、アクセス制御ディレクトリへの相対ファイルパスを指定する ( 付録 A 「Message Queue データの場所」を参照 ) デフォルト値: <code>imq.accesscontrol.file.filename</code> の値を継承
<code>imq.passfile.enabled</code>	セキュリティ保護される通信用の (SSL、LDAP、JDBC™ の ) ユーザーパスワードをパスファイルで指定するかどうか (true/false) を指定する デフォルト値: false

表 2-6 セキュリティマネージャのプロパティ ( 続き )

プロパティ名	説明
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。これは、オペレーティングシステムによって異なる デフォルト値: <a href="#">付録 A 「Message Queue データの場所」</a> を参照
imq.passfile.name	パスファイル名を指定する デフォルト値: passfile
imq.keystore.property_name	SSL ベースのサービスの場合は、SSL キーストアに関するセキュリティプロパティを指定する。 <a href="#">232 ページの表 8-8</a> を参照

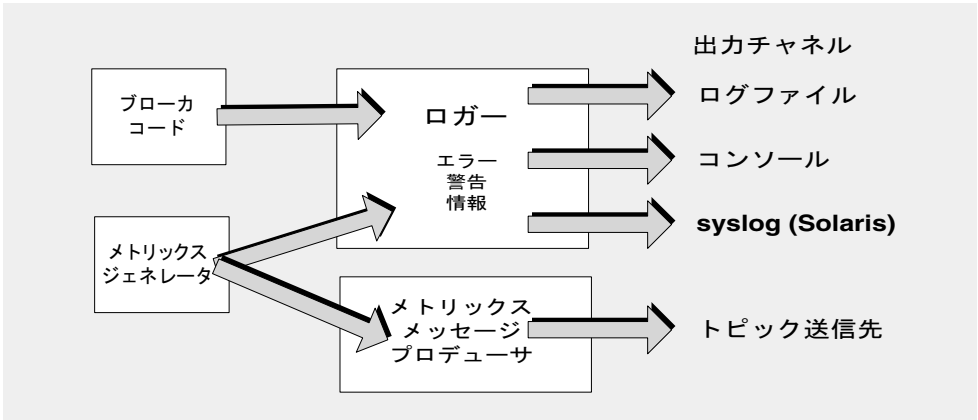
監視サービス

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

- データを生成するコンポーネント ( イベントを記録するブローカコードとメトリックスジェネレータ )
- 多数の出力チャネルを使用して情報を書き込むロガーコンポーネント ( [「ロガー」](#) を参照 )
- メトリックス情報を含む JMS メッセージを、JMS 監視クライアントによってコンシュームさせるためにトピック送信先へ送るメッセージプロデューサ

この仕組みの概略を、[図 2-6](#) に示します。

図 2-6 監視サービスのサポート



## メトリックスジェネレータ

メトリックスジェネレータは、ブローカとの間で入出力されるメッセージフロー、ブローカメモリー内のメッセージ数とそれらが消費するメモリー量、開かれているコネクションの数、使用中のスレッドの数など、ブローカの動作に関する情報を提供します。

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

## ロガー

Message Queue のロガーは、ブローカコードとメトリックスジェネレータによって生成された情報を取得し、標準出力 ( コンソール )、ログファイル、Solaris™ プラットフォーム、syslog デーモンプロセスなどの多数の出力チャネルに書き込みます。

ロガーが収集する情報のタイプと、各出力チャネルに書き込む情報のタイプを指定できます。

たとえば、ロガーが収集する情報のタイプとして、もっとも深刻で重要な情報 ( エラー ) からあまり重要でない情報 ( メトリックスデータ ) までロガーのレベルを指定できます。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 ログイングのカテゴリ

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
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、ログイングに WARNING レベルを指定すると、ロガーは、警告情報とエラー情報の両方を出力します。

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

Solaris syslog の設定および使用方法については、syslog (1M)、syslog.conf (4)、および syslog (3C) のマニュアルページを参照してください。

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、そのログファイルが関連付けられているブローカインスタンスの名前 (*instanceName*) によって識別されたディレクトリに書き込まれます ( [付録 A「Message Queue データの場所」](#) を参照 )。

```
.../instances/instanceName/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。

ローガーの設定方法については、[77 ページの表 2-9](#) と [155 ページの「ローガー設定の変更」](#) を参照してください。

**メトリクスメッセージプロデューサ (Enterprise Edition)**

メッセージプロデューサのコンポーネントは、定期的にメトリクスジェネレータのコンポーネントから情報を受け取り、その情報をメッセージに書き込みます。その後、そのメッセージは、メッセージに含まれるメトリクス情報のタイプに応じて、多数あるメトリクストピック送信先の1つに送信されます。

5つのメトリクストピック送信先があります。それらの名前と、各送信先へ配信されるメトリクスメッセージのタイプを[表 2-8](#)に示します。

**表 2-8**           メトリクスのトピック送信先

トピック送信先名	メトリクスメッセージのタイプ
mq.metrics.broker	ブローカのメトリクス
mq.metrics.jvm	Java 仮想マシンのメトリクス
mq.metrics.destination_list	送信先とそれらのタイプのリスト
mq.metrics.destination.queue. monitoredDestinationName	指定した名前のキューの送信先メトリクス
mq.metrics.destination.topic. monitoredDestinationName	指定した名前のトピックの送信先メトリクス

これらのメトリクストピック送信先にサブスクライブした Message Queue クライアントは、送信先内でメッセージをコンSUMEし、メッセージに含まれるメトリクス情報を処理します。たとえば、クライアントは、mq.metrics.broker 送信先にサブスクライブし、ブローカ内のメッセージの合計数といった情報を受け取り処理することができます。

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

メトリックスメッセージプロデューサによって生成されるメッセージのタイプは、**MapMessage** です。このメッセージは、含まれるメトリックスのタイプに応じて、多数の名前と値のペアから構成されます。各名前と値のペアは、メトリックスの数とその値に相当します。たとえば、ブローカメトリックスメッセージは、ブローカとの間で送受信されたメッセージの数、これらのメッセージのサイズ、現在メモリー内にあるメッセージの数とサイズなど、多数のメトリックスに関する値を含んでいます。各タイプのメトリックスメッセージで報告されるメトリックスの数に関する詳細は、メトリックスメッセージをコンシュームさせる **Message Queue** クライアントのプログラミング方法が説明されている『**Message Queue Java Client Developer's Guide**』を参照してください。

メトリックスメッセージの本体に含まれるメトリックス情報以外に、各メッセージのヘッダーには2つのプロパティがあります。1つはメトリックスメッセージのタイプを指定し、もう1つはタイムスタンプを記録します。**Message Queue** クライアントアプリケーションは、これらのヘッダープロパティを使用して、メトリックスメッセージからデータを抽出し該当するタイムスタンプを記録できます。

### 監視サービスのプロパティ

ブローカによる情報の生成、ロギング、およびメトリックスメッセージの生成を設定する設定可能なプロパティを表 2-9 に示します。各プロパティの設定に関する説明は、[第5章「ブローカの起動と設定」](#)を参照してください。

表 2-9 監視サービスのプロパティ

プロパティ名	説明
<code>imq.metrics.enabled</code>	メトリックス情報をロガーへ書き込むかどうかを指定する (true/false)。メトリックスメッセージの生成への影響はない ( <code>imq.metrics.topic.enabled</code> を参照) デフォルト値: true
<code>imq.metrics.interval</code>	メトリックスのロギングが有効な場合 ( <code>imq.metrics.enabled=true</code> ) は、メトリックス情報がロガーへ書き込まれる時間間隔を秒単位で指定する。値を -1 に設定した場合は、情報が報告されない。メトリックスメッセージの生成の時間間隔には影響しない ( <code>imq.metrics.topic.interval</code> を参照) デフォルト値: -1

表 2-9 監視サービスのプロパティ ( 続き )

プロパティ名	説明
<code>imq.log.level</code>	<p>ロガーレベル、つまり、出力チャネルへ書き込み可能な出力のカテゴリを指定する。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、高いものから降順に、ERROR、WARNING、INFO である</p> <p>デフォルト値: INFO</p>
<code>imq.log.file.output</code>	<p>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</p> <p>デフォルト値: ALL</p>
<code>imq.log.file.dirpath</code>	<p>ログファイルが格納されているディレクトリへのパスを指定する。これは、オペレーティングシステムによって異なる</p> <p>デフォルト値: 付録 A 「Message Queue データの場所」を参照</p>
<code>imq.log.file.filename</code>	<p>ログファイル名を指定する</p> <p>デフォルト値: log.txt</p>
<code>imq.log.file.rolloverbytes</code>	<p>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 -1 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</p> <p>デフォルト値: -1</p>
<code>imq.log.file.rolloversecs</code>	<p>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 -1 に設定した場合、ファイルの有効期間に基づいたロールオーバーは行われない</p> <p>デフォルト値: 604800 (1 週間)</p>
<code>imq.log.console.output</code>	<p>コンソールへ書き込むロギング情報のカテゴリを指定する。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</p> <p>デフォルト値: ERROR  WARNING</p>
<code>imq.log.console.stream</code>	<p>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</p> <p>デフォルト値: ERR</p>

表 2-9 監視サービスのプロパティ ( 続き )

プロパティ名	説明
<code>imq.log.syslog.facility</code>	(Solaris のみ) Message Queue ブローカが記録する <code>syslog</code> 機能を指定する。値は、 <code>syslog(3C)</code> のマニュアルページに示される値をミラー化する。Message Queue で使用できる値は、 <code>LOG_USER</code> 、 <code>LOG_DAEMON</code> 、および <code>LOG_LOCAL0</code> ～ <code>LOG_LOCAL7</code> デフォルト値: <code>LOG_DAEMON</code>
<code>imq.log.syslog.logpid</code>	(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 ( <code>true/false</code> ) デフォルト値: <code>true</code>
<code>imq.log.syslog.logconsole</code>	(Solaris のみ) メッセージを <code>syslog</code> 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 ( <code>true/false</code> ) デフォルト値: <code>false</code>
<code>imq.log.syslog.identity</code>	(Solaris のみ) <code>syslog</code> 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 デフォルト値: <code>imqbrokerd_</code> の後にブローカインスタンス名
<code>imq.log.syslog.output</code>	(Solaris のみ) <code>syslogd(1M)</code> 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、 <code>ALL</code> または <code>NONE</code> デフォルト値: <code>ERROR</code>
<code>imq.log.timezone</code>	ログのタイムスタンプのタイムゾーンを指定する。識別子は、 <code>java.util.TimeZone.getTimeZone()</code> が使用しているものと同じである 次に例を示す。GMT、America/LosAngeles、Europe/Rome、Asia/Tokyo デフォルト値: 該当地域のタイムゾーン
<code>imq.metrics.topic.enabled</code>	メトリックスメッセージの生成を有効にするかどうかを指定する ( <code>true/false</code> )。 <code>false</code> の場合、メトリックストピック送信先へサブスクライブしようとする、クライアント側の例外がスローされる デフォルト値: <code>true</code>

表 2-9 監視サービスのプロパティ ( 続き )

プロパティ名	説明
<code>imq.metrics.topic.interval</code>	メトリックトピック送信先へ送信するメトリックメッセージを生成する時間間隔を秒単位で指定する デフォルト値: 60
<code>imq.metrics.topic.persist</code>	メトリックスメッセージが持続メッセージかどうかを指定する (true/false) デフォルト値: false
<code>imq.metrics.topic.timetolive</code>	メトリックストピック送信先へ送信されるメトリックスメッセージの有効期間を秒単位で指定する。デフォルト値: 300

## 物理的な送信先

Message Queue 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。2 つのタイプの送信先があります ([46 ページの「プログラミングドメイン」](#)を参照)。それらのタイプは、キュー (ポイントツーポイント配信モデル) とトピック (パブリッシュ / サブスクライブ配信モデル) です。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、Message Queue 管理ツールを使用します ([176 ページの「コネクション情報の取得」](#)を参照)。送信先は、[82 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、キューとトピックという 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。メッセージはメッセージプロデューサから到着すると、キューに入って、単一のメッセージコンシューマに配信されます。



## 複数のコンシューマへのキュー配信

1 つのキュー送信先内のメッセージはシングルコンシューマにだけ配信されますが、Message Queue は、複数のコンシューマをキューに登録できます。ブローカは受信メッセージを異なる複数のコンシューマにルートし、それらの間で負荷を分散できます。

複数のコンシューマへのキュー配信を実装するには、次の、キュー送信先属性に基づくロードバランスの方法を使用します。

- `maxNumActiveConsumers`: ロードバランスされたキュー配信でアクティブとなるコンシューマの数を指定する (1 以上)。
- `maxNumBackupConsumers`: アクティブなコンシューマに障害が発生したときに、アクティブコンシューマの機能を引き継ぐバックアップコンシューマの数を指定する (0 以上)。

コンシューマの数がこの 2 つの属性値の合計を超えた場合、新しいコンシューマは拒否されます。(Message Queue Platform Edition はキューあたり最大 3 つのコンシューマ (2 つはアクティブ、1 つはバックアップ) をサポートします。Message Queue Enterprise Edition がサポートするコンシューマの数は無制限です。

ロードバランスメカニズムでは、さまざまなコンシューマがメッセージをコンシュームする度合いを考慮します。このメカニズムは、次のように動作します。

キュー送信先内のメッセージは、設定可能なサイズ (キュー送信先の `consumerFlowLimit` 属性) にまとめられ、新たに使用可能になったアクティブコンシューマにキューに登録された順序でルートされます。これらのメッセージが配信された後、キューに到着した追加メッセージは、コンシューマが使用可能になったとき、つまり設定可能なパーセンテージ分だけ、以前に配信されたメッセージをコンシューマがコンシュームしたときに、一括してそのコンシューマにルートされます。各コンシューマへのディスパッチ速度は、コンシューマの現在の容量とメッセージの処理速度によって異なります。

メッセージの生成速度が遅い場合は、ブローカがアクティブコンシューマ間で不均等にメッセージをディスパッチしている可能性があります。必要な数以上のアクティブコンシューマが存在する場合、一部のコンシューマはメッセージを受信しないことがあります。

アクティブコンシューマに障害が生じると、1 番目のバックアップコンシューマがアクティブになり、障害の生じたコンシューマの動作を引き継ぎます。キュー送信先に複数のアクティブコンシューマがある場合は、メッセージがコンシュームされる順序は保証されません。

ブローカクラスタ環境における複数のコンシューマへの配信では、ローカルコンシューマに高いプライオリティを設定できます。キュー送信先属性である `localDeliveryPreferred` を使用すると、プロデューサのホームブローカ、つまりプロデューサのメッセージ送信先となるブローカ (ローカルブローカ) にコンシューマが存在しない場合にだけ、メッセージをリモートコンシューマへ配信するように指定で

きます。このように指定すると、リモートコンシューマへのルーティング (それらのホームブローカを経由) がスルーブットの低下を招く可能性のある状況で、パフォーマンスを向上させることができます。この属性を使用する場合は、送信先の範囲をローカルだけの配信に制限しないようにしてください (180 ページの表 6-10 を参照)。

### メモリーの考慮事項

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、メモリー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ーが多すぎると、メモリー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。柔軟性を考慮し、各キューの負荷需要に基づいて、キューの作成時に物理的なプロパティを設定できます。このようなプロパティには、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る最大メモリー、キュー内のメッセージの最大サイズがあります (180 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、メモリー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 (180 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

Message Queue 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、リソースを著しく消費する可能性があるため、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、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。

Message Queue は、この自動作成 機能を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ーは、自動作成の権限を保持する必要があります (229 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、自動作成された送信先は、それ以降使用されない状態、つまり、それ以降メッセージコンシューマクライアントを保持せず、メッセージを一切含まない状態になると、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。

Message Queue メッセージサーバでは、表 2-10 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設定できます。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 2-10 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) デフォルト値: 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) デフォルト値: true
imq.autocreate.destination.maxNumMsgs	自動作成された送信先で許容されるコンシュームされないメッセージの最大数を指定する デフォルト値: 100,000
imq.autocreate.destination.maxTotalMsgBytes	送信先でコンシュームされないメッセージ用として許容されるメモリの最大量 (バイト単位) を指定する デフォルト値: 10m (M バイト)

表 2-10 自動作成の設定プロパティ ( 続き )

プロパティ名	説明
<code>imq.autocreate.destination.limitBehavior</code>	<p>メモリー制限のしきい値に達したときのブローカの応答方法を指定する。値は、次のどれかになる</p> <p>FLOW_CONTROL - プロデューサの低速化</p> <p>REMOVE_OLDEST - もっとも古いメッセージの廃棄</p> <p>REMOVE_LOW_PRIORITY - メッセージの有効期限に従い優先度が最低のメッセージを破棄する</p> <p>REJECT_NEWEST - 最新のメッセージを拒否する デフォルト値: REJECT_NEWEST</p>
<code>imq.autocreate.destination.maxBytesPerMsg</code>	<p>自動作成された送信先で許容されるシングルメッセージの最大サイズをバイト単位で指定する デフォルト値: 10k (10,240)</p>
<code>imq.autocreate.destination.maxNumProducers</code>	<p>送信先で許容されるプロデューサの最大数を指定する。この制限に達すると、新しいプロデューサを作成できない デフォルト値: 100</p>
<code>imq.autocreate.destination.isLocalOnly</code>	<p>ブローカクラスタに対してのみ適用。送信先がそのほかのブローカに複製されないように指定する。つまり、メッセージの配信をローカルコンシューマ ( 送信先の作成元にあるブローカに接続されたコンシューマ ) だけに制限する。いったん送信先が作成されると、この属性は更新できない デフォルト値: false</p>
<code>imq.autocreate.queue.maxNumActiveConsumers</code>	<p>自動作成されたキュー送信先からのロードバランスされた配信でアクティブにできるコンシューマの最大数を指定する。値を -1 に設定した場合、無制限になる デフォルト値: 1</p>
<code>imq.autocreate.queue.maxNumBackupConsumers</code>	<p>自動作成されたキュー送信先からのロードバランスされた配信で障害が生じた場合に、アクティブコンシューマに取って代わることができるバックアップコンシューマの最大数を指定する。値を -1 に設定した場合、無制限になる デフォルト値: 0</p>

表 2-10 自動作成の設定プロパティ ( 続き )

プロパティ名	説明
<code>imq.autocreate.queue.consumerFlowLimit</code>	1 つのバッチでコンシューマに配信されるメッセージの最大数を指定する。ロードバランスされたキュー配信では、ロードバランスされる前に、最初にキューに入っていてアクティブコンシューマにルートされるメッセージの数となる (81 ページの「複数のコンシューマへのキュー配信」を参照)。この制限は、各コネクションの送信先のコンシューマに対して小さい値を設定することでオーバーライドできる (『Message Queue Java Client Developer's Guide』のコネクションファクトリ属性の説明を参照)。値を -1 に設定した場合、無制限になる デフォルト値: 1000
<code>imq.autocreate.topic.consumerFlowLimit</code>	1 つのバッチでコンシューマに配信されるメッセージの最大数を指定する。値を -1 に設定した場合、無制限になる。 デフォルト値: 1,000
<code>imq.autocreate.queue.localDeliveryPreferred</code>	ブローカクラスタ内のロードバランスされたキュー配信にだけ適用される。ローカルブローカ上にコンシューマが存在しない場合にだけ、メッセージがリモートコンシューマに配信されるように指定する。自動作成された送信先をローカルだけの配信に制限してはならない ( <code>isLocalOnly = false</code> ) デフォルト値: false

## 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、クライアントが JMS API を使用して明示的に作成および破棄します。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。ただし、Message Queue 管理ツールからは認識できます (177 ページの表 6-9 を参照)。

## マルチブローカクラスタ (Enterprise Edition)

Message Queue Enterprise Edition は、複数の連結したブローカインスタンス、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を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ーの制限などのリソースの限界を超えてしまいます。増え続ける負荷に対処するための1つの方法は、Message Queue メッセージサーバにブローカ、つまり、ブローカインスタンスを追加して、クライアントの接続とメッセージの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、個々のブローカインスタンス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。Message Queue は、障害の生じた接続を、クラスタ内の別のブローカを使用して再確立できますが、状態情報は失われてしまいます。そのため、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。

つまり、現時点では、Message Queue は高可用性メッセージサーバをサポートしていません。ただし、Sun クラスタソフトウェアと高可用性データベースを使用して、ブローカのフェイルオーバーを提供できます。また、複数のブローカを使用して、カスタマイズされたフェイルオーバーソリューションを実装するように、メッセージングアプリケーションを設計することもできます。

ブローカのクラスタの設定および管理については、[147 ページの「クラスタの操作 \(Enterprise Edition\)」](#)で説明します。

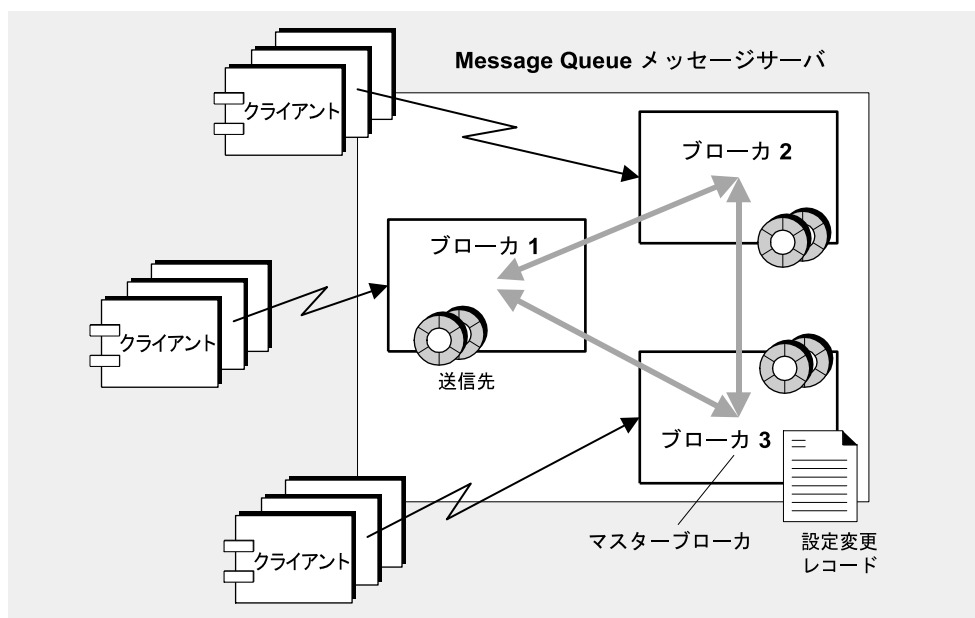
次の節では、Message Queue のブローカのクラスタのアーキテクチャと内部機能について説明します。

### マルチブローカのアーキテクチャ

[図 2-7](#) に示すように、マルチブローカのメッセージサーバを使用すると、クライアントの接続を多数のブローカインスタンス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。

理論的には、クラスタ内のブローカに、任意のトポロジで接続することは可能です。ただし、[図 2-7](#) に示すように、Message Queue 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ブローカすべてに直接接続するトポロジだけをサポートしています。

図 2-7 マルチブローカ (クラスタ) のアーキテクチャ



## メッセージ配信

マルチブローカの設定では、各送信先がクラスタ内のすべてのブローカで複製されます。一部の例外を除き、クラスタ環境内の送信先属性は一括して送信先のすべてのインスタンス、つまり、個々の送信先インスタンスではなくクラスタ全体に適用されます。また、`isLocalOnly` 属性が `true` に設定された送信先は、クラスタ内で複製されません。詳細は、[180 ページの表 6-10](#) の送信先属性の説明を参照してください。

各ブローカは、ほかのすべてのブローカの送信先に対して登録されたメッセージコンシューマを認識しています。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。

クラスタ内のブローカ間のトラフィックを最小限に抑えるため、送信先からクライアントランタイムへのメッセージの配信は、コンシューマ接続のフロー制御メカニズムによって制限されています。この方法で、メッセージは、ターゲットブローカに接続されたコンシューマに配信される場合にだけブローカ間で送信されるため、ブローカ間での不要なメッセージの受け渡しをなくすることができます。また、複数のコンシューマへのキュー配信など、場合によっては、ローカルコンシューマへの配信にリモートコンシューマへの配信より高いプライオリティを指定することで、ブローカ間のトラフィックを最小限にすることもできます ([180 ページの表 6-10](#) の

localDeliveryPreferred キュー送信先属性を参照)。

クライアントとメッセージサーバ間に安全で暗号化されたメッセージ配信が必要となる場合は、クラスタ内のブローカ間のメッセージ配信も保護するようにクラスタを設定できます ([150 ページの「ブローカ間の安全な接続」](#) を参照)。

## クラスタの同期化

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。

---

<b>注</b>	<p>負荷の高いネットワークトラフィックやサイズの大きいメッセージは、クラスタ内の接続を妨げることがあります。遅延が多くなると、ロックプロトコルがタイムアウトになることがあります。その結果、クライアントは、永続サブスクリバまたはキューメッセージコンシューマを作成しようとしたときに、例外を受け取ることがあります。通常、これらの問題は、高速のコネクションを使用することによって回避できます。</p>
----------	--

---

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。

オフラインになったブローカや新たに追加されたブローカを収容するため、Message Queue は、クラスタ内のすべての持続エンティティに加えられた変更の記録を維持します。この記録は、作成または破棄されたすべての送信先とすべての永続サブスクリプションに関する記録です。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リ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に管理されます。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プ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、Message Queue 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（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 (143 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 (147 ページの表 5-3 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、Message Queue 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 (82 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、Message Queue 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、共通のクラスタ設定プロパティを使用する必要があります。この共通化は、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。

また、クラスタ設定プロパティを複製して、個別のブローカに提供することもできます。ただし、これを行うとクラスタ設定に矛盾を引き起こすことがあるため、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。

クラスタ設定プロパティの詳細については、[147 ページの「クラスタの操作 \(Enterprise Edition\)」](#)を参照してください。

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。

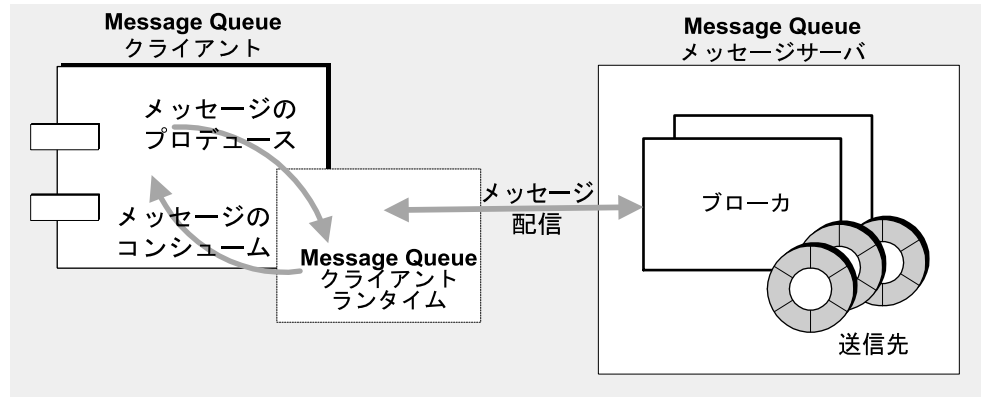
# Message Queue クライアントランタイム

Message Queue クライアントランタイムは、クライアントアプリケーションに Message Queue メッセージサービスへのインタフェースを提供します。つまり、クライアントランタイムによって、[40 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 Java クライアントアプリケーションに提供され、該当する C インタフェースが C クライアントアプリケーションに提供されます。Message Queue 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します。

この節では、Message Queue 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。クライアントアプリケーションの設計と、Java クライアントランタイムおよび C クライアントランタイムのパフォーマンスに影響を及ぼす要因については、それぞれ『Message Queue Java Client Developer's Guide』と『Message Queue C Client Developer's Guide』で説明されています。

図 2-8 に、クライアントアプリケーションと Message Queue 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 Message Queue クライアントランタイムと Message Queue 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、クライアントによってメッセージが作成され、ブローカ上の送信先への接続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、持続的 (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保存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がブロックされます。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。

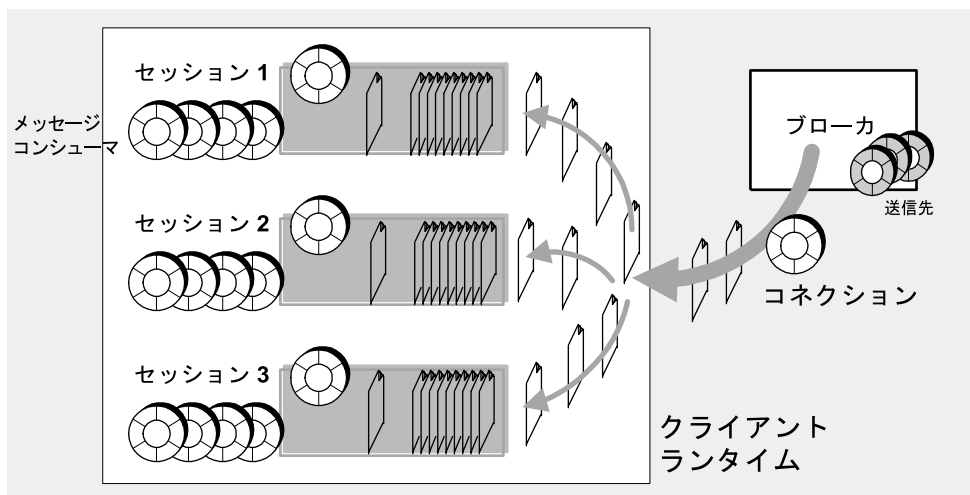
## メッセージのコンシューム

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 Message Queue クライアントランタイムへの接続で配信されます。

- 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、接続が指示されている

図 2-9 に示されるように、接続で配信されるメッセージは、適切な Message Queue セッションに分散されます。ここで、メッセージは適切な MessageConsumer オブジェクトによってコンシュームされるために、キューに入れられます。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 Message Queue クライアントランタイムへのメッセージの配信



ブローカがクライアントランタイムにメッセージを配信する場合、ブローカはそれに応じてメッセージをマークしますが、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。

# Message Queue 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。

Message Queue は、ConnectionFactory と Destination の 2 つのタイプの管理対象オブジェクトを提供します。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、Message Queue 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 (197 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 (82 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 (197 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属性」を参照) することによって、Message Queue 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、Message Queue 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 (45 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」に記載された Message Queue 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 Message Queue 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を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。また、[96 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、管理対象オブジェクトの基本目的 (Message Queue 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 Message Queue 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 ([49 ページの「ローカルトランザクション」](#)を参照) をサポートする場合は、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、[206 ページの「コネクションファクトリの追加」](#)を参照してください。

ConnectionFactory 管理対象オブジェクトを構成することによって、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
- 自動再コネクションの動作
- 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、『Message Queue Java Client Developer's Guide』で詳しく説明します。Message Queue 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。198 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、207 ページの「トピックまたはキューの追加」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。文字列「mq.」で開始することはできない 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。 デフォルト値: 送信先オブジェクトの説明

## クライアントの起動時の属性値のオーバーライド

Java アプリケーションだけでなく、メッセージングアプリケーションの起動時にも、コマンド行からシステムプロパティを指定できます。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、[197 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属性](#)」に記載されている任意の `ConnectionFactory` 管理対象オブジェクトの属性です。

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。オーバーライドの指定は無視されます。



# Message Queue の管理タスクと管理ツール

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

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

## Message Queue の管理タスク

開発環境、または運用環境のどちらに在るかによって、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 Message Queue 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。Message Queue 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、柔軟性が重視されるため、通常は次の要件を適用します。

- 開発者がテストで使用するブローカを起動する程度に、管理の手間を最小限に抑える
- データストア (ファイルベースの組み込み持続)、ユーザーリポジトリ (ファイルベースのユーザーリポジトリ)、アクセス制御プロパティファイル、オブジェクトストア (ファイルシステムのストア) のデフォルトの環境を使用する。通常の開発テストでは、これらのデフォルトの実装で十分である
- 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ない
- 一般に、明示的に作成した送信先でなく、自動作成された送信先を使用する
- 集中管理された管理対象オブジェクトではなく、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することがある

## 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

### ► 運用環境を設定するには

通常、次のセットアップ操作のうち少なくとも一部を実行する必要があります。

- **管理者のセキュリティ** (保護された管理ツールの使用)
  - 管理コネクションサービスがアクティブであることを確認する (59 ページの表 2-3 を参照)
  - 承認: 特定の個人または管理グループに対して管理コネクションサービスへのアクセスを許可する (227 ページの「コネクションアクセス制御」を参照)
  - グループに対して承認を実行する場合は、管理者が **admin** グループに属していることを確認する
  - ファイルベースのユーザーリポジトリ。デフォルトの **admin** グループを持つ。管理者が **admin** グループに属していることを確認するか、デフォルトの **admin** ユーザーを使用する場合は、**admin** パスワードを変更する (220 ページの「デフォルトの管理者パスワードの変更」を参照)
  - LDAP ユーザーリポジトリ。管理者が **admin** グループに属していることを確認する
- **全般的なセキュリティ** (第 8 章「セキュリティの管理」を参照)
  - 認証: ファイルベースのユーザーリポジトリにエントリを作成するか、あるいはブローカを設定して、既存の LDAP ユーザーリポジトリを使用する  
 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
  - 承認: アクセス制御プロパティファイルのアクセス設定を変更する
  - 暗号化: 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** (147 ページの「クラスタの操作 (Enterprise Edition)」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 B「プラグイン持続の設定」を参照)
- **メモリー管理**: メッセージ数とメッセージに割り当てられるメモリー量が使用可能なブローカのメモリーリソース内に納まるように送信先属性を設定する (180 ページの表 6-10 を参照)

## メンテナンス操作

### ► 運用環境を設定するには

さらに、運用環境では、Message Queue 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 Message Queue の管理ツールを使用して実行する必要があります。

- **アプリケーション管理**
  - ブローカの自動作成機能を無効にする (83 ページの表 2-10 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 (179 ページの「送信先の作成」を参照)
  - 送信先へのユーザーアクセスを設定する (224 ページの「ユーザーの承認: アクセス制御プロパティファイル」を参照)
  - 送信先を監視および管理する (177 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 (188 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 (189 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 (237 ページの第 9 章「メッセージサービスの分析と調整」を参照)
  - ブローカのメモリーリソースを管理する (237 ページの第 9 章「メッセージサービスの分析と調整」を参照)
  - クラスタにブローカを追加して、ロードバランスを実行する (147 ページの「クラスタの操作 (Enterprise Edition)」を参照)
  - 障害が発生したブローカを復元する (141 ページの「ブローカの起動」を参照)
- **管理対象オブジェクトの管理**
  - 必要に応じて、追加のコネクションファクトリと送信先の管理対象オブジェクトを作成する (206 ページの「管理対象オブジェクトの追加および削除」を参照)

- コネクションファクトリ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 (197 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属性」および 211 ページの「管理対象オブジェクトの更新」を参照)

## Message Queue 管理ツール

Message Queue の管理ツールは 2 つの種類に分けられます。それは、コマンド行ユーティリティとグラフィカルユーザーインターフェース (GUI) の管理コンソール (imqadmin) です。コンソールは、コマンドユーティリティ (imqcmd) とオブジェクトマネージャユーティリティ (imqobjmgr) の 2 つのコマンド行ユーティリティの機能が組み合わされたものです。コンソール (および、これらの 2 つのコマンド行ユーティリティ) を使用すると、ブローカをリモートで管理したり、Message Queue の管理対象オブジェクトを管理したりすることができます。その他のコマンド行ユーティリティ (imqbrokerd、imqusermgr、imqdbmgr、および imqkeytool) は、101 ページの図 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
- ブローカで物理的送信先を作成および管理する
- オブジェクトストアに接続する
- オブジェクトストアに管理対象オブジェクトを追加し、それらを管理する

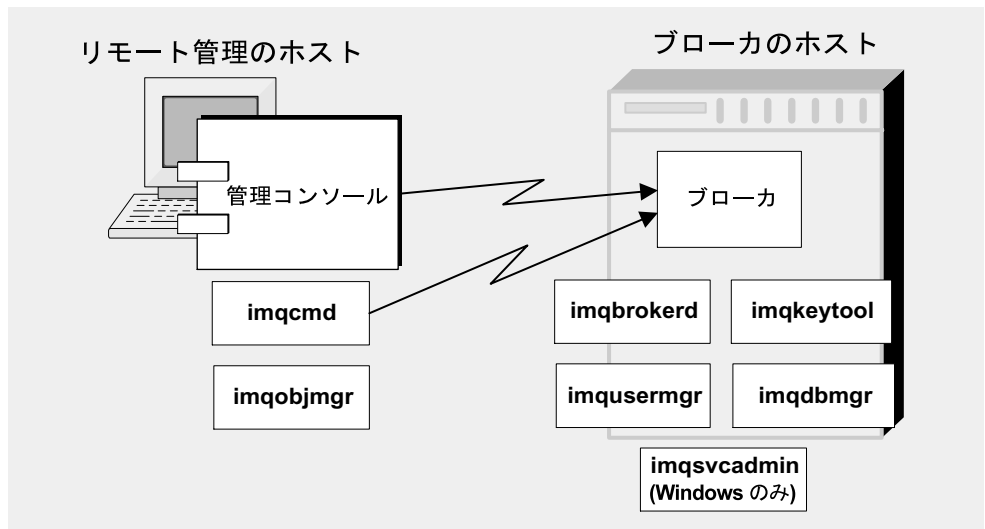
管理コンソールでは、実行できないタスクがいくつかあります。これには、ブローカの起動、ブローカのクラスタの作成、ブローカと物理的な送信先の特殊なプロパティの設定、ユーザーデータベースの管理などが挙げられます。

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

図 3-1 ローカルおよびリモートの管理ユーティリティ



Message Queue のユーティリティはすべて、コマンド行インタフェース (CLI) からアクセスできます。ユーティリティコマンドは、この章の後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。imqbrokerd コマンドについては、[第 5 章「ブローカの起動と設定」](#)で説明されています。

**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接続サービスの管理、およびブローカのリソースの管理を行います。imqcmd コマンドについては、[第 6 章「ブローカとアプリケーションの管理」](#)で説明されています。

**オブジェクトマネージャ (imqobjmgr):** オブジェクトマネージャユーティリティを使用して、JNDI を介したアクセスが可能なオブジェクトストア内の管理対象オブジェクトの追加、一覧表示、更新、および削除を行います。管理対象オブジェクトを使用すると、JMS プロバイダ固有の命名および設定形式から独立するため、JMS クライアントはプロバイダに依存しなくなります。imqobjmgr コマンドについては、[第7章「管理対象オブジェクトの管理」](#)で説明されています。

**ユーザーマネージャ (imqusermgr):** ユーザーマネージャユーティリティを使用して、ユーザーの認証および承認に使用するファイルベースのユーザーリポジトリを設定します。imqusermgr コマンドについては、[第8章「セキュリティの管理」](#)で説明されています。

**キーツール (imqkeytool):** キーツールユーティリティを使用して、SSL 認証で 사용되는自己署名型証明書を生成できます。imqkeytool コマンドについては、[第8章「セキュリティの管理」](#)および[付録C「HTTP/HTTPS サポート \(Enterprise Edition\)」](#)で説明されています。

**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。imqdbmgr コマンドについては、[付録B「プラグイン持続の設定」](#)で説明されています。

**サービス管理 (imqsvcadm):** サービス管理ユーティリティを使用して Windows のサービスとして、ブローカをインストール、クエリー、および削除します。imqsvcadm コマンドについては、[付録D「Windows のサービスとしてのブローカの使用」](#)で説明されています。

## コマンド行の構文

Message Queue 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] [argument] [[-option_name [-option_argument]]...]
```

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

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、サブコマンドと引数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、[103 ページの表 3-1](#) を参照
- サブコマンド、引数、オプション、オプションの引数はスペースで区切る

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 (ユーザー) 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

Message Queue の管理ユーティリティ全体に共通するオプションを[表 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 Message Queue コマンド行オプション

オプション	説明
-h	特定のユーティリティの使用方法に関するヘルプを表示する
-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	サイレントモードに切り替える。出力は表示されない。imqbrokerd に <code>-silent</code> として指定する
-v	バージョン情報を表示する
-f	ユーザー確認の要求なしで、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 <i>path</i>	使用する代替の Java 2 互換のランタイムを指定する。デフォルトではシステム上のランタイムまたは Message Queue にバンドルされたランタイムを使用する





# 管理コンソールのチュートリアル

この章では、Message Queue 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

チュートリアルでは、単純な JMS 互換のアプリケーションであり、アプリケーション例のある /demo ディレクトリの helloworld サブディレクトリにある HelloWorldMessageJNDI を実行するために必要な送信先と管理対象オブジェクトを設定します ( 付録 A 「Message Queue データの場所」を参照 )。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、『Message Queue Java Client Developer's Guide』や、管理ガイドのほかの章とは内容が異なります。

一部の Message Queue 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。

- 特定の物理的な送信先プロパティを設定する  
一部の物理的な送信先プロパティは、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、177 ページの「送信先の管理」の手順を実行すると、設定できます。
- ブローカクラスタを作成する  
詳細は、147 ページの「クラスタの操作 (Enterprise Edition)」を参照
- ユーザーのデータベースを管理する

詳細は、[214 ページの「ユーザーの認証」](#)を参照

## 準備

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

この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
- Message Queue 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、[93 ページの「Message Queue 管理対象オブジェクト」](#)を参照

### ► 管理コンソールを起動するには

1. 「スタート」>「プログラム」>「Sun Java System Message Queue 3.5 SP1」>「Administration」を選択します。

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。

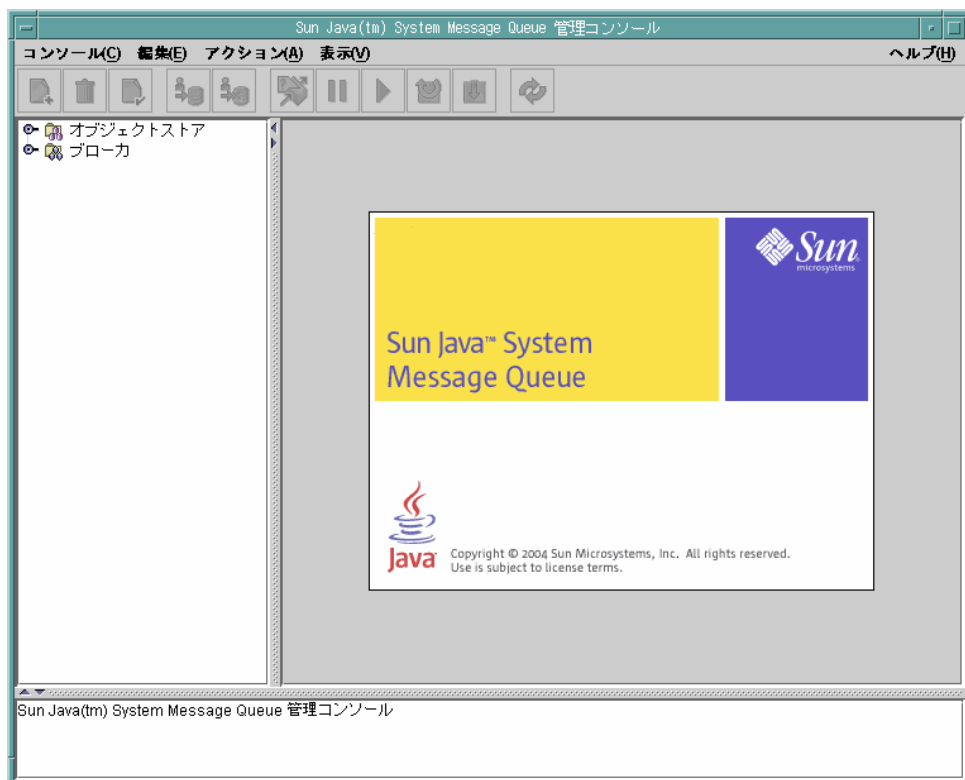
**Windows 以外のユーザー:** コマンドプロンプトで次のとおり入力します。

`/usr/bin/imqadmin` (Solaris の場合)

`/opt/imq/bin/imqadmin` (Linux の場合)

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、一番上にメニューバー、メニューバーのすぐ下にツールバー、左側にナビゲーションのペイン、右側に結果を表すペイン(この図では Sun Java System 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。



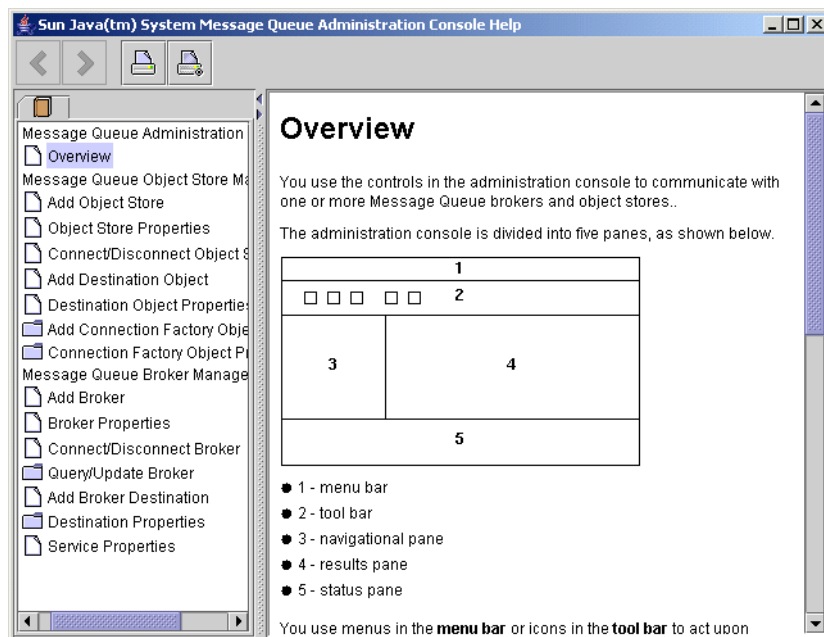
チュートリアルには完全な情報が記載されているわけではないので、まず管理コンソールに関するヘルプ情報の使い方を理解しておきましょう。

## 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。左のナビゲーションペインには目次が表示され、右の結果ペインにはナビゲーションペインで選択したアイテムの内容が表示されます。

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

2. ヘルプウィンドウのナビゲーションペインを見ます。トピックが、概要、オブジェクトストアの管理、ブローカの管理の3つの分野に分類されています。それぞれの領域に、ファイルやフォルダがあります。各フォルダには、複数タブのついたダイアログボックスのヘルプがあり、各ファイルはダイアログボックス、またはタブの簡単なヘルプです。

最初に行うコンソールの管理タスクは、[111 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。

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

結果ペインが変更されています。ここには、ブローカを追加するとはどういうことかを説明するテキストと、「Add Broker (ブローカを追加)」ダイアログボックスにある各フィールドの使用方法が表示されます。フィールド名は、太字で表示されます。

- ヘルプテキストを読みます。
- 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、Message Queue 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 階層のプロセスです。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 つ以上のコンシューミング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 `mqbrokerd` コマンドを使用または行います。`mqbrokerd` 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。

---

注	管理コンソールを使用してブローカインスタンスを起動することはありません。
---	--------------------------------------

---

- 管理コンソールまたは `mqcmd` コマンド行ユーティリティを使用して、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、56 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

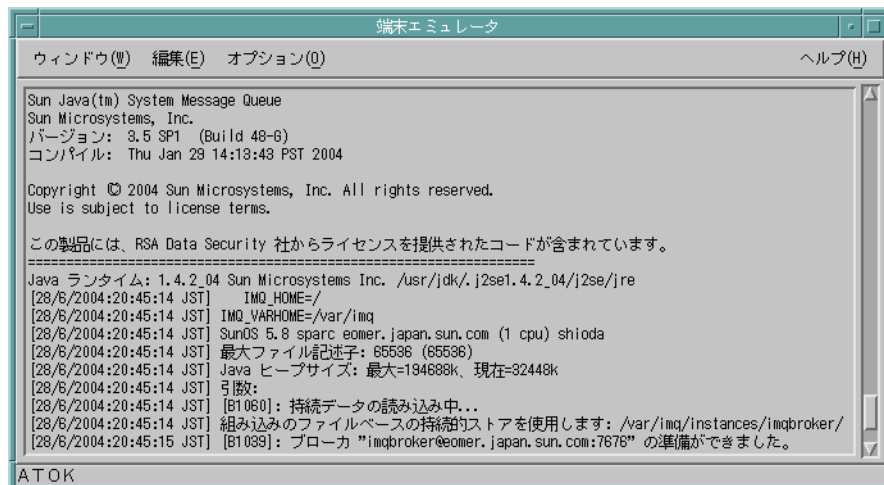
1. 「スタート」>「プログラム」>「Sun Java System Message Queue3.5 SP1」>「Message Broker」を選択します。

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

`/usr/bin/imqbrokerd` (Solaris の場合)

`/opt/imq/bin/imqbrokerd` (Linux の場合)

コマンドプロンプトウィンドウが表示されます。これは、ブローカが使用可能であることを示しています。



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。

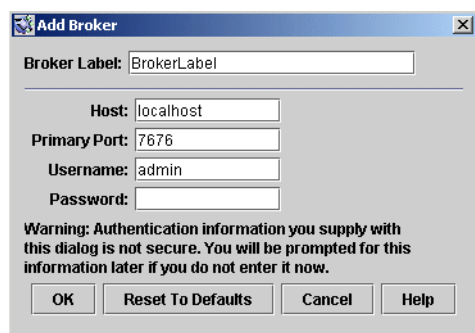
管理コンソールでブローカへの参照を追加する前に、ブローカを起動する必要はありませんが、接続する前にはブローカを起動する必要があります。

## 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。

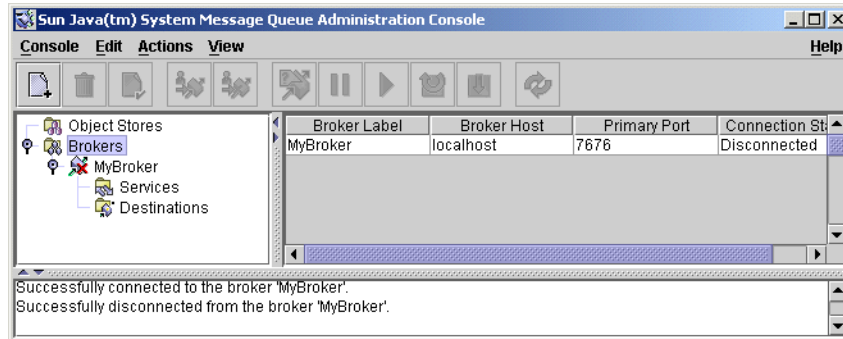


ダイアログボックスで指定されたデフォルトのホスト名 (localhost) と、プライマリポート (7676) を控えておきます。あとで、クライアントがこのブローカへのコネクションを設定するのに必要なコネクションファクトリを設定するときに、この値を指定する必要があります。

「Password (パスワード)」フィールドは空欄のままにしておきます。パスワードは、コネクションを実行するときに指定したほうがより安全です。

3. 「OK」をクリックして、ブローカを追加します。

ナビゲーションペインを見ます。追加したブローカが、「Brokers (ブローカ)」の下に一覧表示されています。ブローカアイコンの上についている赤い×印は、そのブローカが現在コンソールに接続されていないことを表しています。



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

ブローカのプロパティダイアログボックスが表示されます。このダイアログボックスを使用して、ブローカの追加時に指定したプロパティを更新できます。

5. 「Cancel (取り消し)」をクリックしてダイアログを閉じます。

## 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。デフォルトでは、管理コンソールは admin というパスワードを持つ admin というユーザーとしてブローカに接続できます。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
```

変更は直ちに反映されます。Message Queue 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。



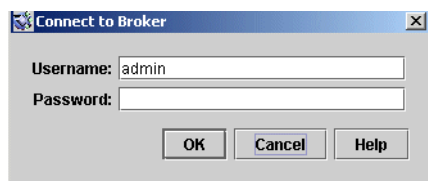
クライアントは、管理者と異なるコネクションサービスを使用しますが、デフォルトのユーザー名とパスワードが割り当てられているため、詳細な管理の設定を行わなくても Message Queue をテストできます。デフォルトでは、クライアントは guest というパスワードを持つ guest というユーザーとしてブローカに接続できます。ただし、できるだけ早くクライアントに安全なユーザー名とパスワードを設定してください。詳細は、[214 ページ](#)の「[ユーザーの認証](#)」を参照してください。

## 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」を選択します。

ユーザー名とパスワードを指定するダイアログボックスが表示されます。



2. 「Password」フィールドに admin または [112 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。

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

3. 「OK」をクリックして、ブローカに接続します。

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

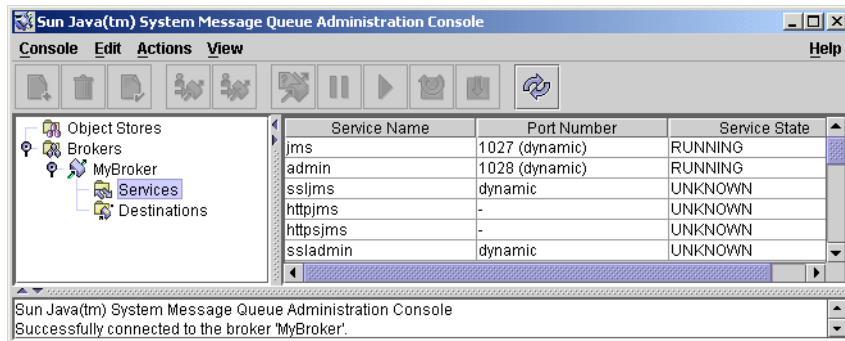
## コネクションサービスを表示する

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

### ► 利用可能なコネクションサービスを表示するには

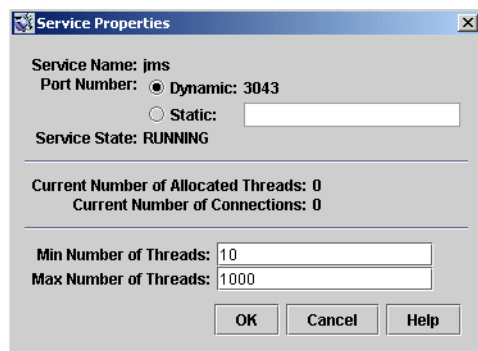
1. ナビゲーションペインで「Services ( サービス )」を選択します。

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 ( サービスプロパティ )」ダイアログを使用して、サービスに静的なポート番号を割り当て、このサービスに割り当てられるスレッドの最小数および最大数を変更できます。



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

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

## ブローカに物理的送信先を追加する

デフォルトでは、送信先の自動作成はブローカに対して有効になっています。そのためブローカの物理的な送信先を動的に作成できます。そのため、開発環境で、クライアントコードをテストするための送信先を明示的に作成する必要はありません。

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

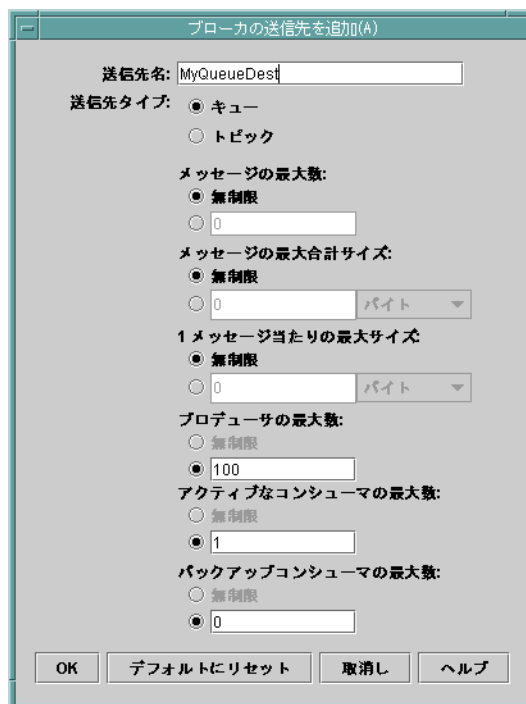
`imq.autocreate.topic` または `imq.autocreate.queue` プロパティを設定して、ブローカの自動作成送信先の追加を制御できます。詳細は、[82 ページの「自動作成 \(および管理者によって作成\) された送信先」](#) を参照してください。

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。

次のダイアログボックスが表示されます。



ブローカの送信先を追加(A)

送信先名: MyQueueDest

送信先タイプ: ☒ キュー  
☐ トピック

メッセージの最大数:  
☒ 無制限  
☐ 0

メッセージの最大合計サイズ:  
☒ 無制限  
☐ 0 バイト

1メッセージ当たりの最大サイズ:  
☒ 無制限  
☐ 0 バイト

プロデューサの最大数:  
☐ 無制限  
☒ 100

アクティブなコンシューマの最大数:  
☐ 無制限  
☒ 1

バックアップコンシューマの最大数:  
☐ 無制限  
☒ 0

OK デフォルトにリセット 取消し ヘルプ

2. 「Destination Name (送信先名)」フィールドに MyQueueDest と入力します。
  3. 選択されていない場合は、「Queue (キュー)」ラジオボタンを選択します。
  4. 「OK」をクリックして、物理的送信先を追加します。
- 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、以下の手順に従い、次のタスクを実行できるようになります。

- 物理的送信先のプロパティを表示および更新する
- 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。

次のダイアログボックスが表示されます。

ブローカの送信先のプロパティ

基本 **名前とスクリプション**

送信先名: MyQueueDest  
送信先タイプ: キュー  
送信先の状態: RUNNING

---

現在のメッセージ数: 0  
現在のメッセージの合計サイズ: 0 バイト  
現在のプロデューサの数: 0  
現在のアクティブなコンシューマの数: 0  
現在のバックアップコンシューマの数: 0

---

メッセージの最大数: ☒ 無制限  
☐ 0

メッセージの最大合計サイズ: ☒ 無制限  
☐ 0 バイト ▼

1メッセージ当たりの最大サイズ: ☒ 無制限  
☐ 0 バイト ▼

プロデューサの最大数: ☒ 無制限  
☒ 100

アクティブなコンシューマの最大数: ☒ 無制限  
☒ 1

バックアップコンシューマの最大数: ☒ 無制限  
☒ 0

OK 取消し ヘルプ

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

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

#### ► 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。

確認ダイアログボックスが表示されます。

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

#### ► 送信先を削除するには

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。

確認ダイアログボックスが表示されます。

---

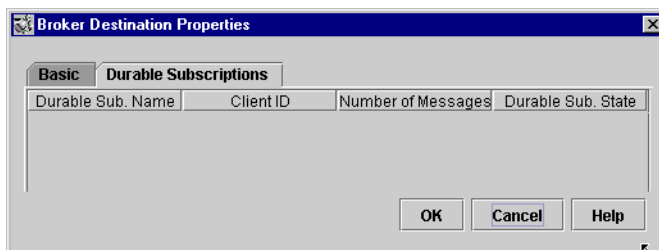
**注** MyQueueDest キュー送信先は削除しないでください。

---

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

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



このダイアログボックスを使用して、次のことを実行できます。

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

## オブジェクトストアを操作する

オブジェクトストアは、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、Message Queue 管理対象オブジェクトを格納するために使用されます。この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 Message Queue 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、[93 ページの「Message Queue 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。

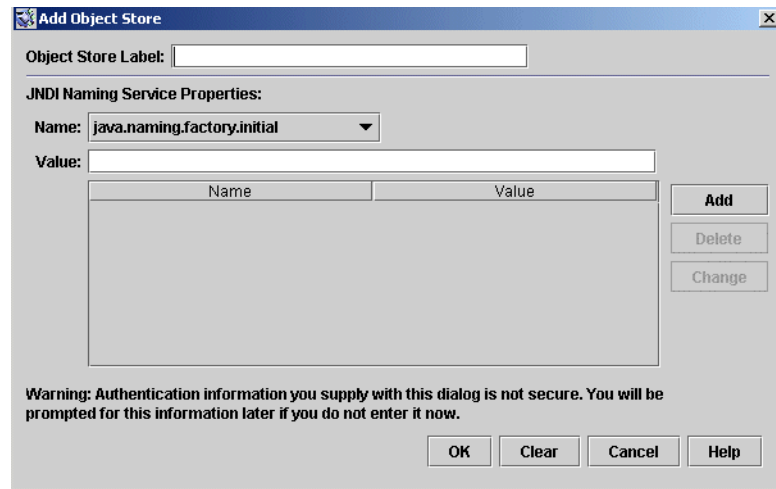
このチュートリアルで使用するサンプルアプリケーションでは、オブジェクトストアが C ドライブの Temp というフォルダにあることが前提になっています。一般には、ファイルシステムオブジェクトストアは、任意のドライブの任意のディレクトリに置くことができます。

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



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

次のダイアログボックスが表示されます。



The dialog box titled "Add Object Store" contains the following elements:

- Object Store Label:** A text input field.
- JNDI Naming Service Properties:**
  - Name:** A dropdown menu with "java.naming.factory.initial" selected.
  - Value:** A text input field.
- Table:** A table with two columns, "Name" and "Value", and an empty body.
- Buttons:** "Add", "Delete", and "Change" buttons are located to the right of the table.
- Warning:** A warning message at the bottom states: "Warning: Authentication information you supply with this dialog is not secure. You will be prompted for this information later if you do not enter it now."
- Footer Buttons:** "OK", "Clear", "Cancel", and "Help" buttons are at the bottom right.

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

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

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

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

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

5. 「Value ( 値 )」フィールドに、次のように入力します。

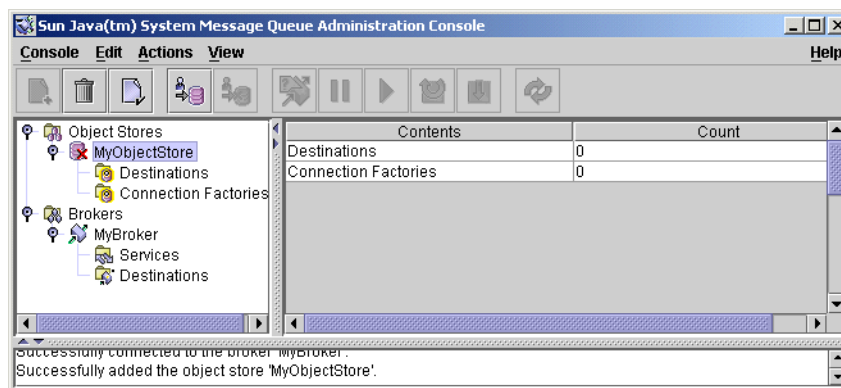
```
com.sun.jndi.fscontext.RefFSContextFactory
```

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

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、[194 ページの「LDAP サーバオブジェクトストア」](#)を参照してください。

6. 「Add ( 追加 )」 ボタンをクリックします。  
プロパティの要約ペインにプロパティとその値が一覧表示されます。
7. 「Name ( 名前 )」 ドロップダウンリストから、`java.naming.provider.url` を選択します。  
このプロパティで、オブジェクトストアの正確な場所を指定できます。ファイルシステムのオブジェクトストアでは、これが既存のディレクトリ名になります。
8. 「Value ( 値 )」 フィールドに、次のように入力します。  
`file:///C:/Temp`  
(Solaris と Linux 上では、`file:///tmp` )
9. 「Add ( 追加 )」 ボタンをクリックします。  
プロパティの要約ペインにプロパティとその値がともに一覧表示されます。  
LDAP サーバ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ボックスが表示されます。任意のプロパティを変更し、「OK」をクリックして古い情報を更新します。

3. 「OK」をクリックするか、「Cancel (取消し)」をクリックして、ダイアログボックスを終了します。

## オブジェクトストアに接続する

オブジェクトストアにオブジェクトを追加する前に、まずオブジェクトストアに接続する必要があります。

### ▶ オブジェクトストアに接続するには

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『[Message Queue Java Client Developer's Guide](#)』を参照してください。

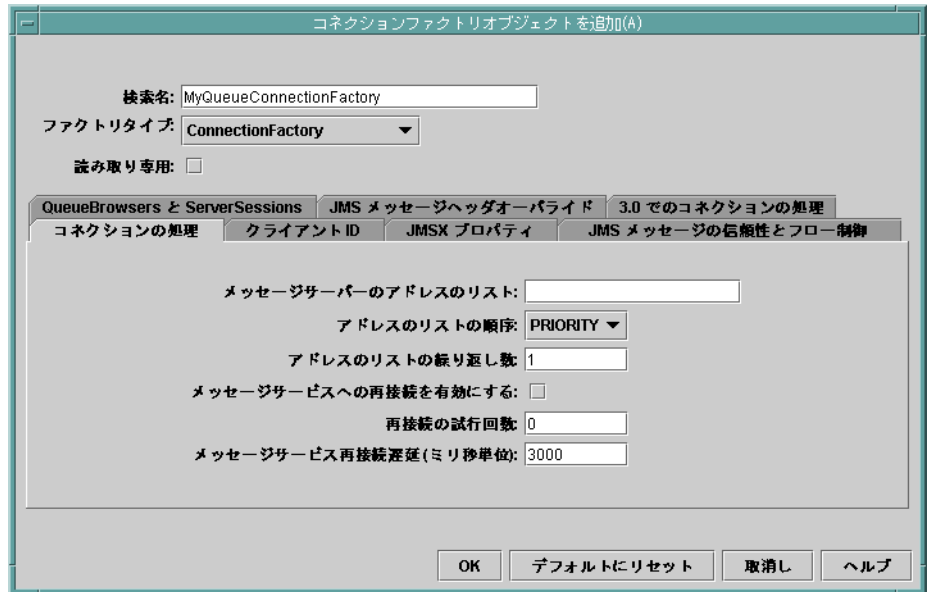
---

**注**                    管理コンソールは、**Message Queue** 管理対象オブジェクトだけを一覧し表示します。オブジェクトストアに、追加したい管理対象オブジェクトと同じ検索名の **Message Queue** 以外のオブジェクトが含まれている場合は、追加操作を実行するとエラーが表示されます。

---

### ► コネクションファクトリをオブジェクトストアに追加するには

1. まだ接続されていない場合は、MyObjectStore に接続します ([124 ページの「オブジェクトストアに接続する」](#)を参照)。
2. 「Connection Factories (コネクションファクトリ)」ノードを右クリックし、「Add Connection Factory Object (コネクションファクトリオブジェクトを追加)」を選択します。  
  
「Add Connection Factory Object (コネクションファクトリオブジェクトを追加)」ダイアログボックスが表示されます。



3. 「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 HelloWorldMessageJNDI.java から接続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、接続ファクトリのタイプを指定します。
5. 「接続の処理」タブをクリックします。
6. 「Message Server Address List ( メッセージサーバアドレスリスト )」フィールドには、通常、クライアントの接続先となるブローカのアドレスを入力します。このフィールドの例を次に示します。

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

デフォルトでは、接続ファクトリは、ポート 7676 の localhost で実行しているブローカに接続するように設定されているため、値を入力する必要はありません。チュートリアル例でも、このデフォルト設定を前提にしています。

7. このダイアログボックスのタブをクリックして、コネクションファクトリの設定可能な情報の種類を確認します。「Add Connection Factory Object (コネクションファクトリオブジェクトを追加)」ダイアログボックスの右側下部にある「Help (ヘルプ)」ボタンを使用して、それぞれのタブの情報を確認します。ここでは、デフォルト値を変更しないでください。
8. 「OK」をクリックして、キューコネクションファクトリを作成します。
9. 結果ペインを確認します。新規作成されたコネクションファクトリの検索名と検索タイプが一覧表示されています。

## 送信先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。

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。

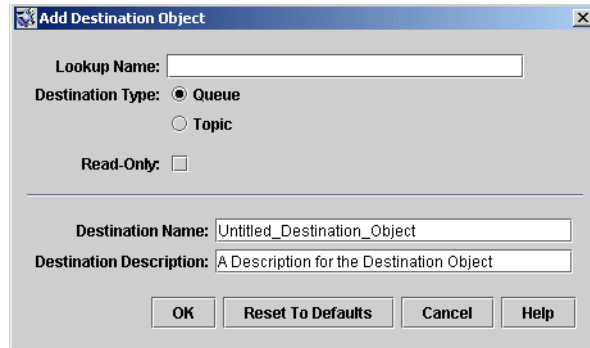
- その管理対象オブジェクトに関連する物理的送信先を作成した場合、ブローカはその物理的送信先にメッセージを配信する
- 物理的送信先を作成せず、物理的送信先の自動作成が有効になっている場合、ブローカ自身が物理的送信先を作成し、その送信先にメッセージを配信する
- 物理的送信先を作成せず、物理的送信先の自動作成が無効になっている場合、ブローカは物理的送信先を作成できず、メッセージも配信できない

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「MyObjectStore」ノードの下にある「Destinations (送信先)」ノードを右クリックします。
2. 「Add Destination Object (送信先オブジェクトを追加)」を選択します。

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

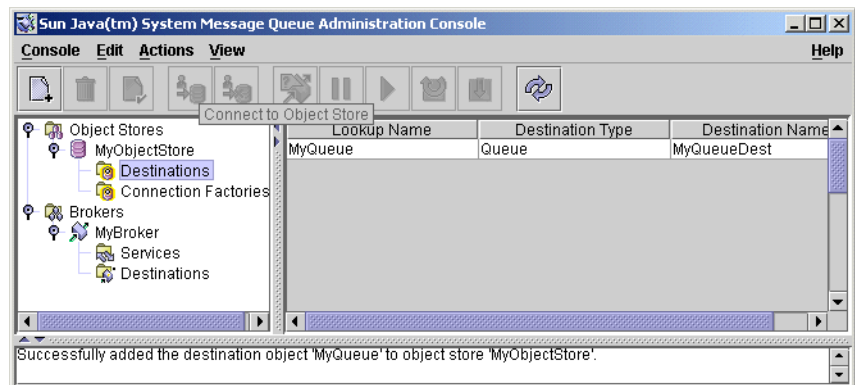


The dialog box titled "Add Destination Object" contains the following fields and controls:

- Lookup Name:** A text input field.
- Destination Type:** Two radio buttons: ☒ Queue and ☐ Topic.
- Read-Only:** A checkbox, currently unchecked.
- Destination Name:** A text input field containing "Untitled\_Destination\_Object".
- Destination Description:** A text input field containing "A Description for the Destination Object".
- Buttons:** OK, Reset To Defaults, Cancel, and Help.

- 「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" );
```
- 「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。
- 「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 ([116 ページの「ブローカに物理的送信先を追加する」](#)を参照)。
- 「OK」をクリックします。
- 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

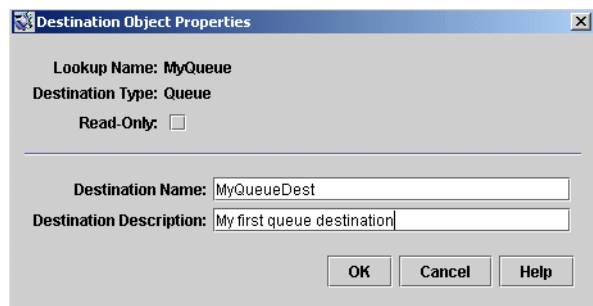
管理対象オブジェクトのプロパティを表示または更新するには、ナビゲーションペインにある「Destinations (送信先)」または「Connection Factories (コネクションファクトリ)」を選択し、結果ペインで特定のオブジェクトを選択してから、「Actions (アクション)」>「Properties (プロパティ)」の順に選択します。

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

ここで変更できる値は、送信先名と説明だけです。検索名を変更するには、オブジェクトを削除してから、新しいキュー管理対象オブジェクトを、希望する検索名で追加する必要があります。

4. 「Cancel (取消し)」をクリックしてダイアログを閉じます。





## コンソール情報を更新する

オブジェクトストアを操作している場合も、ブローカを操作している場合も、「View (表示)」>「Refresh (更新)」の順に選択すると、あらゆる要素または要素グループの視覚的表示を更新できます。

## サンプルアプリケーションを実行する

サンプルアプリケーションの `HelloWorldMessageJNDI` は、このチュートリアルで使用するために用意されています (場所については、以下のステップ 1 を参照)。このサンプルアプリケーションは、チュートリアルを進めていく中で作成した物理的な送信先と管理対象オブジェクトを使用します。つまり、`MyQueueDest` という名前のキューの物理的な送信先、`MyQueueConnectionFactory` および `MyQueue` という JNDI 検索名をそれぞれ持つ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キュー管理対象オブジェクトです。

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。

### ► HelloWorldMessageJNDI アプリケーションを実行するには

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

```
cd IMQ_HOME\demo\helloworld\helloworldmessagejndi (Windows の場合)
```

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

```
cd /opt/imq/demo/helloworld/helloworldmessagejndi (Linux の場合)
```

`HelloWorldMessageJNDI.class` ファイルが存在していることを確認します。アプリケーションに変更を加える場合は、『*Message Queue C Client Developer's Guide*』のクイックスタートチュートリアルに記載されたクライアントアプリケーションのコンパイルに関する指示に従って、アプリケーションを再コンパイルする必要があります。

2. `CLASSPATH` 変数に `HelloWorldMessageJNDI.class` ファイルと `Message Queue` 製品に組み込まれた `jms.jar`、`imq.jar`、および `fscontext.jar` といった `jar` ファイルを含む現在のディレクトリを追加します。`CLASSPATH` の設定方法については、『*Message Queue Java Client Developer's Guide*』を参照してください。

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

3. アプリケーションを実行する前に、ソースファイルの HelloWorldMessageJNDI.java を開き、ソース全体に目を通してください。このファイルには短いながら詳細な説明が用意されており、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。
4. 次のコマンドのどちらかを実行して、HelloWorldMessageJNDI アプリケーションを稼働します。

java HelloWorldMessageJNDI (Windows の場合)

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

アプリケーションが問題なく実行されると、次の出力があります。

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

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

Creating connection to broker.
Connection to broker created.

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

# ブローカの起動と設定

Sun Java™ System Message Queue 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。ブローカインスタンスの設定は、設定ファイルセットと、設定ファイル内の対応するプロパティをオーバーライドする `imqbrokerd` コマンドで渡されるオプションによって決まります。

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インスタンス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、使用方法については、[347 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されるインストール済みのブローカ設定ファイルテンプレートは、[付録 A 「Message Queue データの場所」](#)に示すとおり、オペレーティングシステムごとに異なるディレクトリに格納されています。

このディレクトリには、次のファイルが格納されています。

- **起動時に読み込まれるデフォルトの設定ファイル。**このファイルは、`default.properties` と呼ばれ、編集はできない。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
- **Message Queue のインストール時に指定されたプロパティを格納するインストール設定ファイル。**このファイルは、`install.properties` と呼ばれ、インストール後は編集できない

## インスタンス設定ファイル

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。インスタンス設定ファイルは、その設定ファイルが関連付けられているブローカインスタンスの名前 (*instanceName*) によって識別されたディレクトリに書き込まれます ( [付録 A「Message Queue データの場所」](#) を参照 )。

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

---

**注**

.../instances/*instanceName* ディレクトリとインスタンス設定ファイルは、対応するブローカインスタンスを作成したユーザーが所有します。それ以降、作成したブローカインスタンスの起動は、同じユーザーが行う必要があります。

---

インスタンス設定ファイルは、ブローカインスタンスによって管理されます。このファイルは、管理ツールを使用して設定に変更が加えられた場合に変更されます。インスタンス設定ファイルを手作業で編集して、設定を変更できます ( [136 ページの「インスタンス設定ファイルの編集」](#) を参照 )。手作業で変更するには、

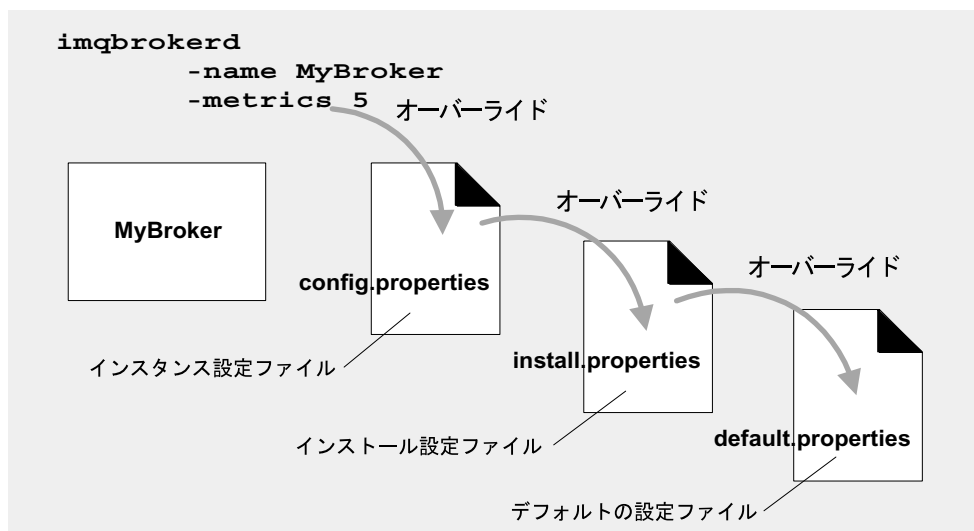
.../instances/*instanceName* ディレクトリの所有権が必要です。所有権がなければ、**root** としてログインしてディレクトリの権限を変更する必要があります。

クラスタでブローカインスタンスを接続する場合 ( [86 ページの「マルチブローカクラスタ \(Enterprise Edition\)」](#) 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、 [147 ページの「クラスタ設定プロパティ」](#) を参照

## 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 [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 Message Queue プロパティの定義では、次の命名構文を使用します。

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

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

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

次のエントリは、毎日、つまり 86400 秒ごとに新しいログファイルを作成するように指定します。

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

136 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。このインスタンス設定ファイルを編集して、対応するブローカインスタンスの動作やリソースの使用をカスタマイズできます。

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

- 管理ツールを使用する。`imqcmd` を使用して設定できるプロパティについての詳細は、[168 ページの表 6-4](#) を参照
- ブローカインスタンスをシャットダウンしてから `config.properties` ファイルを編集し、その後インスタンスを再起動する。**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ーだけが `config.properties` ファイルを編集するためのアクセス権を持つ

[表 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
<code>imq.accesscontrol.enabled</code>	ブール	<code>true</code>	<a href="#">72 ページの表 2-6</a>
<code>imq.accesscontrol.file.filename</code>	文字列	<code>accesscontrol.properties</code>	<a href="#">72 ページの表 2-6</a>
<code>imq.authentication.basic.user_repository</code>	文字列	<code>file</code>	<a href="#">72 ページの表 2-6</a>
<code>imq.authentication.client.response.timeout</code>	整数 (秒)	<code>180</code>	<a href="#">72 ページの表 2-6</a>
<code>imq.authentication.type</code>	文字列	<code>digest</code>	<a href="#">72 ページの表 2-6</a>
<code>imq.autocreate.destination.isLocalOnly</code>	ブール	<code>false</code>	<a href="#">83 ページの表 2-10</a>
<code>imq.autocreate.destination.limitBehavior</code>	文字列	<code>REJECT_NEWEST</code>	<a href="#">83 ページの表 2-10</a>
<code>imq.autocreate.destination.maxBytesPerMsg</code>	バイト文字列 <sup>1</sup>	<code>10k</code>	<a href="#">83 ページの表 2-10</a>

<sup>1</sup> 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。たとえば、次のように指定します。  
1000 は 1000 バイト、7500b は 7500 バイト、77k は 77K バイト (77 x 1024 = 78848 バイト)、17m は 17M バイト (17 x 1024 x 1024 = 17825792 バイト) をそれぞれ表します。

表 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.autocreate.destination. maxNumMsgs	整数	100,000	<a href="#">83 ページの表 2-10</a>
imq.autocreate.destination. maxNumProducers	整数	100	<a href="#">83 ページの表 2-10</a>
imq.autocreate.destination. maxTotalMsgBytes	バイト文字列 <sup>1</sup>	10m	<a href="#">83 ページの表 2-10</a>
imq.autocreate.queue	ブール	true	<a href="#">83 ページの表 2-10</a>
imq.autocreate.queue. consumerFlowLimit	整数	1000	<a href="#">83 ページの表 2-10</a>
imq.autocreate.queue. localDeliveryPreferred	ブール	false	<a href="#">83 ページの表 2-10</a>
imq.autocreate.queue. maxNumActiveConsumers	整数	1	<a href="#">83 ページの表 2-10</a>
imq.autocreate.queue. maxNumBackupConsumers	整数	0	<a href="#">83 ページの表 2-10</a>
imq.autocreate.topic	ブール	true	<a href="#">83 ページの表 2-10</a>
imq.autocreate.topic. consumerFlowLimit	整数	1,000	<a href="#">83 ページの表 2-10</a>
imq.cluster. <i>property_name</i>			<a href="#">147 ページの表 5-3</a>
imq.hostname	文字列	すべての使用可能な IP アドレス	<a href="#">59 ページの表 2-3</a>
imq.httpjms.http. <i>property_name</i>			<a href="#">323 ページの表 C-1</a>
imq.httpsjms.https. <i>property_name</i>			<a href="#">335 ページの表 C-3</a>
imq.keystore. <i>property_name</i>			<a href="#">232 ページの表 8-8</a>
imq.log.console.output	文字列	ERROR WARNING	<a href="#">77 ページの表 2-9</a>
imq.log.console.stream	文字列	ERR	<a href="#">77 ページの表 2-9</a>
imq.log.file.dirpath	文字列	<a href="#">付録 A 「Message Queue データの場所」 を参照</a>	<a href="#">77 ページの表 2-9</a>
imq.log.file.filename	文字列	log.txt	<a href="#">77 ページの表 2-9</a>

<sup>1</sup> 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。たとえば、次のように指定します。  
 1000 は 1000 バイト、7500b は 7500 バイト、77k は 77K バイト (77 x 1024 = 78848 バイト)、17m は 17M バイト (17 x 1024 x 1024 = 17825792 バイト) をそれぞれ表します。

表 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.output	文字列	ALL	<a href="#">77 ページの表 2-9</a>
imq.log.file.rolloverbytes	整数 ( バイト )	-1 ( ロールオーバーなし )	<a href="#">77 ページの表 2-9</a>
imq.log.file.rolloversecs	整数 ( 秒 )	604800	<a href="#">77 ページの表 2-9</a>
imq.log.level	文字列	INFO	<a href="#">77 ページの表 2-9</a>
imq.log.syslog.facility	文字列	LOG_DAEMON	<a href="#">77 ページの表 2-9</a>
imq.log.syslog.identity	文字列	imqbrokerd_\${imq. instanceName}	<a href="#">77 ページの表 2-9</a>
imq.log.syslog.logconsole	ブール	false	<a href="#">77 ページの表 2-9</a>
imq.log.syslog.logpid	ブール	true	<a href="#">77 ページの表 2-9</a>
imq.log.syslog.output	文字列	ERROR	<a href="#">77 ページの表 2-9</a>
imq.log.timezone	文字列	該当地域のタイム ゾーン	<a href="#">77 ページの表 2-9</a>
imq.message.expiration. interval	整数 ( 秒 )	60	<a href="#">65 ページの表 2-4</a>
imq.message.max_size	バイト文字列 <sup>1</sup>	70m	<a href="#">65 ページの表 2-4</a>
imq.metrics.enabled	ブール	true	<a href="#">77 ページの表 2-9</a>
imq.metrics.interval	整数 ( 秒 )	-1 ( 報告なし )	<a href="#">77 ページの表 2-9</a>
imq.metrics.topic.enabled	ブール	true	<a href="#">77 ページの表 2-9</a>
imq.metrics.topic.interval	整数 ( 秒 )	60	<a href="#">77 ページの表 2-9</a>
imq.metrics.topic.persist	ブール	false	<a href="#">77 ページの表 2-9</a>
imq.metrics.topic.timetolive	整数 ( 秒 )	300	<a href="#">77 ページの表 2-9</a>
imq.passfile.dirpath	文字列	付録 A 「Message Queue データの場所」 を参照	<a href="#">72 ページの表 2-6</a>
imq.passfile.enabled	ブール	false	<a href="#">72 ページの表 2-6</a>
imq.passfile.name	文字列	passfile	<a href="#">72 ページの表 2-6</a>

<sup>1</sup> 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。たとえば、次のように指定します。  
 1000 は 1000 バイト、7500b は 7500 バイト、77k は 77K バイト (77 x 1024 = 78848 バイト)、17m は 17M バイト (17 x 1024 x 1024 = 17825792 バイト) をそれぞれ表します。



表 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. destination.message. filepool.limit	整数	100	68 ページの表 2-5
imq.persist.file.message. cleanup	ブール	false	68 ページの表 2-5
imq.persist.file.message. filepool.cleanratio	整数	0	68 ページの表 2-5
imq.persist.file.message. max_record_size	バイト文字列 <sup>1</sup>	1m	68 ページの表 2-5
imq.persist.file.sync. enabled	ブール	false	68 ページの表 2-5
imq.persist.jdbc. <i>property_name</i>			312 ページの表 B-1
imq.persist.store	文字列	file	68 ページの表 2-5
imq.ping.interval	整数	120	59 ページの表 2-3
imq.portmapper.backlog	整数	50	59 ページの表 2-3
imq.portmapper.hostname	文字列	imq.hostname から継承	59 ページの表 2-3
imq.portmapper.port	整数	7676	59 ページの表 2-3
imq.resource_state.count	整数 ( パーセント )	5000 (green) 500 (yellow) 50 (orange) 0 (red)	65 ページの表 2-4
imq.resource_state. threshold	整数 ( パーセント )	0 (green) 80 (yellow) 90 (orange) 98 (red)	65 ページの表 2-4
imq.service.activelist	list	jms、admin	59 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	72 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	72 ページの表 2-6

<sup>1</sup> 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。たとえば、次のように指定します。  
1000 は 1000 バイト、7500b は 7500 バイト、77k は 77K バイト (77 x 1024 = 78848 バイト)、17m は 17M バイト (17 x 1024 x 1024 = 17825792 バイト) をそれぞれ表します。

表 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
<code>imq.service_name.authentication.type</code>	文字列	システム全体のプロパティから値を継承	<a href="#">72 ページの表 2-6</a>
<code>imq.service_name.max_threads</code>	整数	1000 (jms) 500 (ssljms) 500 (httpjms) 500 (httpsjms) 10 (admin) 10 (ssladmin)	<a href="#">59 ページの表 2-3</a>
<code>imq.service_name.min_threads</code>	整数	10 (jms) 10 (ssljms) 10 (httpjms) 10 (httpsjms) 4 (admin) 4 (ssladmin)	<a href="#">59 ページの表 2-3</a>
<code>imq.service_name.protocol_type.hostname</code>	文字列	<code>imq.hostname</code> から継承	<a href="#">59 ページの表 2-3</a>
<code>imq.service_name.protocol_type.port</code>	整数	0 ( 動的に割り当てられる )	<a href="#">59 ページの表 2-3</a>
<code>imq.service_name.threadpool_model</code>	文字列	dedicated (jms) dedicated (ssljms) dedicated (httpjms) dedicated (httpsjms) dedicated (admin) dedicated (httpsjms)	<a href="#">59 ページの表 2-3</a>
<code>imq.shared.connectionMonitor_limit</code>	整数	512 (Solaris および Linux の場合 ) 64 (Windows の場合 )	<a href="#">59 ページの表 2-3</a>
<code>imq.system.max_count</code>	整数、 0 ( 制限なし )	-1	<a href="#">65 ページの表 2-4</a>
<code>imq.system.max_size</code>	バイト文字列 <sup>1</sup> 0 ( 制限なし )	-1	<a href="#">65 ページの表 2-4</a>
<code>imq.transaction.autorollback</code>	ブール	false	<a href="#">65 ページの表 2-4</a>
<code>imq.user_repository.ldap.property_name</code>			<a href="#">222 ページの表 8-5</a>

<sup>1</sup> 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。たとえば、次のように指定します。  
 1000 は 1000 バイト、7500b は 7500 バイト、77k は 77K バイト (77 x 1024 = 78848 バイト)、17m は 17M バイト (17 x 1024 x 1024 = 17825792 バイト) をそれぞれ表します。

# ブローカの起動

ブローカインスタンスを起動するには、`imqbrokerd` コマンドを使用します。

---

**注**            管理コンソール (`imqadmin`) またはコマンドユーティリティ (`imqcmd`) を使用してブローカインスタンスを起動することはできません。これらの Message Queue 管理ツールを使用するには、事前にブローカインスタンスを実行する必要があります。

---

1 つ以上のプロパティ値をオーバーライドするには、有効な `imqbrokerd` コマンド行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、インスタンス設定ファイルに書き込まれません。

## imqbrokerd コマンドの構文

`imqbrokerd` コマンドの構文は、次のとおりです。オプションはスペースで区切ります。

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

---

**注** Solaris では、`/etc/imq/imqborkerd.conf` 設定ファイルの `RESTART` プロパティを `YES` に指定すると、異常終了後に、ブローカが自動的に再起動するように設定できます。

---

---

**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ーの `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ーだけが起動する必要があります。

---

## 起動の例

次の例は、`imqbrokerd` コマンドの使用方を示しています。`imqbrokerd` コマンド行オプションの詳細は、[143 ページの表 5-2](#) を参照してください。

- デフォルトのブローカ名と設定を使用するブローカインスタンス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 のポートマップ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。

- **Enterprise Edition** のトライアルライセンスでブローカインスタンスを起動するには Platform Edition ライセンスは持っているが、90 日間の Enterprise Edition 機能を試してみたい場合は、`-license` コマンド行オプションで試用ライセンスとして「`try`」を指定すると、Enterprise Edition のトライアルライセンスを有効にできます。

```
imqbrokerd -license try
```

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

- プラグイン持続で名前付きブローカインスタンスを起動するにはプラグインデータストア ([309 ページの付録 B 「プラグイン持続の設定」](#) を参照) を使用する、ユーザー名とパスワードが必要な `myBroker` という名前のブローカを起動するには、次のコマンドを使用します。

```
imqbrokerd -name myBroker -dbuser myName -dbpassword myPassword
```

## imqbrokerd オプションの概要

表 5-2 に、imqbrokerd コマンドのオプションと、各オプションから影響を受ける設定プロパティがあれば、そのプロパティを示します。

表 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 <i>fileName</i>	影響なし	ブローカクラスタに対してのみ適用。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。 <a href="#">153 ページの「設定変更レコードのバックアップ」</a> を参照
-cluster" [ <i>broker1</i> ] [ [ <i>broker2</i> ] ... ]"	imq.cluster.brokerlist に 接続先となるブローカのリス トを設定する	ブローカクラスタに対してのみ適用。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。 <i>host</i> に値を指定しない場合、localhost が使用さ れる。 <i>port</i> 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、 <a href="#">147 ページの「クラスタの操作 (Enterprise Edition)」</a> を参照
<i>broker</i> は次のどちらかに なる		
• <i>host[:port]</i>		
• [ <i>host</i> ]: <i>port</i>		
-dbpassword <i>password</i>	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 準拠のデータストアに対 するパスワードを指定する。 <a href="#">付録 B 「プラグ イン持続の設定」</a> を参照
-dbuser <i>userName</i>	imq.persist.jdbc.user を指定されたユーザー名に設 定する	プラグインの JDBC 準拠のデータベースに対 するユーザー名を指定する。 <a href="#">付録 B 「プラグ イン持続の設定」</a> を参照
-D <i>property=value</i>	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、 <a href="#">136 ページの表 5-1</a> を参照
		<b>注意</b> : 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 Message Queue でプロパティを設 定できなくなる
-force	影響なし	ユーザーの確認なしで、アクションを実行す る。このオプションは、通常は確認が必要な -remove instance オプションと -upgrade-store-nobackup オプションにだ け適用される

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
-h -help	影響なし	ヘルプを表示する。コマンド行ではそれ以外のことは実行されない
-javahome <i>path</i>	影響なし	代替の Java 2 準拠の 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
-ldappassword <i>password</i>	imq.user_repository. ldap.password を指定された パスワードに設定する	LDAP ユーザーリポジトリにアクセスするためのパスワードを指定する。 <a href="#">221 ページの「ユーザーリポジトリに LDAP サーバを使用する」</a> を参照
-license [ <i>licenseName</i> ]	影響なし	使用している Message Queue 製品エディションのデフォルトと異なる場合、読み込みのためのライセンスを指定する。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 Message Queue エディションによって、 <i>licenseName</i> 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。 <a href="#">35 ページの「製品エディション」</a> を参照
-loglevel <i>level</i>	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、 <a href="#">75 ページの「ロガー」</a> を参照
-metrics <i>interval</i>	imq.metrics. interval を特定の秒数に設 定する	ブローカメトリックスが一定間隔 ( 秒単位 ) でロガーに書き込まれるように指定する
-name <i>instanceName</i>	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、インスタンス名は imqbroker に設定される <b>注:</b> 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 <i>fileName</i>	imq.passfile. enabled を true に設定する。 jmq.passfile.dirpath にファ イルを含むパスを指定する imq.passfile.name にファ イル名を設定する	SSL キーストア、LDAP ユーザーリポジトリ、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、 <a href="#">236 ページの「passfile の使用」</a> 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
<code>-password keypassword</code>	<code>imq.keystore.password</code> 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、 <a href="#">69 ページの「セキュリティマネージャ」</a> を参照
<code>-port number</code>	<code>imq.portmapper.port</code> に特定の数を設定する	ブローカのポートマップのポート番号を指定する。デフォルトでは <b>7676</b> 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プが異なるポート番号となる必要がある。 <b>Message Queue</b> クライアントはこのポート番号を使用してブローカインスタンスに接続する
<code>-remove instance</code>	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。 <code>-force</code> オプションと一緒に指定しないかぎり、ユーザーの確認が求められる
<code>-reset store  messages  durables  props</code>	影響なし	指定された引数に応じて、データストアまたはデータストアのサブセットをリセットするか、あるいはブローカインスタンスのプロパティをリセットする  データストアをリセットすると、持続メッセージ、永続サブスクリプション、トランザクションの情報など、すべての持続データが消去される。このため、データが消去された状態で、ブローカインスタンスを起動できる。また、すべての持続メッセージだけを消去したり、すべての永続サブスクリプションだけを消去したりすることもできる。その後の再起動時に持続ストアをリセットしない場合は、 <code>-reset</code> オプションを指定せずにブローカインスタンスを再起動する。詳細は、 <a href="#">66 ページの「持続マネージャ」</a> 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 ( <code>config.properties</code> ) が空のファイルに置き換えられる。プロパティはすべてデフォルト値となる

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
-restore <i>fileName</i>	影響なし	ブローカクラスタに対してのみ適用。マスターブローカの設定変更レコードを、指定したバックアップファイルに置き換える。このファイルは、-backup オプションを使用して事前に作成しておく必要がある。 <a href="#">153 ページ</a> の「 <a href="#">設定変更レコードの復元</a> 」を参照
-shared	imq.jms. threadpool_model を shared に設定する	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、ブローカインスタンスにサポートされるコネクション数が増加する。詳細は、 <a href="#">56 ページ</a> の「 <a href="#">コネクションサービス</a> 」を参照
-silent -s	imq.log.console. output を NONE に設定する	コンソールへのログ作成をオフにする
-tty	imq.log.console. output を ALL に設定する	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
-upgrade-store-nobackup	影響なし	非互換のバージョンから Message Queue 3.5 または Message Queue 3.5 SPx へアップグレードすると、自動的に古いデータストアが削除されるように指定する。詳細は、『 <a href="#">Message Queue インストールガイド</a> 』を参照
-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 <i>arg1</i> [ <i>arg2</i> ] ... ]	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。たとえば、次のように指定する imqbrokerd -tty -vmargs "-Xmx128m -Xincgc"



# クラスタの操作 (Enterprise Edition)

この節では、マルチ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。クラスタの概要については、[86 ページの「マルチブローカクラスタ \(Enterprise Edition\)」](#)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時計の同期が取れていることを確認してください ([353 ページの「システムの時計の設定」](#)を参照)。

## 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に、クラスタ設定プロパティを一括して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。アスタリスク (\*) が付いたプロパティは、クラスタ内のすべてのブローカで同じ値にする必要があります。

表 5-3 クラスタ設定プロパティ

プロパティ名	説明
<code>imq.cluster.brokerlist*</code>	クラスタ内のすべてのブローカを指定する。 <i>host:port</i> エントリのコンマで区切られたリストで構成される。 <i>host</i> は、各ブローカのホスト名で、 <i>port</i> はそのポートマップのポート番号。たとえば、次のように指定する <code>host1:3000, host2:8000, ctrhost</code>
<code>imq.cluster.masterbroker*</code>	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、 <i>host:port</i> で構成される。 <i>host</i> は、マスターブローカのホスト名で、 <i>port</i> はそのポートマップのポート番号。このプロパティは運用環境に合わせて設定する。たとえば、 <code>ctrhost:7676</code>

表 5-3 クラスタ設定プロパティ ( 続き )

プロパティ名	説明
imq.cluster.url*	<p>クラスタ設定ファイルの場所を指定する。個別に設定するのではなく、ブローカが中央の 1 つのクラスタ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</p> <p>たとえば、次のように指定する http://webserver/imq/cluster.properties file:/net/mfsserver/imq/cluster.properties</p>
imq.cluster.port	<p>クラスタ内にある各ブローカに対して、cluster コネクションサービスのポート番号を指定する。cluster 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。</p> <p>デフォルト値: 0 ( ポートは動的に割り当てられる )</p>
imq.cluster.hostname	<p>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、cluster 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。cluster 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。</p> <p>デフォルト値: imq.hostname の値を継承する (59 ページの表 2-3 を参照)</p>
imq.cluster.transport*	<p>cluster コネクションサービスがクラスタ内のブローカ間の内部通信に使用するネットワークトランスポートを指定する。ブローカ間で安全な暗号化されたメッセージ配信を行うには、クラスタ内のすべてのブローカに対して、このプロパティを ssl に設定する。デフォルト値: tcp</p>

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、接続先のブローカのリスト (`imq.cluster.brokerlist`)、クラスタコネクションサービスに使用するネットワークトランスポート (`imq.cluster.transport`)、およびオプションでマスターブローカのアドレス (`imq.cluster.masterbroker`) が含まれる可能性があります。

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、host2、および 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には、クラスタ設定ファイルの URL を含む必要があります。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方法、およびクラスタ内のブローカ間で安全な暗号化されたメッセージ配信を実行できるようにクラスタを設定する方法を説明します。

### 接続方法

ブローカをクラスタに接続する方法は、クラスタ設定ファイルを使用する場合と、クラスタ設定ファイルを使用しない場合の 2 とおりに大別されます。

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、マスターブローカが起動されるまでは、接続できません。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せずに接続する

##### ➤ 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して接続する

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。この方法を使用する場合は、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。

### ブローカ間の安全なコネクション

クラスタ内のブローカ間で安全な暗号化されたメッセージの配信を実行したい場合は次のようにして、`cluster` コネクションサービスが SSL ベースのトランスポートプロトコルを使用するように設定する必要があります。

## ▶ クラスタで安全なコネクションを設定するには

1. クラスタ内のブローカごとに、SSL ベースのコネクションサービスを設定します。  
設定方法は、[230 ページ](#)の「TCP/IP を介した SSL ベースのサービスの設定」を参照してください。
2. `imq.cluster.transport` クラスタ設定プロパティを `ssl` に設定します。  
クラスタ設定ファイルを使用しない場合は、クラスタ内のブローカごとにこのプロパティを設定する必要があります。

## クラスタ内のブローカの管理

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、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か、あるいはクラスタからブローカを削除する必要があります。

### クラスタへのブローカの追加

## ▶ 新しいブローカを既存のクラスタに追加するには

- クラスタ設定ファイルを使用する場合
  - a. 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。
  - b. クラスタ内の各ブローカに次のコマンドを実行します。  

```
imqcmd reload cls
```

  
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。
  - c. `imq.cluster.url` 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、新しいブローカを起動します。  
  
これにより、ブローカがクラスタ設定ファイルを指すようになります。
- 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`、`imq.cluster.transport` (安全なクラスタコネクションサービスを使用している場合)、および必要場合は `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。

### クラスタ内のブローカの再起動

クラスタ内のブローカが何らかの理由でクラッシュしたかシャットダウンされた場合は、そのブローカをクラスタのメンバーとして再起動する必要があります。

► 既存のクラスタのメンバーであるブローカを再起動するには

- 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、ブローカを再起動するときに、`-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、ブローカの再起動時に、`-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。

## クラスタからのブローカの削除

► ブローカを既存のクラスタから削除するには

- 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。
  - a. 設定ファイルからブローカの詳細を削除します。
  - b. 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。
  - c. `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管理

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。この状態では、永続サブスクリプションと管理者が作成した物理的な送信先に関する情報が含まれます。すべてのブローカは、これらの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、次に、設定変更レコードを確認します。したがって、マスターブローカに障害があると、このような同期が不可能となります。その結果、マスターブローカに障害が生じると、物理的な送信先や永続サブスクリプションを作成したり削除したりできません。

マスターブローカには重要な情報が含まれているため、マスターブローカの設定変更レコードを定期的にバックアップし、障害発生時にバックアップから復元できるようにする必要があります。

次の節では、設定変更レコードのバックアップと復元の方法を説明します。

### 設定変更レコードのバックアップ

#### ► 設定変更レコードをバックアップするには

`imqbrokerd` コマンドに `-backup` オプションを使用します。たとえば、次のように指定します。

```
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。つまり、バックアップが最後に実行されたとき以降に行われた、物理的な送信先または永続サブスクリプションの変更は失われます。

### 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、古いデータの再読み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たりという効果が得られます。

## ログ作成

この節では、デフォルトでのブローカのログ作成の設定と、代替の出力チャネルへログ情報をリダイレクトするように設定を変更したり、ログファイルのロールオーバー条件を変更したりする方法について説明します。ログ作成の説明については、[75 ページの「ロガー」](#)を参照してください。ログ作成を使用してブローカメトリックスを報告する方法については、[256 ページの「監視ツール」](#)を参照してください。

### デフォルトのログ作成の設定

ブローカを起動すると、ログファイルが関連付けられたブローカインスタンスの名前 (*instanceName*) で識別されるディレクトリにあるローリングログファイルにログ出力を保存するように自動的に設定されます ([付録 A 「Message Queue データの場所」](#)を参照)。

```
.../instances/instanceName/log/
```

ログファイルは、簡単なテキストファイルです。ログファイルは、次のように順番に名前が付けられています。

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

デフォルトでは、ログファイルは週に 1 回ロールオーバーされ、システムは 9 つのバックアップファイルを保持します。

- ログファイルが保管されるディレクトリを変更するには、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、3 つのログカテゴリをサポートしています。それらは、ERROR、WARNING、INFO です ([75 ページの表 2-7](#)を参照)。ログ作成レベルを設定すると、そのレベル以上のメッセージが収集されます。デフォルトのログレベルは INFO です。つまり、デフォルトでは、ERROR、WARNING、および INFO レベルのメッセージが記録されます。



## ログメッセージの書式設定

記録するメッセージは、タイムスタンプ (タイムスタンプのタイムゾーンを変更する方法は、[77 ページの表 2-9](#) を参照)、メッセージコード、およびメッセージ自体から構成されます。情報量は、設定したログレベルにより異なります。INFO メッセージの例を次に示します。

```
[13/Sep/2000:16:13:36 PDT] B1004 Starting the broker service
using tcp [ 25374,100] with min threads 50 and max threads of 500
```

## ローガー設定の変更

[77 ページの表 2-9](#) にすべてのローガープロパティを示します。

### ➤ ブローカのロー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。

ローガープロパティを設定すると、手順は完了します。ロー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` ファイルにロー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` コマンドでロー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、ローガープロパティ、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

<code>imqbrokerd</code> オプション	説明
<code>-metrics interval</code>	メトリックス情報がローガーに書き込まれる間隔を、秒単位で指定する
<code>-loglevel level</code>	ログレベルを、ERROR、WARNING、INFO のいずれかに設定する

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
-silent	コンソールへのログ作成をオフにする
-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、出力チャンネルの変更
- ロールオーバー条件の変更

出力チャンネルの変更

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ンネルを変更できます。

- 指定したレベルのすべてのログカテゴリを画面に表示するには、imqbrokerd コマンドの -tty オプションを使用する
- 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	ロガー出力チャンネルを変更する前に、出力チャンネルにマッピングされた情報をサポートするレベルにログ作成が設定されていることを確認する必要があります。たとえば、ログレベルを ERROR に設定し、imq.log.console.output プロパティを WARNING に設定すると、WARNING メッセージのロ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。
---	---

## ログファイルの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ります。たとえば、次のようにプロパティを定義すると、間隔が10日に変更される

```
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ります。たとえば、次のように定義すると、500,000バイトの制限に達したあと、ブローカが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。

ログ作成

# 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。次の項目が含まれます。

- 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ーおよび更新
- 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。

- **Message Queue** 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、アクティブでない場合でも、**Message Queue** 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。**Message Queue** のリソースを節約するために、**Message Queue** 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
- **Message Queue** 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。**Message Queue** 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。

## imqcmd 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]
imqcmd -h|H
imqcmd -v
```

`-v`、`-h`、および `-H` オプションを指定した場合、そのコマンド行で指定するサブコマンドは実行されません。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。サブコマンドについては、この章のタスクに関する節で詳しく説明します。

表 6-1 `imqcmd` のサブコマンド

サブコマンドと引数	説明
<code>commit txn</code>	トランザクションをコミットする
<code>compact dst</code>	1 つ以上の送信先に対応する組み込みのファイルベースのデータストアを圧縮する
<code>create dst</code>	送信先を作成する
<code>destroy dst</code>	送信先を破棄する
<code>destroy dur</code>	永続サブスクリプションを破棄する
<code>list cxn</code>	ブローカのコネクションを一覧表示する
<code>list dst</code>	ブローカの送信先を一覧表示する
<code>list dur</code>	トピックの永続サブスクリプションを一覧表示する

表 6-1 imqcmd のサブコマンド ( 続き )

サブコマンドと引数	説明
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 dst	送信先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 dst	ブローカの 1 つ以上の送信先を停止する
pause svc	ブローカのシングル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ページする
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ページする
query bkr	ブローカの情報をクエリーおよび表示する
query cxn	コネクションの情報をクエリーおよび表示する
query dst	送信先の情報をクエリーおよび表示する
query svc	サービスの情報をクエリーおよび表示する
query txn	トランザクションの情報をクエリーおよび表示する
reload cls	ブローカクラスタ設定の再読み込みを行う
restart bkr	現在実行中のブローカ インスタンスを再起動する。新たにブローカインスタンスを起動するためには使用できない
resume bkr	ブローカのすべてのサービスを再開する
resume dst	ブローカの 1 つ以上の停止された送信先を再開する
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	ブローカインスタンスのシャットダウン。その後、imqbrokerd コマンドで再起動できるが、imqcmd コマンドの restart bkr サブコマンドでは再起動できない
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 <i>hostName:port</i>	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：-b :7878 名前だけを指定する場合：-b somehost
-c <i>clientID</i>	トピックの永続サブスクリイバの ID を指定する。 <a href="#">188 ページ</a> の「永続サブスクリプションの管理」を参照
-d <i>destinationName</i>	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。 <a href="#">188 ページ</a> の「永続サブスクリプションの管理」を参照
-f	ユーザーの確認なしで、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。コマンド行ではそれ以外のことは実行されない
-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。コマンド行ではそれ以外のことは実行されない
-int <i>interval</i>	metrics bkr、metrics dst、および metrics svc サブコマンドがメトリックス出力を表示する間隔を秒単位で指定する
-javahome <i>path</i>	使用する代替の Java 2 互換のランタイムを指定する。デフォルトではシステム上のランタイムまたは Message Queue にバンドルされたランタイムを使用する
-m <i>metricType</i>	表示するメトリックス情報のタイプを指定する。このオプションは、metrics dst、metrics svc、または metrics bkr サブコマンドと同時に使用する。metricType の値は、メトリックスが送信先、サービス、ブローカのどれに對して生成されたかによって異なる
-msp <i>numSamples</i>	metrics bkr、metrics dst、および metrics svc サブコマンドがメトリックス出力で表示するメトリックスのサンプル数を指定する
-n <i>argumentName</i>	サブコマンドの引数の名前を指定する。これはサブコマンドに応じて、物理的な送信先、永続サブスクリプション、コネクション ID、またはトランザクション ID の名前になる



表 6-2 imqcmd のオプション ( 続き )

オプション	説明
-o <i>attribute=value</i>	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 (165 ページの「ブローカの管理」を参照)、サービス (171 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 (177 ページの「送信先の管理」を参照) になる
-p <i>password</i>	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
-pst <i>pauseType</i>	送信先を停止したときに、プロデューサ、コンシューマ、または両方を停止させるどうかを指定する。177 ページの「送信先の管理」を参照
-rtm <i>timeout</i>	imqcmd のサブコマンドの初期 ( 再試行 ) タイムアウト期間を秒単位で指定する。タイムアウトとは、imqcmd のサブコマンドがブローカへの要求を作成した後、待機している時間の長さ。それ以降、サブコマンドが再試行されるたびに、タイムアウト値として初期タイムアウト値の倍数が使用される。デフォルト値: 10
-rtr <i>numRetries</i>	imqcmd のサブコマンドが最初にタイムアウトになった後の再試行回数を指定する。デフォルト値: 5
-s	サイレントモード。出力が表示されない
-secure	ssladmin コネクションサービス (234 ページの「手順 4: SSL ベースのクライアントを設定および実行する」を参照) を使用して、セキュリティ保護されたブローカへの管理コネクションを指定する
-svn <i>serviceName</i>	どのコネクションのサービスを一覧表示するかを指定する。176 ページの「コネクション情報の取得」を参照
-t <i>destType</i>	送信先のタイプを指定する。t ( トピック )、または q ( キュー ) のどちらかとなる。177 ページの「送信先の管理」を参照
-tmp	一時送信先を表示する 177 ページの表 6-9 を参照
-u <i>userName</i>	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。コマンド行ではそれ以外のことは実行されない

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)、ユーザー名 (-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。ユーザー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。

---

**注** -secure オプションを使用するには、[230 ページの「TCP/IP を介した SSL ベースのサービスの設定」](#)に記載されているとおり、ターゲットブローカインスタンスに ssladmin サービスを設定し有効にする必要があります。

---

## imqcmd コマンドの使用

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  
[141 ページの「ブローカの起動」](#)を参照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。
- ブローカをローカルホストのポート 7676 で実行していない場合、-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ーリポジトリに照らし合わされて認証される。詳細は、[214 ページの「ユーザーの認証」](#)を参照

Message Queue をインストールすると、デフォルトの単層ファイルのユーザーリポジトリがインストールされます。リポジトリは 2 つのエントリと一緒に出荷されます。1 つは管理ユーザー用、もう 1 つはゲストユーザー用です。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インスタンスに接続できます。たとえば、Message Queue をテストするだけなら、デフォルトのユーザー名とパスワード (admin/admin) を使用して、imqcmd ユーティリティを実行できます。

運用システムをセットアップする場合は、管理ユーザー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 ( [第 8 章「セキュリティの管理」](#) を参照 )。特に、Message Queue のユーザーリポジトリ内にエントリを作成する必要があります ( [214 ページの「単層型ファイルユーザーリポジトリを使用する」](#) を参照 )。また、ユーザーリポジトリについて、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 ( [221 ページの「ユーザーリポジトリに LDAP サーバを使用する」](#) を参照 )。

## imqcmd の使用例

次の例は、imqcmd コマンドの使用方法を示しています。

- localhost のポート 7676 で実行しているブローカのプロパティを一覧表示する場合  

```
imqcmd query bkr -u admin -p admin
```
- myserver のポート 1564 で、ユーザー名が alladin、ユーザーパスワードが abracadabra で実行しているブローカのプロパティを一覧表示する場合  

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

ユーザー名の 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。
- localhost のポート 7676 で実行し、コマンドの初期タイムアウトが 20 秒、再試行回数 (タイムアウト後) が 7 に設定されているブローカのプロパティを一覧表示する場合  

```
imqcmd query bkr -u admin -p admin -rtm 20 -rtr 7
```

## ブローカの管理

コマンドユーティリティには、次のブローカ管理タスクを実行するために使用できるサブコマンドが含まれています。

- [ブローカ情報の表示](#)
- [ブローカのプロパティの更新](#)
- [ブローカのメトリックスの表示](#)
- [ブローカの状態の制御](#)

ブローカの接続サービスを管理する方法については、[171 ページの「コネクションサービスの管理」](#)を参照してください。ブローカの送信先を管理する方法については、[177 ページの「送信先の管理」](#)を参照してください。

[表 6-3](#) は、ブローカを管理するのに使用する imqcmd のサブコマンドについてまとめたものです。ホスト名、またはポートを指定しない場合は、デフォルトの localhost:7676 が使用されます。

表 6-3           ブローカを管理する imqcmd のサブコマンド

サブコマンドの構文	説明
<code>metrics bkr [-b <i>hostName:port</i>]           [-m <i>metricType</i>]           [-int <i>interval</i>]           [-msp <i>numSamples</i>]</code>	<p>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。</p> <p>表示するメトリックスのタイプを指定するには、<code>-m</code> オプションを使用する</p> <p><b>ttl</b>   ブローカとの間のメッセージとパケットのフローに関するメトリックスを表示する (デフォルトのメトリックスタイプ)</p> <p><b>rts</b>   ブローカとの間のメッセージとパケットのフローレートに関するメトリックスを表示する (秒単位)</p> <p><b>cxn</b>   コネクション、仮想メモリーヒープ、およびスレッドを表示する</p> <p>メトリックスを表示する間隔を秒単位で指定するには、<code>-int</code> オプションを使用する。デフォルトは 5 秒</p> <p>出力で表示するサンプル数を指定するには、<code>-msp</code> オプションを使用する。デフォルトは無制限 (無限)</p>
<code>pause bkr [-b <i>hostName:port</i>]</code>	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。 <a href="#">169 ページの「ブローカの停止および再開」</a> を参照
<code>query bkr -b <i>hostName:port</i></code>	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
<code>reload cls</code>	ブローカクラスタに対してのみ適用。クラスタ内のすべてのブローカで、 <code>imq.cluster.brokerlist</code> 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、 <a href="#">151 ページの「クラスタへのブローカの追加」</a> を参照

表 6-3          ブローカを管理する imqcmd のサブコマンド ( 続き )

サブコマンドの構文	説明
restart bkr [-b <i>hostName:port</i> ]	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
resume bkr [-b <i>hostName:port</i> ]	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 <i>hostName:port</i> ]	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
update bkr [-b <i>hostName:port</i> ] -o <i>attribute=value</i> [-o <i>attribute=value1</i> ] ò	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

localhost のポート 7676 で実行するブローカを対象にしない場合、表 6-3 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。

## ブローカ情報の表示

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

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

このコマンドでは、次のような情報が出力されます。

Version	3.5 SP1
Instance Name	imqbroker
Primary Port	7676
Current Number of Messages in System	0
Current Total Message Bytes in System	0
Max Number of Messages in System (-1)	unlimited

Max Total Message Bytes in System (-1)	unlimited
Max Message Size	70m
Auto Create Queues	true
Auto Create Topics	true
Auto Created Queue Max Number of Active Consumers	1
Auto Created Queue Max Number of Backup Consumers	0
Cluster Broker List (active)	
Cluster Broker List (configured)	
Cluster Master Broker	
Cluster URL	
Log Level	INFO
Log Rollover Interval (seconds)	604800
Log Rollover Size (bytes)	unlimited
(-1)	

## ブローカのプロパティの更新

表 6-4 に示すブローカのプロパティを更新するには、`update bkr` 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 6-4 imqcmd によって更新されるブローカプロパティ

プロパティ	参照先
imq.autocreate.queue	<a href="#">83 ページの表 2-10</a>
imq.autocreate.topic	<a href="#">83 ページの表 2-10</a>
imq.autocreate.queue.maxNumActiveConsumers	<a href="#">83 ページの表 2-10</a>
imq.autocreate.queue.maxNumBackupConsumers	<a href="#">83 ページの表 2-10</a>
imq.cluster.url	<a href="#">147 ページの表 5-3</a>
imq.log.level	<a href="#">77 ページの表 2-9</a>
imq.log.file.rolloversecs	<a href="#">77 ページの表 2-9</a>
imq.log.file.rolloverbytes	<a href="#">77 ページの表 2-9</a>
imq.system.max_count	<a href="#">65 ページの表 2-4</a>
imq.system.max_size	<a href="#">65 ページの表 2-4</a>
imq.message.max_size	<a href="#">65 ページの表 2-4</a>
imq.portmapper.port	<a href="#">59 ページの表 2-3</a>

たとえば、次のコマンドはキュー送信先の自動作成を無効にします。

```
imgcmd update bkr -o "img.autocreate.queue=false"
-u admin -p admin
```

## ブローカの状態の制御

ブローカの起動後に、次の `imgcmd` 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。

### ブローカの停止および再開

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

ただし、ブローカを停止しても管理接続サービスは中断されないため、ブローカへのメッセージを制限するために必要な管理タスクは実行できます。たとえば、特定の送信先にメッセージが集中した場合には、ブローカを停止し、問題の修復に役立つ次のいずれかを実行できます。これらは、メッセージの送信元の追跡、送信先のサイズの制限、または送信先の破棄のいずれかです。

ブローカを停止しても、クラスタ接続サービスは継続されます。ただし、クラスタ内のメッセージ配信は、クラスタ内のブローカによって実行される配信機能によって異なります。

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

```
imgcmd pause bkr -b myhost:1588 -u admin -p admin
```

また、個々の接続サービスを停止することもできます。[175 ページの「接続サービスの停止および再開」](#)を参照してください。個々の送信先の停止については、[185 ページの「送信先の停止と再開」](#)を参照してください。

- **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、`localhost` のポート `7676` で実行していたブローカが再開されます。

```
imgcmd resume bkr -u admin -p admin
```

ブローカのシャットダウンと再起動

- 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、正常な終了です。ブローカは新しいコネクションやメッセージを受け入れるのをやめて、既存のメッセージの配信を完了し、ブローカプロセスを終了します。次のコマンドでは、ctrlsrv のポート 1572 で実行していたブローカがシャットダウンされます。

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

- 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、localhost のポート 7676 で実行していたブローカが再起動されます。

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

ブローカのメトリックスの表示

ブローカに関するメトリックス情報を表示するには、metrics bkr サブコマンドを使用します。たとえば、ブローカに入力するメッセージフローとブローカから出力されるメッセージのフローレートを 10 秒間隔で取得するには、次のコマンドを使用します。

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

このコマンドでは、次のような情報が出力されます。

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

imqcmd を使用してブローカのメトリックスをレポートする方法の詳細は、[256 ページ](#)の「監視ツール」を参照してください。



# コネクションサービスの管理

コマンドユーティリティには、次のコネクションサービス管理タスクを実行するために使用できるサブコマンドが含まれています。

- [コネクションサービスの一覧表示](#)
- [コネクションサービス情報の表示](#)
- [コネクションサービスのプロパティの更新](#)
- [コネクションサービスのメトリックスの表示](#)
- [コネクションサービスの停止および再開](#)

Message Queue のコネクションサービスの概要については、[56 ページ](#)の「コネクションサービス」を参照してください。

[表 6-5](#) は、コネクションサービスを管理するのに使用する `imqcmd` のサブコマンドについてまとめたものです。ホスト名、またはポートを指定しない場合は、デフォルトの `localhost:7676` が使用されます。

表 6-5      コネクションサービスを管理する <code>imqcmd</code> のサブコマンド	
サブコマンドの構文	説明
<code>list svc [-b <i>hostName:port</i>]</code>	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

表 6-5          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
<code>metrics svc -n serviceName</code> <code>[-b hostName:port]</code> <code>[-m metricType]</code> <code>[-int interval]</code> <code>[-msp numSamples]</code>	<p>デフォルトのブローカ、または指定したホストとポートのブローカで実行している特定のサービスのメトリックスを表示する</p> <p>表示するメトリックスのタイプを指定するには、<code>-m</code> オプションを使用する</p> <p><b>ttl</b>      指定したサービスを使ってブローカとの間で入出力されているメッセージとパケットのフローに関するメトリックスを表示する ( デフォルトのメトリックスタイプ )</p> <p><b>rts</b>      指定したサービスを使ってブローカとの間で入出力されているメッセージとパケットのフローレートに関するメトリックスを表示する ( 秒単位 )</p> <p><b>cxn</b>      コネクション、仮想メモリーヒープ、およびスレッドを表示する</p> <p>メトリックスを表示する間隔を秒単位で指定するには、<code>-int</code> オプションを使用する。デフォルトは 5 秒</p> <p>出力で表示するサンプル数を指定するには、<code>-msp</code> オプションを使用する。デフォルトは無制限 ( 無限 )</p>
<code>pause svc -n serviceName</code> <code>[-b hostName:port]</code>	<p>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。<code>admin</code> サービスは停止できない</p>
<code>query svc -n serviceName</code> <code>[-b hostName:port]</code>	<p>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</p>
<code>resume svc -n serviceName</code> <code>[-b hostName:port]</code>	<p>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</p>
<code>update svc -n serviceName</code> <code>[-b hostName:port]</code> <code>-o attribute=value</code> <code>[-o attribute=value1] ...</code>	<p>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、<a href="#">174 ページの表 6-7</a> を参照</p>

ブローカは、アプリケーションクライアントと管理クライアントの両方からの通信をサポートしています。**Message Queue** ブローカで現在使用できるコネクションサービスを、表 6-6 に示します。「サービス名」列の値は、**-n** オプションでサービス名を指定するのに使用する値になります。表が示すように、各サービスは使用するサービスタイプ (アプリケーションクライアントの場合は **NORMAL**、管理クライアントの場合は **ADMIN**) と基礎となるトランスポートレイヤーによって指定されます。

表 6-6           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	tcp
ssljms (Enterprise Edition)	NORMAL	tls (SSL ベースセキュリティ)
httpjms (Enterprise Edition)	NORMAL	http
httpsjms (Enterprise Edition)	NORMAL	https (SSL ベースセキュリティ)
admin	ADMIN	tcp
ssladmin (Enterprise Edition)	ADMIN	tls (SSL ベースセキュリティ)

## コネクションサービスの一覧表示

ブローカ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] -u admin -p admin
```

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

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

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

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

このコマンドでは、次のような情報が出力されます。

-----	-----	-----
Service Name	Port Number	Service State
-----	-----	-----

admin	41844 (dynamic)	RUNNING
httpjms	-	UNKNOWN
httpsjms	-	UNKNOWN
jms	41843 (dynamic)	RUNNING
ssladmin	dynamic	UNKNOWN
ssljms	dynamic	UNKNOWN

## コネクションサービス情報の表示

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

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

このコマンドでは、次のような情報が出力されます。

Service Name	jms
Service State	RUNNING
Port Number	60920 (dynamic)
Current Number of Allocated Threads	0
Current Number of Connections	0
Min Number of Threads	10
Max Number of Threads	1000

## コネクションサービスのプロパティの更新

表 6-7 に示す 1 つ以上のサービスのプロパティを変更するには、`update` サブコマンドを使用します。

表 6-7 imqcmd によって更新されるコネクションサービスプロパティ

プロパティ	説明
port	更新するサービスに割り当てられるポート ( <code>httpjms</code> または <code>httpsjms</code> には適用しない)。値 0 は、ポートマッパーによって動的に割り当てられるポートを示している
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"
```

## コネクションサービスのメトリックスの表示

シングルサービスに関するメトリックス情報を表示するには、metrics サブコマンドを使用します。たとえば、jms コネクションサービスによって処理されたメッセージとパケットの累計数を取得するには、次のコマンドを使用します。

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

このコマンドでは、次のような情報が出力されます。

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

imqcmd を使用してコネクションサービスのメトリックスをレポートする方法の詳細は、[256 ページの「監視ツール」](#)を参照してください。

## コネクションサービスの停止および再開

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 -u admin -p admin
```

サービスを停止すると、次のような結果になります。

- ブローカは、停止したサービスでの新たなクライアントコネクションの受け入れをやめる。Message Queue クライアントが新しいコネクションを開こうとすると、例外が発生する
- 停止したサービスの既存のコネクションはすべて維持されるが、ブローカはサービスが再開されるまでこれらのコネクションのすべてのメッセージ処理を中断する。たとえば、クライアントがメッセージを送信しようとしても、サービスが再開されるまでは、send() メソッドがそれを阻止する

- すでにブローカが受信済みのメッセージのメッセージ配信状態は維持される。たとえば、トランザクションは中断されず、サービスが再開された時点でメッセージ配信も再開される

サービスを再開するには、次のようなコマンドを使用します。

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

# コネクション情報の取得

コマンドユーティリティには、コネクションに関する情報を一覧表示し取得するために使用できるサブコマンドが含まれています。

表 6-8 に、コネクションに適用する `imqcmd` のサブコマンドを一覧表示します。ホスト名、またはポートを指定しない場合は、`localhost`、`7676` を使用するものと仮定します。

表 6-8          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
<code>list cxn [-svn serviceName] [-b hostName:port]</code>	デフォルトのブローカ、または指定したホストとポートのブローカで実行している指定したサービス名のコネクションをすべて一覧表示する。サービス名が指定しない場合は、すべてのコネクションが一覧表示される
<code>query cxn -n connectionID [-b hostName:port]</code>	デフォルトのブローカ、または指定したホストとポートのブローカで実行している指定したコネクションに関する情報を表示する

シングルコネクションサービスに関する情報のクエリーと表示を行うには、`query` サブコマンドを使用します。たとえば、次のように指定します。

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

このコマンドでは、次のような情報が出力されます。

Connection ID	421085509902214374
User	guest
Service	jms
Producers	0
Consumers	1
Host	111.22.333.444

Port	60953
Client ID	
Client Platform	

# 送信先の管理

Message Queue のメッセージはすべて、特定のブローカで作成されたキューとトピック送信先を経由して、コンシューマクライアントにルートされます。

コマンドユーティリティには、次の送信先管理タスクを実行するために使用できるサブコマンドが含まれています。

- [送信先の作成](#)
- [定義の一覧表示](#)
- [送信先情報の表示](#)
- [送信先の属性の更新](#)
- [送信先のメトリックスの表示](#)
- [送信先の停止と再開](#)
- [送信先のページ](#)
- [送信先の破棄](#)
- [送信先の圧縮](#)

送信先の概要については、[80 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**          送信先の管理に使用される `imqcmd` のサブコマンド

サブコマンドの構文	説明
<code>compact dst [-t <i>destType</i> -n <i>destName</i>]</code>	特定のタイプと名前の送信先に対応する組み込みのファイルベースのデータストアを圧縮する。送信先のタイプと名前が指定されていない場合は、すべての送信先が圧縮される。圧縮する前に送信先を停止する必要がある

表 6-9 送信先の管理に使用される imqcmd のサブコマンド ( 続き )

サブコマンドの構文	説明
<code>create dst -t destType</code> <code>-n destName</code> <code>[-o attribute=value]</code> <code>[-o attribute=value1]...</code>	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 " " および "\$" で開始できる。文字列「mq.」で開始することはできない
<code>destroy dst -t destType</code> <code>-n destName</code>	特定のタイプと名前の送信先を破棄する
<code>list dst [-t destType]</code> <code>[-tmp]</code>	指定したタイプの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 (85 ページの「一時送信先」を参照)  type 引数には次の 2 つの値がある  <code>destType = q</code> ( キュー ) <code>destType = t</code> ( トピック )  タイプが指定されない場合は、すべてのタイプの送信先すべてが一覧表示される
<code>metrics dst -t destType</code> <code>-n destName</code> <code>[-m metricType]</code> <code>[-int interval]</code> <code>[-msp numSamples]</code>	特定のタイプと名前の送信先に関するメトリックス情報を表示する  表示するメトリックスのタイプを指定するには、 <code>-m</code> オプションを使用する  <code>ttl</code> 送信先との間でやり取りされメモリーに常駐しているメッセージとパケットに関するメトリックスを表示する ( デフォルトのメトリックスタイプ )  <code>rts</code> 送信先との間のメッセージとパケットのフローレート ( 秒単位 ) に関するメトリックスと、その他のレート情報を表示する  <code>cn</code> コンシューマ関連のメトリックスを表示する  <code>disk</code> ディスク使用量のメトリックスを表示する  メトリックスを表示する間隔を秒単位で指定するには、 <code>-int</code> オプションを使用する。デフォルトは 5 秒  出力で表示するサンプル数を指定するには、 <code>-msp</code> オプションを使用する。デフォルトは無制限 ( 無限 )



表 6-9 送信先の管理に使用される imqcmd のサブコマンド ( 続き )

サブコマンドの構文	説明
<pre>pause dst [-t destType -n destName] [-pst pauseType]</pre>	特定のタイプと名前の送信先について、コンシューマへのメッセージ (-pst CONSUMERS)、プロデューサからのメッセージ (-pst PRODUCERS)、またはその両方 (-pst ALL) を停止する。送信先のタイプと名前が指定されていない場合は、すべての送信先が停止される。デフォルト値は ALL
<pre>purge dst -t destType -n destName</pre>	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
<pre>query dst -t destType -n destName</pre>	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
<pre>resume dst [-t destType -n destName]</pre>	特定のタイプと名前の停止された送信先についてメッセージの配信を再開する。送信先のタイプと名前が指定されていない場合は、すべての送信先が再開される
<pre>update dst -t destType -n destName -o attribute=value [-o attribute=value1] ...</pre>	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

送信先を作成するときには、次の情報を指定する必要があります。

- 送信先のタイプ: トピックまたはキュー
- 送信先名: 英数字 ( 空白文字は含まない ) だけを使用でき、英字、または文字「\_」と「\$」で開始できる。文字列「mq.」で開始することはできない
- 送信先の属性に対するデフォルト以外の値

送信先の属性の多くは、ブローカメモリーリソースとメッセージフローを管理するために使用されます。たとえば、送信先で許容されるプロデューサの最大数、または送信先で許容されるメッセージの最大数 ( または、サイズ ) を指定できます。これらの制限は、ブローカ設定プロパティを使用してブローカ全体に設定できる制限に似ています ( 63 ページの「メモリーリソースとメッセージフローの管理」を参照 )。また、これらの制限に達したときの、ブローカの応答方法も指定できます。

キュー送信先だけに適用される送信先属性もあります。これらの属性は、複数のコンシューマにロードバランスされたメッセージ配信で使用するアクティブコンシューマ数とバックアップコンシューマ数を指定するために使用されます ( 80 ページの「キューの送信先」を参照 )。

表 6-10 で、各タイプの送信先に適用される属性を説明します。送信先の作成または更新時に属性値を設定できます。自動作成される送信先の場合は、ブローカのインスタンス設定ファイルにデフォルトのプロパティ値を設定します (133 ページの「設定ファイル」を参照)。

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
キューとトピック	maxNumMsgs <sup>1</sup>	-1 (無制限)	送信先で許容されるコンシュームされないメッセージの最大数を指定する
キューとトピック	maxTotalMsgBytes <sup>1</sup>	-1 (無制限)	送信先でコンシュームされないメッセージ用として許容されるメモリの最大量をバイト単位で指定する
キューとトピック	limitBehavior	REJECT_OLDEST	<p>メモリ制限のしきい値に達したときのブローカの応答方法を指定する。値は、次のどれかになる</p> <p>FLOW_CONTROL - プロデューサの低速化</p> <p>REMOVE_OLDEST - もっとも古いメッセージの破棄</p> <p>REMOVE_LOW_PRIORITY - メッセージの有効期間に従い優先度のもっとも低いメッセージを破棄する (生成元のクライアントはメッセージの削除に関する通知は受信しない)</p> <p>REJECT_NEWEST - 最新のメッセージを拒否する (プロデュースングクライアントは、持続メッセージの場合は拒否に関する例外を受け取るが、持続メッセージ以外の場合は通知を受信しない)</p>
キューとトピック	maxBytesPerMsg	-1 (無制限)	送信先で許容されるシングルメッセージの最大サイズをバイト単位で指定する。プロデュースングクライアントは持続メッセージの場合は拒否に関する例外を受け取るが、持続メッセージ以外の場合は通知を受信しない
キューとトピック	maxNumProducers <sup>1</sup>	-1 (無制限)	送信先で許容されるプロデューサの最大数を指定する。この制限に達すると、新しいプロデューサを作成できない

1. クラスタ環境では、このプロパティはクラスタ内のすべてのインスタンスに一括して適用されるのではなく、クラスタ内の送信先の各インスタンスに適用されます。

表 6-10 送信先の属性 ( 続き )

送信先タイプ	属性	デフォルト値	説明
キューのみ	maxNumActiveConsumers	1	ロードバランスされたキュー送信先からの配信でアクティブにできるコンシューマの最大数を指定する。値を -1 に設定した場合、無制限になる (Platform Edition では、この値は 2 に制限される)
キューのみ	maxNumBackupConsumers	0	キュー送信先からのロードバランスされた配信で障害が生じた場合に、アクティブコンシューマに代わることができるバックアップコンシューマの最大数を指定する。値を -1 に設定した場合、無制限になる
キューとトピック	consumerFlowLimit	トピック : 1000  キュー : 1000	シングルバッチでコンシューマに配信されるメッセージの最大数を指定する。ロードバランスされたキュー配信では、ロードバランスされる前に、最初にキューに入ることができるアクティブコンシューマにルートされるメッセージの数となる (81 ページの「複数のコンシューマへのキュー配信」を参照)。この制限は、各コネクションの送信先のコンシューマに対して小さい値を指定することでオーバーライドできる (『Message Queue Java Client Developer's Guide』のコネクションファクトリ属性の説明を参照)。値を -1 に設定した場合、無制限になる
キューのみ	localDeliveryPreferred	false	ブローカクラスタ内のロードバランスされたキュー配信にだけ適用される。ローカルブローカ上にコンシューマが存在しない場合にだけ、メッセージがリモートコンシューマに配信されるように指定する。送信先をローカルだけの配信に制限してはならない (isLocalOnly = false)

1. クラスタ環境では、このプロパティはクラスタ内のすべてのインスタンスに一括して適用されるのではなく、クラスタ内の送信先の各インスタンスに適用されます。

表 6-10 送信先の属性 ( 続き )

送信先タイプ	属性	デフォルト値	説明
キューとトピック	isLocalOnly	false	ブローカクラスタに対してのみ適用。送信先がそのほかのブローカに複製されないように指定する。さらに、メッセージの配信をローカルコンシューマ ( 送信先の作成元にあるブローカに接続されたコンシューマ ) だけに制限する。いったん送信先が作成されると、この属性は更新できない

1. クラスタ環境では、このプロパティはクラスタ内のすべてのインスタンスに一括して適用されるのではなく、クラスタ内の送信先の各インスタンスに適用されます。

- 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 "maxNumActiveConsumers=5"
```

送信先名は、英数字 ( 空白文字は含まない ) だけを使用でき、英字、または記号「\_」と「\$」で開始できます。文字列「mq\_」はメトリックストピック送信先用に予約されているため、この文字列で開始することはできません (76 ページの表 2-8 を参照)。
- 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"
```

定義の一覧表示

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

情報を取得したい送信先を検索するために、最初に `list dst` サブコマンドを使用して特定のブローカ上のすべての送信先を一覧表示できます。たとえば、`myHost` のポート 4545 上で実行しているブローカ上の送信先すべてのリストを取得するには次のコマンドを入力します。

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

`list dst` サブコマンドを使用すると、任意で、一覧表示する送信先のタイプを指定したり、一次送信先を含めたりすることができます ( `-tmp` オプションを使用 )。一次送信先はクライアントによって作成され、通常は、そのほかのクライアントへ送信されたメッセージへの返信を受信することを目的としています (85 ページの「一時送信先」を参照)。

# 送信先情報の表示

送信先の現在の属性値に関する情報を取得するには、次に示すように、`query dst` サブコマンドを使用します。

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

このコマンドでは、次のような情報が出力されます。

-----	
Destination Name	Destination Type
-----	
XQueue	Queue
On the broker specified by:	
-----	
Host	Primary Port
-----	
localhost	7676
Destination Name	XQueue
Destination Type	Queue
Destination State	RUNNING
Created Administratively	true
Current Number of Messages	0
Current Total Message Bytes	0
Current Number of Producers	0
Current Number of Active Consumers	0
Current Number of Backup Consumers	0
Max Number of Messages	unlimited (-1)
Max Total Message Bytes	unlimited (-1)
Max Bytes per Message	unlimited (-1)
Max Number of Producers	100
Max Number of Active Consumers	1
Max Number of Backup Consumers	0
Limit Behavior	REJECT_NEWEST
Consumer Flow Limit	100
Is Local Destination	false
Local Delivery is Preferred	false

また、出力は送信先に関連付けられたプロデューサとコンシューマの数を示しています。キュー送信先に関しては、アクティブコンシューマとバックアップコンシューマの両方が含まれています。

`update dst` サブコマンドを使用すると、1 つ以上の属性の値を変更できます (184 ページの「[送信先の属性の更新](#)」を参照)。

## 送信先の属性の更新

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" -u admin -p admin
```

更新が可能な属性については、[180 ページの表 6-10](#) を参照してください。

送信先の *type* や `isLocalOnly` 属性を更新する場合、`update dst` サブコマンドは使用できません。

## 送信先のメトリックスの表示

送信先に関するメッセージメトリックス情報を取得するには、次に示すように、`metrics dst` サブコマンドを使用します。

```
imqcmd metrics dst -t q -n XQueue -m ttl -u admin -p admin
```

このコマンドでは、次のような情報が出力されます。

-----										
Msgs		Msg Bytes		Msg Count			Total Msg Bytes			
(k)		Largest								
In	Out	In	Out	Current	Peak	Avg	Current	Peak	Avg	Msg (k)
200	200	147200	147200	0	200	0	0	143	71	0
300	200	220800	147200	100	200	10	71	143	64	0
300	300	220800	220800	0	200	0	0	143	59	0

`imqcmd` を使用して送信先のメトリックスをレポートする方法の詳細は、[256 ページの「監視ツール」](#) を参照してください。

## 送信先の停止と再開

プロデューサから送信先、送信先からコンシューマ、またはその両方のメッセージの配信を制御するために、送信先を停止できます。特に、メッセージのプロデュースがコンシュームよりかなり高速な場合に、送信先がメッセージによって過負荷にならないように、送信先へのメッセージフローを停止できます。

送信先へ、または送信先からのメッセージの配信を停止するには、次に示すように、`pause dst` サブコマンドを使用します。

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

送信先を停止しており、配信を再開したい場合は、次のコマンドを入力します。

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

マルチブローカ クラスタでは、送信先のインスタンスはクラスタ内の各ブローカに常駐します。これらの送信先はそれぞれ個別に停止する必要があります。

## 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。

送信先でメッセージをパージするには、次に示すように、`purge dst` サブコマンドを使用します。

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

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

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

これで、ブローカを再起動すると、送信先のパージに関する問題が解消されます。

マルチブローカ クラスタでは、送信先のインスタンスはクラスタ内の各ブローカに常駐します。これらの送信先はそれぞれ個別にパージする必要があります。

## 送信先の破棄

送信先を破棄するには、次に示すように、`destroy dst` サブコマンドを使用します。

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

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。

## 送信先の圧縮

メッセージの持続ストアとして、プラグインされた JDBCS 互換のデータストアではなく、組み込みのファイルベースのデータストアを使用している場合は、ディスク利用率を監視し、必要に応じてディスクを圧縮できます。

ファイルベースのメッセージストアは、保持される送信先に応じてメッセージがディレクトリに格納されるように構成されています。各送信先のディレクトリでは、大半のメッセージが可変長のレコードから成る 1 つのファイル、つまり可変長のレコードファイルに格納されます。断片化を減らすため、サイズが設定可能なしきい値を超えているメッセージは専用の個別のファイルに格納されます。可変サイズのメッセージが保持されていて、その後可変長のレコードファイルから削除された場合、空きレコードが再利用されていないファイルに空白ができることがあります。

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

## 送信先のディスク利用率の監視

送信先のディスク利用率を監視するには、次の `imqcmd` のサブコマンドを使用します。

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

このコマンドでは、次のような情報が出力されます。

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

サブコマンド出力の各列の意味は次のとおりです。



表 6-11 送信先ディスク利用率のメトリックス

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

### 未使用の送信先ディスクスペースの再利用

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

メッセージのプロデュースレートがメッセージのコンシュームレートを大きく上回る場合は、一般に、空きレコードが再利用され利用率が高くなります。ただし、メッセージのプロデュースレートがメッセージのコンシュームレートと同程度かそれより低い場合は、利用率は低いと予測できます。

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

予約済みのディスクスペースが時間の経過とともに増え続けている場合は、送信先メモリの制限プロパティと制限動作を設定して送信先のメモリ管理を設定し直す必要があります (180 ページの表 6-10 を参照)。

#### ► 未使用の送信先ディスクスペースを再利用するには

1. 送信先を停止します。

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

2. ディスクを圧縮します。

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

3. 送信先を再開します。

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

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

# 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 6-12 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
<code>list dur -d destName</code>	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
<code>destroy dur -n subscrName -c client_id</code>	特定のクライアント識別子を持つ特定の永続サブスクリプションを破棄する (47 ページの「クライアント識別子」を参照)
<code>purge dur -n subscrName -c client_id</code>	特定のクライアント識別子を持つ特定の永続サブスクリプションのすべてのメッセージをパージする (47 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
```

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

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

list dur サブコマンドから返される情報を使用して、破棄する必要がある永続サブスクリプションやメッセージをパージする必要がある永続サブスクリプションを識別することができます。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 Message Queue の単純なトランザクション、または分散トランザクションです (49 ページの「ローカルトランザクション」を参照)。各トランザクションには、Message Queue トランザクション ID が付けられています。これは、ブローカのトランザクションを一意に識別するための 64 ビットの数字です。また、分散トランザクションには、分散トランザクションマネージャによって割り当てられる最大 128 バイトの分散トランザクション ID (XID) が付けられます。Message Queue は、Message Queue トランザクション ID と XID の関連付けを保持します。

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

表 6-13 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 6-13** 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

表 6-13 トランザクションを管理する imqcmd のサブコマンド ( 続き )

サブコマンド	説明
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、ユーザー名、メッセージまたは通知の数、および作成時間を返します。たとえば、次のように表示されます。

Transaction ID	State	User name	# Msgs/ # Acks	Creation time
64248349708800	PREPARED	guest	4/0	1/30/02 10:08:31 AM
64248371287808	PREPARED	guest	0/4	1/30/02 10:09:55 AM

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

たとえば、ブローカの自動ロールバックプロパティを false に設定した場合 (65 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、同じ情報のほかに、クライアント ID、セッション識別子、分散トランザクション ID (XID) などの多数の追加された値を確認できます。たとえば、次のように指定します。

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

次のような情報が出力されます。

```
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
```

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、[65 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。



# 管理対象オブジェクトの管理

管理対象オブジェクトを使用すると、ほかの JMS プロバイダへの移植が可能なクライアントアプリケーションを開発できます。管理対象オブジェクトは、プロバイダ固有の設定およびネーミング情報をカプセル化するオブジェクトです。通常、これらのオブジェクトは **Message Queue** 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、[93 ページの「Message Queue 管理対象オブジェクト」](#)を参照してください。

**Message Queue** は、管理対象オブジェクトを作成したり管理したりするための 2 つの管理ツールを提供しています。それは、コマンド行オブジェクトマネージャユーティリティ (`imqobjmgr`) と GUI 管理コンソールです。これらのツールでは、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ーおよび表示
- オブジェクトストアにある既存の管理対象オブジェクトの変更

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

管理コンソールの使用法については、[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 サーバオブジェクトストア

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 サーバは、運用環境で役立つセキュリティ機能も備えています。

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

また、LDAP サーバをオブジェクトストアとして使用している場合は、表 7-1 に示す属性を指定する必要があります。これらの属性は、次のように分類されます。

- **初期コンテキスト**: LDAP サーバオブジェクトストアの場合、この属性は固定である
- **ロケーション**: LDAP サーバの設定時に、管理対象オブジェクトの URL とディレクトリパスを指定する。特に、指定したパスが存在することを確認する必要があります
- **セキュリティ情報**: LDAP プロバイダによって異なる。セキュリティ情報をすべての操作で必要とするのか、あるいは格納データを変更する操作にだけ必要とするのか決める場合、使用する LDAP 実装に付属するマニュアルを参照する必要があります

表 7-1 LDAP オブジェクトストアの属性

属性	説明
java.naming.factory.initial	LDAP サーバ上の JNDI 検索用の初期コンテキスト com.sun.jndi.ldap.LdapCtxFactory



表 7-1 LDAP オブジェクトストアの属性 ( 続き )

属性	説明
java.naming.provider.url	<p>LDAP サーバの URL とディレクトリパス情報。たとえば、次のように指定する</p> <pre>ldap://mydomain.com:389/ou=mqobjs, o=myapp</pre> <p>管理対象オブジェクトは、/myapp/mqobjs ディレクトリに格納される</p>
java.naming.security.principal	<p>LDAP サーバの呼び出し元を認証するための主体の識別情報。このエントリの形式は、認証スキーマによって異なる。たとえば、次のように指定する</p> <pre>uid=fooUser, ou=People, o=mq</pre> <p>このプロパティを指定しない場合は、LDAP サービスプロバイダによって動作が決定される</p>
java.naming.security.credentials	<p>LDAP サーバの呼び出し元を認証するための主体の証明書。プロパティの値は、認証スキーマによって異なる。ハッシュ化されたパスワード、クリアテキストのパスワード、キー、証明書などが使用できる。たとえば、次のように指定する</p> <pre>fooPasswd</pre> <p>このプロパティを指定しない場合は、LDAP サービスプロバイダによって動作が決定される</p>
java.naming.security.authentication	<p>使用するセキュリティのレベル。値は、none、simple、strong のどれかのキーワードになる</p> <p>たとえば、simple を指定した場合、値がまだ指定されていない主体または証明書の値を入力するよう要求される。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</p> <p>このプロパティを指定しない場合は、LDAP サービスプロバイダによって動作が決定される</p>

## ファイルシステムオブジェクトストア

Message Queue は、ファイルシステムのオブジェクトストア実装もサポートしています。ファイルシステムのオブジェクトストアは、まだ十分なテストが行われていないため、運用システムでの使用はお勧めしませんが、開発環境で使いやすいという利点があります。LDAP サーバをセットアップする必要はなく、ローカルのファイルシステム上にディレクトリを作成するだけで利用できます。

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

ファイルシステムオブジェクトストアを使用している場合は、[表 7-2](#) に示す属性を指定する必要があります。これらの属性は、次のように分類されます。

- **初期コンテキスト** : ファイルシステムオブジェクトストアの場合、この属性の値は固定である
- **ロケーション** : この属性の値は、使用する管理対象オブジェクトを格納するディレクトリパスを指定する。ディレクトリが存在していて、**Message Queue** 管理ツールのユーザーと、ストアにアクセスするクライアントアプリケーションのユーザーは適切なアクセス権を持っている必要がある

表 7-2            ファイルシステムオブジェクトストアの属性

属性	説明
java.naming.factory.initial	ファイルシステムオブジェクトストアの JNDI 検索の初期コンテキスト  com.sun.jndi.fscontext. RefFSContextFactory
java.naming.provider.url	ディレクトリパス情報。たとえば、次のように指定する  file:///C:/myapp/mqobjs

# 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、[93 ページ](#)の「[Message Queue 管理対象オブジェクト](#)」を参照してください。

Message Queue の管理対象オブジェクトには 2 つの基本的な種類があります。それは、コネクションファクトリと送信先です。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、『[Message Queue Java Client Developer's Guide](#)』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、キューコネクションファクトリ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、トピックコネクションファクトリ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、[47 ページ](#)の表 1-1 を参照)。

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の属性

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](#) に示す属性があります。主に使用する属性は、`imqAddressList` です。この属性を使用して、クライアントがコネクションを確立するブローカを指定します。[206 ページ](#)の「[コネクションファクトリの追加](#)」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、『[Message Queue Java Client Developer's Guide](#)』と次の Message Queue クラスに関する JavaDoc API ドキュメントを参照してください。

`com.sun.messaging.ConnectionConfiguration`

表 7-3 コネクションファクトリ管理対象オブジェクトの属性

属性 / プロパティ名	データ型	デフォルト値
imqAckOnAcknowledge	文字列型	値なし
imqAckOnProduce	文字列型	値なし
imqAckTimeout	文字列型	0 ミリ秒
imqAddressList	文字列型	値なし
imqAddressListIterations	整数型	1
imqAddressListBehavior	文字列型	PRIORITY
imqBrokerHostName (Message Queue 3.0)	文字列型	localhost
imqBrokerHostPort (Message Queue 3.0)	整数型	7676
imqBrokerServicePort (Message Queue 3.0)	整数型	0
imqConfiguredClientID	文字列型	値なし
imqConnectionFlowCount	整数型	100
imqConnectionFlowLimit	整数型	1000
imqConnectionFlowLimitEnabled	ブール型	false
imqConnectionType (Message Queue 3.0)	文字列型	TCP
imqConnectionURL (Message Queue 3.0)	文字列型	http://localhost/imq/ tunnel
imqConsumerFlowLimit	整数型	1000
imqConsumerFlowThreshold	整数型	50
imqDefaultPassword	文字列型	guest
imqDefaultUsername	文字列型	guest
imqDisableSetClientID	ブール型	false
imqJMSDeliveryMode	整数型	2 ( 持続的 )
imqJMSExpiration	ロング型	0 ( 有効期限なし )
imqJMSPriority	整数型	4 ( 標準 )
imqLoadMaxToServerSession	ブール型	true
imqOverrideJMSDeliveryMode	ブール型	false
imqOverrideJMSExpiration	ブール型	false

表 7-3 コネクションファクトリ管理対象オブジェクトの属性 ( 続き )

属性 / プロパティ名	データ型	デフォルト値
imqOverrideJMSHeadersToTemporaryDestinations	ブール型	false
imqOverrideJMSPriority	ブール型	false
imqQueueBrowserMaxMessagesPerRetrieve	整数型	1000
imqQueueBrowserRetrieveTimeout	ロング型	60,000 ( ミリ秒 )
imqReconnectAttempts	整数型	0
imqReconnectEnabled	ブール型	false
imqReconnectInterval	ロング型	3000 ( ミリ秒 )
imqSetJMSXAppID	ブール型	false
imqSetJMSXConsumerTXID	ブール型	false
imqSetJMSXProducerTXID	ブール型	false
imqSetJMSXRcvTimestamp	ブール型	false
imqSetJMSXUserID	ブール型	false
imqSSLIsHostTrusted (Message Queue 3.0)	ブール型	true

## 送信先管理対象オブジェクトの属性

物理的なトピックやキューの送信先を指定する、送信先管理対象オブジェクトには、[表 7-4](#) に示すような属性があります。[207 ページの「トピックまたはキューの追加」](#)では、送信先管理対象オブジェクトをオブジェクトストアに追加する場合に、これらの属性を指定する方法について説明します。

主に使用する属性は、`imqDestinationName` です。これは、トピックまたはキューの管理対象オブジェクトに対応する物理的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、`Message Queue` クラスの `com.sun.messaging.DestinationConfiguration` に関する `JavaDoc API` ドキュメントを参照してください。

表 7-4 送信先管理対象オブジェクトの属性

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

表 7-4 送信先管理対象オブジェクトの属性 (続き)

属性 / プロパティ名	データ型	デフォルト値
imqDestinationName	文字列 <sup>1</sup>	Untitled_Destination_Object

1. 送信先名には、英数字 (空白文字は含まない) だけを使用できます。送信先名は、英字や、「\_」または「\$」で開始する必要があります。

# 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、Message Queue の管理対象オブジェクトを作成および管理することができます。この節では、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、imqobjmgr サブコマンドを使用して、特定のタスクを実行する方法について説明します。

## imqobjmgr コマンドの構文

imqobjmgr コマンドの一般的な構文は、次のとおりです。

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

-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。

```
imqobjmgr 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 7-5 に示すようなサブコマンドが含まれています。

表 7-5 imqobjmgr サブコマンド

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

表 7-5 imgobjmgr サブコマンド ( 続き )

サブコマンド	説明
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
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

## imgobjmgr コマンドのオプションの概要

表 7-6 に、imgobjmgr 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 7-6 imgobjmgr のオプション

オプション	説明
-f	ユーザーの確認なしで、アクションを実行する
-h	使用方法に関するヘルプを表示する。コマンド行ではそれ以外のことは実行されない
-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 コマンド行ではそれ以外のことは実行されない
-i <i>fileName</i>	オブジェクトタイプ、検索名、オブジェクト属性、オブジェクトストア属性などのオプションを指定するサブコマンド句の一部またはすべてを含むコマンドファイルの名前を指定する。通常、オブジェクトストア属性などの反復の多い情報に使用される
-j <i>attribute=value</i>	JNDI オブジェクトストアを識別しアクセスするために必要な属性を指定する。 <a href="#">194 ページ</a> の「LDAP サーバオブジェクトストア」および <a href="#">195 ページ</a> の「ファイルシステムオブジェクトストア」を参照
-javahome <i>path</i>	使用する代替の Java 2 互換のランタイムを指定する。デフォルトではシステム上のランタイムまたは Message Queue にバンドルされたランタイムを使用する
-l <i>lookupName</i>	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

表 7-6          imgobjmgr のオプション ( 続き )

オプション	説明
-o <i>attribute=value</i>	管理対象オブジェクトの属性を指定する。197 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属性」および 199 ページの「送信先管理対象オブジェクトの属性」を参照
-pre	プレビューモード。コマンドを実行せずに、実行される内容を示す
-r <i>read-only_state</i>	管理対象オブジェクトが読み取り専用オブジェクトかどうかを指定する。値 <code>true</code> は、管理対象オブジェクトが読み取り専用オブジェクトであることを示す。クライアントは読み取り専用管理対象オブジェクトの属性は変更できない。デフォルトでは、読み取り専用の状態は <code>false</code> に設定されている
-s	サイレントモード。出力が表示されない
-t <i>objectType</i>	Message Queue 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 = connection factory qf = queue connection factory tf = topic connection factory xcf = XA connection factory ( 分散トランザクション ) xqf = XA queue connection factory ( 分散トランザクション ) xtf = XA topic connection factory ( 分散トランザクション ) e = SOAP 終端 <sup>1</sup>
-v	バージョン情報を表示する。コマンド行ではそれ以外のことは実行されない

1. 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 (『Message Queue Java Client Developer's Guide』を参照)。

次に、imgobjmgr 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タイプ**

使用可能なタイプを表 7-6 に示します。

- **管理対象オブジェクトの JNDI 検索名**

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

- **管理対象オブジェクトの属性 (特に、add および update サブコマンドで必要)**

- 送信先の場合: ブローカの物理的な送信先の名前。これは imqcmd create dst サブコマンドの -n オプションで指定した名前となる。この名前を指定しない場合、デフォルト名の Untitled\_Destination\_Object が使用されます。
- コネクションファクトリの場合: もっとも一般的に使用される属性は、1 つ以上のメッセージサーバアドレスをクライアントの接続先に指定するためのアドレスリスト (imqAddressList) である。この情報を指定しないと、ローカルホストとデフォルトのポート番号 (7676) が使用される。つまり、クライアントはローカルホスト上のポート 7676 のブローカに対してコネクションの確立を試みる。[206 ページの「コネクションファクトリの追加」](#)では、オブジェクトの属性を指定する方法について説明する。

その他の属性については、[197 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属性」](#)を参照してください。

- **オブジェクトストアの属性**

この情報は、ファイルシステムストアと LDAP サーバ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ー名、パスワード、および認証タイプ

オブジェクトストアの属性については、[194 ページの「LDAP サーバオブジェクトストア」](#)および [195 ページの「ファイルシステムオブジェクトストア」](#)を参照してください。

## コマンドファイルの使用

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

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

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

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

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

```
imgobjmgr add
-t qf
-l "cn=myQCF"
-o "imgAddressList=mq://foo:777/jms"
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=img"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=img"
-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
-o "imqAddressList=mq://foo:777/jms"
objstore.attrs.java.naming.factory.initial=¥
com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=¥
ldap://mydomain.com:389/o=imq
objstore.attrs.java.naming.security.principal=¥
uid=fooUser, ou=People, o=imq
objstore.attrs.java.naming.security.credentials=fooPasswd
objstore.attrs.java.naming.security.authentication=simple
```

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

```
imqobjmgr -i MyCmdFile
```

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

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

この場合、MyCmdFile ファイルには次の定義が含まれます。

```
version=2.0
objstore.attrs.java.naming.factory.initial=¥
com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=¥
ldap://mydomain.com:389/o=imq
```

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

コマンドファイルの別の例は、次の場所で参照できます。

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。

---

**注** オブジェクトマネージャユーティリティ (imqobjmgr) は、Message Queue 管理対象オブジェクトだけを一覧表示します。オブジェクトストアに、追加したい管理対象オブジェクトと同じ検索名の Message Queue 以外のオブジェクトが含まれている場合は、追加操作を実行するとエラーが表示されます。

---

## コネクションファクトリの追加

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

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

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

上記のコマンドでは、検索名が `cn=myQCF` で、`myHost` 上で実行され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。引数としてコマンドファイルに `imgobjmgr` コマンドを指定すると、同じことを実行できます。詳細は、[204 ページの「コマンドファイルの使用」](#)を参照してください。

**注**      **命名規則** : LDAP サーバを使用して、管理対象オブジェクトを格納する場合、前述の例のように、接頭辞「`cn=`」が付いた検索名 (`cn=myQCF`) を割り当てることが重要です。検索名は、`-l` オプションを使用して指定します。ファイルシステムオブジェクトストアを使用している場合は、`cn` 接頭辞を使用する必要はありません。ただし、「`/`」を含む検索名は使用しないでください。[表 7-7](#) を参照してください。

**表 7-7**      命名規則の例

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

## トピックまたはキューの追加

クライアントアプリケーションがブローカ上の物理的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 (`imgcmd`) 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、[176 ページの「コネクション情報の取得」](#)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。

```
imgobjmgr add
-t t
-l "cn=myTopic"
-o "imgDestinationName=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 というコマンドファイルを使用して、サブコマンド句を指定できます。ファイルには、次のテキストが含まれます。

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

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

```
imqobjmgr -i MyCmdFile
```

---

**注** LDAP サーバを使用して、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。

```
imgobjmgr 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=img"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=img"
-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。

```
imgobjmgr query
-l "cn=myTopic"
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=imgq"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=imgq"
-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。

```
imgobjmgr update
-t tf
-l "cn=MyTCF"
-o imgReconnectAttempts=3
-j "java.naming.factory.initial=
    com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
    ldap://mydomain.com:389/o=imgq"
-j "java.naming.security.principal=
    uid=fooUser, ou=People, o=imgq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```



# セキュリティの管理

この章では、セキュリティに関連するタスクを実行する方法を説明します。具体的には、認証、承認、および暗号化について取り上げます。

**ユーザーの認証**：管理者は、ユーザー、ユーザーグループ、およびパスワードのリストをユーザーリポジトリに保持しておく責任があります。ブローカインスタンスごとに異なるユーザーリポジトリを使用できます。この章の最初の節では、ユーザーリポジトリの作成、設定、および管理の方法を説明します。**Message Queue** セキュリティについては、[69 ページの「セキュリティマネージャ」](#)を参照してください。

**ユーザーの承認：アクセス制御プロパティファイル**：管理者は、アクセス制御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ーのアクセスをユーザー名またはグループメンバーシップにマッピングします。ブローカインスタンスごとに異なるアクセス制御プロパティファイルを使用できます。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 ベースのサービスとの連動 (Enterprise Edition)**：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。**Message Queue** による暗号化の処理方法については、[72 ページの「暗号化 \(Enterprise Edition\)」](#)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 ユーザーリポジトリ、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を提供します。

- ブローカの起動時に、システムがユーザーに要求する
- 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 ([141 ページの「ブローカの起動」](#) および [143 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 ([236 ページの「passfile の使用」](#) を参照)

# ユーザーの認証

ユーザーがブローカーへの接続を試みると、ブローカーは提供された名前とパスワードを調べて、それぞれ参照するように設定されているブローカー固有のユーザーリポジトリの名前およびパスワードと一致した場合に、コネクションを許可します。このリポジトリには、次の2つのタイプがあります。

- Message Queue に付属している単層型ファイルリポジトリ

このタイプのユーザーリポジトリはかなり簡単に使用できますが、セキュリティ攻撃に対して脆弱なため、評価および開発を目的とする場合にだけ使用してください。ユーザーマネージャユーティリティ (`imqusermgr`) を使用してリポジトリを設定および管理します。認証を有効にするには、ユーザーリポジトリにユーザー名、パスワード、およびユーザーグループの名前を設定します。

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

- LDAP サーバ

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

LDAP ユーザーリポジトリを使用している場合、LDAP ベンダーから提供されているツールを使用して、ユーザーリポジトリを設定、管理する必要があります。詳細は、[221 ページ](#)の「[ユーザーリポジトリに 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ーリポジトリを使用する

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

### ユーザーリポジトリの作成

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

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

表 8-1 に示すように、リポジトリは 2 つのエントリ (行) 構成されます。

表 8-1 ユーザーリポジトリでの初期エントリ

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

これらの初期エントリにより、管理者が介入しなくても、インストール後直ちに Message Queue ブローカを使用できます。言い換えれば、Message Queue ブローカを使用する上で、ユーザーやパスワードの初期設定は必要ありません。

たとえばテストの目的で、初期設定された guest ユーザーエントリを使って、クライアントはデフォルトのユーザー名とパスワード guest でブローカに接続できます。

初期設定された admin ユーザーエントリでは、デフォルトのユーザー名とパスワード admin で、`imqcmd` コマンドを使用してブローカを管理できます。この初期設定された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 (220 ページの「[デフォルトの管理者パスワードの変更](#)」を参照)。

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

## ユーザーマネージャユーティリティ (imqusermgr)

ユーザーマネージャユーティリティ (imqusermgr) を使って、単層型ファイルユーザーリポジトリを編集したり設定したりできます。

この節では、imqusermgr コマンドの基本構文、サブコマンドのリスト、imqusermgr 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、imqusermgr サブコマンドを使用して、特定のタスクを実行する方法について説明します。

imqusermgr の使用の先立ち、次の点に留意してください。

- ブローカ固有のユーザーリポジトリが存在していない場合は、それを作成するために該当するブローカインスタンスを起動する必要がある
- imqusermgr コマンドは、ブローカがインストールされているホスト上で実行する必要がある
- 管理者はリポジトリに書き込むための適切なアクセス権を持っている必要がある。すなわち、Solaris と Linux では、root ユーザーまたは最初にブローカインスタンスを作成したユーザーでなければならない。

### imqusermgr コマンドの構文

imqusermgr コマンドの一般的な構文は、次のとおりです。

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

-v、または -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。

```
imqusermgr list -v
```

imqusermgr サブコマンド

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 [-i instanceName] -u userName -p passwd [-g group] [-s]	ユーザーとそのパスワードを指定した、またはデフォルトのブローカインスタンスリポジトリに追加し、オプションでユーザーグループを指定する
delete [-i instanceName] -u userName [-s] [-f]	指定したユーザーを、指定した、またはデフォルトのブローカインスタンスリポジトリから削除する
list [-i instanceName] [-u userName]	指定した、またはデフォルトのブローカインスタンスリポジトリの指定したユーザーまたはすべてのユーザーに関する情報を表示する
update [-i instanceName] -u userName -p passwd [-a state] [-s] [-f]	指定した、またはデフォルトのブローカインスタンスリポジトリの指定ユーザーのパスワードまたは状態、もしくは両方を更新する
update [-i instanceName] -u userName -a state [-p passwd] [-s] [-f]	

注 次 の 節 の 例 は、デフォルトのブローカインスタンスを前提としています。

imqusermgr コマンドのオプションの概要

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 <i>active_state</i>	ユーザーの状態をアクティブにするかどうかを指定する (true/false)。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	ユーザーの確認なしで、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。コマンド行ではそれ以外のことは実行されない
-i <i>instanceName</i>	コマンドを適用するブローカインスタンスユーザーリポジトリを指定する。指定しない場合は、デフォルトの <i>instanceName</i> 、imqbroker が使用される
-p <i>passwd</i>	ユーザーのパスワードを指定する
-g <i>group</i>	ユーザーグループを指定する。指定できる値は、admin、user、anonymous
-s	サイレントモードに設定する
-u <i>userName</i>	ユーザー名を指定する
-v	バージョン情報を表示する。コマンド行ではそれ以外のことは実行されない

## グループ

ブローカインスタンスのユーザーリポジトリにユーザーエントリを追加する場合は、オプションでユーザーに定義済みの 3 つのグループ、つまり、admin、user、anonymous のどれかを指定できます。グループが指定されない場合、デフォルトの user が割り当てられます。

- **admin グループ**: ブローカの管理者用。このグループに割り当てられたユーザーは、デフォルトでブローカを設定および管理できるようになっています。管理者は、複数のユーザーを admin グループに割り当てることができます。
- **user グループ**: 管理ユーザーでない Message Queue 通常のクライアントユーザー用。ほとんどのクライアントユーザーは、user グループで認証されたブローカにアクセスします。デフォルトでは、このグループのアプリケーションクライアントユーザーは、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を実行できます。

- **anonymous グループ**: ブローカが認識しているユーザー名を使用しない Message Queue クライアント用。クライアントアプリケーションが実際に使用するユーザー名を認識していないなどの理由がある場合に使います。このグループは、多くの FTP サーバにある匿名アカウントに似ています。一度に anonymous グループに割り当てられるのは、1 人のユーザーだけです。user グループと比較してアクセス権を制限するか、配置時にこのグループからユーザーを削除すること推奨します。

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

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、[224 ページの「ユーザーの承認: アクセス制御プロパティファイル」](#)を参照してください。

状態

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

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

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

ユーザー名とパスワードの形式

ユーザー名とパスワードは、次の規則に従う必要があります。

- ユーザー名には、[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 ユーザー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
:	コロンの

- ユーザー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



- ユーザー名やパスワードに空白を入れる場合は、ユーザー名やパスワード全体を引用符で囲む
- ユーザー名やパスワードは、1 文字以上であることが必要
- 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ー名の長さに制限はない

## ユーザーリポジトリの設定と管理

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

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

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

```
imqusermgr delete -u Bob
```

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

```
imqusermgr update -u Katharine -p alladin
```

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

```
imqusermgr list -u isa
```

```
% imqusermgr list -u isa

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

次のコマンドでは、すべてのユーザーに関する情報が表示されます。

```
imqusermgr list
```

```
% imqusermgr list

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

## デフォルトの管理者パスワードの変更

セキュリティのために、admin のデフォルトパスワードを自分だけが知っているパスワードに変更する必要があります。この変更を行うには、imqusermgr ツールを使用します。

次のコマンドは、mybroker ブローカインスタンスのデフォルトのパスワードを grandpoobah に変更します。

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

ブローカインスタンスの実行中に任意のコマンド行ツールを実行すれば、この変更が反映されていることをすぐに確認できます。たとえば、次のコマンドを実行します。

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

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 Message Queue 管理ツールを使用するときに、新しいパスワードを使用してください。

## ユーザーリポジトリに LDAP サーバを使用する

ユーザーリポジトリに 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、ユーザーがブローカインスタンスへ接続しようとしたり、特定のメッセージング操作を実行したりすると、ブローカインスタンスがユーザーとグループに関する情報について LDAP にクエリー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 (config.properties) は、設定ファイルが関連付けられているブローカインスタンスの名前 (instanceName) によって識別されたディレクトリに書き込まれます ( [付録 A「Message Queue データの場所」](#) を参照 )。

```
.../instances/instanceName/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 ユーザーリポジトリの使用を指定します。

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

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

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

3. LDAP へのアクセスを制御するブローカプロパティを設定する必要もあります。こ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。Message Queue は、JNDI API を使用して、LDAP ディレクトリサーバと通信します。このプロパティで使用されている構文と用語の詳細については JNDI のドキュメントを参照してください。Message Queue は、Sun JNDI LDAP プロバイダと簡単な認証を使用しています。

Message Queue は、LDAP 認証フェイルオーバーをサポートしているため、認証の対象とする LDAP ディレクトリサーバのリストを指定できます ( [表 8-5](#) の imq.user.repos.ldap.server プロパティを参照 )。

表 8-5 LDAP 関連プロパティ

プロパティ	説明
imq.user_repository. ldap.server	LDAP サーバの場合、 <i>host:port</i> の <i>host</i> にはディレクトリサーバを実行しているホストの完全指定 DNS 名を指定し、 <i>port</i> にはディレクトリサーバが通信に使用しているポート番号を指定する。フェイルオーバーサーバのリストを指定するには、次の構文を使用する。 <i>host1:port1 ldap://host2:port2 ldap://host3:port3...</i> リスト内のエントリはスペースで区切る。2 番目以降の各フェイルオーバーサーバアドレスは <i>ldap</i> で始まることに注意する
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
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。このプロパティは <i>passfile</i> 内だけで指定可能 (236 ページの「 <i>passfile</i> の使用」を参照)  さまざまな方法でパスワードを指定できる。もっとも安全な方法は、ブローカプロンプトでパスワードを入力することである。安全性は低くなるが、パスファイルを使用してパスファイルを読み取り保護することもできる。安全性はもっとも低くなるが、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こともできる。 <i>imqbrokerd -ldappassword</i>  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	ユーザー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、その値はユーザーを一意に識別する。たとえば、 <i>uid</i> 、 <i>cn</i> 、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ー)。ユーザーに検索フィルタを指定すると、ブローカが検索範囲を絞り込めるので検索効率が上がる。詳細は、次の場所にある JNDI チュートリアルを参照 <a href="http://java.sun.com/products/jndi/tutorial">http://java.sun.com/products/jndi/tutorial</a>  このプロパティの設定は任意

表 8-5 LDAP 関連プロパティ ( 続き )

プロパティ	説明
imq.user_repository. ldap.grpsearch	グループ検索を有効にするかどうかを指定するブール値。ユーザーをグループに関連付けるかどうかを決定するには、LDAP プロバイダから提供されているマニュアルを参照  ネストされたグループは、Message Queue ではサポートされていないので注意する  デフォルト値: false
imq.user_repository. ldap.grpbases	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
imq.user_repository. ldap.grpfiltler	JNDI 検索フィルタ ( 論理式で表現される検索クエリー )。グループに検索フィルタを指定すると、ブローカが検索範囲を絞り込む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  <a href="http://java.sun.com/products/jndi/tutorial">http://java.sun.com/products/jndi/tutorial</a>  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 ユーザー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。

- 必要に応じて、アクセス制御プロパティファイルにあるユーザーまたはグループ、および規則を編集する必要があります。アクセス制御プロパティファイルの使用に関する詳細は、[224 ページの「ユーザーの承認: アクセス制御プロパティファイル」](#)を参照してください。
- 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 サーバが SSL 通信に使用するポートを指定する。たとえば、次のように指定する  
`imq.user_repository.ldap.server=myhost:7878`
- ブローカプロパティの `imq.user_repository.ldap.ssl.enabled` を `true` に設定する

## ユーザーの承認：アクセス制御プロパティファイル

ブローカインスタンスを接続したあと、ユーザー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。ユーザーがこういった操作を試行すると、ブローカはブローカ固有の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、ユーザーが操作を実行する承認を受けているか確認します。ACL ファイルには、特定のユーザー (またはユーザーのグループ) 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ーは、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。ACL ファイルを編集して、特定のユーザーやグループによるこれらの操作を制限できます。

ユーザー情報が単層型ファイルのユーザーリポジトリにあるか ([214 ページの「単層型ファイルユーザーリポジトリを使用する」](#)を参照) LDAP ユーザーリポジトリにあるか ([221 ページの「ユーザーリポジトリに LDAP サーバを使用する」](#)を参照) によって、異なった ACL ファイルが使用されます。

## アクセス制御プロパティファイルの作成

ACL ファイルはインスタンス固有です。起動するブローカインスタンスごとに、`accesscontrol.properties` という名前のデフォルトファイルが作成されます。この ACL プロパティは、その ACL ファイルが関連付けられているブローカインスタンスの名前 (`instanceName`) によって識別されたディレクトリに書き込まれます ([付録 A「Message Queue データの場所」](#)を参照)。

```
.../instances/instanceName/etc/accesscontrol.properties
```

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、特定のユーザー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
<i>resourceType</i>	connection、queue、topic のどれか
<i>resourceVariant</i>	<i>resourceType</i> 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
<i>operation</i>	公式化されているアクセス規則の種類に依存する値
<i>access</i>	allow か deny のどちらか
<i>principalType</i>	user か group のどちらか。詳細は、217 ページの「グループ」を参照
<i>principals</i>	規則の左側で指定されるアクセス権を保持するユーザーを示す。ここでは、principalType が user の場合は個々のユーザーまたはカンマで区切られたユーザーのリストとなり、principalType が group の場合は 1 つのグループまたは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、すべてのユーザー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、あらゆるユーザーがメッセージを ql という名前のキューに送信する

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

- 次の規則では、あらゆるユーザーがあらゆるキューにメッセージを送信する

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

---

**注** ASCII でないユーザー、グループ、または送信先の名前を指定するには、Unicode エスケープ (\uXXXX) の表記法を使用する必要があります。ASCII コードではない名前を含む ACL ファイルを編集して、保存した場合、Java native2ascii ツールを使用して、ファイルを ASCII に変換できます。詳細は、<http://java.sun.com/j2se/1.4/docs/guide/intl/faq.html> を参照してください。

---

## 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、ユーザー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 はメッセージをプロデュースできないが、その他のユーザーは 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
```

- ユーザーの \* *principal* 規則は、グループの対応する \* *principal* 規則をオーバーライドする。たとえば、次の 2 つの規則では、すべての認証済みユーザーがメッセージを tq1 に送信できる

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

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

- ユーザーに許可されたアクセスは、ユーザーのグループに許可されたアクセスをオーバーライドする。次の例では、Bob が User のメンバーである場合、tq1 にメッセージをプロデュースできないが、ほかの User のメンバーは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、すべてのユーザーはすべての操作を実行できない
- 同じユーザーまたはグループにアクセス権の拒否と許可を行うと、すべてが取り消される。たとえば、次の 2 つの規則では、結果として Bob が q1 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
```

*resourceVariant* には、NORMAL と ADMIN の 2 つの値が定義されています。デフォルトでは、すべてのユーザーが NORMAL タイプにアクセスできますが、admin グループのユーザーは、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、ユーザー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、ほかのユーザーはすべてアクセスが許可されます。

```
connection.NORMAL.deny.user=Bob
connection.NORMAL.allow.user=*
```

アスタリスク (\*) 文字を使用して、すべての認証済みユーザーまたはグループを指定できます。

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（ユーザーまたはグループ）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、あらゆるユーザーまたはグループが、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
<i>resourceType</i>	queue または topic のどちらか
<i>resourceVariant</i>	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 (*)
<i>operation</i>	produce、consume、または browse のどれか
<i>access</i>	allow または deny のどちらか
<i>principalType</i>	user または group のどちらか

アクセス権は、1 人以上のユーザーまたは 1 つ以上のグループ、あるいはその両方に対して指定できます。

次の例では、さまざまな種類の送信先アクセス制御規則を示します。

- すべてのユーザーが、あらゆるキュー送信先に対してメッセージを送信できる  
`queue.*.produce.allow.user=*`
- user グループのメンバーがトピック `Admissions` にサブスクライブするのを拒否する  
`topic.Admissions.consume.deny.group=user`

## 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、どのユーザー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。

ユーザー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、任意のユーザーやグループは、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、どのユーザーも送信先にメッセージを送信できないようにしてから、2) 送信先の自動作成を有効にすると、ブローカは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 ベースのサービスとの連動 (Enterprise Edition)

Message Queue Enterprise Edition は、Secure Socket Layer (SSL) 規格に基づいてコネクションサービスをサポートしています。SSL では、TCP/IP (`ssljms` および `ssladmin`) と HTTP (`httpsjms`) を使用できます。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 Message Queue 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 (`imkeytool`) 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。

Message Queue の SSL ベースのコネクションサービスはこれと同様の概念ですが、設定の方法に若干の違いがあります。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 ベースのサービスの設定

TCP/IP を使用して直接的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、3 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 Message Queue コマンドユーティリティ (`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。

**クラスタ (cluster):** このコネクションサービスは、クラスタ内の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メッセージの配信に使用されます ([150 ページの「ブローカ間の安全なコネクション」](#)を参照)。

### ➤ SSL ベースの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. `ssljms` コネクションサービス、`ssladmin` コネクションサービス、またはブローカ内のクラスタ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し実行します (ssljms コネクションサービスだけに適用)。
- ssljms および ssladmin コネクションサービスを設定する手順は、手順 4 のクライアントの設定と実行以外は同じです。
- 各手順については、次で詳しく説明します。

## 手順 1: 自己署名型証明書の生成

Message Queue の 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、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 コネクションサービス、ssladmin コネクションサービス、またはクラスタコネクションサービスに対して同じ証明書を使用できます。コマンドプロンプトで次のとおり入力します。

```
imqkeytool -broker
```

ユーティリティが、必要な情報を要求します。UNIX® システムでは、キーストアを作成するアクセス権を取得するためにスーパーユーザー (root) として imqkeytool を実行する必要があります。

imqkeytool は、まず、キーストアに対するパスワードの入力を要求します。次に一部の組織情報の入力、続いて確認を要求します。確認が取れると、キーの組み合わせを生成している間、このコマンドは停止します。その後、特定のキーの組み合わせをロックするためのパスワード (キーパスワード) の入力を要求してくるので、Return キーを押します。これで、キーパスワードに、キーストアと同じパスワードが設定されます。

---

<b>注</b>	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 (236 ページの「passfile の使用」を参照) に格納できます。
----------	--

---

imqkeytool を実行すると、JDK keytool ユーティリティが実行されて、自己署名型証明書が生成されます。生成された証明書は、付録 A 「Message Queue データの場所」に記載されているとおり、オペレーティングシステムに応じたディレクトリにある Message Queue のキーストアに配置されます。

キーストアは、JDK1.2 keytool ユーティリティでサポートされているのと同じフォーマットになっています。

Message Queue 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。各プロパティの設定に関する説明は、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は、キーストアファイルが配置されているディレクトリへのパスを指定する。 デフォルト値: <a href="#">付録 A 「Message Queue データの場所」</a> を参照
imq.keystore.file.name	SSL ベースのサービスの場合は、キーストアファイル名を指定する デフォルト値: keystore
imq.keystore.password	SSL ベースのサービスの場合は、キーストアのパスワードを指定する。このプロパティは passfile 内だけで指定可能 ( <a href="#">236 ページの「passfile の使用」</a> を参照)  さまざまな方法でパスワードを指定できる。もっとも安全な方法は、ブローカプロンプトでパスワードを入力することである。安全性は低くなるが、パスファイルを使用してパスファイルを読み取り保護することもできる。安全性はもっとも低くなるが、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こともできる。imqbrokerd -ldappassword

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
- ブローカの起動時に、SSL ベースのサービスが初期化に失敗し、次の例外を受け取った  
java.security.UnrecoverableKeyException:Cannot recover key.

この例外は、自己署名型証明書を [231 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. [付録 A 「Message Queue データの場所」](#) に示すとおり、ブローカのキーストアを削除します。
2. imqkeytool をもう 1 度実行し、[231 ページの「手順 1: 自己署名型証明書の生成」](#) の説明に従って、キーの組み合わせを生成します。

## 手順 2: ブローカでの SSL ベースのサービスを有効にする

ブローカでの SSL ベースのサービスを有効にするには、`ssljms` (または、`ssladmin`) を `imq.service.activelist` プロパティに追加する必要があります。

---

**注** `imq.service.activelist` プロパティではなく、`imq.cluster.transport` プロパティを使用して、SSL ベースの `cluster` コネクションサービスを有効にします。[150 ページの「ブローカ間の安全なコネクション」](#)を参照

---

### ► ブローカで SSL ベースのサービスを有効にするには

1. ブローカのインスタンス設定ファイルを開きます。

インスタンス設定ファイルは、その設定ファイルが関連付けられているブローカインスタンスの名前 (`instanceName`) によって識別されたディレクトリに書き込まれます ([付録 A「Message Queue データの場所」](#)を参照)。

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

2. 存在していない場合は、`imq.service.activelist` プロパティのエントリを追加し、SSL ベースのサービスをリストに含めます。

デフォルトでは、プロパティには `jms` コネクションサービスと `admin` コネクションサービスが含まれます。アクティブ化するサービスに応じて、`ssljms` コネクションサービスか `ssladmin` コネクションサービス、またはその両方を追加する必要があります。

```
imq.service.activelist=jms,admin,ssljms,ssladmin
```

## 手順 3: ブローカを起動する

キーストアパスワードを入力して、ブローカを起動します。パスワードの入力は、次のいずれかの方法で行います。

- ブローカの起動時にパスワードを要求するように許可する

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

- `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 ([236 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 ([136 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、[72 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、Message Queue が JSSE (Java Secure Socket Extension) を使用して SSL を実装するためです。JSSE は `java.security.SecureRandom()` を使用して、ランダムな数を生成します。このメソッドで初期ランダム番号シードを作成するにはかなりの時間がかかり、そのために CPU の使用率が増加します。シードが作成されたあと、CPU レベルは通常に戻ります。

## 手順 4: SSL ベースのクライアントを設定および実行する

最後に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` を使用するアプリケーションクライアントと、`ssladmin` を使用する `imqcmd` などの Message Queue 管理クライアントです。後続の節では、これらについて次で別個に説明します。

### アプリケーションクライアント

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`、`jcrt.jar`、`jndi.jar`

2. クライアントのクラスパスに次の Message Queue jar ファイルがあることを確認します。

`imq.jar`、`jms.jar`

3. クライアントを起動し、ブローカの `ssljms` サービスに接続します。これを行う 1 つの方法として、次のようなコマンドを入力します。

`java -DimqConnectionType=TLS clientAppName`



`imqConnectionType` を設定すると、コネクションに SSL を使用するよう指示が出されます。

クライアントアプリケーションでの `ssljms` コネクションサービスの使用についての詳細は、『[Message Queue Java Client Developer's Guide](#)』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。

### 管理クライアント (*imqcmd*)

`imqcmd` を使用するとき `-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 ([162 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、Message Queue ユーザーリポジトリ内の有効なエントリを指定します。単層型ファイルリポジトリを使用している場合は、[220 ページの「デフォルトの管理者パスワードの変更」](#)を参照してください。

コネクションサービスを一覧表示すると、次の出力のように、`ssladmin` サービスが実行中で、安全な管理コネクションが問題なく確立されたことが示されます。

```
Listing all the services on the broker specified by:
```

Host	Primary Port	
localhost	7676	
Service Name	Port Number	Service State
admin	33984 (dynamic)	RUNNING
httpjms	-	UNKNOWN
httpsjms	-	UNKNOWN
jms	33983 (dynamic)	RUNNING
ssladmin	35988 (dynamic)	RUNNING
ssljms	dynamic	UNKNOWN

```
Successfully listed services.
```

# 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、[319 ページの付録 C 「HTTP/HTTPS サポート \(Enterprise Edition\)」](#) 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。その後、ブローカの起動時に -passfile オプションを使用して、この *passfile* を指定できます。

```
imqbrokerd -passfile myPassfile
```

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ーを制限できます。ブローカを起動したユーザーに対して読み取りアクセスを許可するように、passfile にアクセス権を設定する必要があります。

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 ベースのサービスにキーストアパスワードを指定する ( <a href="#">232 ページの表 8-8</a> を参照)
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 ( <a href="#">222 ページの表 8-5</a> を参照)
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 ( <a href="#">312 ページの表 B-1</a> を参照)

passfile のサンプルは、[付録 A 「Message Queue データの場所」](#) に示すとおり、オペレーティングシステムに応じて該当する場所に格納されています。

# メッセージサービスの分析と調整

この章では、Message Queue サービスの分析と調整を行い、メッセージングアプリケーションのパフォーマンスを最適化する方法に関連するさまざまなトピックを取り上げます。次のトピックが含まれます。

- [パフォーマンス関連](#)
- [パフォーマンスに影響する要因](#)
- [メッセージサーバの監視](#)
- [パフォーマンス問題のトラブルシューティング](#)
- [パフォーマンスを改善するための設定の調整](#)

## パフォーマンス関連

### パフォーマンス調整プロセス

メッセージングアプリケーションのパフォーマンスは、アプリケーションと Message Queue サービスの相互関係に左右されます。そのため、パフォーマンスを最大化するには、アプリケーション開発者と管理者が協力し合う必要があります。

パフォーマンスを最適化するプロセスは、アプリケーションの設計から始まります。アプリケーションが配置された後も、継続してメッセージサービスの調整を行います。パフォーマンス調整プロセスには、次の段階があります。

- アプリケーションのパフォーマンス要件を定義する
- 特に信頼性とパフォーマンスの兼ね合いなど、パフォーマンスに影響する要因を考慮してアプリケーションを設計する
- パフォーマンスの基準を設ける

- パフォーマンスを最適化するためにメッセージサービスを調整または再設定する
- 通常は、上記に概略したプロセスを繰り返し実行します。アプリケーションの配置時に、**Message Queue** 管理者は、メッセージサーバの適合性を評価し、アプリケーションの全般的なパフォーマンス要件を満たしているかどうかを判断します。ベンチマークテストがこれらの要件を満たす場合は、この章で説明するとおり、管理者はシステムの調整段階に入ることができます。一方、ベンチマークテストがパフォーマンス要件を満たしていない場合は、アプリケーションの再設計や配置アーキテクチャの変更が必要となる場合があります。

## パフォーマンスのさまざまな側面

一般に、パフォーマンスの基準は、メッセージサービスがプロデューサからコンシューマへメッセージを配信するときの速度と効率です。ただし、パフォーマンスには、ニーズに応じて重要度が変わるさまざまな側面があります。

**コネクションの負荷**：メッセージプロデューサまたはメッセージコンシューマの数、もしくは、システムがサポート可能な同時コネクションの数

**メッセージのスループット**：メッセージングシステムが 1 秒間に扱えるメッセージ数、またはメッセージのバイト数

**遅延**：特定のメッセージがメッセージプロデューサからメッセージコンシューマへ配信されるまでに要する時間

**安定性**：メッセージサービス全体の可用性、つまり過負荷や障害による影響をどれだけ抑えられるか

**効率**：メッセージ配信の効率。使用するコンピュータリソースに関係する、メッセージスループットの評価

パフォーマンスのこれらの異なる評価基準は、一般に相互に関連しています。メッセージスループットが高い場合、メッセージがメッセージサーバへバックログされることは、ほとんどありません。その結果、遅延も短くなり、シングルメッセージはすぐに配信されます。ただし、遅延はさまざまな要因に左右されます。そのような要因の例としては、通信リンクの速度、メッセージサーバの処理速度、クライアントの処理速度などがあります。

どのような場合でも、パフォーマンスには複数の異なる側面があります。一般に、その中でどれがもっとも重要となるかは、特定のアプリケーションの要件によって決まります。

## ベンチマーク

ベンチマークとは、使用中のメッセージングアプリケーション用のテスト群を作成し、このテスト群を用いてメッセージスループットや、そのほかの観点からパフォーマンスを評価するプロセスです。

たとえば、複数のプロデュースングクライアントを対象に、複数の、コネクション、セッション、メッセージプロデューサを使用し、標準サイズの持続的または持続性のないメッセージを一部のキューやトピック（すべてメッセージングアプリケーションの設計に依存）へ一定レートで送信するテスト群を作成できます。同様に、複数の、コネクション、セッション、および特定タイプのメッセージコンシューマを使用し、特定の通知モードでメッセージをコンシュームする複数のコンシューミングクライアントをテスト群に含められます。

標準のテスト群を使用することで、メッセージがプロデュースされてからコンシュームされるまでに要する時間やメッセージの平均スループットレートを測定したり、システムを監視して、コネクションスレッド使用率、メッセージストレージデータ、メッセージフローデータ、そのほかの関連するメトリックスを監視したりできます。その後、パフォーマンスに悪影響が出る上限まで、メッセージのプロデュースレート、メッセージプロデューサの数、その他の変数を増加させることができます。実現可能な最大スループットが、メッセージサービス設定のベンチマークになります。

このベンチマークを基に、テスト群の特性の一部を変更できます。パフォーマンスに影響しそうな要因すべてを慎重に制御すれば（[242 ページの「パフォーマンスに影響するアプリケーション設計の要因」](#)を参照）、これらの要因の変化によるベンチマークへの影響を理解できます。たとえば、コネクション数またはメッセージ数を 5 倍もしくは 10 倍に増やし、パフォーマンスに与える影響を調べることができます。

逆に、アプリケーションベースの要因を一定に保ち、たとえば、コネクションプロパティ、スレッドプールプロパティ、JVM メモリ制限、制限の動作、組み込み持続とプラグイン持続などを変更するといった、制御方法でブローカ設定を変更して、これらの変更がパフォーマンスに及ぼす影響を判断することもできます。

アプリケーションのこのようなベンチマークから、メッセージサービスを調整して配置済みのアプリケーションのパフォーマンスを向上させたいときに有用な情報を得られます。ベンチマークによって、1 か所の変更や一連の変更による影響を正確に予測できます。

原則として、ベンチマークは、管理されたテスト環境で、メッセージサービスを安定させるため長期間実施する必要があります。Java コードをマシンコードに変換する JIT コンパイルによる起動時には、パフォーマンスに悪影響が及びます。

## 基準になる使用パターン

メッセージングアプリケーションが配置され稼働された後は、基準になる使用パターンを確立することが重要となります。要求のピークがいつ発生するか把握し、その要求の定量化を図ります。たとえば、通常、要求はエンドユーザー数、アクティビティレベル、時間帯、またはこれらすべてによって左右されます。

基準になる使用パターンを確立するには、十分長い期間メッセージサーバを監視し、接続数、ブローカまたは特定の送信先に格納されたメッセージ数、ブローカまたは特定の送信先との間のメッセージフロー、アクティブなコンシューマの数などを確認する必要があります。また、メトリクスデータにより提供される平均値とピーク値を使用することもできます。

これらの基準になるメトリックスを設計時の期待値と比較することが重要です。それによって、クライアントコードが正常に動作していることを確認します。たとえば、接続が開いたままになっている、または、コンシュームされたメッセージが未通知のままになっているといった状態を確認できます。これらのコーディングエラーはメッセージサーバのリソースを消費し、パフォーマンスに大きな影響を及ぼします。

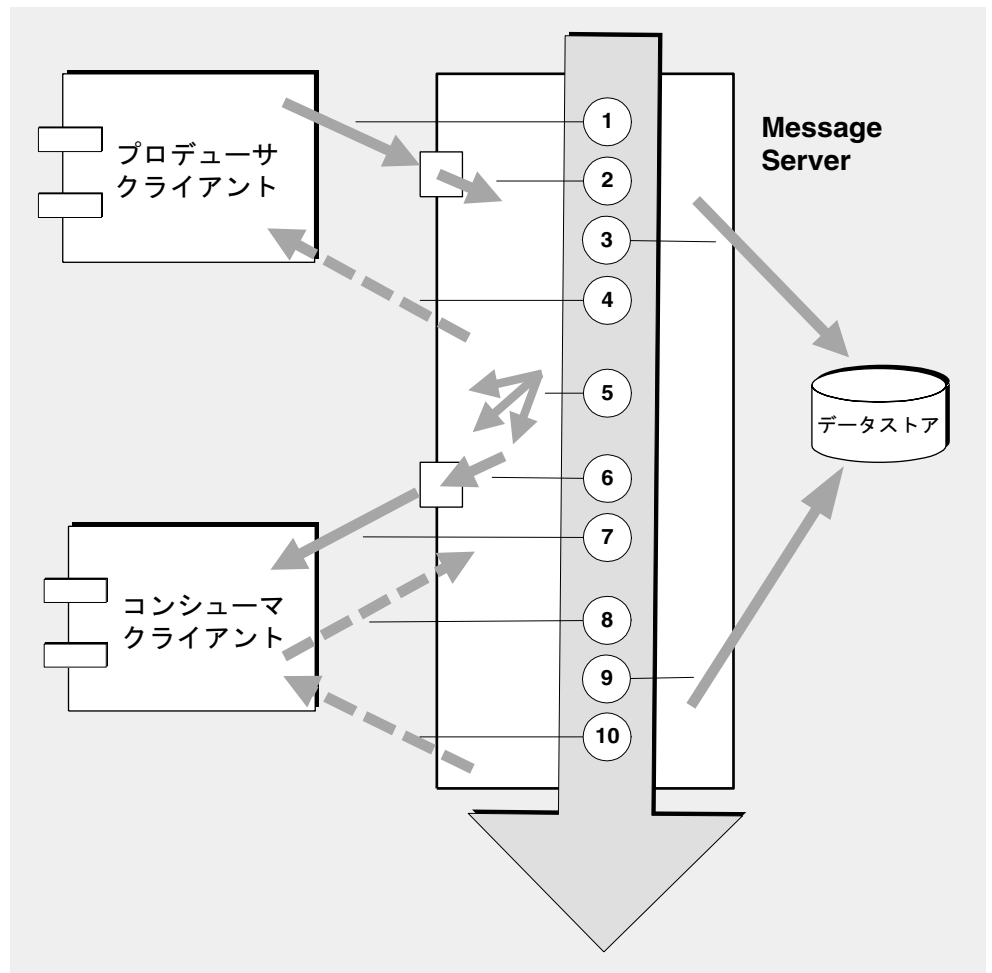
基準になる使用パターンは、最適なパフォーマンスを得るためにシステムを調整する方法を決定する上で役立ちます。たとえば、1つの送信先がそのほかの送信先に比べ頻繁に使用されている場合は、その送信先のメッセージメモリー制限をそのほかの送信先より高く設定したり、使用率に応じて制限の動作を調整したりできます。必要な接続数が最大スレッドプールサイズによる許容値を大きく上回る場合は、スレッドプールサイズを増やすか、共有スレッドモデルを採用することができます。ピーク時のメッセージフローが平均フローより多い場合は、メモリーが不足したときに使用する制限の動作に影響することがあります。

一般に、使用パターンをより綿密に理解しているほど、より適切に、使用パターンに応じてシステムを調整し、将来ニーズに合わせてプランニングすることができます。

## パフォーマンスに影響する要因

メッセージの遅延とメッセージのスループットは、2つの主要なパフォーマンスの評価基準です。これらは一般に、標準的なメッセージがメッセージ配信プロセスの各手順を完了するまでに要する時間に依存します。メッセージを持続的で信頼できる方法で配信する場合の各手順は次のとおりです。各手順を以下で図示します。

図 9-1 Message Queue サービスを使用したメッセージの配信



1. メッセージはプロデューシングクライアントからメッセージサーバへ配信される
2. メッセージサーバはメッセージの内容を読み取る

3. メッセージは、信頼性を維持するために持続ストレージに配置される
4. メッセージサーバは、信頼性を維持するためにメッセージの受信確認を発行する
5. メッセージサーバは、メッセージのルーティングを決定する
6. メッセージサーバはメッセージを書き込む
7. メッセージはメッセージサーバからコンシューミングクライアントへ配信される
8. コンシューミングクライアントは、信頼性を維持するためにメッセージの受信確認を発行する
9. メッセージサーバは、信頼性を維持するために、クライアントの通知を処理する
10. メッセージサーバは、クライアントの通知が処理されたことを通知する

これらの手順は順次実行されるため、プロデュースングクライアントからコンシューミングクライアントへメッセージを配信する際には、どの手順もボトルネックとなる恐れがあります。これらの手順の大半は、メッセージングシステムの物理的な特性に依存しています。物理的な特性には、ネットワーク帯域幅、コンピュータの処理速度、メッセージサーバのアーキテクチャなどが含まれます。ただし、一部の手順は、メッセージングアプリケーションの特性と必要とされる信頼性のレベルにも依存しています。

次の節では、アプリケーション設計の要因とメッセージングシステムの要因の両方がパフォーマンスに及ぼす影響について説明します。アプリケーション設計の要因とメッセージングシステムの要因はメッセージの配信に密接に関係しますが、各カテゴリは個別に考慮します。

## パフォーマンスに影響するアプリケーション設計の要因

アプリケーション設計の決定は、メッセージングのパフォーマンス全体に大きく影響することがあります。

パフォーマンスに影響するもっとも重要な要因は、メッセージ配信の信頼性に影響を及ぼす要因です。次のような要因が含まれています。

- 配信モード ( 持続的 / 持続性のないメッセージ )
- トランザクションの使用
- 通知モード
- 永続的サブスクリプションと永続的でないサブスクリプション

そのほかに、パフォーマンスに影響するアプリケーション設計の要因には、次のものがあります。

- セレクタの使用 ( メッセージのフィルタリング )



- [メッセージのサイズ](#)
- [メッセージ本体のタイプ](#)

以降の節では、これらの各要因がメッセージングパフォーマンスに及ぼす影響について説明します。原則として、パフォーマンスと信頼性は相反しています。つまり、信頼性が高くなるとパフォーマンスは低下します。

次の表は、さまざまなアプリケーション設計の要因が一般にどのようにメッセージングパフォーマンスに影響するかを示しています。表には、信頼性が高くパフォーマンスが低いシナリオと、パフォーマンスが高く信頼性の低いシナリオの2つのシナリオと、それぞれを特徴付ける主要なアプリケーション設計の要因を示します。これらの極端なシナリオの間には、信頼性とパフォーマンスの両方に影響する、多数の選択肢と兼ね合いがあります。

**表 9-1**      高信頼性シナリオと高パフォーマンスシナリオの比較

アプリケーション設計要因	高信頼性 低パフォーマンスシナリオ	高パフォーマンス 低信頼性シナリオ
配信モード	持続性メッセージ	持続性のないメッセージ
トランザクションの使用	処理済みセッション	トランザクションなし
通知モード	AUTO_ACKNOWLEDGE または CLIENT_ACKNOWLEDGE	DUPS_OK_ACKNOWLEDGE
永続的 / 永続的でないサブスクリプション	永続的サブスクリプション	永続的でないサブスクリプション
セレクトタの使用	メッセージのフィルタリング	メッセージのフィルタリングなし
メッセージのサイズ	サイズの小さいメッセージ	サイズの大きいメッセージ
メッセージ本体のタイプ	複合本体タイプ	単純本体タイプ

## 注

以下のグラフのパフォーマンスデータは、2 基の CPU を搭載した 1002 MHz の Solaris 8 システムでファイルベースの持続を使用して生成されたものです。パフォーマンステストでは、JIT コンパイラにより、システムを最適化し、持続データベースの準備をするために、最初に Message Queue ブローカを起動しました。

ブローカを起動した後、1 つのプロデューサと 1 つのコンシューマが作成され、メッセージが 30 秒間プロデュースされました。コンシューマがすべてのプロデュースされたメッセージを受信するために要した時間が記録され、スループットレートつまり 1 秒あたりのメッセージ数が計算されました。このシナリオは、表 9-1 に示すアプリケーション設計の要因の異なる組み合わせで繰り返し実行されました。

## 配信モード ( 持続的 / 持続性のないメッセージ )

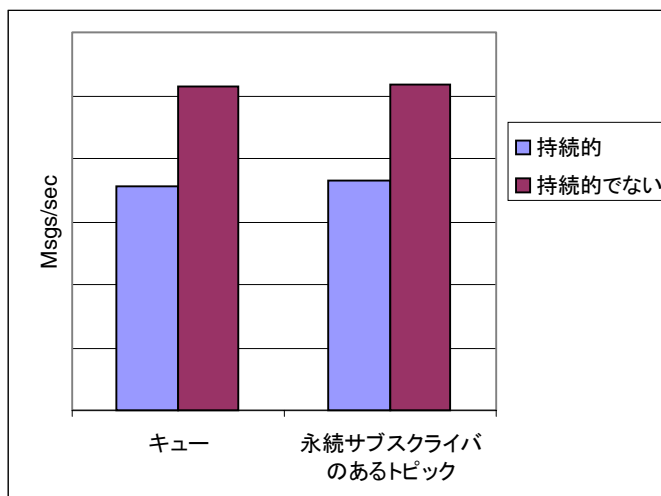
48 ページの「信頼性のあるメッセージング」で説明したとおり、持続性メッセージはメッセージサーバの障害時にもメッセージの配信を保証します。すべての対象のコンシューマが、メッセージをコンシュームしたことを通知するまで、ブローカはメッセージを持続ストアに格納します。

持続性メッセージのブローカの処理速度は、次の理由から、持続性のないメッセージの場合より低速です。

- ブローカに障害が生じても持続性メッセージが失われないように、ブローカは信頼できる方法で持続性メッセージを格納する必要がある
- ブローカは受信した持続性メッセージごとに、受信確認をする必要がある。メッセージをプロデュースするメソッドが例外を返さなければ、ブローカへの配信は保証される
- クライアントの通知モードによっては、クライアントからの持続性メッセージの受信通知がコンシュームされたことを、ブローカが確認しなければならない場合がある

持続モードと非持続モードではパフォーマンスに大きな差が生じることがあります。図 9-2 は、信頼できる配信を行う 2 つのケースで、持続性メッセージと持続性のないメッセージのスループットを比較したものです。2 つのケースとは、永続的サブスクリプションを使用して 10K バイトのメッセージをキューに配信した場合とトピックに配信した場合です。どちらの場合も AUTO\_ACKNOWLEDGE 通知モードを使用しています。

図 9-2 配信モードによるパフォーマンスへの影響



## トランザクションの使用

トランザクションとは、処理済みセッションでプロデュースされたすべてのメッセージと処理済みセッションでコンSUMEされたすべてのメッセージが、一体として処理されるか、または一体として処理されない、つまりロールバックされることを保証するものです。

Message Queue は、ローカルトランザクションと分散トランザクションの両方をサポートします (詳細は、それぞれ「[ローカルトランザクション](#)」と [50 ページの「分散トランザクション](#)」を参照)。

処理済みセッションでのメッセージのプロデュースまたは通知の処理速度は、次の理由から、処理済みでないセッションの場合より低速です。

- プロデュースされたメッセージごとに追加情報を格納する必要がある
- 状況によっては、通常は格納されないトランザクション内のメッセージを格納する場合がある。たとえば、サブスクリプションを使用しないトピック送信先へ配信された持続性メッセージは、通常削除されるが、トランザクションの開始時にサブスクリプションに関する情報が使用できなくなってしまう
- トランザクションでのメッセージのコンSUMEと通知に関する情報は、トランザクションがコミットされた時点で格納し処理する必要がある

## 通知モード

JMS メッセージの配信の信頼性を保証する手段の 1 つは、Message Queue メッセージサーバによってクライアントへ配信されたメッセージのコンシュームをクライアントに通知するという方法です (62 ページの「[信頼性の高い配信 : 通知およびトランザクション](#)」を参照)。

クライアントがメッセージを通知することなくセッションが閉じられた場合や、通知が処理される前にメッセージサーバに障害が生じた場合には、ブローカはメッセージを再配信して JMSRedelivered フラグをセットします。

処理済みでないセッションの場合、クライアントは、それぞれ固有のパフォーマンス特性をもつ 3 つの通知モードの中から 1 つを選択できます。

- **AUTO\_ACKNOWLEDGE** コンシューマがメッセージを処理した後、システムは自動的にメッセージを通知する。このモードでは、プロバイダで障害が生じた後は、多くても 1 つのメッセージの再配信を保証するだけである
- **CLIENT\_ACKNOWLEDGE** アプリケーションは、メッセージが通知されるポイントを制御する。直前の通知以降にそのセッションで処理されたすべてのメッセージが通知される。一連の通知の処理中にメッセージサーバに障害が生じた場合は、そのグループ内の複数のメッセージが再配信されることがある
- **DUPS\_OK\_ACKNOWLEDGE** このモードは、時間をかけてメッセージを通知するようにシステムに指示する。プロバイダに障害が生じた後でも、複数のメッセージを再配信できる。

**CLIENT\_ACKNOWLEDGE** モードの使い方はトランザクションの使い方に似ています。ただし、処理中にプロバイダに障害が生じた場合に、すべての通知が一括して処理されることを保証していない点を除きます。

次の理由から、パフォーマンスは通知モードによる影響を受けます。

- **AUTO\_ACKNOWLEDGE** モードと **CLIENT\_ACKNOWLEDGE** モードでは、ブローカとクライアント間で特別な制御メッセージが必要である。追加の制御メッセージは、処理オーバーヘッドを高め、JMS ペイロードメッセージに干渉して処理遅延を引き起こすことがある
- **AUTO\_ACKNOWLEDGE** モードと **CLIENT\_ACKNOWLEDGE** モードでは、ブローカがクライアントの通知を処理したことを確認するまで、クライアントは待機する必要がある。その後、クライアントは追加メッセージをコンシュームできるようになる。このブローカの確認によって、何らかの理由でブローカがこれらのメッセージを再配信しないように保証する
- **Message Queue** 持続ストアは、コンシューマが受信したすべての持続性メッセージに関する通知情報を使って更新する必要がある。そのため、パフォーマンスは低下する

## 永続的サブスクリプションと永続的でないサブスクリプション

トピック送信先へのサブスクライバは、[46 ページの「パブリッシュ / サブスクライブ \(トピック送信先\)」](#)で説明したとおり、永続的サブスクリプションをもつものと、永続的でないサブスクリプションをもつものの2つのカテゴリに分かれます。

永続的サブスクリプションは、次の理由から、スループットを低下させる代わりに信頼性を向上させます。

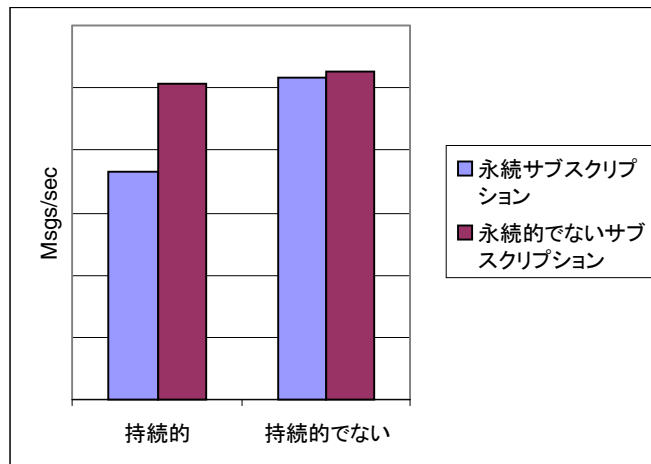
- **Message Queue** メッセージサーバは、メッセージサーバに障害が生じた場合でも回復後にリストを使用できるように、各永続的サブスクリプションに割り当てられたメッセージのリストを持続的に格納する必要がある
- メッセージサーバに障害が生じた場合でも、回復後に、対応するコンシューマがアクティブになったときに、メッセージを引き続き配信できるように、永続的サブスクリプションの持続性メッセージは持続ストアに格納される。対照的に、永続的でないサブスクリプションの持続性メッセージは持続ストアには格納されない。したがって、メッセージサーバに障害が生じると、対応するコンシューマコネクションは失われ、メッセージは配信されない

[図 9-3](#) は、10K バイトの、持続性メッセージと持続性のないメッセージの2通りのケースで、永続的サブスクリプションと永続的でないサブスクリプションを使用したトピック送信先のスループットを比較したものです。どちらの場合も

AUTO\_ACKNOWLEDGE 通知モードを使用しています。

[図 9-3](#) から、パフォーマンスへの影響が顕著となるのは、永続的サブスクリプションを使用した持続性メッセージの場合だけであることがわかります。上記のとおり、この場合の影響は、永続的サブスクリプションを使用したときに持続性メッセージだけが持続ストアに格納されることに起因しています。

**図 9-3** サブスクリプションタイプによるパフォーマンスへの影響



## セレクトタの使用 (メッセージのフィルタリング)

アプリケーション開発者は、通常、特定のコンシューマへの一連のメッセージを対象にしています。それは、一意の送信先への一連のメッセージごとを対象とするか、単一の送信先を使用しコンシューマごとに複数のセレクトタを登録することで実現できます。

セレクトタとは文字列で、この文字列に一致するプロパティ値 (40 ページの「JMS メッセージ構造」を参照) を持ったメッセージだけを特定のコンシューマに配信します。たとえば、セレクトタ `NumberOfOrders >1` は、`NumberOfOrders` プロパティ値が 2 以上のメッセージだけを配信します。

コンシューマにセレクトタを登録すると、各メッセージを取り扱うために追加処理が必要となり、複数の送信先を使用する場合に比べ、パフォーマンスは低下します。以降のメッセージを比較する際にも構文解析できるセレクトタを使用する必要があります。さらに、各メッセージがルートされるたびに、各メッセージのメッセージプロパティを読み取り、比較する必要があります。ただし、セレクトタを使用すると、メッセージングアプリケーションの柔軟性が向上します。

## メッセージのサイズ

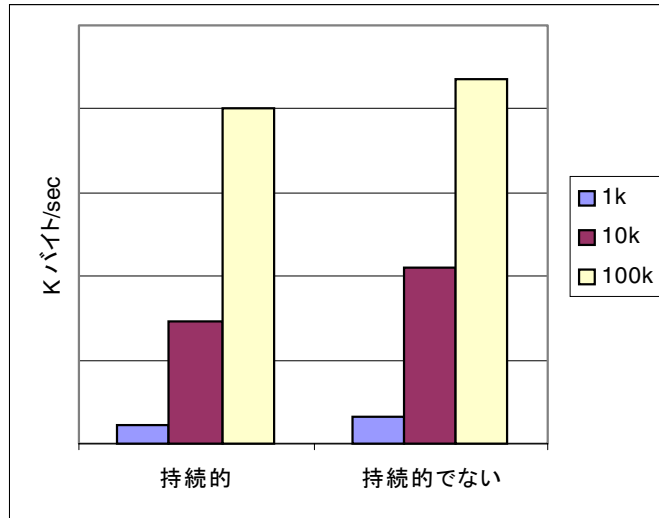
メッセージのサイズはパフォーマンスに影響します。プロデュースングクライアントからブローカへ、さらにブローカからコンシューミングクライアントへは、より多くのデータを渡す必要があり、持続性メッセージの場合はサイズの大きいメッセージを格納する必要があるからです。

ただし、複数のサイズの小さいメッセージを 1 つのメッセージにまとめることで、個々のメッセージの転送と処理を最小限に抑え、パフォーマンス全体を向上させることができます。この場合、個々のメッセージの状態に関する情報は失われてしまいます。

図 9-4 は、持続性メッセージと持続性のないメッセージの 2 とおりのケースで、1K、10K、および 100K バイトのメッセージのスループットを 1 秒あたりの K バイト数で比較したものです。どのケースも、メッセージはキュー送信先へ送信し、`AUTO_ACKNOWLEDGE` 通知モードを使用しています。

図 9-4 は、両方のケースで、小さいサイズのメッセージの場合に比べ、よりサイズの大きいメッセージを配信するほどオーバーヘッドが低くなることを示しています。また、1K バイトと 10K バイトのメッセージの場合は、持続性メッセージと持続性のないメッセージを比べたパフォーマンスの上昇は約 50% ですが、100K バイトのメッセージの場合、この差が維持されないことがわかります。おそらくこれは、100K バイトの場合には、ネットワークの帯域幅がメッセージスループットのボトルネックになるからでしょう。

図 9-4 メッセージのサイズによるパフォーマンスへの影響



## メッセージ本体のタイプ

JMS がサポートするメッセージ本体のタイプ 5 種類の概要を複雑な順に次に示します。

- **BytesMessage**: 一連のバイトデータを含み、形式はアプリケーションによって決まる
- **TextMessage**: 単純な `java.lang.String`
- **StreamMessage**: Java プリミティブ値によるストリームを含む
- **MapMessage**: 一連の名前と値のペアが含まれる
- **ObjectMessage**: Java のシリアライズされたオブジェクトが含まれる

一般に、メッセージのタイプはアプリケーションのニーズによって決定され、**MapMessage** や **ObjectMessage** などのより複雑なタイプほどパフォーマンスは低下します。データのシリアライズとデシリアライズがパフォーマンスを低下させます。パフォーマンスは、データがどの程度単純か、またはどの程度複雑かによって異なります。

## パフォーマンスに影響するメッセージサービスの要因

メッセージングアプリケーションのパフォーマンスは、アプリケーション設計だけでなく、メッセージのルーティングと配信を実行するメッセージサービスによっても影響を受けます。

次の節では、パフォーマンスに影響することのあるさまざまなメッセージサービスの要因について説明します。これらの要因の影響を理解しておくことは、メッセージサービスの内容を変更したり、配置済みのアプリケーションで発生することのあるパフォーマンスボトルネックを診断し解決したりする上で重要となります。

Message Queue サービスのパフォーマンスに影響する最も重要な要因は、次のとおりです。

- [ハードウェア](#)
- [オペレーティングシステム](#)
- [Java 仮想マシン \(JVM\)](#)
- [コネクション](#)
- [ブローカの制限と動作](#)
- [メッセージサービスのアーキテクチャ](#)
- [データストアのパフォーマンス](#)
- [クライアントランタイムの設定](#)

以降の節では、これらの各要因がメッセージングパフォーマンスに及ぼす影響について説明します。

### ハードウェア

Message Queue メッセージサーバとクライアントアプリケーションのどちらの場合も、CPU の処理速度と使用可能なメモリーはメッセージサービスのパフォーマンスを決定する主要な要因となります。処理性能を強化して、多数のソフトウェア制限をなくす一方で、メモリーを追加して処理速度と能力を増加させることができます。ただし、一般に、単にハードウェアをアップグレードするだけでボトルネックを解消すると多額の費用がかかります。



## オペレーティングシステム

同じハードウェアプラットフォームを前提とした場合でも、異なるオペレーティングシステムの効率によって、パフォーマンスも変わってきます。たとえば、オペレーティングシステムが採用しているスレッドモデルが、メッセージサーバがサポート可能な同時接続数に大きく影響することがあります。一般に、すべてのハードウェアが同じであれば、Solaris は通常 Linux より高速で、Linux は Windows より高速です。

## Java 仮想マシン (JVM)

メッセージサーバは、ホスト JVM 内で実行され、ホスト JVM によってサポートされる Java プロセスです。そのため、JVM 処理は、メッセージサーバがメッセージをいかに早く効率良く転送し配信できるかを決定する重要な要因となります。

特に、JVM のメモリーリソースの管理が不可欠となる場合があります。増加し続けるメモリーの負荷に対応するには、JVM に十分なメモリーを割り当てる必要があります。さらに、JVM は定期的に未使用のメモリーを再利用します。このメモリー再利用がメッセージの処理を遅らせることがあります。JVM のメモリーヒープが大きくなるほど、メモリー再利用時に経験することのある潜在的遅延も長くなります。

## コネクション

クライアントとブローカ間のコネクションの数と速度は、メッセージサーバが処理可能なメッセージ数とメッセージ配信速度に影響することがあります。

### メッセージサーバのコネクションの制限

メッセージサーバへのアクセスはすべて、コネクション経由で行われます。同時コネクション数の制限によって、メッセージサーバが同時に使用できるプロデュースングクライアントまたはコンシューミングクライアントの数が左右されることがあります。

メッセージサーバへのコネクションの数は、一般に、使用可能なスレッド数によって制限されます。Message Queue は、スレッドプールマネージャを使用します。このマネージャは、専用スレッドモデルまたは共有スレッドモデルのどちらかを使用するように設定できます (58 ページの「スレッドプールマネージャ」を参照)。専用スレッドモデルは、各コネクションが専用のスレッドを持つため非常に高速ですが、コネクションの数は使用可能なスレッド数によって制限されます。この場合、コネクションごとに、入力スレッドと出力スレッドが 1 つずつ必要です。共有スレッドモデルには、コネクション数の制限はありませんが、多数のコネクションでスレッドを共有するため、オーバーヘッドが増え、スループットが悪化します。これは、多くのコネクションが使用中のとき特に顕著になります。

## トランスポートプロトコル

Message Queue ソフトウェアを使うと、クライアントは各種の低レベルのトランスポートプロトコルを使用してメッセージサーバと通信できます。Message Queue は、[57 ページの「コネクションサービスのサポート」](#)に示すコネクションサービスとそれに対応するプロトコルをサポートします。暗号化、ファイアウォールを介したアクセスなどのプロトコルは、アプリケーション要件に基づいて選択されますが、選択結果はパフォーマンス全体に影響を及ぼします。

図 9-5 トランスポートプロトコルの速度

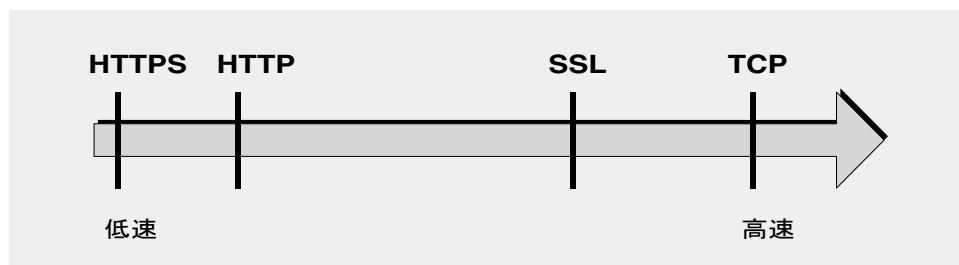


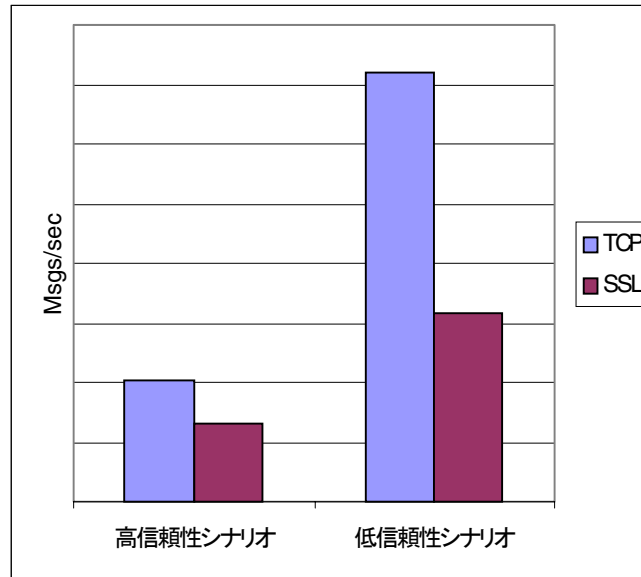
図 9-5 は、さまざまなプロトコルテクノロジーのパフォーマンス特性を示しています。

- TCP は、ブローカと通信する最速の方法を提供する
- SSL は、メッセージの送信と受信に関しては、TCP より 50 ～ 70% 低速である。持続性メッセージの場合は 50%、持続性のないメッセージの場合は約 70%。さらに、初期接続の確立は、SSL を使用した場合の方が低速で数秒かかる。これは、クライアントとブローカ、または HTTPS の場合は Web Server で、送信するデータの暗号化に使う非公開キーの作成が必要なためである。低レベルの各 TCP パケットの暗号化と復号化に必要な追加処理によって、パフォーマンスの低下が引き起こされる。

図 9-6 は、2 つのケースでの TCP と SSL のスループットを比較したものである。2 つのケースとは、1K バイトの持続性メッセージを、永続的サブスクリプションと AUTO\_ACKNOWLEDGE 通知モードを使用しているトピック送信先へ送信する高信頼性シナリオと、1K バイトの持続性のないメッセージを、永続的サブスクリプションと DUPS\_OK\_ACKNOWLEDGE 通知モードを使用しているトピック送信先へ送信するハイパフォーマンスシナリオである。

図 9-6 は、高信頼性ケースの方がプロトコルによる影響は少ないことを示す。これは、高信頼性のケースに必要な持続性メッセージのためのオーバーヘッドの方が、プロトコルの速度より、スループットを制限する重要な要因となるからである。

図 9-6 トランSPORTプロトコルによるパフォーマンスへの影響



- HTTP は、TCP または SSL より低速である。HTTP ではクライアントとブローカ間のプロキシとして、Web サーバ上で実行しているサーブレットを使用する。パケットを HTTP 要求へカプセル化する必要がある点と、メッセージがブローカへ到達するには、クライアントからサーブレットへ、サーブレットからブローカへという 2 つの段階が必要である点から、パフォーマンスへのオーバーヘッドが発生する
- HTTPS は HTTP より低速である。これは、クライアントとサーブレット間、およびサーブレットとブローカ間でパケットを暗号化するためにオーバーヘッドが必須となるからである

## メッセージサービスのアーキテクチャ

Message Queue メッセージサーバは、シングルフローカ、または複数の連結されたブローカインスタンスであるブローカクラスタとして実装できます。

ブローカに接続するクライアントの数や配信されるメッセージの数が増えると、ブローカは最終的に、ファイル記述子、スレッド、メモリーの制限などのリソースの限界を超えてしまいます。増え続ける負荷に対処するための 1 つの方法は、Message Queue メッセージサーバにブローカインスタンスを追加して、クライアントのコネクションとメッセージのルーティングおよび配信を複数のブローカに分散することです。

一般に、ブローカインスタンスの追加は、クライアント、特にメッセージプロデュースングクライアントがクラスタ間で均等に分散されている場合に最適に動作します。クラスタ内のブローカ間でのメッセージ配信にはオーバーヘッドが伴うため、コネクション数とメッセージ配信レートが制限されているクラスタでは、シングルブローカよりパフォーマンスが低くなります。

また、ブローカクラスタを使用してネットワークの帯域幅を最適化することもできます。たとえば、クラスタ内の一連のリモートブローカ間で、速度の遅い、長距離のネットワークリンクを使用する一方で、個々のブローカインスタンスへのクライアントの接続に、高速なリンクを使用することができます。

クラスタの詳細については、[86 ページの「マルチブローカクラスタ \(Enterprise Edition\)」](#)と [147 ページの「クラスタの操作 \(Enterprise Edition\)」](#)を参照してください。

## ブローカの制限と動作

メッセージを処理するためにメッセージサーバに要求されるメッセージスループットは、メッセージサーバがサポートするメッセージングアプリケーションの使用パターンによって異なります。ただし、メッセージサーバでは、メモリー、CPU サイクルなどのリソースに制限があります。リソースの制限により、メッセージサーバは、過負荷となり、無応答または不安定となるポイントに達してしまことがあります。

**Message Queue** メッセージサーバには、メモリーリソースを管理し、ブローカのメモリー不足を防ぐためのメカニズムが組み込まれています。これらのメカニズムに含まれるのは、ブローカまたは個々の送信先が保持できるメッセージ数またはメッセージのバイト数についての設定可能な制限と、送信先の制限に達したときに起動できる一連の動作です ([63 ページの「メモリーリソースとメッセージフローの管理」](#)を参照)。

これらの設定可能なメカニズムを使用して、システムが過負荷にならないように、メッセージの受信と送信のバランスを取ることができますが、これには注意深い監視と調整が必要です。これらのメカニズムは、オーバーヘッドを増加させ、メッセージのスループットを制限することがありますが、それでも動作の完全性を維持します。

## データストアのパフォーマンス

**Message Queue** は、組み込み持続とプラグイン持続の両方をサポートしています ([66 ページの「持続マネージャ」](#)を参照)。組み込み持続は、ファイルベースのデータストアです。プラグイン持続は、JDBC (Java Database Connectivity) インタフェースを使用し、JDBC 互換のデータストアを必要とします。

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

組み込み持続の場合は、持続的な操作によりメモリー内の状態とデータストアとを同期化するように指定することで、信頼性を高められます。この同期化は、システム破壊によるデータの損失をなくす上で役立ちますが、パフォーマンスが犠牲になります。

## クライアントランタイムの設定

Message Queue クライアントランタイムは、クライアントアプリケーションに Message Queue メッセージサービスへのインタフェースを提供します。クライアントランタイムでは、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場合に、クライアントに必要なすべての処理をサポートします。クライアントランタイムは、コネクションファクトリ属性値を使って設定可能で、一般的にパフォーマンスとメッセージスループットを向上させるようにプロパティと動作を設定できます。

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

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

これらの動作とそれを設定するために使用される属性の詳細は、[299 ページの「クライアントランタイムのメッセージフローの調整」](#)を参照してください。

# メッセージサーバの監視

Message Queue サーバは、パフォーマンスの監視に使用可能なメトリックス情報を提供するように設定できます。この節では、メッセージサーバの監視に使用可能なさまざまなツールと、これらのツールを使用して取得できるメトリックスデータについて説明します。

メトリックスデータを使用して、パフォーマンス問題を解決する方法やメッセージサーバのパフォーマンスを分析し調整する方法については、[276 ページの「パフォーマンス問題のトラブルシューティング」](#)を参照してください。

## 監視ツール

次のツールを使用してメトリックス情報を取得できます。

- [Message Queue コマンドユーティリティ \(imqcmd\)](#)
- [Message Queue ブローカログファイル](#)
- [メッセージベースの監視 API](#)

次の節では、これらの各ツールを使用してメトリックス情報を取得する方法を説明します。さまざまなツールの比較については、[265 ページの「適切な監視ツールの選択」](#)を参照してください。

### Message Queue コマンドユーティリティ (imqcmd)

コマンドユーティリティ (imqcmd) は、Message Queue の基本的なコマンド行管理ツールです。このツールを使用して、ブローカとブローカのコネクションサービス、およびアプリケーション固有のリソースである物理的な送信先、永続的サブスクリプション、トランザクションなどを管理できます。imqcmd コマンドについては、[第6章「ブローカとアプリケーションの管理」](#)で説明されています。

imqcmd コマンドの機能の1つは、ブローカ全体、個々のコネクションサービス、個々の送信先に関するメトリックス情報を取得できる機能です。メトリックスデータを取得するには、一般に、imqcmd の `metrics` サブコマンドを使用します。メトリックスデータは、指定した間隔で、または指定した回数だけ、コンソール画面に表示されます。

また、`query` サブコマンド ([260 ページの「imqcmd query」](#)を参照) を使用して、より限定されたメトリックスデータのサブセットを取得することもできます。

#### *imqcmd metrics*

imqcmd metrics の構文とオプションを、それぞれ[表 9-2](#)と[表 9-3](#)に示します。

表 9-2 imqcmd metrics サブコマンドの構文

サブコマンドの構文	提供されるメトリックスデータ
<pre>metrics bkr   [-b <i>hostName:port</i>]   [-m <i>metricType</i>]   [-int <i>interval</i>]   [-msp <i>numSamples</i>]   [-u <i>userName</i>]   [-p <i>password</i>]</pre>	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
または	
<pre>metrics svc -n <i>serviceName</i>   [-b <i>hostName:port</i>]   [-m <i>metricType</i>]   [-int <i>interval</i>]   [-msp <i>numSamples</i>]   [-u <i>userName</i>]   [-p <i>password</i>]</pre>	デフォルトのブローカ、または指定したホストとポートのブローカで実行している特定のサービスのメトリックスを表示する
または	
<pre>metrics dst -t <i>destType</i>   -n <i>destName</i>   [-b <i>hostName:port</i>]   [-m <i>metricType</i>]   [-int <i>interval</i>]   [-msp <i>numSamples</i>]   [-u <i>userName</i>]   [-p <i>password</i>]</pre>	特定のタイプと名前の送信先に関するメトリックス情報を表示する

表 9-3 imqcmd metrics サブコマンドのオプション

サブコマンドのオプション	説明
<code>-b <i>hostName:port</i></code>	メトリックスデータを報告するホスト名とブローカのポートを指定する デフォルト値は <code>localhost:7676</code>
<code>-int <i>interval</i></code>	メトリックスが表示される間隔を、秒単位で指定する デフォルトは 5 秒

表 9-3          imqcmd metrics サブコマンドのオプション ( 続き )

サブコマンドのオプション	説明
-m <i>metricType</i>	表示するメトリックスのタイプを指定する  <b>ttl</b> ブローカとの間のメッセージとパケットのフローに関するメトリックスを表示する ( デフォルトのメトリックスタイプ )  <b>rts</b> ブローカとの間のメッセージとパケットの 1 秒あたりのフローの量に関するメトリックスを表示する  <b>cxn</b> コネクション、仮想メモリーヒープ、およびスレッドを表示する ( ブローカとコネクションサービスだけ )  <b>con</b> コンシューマ関連のメトリックスを表示する ( 送信先だけ )  <b>dsk</b> ディスク使用量のメトリックスを表示する ( 送信先だけ )
-msp <i>numSamples</i>	出力に表示するサンプルの数を指定する。デフォルトは無制限 ( 無限 )
-n <i>destName</i>	必要に応じて、メトリックスデータを報告する送信先の送信先名を指定する。デフォルトはない
-n <i>serviceName</i>	必要に応じて、メトリックスデータを報告するコネクションサービスを指定する。デフォルトはない
-t <i>destTyp</i>	必要に応じて、メトリックスデータを報告する送信先のタイプ ( キューまたはトピック ) を指定する。デフォルトはない
-u <i>userName</i>	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
-p <i>password</i>	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

**手順 : metrics サブコマンドを使用したメトリックスデータの表示**

この節では、metrics サブコマンドを使用してメトリックス情報を報告するための手順を説明します。



## ► metrics サブコマンドを使用するには

1. メトリックス情報が必要なブローカを起動します。  
141 ページの「ブローカの起動」を参照してください。
2. 表 9-2 と表 9-3 に示すオプションを指定して、適切な `imqcmd metrics` サブコマンドを実行します。

### メトリックスの出力: `imqcmd metrics`

この節では、ブローカ全体、コネクションサービス、送信先のメトリックスに関する `metrics` サブコマンドの出力例を示します。

**ブローカ全体のメトリックス:** ブローカとの間のメッセージとパケットのフローレートを 10 秒間隔で取得するには、`metrics bkr` サブコマンドを使用します。

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

このコマンドは、次のような出力を生成します (268 ページの表 9-8 のデータの説明を参照)。

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

**コネクションサービスのメトリックス:** `jms` コネクションサービスが処理したメッセージとパケットの累計を取得するには、`metrics svc` サブコマンドを使用します。

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

このコマンドは、次のような出力を生成します (270 ページの表 9-9 のデータの説明を参照)。

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

**送信先メトリックス** : 送信先に関するメトリックス情報を表示するには、`metrics dst` サブコマンドを使用します。

```
imqcmd metrics dst -t q -n XQueue -m ttl -u admin -p admin
```

このコマンドは、次のような出力を生成します (273 ページの表 9-10 のデータの説明を参照)。

Msgs		Msg Bytes		Msg Count			Total Msg Bytes			
(k)		Largest								
In	Out	In	Out	Current	Peak	Avg	Current	Peak	Avg	Msg (k)
200	200	147200	147200	0	200	0	0	143	71	0
300	200	220800	147200	100	200	10	71	143	64	0
300	300	220800	220800	0	200	0	0	143	59	0

送信先のコンシューマに関する情報を取得するには、`metrics dst` サブコマンドを使用します。

```
imqcmd metrics dst -t q -n SimpleQueue -m con -u admin -p admin
```

このコマンドは、次のような出力を生成します (273 ページの表 9-10 のデータの説明を参照)。

Active Consumers			Backup Consumers			Msg Count		
Current	Peak	Avg	Current	Peak	Avg	Current	Peak	Avg
1	1	0	0	0	0	944	1000	5

*imqcmd query*

`imqcmd query` の構文とオプションを、コマンドによって提供されるメトリックスデータの説明とともに表 9-4 に示します。

表 9-4          imqcmd query サブコマンドの構文

サブコマンドの構文	提供されるメトリックスデータ
<code>query bkr</code> <code>[-b hostName:port]</code> <code>[-int interval]</code> <code>[-msp numSamples]</code>  または  <code>metrics svc -n serviceName</code> <code>[-b hostName:port]</code> <code>[-int interval]</code> <code>[-msp numSamples]</code>  または  <code>metrics dst -t destType</code> <code>-n destName</code> <code>[-b hostName:port]</code> <code>[-int interval]</code> <code>[-msp numSamples]</code>	<p>ブローカのメモリーと持続ストアに格納されている現在のメッセージ数とメッセージバイト数に関する情報 (167 ページの「ブローカ情報の表示」を参照)</p> <p>指定したコネクションサービスに現在割り当てられているスレッドの数とそのサービスのコネクション数に関する情報 (174 ページの「コネクションサービス情報の表示」を参照)</p> <p>指定した送信先のメモリーと持続ストアに格納されている現在のプロデューサ数、アクティブコンシューマとバックアップコンシューマの数、メッセージ数とメッセージバイト数に関する情報 (183 ページの「送信先情報の表示」を参照)</p>

注	imqcmd query は限定されたメトリックスデータを提供するため、このツールは、267 ページの「メトリックスデータの送信先」の節の表には記載されていません。
---	--

Message Queue ブローカログファイル

Message Queue のロガーは、ブローカコード、デバッガ、およびメトリックスジェネレータによって生成された情報を取得し、標準出力 ( コンソール)、ログファイル、Solaris™ プラットフォーム、syslog デーモンプロセスなどの多数の出力チャネルに書き込みます。ロガーについては、75 ページの「ロガー」で説明されています。

ロガーが収集する情報のタイプと、各出力チャネルに書き込む情報のタイプを指定できます。特に、メトリックス情報のログファイルへの書き込みを指定できます。

手順：ブローカログファイルを使用したメトリックスデータの報告

この節では、ブローカログファイルを使用してメトリックス情報を報告するための手順を説明します。ロガーの設定方法については、154 ページの「ログ作成」を参照してください。

## ► ログファイルを使用してメトリックス情報を報告するには

1. ブローカのメトリックス生成機能を設定します。

- a. `imq.metrics.enabled=true` に設定されていることを確認します。

デフォルトでは、ログ用のメトリックスの生成は有効になっています。

- b. メトリックスの生成間隔を適切な秒数に設定します。

`imq.metrics.interval=interval`

この値は、`config.properties` ファイル内で設定するか、または、ブローカの起動時に `-metrics interval` コマンド行オプションを使用して設定できます。

2. ロガーがメトリックス情報を収集していることを確認します。

`imq.log.level=INFO`

デフォルト値は `true` この値は、`config.properties` ファイル内で設定するか、または、ブローカの起動時に `-loglevel level` コマンドオプションを使用して設定できます。

3. ロガーが、メトリックス情報をログファイルへ書き込むように設定されていることを確認します。

`imq.log.file.output=INFO`

これはデフォルト値です。`config.properties` ファイル内で設定できます。

4. ブローカを起動します。

## メトリックスの出力：ログファイル

次は、ログファイルに出力されたブローカメトリックスのサンプルを示しています (表 9-7 と 268 ページの表 9-8 のメトリックスデータの説明を参照)。

```
[21/Jul/2003:11:21:18 PDT]
Connections: 0      JVM Heap: 8323072 bytes (7226576 free) Threads: 0 (14-1010)
  In: 0 msgs (0bytes) 0 pkts (0 bytes)
  Out: 0 msgs (0bytes) 0 pkts (0 bytes)
  Rate In:0 msgs/sec (0 bytes/sec) 0 pkts/sec (0 bytes/sec)
  Rate Out:0 msgs/sec (0 bytes/sec) 0 pkts/sec (0 bytes/sec)
```

## メッセージベースの監視 API

Message Queue には、ブローカがメトリックスデータを JMS メッセージへ書き込み、そのメッセージに含まれるメトリックス情報のタイプに応じて、そのメッセージを多数のメトリックストピック送信先のどれかに送信する際に使用できるメトリックス監視機能が用意されています。

メトリックストピックを送信先へサブスクライブし、これらの送信先のメッセージをコンシュームし、メッセージに含まれるメトリックス情報を処理するクライアントアプリケーションをプログラミングすることで、このメトリックス情報にアクセスできます。このスキーマの概略については、[76 ページの「メトリックスメッセージプロデューサ \(Enterprise Edition\)」](#)で説明されています。

5 つのメトリックストピック送信先があります。それらの名前と、各送信先へ配信されるメトリックスメッセージのタイプを表 [9-5](#) に示します。

**表 9-5**          メトリックスのトピック送信先

トピック名	メトリックスメッセージのタイプ
mq.metrics.broker	ブローカのメトリックス
mq.metrics.jvm	Java 仮想マシンのメトリックス
mq.metrics.destination_list	送信先とそれらのタイプのリスト
mq.metrics.destination.queue. monitoredDestinationName	指定した名前のキューの送信先メトリックス
mq.metrics.destination.topic. monitoredDestinationName	指定した名前のトピックの送信先メトリックス

### 手順：メッセージベースの監視の設定

この節では、メッセージベースの監視機能を使用してメトリックス情報を収集するための手順を説明します。手順には、クライアント開発タスクと管理タスクの両方が含まれます。

#### ➤ メッセージベースの監視を設定するには

1. メトリックス監視クライアントを作成します。

メトリックストピック送信先へサブスクライブし、メトリックスメッセージをコンシュームし、これらのメッセージからメトリックスデータを抽出するクライアントをプログラミングする手順については、『[Message Queue Java Client Developer's Guide](#)』を参照してください。

2. config.properties ファイルにブローカプロパティ値を設定して、ブローカのメトリックスメッセージプロデューサを設定します。
  - a. メトリックスメッセージのプロデュースを有効にする  
`mq.metrics.topic.enabled=true` と設定する  
デフォルト値は true

- b. メトリックスメッセージを生成する間隔を、秒単位で指定する

`imq.metrics.topic.interval=interval` と設定する

デフォルトは 60 秒

- c. メトリックスメッセージを持続的にするかどうか、つまり、ブローカ障害時にもそのまま保持するかどうかを指定する

`imq.metrics.topic.persist` を設定する

デフォルト値は `false`

- d. 各送信先で、メトリックスメッセージを削除するまでに保持しておく期間を指定する

Set `imq.metrics.topic.timetolive`

デフォルト値は 300 秒

3. メトリックストピック送信先に必要なアクセス制御を設定します。

設定方法は次の、「[セキュリティとアクセスで考慮すること](#)」を参照してください。

4. メトリックス監視クライアントを起動します。

コンシューマがメトリックストピックをサブスクライブすると、メトリックストピック送信先が自動的に作成されます。メトリックストピックが作成されると、ブローカのメトリックスメッセージプロデューサがメトリックスメッセージをメトリックストピックへ送信し始めます。

## セキュリティとアクセスで考慮すること

メトリックストピック送信先へのアクセスを制限する理由は 2 つあります。

- メトリックスデータにブローカとそのリソースに関する機密情報が含まれることがある
- メトリックストピック送信先へのサブスクリプション数が過剰になると、ブローカのオーバーヘッドが増加し、パフォーマンスに悪影響を及ぼすことがある

これらの点を考慮して、メトリックストピック送信先へのアクセスは制御することをお勧めします。

監視クライアントは、そのほかのクライアントと同じ認証制御と権限を前提にしています。ブローカへの接続が許可されるのは、**Message Queue** ユーザーリポジトリに登録されているユーザーだけです。

[224 ページの「ユーザーの承認: アクセス制御プロパティファイル」](#)に説明されているとおり、アクセス制御プロパティファイルを使用して特定のメトリックストピック送信先へのアクセスを制限することで、さらに保護を強化できます。

たとえば、`accesscontrol.properties` ファイル内の次のエントリは、`user1` と `user2` を除き、すべてのユーザーについて `mq.metrics.broker` メトリックストピックへのアクセスを拒否します。

```
topic.mq.metrics.broker.consume.deny.user=*
topic.mq.metrics.broker.consume.allow.user=user1,user2
```

次のエントリは、ユーザー `user3` だけにトピック `t1` の監視を許可します。

```
topic.mq.metrics.destination.topic.t1.consume.deny.user=*
topic.mq.metrics.destination.topic.t1.consume.allow.user=user3
```

メトリックスデータの機密性に応じて、暗号化されたコネクションを使用してメトリックス監視クライアントをブローカへ接続することもできます。暗号化されたコネクションの使用方法については、[230 ページの「暗号化: SSL ベースのサービスとの連動 \(Enterprise Edition\)」](#)を参照してください。

## メトリックスの出力: メトリックスメッセージ

メッセージベースの監視 API を使用して取得したメトリックスデータ出力は、プログラミングしたメトリックス監視クライアントによって異なります。出力されるデータは、ブローカ内のメトリックスジェネレータによって提供されるデータだけに限定されます。このデータの完全なリストは、[267 ページの「メトリックスデータの送信先」](#)を参照してください。

## 適切な監視ツールの選択

前の節で取り上げた監視ツールにはそれぞれ長所と短所があります。

たとえば、`imgcmd metrics` コマンドを使用すると、必要なときにニーズにあったサンプル情報をすばやく作成できますが、履歴情報を確認したりプログラム上データを操作したりするのは難しくなります。

一方、ログファイルはメトリックスデータの長期間の記録を提供しますが、ログファイルの情報を解析して有意義な情報を取得するのが難しくなります。

メッセージベースの監視 API を使うと、必要な情報を容易に抽出し、それを処理して、プログラム上データを操作するか書式化して、グラフを提示したり警告を送信したりできます。ただし、データをキャプチャし分析するためのカスタムアプリケーションをプログラミングする必要があります。

さらに、これらの各ツールは、ブローカによって生成された比較的難解なメトリックス情報のサブセットを収集します。どの監視ツールがどのメトリックスデータを収集するかについては、[267 ページの「メトリックスデータの送信先」](#)を参照してください。

表 9-6 は、それぞれの長所と短所を示すことで、さまざまなツールを比較しています。

表 9-6          メトリックス監視ツールの長所と短所

メトリックス監視ツール	長所	短所
imgcmd metrics	リモート監視  スポット検査に適している  報告間隔はコマンドのオプションで設定されるため、即座に変更可能  必要とする特定のデータを容易に選択できる  わかり易い表形式でデータを表示	シングルコマンドではすべてのデータを取得できない  データ分析のプログラム化が困難  履歴レコードを作成しない  履歴的な傾向を確認するのが困難
ログファイル	定期的なサンプリング  履歴レコードの作成	ブローカプロパティの設定が必要。有効にするにはブローカをシャットダウンし再起動する必要がある  ローカル監視だけ  読み取りや解析が非常に困難なデータ形式。解析ツールなし  報告間隔を即座に変更できない。すべてのメトリックスデータについて同じ  柔軟にデータを選択できない  ブローカメトリックスだけ。送信先とコネクションサービスのメトリックスは含まれていない  間隔が短過ぎるとパフォーマンスに影響する可能性がある



表 9-6      メトリックス監視ツールの長所と短所 ( 続き )

メトリックス 監視ツール	長所	短所
メッセージベース の監視 API	リモート監視	ブローカプロパティの設定が必要。 有効にするにはブローカをシャット ダウンし再起動する必要がある
	対象となる特定のデータを 容易に選択できる	専用のメトリックス監視クライア ントをプログラミングする必要が ある
	データを電子的に分析し、 任意の形式で提示できる	報告間隔を即座に変更できない。 すべてのメトリックスデータにつ いて同じ

## メトリックスデータの送信先

ブローカによって報告されるメトリックス情報は、次のカテゴリにグループ化できます。

- **Java 仮想マシン (JVM) メトリックス**      JVM ヒープサイズに関する情報
- **ブローカ全体のメトリックス**      ブローカに格納されたメッセージ、ブローカとの間のメッセージフロー、メッセージ数とバイト数 ( 絶対値と比較値 ) の両方に関する情報。このカテゴリには、メモリー使用量に関する情報も含まれます。
- **コネクションサービスのメトリックス**      コネクションとコネクションスレッドのリソースに関する情報、および特定のコネクションサービスのメッセージフローに関する情報
- **送信先のメトリックス**      特定の送信先との間のメッセージフロー、送信先のコンシューマ、メモリーとディスクスペースの使用率に関する情報

次の節では、これらの各カテゴリで使用可能なメトリックスデータについて取り上げます。次の表に記載されている監視ツールについては、[256 ページの「監視ツール」](#)を参照してください。

## JVM メトリックス

[表 9-7](#) は、JVM ヒープを処理するためにブローカが生成するメトリックスとその説明、さらに各種監視ツールを使用してどのデータを取得できるかを一覧しています。

表 9-7 JVM メトリックス

メトリックス量	説明	imqcmd metrics bkr (metricType)	ログファイル	メトリックス メッセージ (metrics topic) <sup>2</sup>
JVM heap: free memory	JVM ヒープに使用可能な空きメモリー 量	必須 (cxn)	必須	必須 (...jvm)
JVM heap: total memory	現在の JVM ヒープサイズ	必須 (cxn)	必須	必須 (...jvm)
JVM heap: max memory	JVM ヒープサイズを拡大可能な上限	オプション	必須 <sup>1</sup>	必須 (...jvm)

1. ブローカの起動時にだけ表示される
2. メトリックスのトピック送信先名については、[263 ページの表 9-5](#) を参照

ブローカ全体のメトリックス

[表 9-8](#) は、ブローカ全体のメトリックス情報についてブローカが報告するデータとその説明を一覧しています。また、各種メトリックス監視ツールを使用してどのデータを取得できるかも示しています。

表 9-8 ブローカ全体のメトリックス

メトリックス量	説明	imqcmd metrics bkr (metricType)	ログファ イル	メトリックス メッセージ (metrics topic) <sup>1</sup>
コネクションデータ				
Num connections	現在開いているブローカへのコネク ションの数	必須 (cxn)	必須	必須 (...broker)
Num threads	現在使用中のスレッドの数	必須 (cxn)	必須	オプション
Min threads	コネクションサービスで使用するた めに、スレッドプールに保持している作 成済みのスレッド数	必須 (cxn)	必須	オプション
Max threads	コネクションサービスが使用するた めに、スレッドプールに保持するスレ ッドの最大数。新しいスレッドは、それ 以上追加されなくなる	必須 (cxn)	必須	オプション
格納済みのメッセージデータ				

表 9-8 ブローカ全体のメトリックス (続き)

メトリックス量	説明	imqcmd metrics bkr (metricType)	ログファイ ル	メトリックス メッセージ (metrics topic) <sup>1</sup>
Num messages	現在、ブローカメモリーと持続ストアに格納されている JMS メッセージの数	オプション query bkr を 使用	オプ ション	必須 (...broker)
Total message bytes	現在、ブローカメモリーと持続ストアに格納されている JMS メッセージバイトの数	オプション query bkr を 使用	オプ ション	必須 (...broker)
メッセージフローデータ				
Num messages in	ブローカが最後に起動された以降にブローカが受信した JMS メッセージの数	必須 (ttl)	必須	必須 (...broker)
Message bytes in	ブローカが最後に起動された以降にブローカが受信した JMS メッセージのバイト数	必須 (ttl)	必須	必須 (...broker)
Num packets in	ブローカが最後に起動された以降にブローカが受信したパケットの数。JMS メッセージと制御メッセージの両方が含まれる	必須 (ttl)	必須	必須 (...broker)
Packet bytes in	ブローカが最後に起動された以降にブローカが受信したパケットのバイト数。JMS メッセージと制御メッセージの両方が含まれる	必須 (ttl)	必須	必須 (...broker)
Num messages out	ブローカが最後に起動された以降にブローカから送信した JMS メッセージの数	必須 (ttl)	必須	必須 (...broker)
Message bytes out	ブローカが最後に起動された以降にブローカから送信した JMS メッセージのバイト数	必須 (ttl)	必須	必須 (...broker)
Num packets out	ブローカが最後に起動された以降にブローカから送信したパケットの数。JMS メッセージと制御メッセージの両方が含まれる	必須 (ttl)	必須	必須 (...broker)
Packet bytes out	ブローカが最後に起動された以降にブローカから送信したパケットのバイト数。JMS メッセージと制御メッセージの両方が含まれる	必須 (ttl)	必須	必須 (...broker)
Rate messages in	ブローカへの JMS メッセージの現在のフローレート	必須 (rts)	必須	オプション

表 9-8      ブローカ全体のメトリックス ( 続き )

メトリックス量	説明	imqcmd metrics bkr (metricType)	ログファ イル	メトリックス メッセージ (metrics topic) <sup>1</sup>
Rate message bytes in	ブローカへの JMS メッセージバイトの現在のフローレート	必須 (rts)	必須	オプション
Rate packets in	ブローカへのパケットの現在のフローレート。JMS メッセージと制御メッセージの両方を含む	必須 (rts)	必須	オプション
Rate packet bytes in	ブローカへのパケットバイトの現在のフローレート。JMS メッセージと制御メッセージの両方を含む	必須 (rts)	必須	オプション
Rate messages out	ブローカからの JMS メッセージの現在のフローレート	必須 (rts)	必須	オプション
Rate message bytes out	ブローカからの JMS メッセージのバイトの現在のフローレート	必須 (rts)	必須	オプション
Rate packets out	ブローカからのパケットの現在のフローレート。JMS メッセージと制御メッセージの両方を含む	必須 (rts)	必須	オプション
Rate packet bytes out	ブローカからのパケットバイトの現在のフローレート。JMS メッセージと制御メッセージの両方を含む	必須 (rts)	必須	オプション
送信先データ				
Num destinations	ブローカ内の物理的な送信先の数	オプション	オプション	必須 (...broker)

1. メトリックスのトピック送信先名については、263 ページの表 9-5 を参照

コネクションサービスのメトリックス

表 9-9 は、個々のコネクションサービスについてブローカが報告するメトリックスデータとその説明を一覧しています。また、各種メトリックス監視ツールを使用してどのデータを取得できるかも示しています。

表 9-9      コネクションサービスのメトリックス

メトリックス量	説明	imqcmd metrics svc (metricType)	ログファ イル	メトリックス メッセージ (metrics topic)
コネクションデータ				

表 9-9 コネクションサービスのメトリックス ( 続き )

メトリックス量	説明	imqcmd metrics svc (metricType)	ログファ イル	メトリックス メッセージ (metrics topic)
Num connections	現在開かれているコネクション数	必須 (cxn) query svc も 使用	オプ ション	オプション
Num threads	すべてのコネクションサービスで、現 在使用中のスレッドの合計数	必須 (cxn) query svc も 使用	オプ ション	オプション
Min threads	すべてのコネクションサービスで、コ ネクションサービスが使用するスレッ ドプールに初めに保持されるスレッド の合計数	必須 (cxn)	オプ ション	オプション
Max threads	すべてのコネクションサービスで、コ ネクションサービスが使用するスレッ ドプールに保持されるスレッドの最大 合計数。新しいスレッドは、それ以上 追加されなくなる	必須 (cxn)	オプ ション	オプション
メッセージフローデータ				
Num messages in	ブローカが最後に起動された以降にコ ネクションサービスが受信した JMS メッセージの数	必須 (ttl)	オプ ション	オプション
Message bytes in	ブローカが最後に起動された以降にコ ネクションサービスが受信した JMS メッセージのバイト数	必須 (ttl)	オプ ション	オプション
Num packets in	ブローカが最後に起動された以降にコ ネクションサービスが受信したパケッ トの数。JMS メッセージと制御メッ セージの両方が含まれる	必須 (ttl)	オプ ション	オプション
Packet bytes in	ブローカが最後に起動された以降にコ ネクションサービスが受信したパケッ トのバイト数。JMS メッセージと制御 メッセージの両方が含まれる	必須 (ttl)	オプ ション	オプション
Num messages out	ブローカが最後に起動された以降にコ ネクションサービスから送信した JMS メッセージの数	必須 (ttl)	オプ ション	オプション

表 9-9 コネクションサービスのメトリックス ( 続き )

メトリックス量	説明	imqcmd metrics svc (metricType)	ログファイ イル	メトリックス メッセージ (metrics topic)
Message bytes out	ブローカが最後に起動された以降にコ ネクションサービスから送信した JMS メッセージのバイト数	必須 (ttl)	オブ ション	オブション
Num packets out	ブローカが最後に起動された以降にコ ネクションサービスから送信したパ ケットの数。JMS メッセージと制御 メッセージの両方が含まれる	必須 (ttl)	オブ ション	オブション
Packet bytes out	ブローカが最後に起動された以降にコ ネクションサービスから送信したパ ケットのバイト数。JMS メッセージと 制御メッセージの両方が含まれる	必須 (ttl)	オブ ション	オブション
Rate messages in	コネクションサービス経由でブローカ が受信する JMS メッセージの現在のフ ローレート	必須 (rts)	オブ ション	オブション
Rate message bytes in	コネクションサービスへの JMS メッ セージバイトの現在のフローレート	必須 (rts)	オブ ション	オブション
Rate packets in	コネクションサービスへのパケットの 現在のフローレート。JMS メッセージ と制御メッセージの両方を含む	必須 (rts)	オブ ション	オブション
Rate packet bytes in	コネクションサービスへのパケットの 現在のフローレート。JMS メッセージ と制御メッセージの両方を含む	必須 (rts)	オブ ション	オブション
Rate messages out	コネクションサービスからの JMS メッ セージの現在のフローレート	必須 (rts)	オブ ション	オブション
Rate message bytes out	コネクションサービスからの JMS メッ セージバイトの現在のフローレート	必須 (rts)	オブ ション	オブション
Rate packets out	コネクションサービスからのパケット の現在のフローレート。JMS メッセー ジと制御メッセージの両方を含む	必須 (rts)	オブ ション	オブション
Rate packet bytes out	コネクションサービスからのパケット バイトの現在のフローレート。JMS メッセージと制御メッセージの両方を 含む	必須 (rts)	オブ ション	オブション

## 送信先メトリックス

表 9-9 は、個々の送信先についてブローカが報告するメトリックスデータとその説明を一覧しています。また、各種メトリックス監視ツールを使用してどのデータを取得できるかも示しています。

表 9-10 送信先メトリックス

メトリックス量	説明	imqcmd metrics dst (metricType)	ログファイ ル	メトリックス メッセージ (metrics topic) <sup>1</sup>
コンシューマのデータ				
Num active consumers	現在のアクティブなコンシューマの数	必須 (con)	オブ ション	必須 (...destName)
Avg num active consumers	ブローカが最後に起動された以降のアクティブなコンシューマの数の平均	必須 (con)	オブ ション	必須 (...destName)
Peak num active consumers	ブローカが最後に起動された以降のアクティブなコンシューマの最大時の数	必須 (con)	オブ ション	必須 (...destName)
Num backup consumers	現在のバックアップコンシューマの数。キューにだけ適用	必須 (con)	オブ ション	必須 (...destName)
Avg num backup consumers	ブローカが最後に起動された以降のバックアップコンシューマ数の平均。キューにだけ適用	必須 (con)	オブ ション	必須 (...destName)
Peak num backup consumers	ブローカが最後に起動された以降のバックアップコンシューマの最大時の数。キューにだけ適用	必須 (con)	オブ ション	必須 (...destName)
格納済みのメッセージデータ				
Num messages	現在、送信先メモリーと持続ストアに格納されている JMS メッセージの数	必須 (con) (ttl) (rts) query dst も使用	オブ ション	必須 (...destName)
Avg num messages	ブローカが最後に起動された以降に、送信先メモリーと持続ストアに格納された JMS メッセージ数の平均	必須 (con) (ttl) (rts)	オブ ション	必須 (...destName)

表 9-10 送信先メトリックス ( 続き )

メトリックス量	説明	imqcmd metrics dst (metricType)	ログファイ イル	メトリックス メッセージ (metrics topic) <sup>1</sup>
Peak num messages	ブローカが最後に起動された以降に、送信先メモリと持続ストアに格納された JMS メッセージの最大時の数	必須 (con) (ttl) (rts)	オブ ション	必須 (...destName)
Total message bytes	現在、送信先メモリと持続ストアに格納されている JMS メッセージのバイト数	必須 (ttl) (rts) query dst も使用	オブ ション	必須 (...destName)
Avg total message bytes	ブローカが最後に起動された以降に、送信先メモリと持続ストアに格納された JMS メッセージのバイト数の平均	必須 (ttl) (rts)	オブ ション	必須 (...destName)
Peak total message bytes	ブローカが最後に起動された以降に、送信先メモリと持続ストアに格納された JMS メッセージの最大時のバイト数	必須 (ttl) (rts)	オブ ション	必須 (...destName)
Peak message bytes	ブローカが最後に起動された以降に、送信先が受信したシングルメッセージ内の JMS メッセージの最大時のバイト数	必須 (ttl) (rts)	オブ ション	必須 (...destName)
メッセージフローデータ				
Num messages in	ブローカが最後に起動された以降に、この送信先が受信した JMS メッセージの数	必須 (ttl)	オブ ション	必須 (...destName)
Msg bytes in	ブローカが最後に起動された以降に、この送信先が受信した JMS メッセージのバイト数	必須 (ttl)	オブ ション	必須 (...destName)
Num messages out	ブローカが最後に起動された以降に、この送信先から送信した JMS メッセージの数	必須 (ttl)	オブ ション	必須 (...destName)
Msg bytes out	ブローカが最後に起動された以降に、この送信先から送信した JMS メッセージのバイト数	必須 (ttl)	オブ ション	必須 (...destName)
Rate num messages in	送信先への JMS メッセージの現在のフローレート	必須 (rts)	オブ ション	オブション



表 9-10 送信先メトリックス ( 続き )

メトリックス量	説明	imqcmd metrics dst (metricType)	ログファイル	メトリックス メッセージ (metrics topic) <sup>1</sup>
Rate num messages out	送信先からの JMS メッセージの現在のフローレート	必須 (rts)	オプション	オプション
Rate msg bytes in	送信先への JMS メッセージバイトの現在のフローレート	必須 (rts)	オプション	オプション
Rate Msg bytes out	送信先からの JMS メッセージのバイトの現在のフローレート	必須 (rts)	オプション	オプション
ディスク利用率データ				
Disk reserved	メッセージレコードが使用する、送信先のファイルベースのストアにある、アクティブレコードと空きレコードを含むすべてのメッセージレコードによって使用されているディスクスペース ( バイト単位 )	必須 (dsk)	オプション	必須 (...destName)
Disk used	送信先のファイルベースのストアにあるアクティブメッセージレコードによって使用されているディスクスペース ( バイト単位 )	必須 (dsk)	オプション	必須 (...destName)
Disk utilization ratio	使用されているディスクスペースを予約済みのディスクスペースで割ったときの商。割合が高いほど、アクティブメッセージを保持するためにより多くのディスクスペースが使用されている	必須 (dsk)	オプション	必須 (...destName)

1. メトリックスのトピック送信先名については、[263 ページの表 9-5](#) を参照

# パフォーマンス問題のトラブルシューティング

Message Queue サービスを使用してアプリケーションをサポートする際に発生する恐れのある多数のパフォーマンス問題があります。このような問題には次のものが含まれます。

- 問題: クライアントがコネクションを確立できない
- 問題: コネクションスループットが遅過ぎる
- 問題: クライアントがメッセージプロデューサを作成できない
- 問題: メッセージのプロデュースが遅れるまたは低速である
- 問題: メッセージサーバでメッセージがバックログされる
- 問題: メッセージサーバのスループットが散発的である
- 問題: メッセージがコンシューマに到達しない

各問題を取り上げ、その考えられる原因と解決方法を説明します。

## 問題: クライアントがコネクションを確立できない

### 現象:

- クライアントが新しいコネクションを確立できない
- クライアントが障害の生じたコネクションを自動的に再接続できない

### 考えられる原因:

- クライアントアプリケーションがコネクションを閉じていないため、コネクション数がリソース制限を越えてしまった

この問題の原因を確認するには:

ブローカへのコネクションをすべて一覧表示します。

```
imqcmd list cxn
```

出力にはすべてのコネクションと各コネクションの確立元のホストが一覧表示されます。異常な数のコネクションが開かれている特定のクライアントがわかります。

### 問題を解決するには

原因となっているクライアントが未使用のコネクションを閉じるようにプログラムし直します。

- ブローカが実行されていないか、ネットワーク接続の問題が存在している

#### この問題の原因を確認するには

- ブローカのプライマリポートへ **telnet** で接続し、ブローカがポートマップパ出力を返すか確認します。プライマリポートのデフォルトは **7676** です。
- ブローカプロセスがホスト上で実行されていることを確認します。

#### 問題を解決するには

- ブローカを起動します。
- ネットワーク接続の問題を修復します。

- コネクションサービスが非アクティブであるか停止される

#### この問題の原因を確認するには

すべてのコネクションサービスのステータスを確認します。

```
imqcmd list svc
```

コネクションサービスのステータスが **unknown** または **paused** と表示された場合、クライアントはそのサービスを使用するコネクションを確立できません。

#### 問題を解決するには

- コネクションサービスのステータスが **unknown** と表示された場合、そのサービスはアクティブサービスリスト (**imq.service.active**) に含まれていません。SSL ベースのサービスの場合は、サービスが不適切に設定されているため、ブローカがブローカログに次のエントリを作成する可能性があります。**ERROR [B3009]: Unable to start service ssljms:[B4001]: Unable to open protocol tls for ssljms service...** 例外の根本的な原因の説明が続きます。

SSL サービスを適切に設定する方法については、[230 ページの「TCP/IP を介した SSL ベースのサービスの設定」](#)を参照してください。

- コネクションサービスのステータスが **paused** と表示された場合は、サービスを再開します ([175 ページの「コネクションサービスの停止および再開」](#)を参照)。

- 必要なコネクション数に対して使用可能なスレッドが少な過ぎる

#### この問題の原因を確認するには

ブローカログの次のエントリを確認します。**WARNING [B3004]: No threads are available to process a new connection on service ... Closing the new connection.**

また、コネクションサービスのコネクション数と現在使用中のスレッド数を確認します。

```
imqcmd query svc -n serviceName
```

または

```
imqcmd metrics svc -n serviceName -m cxn
```

コネクションごとに2つのスレッドが必要です。1つは受信メッセージ用、もう1つは出力メッセージ用です (58 ページの「スレッドプールマネージャ」を参照)。

### 問題を解決するには

- 専用のスレッドプールモデル (`imq.service_name.threadpool_model=dedicated`) を使用している場合は、コネクションの最大数はスレッドプールにあるスレッドの最大数の半分です。そのため、コネクション数を増やすには、スレッドプールのサイズを拡大するか (`imq.service_name.max_threads`)、共有スレッドプールモデルに切り換えます。
- 共有スレッドプールモデル (`imq.service_name.threadpool_model=shared`) を使用している場合は、コネクションの最大数は次の2つのプロパティの結果の半分です。それは、コネクション監視制限 (`imq.service_name.connectionMonitor_limit`) とスレッドの最大数 (`imq.service_name.max_threads`) です。そのため、コネクション数を増やすには、スレッドプールのサイズを拡大するか、コネクション監視制限の値を大きくします。
- 最終的に、サポート可能なコネクションの数またはコネクションのスループットが入力 / 出力制限に達してしまいます。このような場合は、マルチブローカクラスター (147 ページの「クラスターの操作 (Enterprise Edition)」を参照) を使用して、クラスター内のブローカインスタンス間でコネクションを分散します。
- Solaris または Linux プラットフォーム上で必要なコネクション数に対してファイル記述子が少な過ぎる (354 ページの「OS で定義されるファイル記述子の制限事項」を参照)

### この問題の原因を確認するには

次のようなブローカログのエントリを確認します。Too many open files.

### 問題を解決するには

マニュアルの `ulimit` で説明しているとおり、ファイル記述子の制限を増やします。

- TCP バックログにより、確立可能な新しい同時コネクション要求の数が制限される

TCP バックログが、同時コネクション要求の数を制限する。ポートマッパーが追加要求を拒否しなければ、同時コネクション要求はシステムバックログ (`imq.portmapper.backlog`) に格納される。Windows プラットフォームでは、バックログ制限は固定されていて、Windows クライアントでは5、Windows サーバでは200に制限されている

通常、バックログ制限が原因の要求拒否は過渡的な現象であり、非常に多数の同時コネクション要求があると発生する

### この問題の原因を確認するには

ブローカログで、一部のコネクション要求は受け入れられたが、ほぼ同時に受け取ったほかのコネクション要求が拒否されていないかどうかを確認します。拒否されたコネクション要求は、`java.net.ConnectException: Connection refused` を返します。

### 問題を解決するには

次の手順で、TCP バックログ制限を解消できます。

- クライアントが確立しようとするコネクションの再試行を短い間隔で行うようにプログラミングします。この問題の過渡的な性質上、このようにプログラミングしても正常に動作します。
- `imq.portmapper.backlog` の値を大きくします。
- クライアントがコネクションを閉じずに、多くのコネクションを開いていないか確認します。
- オペレーティングシステムによって同時コネクションの数が制限される

Windows オペレーティングシステムのライセンスは、サポートされる同時リモートコネクションの数を制限する

### この問題の原因を確認するには

`imqcmd query svc` を使用して、コネクション用のスレッドが十分にあることを調べ、さらに Windows ライセンス契約書の条項を確認します。ローカルクライアントからはコネクションを確立できるが、リモートクライアントからは確立できない場合は、オペレーティングシステムの制限が問題の原因と考えられます。

### 問題を解決するには

- より多くのコネクションが許可されるように Windows ライセンスをアップグレードします。
- マルチブローカクラスタを設定して、多数のブローカインスタンスにコネクションを分散します。
- ユーザーの認証に失敗するか権限が与えられない

不正なパスワード、ユーザーリポジトリにユーザーのエントリがない、またはユーザーがコネクションサービスへのアクセス許可を持っていないといった原因で、認証は失敗することがあります。

### この問題の原因を確認するには

ブローカログのエントリで `Forbidden` エラーメッセージを確認します。このメッセージは、認証エラーを示しているだけで、その理由は示していません。

- ファイルベースのユーザーリポジトリを使用している場合は、次のコマンドを入力します。

```
imqusermgr list -i instanceName -u userName
```

出力にユーザーが表示された場合は、不正なパスワードの入力が原因と考えられます。出力に、[B3048]: User does not exist in the password file と表示された場合は、ユーザーリポジトリにエン트리がありません。

- LDAP サーバのユーザーリポジトリを使用している場合は、適切なツールを使用して、ユーザーのエン트리があるかどうかを確認します。
- アクセス制御プロパティファイルで、コネクションサービスへのアクセスが制限されていないかどうかを確認します。

#### 問題を解決するには

- ユーザーリポジトリにユーザーのエン트리がない場合は、ユーザーリポジトリにユーザーを追加します (219 ページの「ユーザーリポジトリの設定と管理」を参照)。
- 不正がパスワードが使用された場合は、正しいパスワードを入力し直します。
- アクセス制御プロパティが不正に設定されていた場合は、アクセス制御プロパティファイルを編集し、コネクションサービスへのアクセス許可を与えます (227 ページの「コネクションアクセス制御」を参照)。

## 問題：コネクションスループットが遅過ぎる

### 現象：

- メッセージスループットが期待どおりでない
- サポートされるブローカへのコネクション数は、276 ページの「問題：クライアントがコネクションを確立できない」で説明したように制限されているわけではなく、メッセージの入力 / 出力レートによって制限されている

### 考えられる原因：

- ネットワークコネクションまたは WAN が遅過ぎる

#### この問題の原因を確認するには

ネットワークへ ping し、ping が戻るまでに要する時間を確認し、ネットワーク管理者に相談します。また、ローカルクライアントを使用してメッセージを送受信し、ネットワークリンク経由でリモートクライアントを使用した場合と配信時間を比較することもできます。

#### 問題を解決するには

コネクションが遅過ぎる場合は、ネットワークリンクをアップグレードします。

- コネクションサービスプロトコルは、TCP に比べ、本質的に低速です。たとえば、SSL ベースのプロトコルや HTTP ベースのプロトコルは、TCP より低速です (252 ページの図 9-5 を参照)。

### この問題の原因を確認するには

SSL ベースのプロトコルまたは HTTP ベースのプロトコルを使用している場合は、TCP を使用して配信時間を比較してみます。

### 問題を解決するには

通常、アプリケーション要件によって使用するプロトコルが決定されます。そのため、[294 ページの「トランスポートプロトコルの調整」](#)の説明にしたがいプロトコルを調整する以外に、対象方法はほとんどありません。

- コネクションサービスプロトコルが最適に調整されていない

### この問題の原因を確認するには

プロトコルを調整し、違いが生じるかどうかを確認します。

### 問題を解決するには

[294 ページの「トランスポートプロトコルの調整」](#)の説明にしたがいプロトコルを調整します。

- メッセージのサイズが大きく、多くの帯域幅を占有してしまう

### この問題の原因を確認するには

小さいサイズのメッセージでベンチマークを実行します。

### 問題を解決するには

- `java.util.zip` を使用してメッセージの本体を圧縮します。
- データを送信することの通知としてメッセージを使用し、データの送信には別のプロトコルを使用します。
- コネクションスループットが低速であるように見えるが、実際は、メッセージ配信プロセスのほかの手順にボトルネックがある

### この問題の原因を確認するには

コネクションスループットが低速であるように見えるが、前に述べたような原因が見当たらない場合は、[241 ページの図 9-1](#)を参照して、そのほかの考えられるボトルネックを特定し、次の問題に関連する現象が出ていないかどうかを確認します。

- [283 ページの「問題：メッセージのプロデュースが遅れるまたは低速である」](#)
- [285 ページの「問題：メッセージサーバでメッセージがバックログされる」](#)
- [290 ページの「問題：メッセージサーバのスループットが散発的である」](#)

### 問題を解決するには

前に述べた問題のトラブルシューティングの節に記載された問題の解決方法に従います。

## 問題：クライアントがメッセージプロデューサを作成できない

### 現象：

- メッセージプロデューサが送信先に対して作成できず、クライアントは例外を受け取った

### 考えられる原因：

- 限定された数のプロデューサだけを許可するように送信先が設定されている  
送信先でのメッセージの蓄積を回避する 1 つの方法は、送信先がサポート可能なプロデューサの数を (`maxNumProducers`) で限定することです。

#### この問題の原因を確認するには

送信先を確認します ([183 ページの「送信先情報の表示」](#) を参照)。

```
imqcmd query dst
```

出力に現在のプロデューサ数と `maxNumProducers` の値が表示されます。2 つの値が同じ場合、プロデューサ数は設定済みの制限に達しています。ブローカは新しいプロデューサを拒否したときには、`ResourceAllocationException [C4088]: A JMS destination limit was reached` を返し、ブローカログに次のエントリを作成します。[B4183]: `Producer can not be added to destination.`

#### 問題を解決するには

`maxNumProducers` 属性の値を大きくします ([184 ページの「送信先の属性の更新」](#) を参照)。

- アクセス制御プロパティファイル内の設定により、ユーザーがメッセージプロデューサの作成を承認されていない

#### この問題の原因を確認するには

ブローカは新しいプロデューサを拒否したときには、`JMSSecurityException [C4076]: Client does not have permission to create producer on destination` を返し、ブローカログに次のエントリを作成します。[B2041]: `Producer on destination denied` および [B4051]: `Forbidden guest`

#### 問題を解決するには

ユーザーがメッセージをプロデュースできるようにアクセス制御プロパティを変更します ([228 ページの「送信先アクセス制御」](#) を参照)。



## 問題：メッセージのプロデュースが遅れるまたは低速である

### 現象：

- 持続性メッセージを送信したときに、`send()` メソッドが戻らずクライアントがブロックする
- 持続性メッセージを送信したときに、クライアントが例外を受け取る
- プロデュースングクライアントの処理速度が低下する

### 考えられる原因：

- メッセージサーバがバックログされ、メッセージがブローカメモリーに蓄積されており、処理速度の低下したメッセージプロデュースが応答している

送信先メモリー内のメッセージ数またはメッセージのバイト数が設定された制限に達すると、ブローカは指定された制限の動作に従いメモリーリソースを節約しようとし、次の制限の動作により、メッセージプロデュースの処理速度が低下します。

- `FLOW_CONTROL`: ブローカが持続性メッセージの受信を即時に通知しないため、プロデュースングクライアントがブロックされる
- `REJECT_NEWEST`: ブローカが新しいメッセージを拒否し、拒否された持続性メッセージごとに例外をスローする

同様に、ブローカ全体のメモリー内 (すべての送信先に対応) のメッセージ数またはメッセージのバイト数が設定済みの制限に達すると、ブローカは最新のメッセージを拒否してメモリーリソースを節約しようとし、

また、送信先またはブローカ全体の制限が適切に設定されていないために、システムメモリーの制限に達すると、ブローカはメッセージプロデュースを徐々に減らすなどのさらに重大なアクションを実行してメモリーの過負荷を防ぎます。

これらのメカニズムについては、[63 ページの「メモリーリソースとメッセージフローの管理」](#)を参照してください。

### この問題の原因を確認するには

設定済みのメッセージの制限が原因でブローカによってメッセージが拒否された場合は、ブローカが `JMSEException [C4036]: A server error occurred` を返し、ブローカログに次のエントリを作成します。 `WARNING [B2011]: Storing of JMS message from IMQconn failed` に続き、制限に達したことを示すメッセージが表示されます。

- 送信先上でメッセージが制限されている場合は、ブローカは次のようなエントリを作成する `[B4120]: Can not store message on destination destName because capacity of maxNumMsgs would be exceeded.`

- 。 ブローカ全体でメッセージが制限されている場合は、ブローカは次のようなエントリを作成する [B4024]: The Maximum Number of messages currently in the system has been exceeded, rejecting message.

多くの場合は、送信先とブローカにクエリーし設定済みのメッセージ制限の設定値を検査するか、送信先またはブローカ全体の現在のメッセージ数とメッセージのバイト数を監視し適切な `imgcmd` コマンドを使用することで、拒否される前にメッセージの制限条件を確認できます (それぞれ、[273 ページの表 9-10](#) と [268 ページの表 9-8](#) を参照)。

### 問題を解決するには

メッセージがバックログされたことでプロデューサの処理が低下する問題を解消するためのアプローチは多数あります。

- 。 送信先またはブローカ全体のメッセージ制限がメモリーリソースを超えないように注意深く変更する。一般に、ブローカ全体のメッセージ制限に達しないように、送信先単位でメモリーを管理した方が良い。詳細は、[297 ページの「ブローカの調整」](#)を参照
- 。 メッセージ制限に達したときに、メッセージのプロデュースが低速化しないようにする代わりに、メモリー内のメッセージを廃棄するよう、送信先の制限の動作を変更する。たとえば、メモリーに累積されたメッセージを削除する `REMOVE_OLDEST` および `REMOVE_LOW_PRIORITY` といった制限の動作を指定できる ([180 ページの表 6-10](#) を参照)
- ブローカは持続性メッセージをデータストアに保存できない

ブローカがデータストアにアクセスできないか、または持続性メッセージをデータストアに書き込めない場合は、プロデュースングクライアントがブロックされます。前に述べたとおり、この状態は、送信先またはブローカ全体のメッセージ制限に達したときにも発生します。

### この問題の原因を確認するには

ブローカは、データストアに書き込めない場合には、ブローカログに次のエントリのどれかを作成します。[B2011]: Storing of JMS message from connectionID failed... または [B4004]: Failed to persist message messageID...

### 問題を解決するには

- 。 組み込み持続の場合は、ファイルベースのデータストアのディスクスペースを増やしてみる
- 。 JDBC 互換のデータストアの場合は、プラグイン持続が正しく設定されていることを確認する ([付録 B「プラグイン持続の設定」](#)を参照)。正しく設定されている場合は、データベース管理者にほかのデータベース問題の解決を依頼する
- ブローカによる通知のタイムアウトが短過ぎる

低速なコネクションまたは、CPU 使用率が高いかメモリーリソースが不十分なためにメッセージサーバの能力が低下したことが原因で、ブローカが持続性メッセージの受信を通知するまでに、コネクションファクトリの `imqAckTimeout` 属性値で許容されている以上の時間を必要としています。

#### この問題の原因を確認するには

`imqAckTimeout` 値を超えると、ブローカは `JMSEException [C4000]: Packet acknowledge failed.` を返します。

#### 問題を解決するには

`imqAckTimeout` コネクションファクトリ属性値を変更します (197 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属性」を参照)。

- プロデュースングクライアントが JVM 制限に達している

#### この問題の原因を確認するには

- クライアントアプリケーションが `Out of Memory` エラーを受け取ったかどうかを確認する
- `freeMemory()`、`MaxMemory()`、および `totalMemory()` などのランタイムメソッドを使用して JVM ヒープの使用可能な空きメモリーを確認する

#### 問題を解決するには

JVM を調整します (293 ページの「Java 仮想マシン (JVM) の調整」を参照)。

## 問題：メッセージサーバでメッセージがバックログされる

### 現象：

- ブローカまたは特定の送信先のメッセージ数またはメッセージのバイト数が時間の経過とともに徐々に増えていく

メッセージが蓄積されているかどうかを確認するため、ブローカ内のメッセージ数またはメッセージのバイト数が時間の経過とともにどのように変化するかを確認し、設定済みの制限と比較します。最初に、設定済みの制限を確認します。

```
imqcmd query bkr
```

注: `imqcmd metrics bkr` サブコマンドは、この情報を表示しません。

その後、各送信先でのメッセージの蓄積を確認します。

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

または

```
imqcmd metrics dst -t destType -n destName -m ttl
```

メッセージが設定済みの送信先またはブローカ全体の制限を越えているかどうかを判断するため、ブローカログで次のエントリを確認します。WARNING [B2011]: Storing of JMS message from...failed. このエントリには、制限を越えたことを示す別のエントリが続きます。

- メッセージのプロデュースが遅い、またはプロデュースされたメッセージがブローカによって拒否される
- メッセージがコンシューマに到達するまでに異常に長い時間がかかる

## 考えられる原因:

- クライアントコードの欠陥: コンシューマがメッセージを通知していない

メッセージは、すべてのコンシューマによってメッセージの送信先へ通知されるまで、送信先で保持されます。したがって、クライアントがコンシュームしたメッセージを通知しない場合、メッセージは削除されずに送信先で蓄積されます。

たとえば、クライアントコードは次の欠陥を持っている可能性があります。

- CLIENT\_ACKNOWLEDGEMENT または処理済みセッションを使用しているコンシューマが、定期的に `Session.acknowledge()` または `Session.commit()` を呼び出していない可能性がある
- AUTO\_ACKNOWLEDGE セッションを使用しているコンシューマが何らかの理由で停止している可能性がある

## この問題の原因を確認するには

メッセージサーバの負荷が高くない場合、つまり、送信先との間のメッセージのフローレートが低い場合は、通知されていないため、メッセージが蓄積されている可能性があります。

ブローカとの間のメッセージのフローレートを確認します。

```
imqcmd metrics bkr -m rts
```

その後、個々の送信先についてそれぞれのフローレートを確認します。

```
imqcmd metrics bkr -t destType -n destName -m rts
```

また、メッセージが正しく通知されているかどうかを判断するため、クライアントコードを確認します。

- トピック送信先に非アクティブな永続的サブスクリプションがある

永続的サブスクリプションが非アクティブな場合は、該当するコンシューマがアクティブになりメッセージをコンシュームできるようになるまで、メッセージは送信先に格納されます。

## この問題の原因を確認するには

各トピック送信先の永続的サブスクリプションの状態を確認します。

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

### 問題を解決するには

次のアクションのどれかを実行できます。

- 原因となっている永続的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 (188 ページの「永続サブスクリプションの管理」を参照)
- トピックのメッセージの制限と制限の動作属性を指定する (180 ページの表 6-10 を参照)。たとえば、メモリーに累積されたメッセージを削除する REMOVE\_OLDEST および REMOVE\_LOW\_PRIORITY といった制限の動作を指定できる
- 該当する送信先からすべてのメッセージをパージする (185 ページの「送信先のパージ」を参照)
- メッセージをメモリー内で存続できる時間を制限する。プロデュースングクライアントをプログラムし直し、メッセージごとに生存時間の値を設定できる。imqOverrideJMSEExpiration および imqJMSEExpiration コネクションファクトリ属性を設定することで、コネクションを共有するすべてのプロデュースアのこれらの設定値をオーバーライドできる (198 ページの表 7-3 を参照)
- キュー内のメッセージをコンシュームするために使用可能なコンシューマが少な過ぎる

メッセージを配信可能なアクティブなコンシューマが少な過ぎる場合は、メッセージが蓄積するにつれ、キュー送信先がバックログされる恐れがあります。この状態は、次の理由のどれかが原因で発生することがあります。

- 送信先に対応するアクティブなコンシューマが少な過ぎる
- コンシューミングクライアントがコネクションの確立に失敗した
- アクティブなコンシューマがキュー内のメッセージに一致するセクタを使用していない

### この問題の原因を確認するには

コンシューマ使用できない理由を判断するために、送信先のアクティブなコンシューマの数を確認します。

```
imqcmd metrics dst -n destName -t q -m con
```

### 問題を解決するには

コンシューマが使用できない理由に応じて、次のアクションのどれかを実行できます。

- 追加のコンシューミングクライアントを起動して、キューに対応するアクティブなコンシューマを増やす
- imq.consumerFlowLimit ブローカプロパティを調整して、複数のコンシューマへのキュー配信を最適化する (298 ページの「複数のコンシューマキューのパフォーマンス」を参照)

- キューのメッセージの制限と制限の動作属性を指定する (180 ページの表 6-10 を参照)。たとえば、メモリーに累積されたメッセージを削除する REMOVE\_OLDEST および REMOVE\_LOW\_PRIORITY といった制限の動作を指定できる
- 該当する送信先からすべてのメッセージをパージする (185 ページの「送信先のページ」を参照)
- メッセージをメモリー内で存続できる時間を制限する。プロデュースングクライアントをプラグラミングし直し、メッセージごとに生存期間の値を設定できる。imqOverrideJMSEExpiration および imqJMSEExpiration コネクションファクトリ属性を設定することで、コネクションを共有するすべてのプロデュースのこれらの設定値をオーバーライドできる (198 ページの表 7-3 を参照)
- メッセージプロデュースの処理速度についていくには、メッセージコンシューマの処理速度が遅過ぎる

この場合、トピックのサブスクライバまたはキューの受信側は、プロデュースがメッセージを送信する速度より遅い速度でメッセージをコンシュームしています。この不均衡が原因で、複数の送信先にメッセージがバックログされています。

### この問題の原因を確認するには

ブローカとの間のメッセージのフローレートを確認します。

```
imqcmd metrics bkr -m rts
```

その後、個々の送信先についてそれぞれのフローレートを確認します。

```
imqcmd metrics bkr -t destType -n destName -m rts
```

### 問題を解決するには

- コンシューミングクライアントコードを最適化する
  - キュー送信先の場合は、アクティブなコンシューマの数を増やす (298 ページの「複数のコンシューマキューのパフォーマンス」を参照)
  - クライアントの通知処理が、メッセージのコンシュームを遅くする
- クライアントの通知処理には 2 つの要因が影響しています。
- クライアント通知の処理時に、大量のブローカリソースが使用されることがある。その結果、このような通知モードでは、ブローカがクライアント通知を確認するまでコンシューミングクライアントがブロックされるので、メッセージのコンシュームが遅くなることがある。
  - JMS ペイロードメッセージと、クライアント通知などの Message Queue 制御メッセージは同じコネクションを共有する。その結果、制御メッセージが JMS ペイロードメッセージによって保留され、メッセージのコンシュームを低速化させることがある。

### この問題の原因を確認するには

メッセージのフローをパケットのフローと比較して確認します。1 秒当たりのパケット数がメッセージの数と比例していない場合は、クライアントの通知が問題と考えられます。

また、クライアントが `JMSEException [C4000]: Packet acknowledge failed` メッセージを受信したかどうかを確認します。

### 問題を解決するには

- クライアントの通知モードを変更する。たとえば、`DUPS_OK_ACKNOWLEDGEMENT` または `CLIENT_ACKNOWLEDGEMENT` に切り換える
- `CLIENT_ACKNOWLEDGEMENT` または処理済みのセッションを使用している場合は、より多数のメッセージを単一の通知にグループ化する
- コンシューマとコネクションのフロー制御パラメータを調整する (299 ページの「クライアントランタイムのメッセージフローの調整」を参照)
- ブローカがプロデュースされたメッセージの処理に追いつけない

この場合、ブローカがメッセージをコンシューマにルートおよび配信可能な速度より早い速度でブローカにメッセージが流入しています。ブローカの遅滞は、次のどれかまたはすべてにおける制限が原因と考えられます。それは、CPU、ネットワークソケットの読み取り / 書き込み操作、ディスク読み取り / 書き込み操作、メモリーのページング、持続ストア、または JVM メモリー制限です。

### この問題の原因を確認するには

この問題にそれ以外の原因が関与していないことを確認します。

### 問題を解決するには

- コンピュータまたはデータストアの速度をアップグレードする
- ブローカクラスタを使用して、多数のブローカインスタンスに負荷を分散する

## 問題：メッセージサーバのスループットが散発的である

### 現象：

- メッセージのスループットがときどき低下し、その後通常のパフォーマンスに戻る

### 考えられる原因：

- ブローカのメモリーリソースがかなり不足している

送信先とブローカに制限が適切に設定されなかったため、ブローカはメモリーが過負荷になるのを防ぐためにさらに重大なアクションを実行します。このため、メッセージのバックログがクリアされるまでは、ブローカの処理がかなり遅くなります。

#### この問題の原因を確認するには

ブローカのログで、メモリー不足の状態になっていないかどうかを確認します。  
[B1089]: In low memory condition, broker is attempting to free up resources に続き、メモリーの最新の状態と、使用中のメモリーの合計を示すエントリが表示されます。

また、JVM ヒープ内の使用可能な空きメモリーも確認します。

```
imqcmd metrics bkr -m cxn
```

JVM メモリーの合計値が JVM メモリーの最大値に近くなると、空きメモリーは不足がちになります。

#### 問題を解決するには

- JVM を調整する (293 ページの「[Java 仮想マシン \(JVM\) の調整](#)」を参照)
- システムスワップスペースを増やす
- JVM メモリーの再利用 (ガベージコレクション) を実行する  
定期的なメモリー再利用によりシステム全体を一掃し、メモリーを開放します。これが実行されると、すべてのスレッドがブロックされます。より多くのメモリーが開放され、JVM ヒープサイズがより大きくなるほど、メモリー再利用に起因する遅延も長くなります。

#### この問題の原因を確認するには

コンピュータ上の CPU 使用率を監視します。メモリー再利用が実行されると、大幅に低下します。

また、次のコマンド行オプションを使用してブローカを起動します。



```
-vmargs -verbose:gc
```

標準出力では、メモリ再利用に要した時間が示されます。

### 問題を解決するには

複数の CPU を持つコンピュータでは、メモリ再利用を並行して実行するように設定します。

```
-XX:+UseParallelGC=true
```

- JVM は JIT コンパイラを使用してパフォーマンスを高速化させる

### この問題の原因を確認するには

この問題にそれ以外の原因が関与していないことを確認します。

### 問題を解決するには

しばらくの間システムを稼働させておくと、パフォーマンスは改善するはずです。

## 問題：メッセージがコンシューマに到達しない

### 現象：

- プロデューサによって送信されたメッセージをコンシューマが受信しない

### 考えられる原因：

- 制限の動作が、ブローカでのメッセージの削除を引き起こしている

送信先メモリ内のメッセージ数またはメッセージのバイト数が設定済みの制限に達すると、ブローカはメモリリソースを節約しようとします。これらの制限に達したときにブローカが実行する 3 つの設定可能な動作によって、メッセージが失われることがあります。

- REMOVE\_OLDEST: もっとも古いメッセージを削除する
- REMOVE\_LOW\_PRIORITY: メッセージの有効期間に従いもっとも優先度の低いメッセージを削除する
- REJECT\_NEWEST: 新しいメッセージを拒否する。拒否した持続性メッセージに関する例外をスローする

ブローカのメモリ内のメッセージ数またはメッセージのバイト数が設定済みの制限に達すると、ブローカは最新のメッセージを拒否してメモリリソースを節約しようとします。

### この問題の原因を確認するには

ブローカログで次のエントリを確認します。WARNING [B2011]: Storing of JMS message from...failed. このエントリには、制限を越えたことを示す別のエントリが続きます。エントリがない場合でも、メッセージが削除されたことが示されています。

### 問題を解決するには

制限を変更するか、動作を変更します。

- メッセージのタイムアウトし期限切れになった

ブローカは、タイムアウトして期限切れになったメッセージを削除します。送信先がメッセージで過分にバックログされている場合、生存期間の値が短過ぎるメッセージは削除されます。

### この問題の原因を確認するには

ブローカログファイルで次のエントリを確認します。Expiring Messages: Expired *n* messages.

### 問題を解決するには

オーバーライドを使用します。

- 異なるコンピュータ間で時計の時間が同期化していない

時計が同期化されていない場合、ブローカによるメッセージの生存期間の計算がエラーとなり、メッセージが有効期限を越え、削除される場合があります。

### この問題の原因を確認するには

すべてのコンピュータの時計を確認します。

### 問題を解決するには

時計を同期化します ([353 ページの「システムの時計の設定」](#)を参照)。

- コンシューミングクライアントがコネクションでのメッセージ配信の起動に失敗した

クライアントコードがコネクションを確立し、そのコネクション上でメッセージ配信を開始するまで、メッセージは配信できません。

### この問題の原因を確認するには

クライアントコードがコネクションを確立しメッセージ配信を開始したことを確認します。

### 問題を解決するには

コネクションを確立しメッセージ配信を開始するように、クライアントコードをプログラミングし直します。

# パフォーマンスを改善するための設定の調整

## システムの調整

次の節では、オペレーティングシステム、JVM、および通信プロトコルで実行できる調整について説明します。

### Solaris での調整 : CPU 使用率、ページング / スワッピング / ディスク I/O

オペレーティングシステムの調整については、システムのマニュアルを参照してください。

### Java 仮想マシン (JVM) の調整

デフォルトでは、ブローカは 192M バイトの JVM ヒープサイズを使用します。通常、大量のメッセージ負荷がある場合はこのサイズでは小さ過ぎるため、大きくする必要があります。

Java オブジェクトが使用する JVM のヒープ容量を使い果たしそうになると、ブローカは、フロー制御やメッセージスワップなどのさまざまな技術を使用して、メモリーを解放します。極端な状況のもとでは、メモリーを開放し、メッセージの流入を減少させるために、クライアントコネクションを閉じることもあります。このような状況を避けるため、最大 JVM ヒープ容量を十分に高く設定するようお勧めします。

ただし、Java の最大ヒープ容量がシステムの物理メモリーに対して高くしすぎた場合、ブローカは Java ヒープ容量を増加し続け、システム全体のメモリーを使い果たしてしまうことがあります。これは、パフォーマンスの低下、予期しないブローカのクラッシュにつながり、そのシステムで実行されているほかのアプリケーションやサービスの動作にも影響を与える場合があります。一般に、オペレーティングシステムとそのほかのアプリケーションがマシンをマシン上で実行させるために十分な物理メモリーを使用させる必要があります。

一般に、通常時とピーク時のシステムメモリーフットプリントを評価して、十分なパフォーマンスを得られて、しかもシステムメモリーに問題を生じさせるほどではない大きさに Java ヒープサイズを設定するのがよい方法です。

ブローカの最小ヒープサイズと最大ヒープサイズを変更するには、ブローカの起動時に `-vmargs` コマンド行オプションを使用します。たとえば、次のように指定します。

```
/usr/bin/imqbrokerd -vmargs "-Xms256m -Xmx1024m"
```

このコマンドは、起動時の Java ヒープサイズを 256M バイトに、最大 Java ヒープサイズを 1G バイトに設定します。

- Solaris では、`/etc/rc`、つまり `/etc/init.d/imq` を介してブローカを起動する場合には、`/etc/imq/imqbrokerd.conf` ファイルにブローカコマンド行引数を指定する。詳細は、そのファイルのコメントを参照
- Windows では、ブローカを Windows のサービスとして起動する場合には、`imqsvcadmin install` コマンドに `-vmargs` オプションを使用して JVM 引数を指定する [348 ページの「サービス管理ユーティリティ \(imqsvcadmin\)」](#) を参照

どの場合も、ブローカのログファイルを確認するか、`imqcmd metrics bkr -m cxn` コマンドを使用して、設定を検証します。

## トランスポートプロトコルの調整

アプリケーションのニーズを満たすプロトコルが選択されたら、選択されたプロトコルに基づいて調整を加えることでパフォーマンスを改善できます。

プロトコルのパフォーマンスは、次の 3 つのブローカプロパティを使用して修正できます。

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

TCP と SSL プロトコルの場合、これらのプロパティがクライアントとブローカ間のメッセージ配信の速度に影響します。HTTP プロトコルと HTTPS プロトコルの場合は、これらのプロパティが、Web サーバ上で実行している Message Queue トンネルサブレットとブローカ間のメッセージ配信の速度に影響します。HTTP/HTTPS プロトコルの場合、そのほかにもパフォーマンスに影響することのあるプロパティがあります ([296 ページの「HTTP/HTTPS の調整」](#) を参照)。

プロトコルを調整するためのプロパティについては、次の節で説明します。

### *nodelay*

`nodelay` プロパティは、特定のプロトコルの Nagle のアルゴリズム、つまり TCP/IP 上の TCP\_NODELAY ソケットレベルのオプションの値に影響します。Nagle のアルゴリズムは、広域ネットワーク (WAN) などの低速コネクションを使用しているシステム上で TCP パフォーマンスを改善するために使用されます。

このアルゴリズムが使用されている場合、TCP は、データをサイズの大きいパケットにバンドルすることで、複数の小さいデータの塊がリモートシステムへ送信されるのを防ぎます。ソケットに書き込まれたデータが必要なバッファサイズを満たしていない場合、プロトコルは、バッファが満たされるか、一定の遅延時間が経過するまで、パケットの送信を遅らせます。バッファがいっぱいになるか、タイムアウトが発生すると、パケットが送信されます。

大半のメッセージングアプリケーションでは、パケットの送信に遅延がない、つまり Nagle のアルゴリズムが無効な場合にパフォーマンスは最適となります。これは、クライアントとブローカ間の大半の対話が、要求 / 応答型の対話だからです。つまり、クライアントはデータの packets をブローカへ送信し、その応答を待ちます。たとえば、典型的な対話には次のものがあります。

- コネクションの作成
- プロデューサまたはコンシューマの作成
- 持続性メッセージの送信。ブローカはメッセージの受信を確認する
- `AUTO_ACKNOWLEDGE` セッションまたは `CLIENT_ACKNOWLEDGE` セッションでのクライアント通知の送信。ブローカは通知の処理を確認する

これらの対話では、大半の packets がバッファサイズより小さいサイズです。つまり、Nagle のアルゴリズムが使用されている場合は、ブローカは数ミリ秒遅れて、コンシューマに応答を送信します。

ただし、Nagle のアルゴリズムは、コネクションが低速でブローカの応答が必要ない状況で、パフォーマンスを改善することができます。これは、クライアントが持続性のないメッセージを送信する場合や、クライアント通知がブローカによって確認されない場合 (`DUPS_OK_ACKNOWLEDGE` セッション) です。

### *inbufsz/outbufsz*

`inbufsz` プロパティは、ソケットからのデータを読み取る入力ストリームのバッファサイズを設定します。同様に、`outbufsz` は、ブローカがデータをソケットに書き込むために使用する出力ストリームのバッファサイズを設定します。

一般に、どちらのパラメータも受信 packets または送信 packets の平均サイズより多少大きい値に設定する必要があります。経験上、これらのプロパティは packets の平均サイズに 1K バイトを足した値 (K バイト単位で四捨五入) に設定すると良いでしょう。

たとえば、本体が 1K バイトの packets をブローカで受信している場合、packets 全体のサイズ (メッセージ本体 + ヘッダー + プロパティ) は約 1200 バイトです。

`inbufsz` を 2K バイト (2048 バイト) にすると、妥当なパフォーマンスが得られます。

`inbufsz` または `outbufsz` をそのサイズより大きくすると多少パフォーマンスは改善しますが、コネクションごとに必要なメモリーも増えます。

図 9-6 は、1K バイトの packets の `inbufsz` を変更した結果を示しています。

図 9-7 1K バイト (1024 バイト) パケットの `inbufsz` を変更した結果

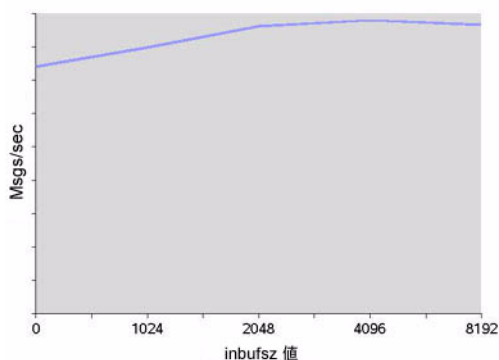
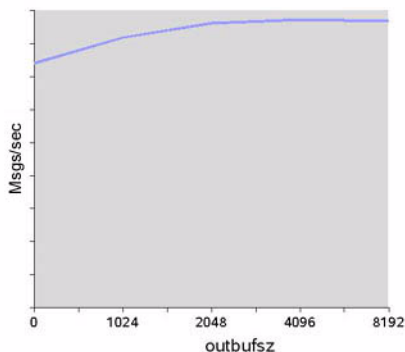


図 9-8 は、1K バイトの packets の `outbufsz` を変更した結果を示しています。

図 9-8 1K バイト (1024 バイト) パケットの `outbufsz` を変更した結果



## HTTP/HTTPS の調整

前の 2 つの節で説明した一般的なプロパティに加え、HTTP/HTTPS のパフォーマンスは、Message Queue トンネルサブレットをホスティングする Web サーバへの HTTP 要求をクライアントが作成する速度によっても制限されます。

Web サーバは、シングルソケットで複数の要求を処理するように最適化する必要があります。JDK バージョン 1.4 以降では、Web サーバが複数の HTTP 要求を処理する際に使用するリソースを最小限にするために、Web サーバへの HTTP コネクション (Web サーバへのソケット) は開かれたままになっています。JDK 1.4 を使用しているクライアントアプリケーションのパフォーマンスが JDK の旧リリースで稼働している同じアプリケーションより低速な場合は、パフォーマンスを改善するために Web サーバのキープアライブ設定パラメータの調整が必要となることがあります。

このような Web サーバの調整に加え、クライアントが Web サーバをポーリングする頻度を調整することもできます。HTTP は要求ベースのプロトコルです。つまり、HTTP ベースのプロトコルを使用しているクライアントは、メッセージが待機中かどうかを判断するために Web サーバを定期的に確認する必要があります。

`imq.httpjms.http.pullPeriod` ブローカプロパティとそれに対応する `imq.httpsjms.https.pullPeriod` プロパティは、Message Queue クライアントが Web サーバをポーリングする頻度を指定します。

`pullPeriod` 値が -1 (デフォルト値) の場合、クライアントランタイムは直前の要求が戻るとすぐにサーバをポーリングし、個々のクライアントのパフォーマンスを最大化します。その結果、各クライアントコネクションが Web サーバ内の要求スレッドを 1 つずつ占有するため、Web サーバのリソースにかなりの負荷がかかる場合があります。

`pullPeriod` 値が正の数字である場合、クライアントランタイムは要求を定期的に Web サーバへ送信し、データが保留されているかどうかを確認します。この場合、クライアントは Web サーバの要求スレッドを占有しません。したがって、多数のクライアントが Web サーバを使用している場合は、`pullPeriod` を正の値に設定することで Web サーバのリソースを節約できます。

## ファイルベースの持続ストアの調整

ファイルベースの持続ストアの調整については、[67 ページの「組み込み持続」](#)を参照してください。

## ブローカの調整

次の節では、パフォーマンスを改善するためにブローカのプロパティに対して実行できる調整について説明します。

### メモリー管理：負荷のある状態でブローカの安定性を高める

メモリー管理は、送信先単位で、またはシステム全体に対し、すべての送信先を一括で設定できます。

### 送信先の制限の使い方

送信先の制限については、[177 ページの「送信先の管理」](#)を参照してください。

## システム全体の制限の使い方

メッセージプロデューサの処理速度がメッセージコンシューマの処理速度を上回る傾向がある場合には、メッセージをブローカに蓄積できます。ブローカにはメモリーが不足した場合に、プロデューサの処理速度を低下させ、アクティブメモリーからメッセージをスワップさせるメカニズムが組み込まれていますが ([63 ページの「メモリーリソースとメッセージフローの管理」](#)を参照)、ブローカが保持可能なメッセージの合計数とメッセージのバイトの合計数に厳密な制限を設定した方が賢明です。

`imq.system.max_count` ブローカプロパティと `imq.system.max_size` ブローカプロパティを設定して、これらの制限を制御します。ブローカプロパティの設定方法については、[136 ページの「インスタンス設定ファイルの編集」](#) または [143 ページの「`imqbrokerd` オプションの概要」](#) を参照してください。

たとえば、次のように指定します。

```
imq.system.max_count=5000
```

上記で定義された値は、ブローカが未配信 / 未通知のメッセージを最大 5000 までしか保持しないことを示しています。それ以上のメッセージが送信されると、メッセージはブローカによって拒否されます。メッセージが持続的な場合は、プロデューサがメッセージを送信しようとする例外を受け取ります。メッセージが持続的でない場合は、ブローカは暗黙のうちにメッセージを廃棄します。

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

```
imqAckOnProduce = true
```

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

メッセージの送信時に例外が戻った場合、クライアントは一時停止してから、送信を再試行します。

## 複数のコンシューマキューのパフォーマンス

複数のキューコンシューマがキュー送信先のメッセージを処理する効率性は、設定可能なキュー送信先属性、すなわち、アクティブコンシューマの数

(`maxNumActiveConsumers`) とシングルバッチでコンシューマへ配信可能なメッセージの最大数 (`consumerFlowLimit`) によって左右されます。これらの属性は、[180 ページの表 6-10](#) に示されています。



最適なメッセージスループットを実現するには、十分な数のアクティブコンシューマがキューでのメッセージのプロデュースに遅れずに対応し、コンシュームする割合を最大にするような方法で、キュー内のメッセージをルートし、アクティブコンシューマへ配信しなければなりません。メッセージ配信を複数のコンシューマに分散させる一般的なメカニズムについては、[81 ページの「複数のコンシューマへのキュー配信」](#)で説明されています。

メッセージがキューに蓄積している場合、メッセージ負荷を処理するアクティブコンシューマの数が不十分であることが考えられます。また、複数のメッセージがバッチサイズでコンシューマに配信されるため、メッセージがコンシューマ上でバックアップされていることも考えられます。たとえば、バッチサイズ (consumerFlowLimit) が大き過ぎる場合は、あるコンシューマがキュー内のすべてのメッセージを受信し、そのほかのコンシューマは何も受信していないことがあります。コンシューマが非常に高速であれば、これは問題にはなりません。

ただし、コンシューマが比較的低速で、メッセージをコンシューマに均等に分散させたい場合は、バッチサイズを小さくする必要があります。バッチサイズが小さいほど、メッセージをコンシューマへ配信するのに必要なオーバーヘッドは増加します。それでも、低速なコンシューマの場合は、一般に、小さいバッチサイズを使用した方がパフォーマンスは向上します。

## クライアントランタイムのメッセージフローの調整

この節では、パフォーマンスに影響するフロー制御の動作について説明します ([255 ページの「クライアントランタイムの設定」](#)を参照)。これらの動作は、コネクションファクトリの管理対象オブジェクトの属性として設定されます。コネクションファクトリ属性の設定方法については、[第7章「管理対象オブジェクトの管理」](#)を参照してください。

### メッセージフロー測定

クライアントによって送受信されるメッセージ (JMS メッセージ) と Message Queue 制御メッセージは、同じクライアントとブローカ間のコネクションを使って伝送されます。ブローカ通知などの制御メッセージの配信における遅延は、JMS メッセージの配信によって制御メッセージが保留された場合に結果として生じます。このようなネットワークの輻輳を防止するため、Message Queue はコネクション全体の JMS メッセージのフローを測定します。

JMS メッセージは、設定した数だけ配信されるようにバッチされます ( `imqConnectionFactoryCount` プロパティの指定に従う )。つまり、バッチが配信されると、JMS メッセージの配信は中断され、保留中の制御メッセージが配信されます。JMS メッセージのそのほかのバッチが配信されるときも、このサイクルが繰り返され、その後、制御メッセージがキューに入れます。

クライアントが、ブローカからの多数の応答を必要とする操作を実行している場合、たとえば、クライアントが `CLIENT_ACKNOWLEDGE` または `AUTO_ACKNOWLEDGE` モード、持続性メッセージ、トランザクション、キューブラウザを使用している場合や、クライアントがコンシューマを追加または削除する場合などには、`imqConnectionFactoryCount` の値を小さいままにしておく必要があります。一方、クライアントが `DUPS_OK_ACKNOWLEDGE` モードを使用しており、コネクション上に単純なコンシューマしかない場合は、パフォーマンスを犠牲にすることなく `imqConnectionFactoryCount` の値を増やすことができます。

## メッセージフロー制限

**Message Queue** クライアントランタイムがメモリーなどのローカルリソースの上限に達する前に処理可能な JMS メッセージの数には制限があります。この数に達すると、パフォーマンスに悪影響が出ます。したがって、**Message Queue** では、コンシューマあたりのメッセージ数またはコネクションあたりのメッセージ数を制限できます。この制限は、コネクションを介して配信し、クライアントランタイムにバッファし、コンシュームを待機できるメッセージの数を示しています。

## コンシューマベースの制限

クライアントランタイムへ配信された JMS メッセージの数が、どれかのコンシューマの `imqConsumerFlowLimit` 値を超えた場合、そのコンシューマへのメッセージ配信は停止します。そのコンシューマのコンシュームされないメッセージの数が、`imqConsumerFlowThreshold` で設定された値を下回ったばあいだけに、配信処理が再開されます。

次の例は、これらの制限の使い方を示しています。トピックコンシューマのデフォルト設定値を前提としています。

```
imqConsumerFlowLimit=1000
imqConsumerFlowThreshold=50
```

コンシューマが作成され、1000 のメッセージがあれば、ブローカはこれらのメッセージを最初のバッチとしてコンシューマに配信します。このとき一時停止はありません。1000 メッセージの送信後、ブローカはクライアントランタイムが追加のメッセージを要求するまで、配信を停止します。アプリケーションがこれらのメッセージを処理す

るまで、クライアントランタイムはそれらを保持します。その後、クライアントランタイムがブローカに次のバッチを送信するように要求するまでの間、アプリケーションは、少なくともメッセージバッファ容量の 50% (`imqConsumerFlowThreshold`) つまり 500 メッセージをコンシュームできます。

同じ状況で、しきい値が 10% の場合、クライアントランタイムは、アプリケーションが少なくとも 900 メッセージをコンシュームしてから、次のバッチを要求します。

次のバッチサイズの計算方法：

`imqConsumerFlowLimit` - ( 現在、バッファに保留中のメッセージ数 )

そのため、`imqConsumerFlowThreshold` が 50% の場合、次のバッチサイズは、アプリケーションがメッセージを処理する速度に応じて 500 ～ 1000 の間になります。

`imqConsumerFlowThreshold` がかなり高く (100% 近くに ) 設定された場合、ブローカは比較的小さいサイズのバッチを送信するため、メッセージのスループットは低下することがあります。値が低過ぎる (0% に近い) 場合は、クライアントは次のセットを配信する前に残りのバッファされたメッセージの処理を完了してしまい、メッセージスループットを低下させることがあります。一般に、特定のパフォーマンスや信頼性を考慮しない限り、`imqConsumerFlowThreshold` 属性のデフォルト値を変更する必要はありません。

コンシューマベースのフロー制御、特に、`imqConsumerFlowLimit` は、クライアントランタイム内のメモリーを管理する最適な手段です。一般に、クライアントアプリケーションに応じて、コネクションでサポートする必要のあるコンシューマの数、メッセージのサイズ、クライアントランタイムで使用可能なメモリー総量がわかります。

## コネクションベースの制限

一部のクライアントアプリケーションでは、エンドユーザーの選択によって、コンシューマの数が不確定な場合があります。そのような場合は、引き続き、コネクションレベルのフロー制限を使用してメモリーを管理できます。

コネクションレベルのフロー制御は、コネクション上のすべてのコンシューマについてバッファされたメッセージの合計数を制限します。この数が

`imqConnectionFlowLimit` を超えると、合計数がコネクションの制限を下回るまで、コネクション経由のメッセージの配信は停止します。`imqConnectionFlowLimit` は、`imqConnectionFlowLimitEnabled` プロパティを `true` に設定した場合にだけ使用できます。

1 つのセッションでキューに入るメッセージの数は、そのセッションを使用するメッセージコンシューマの数と、各コンシューマのメッセージ負荷によって決まります。クライアント側のメッセージのプロデュースまたはメッセージのコンシュームに遅延が発生する場合は、通常は、アプリケーションを再設計し、より多くのセッションにメッセージプロデュースとメッセージコンシューマを分散し、またはより多くのコネクションにセッションを分散してパフォーマンスを改善できます。



# Message Queue データの場所

Sun Java System Message Queue 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、次の節に示すように、オペレーティングシステムに応じてそれぞれ異なる場所に保存されています。次の表の *instanceName* は、データが関連付けられているブローカインスタンスの名前を示しています。

## Solaris

表 A-1 は、Solaris プラットフォーム上での Message Queue データの場所を示しています。

注	Solaris で、Sun Java System Application Server にバンドルされている Message Queue のデータの場所は、 <a href="#">306 ページの表 A-3</a> に示されています。
---	---

表 A-1      Solaris 上での Message Queue データの場所	
データのカテゴリ	Solaris 上での場所
ブローカインスタンスの設定プロパティ	<code>/var/imq/instances/instanceName/props/config.properties</code>
ブローカ設定ファイルのテンプレート	<code>/usr/share/lib/imq/props/broker/</code>
持続ストア (メッセージ、送信先、永続サブスクリプション、トランザクション)	<code>/var/imq/instances/instanceName/fs350/</code> または JDBC のアクセス可能なデータストア
ブローカインスタンスのログファイルのディレクトリ (デフォルトの場所)	<code>/var/imq/instances/instanceName/log/</code>

表 A-1        Solaris 上での Message Queue データの場所 ( 続き )

データのカテゴリ	Solaris 上での場所
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 : ユーザーリポジトリ	/var/imq/instances/ <i>instanceName</i> /etc/passwd または LDAP サーバ
セキュリティ : アクセス制御 ファイル ( デフォルトの場所 )	/var/imq/instances/ <i>instanceName</i> /etc/ accesscontrol.properties
セキュリティ : passfile のディ レクトリ ( デフォルトの場所 )	/var/imq/instances/ <i>instanceName</i> /etc/
セキュリティ : passfile の例	/etc/imq/passfile.sample
セキュリティ : ブローカのキー ストアファイルの場所	/etc/imq/
JavaDoc API のマニュアル	/usr/share/javadoc/imq/index.html
アプリケーションと設定の例	/usr/demo/imq/
Java アーカイブ (.jar)、web アーカイブ (.war)、およびリ ソースアダプタアーカイブ (.rar) の各ファイル	/usr/share/lib/

# Linux

表 A-2 は、Linux プラットフォーム上での Message Queue データの場所を示しています。

表 A-2        Linux 上での Message Queue データの場所

データのカテゴリ	Linux 上での場所
ブローカインスタンスの設定 プロパティ	/var/opt/imq/instances/ <i>instanceName</i> /props/ config.properties
ブローカ設定ファイルのテン プレート	/opt/imq/lib/props/broker/
持続ストア ( メッセージ、送信先、永続サ ブスクリプション、トランザ クション )	/var/opt/imq/instances/ <i>instanceName</i> /fs350/ または JDBC のアクセス可能なデータストア

表 A-2 Linux 上での Message Queue データの場所 ( 続き )

データのカテゴリ	Linux 上での場所
ブローカインスタンスのログ ファイルのディレクトリ ( デ フォルトの場所 )	<code>/var/opt/imq/instances/instanceName/log/</code>
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 : ユーザーリポジ トリ	<code>/var/opt/imq/instances/instanceName/etc/ passwd</code> または LDAP サーバ
セキュリティ : アクセス制御 ファイル ( デフォルトの場所 )	<code>/var/opt/imq/instances/instanceName/etc/ accesscontrol.properties</code>
セキュリティ : <code>passfile</code> のディ レクトリ ( デフォルトの場所 )	<code>/var/opt/imq/instances/instanceName/etc/</code>
セキュリティ : <code>passfile</code> の例	<code>/etc/opt/imq/passfile.sample</code>
セキュリティ : ブローカのキー ストアファイルの場所	<code>/etc/opt/imq/</code>
JavaDoc API のマニュアル	<code>/opt/imq/javadoc/index.html</code>
アプリケーションと設定の例	<code>/opt/imq/demo/</code>
Java アーカイブ ( <code>.jar</code> )、web アーカイブ ( <code>.war</code> )、およびリ ソースアダプタアーカイブ ( <code>.rar</code> ) の各ファイル	<code>/opt/imq/lib/</code>

# Windows

表 A-3 は、Windows プラットフォーム上と Sun Java System Application Server にバンドルされた一部の Message Queue インストール上の Message Queue データの場所を示しています。詳細、および IMQ\_HOME と IMQ\_VARHOME の定義については、27 ページの表 3 を参照してください。

表 A-3 Windows 上での Message Queue データの場所	
データのカテゴリ	Windows 上での場所
ブローカインスタンスの設定プロパティ	IMQ_VARHOME¥instances¥instanceName¥props¥config.properties
ブローカ設定ファイルのテンプレート	IMQ_HOME¥lib¥props¥broker¥
持続ストア (メッセージ、送信先、永続サブスクリプション、トランザクション)	IMQ_VARHOME¥instances¥instanceName¥fs350¥ または JDBC のアクセス可能なデータストア
ブローカインスタンスのログファイルのディレクトリ (デフォルトの場所)	IMQ_VARHOME¥instances¥instanceName¥log¥
管理対象オブジェクト (オブジェクトストア)	任意のローカルディレクトリ または LDAP サーバ
セキュリティ: ユーザーリポジトリ	IMQ_VARHOME¥instances¥instanceName¥etc¥passwd  または LDAP サーバ
セキュリティ: アクセス制御ファイル (デフォルト)	IMQ_VARHOME¥instances¥instanceName¥etc¥accesscontrol.properties
セキュリティ: passfile のディレクトリ (デフォルトの場所)	IMQ_HOME¥etc¥
セキュリティ: passfile の例	IMQ_HOME¥etc¥passfile.sample
セキュリティ: ブローカのキーストアファイルの場所	IMQ_HOME¥etc¥
JavaDoc API のマニュアル	IMQ_HOME¥javadoc¥index.html
アプリケーションと設定の例	IMQ_HOME¥demo¥



表 A-3      Windows 上での Message Queue データの場所 ( 続き )

データのカテゴリ	Windows 上での場所
Java アーカイブ (.jar)、web アーカイブ (.war)、およびリソースアダプタアーカイブ (.rar) の各ファイル	IMQ_HOME¥lib¥



# 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

Message Queue のブローカには、持続情報の書き込みおよび取得を管理する持続マネージャのコンポーネントが含まれています (66 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、Message Queue 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。Message Queue 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 プラットフォーム Enterprise Edition (J2EE) SDK にバンドルされる PointBase DBMS を使用して説明しています。バージョン 1.4 は、java.sun.com からダウンロードして入手できます。例では、クライアント / サーババージョンの代わりに、PointBase の組み込みバージョンを使用します。手順の指示は、PointBase の例のパス名とプロパティ名を使用しています。これらは、「例:」という言葉で識別されます。

Oracle と PointBase の設定例は、付録 A 「[Message Queue データの場所](#)」に示す例の格納場所にあります。さらに、PointBase 組み込みバージョン、PointBase サーババージョン、Oracle、および Cloudscape の例は、インスタンス設定ファイル内でコメントアウトされた値として提供されています。

# JDBC アクセスが可能なデータストアへの接続

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

## ▶ JDBC アクセスが可能なデータストアに接続するには

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。

[312 ページの表 B-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。

`/usr/share/lib/imq/ext/` (Solaris の場合)

`/opt/imq/lib/ext/` (Linux の場合)

`IMQ_VARHOME\lib\ext` (Windows の場合)

コピーの例 (Solaris):

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

シンボリックリンクの例 (Solaris):

```
% ln -s j2eeSDK_install_directory/lib/pointbase/pointbase.jar
/usr/share/lib/imq/ext
```

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

組み込みデータベース用の `imqdbmgr create all` コマンドまたは外部データベース用の `imqdbmgr create tbl` コマンドを使用します。[316 ページの「データベース管理ユーティリティ \(imqdbmgr\)」](#) を参照してください。

例:

- a. `imqdbmgr` がある場所にディレクトリを移動します。

`cd /usr/bin` (Solaris の場合)

`cd /opt/imq/bin` (Linux の場合)

`cd IMQ_HOME/bin` (Windows の場合)

- b. `imqdbmgr` コマンドを入力します。

```
imqdbmgr create all
```

注 組み込みデータベースが使用されている場合は、次のディレクトリの下に作成することをお勧めします。

```
.../instances/instanceName/dbstore/databatbseName.
```

組み込みデータベースは、ユーザー名とパスワードで保護されていない場合、ファイルシステムのアクセス権によって保護される可能性があります。ブローカが確実にデータベースに対して読み取りと書き込みを実行できるように、ブローカを実行するユーザーは、`imqdbmgr` コマンドを使用して組み込みデータベースを作成したユーザーと同一でなければなりません (316 ページの「データベース管理ユーティリティ (`imqdbmgr`)」を参照)。

## JDBC 関連のブローカの設定プロパティ

ブローカのインスタンス設定ファイルは、その設定ファイルが関連付けられているブローカインスタンスの名前 (`instanceName`) によって識別されたディレクトリに書き込まれます (付録 A 「Message Queue データの場所」を参照)。

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

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

表 B-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する各ブローカインスタンスのインスタンス設定ファイル (`config.properties`) に、これらのプロパティを設定します。

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

正確な SQL 構文に関してはデータベースベンダー間で互換性がないため、必ず使用中のデータベースのベンダーが提供している相応するマニュアルを確認し、それによって表 B-1 のプロパティを調整してください。たとえば、PointBase データベースの場合は、IMQMSG35 テーブルの MSG 列で許容される最大の長さを調整する必要が生じる場合があります (`imq.persist.jdbc.table.IMQMSG35` プロパティを参照)。

表 B-1 には、PointBase DBMS の例で指定する値が含まれています。

表 B-1 JDBC 関連プロパティ

プロパティ名	説明
<code>imq.persist.store</code>	<p>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</p> <p>例:</p> <p><code>jdbc</code></p>
<code>imq.persist.jdbc.brokerid</code> (省略可能)	<p>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</p> <p>例: PointBase 組み込みバージョンの場合は不要</p>
<code>imq.persist.jdbc.driver</code>	<p>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</p> <p>例:</p> <p><code>com.pointbase.jdbc.jdbcUniversalDriver</code></p>
<code>imq.persist.jdbc.opendburl</code>	<p>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</p> <p>例:</p> <p><code>jdbc:pointbase:embedded:dbName; database.home= .../instances/instanceName/dbstore</code></p>
<code>imq.persist.jdbc.createdburl</code> (省略可能)	<p>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。<code>imqdbmgr</code> 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</p> <p>例:</p> <p><code>jdbc:pointbase:embedded:dbName;new, database.home= .../instances/instanceName/dbstore</code></p>
<code>imq.persist.jdbc.closedburl</code> (省略可能)	<p>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</p> <p>例: PointBase の場合は不要</p>

表 B-1 JDBC 関連プロパティ ( 続き )

プロパティ名	説明
imq.persist.jdbc.user ( 省略可能 )	<p>必要に応じて、データベースコネクションを開くときに使用するユーザー名を指定する。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</p> <pre>imqbrokerd -dbuser および imqdbmgr -u</pre>
imq.persist.jdbc.needpassword ( 省略可能 )	<p>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 <code>true</code> 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</p> <pre>imqbrokerd -dbpassword imqdbmgr -p</pre> <p>コマンド行オプション、またはパスファイル (236 ページの「<a href="#">passfile の使用</a>」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</p>
imq.persist.jdbc.password ( 省略可能 )	<p>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。このプロパティは <code>passfile</code> 内だけで指定可能 (236 ページの「<a href="#">passfile の使用</a>」を参照)。</p> <p>さまざまな方法でパスワードを指定できる。もっとも安全な方法は、ブローカプロンプトでパスワードを入力することである。安全性は低くなるが、パスファイルを使用してパスファイルを読み取り保護することもできる。安全性はもっとも低くなるが、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こともできる</p> <pre>imqbrokerd -dbpassword imqdbmgr -p</pre>
imq.persist.jdbc.table.IMQSV35	<p>バージョンテーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (STOREVERSION INTEGER NOT NULL, BROKERID VARCHAR(100))</pre>
imq.persist.jdbc.table.IMQCCREC35	<p>設定変更レコードテーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (RECORDTIME BIGINT NOT NULL, RECORD BLOB(10k))</pre>

表 B-1 JDBC 関連プロパティ ( 続き )

プロパティ名	説明
imq.persist.jdbc.table.IMQDEST35	<p>送信先テーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (DID VARCHAR(100) NOT NULL, DEST BLOB(10k), primary key(DID))</pre>
imq.persist.jdbc.table.IMQINT35	<p>配信対象テーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (CUID BIGINT NOT NULL, INTEREST BLOB(10k), primary key(CUID))</pre>
imq.persist.jdbc.table.IMQMSG35	<p>メッセージテーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (MID VARCHAR(100) NOT NULL, DID VARCHAR(100), MSGSIZE BIGINT, MSG BLOB(1m), primary key(MID))</pre> <p>MSG 列のデフォルトの最大長は、1 M バイト(1m)。メッセージがこの長さより長くなると予想される場合は、それに応じて長さを設定する。テーブルが既に作成されている場合は、変更を加えるためにテーブルを作成し直す必要がある</p>
imq.persist.jdbc.table.IMQPROPS35	<p>プロパティテーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (PROPNAME VARCHAR(100) NOT NULL, PROPVALUE BLOB(10k), primary key(PROPNAME))</pre>
imq.persist.jdbc.table.IMQILIST35	<p>配信対象の状態テーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (MID VARCHAR(100) NOT NULL, CUID BIGINT, DID VARCHAR(100), STATE INTEGER, primary key(MID, CUID))</pre>
imq.persist.jdbc.table.IMQTXN35	<p>トランザクションテーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (TUID BIGINT NOT NULL, STATE INTEGER, TSTATEOBJ BLOB(10K), primary key(TUID))</pre>



表 B-1 JDBC 関連プロパティ ( 続き )

プロパティ名	説明
<code>imq.persist.jdbc.table.IMQTACK35</code>	<p>トランザクション通知テーブルを作成するための SQL コマンド</p> <p>例:</p> <pre>CREATE TABLE \${name} (TUID BIGINT NOT NULL, TXNACK BLOB(10k))</pre>

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

例:

PointBase の組み込みデータベース例の場合、データベースコネクション URL ( 表 B-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、PointBase システムディレクトリを定義することができます。

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

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:pointbase:embedded:dbName;new
```

および

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

# データベース管理ユーティリティ (imqdbmgr)

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

この節では、imqdbmgr コマンドの基本構文、サブコマンドのリスト、imqdbmgr コマンドのオプションの概要について説明します。

## imqdbmgr コマンドの構文

imqdbmgr コマンドの一般的な構文は、次のとおりです。

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

-v、または -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、create サブコマンドは実行されません。

```
imqdbmgr create all -v
```

## imqdbmgr のサブコマンド

データベース管理ユーティリティ (imqdbmgr) には、次の表 B-2 に示すようなサブコマンドが含まれています。

表 B-2 imqdbmgr のサブコマンド	
サブコマンドおよび引数	説明
create all	新しいデータベースと Message Queue の持続ストアのスキーマを作成する。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、Message Queue の持続ストア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	現在の持続ストアのデータベース内に存在する Message Queue のデータベーステーブルを削除する

表 B-2 imqdbmgr のサブコマンド ( 続き )

サブコマンドおよび引数	説明
delete oldtbl	旧バージョンの持続ストアのデータベース内に存在するすべての Message Queue データベーステーブルを削除する。持続ストアが Message Queue の現在のバージョンへ自動的に移行された後に使用される
recreate tbl	現在の持続ストアのデータベース内に存在する Message Queue のデータベーステーブルを削除した後、Message Queue の持続ストアのスキーマを作成し直す
reset lck	その他のプロセスが持続ストアのデータベースを使用できるようにロックをリセットする

## imqdbmgr コマンドのオプションの概要

表 B-3 に imqdbmgr コマンドのオプションを一覧表示します。

表 B-3 imqdbmgr のオプション

オプション	説明
-Dproperty=value	指定したプロパティを指定した値に設定する。
-b instanceName	ブローカインスタンス名を指定し、対応するインスタンス設定ファイルを使用する
-h	使用方法に関するヘルプを表示する コマンド行ではそれ以外のことは実行されない
-p password	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ー名を指定する
-v	バージョン情報を表示する コマンド行ではそれ以外のことは実行されない

データベース管理ユーティリティ (imqdbmgr)

# HTTP/HTTPS サポート (Enterprise Edition)

Message Queue, Enterprise Edition (35 ページの「製品エディション」を参照) は、HTTP コネクションと HTTPS コネクションの両方をサポートします。HTTPS は、Secure Socket Layer (SSL) 通信用コネクションを介した HTTP です。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有効にするために使用されるアーキテクチャについて説明し、クライアントが Message Queue メッセージングに HTTP ベースのコネクションを使用するために必要となる設定の手順を示します。

---

注	HTTP/HTTPS サポートは、Java クライアントに対応しており、C クライアントには対応していません。
---	---

---

## HTTP/HTTPS サポートのアーキテクチャ

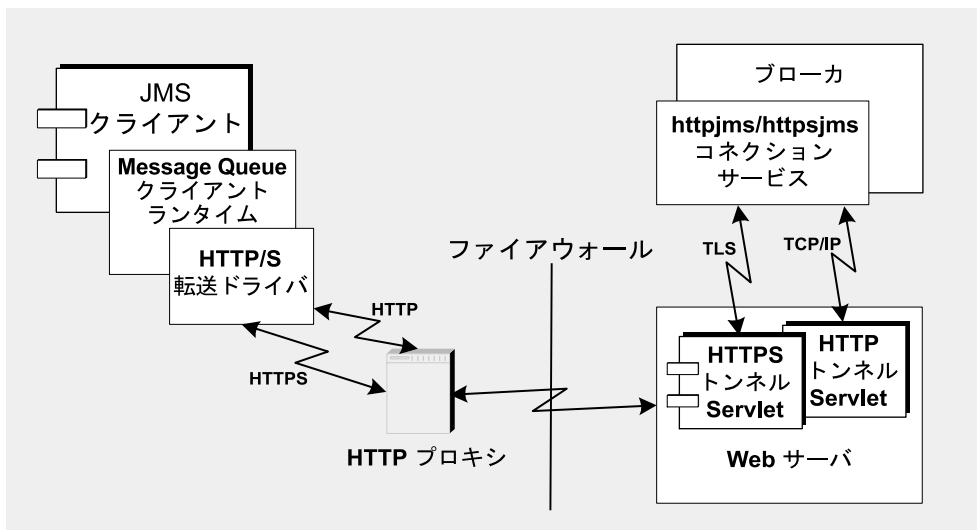
Message Queue メッセージングは、HTTP/HTTPS コネクションで実行できます。HTTP/HTTPS コネクションは、通常ファイアウォールを通して許可されるため、ファイアウォールによってブローカからクライアントアプリケーションを分離できません。

320 ページの図 C-1 に、HTTP/HTTPS サポートの提供に関連する主なコンポーネントを示します。

- クライアント側では、HTTP または HTTPS の転送ドライバが Message Queue メッセージを HTTP 要求にカプセル化し、これらの要求を正しい手順で Web サーバに確実に送信する

- クライアントは、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、[326 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 Message Queue 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 C-1 HTTP/HTTPS サポートのアーキテクチャ



- ブローカ側では、httpjms または httpsjms コネクションサービスが、対応するトンネルサーブレットから送られてくるメッセージを開いて、非多重化する
- Web サーバで障害が生じて再起動すれば、すべてのコネクションが復元され、クライアントへの影響はありません。ブローカに障害が生じ再起動された場合は、例外がスローされ、クライアントはそれぞれのコネクションを再確立する必要があります。発生するのはまれですが、Web サーバとブローカの両方に障害が生じ、ブローカが再起動されなかった場合は、Web サーバはクライアントコネクションを復元し、引き続きブローカコネクションを待機しますがクライアントには通知しません。この状況を避けるために、常に、ブローカを再起動してください。

図 C-1 からわかるとおり、HTTP と HTTPS サポートのアーキテクチャは非常に良く似ています。主な相違点は、HTTPS (httpsjms コネクションサービス) の場合、トンネルサーブレットにクライアントアプリケーションとブローカの両方への安全なコネクションがあることです。

ブローカへの安全なコネクションは、SSL に対応したトンネルサーブレット、つまり Message Queue の HTTPS トンネルサーブレットを通して提供されます。このトンネルサーブレットが、コネクションを要求しているブローカに自己署名型証明書を渡します。ブローカは証明書を使用して、HTTPS トンネルサーブレットへの暗号化されたコネクションを設定します。このコネクションが確立されると、クライアントアプリケーションとトンネルサーブレット間の安全なコネクションについて、クライアントアプリケーションと Web サーバがネゴシエーションを行います。

## HTTP サポートの有効化

次に、HTTP サポートを有効化するのに必要な手順を説明します。

### ► HTTP サポートを有効にするには

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

Message Queue トンネルサーブレットを配置するには、ホスト Web サーバへアクセス可能な適切な jar ファイルを作成し、起動時にサーブレットを読み込むように Web サーバを設定して、サーブレットの URL のコンテキストルート部分を指定します。

トンネルサーブレットの jar ファイル (imqservlet.jar) には、HTTP トンネルサーブレットが必要とするすべてのクラスが含まれます。このファイルは、オペレーティングシステムに応じて該当するディレクトリに格納されています ( [付録 A 「Message Queue データの場所」](#) を参照 )。

Servlet 2.x をサポートする Web サーバは、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。

また、起動時にこのサーブレットを読み込むように Web サーバを設定する必要があります。サーブレットの URL のコンテキストルート部分を指定する必要がある場合があります ( [326 ページの「例 1: HTTP トンネルサーブレットを Sun Java System Web サーバに配置する」](#) を参照 )。

パフォーマンスを向上させるために、Web サーバ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の WAR ファイル (imqhttp.war) は、オペレーティングシステムに応じて、.jar、.war、.rar の各ファイルを含むディレクトリに配置されています ( [付録 A 「Message Queue データの場所」](#) を参照 )。

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な基本的設定情報などの配置記述子が含まれています。Web サーバによっては、サーブレットの URL のコンテキストルート部分を指定しなければならない場合があります ( [330 ページの「例 2: HTTP トンネルサーブレットを Sun Java System Application Server 7.0 に配置する」](#) を参照 )。



## 手順 2: httpjms コネクションサービスを設定する

デフォルトでは、HTTP サポートはブローカに対してアクティブになっていないため、httpjms コネクションサービスをアクティブにするようブローカを再設定する必要があります。設定し直すと、[141 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アクティブにするには

1. ブローカのインスタンス設定ファイルを開きます。

インスタンス設定ファイルは、その設定ファイルが関連付けられているブローカインスタンスの名前 (*instanceName*) によって識別されたディレクトリに書き込まれます ( [付録 A「Message Queue データの場所」](#) を参照 )。

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

2. httpjms の値を imq.service.activelist プロパティに追加します。

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

ブローカは、起動時に Web サーバ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、[323 ページの表 C-1](#) を参照してください。

表 C-1 httpjms コネクションサービスのプロパティ

プロパティ名	説明
imq.httpjms.http.servletHost	必要に応じてこの値を変更し、HTTP トンネルサーブレットを実行するホストの名前 ( ホスト名または IP アドレス ) を指定する。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になる デフォルト値: localhost
imq.httpjms.http.servletPort	この値を変更して、ブローカが HTTP トンネルサーブレットにアクセスするために使用するポート番号を指定する。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 デフォルト値: 7675

表 C-1        httpjms コネクションサービスのプロパティ ( 続き )

プロパティ名	説明
imq.httpjms.http.pullPeriod	メッセージをブローカから取り出すために各クライアントランタイムが出す HTTP 要求の間隔を秒単位で指定する。このプロパティはブローカで設定され、クライアントランタイムに伝達される点に注意。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 デフォルト値: -1
imq.httpjms.http.connectionTimeout	クライアントランタイムが例外をスローする前に HTTP トンネルサーブレットからの応答を待機する時間を秒単位で指定する。このプロパティはブローカで設定され、クライアントランタイムに伝達される点に注意。このプロパティは、ブローカが HTTP トンネルサーブレットと通信した後、コネクションを開放するまでの時間も指定する。ブローカとトンネルサーブレットは、HTTP サーブレットへアクセス中のクライアントが異常終了したかどうかを確認する手段を持っていないため、この場合はタイムアウトが必須となる デフォルト値: 60

## 手順 3: HTTP コネクションを設定する

クライアントアプリケーションは、設定済みのコネクションファクトリ管理対象オブジェクトを適切に使用して、ブローカへの HTTP コネクションを確立する必要があります。この節では、HTTP コネクション設定の問題点を説明します。

### コネクションファクトリを設定する

HTTP サポートを有効にするには、コネクションファクトリの `imqAddressList` 属性を HTTP トンネルサブレット URL に設定する必要があります。HTTP トンネルサブレット URL の一般的な構文は次のとおりです。

```
http://hostName:port/contextRoot/tunnel
```

`hostName:port` は、HTTP トンネルサブレットをホスティングする Web サーバの名前とポートです。`contextRoot` は、Web サーバにトンネルサブレットを配置したときに設定したパスです。

コネクションファクトリ属性の概要と、特に `imqAddressList` 属性の詳細については、『[Message Queue Java Client Developer's Guide](#)』を参照してください。

コネクションファクトリ属性の設定は、次のいずれかの方法で行います。

- `-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 ([206 ページの「コネクションファクトリの追加」](#)を参照)、管理コンソール (`imqadmin`) を使用してコネクションファクトリ管理対象オブジェクト作成時に属性を設定する
- クライアントを起動するコマンドに `-D` オプションを使用する (『[Message Queue Java Client Developer's Guide](#)』を参照)
- プログラム的にクライアントコードに API 呼び出しを作成したあと、これを使用してコネクションファクトリの属性を設定する (『[Message Queue Java Client Developer's Guide](#)』を参照)

### 1 つのサ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 トンネルサブレットインスタンスを共有できます。複数のブローカインスタンスが 1 つのトンネルサブレットを共有している場合は、次に示すとおり、`imqAddressList` コネクションファクトリ属性を設定する必要があります。

```
http://hostName:port/contextRoot/tunnel?ServerName=bkrHostName:instanceName
```

`bkrHostName` の部分にはブローカインスタンスのホスト名が入り、`instanceName` の部分にはクライアントにアクセスさせる特定のブローカインスタンス名が入ります。

*bkrHostName* と *instanceName* に正しい文字列を入力したことを確認するには、ブラウザからサーブレット URL にアクセスして、HTTP トンネルサーブレットの状態レポートを生成します。レポートでは、サーブレットがアクセスしているすべてのブローカが次のように一覧表示されます。

```
HTTP tunnel servlet ready.  
Servlet Start Time :Thu May 30 01:08:18 PDT 2002  
Accepting TCP connections from brokers on port : 7675  
Total available brokers = 2  
Broker List :  
    jpgserv:broker2  
    cochin:broker1
```

## HTTP プロキシを使用する

HTTP プロキシを使用して HTTP トンネルサーブレットにアクセスする場合、次の設定を行います。

- `http.proxyHost` システムプロパティをプロキシサーバ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 1: HTTP トンネルサーブレットを Sun Java System Web サーバに配置する

ここでは、HTTP トンネルサーブレットを Sun Java System Web Server に jar ファイルおよび WAR ファイルとして配置する両方の方法を説明します。どちらを使用するかは、Sun Java System Web Server のバージョンによって決まります。Servlet 2.2 またはそれ以降がサポートされていない場合は、WAR ファイルの配置を行えません。

### jar ファイルとして配置する

次の手順は、ブラウザベースの管理 GUI を使用した Sun Java System Web Server 6.1 への配置を説明しています。この方法では、通常次の手順を実行します。

1. サーブレットを追加する
2. サーブレットの仮想パスを設定する
3. サーブレットを読み込む

4. サーブレットアクセスログを無効にする

次の項で、これらの手順について説明します。Web ブラウザを使用してサーブレットの URL にアクセスすると、HTTP トンネルサーブレットが問題なく配置されたことが確認できます。ステータス情報も表示されます。

サーブレットを追加する

➤ トンネルサーブレットを追加するには

- 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.HttpTunnelServlet
```

- 5. 「Servlet Classpath」フィールドに `imqservlet.jar` への絶対パスを入力します。たとえば、次のように指定します。

```
/usr/share/lib/imq/imqservlet.jar (Solaris の場合 )  
  
/opt/imq/lib/imqservlet.jar (Linux の場合 )  
  
IMQ_HOME/lib/imqservlet.jar (Windows の場合 )
```

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

表 C-2 HTTP トンネルサーブレット jar ファイルの配置に使用するサーブレット引数

引数	デフォルト値	参照先
servletHost	all hosts	323 ページの表 C-1 を参照
servletPort	7675	323 ページの表 C-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。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 (323 ページの表 C-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。

仮想パスは、トンネルサーブレット URL の `/contextRoot/tunnel` 部分です。

`http://hostName:port/contextRoot/tunnel`

たとえば、`contextRoot` を `img` に設定すると、「Virtual Path」フィールドは次のようになります。

`/img/tunnel`

4. 「Servlet Name」フィールドに 327 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、327 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 Java System 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. 「WAR File On and WAR File Path」フィールドでは、imghttp.war ファイルを指す適切な値を選択します。このファイルはオペレーティングシステムに応じて異なるディレクトリに格納されています ( 付録 A 「Message Queue データの場所」を参照 )。
8. 「Application URI」フィールドにパスを入力します。  
「Application URI」フィールドの値は、トンネルサーブレット URL の /contextRoot 部分です。  
`http://hostName:port/contextRoot/tunnel`  
たとえば、contextRoot を img に設定すると、「Application URI」フィールドは次のようになります。  
`/img`
9. サーブレットを配置するインストールディレクトリのパス ( 通常は、Sun Java System Web Server インストールルートの任意の場所 ) 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/contextRoot/tunnel`

クライアントはこの URL を使用して、HTTP コネクションを使用しているメッセージサービスに接続できます。

## 例 2: HTTP トンネルサーブレットを Sun Java System Application Server 7.0 に配置する

この節では、HTTP トンネルサーブレットを WAR ファイルとして Sun Java System Application Server 7.0 に配置する方法を説明します。

2 段階の手順が必要です。

- Application Server 7.0 配置ツールを使用して HTTP トンネルサーブレットを配置する
- アプリケーションサーバインスタンスの `server.policy` ファイルを変更する

### 配置ツールを使用する

#### ► HTTP トンネルサーブレットを Application Server 7.0 環境に配置するには

1. Web ベースの管理 GUI で、次を選択します。

「App Server」 > 「Instances」 > 「server1」 > 「Applications」 > 「Web Applications」

2. 「Deploy」 ボタンをクリックします。

3. 「File Path:」 テキストフィールドに、HTTP トンネルサーブレットの WAR ファイル (`imghttp.war`) の場所を入力します。

`imghttp.war` ファイルの場所は、使用中のオペレーティングシステムによって異なります (付録 A 「[Message Queue データの場所](#)」を参照)。

4. 「OK」 をクリックします。

5. 次の画面で、「Context Root」 テキストフィールドの値を設定します。

「Context Root」 フィールドの値は、トンネルサーブレット URL の `/contextRoot` 部分です。

```
http://hostName:port/contextRoot/tunnel
```

たとえば、「Context Root」 フィールドは次のように設定できます。

```
/img
```

6. 「OK」 をクリックします。

次の画面は、トンネルサーブレットが正常に配置され、デフォルトで有効になっており、この場合は、次の場所に格納されていることを示しています。

```
/var/opt/SUNWappserver7/domains/domain1/server1/applications/  
j2ee-modules/imghttp_1
```

サーブレットは次のアドレスで利用可能となります。



```
http://hostName:port/contextRoot/tunnel
```

クライアントはこの URL を使用して、HTTP コネクションを使用しているメッセージサービスに接続できます。

## server.policy ファイルを変更する

Application Server 7.0 は、変更されないかぎり、強制的にデフォルトのセキュリティポリシーセットを適用し、HTTP トンネルサブレットが Message Queue ブローカからのコネクションを受け入れるのを阻止します。

各アプリケーションサーバインスタンスには、セキュリティポリシーまたはルールを含むファイルがあります。たとえば、Solaris 上の server1 インスタンスのこのファイルは次の場所にあります。

```
/var/opt/SUNWappserver7/domains/domain1/server1/config/
server.policy
```

トンネルサブレットに Message Queue ブローカからのコネクションを受け入れさせるには、このファイルにエントリを追加する必要があります。

### ► アプリケーションサーバの server.policy ファイルを変更するには

1. server.policy ファイルを開きます。
2. 次のエントリを追加します。

```
grant codeBase
"file:/var/opt/SUNWappserver7/domains/domain1/server1/
    applications/j2ee-modules/imqhttp_1/-"
{
    permission java.net.SocketPermission "*",
        "connect,accept,resolve";
};
```

# HTTPS サポートの有効化

次に、HTTPS サポートを有効化するのに必要な手順を説明します。この手順は、[321 ページの「HTTP サポートの有効化」](#)の手順とほとんど同じですが、さらに SSL 証明書の生成とアクセスに必要な手順も追加されています。

## ► HTTPS サポートを有効にするには

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

Message Queue 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、自己署名型の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。

`imkeytool` 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。

```
imkeytool -servlet keystore_location
```

ユーティリティが、必要な情報を要求します。Unix システムでは、キーストアを作成するアクセス権を取得するためにスーパーユーザー (`root`) として `imkeytool` を実行する必要があります。

`imkeytool` は、まず、キーストアに対するパスワードの入力を要求します。次に一部の組織情報の入力、続いて確認を要求します。確認が取れると、キーの組み合わせを生成している間、このコマンドは停止します。その後、特定のキーの組み合わせをロックするためのパスワード (キーパスワード) の入力を要求してくるので、**Return** キーを押します。これで、キーパスワードに、キーストアと同じパスワードが設定されます。

---

**注**            設定したパスワードを忘れないでください。あとでトンネルサーブレットがキーストアを開くために、そのパスワードを入力する必要があります。

---

imqkeytool を実行すると、JDK keytool ユーティリティが実行されて、自己署名型証明書が生成されます。生成された証明書は、*keystore\_location* 引数で指定される場所にある、Message Queue のキーストアファイルに配置されます。キーストアは、JDK1.2 keytool でサポートされているのと同じキーストアの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 (333 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

Message Queue トンネルサーブレットを配置するには、ホスト Web サーバへアクセス可能な適切な jar ファイルを作成し、起動時にサーブレットを読み込むように Web サーバを設定して、サーブレットの URL のコンテキストルート部分を指定します。

トンネルサーブレットの jar ファイル (imqservlet.jar) には、HTTPS トンネルサーブレットが必要とするすべてのクラスが含まれます。このファイルは、オペレーティングシステムに応じて該当するディレクトリに格納されています ( [付録 A 「Message Queue データの場所」](#) を参照 )。

Servlet 2.x をサポートする Web サーバは、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、`imgServlet.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、`imgServlet.jar` ファイルのコピーを置く必要があります。

また、起動時にこのサーブレットを読み込むように Web サーバを設定する必要があります。サーブレットの URL のコンテキストルート部分を指定する必要がある場合があります (339 ページの「例 3: HTTPS トンネルサーブレットを Sun Java System Web サーバに配置する」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合とパスワードを 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。332 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の WAR ファイル (`imghttps.war`) は使用中のオペレーティングシステムに応じて異なるディレクトリに格納されています (付録 A 「Message Queue データの場所」を参照)。

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な基本的設定情報などの配置記述子が含まれています。Web サーバによっては、サーブレットの URL のコンテキストルート部分を指定しなければならない場合があります (344 ページの「例 4: HTTPS トンネルサーブレットを Sun Java System Application Server 7.0 に配置する」を参照)。

ただし、`imghttps.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 (332 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、`imghttps.war` ファイルを配置する前に、トンネルサーブレットの配置記述子 (XML ファイル) を編集し、キーストアの場所を指定する必要があります。

## 手順 3: httpsjms コネクションサービスを設定する

デフォルトでは、HTTPS サポートはブローカに対してアクティブになっていないため、httpsjms コネクションサービスをアクティブにするようブローカを再設定する必要があります。設定し直すと、[141 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アクティブにするには**

1. ブローカのインスタンス設定ファイルを開きます。

インスタンス設定ファイルは、その設定ファイルが関連付けられているブローカインスタンスの名前 (*instanceName*) によって識別されたディレクトリに書き込まれます ( [付録 A「Message Queue データの場所」](#) を参照 )。

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

2. httpsjms の値を imq.service.activelist=jms,admin,httpsjms

```
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 C-3](#) を参照してください。

**表 C-3**      httpsjms コネクションサービスのプロパティ

プロパティ名	説明
imq.httpsjms.https.servletHost	必要に応じて、この値を変更し、HTTPS トンネルサーブレットを実行するホストの名前 ( ホスト名または IP アドレス ) を指定する。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になる デフォルト値: localhost
imq.httpsjms.https.servletPort	この値を変更して、ブローカが HTTPS トンネルサーブレットにアクセスするために使用するポート番号を指定する。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 デフォルト値: 7674

表 C-3        httpsjms コネクションサービスのプロパティ ( 続き )

プロパティ名	説明
imq.httpsjms.https.pullPeriod	メッセージをブローカから取り出すために各クライアントが出す HTTP 要求の間隔を秒単位で指定する。このプロパティはブローカで設定され、クライアントランタイムに伝達される点に注意。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 デフォルト値 : -1
imq.httpsjms.https.connectionTimeout	クライアントランタイムが例外をスローする前に HTTPS トンネルサーブレットからの応答を待機する時間を秒単位で指定する。このプロパティはブローカで設定され、クライアントランタイムに伝達される点に注意。このプロパティは、ブローカが HTTPS トンネルサーブレットと通信した後、コネクションを開放するまでの時間も指定する。ブローカとトンネルサーブレットは、HTTPS サーブレットへアクセス中のクライアントが異常終了したかどうかを確認する手段を持っていないため、この場合はタイムアウトが必要となる デフォルト値 : 60

## 手順 4: HTTPS コネクションを設定する

クライアントアプリケーションは、適切に設定されたコネクションファクトリ管理対象オブジェクトを使用して、ブローカへの HTTPS コネクションを確立する必要があります。

ただし、クライアントは Java Secure Socket Extension (JSSE) で提供される SSL ライブラリへもアクセスし、**root** 証明書を持つ必要もあります。SSL ライブラリは、JDK 1.4 に付属しています。それ以前の JDK バージョンを使用している場合は、「[JSSE を設定する](#)」を参照するか、あるいは「[root 証明書をインポートする](#)」に進みます。

これらの問題点が解決すると、HTTPS コネクションの設定に進みます。

### JSSE を設定する

#### ► JSSE を設定するには

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
```

ここで、*n* には、セキュリティプロバイダパッケージが次に利用可能な優先順位を指定します。

3. JDK 1.4 を使用していない場合は、-D オプションをクライアントアプリケーションを起動するコマンドに使用して、次の JSSE プロパティを設定します。

```
java.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol
```

### root 証明書をインポートする

Web サーバ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を設定する](#)」を参照します。

証明書が *cert\_file* に保存され、*trust\_store\_file* がキーストアであると仮定して、次のコマンドを実行します。

```
JRE_HOME/bin/keytool -import -trustcacerts
-alias alias_for_certificate -file cert_file
-keystore trust_store_file
```

次の質問に YES と答えます。Trust this certificate?

クライアントアプリケーションを起動するコマンドに `-D` オプションを使用して、次の JSSE プロパティを指定する必要もあります。

```
javax.net.ssl.trustStore=trust_store_file
javax.net.ssl.trustStorePassword=trust_store_passwd
```

## コネクションファクトリを設定する

HTTPS サポートを有効にするには、コネクションファクトリの `imqAddressList` 属性を HTTPS トンネルサーブレット URL に設定する必要があります。HTTPS トンネルサーブレット URL の一般的な構文は次のとおりです。

```
https://hostName:port/contextRoot/tunnel
```

`hostName:port` は、HTTPS トンネルサーブレットをホスティングする Web サーバの名前とポートです。`contextRoot` は、Web サーバにトンネルサーブレットを配置したときに設定したパスです。

コネクションファクトリ属性の全般と `imqAddressList` 属性の詳細については、『[Message Queue Java Client Developer's Guide](#)』を参照してください。

コネクションファクトリ属性の設定は、次のいずれかの方法で行います。

- コネクションファクトリ管理対象オブジェクトを作成する `imqobjmgr` コマンドで、`-o` オプションを使用するか (206 ページの「[コネクションファクトリの追加](#)」を参照)、管理コンソール (`imqadmin`) を使用してコネクションファクトリ管理対象オブジェクト作成時に属性を設定する
- 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` オプションを使用する (『[Message Queue Java Client Developer's Guide](#)』を参照)
- クライアントアプリケーションのプログラムでコネクションファクトリを作成してから、API 呼び出しを使用してコネクションファクトリの属性を設定する (『[Message Queue Java Client Developer's Guide](#)』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。複数のブローカインスタンスが 1 つのトンネルサーブレットを共有している場合は、次に示すとおり、`imqAddressList` コネクションファクトリ属性を設定する必要があります。

```
https://hostName:port/contextRoot/tunnel?ServerName=bkrHostName:instanceName
```

`bkrHostName` の部分にはブローカインスタンスのホスト名が入り、`instanceName` の部分にはクライアントにアクセスさせる特定のブローカインスタンス名が入ります。



*bkrHostName* と *instanceName* に正しい文字列を入力したことを確認するには、ブラウザからサーブレット URL にアクセスして、HTTPS トンネルサーブレットの状態レポートを生成します。レポートでは、サーブレットがアクセスしているすべてのブローカが次のように一覧表示されます。

```
HTTPS tunnel servlet ready.  
Servlet Start Time :Thu May 30 01:08:18 PDT 2002  
Accepting TCP connections from brokers on port : 7674  
Total available brokers = 2  
Broker List :  
    jpgserv:broker2  
    cochin:broker1
```

## HTTP プロキシを使用する

HTTP プロキシを使用して HTTPS トンネルサーブレットにアクセスする場合、次の設定を行います。

- `http.proxyHost` システムプロパティをプロキシサーバ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 3: HTTPS トンネルサーブレットを Sun Java System Web サーバに配置する

ここでは、HTTPS トンネルサーブレットを Sun Java System Web Server に jar ファイルおよび WAR ファイルとして配置する両方の方法を説明します。どちらを使用するかは、Sun Java System Web Server のバージョンによって決まります。Servlet 2.2 またはそれ以降がサポートされていない場合は、WAR ファイルの配置を行えません。

### jar ファイルとして配置する

次の手順は、ブラウザベースの管理 GUI を使用した Sun Java System Web Server 6.1 への配置を説明しています。この方法では、通常次の手順を実行します。

1. サーブレットを追加する
2. サーブレットの仮想パスを設定する
3. サーブレットを読み込む

4. サーブレットアクセスログを無効にする

次の項で、これらの手順について説明します。Web ブラウザを使用してサーブレットの URL にアクセスすると、HTTPS トンネルサーブレットが問題なく配置されたことが確認できます。ステータス情報も表示されます。

サーブレットを追加する

➤ トンネルサーブレットを追加するには

- 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。  
  
`/usr/share/lib/imq/imqservlet.jar` (Solaris の場合)  
  
`/opt/imq/lib/imqservlet.jar` (Linux の場合)  
  
`IMQ_HOME/lib/imqservlet.jar` (Windows の場合)
- 6. 「Servlet args」フィールドに、表 C-4 に示す必要なオプションの引数を入力します。

表 C-4      HTTPS トンネルサーブレット jar ファイルの配置に使用するサーブレット引数

引数	デフォルト値	必須 / オプション	関連項目
keystoreLocation	なし	必須	232 ページの表 8-8
keystorePassword	なし	必須	232 ページの表 8-8
servletHost	all hosts	オプション	335 ページの表 C-3
servletPort	7674	オプション	335 ページの表 C-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します。ただしその場合、ブローカ設定プロパティを設定する必要があります (335 ページの表 C-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。

仮想パスは、トンネルサーブレット URL の `/contextRoot/tunnel` 部分です。

```
https://hostName:port/contextRoot/tunnel
```

たとえば、`contextRoot` を `imq` に設定すると、「Virtual Path」フィールドは次のようになります。

```
/imq/tunnel
```

4. 「Servlet Name」フィールドに 340 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、340 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 Java System 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。これを確実にする一番簡単な方法は、jsse.jar、jnet.jar、および jcert.jar を IWS60\_TOPDIR/bin/https/jre/lib/ext にコピーすることです。

HTTPS トンネルサーブレットを配置する前に、配置記述子がキーストアファイルの配置場所を指し、キーストアパスワードを指定するように変更する必要があります。

### ► 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 /usr/share/lib/imq/imqhttps.war /tmp (Solaris の場合)
```

```
cp /opt/imq/lib/imqhttps.war /tmp (Linux の場合)
```

```
cp IMQ_HOME/lib/imqhttps.war /tmp (Windows の場合)
```

2. 一時ディレクトリを現在のディレクトリにします。

```
$ cd /tmp
```

3. WAR ファイルの内容を抽出します。

```
$ jar xvf imqhttps.war
```

4. WAR ファイルの配置記述子を一覧表示します。

```
$ ls -l WEB-INF/web.xml
```

5. web.xml ファイルを編集して、keystoreLocation と keystorePassword という引数に正しい値を設定します。必要に応じて serverPort と serverHost の引数も設定します。

6. WAR ファイルの内容を設定し直します。

```
$ jar uvf imqhttps.war WEB-INF/web.xml
```

これで修正済みの `imghttps.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、`imghttps.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 `imghttps.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 (342 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」フィールドにパスを入力します。  
「Application URI」フィールドの値は、トンネルサーブレット URL の `/contextRoot` 部分です。

`https://hostName:port/contextRoot/tunnel`

たとえば、`contextRoot` を `img` に設定すると、「Application URI」フィールドは次のようになります。

`/img`

9. サーブレットを配置するインストールディレクトリのパス ( 通常は、Sun Java System Web Server インストールルートの中の場所 ) 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/img/tunnel`

クライアントはこの URL を使用して、安全な HTTPS コネクションを使用しているメッセージサービスに接続できます。

## 例 4: HTTPS トンネルサーブレットを Sun Java System Application Server 7.0 に配置する

この節では、HTTPS トンネルサーブレットを WAR ファイルとして Sun Java System Application Server 7.0 に配置する方法を説明します。

2 段階の手順が必要です。

- Application Server 7.0 配置ツールを使用して HTTPS トンネルサーブレットを配置する
- アプリケーションサーバインスタンスの `server.policy` ファイルを変更する

### 配置ツールを使用する

#### ► HTTPS トンネルサーブレットを Application Server 7.0 環境に配置するには

1. Web ベースの管理 GUI で、次を選択します。

「App Server」>「Instances」>「server1」>「Applications」>「Web Applications」

2. 「Deploy」ボタンをクリックします。

3. 「File Path」テキストフィールドに、HTTPS トンネルサーブレットの WAR ファイル (`imqhttps.war`) の場所を入力します。

`imqhttps.war` ファイルの場所は、使用中のオペレーティングシステムによって異なります (付録 A 「[Message Queue データの場所](#)」を参照)。

4. 「OK」をクリックします。

5. 次の画面で、「Context Root」テキストフィールドの値を設定します。

「Context Root」フィールドの値は、トンネルサーブレット URL の `/contextRoot` 部分です。

```
https://hostName:port/contextRoot/tunnel
```

たとえば、「Context Root」フィールドは次のように設定できます。

```
/imq
```

6. 「OK」をクリックします。

次の画面は、トンネルサーブレットが正常に配置され、デフォルトで有効になっており、この場合は、次の場所に格納されていることを示しています。

```
/var/opt/SUNWappserver7/domains/domain1/server1/applications/  
j2ee-modules/imqhttps_1
```

サーブレットは次のアドレスで利用可能となります。

```
https://hostName:port/contextRoot/tunnel
```

クライアントはこの URL を使用して、HTTPS コネクションを使用しているメッセージサービスに接続できます。

## server.policy ファイルを変更する

Application Server 7.0 は、変更されないかぎり、強制的にデフォルトのセキュリティポリシーセットを適用し、HTTPS トンネルサーブレットが Message Queue ブローカからのコネクションを受け入れるのを阻止します。

各アプリケーションサーバインスタンスには、セキュリティポリシーまたはルールを含むファイルがあります。たとえば、Solaris 上の server1 インスタンスのこのファイルは次の場所にあります。

```
/var/opt/SUNWappserver7/domains/domain1/server1/config/
server.policy
```

トンネルサーブレットに Message Queue ブローカからのコネクションを受け入れさせるには、このファイルにエントリを追加する必要があります。

### ► アプリケーションサーバの server.policy ファイルを変更するには

1. server.policy ファイルを開きます。
2. 次のエントリを追加します。

```
grant codeBase
"file:/var/opt/SUNWappserver7/domains/domain1/server1/
    applications/j2ee-modules/imqhttps_1/-"
{
    permission java.net.SocketPermission "*",
        "connect,accept,resolve";
};
```





# 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、クエリー、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

Message Queue 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、Message Queue 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。ブローカに起動オプションを渡すには、imqsvcadmin コマンド (349 ページの表 D-2 を参照) で -args 引数を使用し、imqbrokerd コマンド (141 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、クエリー、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。

## imqsvcadmin 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]
```

```
imqsvcadmin -h
```

-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、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。

```
imqsvcadmin query -h
```

## imqsvcadmin のサブコマンド

Message Queue サービス管理ユーティリティ (imqsvcadmin) には、次の表 D-1 に示すようなサブコマンドが含まれています。

表 D-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 D-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、タスクごとに説明した後続の節を参照してください。

表 D-2 imqsvcadmin のオプション

オプション	説明
-h	使用方法に関するヘルプを表示する。コマンド行ではそれ以外のことは実行されない
-javahome <i>path</i>	使用する代替の Java 2 互換のランタイムへのパスを指定する。デフォルトではシステム上のランタイムまたは Message Queue にバンドルされたランタイムを使用する  例:imqsvcadmin -install -javahome d:\jdk1.4
-jrehome <i>path</i>	Java 2 互換の JRE へのパスを指定する  例:imqsvcadmin -install -jrehome d:\jre1.4
-vmargs <i>arg</i> [ <i>arg</i> ]...	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 <i>arg</i> [ <i>arg</i> ]...	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、 <a href="#">141 ページの「ブローカの起動」</a> 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。たとえば、次のように指定する  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ー

ブローカサービスの起動オプションを指定するには、`imqsvcadmin` コマンドの `-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 3.5¥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。

サービス管理ユーティリティ (imqsvcadmin)

## 技術的な注意点

この付録では、次のトピックについての簡単な説明があります。

- システムの時計の設定
- OS で定義されるファイル記述子の制限事項
- 持続データの保護

## システムの時計の設定

Message Queue システムを使用するときには、システムの時計を同期させ、システムの時計を逆戻りさせないように注意してください。

### 同期の推奨

Message Queue システムとやり取りするすべてのホストの時計を同期させることをお勧めします。メッセージの有効期限 (TimeToLive) を使用する場合には、これが特に重要です。ホストの時計が同期していないと、メッセージが配信されないなど、TimeToLive が期待どおりに動作をしません。時計は、すべてのブローカを起動する前に同期させる必要があります。

**Solaris** リモートホストと同期させようとするローカルホスト上で、`rdate` コマンドを実行できます。このコマンドを実行するには、スーパーユーザー (`root`) としてログインする必要があります。たとえば次のコマンドは、ローカルホスト (`Host 2` とする) を、リモートホストである `Host1` と同期させます。

```
# rdate Host1
```

**Linux** このコマンドは **Solaris** と似ていますが、次のように `-s` オプションをつける必要があります。

```
# rdate -s Host1
```

**Windows** ローカルホストをリモートホストと同期させるには、`net` コマンドに `time` サブコマンドを指定して実行します。たとえば次のコマンドは、ローカルホスト (`Host 2` とする) を、リモートホストである `Host1` と同期させます。

```
net time %YHost1 /set
```

## システムの時計を逆戻りさせない

Message Queue ブローカを実行するシステムでは、システムの時計の設定を逆戻りさせないようにする必要があります。Message Queue では、トランザクションや永続サブスクリプションなどの内部オブジェクトを識別するのにタイムスタンプを使用します。システムの時計の設定が逆戻りすると、論理的には重複した内部識別子が生成されてしまう可能性があります。ブローカは、多少のランダム性を見込んで実行時の時計のずれを検出することにより、これを補うようにしています。しかし、ブローカの実行時以外にシステムの時計が大きく逆戻りした場合は、識別子の重複の危険性がわずかながら発生します。

ブローカを実行するシステムでシステムの時計の設定を数秒以上逆戻りさせる必要がある場合は、トランザクションや永続サブスクリプションがないときに行うようにします。あるいは、ブローカの実行時以外にこれを行い、逆戻りさせた分の時間が経ってからブローカを起動します。

ただし理想的な方法は、すべてのブローカを起動する前に時計を同期させ、導入後は適切な技術を使用して、時計が大きく変動しないようにすることです。

## OS で定義されるファイル記述子の制限事項

Solaris および Linux プラットフォームでは、クライアントやブローカが実行されるシェルによって、クライアントが使用できるファイル記述子の数に対する弱い制限値があります。Message Queue システムでは、クライアントが行うコネクション、あるいはブローカが受け付けるコネクションはすべて、これらのファイル記述子のどれかを使用します。持続メッセージを持つ送信先もすべて、ファイル記述子を使用します。

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



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

## 持続データの保護

ブローカは、持続ストアを使用します。ここには、ほかの情報とともに、一時的に保存されるメッセージファイルを保存できます。データストアには、メッセージとともに専有情報を保持できるため、認可されていないアクセスからデータストアを保護することをお勧めします。

ブローカでは、組み込みまたはプラグインのどちらの持続性も利用できます。

### 組み込み持続ストア

組み込みの持続性を利用しているブローカは、プラットフォームに依存するディレクトリ内にある単層ファイルのデータストアに持続データを書き込みます ( [付録 A「Message Queue データの場所」](#) を参照 )。

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

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

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

**Solaris および Linux** `IMQ_VARHOME/instances/instanceName/filestore/` ディレクトリのアクセス権は、そのブローカインスタンスを開始したユーザーの `umask` によって決まります。したがって、ブローカインスタンスの開始および持続ファイルの読み取りを行うためのアクセス権は、`umask` を適切に設定することによって制限できることになります。あるいは、スーパーユーザーである管理者は、`IMQ_VARHOME/instances` ディレクトリのアクセス権を `700` に設定することによって、持続データを保護できます。

**Windows** `IMQ_VARHOME/instances/instanceName/filestore/` ディレクトリのアクセス権は、使用中の Windows オペレーティングシステムが提供するメカニズムを使って設定できます。通常は、そのディレクトリ用のプロパティダイアログが開かれます。

## プラグイン持続ストア

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

Oracle データベースなど、データベースサーバーによって管理されるデータベースについては、Message Queue のデータベーステーブルにアクセスするためのユーザー名とパスワードを作成することをお勧めします。このようなデータベーステーブルには、「IMQ」で始まる名前が付いています。データベースで個々のテーブルの保護ができない場合、Message Queue ブローカだけが使用する専用のデータベースを作成します。ユーザー名とパスワードのアクセス権を作成する方法については、データベースベンダーのマニュアルを参照してください。

データベースコネクションを開くためにブローカが求めるユーザー名とパスワードは、ブローカ設定プロパティとして与えることができます。ただし、ブローカの起動時にコマンド行オプションとして入力するほうがより安全です。『Message Queue 管理ガイド』の付録 A 「プラグイン持続の設定」を参照してください。

Cloudscape データベースなど、データベースの JDBC™ ドライバ経由でブローカが直接アクセスする組み込みデータベースについては、持続データが保存されるディレクトリにファイルアクセス権を設定することによって、通常セキュリティ保護されます。上記の「[組み込み持続ストア](#)」を参照してください。ただし、ブローカと imqdbmgr ユーティリティのどちらもがデータベースの読み取りと書き込みができるようにするには、両方を同一のユーザーが実行する必要があります。

# Message Queue リソースアダプタ

Message Queue には、JMS リソースアダプタが組み込まれています。

リソースアダプタは、機能を J2EE 1.4 互換のアプリケーションサーバに追加するための標準化された手法であり、J2EE Connector Architecture (J2EECA) 1.5 仕様に準拠しています。このアーキテクチャを使用すると、J2EE 1.4 互換のアプリケーションサーバは標準化された手法で外部システムとやり取りできます。これらの外部システムには、さまざまな企業情報システム (EIS)、および JMS プロバイダなどのさまざまなメッセージングシステムが含まれます。

J2EECA 1.5 によって実装が容易になった標準化されたやり取りには、コネクションプーリング、スレッドプーリング、トランザクションとセキュリティのコンテキスト伝達、各種のメッセージ駆動型 Beans コンテナのサポートがあります。仕様には、コネクションファクトリとそのほかの管理対象オブジェクトを作成する標準化された方法も含まれています。

JMS リソースアダプタをアプリケーションサーバへ接続することで、アプリケーション環境内に配置済みで実行中の J2EE コンポーネントが JMS メッセージを交換できるようになります。これらのコンポーネントに必要となる JMS 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は、J2EE アプリケーションサーバ管理ツールを使用して作成し設定されます。

ただし、メッセージサーバと物理的な送信先の管理など、そのほかの管理操作は J2EECA 仕様には含まれていません。プロバイダ固有のツールを使用しなければ実行できません。

Message Queue リソースアダプタは、Sun J2EE 1.4 Application Server に組み込まれています。ただし、Message Queue リソースアダプタはまだそのほかの J2EE 1.4 アプリケーションサーバでは承認されていません。

Message Queue リソースアダプタは 1 つのファイル (mqjmsra.rar) で、[付録 A「Message Queue データの場所」](#)に示すとおり、オペレーティングシステムに応じて該当するディレクトリに配置されています。mqjmsra.rar ファイルには、リソースアダプタの配置記述子 (ra.xml) とアダプタを利用するためにアプリケーションサーバが使用する必要のある jar ファイルが含まれています。

アプリケーションサーバに添付の説明書にしたがってリソースアダプタを配置し設定すれば、J2EE 1.4 互換のアプリケーションサーバで Message Queue リソースアダプタを使用できます。商用の J2EE 1.4 アプリケーションサーバが市場に出されるにつれ、Message Queue リソースアダプタもそれらのアプリケーションサーバで広く承認されつつあります。この付録には、関連する配置および設定手順についての情報が記載されています。

# Message Queue オプションの JMS 機能の実装

JMS 仕様は、いくつかの項目をオプションとしています。各 JMS プロバイダ (ベンダー) が、これらを実装するかどうかを選択することになります。Message Queue 製品におけるこれらのオプション項目の取り扱いを次に示します。

表 G-1 オプションの JMS 機能

JMS 仕様の節	説明と Message Queue における取り扱い
3.4.3 JMSMessageID	<p>「メッセージ ID に対しては、メッセージの作成とサイズの拡大に力を注いできたことから、一部には、メッセージ ID がアプリケーションで使用されないというヒントが与えられれば、メッセージオーバーヘッドを最適化できる JMS プロバイダも存在します。JMS メッセージプロデューサは、メッセージ ID を無効にするためのヒントを与えてくれます。</p> <p><b>Message Queue 実装:</b> 製品は、メッセージ ID の生成を無効にしません。メッセージプロデューサでの <code>setDisableMessageID()</code> の呼び出しは、すべて無視されます。すべてのメッセージは、有効な MessageID 値を持ちます。</p>
3.4.12 メッセージのヘッダー フィールドのオーバーライド	<p>「JMS では、管理者がこれらのヘッダーフィールド値をオーバーライドする方法を、具体的に規定してはいません。JMS プロバイダは、この管理オプションをサポートする必要はありません。</p> <p><b>Message Queue 実装:</b> Message Queue 製品では、コネクションファクトリの管理対象オブジェクトの設定を通じて、メッセージヘッダーフィールド値の管理的オーバーライドをサポートしています (198 ページの表 7-3 を参照)。</p>

表 G-1 オプションの JMS 機能 ( 続き )

JMS 仕様の節	説明と Message Queue における取り扱い
3.5.9 JMS の定義済みプロパティ	<p>「JMS では、JMS の定義済みプロパティ用として、JMSX というプロパティ名のプレフィックスを予約しています。たとくに記述がないかぎり、これらのプロパティのサポートはオプションです。</p> <p><b>Message Queue 実装：</b> JMS 1.1 仕様で定義済みの JMSX プロパティは、Message Queue 製品でサポートされています (198 ページの表 7-3 を参照)。</p>
3.5.10 プロバイダ固有のプロパティ	<p>「JMS では、プロバイダ固有のプロパティ用として、'JMS_&lt;vendor_name&gt;' というプロパティ名のプレフィックスを予約しています。</p> <p><b>Message Queue 実装：</b> プロバイダ固有のプロパティの目的は、プロバイダにネイティブなクライアントで JMS をサポートするのに必要な、特殊な機能を提供することです。これらは、JMS 対 JMS のメッセージングには使用できません。Message Queue 3.5 SP1 では、プロバイダ固有のプロパティを使用していません。</p>
4.4.8 分散トランザクション	<p>「JMS では、プロバイダが分散トランザクションをサポートすることを必要としていません。</p> <p><b>Message Queue 実装：</b> 分散トランザクションは、このリリースの Message Queue 製品でサポートされています (50 ページの「分散トランザクション」を参照)。</p>
4.4.9 複数のセッション	<p>「JMS では、PTP&lt;ポイントツーポイント分散モデル&gt;について、同じキューに同時にアクセスする QueueReceiver に対するセマンティクスを指定していません。しかし、JMS がこの機能のサポートを禁止しているわけではありません。詳細は、JMS 仕様の 5.8 節を参照してください。</p> <p><b>Message Queue 実装：</b> Message Queue 実装は、複数のコンシューマへのキュー配信をサポートしています。詳細は、81 ページの「複数のコンシューマへのキュー配信」を参照してください。</p>

# Message Queue インタフェースの安定度

Sun Java System Message Queue 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 H-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、363 ページの表 H-2 で説明しています。

表 H-1 Message Queue 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
imqobjmgr コマンドファイル	発展中
imqbrokerd コマンド	安定
imqadmin コマンド	不安定
imqcmd コマンド	安定
imqdbmgr コマンド	不安定
imqkeytool コマンド	安定
imqobjmgr コマンド	安定
imqusermgr コマンド	不安定

表 H-1 Message Queue インタフェースの安定度 ( 続き )

インタフェース	分類
JMS API (javax.jms)	標準
JAXM API (javax.xml)	標準
C-API	発展中
メッセージベースの監視 API	発展中
管理対象オブジェクト API (com.sun.messaging)	発展中
imq.jar の格納場所および名前	安定
jms.jar の格納場所および名前	発展中
imqbroker.jar の格納場所および名前	非公開
imqutil.jar の格納場所および名前	非公開
imqadmin.jar の格納場所および名前	非公開
imqservlet.jar の格納場所および名前	発展中
imqhttp.war の格納場所および名前	発展中
imqhttps.war の格納場所および名前	発展中
imqjmsra.rar の格納場所および名前	発展中
imqxm.jar の格納場所および名前	発展中
jaxm-api.jar の格納場所および名前	発展中
saa-j-api.jar の格納場所および名前	発展中
saa-j-impl.jar の格納場所および名前	発展中
activation.jar の格納場所および名前	発展中
mail.jar の格納場所および名前	発展中
dom4j.jar の格納場所および名前	非公開
fscontext.jar の格納場所および名前	不安定
imqbrokerd、imqadmin、imqcmd、imqdbmgr、imqkeytool、 imqobjmgr、imqusermgr からの出力	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 H-2 インタフェースの安定度の分類方式

分類	説明
非公開	ユーザーは直接使用しない。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	ユーザーが使用する。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	ユーザーが使用する。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	ユーザーが使用する。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	ユーザーが使用する。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、Sun Java System Message Queue 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 Java クライアント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。

**Message Queue クライアントランタイム (client runtime)** JMS クライアントに Message Queue 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。

**Message Queue メッセージサーバ (message server)** JMS クライアントへのコネクション、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 Message Queue 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、JMS クライアント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 object)** 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 Message Queue のオブジェクト。1 つ以上の JMS クライアントで使用するために、管理者が作成する。

管理対象オブジェクトを使用すると、JMS クライアント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、JMS クライアント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 JMS クライアント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 (client)** 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の代わりに、Message Queue 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)** 「Message Queue 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) Message Queue 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、Message Queue 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)** Message Queue 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、ユーザー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 Message Queue 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 **Message Queue** 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、管理対象オブジェクトにより識別されカプセル化される。管理対象オブジェクトを使ってクライアントはメッセージをプロデュースする送信先、メッセージをコンシュームする送信先を指定する。

**データストア** 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、JMS クライアント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。ポリシーには、シングル、ファイルオーバー、ラウンドロビンがある。

**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 には、それぞれ特定のクライアントランタイムオブジェクトと特定の送信先タイプを使用する個別のプログラミングドメイン、並びにユニファイド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 **Message Queue** 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 (message)** JMS クライアント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。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)** 「Message Queue メッセージサーバ」を参照。

**メッセージセクタ (message selector)** JMS メッセージヘッダーのプロパティ値 (セクタ) 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。

**ユーザーグループ (user group)** コネクションや送信先などの Message Queue 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、Message Queue クライアントのユーザーが属するグループ。

# 索引

## A

admin コネクションサービス , 57, 173

API マニュアル , 30, 304, 305, 306

## C

Cloudscape, 309

## E

Enterprise Edition, 36

## H

### HTTP

コネクションサービス、「httpjms コネクションサービス」を参照

サポートのアーキテクチャ , 319

転送ドライバ , 319

プロキシ , 319

### httpjms コネクションサービス

概要 , 56, 173

設定 , 321, 323

### HTTPS

コネクションサービス、「httpsjms コネクションサービス」を参照

サポートのアーキテクチャ , 319

### httpsjms コネクションサービス

概要 , 57, 173

設定 , 332, 335

### HTTPS コネクション

サポート , 319

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 , 338

要求間隔 , 336

### HTTPS トンネルサーブレット

概要 , 320

配置 , 333

### HTTP コネクション

サポート , 319

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 , 325

要求間隔 , 324

### HTTP トンネルサーブレット

概要 , 320

配置 , 321

## I

imq.accesscontrol.enabled プロパティ , 73, 136

imq.accesscontrol.file.filename プロパティ , 73, 136

imq.authentication.basic.user\_repository プロパティ , 72, 136

imq.authentication.client.response.timeout プロパティ , 73, 136

imq.authentication.type プロパティ , 72, 136

imq.autocreate.destination.isLocalOnly プロパティ , 84, 136

imq.autocreate.destination.limitBehavior プロパティ , 84, 136

imq.autocreate.destination.maxBytesPerMsg プロパティ , 84, 136

imq.autocreate.destination.maxCount プロパティ , 83, 137

imq.autocreate.destination.maxNumMsgs プロパティ , 83

imq.autocreate.destination.maxNumProducers プロパティ , 84, 137

imq.autocreate.destination.maxTotalMsgBytes プロパティ , 83, 137

imq.autocreate.queue.consumerFlowLimit プロパティ , 85, 137

imq.autocreate.queue.localDeliveryPreferred プロパティ , 85, 137

imq.autocreate.queue.maxNumActiveConsumers プロパティ , 84, 137, 168

imq.autocreate.queue.maxNumBackupConsumers プロパティ , 84, 137, 168

imq.autocreate.queue プロパティ , 83, 137, 168

imq.autocreate.topic プロパティ , 83, 137, 168

imq.cluster.brokerlist プロパティ , 147

imq.cluster.masterbroker プロパティ , 147

imq.cluster.port プロパティ , 148

imq.cluster.transport プロパティ , 148

imq.cluster.url プロパティ , 148, 168

imq.hostname プロパティ , 59, 137

imq.httpjms.http.connectionTimeout プロパティ , 324

imq.httpjms.http.pullPeriod プロパティ , 324

imq.httpjms.http.servletHost プロパティ , 323

imq.httpjms.http.servletPort プロパティ , 323

imq.httpjms.https.connectionTimeout プロパティ , 336

imq.httpjms.https.pullPeriod プロパティ , 336

imq.httpjms.https.servletHost プロパティ , 335

imq.httpjms.https.servletPort プロパティ , 335

imq.keystore.file.dirpath プロパティ , 232

imq.keystore.file.name プロパティ , 232

imq.keystore.password プロパティ , 232, 236

imq.log.console.output プロパティ , 78, 137

imq.log.console.stream プロパティ , 78, 137

imq.log.file.dirpath プロパティ , 78, 137

imq.log.file.filename プロパティ , 78

imq.log.file.name プロパティ , 137

imq.log.file.output プロパティ , 78, 138

imq.log.file.rolloverbytes プロパティ , 78, 138, 168

imq.log.file.rolloversecs プロパティ , 78, 138, 168

imq.log.level プロパティ , 78, 138, 168

imq.log.syslog.facility プロパティ , 79, 138

imq.log.syslog.identity プロパティ , 79, 138

imq.log.syslog.logconsole プロパティ , 79, 138

imq.log.syslog.logpid プロパティ , 79, 138

imq.log.syslog.output プロパティ , 79, 138

imq.log.timezone プロパティ , 79, 138

imq.message.expiration.interval プロパティ , 65, 138

imq.message.max\_size プロパティ , 65, 138, 168

imq.metrics.enabled プロパティ , 77, 138

imq.metrics.interval プロパティ , 77, 138

imq.metrics.topic.enabled プロパティ , 79, 138

imq.metrics.topic.interval プロパティ , 80, 138

imq.metrics.topic.persist プロパティ , 80, 138

imq.metrics.topic.timetolive プロパティ , 80, 138

imq.passfile.dirpath プロパティ , 74, 138

imq.passfile.enabled プロパティ , 73, 138

imq.passfile.name プロパティ , 74, 138

imq.persist.file.destination.message.filepool.limit プロパティ , 69, 139

imq.persist.file.message.cleanup プロパティ , 69, 139



imq.persist.file.message.filepool.cleanratio プロパティ, 69, 139  
 imq.persist.file.message.max\_record\_size プロパティ, 69, 139  
 imq.persist.file.message.vrfile.max\_record\_size プロパティ, 67  
 imq.persist.file.sync.enabled プロパティ, 69, 139  
 imq.persist.jdbc.brokerid プロパティ, 312  
 imq.persist.jdbc.createdburl プロパティ, 312, 313  
 imq.persist.jdbc.driver プロパティ, 312  
 imq.persist.jdbc.opendburl プロパティ, 312  
 imq.persist.jdbc.password プロパティ, 236, 313  
 imq.persist.jdbc.table.IMQCCREC35 プロパティ, 313  
 imq.persist.jdbc.table.IMQDEST35 プロパティ, 314  
 imq.persist.jdbc.table.IMQINT35 プロパティ, 314  
 imq.persist.jdbc.table.IMQLIST35 プロパティ, 314  
 imq.persist.jdbc.table.IMQMSG35 プロパティ, 314  
 imq.persist.jdbc.table.IMQPROPS35 プロパティ, 314  
 imq.persist.jdbc.table.IMQSV35 プロパティ, 313  
 imq.persist.jdbc.table.IMQTACK35 プロパティ, 315  
 imq.persist.jdbc.table.IMQTXN35 プロパティ, 314  
 imq.persist.store プロパティ, 68, 139, 312  
 imq.ping.interval プロパティ, 59, 139  
 imq.portmapper.backlog プロパティ, 59, 139  
 imq.portmapper.hostname プロパティ, 59, 139  
 imq.portmapper.port プロパティ, 59, 139, 168  
 imq.protocol protocol\_type inbufsz, 294  
 imq.protocol protocol\_type nodelay, 294  
 imq.protocol protocol\_type outbufsz, 294  
 imq.resource\_state.count プロパティ, 65, 139  
 imq.resource\_state.threshold プロパティ, 65, 139  
 imq.service.activelist プロパティ, 59, 139  
 imq.service\_name.accesscontrol.enabled プロパティ, 73, 139  
 imq.service\_name.accesscontrol.file.filename プロパティ, 73, 139  
 imq.service\_name.authentication.type プロパティ, 72, 140  
 imq.service\_name.max\_threads プロパティ, 60, 140  
 imq.service\_name.min\_threads プロパティ, 60, 140  
 imq.service\_name.protocol\_type.hostname プロパティ, 60, 140, 148  
 imq.service\_name.protocol\_type.port property, 60  
 imq.service\_name.protocol\_type.port プロパティ, 140  
 imq.service\_name.threadpool\_model プロパティ, 60, 140  
 imq.shared.connectionMonitor\_limit プロパティ, 61, 140  
 imq.system.max\_count プロパティ, 65, 140, 168  
 imq.system.max\_size プロパティ, 65, 140, 168  
 imq.transaction.autorollback プロパティ, 65, 140, 191  
 imq.user\_repository.ldap.base プロパティ, 222  
 imq.user\_repository.ldap.gidattr プロパティ, 223  
 imq.user\_repository.ldap.grpbasedn プロパティ, 223  
 imq.user\_repository.ldap.grpfiltler プロパティ, 223  
 imq.user\_repository.ldap.grpsearch プロパティ, 223  
 imq.user\_repository.ldap.memattr プロパティ, 223  
 imq.user\_repository.ldap.password プロパティ, 222, 236  
 imq.user\_repository.ldap.principal プロパティ, 222  
 imq.user\_repository.ldap.server プロパティ, 222  
 imq.user\_repository.ldap.ssl.enabled プロパティ, 223  
 imq.user\_repository.ldap.timeout プロパティ, 223  
 imq.user\_repository.ldap.uidattr プロパティ, 222  
 imq.user\_repository.ldap.usrfilter プロパティ, 222  
 IMQ\_HOME ディレクトリ変数, 27  
 IMQ\_JAVAHOME ディレクトリ変数, 28  
 IMQ\_VARHOME ディレクトリ変数, 27  
 imqAckOnAcknowledge 属性, 198  
 imqAckOnProduce 属性, 198  
 imqAckTimeout 属性, 198  
 imqAddressListBehavior 属性, 198  
 imqAddressListIterations 属性, 198

imqAddressList 属性, 198

imqbrokerd コマンド  
オプション, 143  
概要, 101  
コマンド構文, 141  
使用, 141

imqBrokerHostName 属性 (Message Queue 3.0), 198

imqBrokerHostPort 属性 (Message Queue 3.0), 198

imqBrokerServicePort 属性 (Message Queue 3.0), 198

imqcmd コマンド  
オプション, 162  
概要, 101  
コマンド構文, 160  
サブコマンド, 160  
使用, 160  
送信先の管理, 177  
トランザクション管理, 189  
ブローカへの安全なコネクション, 163, 235  
ブローカへの接続, 164  
メトリックスの監視, 256

imqConfiguredClientID 属性, 198

imqConnectionFlowCount 属性, 198

imqConnectionFlowLimitEnabled 属性, 198

imqConnectionFlowLimit 属性, 198

imqConnectionType 属性 (Message Queue 3.0), 198

imqConnectionURL 属性 (Message Queue 3.0), 198

imqConsumerFlowLimit 属性, 198

imqConsumerFlowThreshold 属性, 198

imqdbmgr コマンド  
オプション, 317  
概要, 102  
コマンド構文, 316  
サブコマンド, 316

imqDefaultPassword 属性, 198

imqDefaultUsername 属性, 198

imqDestinationDescription 属性, 95, 199

imqDestinationName 属性, 95, 200

imqDisableSetClientID 属性, 198

imqFlowControlLimit 属性, 198

imqJMSDeliveryMode 属性, 198

imqJMSExpiration 属性, 198

imqJMSPriority 属性, 198

imqkeytool コマンド  
概要, 102  
コマンド構文, 231, 332  
使用, 231, 332

imqLoadMaxToServerSession 属性, 198

imqobjmgr コマンド  
オプション, 201  
概要, 102  
コマンド構文, 200  
サブコマンド, 200  
使用, 200

imqOverrideJMSDeliveryMode 属性, 198

imqOverrideJMSExpiration 属性, 198

imqOverrideJMSHeadersToTemporaryDestinations 属性, 199

imqOverrideJMSPriority 属性, 199

imqQueueBrowserMax MessagesPerRetrieve 属性, 199

imqQueueBrowserRetrieveTimeout 属性, 199

imqReconnectAttempts 属性, 199

imqReconnectEnabled 属性, 199

imqReconnectInterval 属性, 199

imqSetJMSXAppID 属性, 199

imqSetJMSXConsumerTXID 属性, 199

imqSetJMSXProducerTXID 属性, 199

imqSetJMSXRcvTimestamp 属性, 199

imqSetJMSXUserID 属性, 199

imqSSLIsHostTrusted 属性 (Message Queue 3.0), 199

imqsvcadm コマンド  
オプション, 349  
概要, 102  
コマンド構文, 348  
サブコマンド, 348  
使用, 348

imqusermgr コマンド  
オプション, 216

概要 , 102  
コマンド構文 , 215  
サブコマンド , 216  
使用 , 215  
パスワード , 218  
ユーザー名 , 218

## J

J2EE アプリケーション  
EJB 仕様 , 43  
JMS, 43  
メッセージ駆動型 Beans、「メッセージ駆動型 Beans」を参照  
Java 仮想マシン、「JVM」を参照  
JDBC サポート  
概要 , 68  
設定 , 309  
ドライバ , 309, 312  
JDK  
パスの指定 , 144  
パスの指定オプション , 162, 201, 349  
JMS  
仕様 , 30, 33, 40  
プログラミングモデル , 40  
メッセージ構造 , 40  
jms コネクションサービス , 56, 173  
JNDI  
Message Queue のサポート , 34  
オブジェクトストア , 102, 194  
オブジェクトストアの属性 , 194, 203  
管理対象オブジェクト , 42, 46  
検索 , 93, 96, 121, 203  
検索名 , 203, 207  
初期コンテキスト , 194, 196  
メッセージ駆動型 Beans, 44  
ロケーション (プロバイダの URL), 194, 196  
JVM  
パフォーマンスの調整 , 293  
パフォーマンスへの影響 , 251  
メトリックス、「JVM メトリックス」を参照

JVM メトリックス  
imqcmd メトリックスの使用 , 258  
ブローカログファイルの使用 , 262  
メッセージベースの監視の使用 , 263  
メトリックス量 , 267

## L

LDAP サーバ  
オブジェクトストアの属性 , 194  
認証フェイルオーバー , 221  
ユーザーリポジトリのアクセス , 221

## M

MDB、「メッセージ駆動型 Beans」を参照

## O

Oracle, 309

## P

passfile  
コマンド行オプション , 144  
使用 , 236  
場所 , 236, 304, 305, 306  
Platform Edition, 35  
PointBase, 309

## S

SOAP, 34  
SSL  
HTTP 経由 , 236

TCP/IP 経由, 230

暗号化, 230

概要, 72

コネクションサービス、「SSL ベースのコネクションサービス」を参照  
コネクションサービス、「コネクションサービス」を参照

ssladmin コネクションサービス

概要, 57, 173

設定, 230

ssljms コネクションサービス

概要, 56, 173

設定, 230

SSL 標準、「SSL」を参照

SSL ベースのコネクションサービス

起動, 233

設定, 213, 230

syslog, 75, 156

## T

TCP, 56, 173

TLS, 56, 173

## W

Windows のサービス、ブローカの実行, 347

## X

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」も参照

XA コネクションファクトリ、概要, 50

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, 226

アクセス権

admin サービス, 71

Message Queue の操作, 70

passfile, 236

アクセス制御プロパティファイル, 70, 225

キーストア, 332

組み込みデータベース, 311

計算, 226

データストア, 68

ユーザーリポジトリ, 215

アクセスコントロールファイル

アクセス規則, 226

使用, 224

バージョン, 225

フォーマット, 225

アクセス制御ファイル

場所, 304, 305, 306

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

アプリケーションサーバと Message Queue, 45

アプリケーション例, 30, 304, 305, 306

暗号化

SSL ベースのサービス, 230

概要, 72

キーツール, 72

## い

移植性、「プロバイダへの非依存性」を参照

一時送信先, 85, 182

インスタンス設定ファイル、「設定ファイル」を参照

## え

永続的サブスクリプション  
クライアント ID, 47

永続的サブスクリバ、「永続的サブスクリプション」を参照

永続的サブスクリプション

id, 162

一覧表示, 188

概要, 46

管理, 188

破棄, 188, 189

パフォーマンスへの影響, 247

メッセージのページ, 188

エディション、製品

概要, 35

企業向け, 36

プラットフォーム, 35

## お

オブジェクトストア

LDAP サーバ, 194

LDAP サーバの属性, 194

概要, 194

場所, 304, 305, 306

ファイルシステムストア, 195

ファイルシステムストア属性, 196

オペレーティングシステム

Solaris のパフォーマンスの調整, 293

パフォーマンスへの影響, 251

## か

環境変数、「ディレクトリ変数」を参照

監視、「パフォーマンスの監視」を参照

管理

送信先, 177

ブローカ, 165

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, 210

オブジェクトストア、「オブジェクトストア」を参照

概要, 42, 93

キュー、「キュー」を参照

クエリー, 211

更新, 211

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, 209

種類, 42, 93, 197

送信先、「送信先管理対象オブジェクト」を参照

属性, 197

トピック、「トピック」を参照

名前の検索, 201

必要な情報, 203

プロバイダへの非依存性, 94

管理タスク

開発環境, 97

本稼働環境, 98

管理ツール

概要, 100

管理コンソール, 100

コマンド行ユーティリティ, 101

## き

キーストア

file, 333

ファイル, 231, 232

プロパティ, 232

キーツール, 72

キーの組み合わせ

再生成, 232

生成, 231

起動

SSL ベースのコネクションサービス, 233

クラスタ内のブローカ, 152

ブローカ, 141

キュー, 80

管理対象オブジェクトの追加, 208

自動作成, 83, 137

属性, 180

ロードバランスされた配信、「ロードバランスされたキューの配信」を参照  
キュー送信先、「キュー」を参照  
キューのロードバランスされた配信  
属性, 181

## く

クエリー  
    コネクションサービス, 172, 174, 176  
    ブローカ, 167  
組み込み持続, 67  
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), 47  
    プログラミングモデル, 40  
    ランタイム、「クライアントランタイム」を参照  
クライアントアプリケーション  
    システムプロパティ, 96  
    パフォーマンスに影響する要因, 242  
    プロバイダへの非依存性, 45  
    例, 30, 304, 305, 306  
クライアントランタイム  
    概要, 90  
    設定, 255  
    メッセージフローの調整, 299  
クラスタコネクションサービス  
    設定、安全, 150, 230  
    ポート番号, 148  
クラスタ設定ファイル, 90  
クラスタ、「ブローカクラスタ」を参照

## こ

更新  
    コネクションサービス, 172, 174, 176  
    ブローカ, 168  
コネクション  
    一覧表示, 176

概要, 41  
クエリー, 176  
コマンドの影響, 176  
パフォーマンスへの影響, 251  
コネクションサービス  
    admin, 57, 173  
    HTTP、「HTTP コネクション」を参照  
    httpjms, 56, 173  
    HTTPS、「HTTPS コネクション」を参照  
    httpsjms, 57, 173  
    jms, 56, 173  
    ssladmin、「ssladmin コネクションサービス」を参照  
    ssljms、「ssljms コネクションサービス」を参照  
    SSL ベース, 232  
    アクセス制御, 73  
    概要, 56  
    起動時にアクティブ化, 59  
    クエリー, 172, 176  
    クラスタ, 150  
    クラスタ (cluster), 230  
    更新, 172, 174, 176  
    コネクションタイプ, 56  
    コマンドの影響, 171  
    サービスタイプ, 56  
    再開, 172, 175  
    スレッドの割り当て, 174  
    スレッドプールマネージャ, 58  
    静的ポート, 59  
    停止, 172, 175  
    プロパティ, 59, 174  
    プロパティの表示, 174  
    ポートマッパー、ポートマッパー」を参照  
    メトリックスデータ、「コネクションサービスのメトリックス」を参照  
コネクションサービスのメトリックス  
    imqcmd query の使用, 261  
    imqcmd メトリックスの使用, 175, 259  
    メトリックス量, 270  
コネクションファクトリ管理対象オブジェクト  
    JNDI 検索, 42  
    オーバーライド, 96  
    概要, 41, 94

クライアント ID, 47

属性, 94, 197

追加, 206

コマンド行の構文, 102

コマンド行ユーティリティ

imqbrokerd、「imqbrokerd コマンド」を参照

imqcmd、「imqcmd コマンド」を参照

imqdbmgr、「imqdbmgr コマンド」を参照

imqkeytool、「imqkeytool コマンド」を参照

imqobjmgr、「imqobjmgr コマンド」を参照

imqsvcadmin、「imqsvcadmin コマンド」を参照

imqusermgr、「imqusermgr コマンド」を参照

概要, 101

基本構文, 102

共通するオプション, 103

コマンドのオプション, 103

コマンドファイル, 204

コンシューマ, 41

コンテナ

EJB, 44

MDB, 44

コントロールメッセージ, 62

コンポーネント

EJB, 43

MDB, 43

## さ

サービスタイプ

ADMIN, 56

NORMAL, 56

サブレット、トンネル、「HTTP/HTTPS トンネル

サブレット」を参照

再開

コネクションサービス, 172, 175

送信先, 185

ブローカ, 167, 169

再配信フラグ, 63

サブスクリプション

永続的、「永続サブスクリプション」を参照

概要, 46

## し

自己署名型証明書, 231, 332

システムプロパティ、設定, 96

持続マネージャ

JDBC データストア, 311

概要, 66

データストア、「データストア」を参照

プラグイン持続, 309

ブローカのコンポーネント, 56

プロパティ, 68

承認

「アクセス制御ファイル」も参照

概要, 70

管理, 224

ユーザーグループ, 71

証明書, 231, 332

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, 48

パフォーマンスのかね合い, 51, 243

## す

スレッドプールマネージャ

概要, 58

共有スレッド, 58

専用スレッド, 58

## せ

制限の動作

送信先, 63, 179, 180

ブローカ, 64

セキュリティ

暗号化、「暗号化」を参照

オブジェクトストア, 194

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, 69

ブローカのコンポーネント, 56

プロパティ, 72

## セッション

概要, 41

処理済み, 49

通知オプション, 49

## 接続、ブローカへ, 164

### 設定ファイル

インスタンス, 134, 148, 168, 303, 304, 306

インストール, 133

デフォルト, 133

テンプレート, 303, 304, 306

テンプレートの場所, 303, 304, 306

場所, 303, 304, 306

編集, 136

### 設定変更記録, 88

## セレクタ

概要, 51

パフォーマンスへの影響, 248

メッセージプロパティとして, 40

# そ

## 送信先

アクセス制御, 228

一時, 85, 182

一覧表示, 178, 182

概要, 54

管理, 177

キュー、「キュー」を>参照

クラスタ内の限定されたスコープ, 182, 84

再開, 179, 185

作成, 179

自動作成, 82, 229

種類, 80, 178

情報, 183

情報の取得, 179

制限の動作, 63, 179, 180

属性, 180

属性値, 183

属性値の表示, 183

属性の更新, 179

停止, 179, 185

トピック、「トピック」を参照

破棄, 178, 179, 186

ファイルベースのデータストアの圧縮, 177

物理的, 80

メッセージのページ, 179, 185

メッセージを配信するためのバッチ処理, 85, 181

メトリックス、「送信先のメトリックス」を参照

## 送信先管理対象オブジェクト

概要, 41, 95

属性, 199

## 送信先の自動作成

概要, 82

プロパティ, 83

## 送信先メトリックス

imqcmd query の使用, 261

imqcmd メトリックスの使用, 178, 257, 260

メッセージベースの監視の使用, 263

メトリックス量, 273

## 属性

管理対象オブジェクト, 197

送信先, 180

# つ

## 通知

概要, 49, 62

クライアント, 62, 92

待機時間, 198

トランザクション, 63

配信, 62

ブローカ, 62

ツール、管理、「管理ツール」を参照

# て

停止



コネクションサービス , 172, 175

送信先 , 179, 185

ブローカ , 166, 169

ディレクトリ変数

IMQ\_HOME, 27

IMQ\_JAVAHOME, 28

IMQ\_VARHOME, 27

データ、Message Queue、場所 , 303

データストア

JDBC アクセス可能 , 68

概要 , 66

単層型ファイル , 67

場所 , 303, 304, 306

パフォーマンスへの影響 , 254

リセット , 145

転送、「メッセージルーター」を参照

## と

トピック

概要 , 46

管理対象オブジェクトの追加 , 207

自動作成 , 83, 137

属性 , 180

物理的な送信先 , 82

トピック送信先、「トピック」を参照

ドメイン , 46

トラブルシューティング , 276

トランザクション

概要 , 49

管理 , 189

コミット , 190

情報 , 190

通知 , 63

パフォーマンスへの影響 , 245

分散、「分散トランザクション」を参照

ロールバック , 189

トランスポートプロトコル

相対速度 , 252

パフォーマンスの調整 , 294

パフォーマンスへの影響 , 252

プロトコルタイプ、「プロトコルタイプ」を参照  
トンネルサーブレット、「HTTP/HTTPS トンネル  
サーブレット」を参照

## に

認証

概要 , 70

管理 , 214

## は

パーシスタンス

JDBC、「JDBC 持続」を参照

組み込み , 67

持続マネージャ、「持続マネージャ」を参照

データストア、「データストア」を参照

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

パーシスタントメッセージ , 48

ページ、送信先からのメッセージ , 185

ハードウェア、パフォーマンスへの影響 , 250

配信、高信頼性 , 48

配信モード

パーシスタント , 48

パフォーマンスへの影響 , 244

非パーシスタント , 48

パスファイル

broker configuration プロパティ , 74

パスワード

JDBC, 236

LDAP, 236

passfile、「passfile」を参照

SSL キーストア , 145, 232, 236

符号化 , 72

命名規則 , 218

パスワード、デフォルト , 198

パスワードファイル、「passfile」を参照

パフォーマンス

インジケータ , 238

影響する要因、「パフォーマンスに影響する要因」を参照

概要, 237

監視、「パフォーマンスの監視」を参照

基準, 238

基準になるパターン, 240

最適化、「パフォーマンスの調整」を参照

信頼性のかね合い, 51, 243

調整、「パフォーマンスの調整」を参照

トラブルシューティング, 276

ベンチマーク, 239

ボトルネック, 242

パフォーマンスに影響する要因

JVM, 251

永続的サブスクリプション, 247

オペレーティングシステム, 251

コネクション, 251

セレクト, 248

通知モード, 246

データストア, 254

トランザクション, 245

トランスポートプロトコル, 252

ハードウェア, 250

配信モード, 244

ブローカの制限の動作, 254

メッセージサーバのアーキテクチャ, 254

メッセージのサイズ, 248

メッセージフロー制御, 255

メッセージ本体のタイプ, 249

パフォーマンスの監視

ツール、「メトリックス監視ツール」を参照

メトリックスデータ、「メトリックスデータ」を参照

パフォーマンスの調整

クライアントランタイムの調整, 299

システムの調整, 293

ブローカの調整, 297

プロセスの概要, 237

パブリッシュ / サブスクライブ配信, 46

## ふ

ファイアウォール, 58, 319

ファイル記述子の制限, 354

フェイルオーバー, 86

プラグイン持続

概要, 68

設定, 309

パフォーマンスの調整, 297

ブローカ

httpjms コネクションサービスのプロパティ, 323

httpsjms コネクションサービスのプロパティ, 335

HTTPS、サポート, 332

HTTP サポート, 321

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, 233

Windows のサービス、実行, 347

アクセス制御、「承認」を参照

インスタンス設定プロパティ, 136

インスタンス名, 144

概要, 54

監視、「ブローカの監視サービス」を参照

管理, 165

起動, 141

クエリー, 167

クラスタ、「ブローカクラスタ」を参照, 147

コネクションサービス、「コネクションサービス」を参照

コネクションサービスの一覧表示, 173

コンポーネントと機能, 55

再開, 167, 169

再起動, 66, 167, 170

持続マネージャ、「持続マネージャ」を参照

シャットダウン, 170

障害からの復元, 66

状態の制御, 169

制限の動作, 64, 254

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, 150, 164

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, 83

- 通知 (ACK), 62, 198
- 停止, 166, 169
- プロパティ, 168
- プロパティの更新, 168
- プロパティの表示, 167
- マスターブローカ, 89
- 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, 65, 140, 168
- メッセージフロー制御、「メッセージフロー制御」を参照
- メトリックス、「ブローカのメトリックス」を参照
- メモリー管理, 63, 179, 254
- 連結、「ブローカのクラスタ」を参照
- ログ作成、「ロガー」を参照
- ブローカインスタンス、「ブローカ」を参照
- ブローカクラスタ
  - アーキテクチャ, 86, 253
  - 安全なブローカ間のコネクション, 150
  - 開発専用環境, 89
  - クラスタ設定ファイル, 90, 148
  - 指定するオプション, 143
  - 使用する理由, 86, 253
  - 情報の伝播, 88
  - 設定プロパティ, 90, 147
  - 設定変更記録, 88, 89
  - 時計の同期, 147
  - パフォーマンスへの影響, 254
  - ブローカの再起動, 152
  - ブローカの接続, 150
  - ブローカの追加, 151
  - プロパティの設定, 148
  - マスターブローカ, 89, 90
- ブローカの監視サービス
  - 概要, 74
  - プロパティ, 77
- ブローカの再起動, 167, 170
- ブローカのシャットダウン, 167, 170
- ブローカのメトリックス
  - imqcmd の使用, 170, 259, 261
  - ブローカログファイルの使用, 262
  - 報告の間隔、ロガー, 144
  - メッセージベースの監視の使用, 263
  - メトリックスメッセージ, 77
  - メトリックス量, 268
  - ロガーのプロパティ, 77, 261
- フロー制御、「メッセージフロー制御」を参照
- プログラミングドメイン, 46
- プロデューサ
  - 概要, 41
  - 送信先の制限, 84, 180
- プロトコルタイプ
  - HTTP, 56, 173
  - TCP, 56, 173
  - TLS, 56, 173
- プロトコル、「トランスポートプロトコル」を参照
- プロトコル、「トランスポートレイヤー」を参照
- プロバイダへの非依存性
  - 概要, 45
  - 管理対象オブジェクト, 94
- プロパティ
  - httpjms コネクションサービス, 323
  - httpsjms コネクションサービス, 335
  - JDBC 関連, 312
  - 管理対象オブジェクト、「管理対象オブジェクト、属性」を参照
  - キーストア, 232
  - クラスタ設定, 147
  - コネクションサービス, 59
  - 自動作成, 83
  - セキュリティ, 72
  - 送信先、「送信先、属性」を参照
  - パーシスタンス, 68
  - ブローカインスタンス設定, 136
  - ブローカの監視サービス, 77
  - ブローカの更新, 168
  - メッセージルーター, 65
  - メモリー管理, 65, 179
  - ロガー, 77
- 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, 50, 189
  - 概要, 50

## へ

ベンチマーク、パフォーマンス , 239

## ほ

ポイントツーポイント配信 , 46

ポート、動的な割り当て , 58

ポートマップ

    ポートの割り当て , 145

ポートマップパー

    概要 , 57

    ポートの割り当て , 59

ボトルネック、パフォーマンス , 242

## ま

マスターブローカ , 89, 90

## め

メッセージ

    SOAP, 34

    概要 , 40

    構造 , 40

    コンシューム , 92

    コントロール , 62

    サイズ、パフォーマンス , 248

    再配信 , 63

    持続 , 63, 66

    順序付け , 52

    信頼性の高い配信 , 48

    スループットのパフォーマンス , 238

    送信先からのページ , 179

    送信先の制限 , 180

    遅延 , 238

    通知 , 62

    転送および配信 , 61

    パーシスタント , 48

    配信モード、「配信モード」を参照

    配信モデル , 39, 46

    パブリッシュ / サブスクライブ配信 , 46

    フィルタリング、「セレクト」を参照

    ブローカの制限 , 65, 140, 168

    フロー制御、「メッセージフロー制御」を参照

    プロデュース , 91

    ポイントツーポイント配信 , 46

    本体のタイプ、パフォーマンス , 249

    メトリックス , 76

    メトリックスメッセージ、「メトリックスメッセージ」を参照

    有効期限の再利用 , 65

    優先度の決定 , 52

    リスナ , 42, 92

    ロードバランスされたキューの配信 , 81

メッセージ駆動型 Beans

    MDB コンテナ , 44

    アプリケーションサーバのサポート , 45

    概要 , 43

    配置記述子 , 44

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

    アーキテクチャ , 253

    概要 , 54

    マルチブローカ、「ブローカのクラスタ」を参照 , 86

メッセージサービス

    概要 , 38

    パフォーマンスに影響する要因 , 250

メッセージ配信モデル , 39, 46

メッセージフロー制御

    制限 , 300

    測定 , 299

    パフォーマンスの調整 , 299

    パフォーマンスへの影響 , 255

    ブローカ , 63, 179

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 , 61

    ブローカのコンポーネント , 55

- プロパティ, 65
- メッセージングシステム
  - Message Queue アーキテクチャ, 54
  - アーキテクチャ, 38
  - メッセージサービス, 38
- メトリックス
  - 概要, 75
  - 監視ツール、「メトリックス監視ツール」を参照
  - データ、「メトリックスデータ」を参照
  - トピック送信先, 76, 263
  - メッセージ、「メトリックスメッセージ」を参照
- メトリックス監視ツール
  - Message Queue のログファイル, 261
  - 比較, 265
  - メッセージキューコマンドユーティリティ (imqcmd), 256
  - メッセージベースの監視 API, 262
- メトリックスデータ
  - imqcmd メトリックスの使用, 258
  - 記述的なリスト, 267
  - コネクションサービス、「コネクションサービスのメトリックス」を参照
  - 送信先、「送信先のメトリックス」を参照
  - ブローカ、「ブローカのメトリックス」を参照
  - ブローカログファイルの使用, 262
  - メッセージベースの監視 API の使用, 263
- メトリックスメッセージ
  - 概要, 76, 263
  - タイプ, 76, 263
  - 内容, 77
- メモリー管理
  - 送信先属性の使用, 179
  - パフォーマンスの調整, 297
  - ブローカ, 63

## ゆ

- ユーザーグループ
  - 概要, 70
  - 定義済み, 217
  - デフォルト, 71
  - 割り当ての削除, 218

- ユーザー名
  - 形式, 218
  - 属性, 198
  - デフォルト, 215
- ユーザーリポジトリ
  - LDAP サーバ, 221
  - 概要, 70
  - 管理, 219
  - 設定, 219
  - 単層型ファイル, 214
  - 場所, 304, 305, 306
  - プラットフォーム依存, 215
  - プロパティ, 72
  - ユーザグループ, 218
  - ユーザの状態, 218

## ら

- ライセンス
  - Enterprise Edition のトライアルライセンスでの起動, 142
  - Message Queue のエディション, 35
  - 起動オプション, 144

## り

- リスナ, 42, 43
- リソースアダプタ, 45

## ろ

- ロードバランスされたキューの配信
  - 概要, 81
  - 属性, 84
  - パフォーマンスの調整, 298
- ロガー
  - 概要, 75
  - カテゴリ, 75

- コンソールへの書き込み , 78, 146
- 出力チャンネル , 75, 156, 261
- 設定の変更 , 155
- ブローカのコネクト , 56
- メッセージの書式設定 , 155
- メトリクス情報 , 77
- レベル , 75, 78, 144
- ロールオーバー条件 , 157
- ログメッセージのリダイレクト , 156

ログ作成、「ロガー」を参照

ログファイル

- デフォルトの場所 , 76, 303, 305, 306
- ロールオーバー条件 , 78