Oracle9*i*

アプリケーション開発者ガイド - アドバンスト・キューイング

リリース 2（9.2）

2002 年 7 月

部品番号 : J06286-01

ORACLE®

Oracle9*i* アプリケーション開発者ガイド - アドバンスト・キューイング , リリース 2（9.2）

部品番号 : J06286-01

原本名：Oracle9*i* Application Developer's Guide - Advanced Queuing, Release 2 (9.2)

原本部品番号：A96587-01

原本著者：D.K. Bradshaw、Bhagat Nainani、Kevin MacDowell、Den Raphaely

グラフィック・デザイナ：Valarie Moore

原本協力者：Neerja Bhatt、Brajesh Goyal、Shelley Higgins、Rajit Kambo、Anish Karmarkar、Krishna Kunchithapadam、Vivek Maganty、Krishnan Meiyyappan、Shengsong Ni、Wei Wang、Sashi Chandrasekaran、Dieter Gawlick、Mohan Kamath、Goran Olsson、Hilkka Outinen、Madhu Reddy、Mary Rhodes、Ashok Saxena、Ekrem Soylemez、Alvin To、Rahim Yaseen

目次

はじめに xxiii

対象読者 xxiv
このマニュアルの構成 xxiv
関連文書 xxvi
表記規則 xxvii

アドバンスト・キューイングの新機能 xxxiii

Oracle9*i* リリース 2（9.2.0）の新機能 xxxiv
Oracle9*i* リリース 1（9.0.1）の新機能 xxxiv
Oracle8*i* での新機能 xxxvii

1 Oracle Advanced Queuing の概要

アドバンスト・キューイングの概要 1-2
統合アプリケーション環境でのアドバンスト・キューイング 1-2
AQ に対するインタフェース 1-3
キューイング・システムの要件 1-4
アドバンスト・キューイングの一般機能 1-5
Point-to-Point およびパブリッシュ・サブスクライブ・メッセージ機能 1-6
Oracle Internet Directory 1-7
Oracle Enterprise Manager の統合 1-7
メッセージ・フォーマットの変換 1-7
SQL アクセス 1-8
統計ビューのサポート 1-8
構造化ペイロード 1-9

保存およびメッセージ履歴 1-9
追跡およびイベント・ジャーナル 1-9
キュー・レベルのアクセス制御 1-10
非永続キュー 1-10
Oracle9*i* Real Application Clusters のサポート 1-10
XMLType ペイロード 1-11
インターネット統合および Internet Data Access Presentation 1-11
否認防止および AQ$< キュー表名 > ビュー 1-13
エンキュー機能 1-13
相関識別子 1-13
サブスクリプション・リストおよび受信者リスト 1-14
エンキューにおけるメッセージの優先順位および順序付け 1-15
メッセージのグループ化 1-15
伝播 1-15
送信元の識別 1-16
時間指定およびスケジューリング 1-16
ルールベースのサブスクライバ 1-16
非同期通知 1-16
デキュー機能 1-17
受信者 1-17
デキューにおけるメッセージのナビゲーション 1-17
デキューのモード 1-17
メッセージ到着待機の最適化 1-17
遅延を伴う再試行 1-18
トランザクション保護のオプション 1-18
例外処理 1-18
リスニング機能（複数のキューでの待機） 1-18
ペイロードを伴わないメッセージ・ヘッダーのデキュー 1-18
伝播機能 1-19
エンキューおよびデキューの自動調整 1-19
LOB を伴うメッセージの伝播 1-19
伝播スケジュール 1-19
伝播スケジュール機能の拡張 1-19
サード・パーティ・サポート 1-20
アドバンスト・キューイングの要素 1-20

メッセージ 1-20
キュー 1-21
キュー表 1-21
エージェント 1-21
受信者 1-22
受信者およびサブスクリプション・リスト 1-22
ルール 1-22
ルールベースのサブスクライバ 1-23
変換 1-23
キュー・モニター 1-23
JMS 用語 1-23
デモ 1-24

2 基本的なコンポーネント

データ構造 2-2
オブジェクト名（object_name） 2-2
型名（type_name） 2-2
エージェント型（aq$_agent） 2-3
AQ 受信者リスト型（aq$_recipient_list_t） 2-4
AQ エージェント・リスト型（aq$_recipient_list_t） 2-4
AQ サブスクライバ・リスト型（aq$_subscriber_list_t） 2-5
AQ 登録情報リスト型（aq$_reg_info_list） 2-5
AQ 転送情報リスト型（aq$_post_info_list） 2-5
AQ 登録情報型（aq$_reg_info） 2-5
AQ 通知記述子型（aq$_descriptor） 2-7
AQ 転送情報型（aq$_post_info） 2-8
管理インタフェースの列挙定数 2-9
操作インタフェースの列挙定数 2-9
INIT.ORA パラメータ・ファイルの考慮点 2-10
AQ_TM_PROCESSES パラメータ 2-10
JOB_QUEUE_PROCESSES パラメータ 2-10

3 AQ プログラム環境

AQ にアクセスするためのプログラム環境 3-2
PL/SQL を使用した AQ へのアクセス 3-3

OCI を使用した AQ へのアクセス 3-4
例 3-5
Visual Basic（OO4O）を使用した AQ へのアクセス 3-5
詳細情報 3-5
AQ Java（oracle.AQ）クラスを使用した AQ へのアクセス 3-6
Java AQ クラスへのアクセス 3-6
アドバンスト・キューイングの使用例 3-7
Java AQ API の管理 3-7
Oracle JMS を使用した AQ へのアクセス 3-8
標準 JMS 機能 3-8
Oracle JMS 拡張機能 3-8
標準 JMS および Oracle JMS へのアクセス 3-9
詳細情報 3-10
AQ XML サーブレットを使用した AQ へのアクセス 3-10
AQ プログラム環境の比較 3-11
AQ 管理インタフェース 3-12
AQ 操作インタフェース 3-16

4 AQ の管理

セキュリティ 4-2
管理者ロール 4-2
ユーザー・ロール 4-2
AQ オブジェクト型へのアクセス 4-3
Oracle 8.1 形式のキュー 4-3
互換性 4-3
セキュリティ 4-3
権限およびアクセス制御 4-4
LNOCI アプリケーション 4-5
伝播に必要なセキュリティ 4-5
キュー表のエクスポート / インポート 4-5
キュー表データのエクスポート 4-5
キュー表データのインポート 4-6
AQ 管理者およびユーザーの作成 4-7
Oracle Enterprise Manager のサポート 4-8
XA でのアドバンスト・キューイングの使用 4-9
キュー管理の制限事項 4-9

メッセージ・ペイロード（実際に通信される情報）内のコレクション型 4-9
キュー表およびキューにおけるシノニム 4-10
表領域の Point-in-Time リカバリ 4-10
非永続キュー 4-10
伝播の問題点 4-10
伝播に必要な実行権限 4-10
ジョブ・キュー・プロセス数 4-11
伝播の最適化 4-11
オブジェクト・キューからの伝播 4-12
AQ 伝播問題のデバッグについてのガイドライン 4-12
Oracle 8.0 形式のキュー 4-14
8.0 への移行および 8.0 からの移行 4-14
8.0 形式のキューのインポートおよびエクスポート 4-15
8.0 のロール 4-15
8.0 形式のキューのセキュリティ 4-16
AQ オブジェクト型へのアクセス 4-16
LNOCI アプリケーションによる 8.0 形式のキューへのアクセス 4-16
トランスポータブル表領域および 8.0 形式のマルチ・コンシューマ・キュー 4-16
DBMS_AQADM パッケージの自動コミット機能 4-17

5 パフォーマンスおよび拡張性

パフォーマンスの概要 5-2
Oracle Real Application Clusters 環境における AQ 5-2
共有サーバー環境におけるアドバンスト・キューイング 5-2
基本的なチューニングのヒント 5-2
エンキュー・プロセスとデキュー・プロセスの同時実行（単一のキュー表の場合） 5-3
エンキュー・プロセスとデキュー・プロセスのシリアル実行（単一のキュー表の場合） 5-3
伝播のチューニングのヒント 5-3

6 FAQ

一般的な質問 6-2
メッセージ・ゲートウェイに関する質問 6-7
伝播に関する質問 6-12
変換に関する質問 6-14
JMS に関する質問 6-17

インターネットのアクセスに関する質問 6-18
Oracle Internet Directory に関する質問（グローバル・エージェント、グローバル・イベントおよびグローバル・キュー） 6-19
変換に関する質問 6-20
パフォーマンスに関する質問 6-20
インストールに関する質問 6-21

7 モデリングおよび設計

キュー・エンティティのモデリング 7-2
基本キューイング 7-3
基本キューイングの説明 7-4
AQ を使用したクライアント / サーバー通信 7-5
複数のコンシューマによる同一メッセージのデキュー 7-6
指定された受信者による指定されたメッセージのデキュー 7-9
ワークフローの AQ 実装 7-11
パブリッシュ・サブスクライブの AQ 実装 7-12
メッセージの伝播 7-13
伝播およびアドバンスト・キューイング 7-13

8 AQ を使用したサンプル・アプリケーション

サンプル・アプリケーション 8-2
アドバンスト・キューイングの一般機能 8-2
システム・レベルのアクセス制御 8-2
キュー・レベルのアクセス制御 8-4
メッセージ・フォーマットの変換 8-5
構造化ペイロード 8-11
XMLType キューのペイロード 8-14
非永続キュー 8-16
保存およびメッセージ履歴 8-26
パブリッシュ・サブスクライブ・サポート 8-27
Oracle Real Application Clusters のサポート 8-29
統計ビューのサポート 8-33
インターネット・アクセス 8-34
エンキュー機能 8-34
サブスクリプションおよび受信者リスト 8-34

メッセージの優先順位および順序付け 8-36
時間指定 : 遅延 8-44
時間指定 : 期限切れ 8-46
メッセージのグループ化 8-49
エンキュー中のメッセージ変換 8-51
AQ XML サーブレットを使用したエンキュー 8-53
デキュー機能 8-55
デキューの方法 8-56
複数の受信者 8-60
ローカルおよびリモートの受信者 8-61
デキューにおけるメッセージ・ナビゲーション 8-63
デキューのモード 8-67
メッセージ到着待機の最適化 8-72
遅延間隔をおいた後の再試行 8-74
例外処理 8-77
ルールベースのサブスクリプション 8-82
Listen 機能 8-86
デキュー中のメッセージ変換 8-91
AQ XML サーブレットを使用したデキュー 8-92
非同期通知 8-93
AQ XML サーブレットを使用した通知登録 8-101
伝播機能 8-102
伝播 8-102
伝播スケジュール 8-103
LOB 属性を伴うメッセージの伝播 8-106
拡張伝播スケジュール機能 8-109
伝播中の例外処理 8-111
伝播中のメッセージ・フォーマットの変換 8-112
HTTP を使用した伝播 8-113

9 管理インタフェース

利用モデル : 管理インタフェース - 基本操作 9-2
キュー表の作成 9-4
PL/SQL（DBMS_AQADM）: キュー表の作成 9-7
Visual Basic（OO4O）: キュー表の作成 9-9

Java（JDBC）: キュー表の作成 9-10
キュー表の作成（STORAGE 句の設定） 9-13
キュー表の変更 9-15
PL/SQL（DBMS_AQADM）: キュー表の変更 9-16
Java（JDBC）: キュー表の変更 9-17
キュー表の削除 9-18
PL/SQL（DBMS_AQADM）: キュー表の削除 9-19
Java（JDBC）: キュー表の削除 9-20
キューの作成 9-21
PL/SQL（DBMS_AQADM）: キューの作成 9-23
Java（JDBC）: キューの作成 9-25
非永続キューの作成 9-27
PL/SQL（DBMS_AQADM）: 非永続キューの作成 9-28
Java（JDBC）: 非永続キューの作成 9-28
キューの変更 9-29
PL/SQL（DBMS_AQADM）: キューの変更 9-30
Java（JDBC）: キューの変更 9-31
キューの削除 9-32
PL/SQL（DBMS_AQADM）: キューの削除 9-33
Java（JDBC）: キューの削除 9-34
変換の作成 9-35
PL/SQL（DBMS_TRANSFORM）: 変換の作成 9-37
変換の変更 9-38
変換の適用 9-40
変換の削除 9-41
キューの開始 9-43
PL/SQL（DBMS_AQADM）: キューの開始 9-44
Java（JDBC）: キューの開始 9-45
キューの停止 9-46
PL/SQL（DBMS_AQADM）: キューの停止 9-47
Java（JDBC）: キューの停止 9-48
システム権限の付与 9-49
PL/SQL（DBMS_AQADM）: システム権限の付与 9-50
Java（JDBC）: システム権限の付与 9-51
システム権限の取消し 9-52
PL/SQL（DBMS_AQADM）の使用 : システム権限の取消し 9-53

キュー権限の付与 9-54
PL/SQL（DBMS_AQADM）：キュー権限の付与 9-55
Java（JDBC）：キュー権限の付与 9-55
キュー権限の取消し 9-56
PL/SQL（DBMS_AQADM）：キュー権限の取消し 9-57
Java（JDBC）：キュー権限の取消し 9-58
サブスクライバの追加 9-59
PL/SQL（DBMS_AQADM）：サブスクライバの追加 9-61
PL/SQL（DBMS_AQADM）：ルールベースのサブスクライバの追加 9-62
Java（JDBC）：サブスクライバの追加 9-63
サブスクライバの変更 9-65
PL/SQL（DBMS_AQADM）：サブスクライバの変更 9-67
Java（JDBC）：サブスクライバの変更 9-68
サブスクライバの削除 9-69
PL/SQL（DBMS_AQADM）：サブスクライバの削除 9-71
Java（JDBC）：サブスクライバの削除 9-71
キューの伝播のスケジューリング 9-72
PL/SQL（DBMS_AQADM）：キューの伝播のスケジューリング 9-74
Java（JDBC）：キューの伝播のスケジューリング 9-74
キューの伝播スケジュールの解除 9-76
PL/SQL（DBMS_AQADM）：伝播スケジュールの解除 9-77
Java（JDBC）：キューの伝播スケジュールの解除 9-77
キュー・タイプの検証 9-79
PL/SQL（DBMS_AQADM）：キュー・タイプの検証 9-81
Java（JDBC）：キュー・タイプの検証 9-81
伝播スケジュールの変更 9-82
PL/SQL（DBMS_AQADM）：伝播スケジュールの変更 9-83
Java（JDBC）：伝播スケジュールの変更 9-84
伝播スケジュールの使用可能化 9-85
PL/SQL（DBMS_AQADM）：伝播の使用可能化 9-86
Java（JDBC）：伝播スケジュールの使用可能化 9-86
伝播スケジュールの使用不可能化 9-88
PL/SQL（DBMS_AQADM）：伝播スケジュールの使用不可能化 9-89
Java（JDBC）：伝播スケジュールの使用不可能化 9-89
AQ エージェントの作成 9-91
AQ エージェントの変更 9-93

AQ エージェントの削除 9-95
データベース・アクセスの許可 9-97
データベース・アクセスの禁止 9-99
LDAP サーバーへの別名の追加 9-101
LDAP サーバーからの別名の削除 9-103

10 管理インタフェース : ビュー

利用モデル : 管理インタフェース - ビュー 10-2
データベース内のすべてのキュー表の選択 10-3
ユーザーのキュー表の選択 10-5
データベース内のすべてのキューの選択 10-7
すべての伝播スケジュールの選択 10-9
ユーザーがなんらかの権限を持っているキューの選択 10-13
ユーザーがキュー権限を持っているキューの選択 10-15
キュー表のメッセージの選択 10-17
ユーザー・スキーマのキュー表の選択 10-21
ユーザー・スキーマのキューの選択 10-23
ユーザー・スキーマの伝播スケジュールの選択 10-25
キューのサブスクライバの選択 10-29
キューのサブスクライバおよびそのルールの選択 10-31
データベース全体における状態ごとのメッセージ数の選択 10-33
特定のインスタンスにおける状態ごとのメッセージ数の選択 10-35
インターネット・アクセスに登録された AQ エージェントの選択 10-37
ユーザー変換の選択 10-39
ユーザー変換ファンクションの選択 10-40
すべての変換の選択 10-41
すべての変換ファンクションの選択 10-43

11 操作インタフェース : 基本操作

利用モデル : 操作インタフェース - 基本操作 11-2
メッセージのエンキュー 11-4
メッセージのエンキュー（オプションの指定） 11-7
メッセージのエンキュー（メッセージ・プロパティの指定） 11-10
メッセージのエンキュー（メッセージ・プロパティの指定（送信者 ID の指定）） 11-13
メッセージのエンキュー（ペイロードの追加） 11-15
PL/SQL（DBMS_AQ）: オブジェクト型メッセージのエンキュー 11-17
Java（JDBC）: メッセージのエンキュー（ペイロードの追加） 11-19

Visual Basic（OO4O）：メッセージのエンキュー 11-22
1 個以上のシングル・コンシューマ・キューのリスニング 11-24
PL/SQL（DBMS_AQ）：キューのリスニング 11-26
Java（JDBC）：キューのリスニング 11-26
C（OCI）：シングル・コンシューマ・キューのリスニング 11-27
1 個以上のマルチ・コンシューマ・キューのリスニング 11-36
PL/SQL（DBMS_AQ）：キューのリスニング 11-37
C（OCI）：キューのリスニング 11-39
メッセージのデキュー 11-44
シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定） 11-47
PL/SQL（DBMS_AQ）：オブジェクト型メッセージのデキュー 11-49
Java（JDBC）：シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定） 11-49
Visual Basic（OO4O）：メッセージのデキュー 11-50
マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定） 11-52
Java（JDBC）：マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定） 11-54
通知の登録 11-55
通知の登録（サブスクリプション名の指定 - シングル・コンシューマ・キュー） 11-58
通知の登録（サブスクリプション名の指定 - マルチ・コンシューマ・キュー） 11-59
C（OCI）：シングル・コンシューマおよびマルチ・コンシューマ・キューへの通知登録 11-60
サブスクライバの通知の転送 11-66
PL/SQL（DBMS_AQ）：オブジェクト型メッセージの転送 11-68
エージェントの LDAP サーバーへの追加 11-69
エージェントの LDAP サーバーからの削除 11-71

12 JMS を使用したアプリケーションの作成

JMS を使用したサンプル・アプリケーション 12-2
JMS の一般的な機能 12-2
J2EE 準拠 12-3
JMS コネクションおよびセッション 12-5
JMS 宛先 - キューおよびトピック 12-12
JMS でのシステム・レベルのアクセス制御 12-16
JMS での宛先レベルのアクセス制御 12-17
JMS での保存およびメッセージ履歴 12-18
JMS での Oracle Real Application Clusters のサポート 12-18

JMS での統計ビューのサポート 12-20
JMS での構造化ペイロード / メッセージの型 12-21
JMS サンプルで使用したペイロード 12-31
JMS での Point-to-Point モデル機能 12-37
キュー 12-37
キュー・センダー 12-38
キュー・レシーバ 12-39
キュー・ブラウザ 12-41
JMS パブリッシュ・サブスクライブ・モデル機能 12-43
トピック 12-43
永続サブスクライバ 12-45
トピック・パブリッシャ 12-47
受信者リスト 12-49
トピック・レシーバ 12-50
トピック・ブラウザ 12-52
JMS メッセージ・プロデューサ機能 12-55
メッセージの優先順位および順序付け 12-55
時間指定 : 遅延 12-58
時間指定 : 期限切れ 12-60
メッセージのグループ化 12-62
メッセージ・コンシューマ機能 12-66
メッセージの受信 12-66
受信におけるメッセージのナビゲーション 12-69
メッセージ受信モード 12-72
遅延間隔をおける再試行 12-74
メッセージ・リスナーを使用したメッセージの非同期受信 12-76
AQ の例外処理 12-80
JMS 伝播 12-83
リモート・サブスクライバ 12-83
伝播スケジュール 12-88
拡張伝播スケジュール機能 12-90
伝播中の例外処理 12-92
JMS AQ のメッセージ変換 12-92
メッセージ変換の定義 12-92
変換による宛先へのメッセージの送信 12-94

変換による宛先からのメッセージの受信 12-95
トピック・サブスクライバ作成時の変換の指定 12-96
リモート・サブスクライバ作成時の変換の指定 12-97

13 JMS 管理インタフェース : 基本操作

利用モデル : JMS 管理インタフェース - 基本操作 13-2
キュー / トピック・コネクション・ファクトリのデータベースを介した登録 : JDBC コネクション・パラメータの使用 13-4
キュー / トピック・コネクション・ファクトリのデータベースを介した登録 : JDBC URL の使用 13-6
キュー / トピック・コネクション・ファクトリの LDAP を介した登録 : JDBC コネクション・パラメータの使用 13-8
キュー / トピック・コネクション・ファクトリの LDAP を介した登録 : JDBC URL の使用 13-11
LDAP 内のキュー / トピック・コネクション・ファクトリのデータベースを介した登録解除 13-14
LDAP 内のキュー / トピック・コネクション・ファクトリの LDAP を介した登録解除 13-16
キュー・コネクション・ファクトリの取得 : JDBC URL の使用 13-18
キュー・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用 13-20
トピック・コネクション・ファクトリの取得 : JDBC URL の使用 13-22
トピック・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用 13-24
LDAP 内のキュー / トピック・コネクション・ファクトリの取得 13-26
LDAP 内のキュー / トピックの取得 13-28
キュー表の作成 13-30
キュー表の作成（キュー表のプロパティの指定） 13-32
キュー表の取得 13-34
宛先プロパティの指定 13-36
キューの作成 : Point-to-Point 13-38
トピックの作成 : パブリッシュ・サブスクライブ 13-40
システム権限の付与 13-42
システム権限の取消し 13-44
トピック権限の付与 : パブリッシュ・サブスクライブ 13-46
トピック権限の取消し : パブリッシュ・サブスクライブ 13-48
キュー権限の付与 : Point-to-Point 13-50
キュー権限の取消し : Point-to-Point 13-52
宛先の開始 13-54
宛先の停止 13-56
宛先の変更 13-58
宛先の削除 13-60
伝播のスケジューリング 13-62
伝播スケジュールの使用可能化 13-64

伝播スケジュールの変更 13-66
伝播スケジュールの使用不可能化 13-68
伝播スケジュールの解除 13-70

14 JMS 操作インタフェース : 基本操作（Point-to-Point）

利用モデル : JMS 操作インタフェース - 基本操作（Point-to-Point） 14-2
キュー・コネクションの確立 : ユーザー名 / パスワードの使用 14-3
キュー・コネクションの確立 : オープンしている JDBC コネクションの使用 14-5
キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用 14-7
キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用 14-9
キュー・セッションの作成 14-11
キュー・センダーの作成 14-13
メッセージの送信 : デフォルト送信オプションのキュー・センダーの使用 14-14
メッセージの送信 : 送信オプションを指定したキュー・センダーの使用 14-16
Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成 14-19
Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成 : メッセージをロック 14-21
Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成 .. 14-23
Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成 : メッセージをロック 14-26
キュー・ブラウザを使用したメッセージのブラウズ 14-28
キュー・レシーバの作成 : 標準 JMS 型メッセージ・キュー 14-30
キュー・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・キュー 14-32
キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用 14-35

15 JMS 操作インタフェース : 基本操作（パブリッシュ・サブスクライブ）

利用モデル : JMS 操作インタフェース - 基本操作（パブリッシュ・サブスクライブ） 15-2
トピック・コネクションの確立 : ユーザー名 / パスワードの使用 15-4
トピック・コネクションの確立 : オープンしている JDBC コネクションの使用 15-6
トピック・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用 15-8
トピック・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用 15-10
トピック・セッションの作成 15-12
トピック・パブリッシャの作成 15-14
トピック・パブリッシャを使用したメッセージのパブリッシュ : 最小限の指定 15-15
トピック・パブリッシャを使用したメッセージのパブリッシュ : 相関および遅延を指定 15-18
トピック・パブリッシャを使用したメッセージのパブリッシュ : 優先順位および Time-To-Live を指定 15-21

トピック・パブリッシャを使用したメッセージのパブリッシュ :
トピック・サブスクライバをオーバーライドする受信者リストを指定 15-24
JMS トピックに対する永続サブスクライバの作成 : セレクタの指定なし 15-27
JMS トピックに対する永続サブスクライバの作成 : セレクタの指定あり 15-29
ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定なし 15-32
ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定あり 15-34
リモート・サブスクライバの作成 : JMS メッセージ・トピック 15-37
リモート・サブスクライバの作成 : Oracle オブジェクト型（ユーザー定義型）
メッセージ・トピック 15-40
永続サブスクリプションのサブスクライブの解除 : ローカル・サブスクライバ 15-43
永続サブスクリプションのサブスクライブの解除 : リモート・サブスクライバ 15-45
トピック・レシーバの作成 : 標準 JMS 型メッセージ・トピック 15-47
トピック・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック 15-49
Text、Stream、Object、Bytes、MapMessage を使用するトピックに対する
トピック・ブラウザの作成 15-52
Text、Stream、Object、Bytes、MapMessage を使用するトピックに対する
トピック・ブラウザの作成 : メッセージをロック 15-54
Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対する
トピック・ブラウザの作成 15-56
Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対する
トピック・ブラウザの作成 : メッセージをロック 15-59
トピック・ブラウザを使用したメッセージのブラウズ 15-61

16 JMS 操作インタフェース：基本操作（共有インタフェース）

利用モデル : JMS 操作インタフェース - 基本操作（共有インタフェース） 16-2
JMS コネクションの開始 16-5
セッションからの JMS コネクションの取得 16-7
セッションにおけるすべての操作のコミット 16-9
セッションにおけるすべての操作のロールバック 16-11
JMS セッションからの JDBC コネクションの取得 16-13
JMS コネクションからの OracleOCIConnectionPool の取得 16-15
BytesMessage の作成 16-17
MapMessage の作成 16-19
StreamMessage の作成 16-21
ObjectMessage の作成 16-23
TextMessage の作成 16-25
JMS メッセージの作成 16-27
JMS メッセージの作成（ヘッダーのみ） 16-29

ユーザー定義型メッセージの作成 16-30
メッセージの相関識別子の指定 16-32
JMS メッセージ・プロパティの指定 16-34
JMS メッセージ・プロパティを Boolean として指定 16-37
JMS メッセージ・プロパティを String として指定 16-40
JMS メッセージ・プロパティを Int として指定 16-43
JMS メッセージ・プロパティを Double として指定 16-46
JMS メッセージ・プロパティを Float として指定 16-49
JMS メッセージ・プロパティを Byte として指定 16-52
JMS メッセージ・プロパティを Long として指定 16-55
JMS メッセージ・プロパティを Short として指定 16-58
JMS メッセージ・プロパティを Object として指定 16-61
メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの Time-To-Live の設定 16-64
メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの優先順位の設定 ... 16-66
AQjms エージェントの作成 16-68
メッセージ・コンシューマを使用したメッセージの同期受信 : タイムアウトを指定 16-70
メッセージ・コンシューマを使用したメッセージの同期受信 : 待機なし 16-72
メッセージの受信に対するナビゲーション・モードの指定 16-74
メッセージを非同期受信するメッセージ・リスナーの指定 : メッセージ・コンシューマ 16-77
メッセージを非同期受信するメッセージ・リスナーの指定 : セッション 16-80
メッセージの相関識別子の取得 16-82
メッセージのメッセージ ID を Byte として取得 16-83
メッセージのメッセージ ID を String として取得 16-85
JMS メッセージ・プロパティの取得 16-87
JMS メッセージ・プロパティを Boolean として取得 16-89
JMS メッセージ・プロパティを String として取得 16-92
JMS メッセージ・プロパティを Int として取得 16-95
JMS メッセージ・プロパティを Double として取得 16-98
JMS メッセージ・プロパティを Float として取得 16-101
JMS メッセージ・プロパティを Byte として取得 16-104
JMS メッセージ・プロパティを Long として取得 16-107
JMS メッセージ・プロパティを Short として取得 16-110
JMS メッセージ・プロパティを Object として取得 16-113
メッセージ・プロデューサのクローズ 16-116
メッセージ・コンシューマのクローズ 16-117
JMS コネクションの停止 16-118
JMS セッションを閉じる 16-120

JMS コネクションのクローズ 16-122
JMS 例外のエラー・コードの取得 16-124
JMS 例外のエラー番号の取得 16-125
JMS 例外のエラー・メッセージの取得 16-126
JMS 例外にリンクされた例外の取得 16-127
JMS 例外のスタック・トレースの出力 16-129
例外リスナーの設定 16-130
例外リスナーの取得 16-132
例外リスナーの ping 周期の設定 16-133
例外リスナーの ping 周期の取得 16-135

17 AQ へのインターネット・アクセス

インターネット経由のアドバンスト・キューイング操作の概要 17-2
Internet Data Access Presentation（iDAP） 17-3
SOAP メッセージの構造 17-3
SOAP メソッドの起動 17-4
iDAP ドキュメント 17-6
SOAP スキーマおよび AQ XML スキーマ 17-33
SOAP スキーマ 17-33
iDAP スキーマ 17-36
AQ XML サーブレットの配置 17-49
AQ XML サーブレット・クラスの作成 17-49
AQ XML サーブレットのコンパイル 17-50
ユーザー認証 17-52
ユーザー認可 17-53
AQ XML サーブレットとの LDAP の使用 17-55
HTTP を使用した AQ XML サーブレットへのアクセス 17-57
ユーザー・セッションおよびトランザクション 17-60
HTTP および HTTPS を使用したアドバンスト・キューイング伝播 17-61
高水準アーキテクチャ 17-61
AQ サーブレットのカスタマイズ 17-64
コネクション・プール・サイズの設定 17-64
セッション・タイムアウトの設定 17-64
サーブレットからのすべてのレスポンスに対するスタイル・シートの設定 17-65
AQ 操作前後のコールバック 17-67

18 メッセージ・ゲートウェイ

メッセージ・ゲートウェイの機能 18-2
メッセージ・ゲートウェイのアーキテクチャ 18-3
管理パッケージ 18-4
ゲートウェイ・エージェント 18-5
伝播処理の概要 18-5
メッセージ・ゲートウェイの設定 18-6
Oracle9*i* データベースの前提条件 18-6
Oracle 以外のメッセージ・システムの前提条件 18-6
タスクのロードおよび設定 18-7
セットアップの確認 18-12
メッセージ・ゲートウェイのアンロード 18-12
メッセージ・ゲートウェイの操作 18-13
メッセージ・ゲートウェイ・エージェントの管理 18-13
メッセージ・ゲートウェイ・リンクの構成 18-15
Oracle 以外のメッセージ・システムのキューの登録 18-18
伝播ジョブの構成 18-20
メッセージ・ゲートウェイのログ・ファイルの監視 18-26
メッセージの変換 18-27
メッセージの変換処理 18-27
メッセージ・ゲートウェイの標準型 18-28
アドバンスト・キューイングに対するメッセージ変換 18-29
MQSeries に対するメッセージ変換 18-31
メッセージ・ヘッダーの変換 18-33
ヘッダー・プロパティの使用例 18-39
XML メッセージ伝播の使用例 18-40
mgw.ora 初期化ファイル 18-44
ファイルの内容 18-45
初期化パラメータ 18-45
環境変数 18-46
Java プロパティ 18-48

A Oracle Advanced Queuing の使用例

キュー表およびキューの作成 A-4
オブジェクト型のキュー表およびキューの作成 A-4

RAW 型のキュー表およびキューの作成 A-4
優先順位指定メッセージのキュー表およびキューの作成 A-5
マルチ・コンシューマ・キュー表およびキューの作成 A-5
伝播のデモ用のキューの作成 A-5
Java AQ の例の設定 A-6
Java AQ セッションの作成 A-7
キュー表およびキューの作成 : Java A-8
キューの作成およびエンキュー / デキューの開始 : Java A-9
マルチ・コンシューマ・キューの作成およびサブスクライバの追加 : Java A-10
メッセージのエンキューおよびデキュー A-11
オブジェクト型メッセージのエンキューおよびデキュー : PL/SQL A-11
オブジェクト型メッセージのエンキューおよびデキュー : Pro*C/C++ A-12
オブジェクト型メッセージのエンキューおよびデキュー : OCI A-14
オブジェクト型メッセージ（CustomDatum インタフェース）の
エンキューおよびデキュー : Java A-16
オブジェクト型メッセージ（SQLData インタフェース使用）の
エンキューおよびデキュー : Java A-18
RAW 型メッセージのエンキューおよびデキュー : PL/SQL A-21
RAW 型メッセージのエンキューおよびデキュー : Pro*C/C++ A-22
RAW 型メッセージのエンキューおよびデキュー : OCI A-24
RAW 型メッセージのエンキュー : Java A-25
メッセージのデキュー : Java A-27
ブラウズ・モードでのメッセージのデキュー : Java A-28
優先順位によるメッセージのエンキューおよびデキュー : PL/SQL A-29
優先順位によるメッセージのエンキュー : Java A-31
プレビュー後のメッセージのデキュー : PL/SQL A-32
遅延および期限切れによるメッセージのエンキューおよびデキュー : PL/SQL A-36
相関識別子およびメッセージ ID によるメッセージのエンキューおよびデキュー : Pro*C/C++ . A-37
相関識別子およびメッセージ ID によるメッセージのエンキューおよびデキュー : OCI A-41
マルチ・コンシューマ・キューでのメッセージのエンキューおよびデキュー : PL/SQL A-43
マルチ・コンシューマ・キューでのメッセージのエンキューおよびデキュー : OCI A-46
メッセージのグループ化によるメッセージのエンキューおよびデキュー : PL/SQL A-50
LOB 属性を含むオブジェクト型メッセージのエンキューおよびデキュー : PL/SQL A-52
LOB 属性を含むオブジェクト型メッセージのエンキューおよびデキュー : Java A-54
伝播 A-60

リモートのサブスクライバ / 受信者用のメッセージのマルチ・コンシューマ・
キューへのエンキューおよび伝播のスケジュール : PL/SQL A-60
同一データベース内の 1 つのキューから他のキューへの伝播管理 : PL/SQL A-62
1 つのキューから他のデータベース内の他のキューへの伝播管理 : PL/SQL A-62
伝播スケジュールの解除 : PL/SQL A-63
AQ オブジェクトの削除 A-64
ロールおよび権限の取消し A-65
AQ による XA の使用 A-66
AQ およびメモリーの使用 A-70
Creat_types.sql: Scott のスキーマへのペイロード型およびキューの作成 A-70
メッセージのエンキュー（各コール後のメモリー解放）: OCI A-70
メッセージのエンキュー（メモリーの再利用）: OCI A-74
メッセージのデキュー（各コール後のメモリー解放）: OCI A-77
メッセージのデキュー（メモリーの再利用）: OCI A-81

B Oracle JMS インタフェース、クラスおよび例外

Oracle JMS クラス（パート 1） B-6
Oracle JMS クラス（パート 2） B-8
Oracle JMS クラス（パート 3） B-9
Oracle JMS クラス（パート 4） B-10
Oracle JMS クラス（パート 5） B-11
Oracle JMS クラス（パート 6） B-12
Oracle JMS クラス（パート 6 の続き） B-13
Oracle JMS クラス（パート 7） B-15
Oracle JMS クラス（パート 8） B-17
Oracle JMS クラス（パート 9） B-19
Oracle JMS クラス（パート 10） B-21
Oracle JMS クラス（パート 10 の続き） B-22
インタフェース、クラスおよび例外 B-24

C BooksOnLine 用スクリプト

**tkaqdoca.sql: ユーザー、オブジェクト、キュー表、キューおよび
サブスクライバ作成用のスクリプト** C-2
tkaqdocd.sql: 管理インタフェースおよび操作インタフェースの例 C-15
tkaqdoce.sql: 操作例 C-20
tkaqdocp.sql: 操作インタフェースの例 C-21

tkaqdocc.sql: クリーンアップ・スクリプト C-35

D JMS および AQ XML サーブレット・エラー・メッセージ

JMS エラー・メッセージ D-2
AQ XML サーブレット・エラー・メッセージ D-14

E Unified Modeling Language 図

利用図 E-2
状態図 E-8

索引

はじめに

このマニュアルでは、Oracle Advanced Queuing（AQ）を使用したアプリケーションの開発および統合に関する機能について説明します。この情報は、特に指定されないかぎり、すべてのプラットフォームで実行する Oracle データベース・サーバーの各バージョンに適用されます。

ここでは、次の項目について説明します。

- 対象読者
- このマニュアルの構成
- 関連文書
- 表記規則

対象読者

このマニュアルは、アドバンスト・キューイングを使用するアプリケーションを開発するプログラマを対象としています。

このマニュアルの構成

このマニュアルの構成は、次のとおりです。

第 1 章「Oracle Advanced Queuing の概要」
この章では、最適なメッセージ・システムの要件について説明します。

第 2 章「基本的なコンポーネント」
この章では、一般機能、エンキュー機能およびデキュー機能を含む、アドバンスト・キューイングの機能について説明します。

第 3 章「AQ プログラム環境」
この章では、操作が必要な要素、および AQ アプリケーション環境を準備するときに考慮する必要がある問題点を説明します。

第 4 章「AQ の管理」
この章では、キュー表の移行（インポート / エクスポート）、セキュリティ、Oracle Enterprise Manager のサポート、プロトコル、アドバンスト・キューイングを使用するための準備作業で必要となる DBA の操作、現行の制限事項など、アドバンスト・キューイングの管理について説明します。

第 5 章「パフォーマンスおよび拡張性」
この章では、パフォーマンスおよび拡張性の問題について説明します。

第 6 章「FAQ」
この章では、よくある質問事項について説明します。

第 7 章「モデリングおよび設計」
この章では、アドバンスト・キューイングのモデリングおよび設計の基礎について説明します。

第 8 章「AQ を使用したサンプル・アプリケーション」
この章では、サンプル・アプリケーションを使用してアドバンスト・キューイングの機能について説明します。

第 9 章「管理インタフェース」
この章では、アドバンスト・キューイング用の管理インタフェースについて説明します。

第 10 章「管理インタフェース : ビュー」
この章では、利用方法および状態図を使用して、管理インタフェースのビューについて説明します。

第 11 章「操作インタフェース : 基本操作」
この章では、利用方法に沿って、アドバンスト・キューイングの操作インタフェースについて説明します。

第 12 章「JMS を使用したアプリケーションの作成」
この章では、サンプル・アプリケーションを使用してアドバンスト・キューイングに対する Oracle Java Messaging Service（JMS）インタフェースの機能について説明します。

第 13 章「JMS 管理インタフェース : 基本操作」
この章では、利用方法に沿って、アドバンスト・キューイング用の管理インタフェースについて説明します。

第 14 章「JMS 操作インタフェース : 基本操作（Point-to-Point）」
この章では、Point-to-Point の操作について説明します。

第 15 章「JMS 操作インタフェース : 基本操作 （パブリッシュ・サブスクライブ）」
この章では、パブリッシュ・サブスクライブの操作について説明します。

第 16 章「JMS 操作インタフェース : 基本操作 （共有インタフェース）」
この章では、共有インタフェースの操作について説明します。

第 17 章「AQ へのインターネット・アクセス」
この章では、Simple Object Access Protocol（SOAP）および Internet Data Access Presentation（iDAP）を使用したり、Hypertext Transfer Protocol（HTTP）などの転送プロトコルを使用してインターネット上でメッセージを転送することによって、インターネット経由で AQ 操作を実行する方法を説明します。

第 18 章「メッセージ・ゲートウェイ」
この章では、メッセージ・ゲートウェイを使用して、AQ ベースのアプリケーションで Oracle 以外のメッセージ・システムと通信する方法を説明します。

付録 A「Oracle Advanced Queuing の使用例」
この付録では、様々なプログラム環境での使用例を示します。

付録 B「Oracle JMS インタフェース、クラスおよび例外」
この付録では、Oracle Java Message Service（JMS）インタフェース、クラスおよび例外リストを示します。

付録 C「BooksOnLine 用スクリプト」
この付録では、サンプル・アプリケーション「BooksOnLine」で使用するスクリプトを示します。

付録 D「JMS および AQ XML サーブレット・エラー・メッセージ」
この付録では、エラー・メッセージのリストを示します。

付録 E「Unified Modeling Language 図」
この付録では、利用図および UML 表記の概要を説明します。

関連文書

詳細は、次の Oracle リソースを参照してください。

- 『Oracle9*i* アプリケーション開発者ガイド - 基礎編』
- 『PL/SQL ユーザーズ・ガイドおよびリファレンス』
- 『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』
- 『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』

ドキュメント・セットの多くのマニュアルで、Oracle のインストール時にデフォルトとしてインストールされるシード・データベースのサンプル・スキーマを使用しています。スキーマの作成方法および使用方法の詳細は、『Oracle9*i* サンプル・スキーマ』を参照してください。

リリース・ノート、インストール・マニュアル、ホワイト・ペーパーまたはその他の関連文書は、OTN-J（Oracle Technology Network Japan）に接続すれば、無償でダウンロードできます。OTN-J を使用するには、オンラインでの登録が必要です。次の URL で登録できます。

`http://otn.oracle.co.jp/membership/`

すでに OTN-J のユーザー名およびパスワードを取得済であれば、次の OTN-J Web サイトの文書セクションに直接接続できます。

`http://otn.oracle.co.jp/document/`

表記規則

この項では、このマニュアルの本文およびコード例で使用される表記規則について説明します。この項の内容は次のとおりです。

- 本文中の表記規則
- コード例中の表記規則
- Windows オペレーティング・システムの表記規則

本文中の表記規則

本文では、特別な用語をより迅速に識別できるように、様々な表記規則を使用します。次の表に、それらの表記規則を説明し、その使用例を示します。

規則	意味	例
太字	太字は、本文中で定義されている用語または用語集に記載されている用語（あるいはその両方）を示します。	この句を指定すると、**索引構成表**が作成されます。
固定幅フォントの大文字	固定幅フォントの大文字は、システムが提供する要素を示します。このような要素には、パラメータ、権限、データ型、Recovery Manager キーワード、SQL キーワード、SQL*Plus またはユーティリティ・コマンド、パッケージおよびメソッドが含まれます。また、システムが提供する列名、データベース・オブジェクト、データベース構造、ユーザー名およびロールも含まれます。	NUMBER 列に対してのみに、この句を指定できます。 BACKUP コマンドを使用して、データベースのバックアップを取ることができます。 USER_TABLES データ・ディクショナリ・ビュー内の TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
固定幅フォントの小文字	固定幅フォントの小文字は、実行可能ファイル、ファイル名、ディレクトリ名およびユーザーが提供する要素のサンプルを示します。このような要素には、コンピュータ名およびデータベース名、ネット・サービス名および接続識別子が含まれます。また、ユーザーが提供するデータベース・オブジェクトとデータベース構造と列名、パッケージとクラス、ユーザー名とロール、プログラム・ユニットおよびパラメータ値も含まれます。 **注意：**大文字と小文字を組み合せて使用するプログラム要素もあります。これらの要素は、記載されているとおり入力してください。	sqlplus と入力して、SQL*Plus をオープンします。 パスワードは、orapwd ファイルで指定します。 /disk1/oracle/dbs ディレクトリ内のデータ・ファイルおよび制御ファイルのバックアップを取ります。 hr.departments 表には、department_id、department_name および location_id 列があります。 QUERY_REWRITE_ENABLED 初期化パラメータを true に設定します。 oe ユーザーとして接続します。 JRepUtil クラスが次のメソッドを実装します。

規則	意味	例
固定幅フォントの小文字のイタリック	固定幅フォントの小文字のイタリックは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。ここで、*old_release* とはアップグレード前にインストールしたリリースを示します。

コード例中の表記規則

コード例は、SQL、PL/SQL、SQL*Plus または他のコマンドライン文を説明します。コード例は、固定幅フォントで表示され、この例に示すとおり通常のテキストと区別されます。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表に、コード例で使用される表記規則を説明し、その使用例を示します。

規則	意味	例
[ ]	大カッコは、任意に選択する 1 つ以上の項目を囲みます。大カッコは、入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコは、2 つ以上の項目を囲み、そのうちの 1 つの項目は必須です。中カッコは、入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内の 2 つ以上のオプションの選択項目を表します。オプションのうちの 1 つを入力します。縦線は、入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のいずれかを示します。 ■ 例に直接関連しないコードの一部が省略されている。 ■ コードの一部を繰り返すことができる。	CREATE TABLE ... AS *subquery*; SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関連しない複数の行が省略されていることを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.

規則	意味	例
その他の句読点	大カッコ、中カッコ、縦線および省略記号以外の句読点は、表示されているとおり入力する必要があります。	`acctbal NUMBER(11,2);` `acct  CONSTANT NUMBER(4) := 3;`
イタリック体	イタリック体は、特定の値を指定する必要があるプレースホルダや変数を示します。	`CONNECT SYSTEM/`*`system_password`* `DB_NAME = `*`database_name`*

大文字	大文字は、システムが提供する要素を示します。これらの用語は、ユーザー定義の用語と区別するために大文字で示されます。用語が大カッコ内にないかぎり、表示されているとおりの順序および綴りで入力します。ただし、これらの用語は大 / 小文字が区別されないため、小文字でも入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザー定義のプログラム要素を示します。たとえば、表名、列名またはファイル名などです。
	注意 : 大文字と小文字を組み合せて使用するプログラム要素もあります。これらの要素は、記載されているとおり入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Windows オペレーティング・システムの表記規則

次の表に、Windows オペレーティング・システムの表記規則を説明し、その使用例を示します。

規則	意味	例
「スタート」>	プログラムを起動する方法を示します。	Database Configuration Assistant を起動するには、「スタート」>「プログラム」>「Oracle - *HOME_NAME*」>「Configuration and Migration Tools」>「Database Configuration Assistant」を選択します。
ファイル名およびディレクトリ名	ファイル名およびディレクトリ名は大 / 小文字が区別されません。特殊文字の左山カッコ（<）、右山カッコ（>）、コロン（:）、二重引用符（"）、スラッシュ（/）、縦線（|）およびダッシュ（-）は使用できません。円記号（¥）は、引用符で囲まれている場合でも、要素のセパレータとして処理されます。Windows では、ファイル名が ¥¥ で始まる場合、汎用ネーミング規則が使用されていると想定されます。	`c:¥winnt"¥"system32` は `C:¥WINNT¥SYSTEM32` と同じです。

規則	意味	例
C:¥>	現在のハードディスク・ドライブの Windows コマンド・プロンプトを表します。コマンド・プロンプトのエスケープ文字はキャレット（^）です。プロンプトには、作業中のサブディレクトリが表示されます。このマニュアルでは、コマンド・プロンプトと呼びます。	C:¥oracle¥oradata>
特殊文字	円記号（¥）は、Windows コマンド・プロンプトの二重引用符（"）のエスケープ文字として必要な場合があります。カッコおよび一重引用符（'）にはエスケープ文字は必要ありません。エスケープ文字および特殊文字の詳細は、Windows オペレーティング・システムのドキュメントを参照してください。	C:¥>exp scott/tiger TABLES=emp QUERY=¥"WHERE job='SALESMAN' and sal<1600¥" C:¥>imp SYSTEM/*password* FROMUSER=scott TABLES=(emp, dept)
HOME_NAME	Oracle ホームの名前を表します。ホーム名には、英数字で 16 文字まで使用できます。ホーム名に使用可能な特殊文字は、アンダースコアのみです。	C:¥> net start Oracle*HOME_NAME*TNSListener

規則	意味	例
ORACLE_HOME および *ORACLE_BASE*	Oracle8*i* リリース 8.1.3 より前のリリースでは、Oracle コンポーネントをインストールすると、すべてのサブディレクトリが最上位の *ORACLE_HOME* ディレクトリ（デフォルト名は次のいずれか）に配置されました。 ■ C:¥orant（Windows NT の場合） ■ C:¥orawin98（Windows 98 の場合） 今回のリリースは、Optimal Flexible Architecture（OFA）のガイドラインに準拠します。最上位の *ORACLE_HOME* ディレクトリに配置されないサブディレクトリもあります。*ORACLE_BASE* と呼ばれる最上位のディレクトリ（デフォルトは C:¥oracle）があります。他の Oracle ソフトウェアがインストールされていないコンピュータに Oracle9*i* リリース 1（9.0.1）をインストールした場合、最初の Oracle ホーム・ディレクトリは、デフォルトで C:¥oracle¥ora90 に設定されます。Oracle ホーム・ディレクトリは、*ORACLE_BASE* の直下に配置されます。 このマニュアルに示すすべてのディレクトリ・パスの例は、OFA の表記規則に準拠しています。	%*ORACLE_HOME*%¥rdbms¥admin ディレクトリへ移動します。

アドバンスト・キューイングの新機能

ここでは、Oracle9*i* 以下のリリースにおけるアドバンスト・キューイングの新機能について説明します。

内容は次のとおりです。

- Oracle9i リリース 2（9.2.0）の新機能
- Oracle9i リリース 1（9.0.1）の新機能
- Oracle8i での新機能

Oracle9*i* リリース 2（9.2.0）の新機能

- Oracle Messaging Gateway

  異なるメッセージ・システム間での相互通信は、一般的な統合要件です。メッセージ・ゲートウェイによって、アドバンスト・キューイングで Oracle 以外のメッセージ・システムとメッセージを双方向に伝播できます。これによって、Oracle Advanced Queuing と IBM MQSeries v5.1 および v5.2 間で、安全性が高く、トランザクション型の、保証された 1 回のみのメッセージ配信が可能になります。詳細は、第 18 章「メッセージ・ゲートウェイ」を参照してください。

- 標準 JMS のサポート

  Oracle の JMS 実装は、Sun 社の JMS 1.0.2b 標準に準拠しています。12-3 ページの「J2EE 準拠」を参照してください。

- `XMLType` ペイロードのサポート

  Oracle オブジェクト型に `XMLType` 属性を埋め込む必要がありません。`XMLType` メッセージをメッセージ・ペイロードとして直接使用できます。

Oracle9*i* リリース 1（9.0.1）の新機能

Oracle9*i* には、E-Business の統合を改善し、標準のインターネット転送プロトコルを使用するための新しいアドバンスト・キューイング機能が追加されました。

- インターネットの統合

  アドバンスト・キューイングでは、インターネットを経由してキューイング操作を実行するために、XML を使用してメッセージ構造を定義する iDAP を利用します。iDAP を使用すると、エンキュー、デキュー、通知、伝播などの AQ 操作を、標準のインターネット転送プロトコルである HTTP を使用して実行できます。メッセージ機能を提供するサード・パーティ・ベンダーなどのサード・パーティ・クライアントも、メッセージ・ゲートウェイを使用してインターネット経由で AQ と相互運用できます。

  iDAP メッセージは、リクエスト、レスポンスまたはエラー・レスポンスのいずれかです。AQ クライアントから送信される iDAP ドキュメントには、操作データと一緒に、エンキュー、デキュー、登録などの、リモート操作を指定するための属性が含まれます。iDAP の AQ 実装は、メッセージのエンキューおよびデキューのバッチ処理実行にも使用できます。

  AQ での HTTP のサポートは、Oracle データベース・サーバーにバンドルされている AQ サーブレットを使用して実装されます。クライアントは、Web サーバーに送信される HTTP の POST リクエストを介してサーブレットを起動します。Web サーバーは、サーブレットが起動していない場合、POST メソッドに指定されたサーブレットを起動します。サーブレットは、iDAP ドキュメントの内容を解析し、AQ の Java Application Program Interface（API）を使用して、指定された操作を実行します。コールが完了すると、サーブレットは iDAP の指定に従ってレスポンスかエラー・レスポンスのいずれかをフォーマットし、クライアントに戻します。

iDAP はトランスポートに依存しないため、他の転送プロトコルと透過的に動作できます。Oracle9*i* は、HTTP をサポートします。また、変換を介したコールアウト・メカニズムを使用して、他の独自のプロトコルもサポートできます。

- インターネット上でのアドバンスト・キューイングのセキュリティ

  AQ 機能は、権限があるインターネット・ユーザーにのみ AQ キューに対する AQ 操作の実行を許可します。インターネット・ユーザーは Web サーバーに接続します。接続された Web サーバーは、アプリケーション・サーバーを使用してデータベースに接続します。操作を実行しているインターネット・ユーザーは、通常、データベースに接続しているデータベース・ユーザーではありません。また、AQ キューは、データベースの接続ユーザーと同じスキーマに常駐しない場合があります。アドバンスト・キューイングでは、権限があるインターネット・ユーザーのみが AQ キューに対する AQ 操作を実行できるように、プロキシ認証が使用されます。

- LDAP の統合

  Oracle Internet Directory（OID）の統合 : アドバンスト・キューイングは、LDAP を使用して一般情報を集中管理するために、Oracle Internet Directory（OID）サーバーに統合されています。これによって、次の要件が満たされます。

  - グローバル・トピック（キュー）: AQ キュー情報を OID サーバーに格納できます。OID は、要求されるトピックまたはキューの位置を識別するための、単一の接続点として機能します。特定の情報を検索するビジネス・アプリケーション（ユーザー）は、どのデータベースにキューが位置しているかを認識する必要はありません。ユーザーは、業界標準の Java Message Service（JMS）API を使用してキューに直接接続できます。データベース、またはトピックやキューの位置を明示的に指定する必要はありません。

  - グローバル・イベント : OID は、イベント登録のためのリポジトリとして使用できます。クライアントは、データベースが停止していても、データベース・イベントを登録できます。これによって、クライアントは「データベースのオープン」などのイベントを登録できます。これは、以前のリリースではできませんでした。クライアントは、1 回のリクエストで、複数のデータベースのイベントを登録できます。

  XML の統合 : XML は、E-Business におけるデータ表現の標準として誕生しました。Oracle サーバーには、XML データに対する操作をサポートするために、`XMLType` データ型が追加されています。AQ は、`XMLType` データ型のペイロードをサポートするのみでなく、XML メッセージの内容に基づいたサブスクリプションの定義も可能にします。これは、複数のベンダーが注文の内容に基づいてサブスクリプションを定義できるため、オンライン市場にとっては強力な機能です。

- 変換のインフラストラクチャ

  複数のアプリケーションは、相互依存しないように設計されています。したがって、各アプリケーションが認識するメッセージは異なります。これらのアプリケーションを統合するには、メッセージを変換する必要があります。これらの変換を行うには、様々な方法があります。AQ が提供する変換インフラストラクチャを使用すると、Oracle Application Interconnect、または Mercator などのサード・パーティが提供するソリューションの変換機能をプラグインし、同時に AQ の機能も保持できます。変換は、PL/SQL コールバック関数として指定することもできます。これらの関数は、メッセージのエンキュー、デキューまたは伝播に適用できます。また、C、Java または PL/SQL で実装されるサード・パーティの関数もコールできます。XSLT 変換も、XML メッセージに対して指定できます。

- AQ の管理

  拡張した Oracle Enterprise Manager の次の新機能を使用して、アドバンスト・キューイングを管理できます。

  - 向上した UI タスク・フローおよびキューの管理。データベース・レベルおよびキュー・レベルでのトポロジの表示、データベース内のすべてのキューに対するエラー・スケジューリングおよび伝播スケジューリング、適切な初期化パラメータ（`init.ora`）などが使用できます。
  - メッセージ・キューの表示機能

  Oracle Diagnostics Pack および Oracle Tuning Pack は、警告および AQ キューの監視を行います。警告は、特定のサブスクライバに対するメッセージ数がしきい値を超えたときに送信されます。また、伝播にエラーがあるときにも送信されます。さらに、READY 状態のメッセージ数またはサブスクライバごとのメッセージ数についてキューを監視できます。

- その他の拡張

  PL/SQL 通知および電子メール通知 : Oracle9*i* では、キューに関する通知に PL/SQL ファンクションを使用できます。この機能によって、ユーザーは、対象のメッセージがエンキューされたときにコールされる PL/SQL ファンクションを登録できます。電子メール通知機能を使用すると、通知の送信先の電子メール・アドレスを登録できます。電子メールは、対象のメッセージがキューに到着すると送信されます。電子メールの表示も、電子メール通知の登録中に指定できます。ユーザーは、通知の送信先として HTTP URL も指定できます。

  デキューの拡張 : サブスクライバは、デキューを条件機能とともに使用して、指定した条件を満たすメッセージを使用可能なメッセージの中から選択できます。

  全体的なパフォーマンスの向上 : AQ は、コードの最適化およびその他の変更によって、全体的なパフォーマンスが向上しています。

  伝播の拡張 : Oracle9*i* では、ジョブ・キュー・プロセスの最大数が 36 から 1000 に増加しています。インターネット伝播を使用して、HTTP を介したキュー間の伝播を設定で

きます。伝播アルゴリズムに関する設計変更によって、伝播の全体的なパフォーマンスが向上しています。

- JMS の拡張

  Oracle9*i* のすべての新機能は、次の機能と同様に、JMS を介してサポートされます。

  - コネクション・プーリング : コネクションのプールを Oracle データベース・サーバーで確立できます。後で JMS セッションを確立するときに、このプールからコネクションを選択できます。
  - グローバル・トピック : OID との統合によって実現されます。AQ のキュー情報を OID に格納し、OID から検索できます。
  - トピックのブラウズ : 永続サブスクライバが、パブリッシュ・サブスクライブ（トピック）の宛先にあるメッセージをブラウズできます。また、オプションでこれらのサブスクライバは、ブラウズしたメッセージを削除できます（これらのメッセージはそのサブスクライバに対して AQ で保持されなくなります）。
  - 例外リスナーのサポート : 問題が非同期でクライアントに通知されます。一部のコネクションはメッセージのみを処理するため、コネクションが失敗したことを認識する方法は他にありません。

Oracle8*i* での新機能

Oracle8*i* では、次のアドバンスト・キューイング機能が追加されました。

- キュー・レベルのアクセス制御
- 非永続キュー
- Oracle Parallel Server のサポート
- パブリッシュ・サブスクライブに対するルールベースのサブスクライバ
- 非同期通知
- 送信者識別
- リスニング機能（複数のキューでの待機）
- LOB を伴うメッセージの伝播
- 伝播スケジュールの拡張
- メッセージ・ヘッダーのみのデキュー
- 統計ビューのサポート
- Java API（ネイティブ AQ）
- Java Messaing Service（JMS）API
- 履歴管理情報の記憶域の分離

1

Oracle Advanced Queuing の概要

この章では、Oracle Advanced Queuing（AQ）、および統合環境での複雑な情報処理の要件について説明します。内容は次のとおりです。

- アドバンスト・キューイングの概要
- アドバンスト・キューイングの一般機能
- エンキュー機能
- デキュー機能
- 伝播機能
- アドバンスト・キューイングの要素
- JMS 用語
- デモ

アドバンスト・キューイングの概要

Web ベースのビジネス・アプリケーションが相互に通信する場合、プロデューサ・アプリケーションがメッセージをエンキューし、コンシューマ・アプリケーションがメッセージをデキューします。アドバンスト・キューイングは、データベースと統合されたメッセージ・キューイング機能を提供します。また、Oracle データベースの機能を有効活用して、メッセージを永続的に格納し、異なるマシンおよびデータベース上のキュー間で伝播させ、Oracle Net Services および HTTP(S) を使用して転送します。

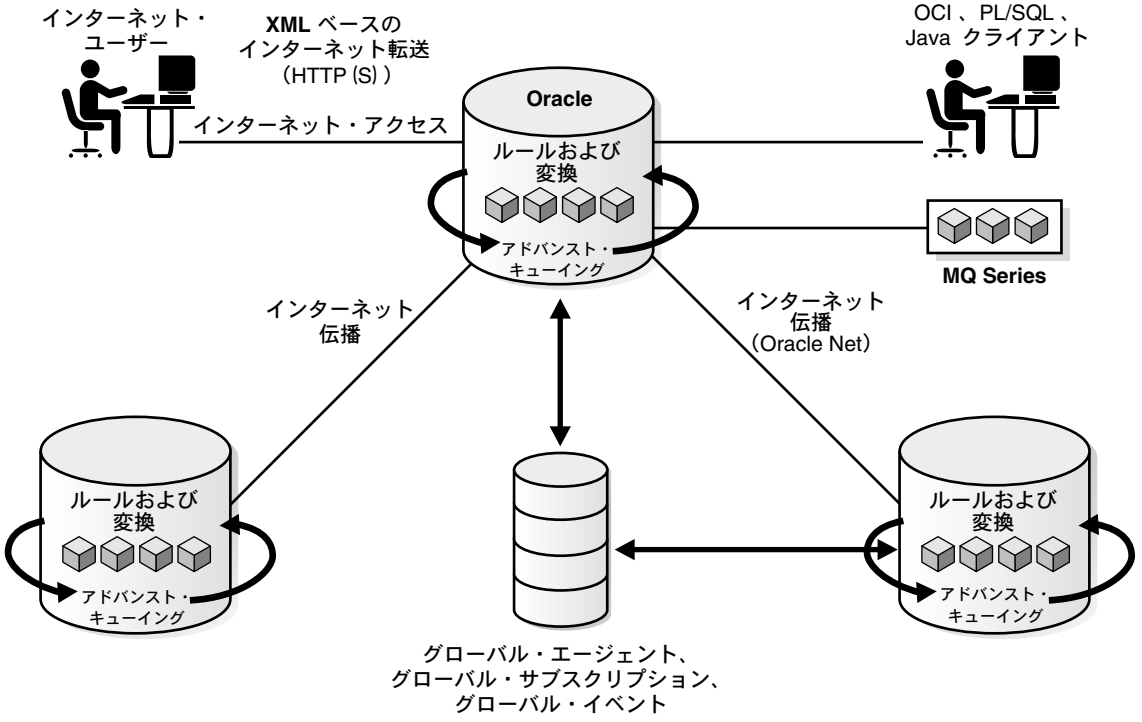
Oracle Advanced Queuing は、データベース表内に実装されるため、高可用性、拡張性および信頼性という操作上のすべてのメリットが、キュー・データに適用されます。アドバンスト・キューイングでは、リカバリ、再起動、セキュリティなどの標準データベース機能がサポートされます。また、キュー表をインポートおよびエクスポートできます。詳細は、第 4 章「AQ の管理」を参照してください。また、Oracle Enterprise Manager などのデータベース開発ツールおよびデータベース管理ツールを使用して、キューを監視することもできます。4-8 ページの「Oracle Enterprise Manager のサポート」を参照してください。

統合アプリケーション環境でのアドバンスト・キューイング

アドバンスト・キューイングは、アプリケーション統合に必要なメッセージ管理機能および非同期通信を提供します。統合環境では、メッセージは、図 1-1 に示すとおり、Oracle データベース・サーバーと、アプリケーションおよびユーザー間を移動します。Oracle Net Services を使用して、クライアントと Oracle データベース間または 2 つの Oracle データベース間でメッセージが交換されます。また、Oracle Net Services は、ある Oracle キューから別のキューにメッセージを伝播します。または、図 1-1 に示すとおり、HTTP、HTTPS などの転送プロトコルを使用して、インターネット経由でアドバンスト・キューイング操作を実行できます。この場合、クライアント、ユーザーまたはインターネット・アプリケーションは構造化 XML メッセージを生成します。インターネット経由での伝播中、Oracle サーバーも構造化 XML を使用して通信します。インターネットとアドバンスト・キューイングの統合の詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

アプリケーションの統合には、異機種間メッセージ・システムの統合も含まれます。AQ によって、IBM MQSeries などの Oracle 以外の既存のメッセージ・システムとの統合がメッセージ・ゲートウェイを介して透過的に行われるため、MQSeries ベースの既存のアプリケーションを Oracle AQ 環境に統合できます。AQ と Oracle 以外のメッセージ・システムとの統合の詳細は、第 18 章「メッセージ・ゲートウェイ」を参照してください。

図 1-1 AQ を使用した統合アプリケーション環境



AQ に対するインタフェース

アドバンスト・キューイング機能は、次のインタフェースを介して使用できます。

- DBMS_AQ、DBMS_AQADM および DBMS_AQELM を使用した PL/SQL（『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照）
- Oracle Object for OLE（OO4O）を使用した Visual Basic（Oracle Object for OLE に関するオンライン・ヘルプを参照）
- Java パッケージ oracle.AQ を使用した Java（『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照）
- Java パッケージ oracle.jms を使用した JMS（『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照）
- HTTP および HTTPS を使用したインターネット・アクセス

キューイング・システムの要件

アドバンスト・キューイングは、キューイング・システムのパフォーマンス、拡張性および永続性の要件を満たしています。詳細は、第 5 章「パフォーマンスおよび拡張性」を参照してください。

パフォーマンス

「サービスに対するリクエスト」と「サービスの供給」を分離することによって、効率を向上させ、複雑なスケジューリングに対するインフラストラクチャを提供する必要があります。アドバンスト・キューイングは、次の基準で測定した場合、高いパフォーマンス特性を示します。

- 1 秒間にエンキュー / デキューされるメッセージ数
- メッセージ・ウェアハウスに対する複雑な問合せの評価にかかる時間
- 障害後にメッセージ処理のリカバリ / 再起動にかかる時間

拡張性

キューイング・システムはスケーラブルである必要があります。アプリケーションを使用するプログラム数が増加しても、メッセージ数が増加しても、またメッセージ・ウェアハウスのサイズが増加しても、アドバンスト・キューイングは高いパフォーマンスを示します。

セキュリティのための永続性

ネットワーク、マシンおよびアプリケーションに障害が発生したときに、遅延実行を正常に動作させるためには、サービスに対するリクエストで構成されるメッセージが永続的に格納され、確実に 1 回のみ処理される必要があります。アドバンスト・キューイングは、次の場合に要件を満たすことができます。

- アプリケーションに、外部クライアントまたは内部プログラムから同時に到着する、複数の未処理メッセージを処理するためのリソースがない場合。
- データベース間の通信リンクが、常に使用可能なわけではなく、他の用途に確保されている場合。システム容量が不足しているためにメッセージをすぐに処理できない場合、アプリケーションは、処理可能になるまでそのメッセージを保存しておく必要があります。
- 外部クライアントまたは内部プログラムが、処理済メッセージを受信する準備ができていない場合。

スケジューリングのための永続性

キューイング・システムには、優先順位を扱えるメッセージの永続性が必要です。後から到着したメッセージが先に到着したメッセージより高い優先順位を持つ場合、先に到着したメッセージが後のメッセージを待ってからアクションを実行する場合、同一のメッセージが異なるプロセスからアクセスされる場合などです。優先順位も変更されます。特定のキューにあるメッセージがより重要になり、遅延や他のキューのメッセージからの介入が少なくなるような処理が必要な場合があります。同様に、ある受信者へのメッセージ送信が、他の受信者への送信より優先順位が高くなる場合があります。

メタデータのアクセスおよび分析のための永続性

メッセージのメタデータがペイロード・データと同様に重要になる場合があるため、メッセージのメタデータを保存できるようなメッセージの永続性が必要です。たとえば、メッセージの受信時刻またはディスパッチ時刻が、ビジネスや正当な理由から重要になることがあります。アドバンスト・キューイングの永続性機能を使用すると、最大需要の周期の分析、または注文を受信してから処理を完了するまでのタイムラグの評価を行うことができます。

アドバンスト・キューイングの一般機能

内容は次のとおりです。

- Point-to-Point およびパブリッシュ・サブスクライブ・メッセージ機能
- Oracle Internet Directory
- Oracle Enterprise Manager の統合
- メッセージ・フォーマットの変換
- SQL アクセス
- 統計ビューのサポート
- 構造化ペイロード
- 保存およびメッセージ履歴
- 追跡およびイベント・ジャーナル
- キュー・レベルのアクセス制御
- 非永続キュー
- Oracle9i Real Application Clusters のサポート
- XMLType ペイロード
- インターネット統合および Internet Data Access Presentation

使用例については、第 8 章「AQ を使用したサンプル・アプリケーション」を参照してください。ここでは、架空のオンライン書籍販売店である BooksOnLine のメッセージ・システムについて説明しています。BooksOnLine の例では、ここで説明する多くの機能を示しています。

Point-to-Point およびパブリッシュ・サブスクライブ・メッセージ機能

いくつかの機能によって、アプリケーション間でパブリッシュ・サブスクライブ・メッセージ機能が実現されています。この機能とは、ルールベースのサブスクライバ、メッセージ伝播、リスニング機能、通知機能などです。

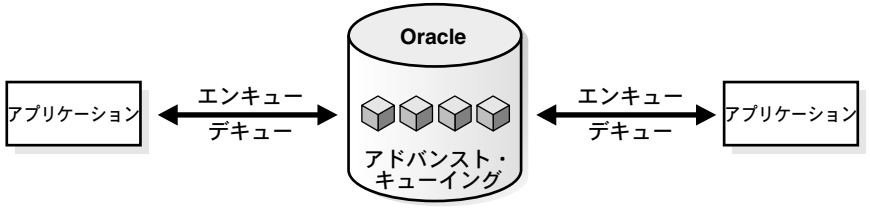
アドバンスト・キューイングは、次の方法でメッセージを送受信します。

- Point-to-Point
- パブリッシュ・サブスクライブ

Point-to-Point

Point-to-Point メッセージは、特定のターゲットを対象としています。送信者および受信者は、メッセージ交換に使用する共通キューを決定します。各メッセージは、1 人の受信者のみによって使用されます。図 1-2 に、各アプリケーションが独自のメッセージ・キュー（**シングル・コンシューマ・キュー**）**を持つことを示します。**

図 1-2 Point-to-Point メッセージ機能



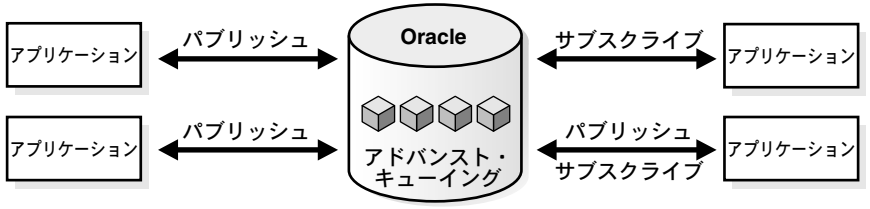
パブリッシュ・サブスクライブ

パブリッシュ・サブスクライブ・メッセージは、図 1-3 に示すとおり、複数の受信者によって使用される場合があります。パブリッシュ・サブスクライブ・メッセージ機能には、広範囲なモード（ブロードキャスト）および目標範囲がより狭いモード（マルチキャスト）があります。マルチキャストは、Point-to-Multipoint ともいいます。

ブロードキャストは、ある番組の聴取者を正確に知らないラジオ放送局と同じです。**サブスクライバ**から**マルチ・コンシューマ・キュー**へデキューされます。それとは対照的に、マルチキャストは、購読者を把握している出版社と同じです。マルチキャストは、単一のパブリッシャが複数の受信者にメッセージを送信するため、Point-to-Multipoint ともいいます。

この**受信者**は、交換メカニズムとして機能するキューのサブスクライバである場合と、そうでない場合があります。

図 1-3 パブリッシュ・サブスクライブ・モード



Oracle Internet Directory

Oracle Internet Directory は、Oracle データベースに組み込まれたシステム固有の LDAPv3 ディレクトリ・サービスで、電子メール・アドレス、電話番号、パスワード、セキュリティ証明書、様々なタイプのネットワーク・デバイスの構成データなど、多様な情報を集中管理します。企業全体のキューイング情報（キュー、サブスクリプションおよびイベント）は、Oracle Internet Directory という 1 つの場所から検索できます。詳細は、『Oracle Internet Directory 管理者ガイド』を参照してください。

Oracle Enterprise Manager の統合

Enterprise Manager を使用して、次の操作を実行できます。

- キュー、キュー表、伝播スケジュールおよび変換の作成および管理。
- データベース・レベルおよびキュー・レベルでの AQ トポロジの使用、およびキュー・エラー、キュー統計およびセッション統計の参照による AQ 環境の監視。4-8 ページの「Oracle Enterprise Manager のサポート」を参照してください。

メッセージ・フォーマットの変換

メッセージ・フォーマットの変換機能によって、異なるフォーマットのデータを使用するアプリケーションがサポートされます。変換は、1 つの Oracle データ型から別のデータ型へのマッピングを定義します。変換は、ソース・データ型を入力として取得し、ターゲット・データ型のオブジェクトを戻す SQL ファンクションによって表現されます。

変換は次のとおり指定できます。

- エンキュー時に、メッセージをキューに挿入する前に適切な型に変換します。

  メッセージは、エンキュー時にキューのペイロード型に変換できます。したがって、エンキューされるメッセージの型が、キューのペイロード型と一致している必要はありません。

- デキュー時に、メッセージを任意のフォーマットで受信します。

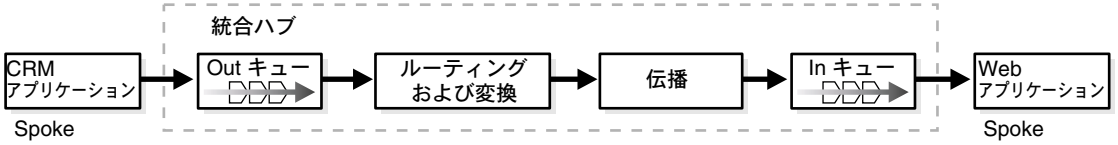
  メッセージは、デキュー元に戻す前に任意のフォーマットに変換できます。

- リモート・サブスクライバは、ソース・キューのフォーマットとは異なるフォーマットでメッセージを受信できます。

  メッセージは、リモート・サブスクライバに伝播される前に、リモート・サブスクライバがキューへのサブスクライブ時に指定した変換に基づいて変換されます。

図 1-4 に示すとおり、キューイング機能、ルーティング機能および変換機能は、統合アプリケーション・アーキテクチャに不可欠な構成ブロックです。この図は、CRM アプリケーションの Out キューのデータが統合ハブで転送および変換され、その後、Web アプリケーションの In キューに伝播される方法を示しています。変換エンジンは、メッセージを Out キューのフォーマットから In キューのフォーマットにマップします。

図 1-4 アプリケーション統合における変換



詳細は、8-5 ページの「メッセージ・フォーマットの変換」を参照してください。

SQL アクセス

メッセージはデータベース表の標準の行に置かれるため、標準 SQL で問い合せることができます。ユーザーは SQL を使用して、メッセージのプロパティ、履歴およびペイロードにアクセスできます。また、SQL アクセスを使用すると、監査および追跡も行うことができます。索引など、使用可能なすべての SQL テクノロジを使用してメッセージへのアクセスを最適化できます。

統計ビューのサポート

GV$AQ ビューを使用すると、キューに関する基本的な統計が使用できます。

構造化ペイロード

ユーザーは、オブジェクト型を使用してメッセージ・ペイロードを構造化および管理できます。一般に RDBMS は、メッセージ・システムより豊富な型指定のシステムを備えています。Oracle はオブジェクト・リレーショナル DBMS であり、従来のリレーショナル型とユーザー定義型の両方をサポートします。強力な型指定を持つ内容（外部の型指定システムによって定義されたフォーマットを持つ内容など）によって、次のような多くの高性能な機能が使用できます。

- 内容ベースのルーティング : アドバンスト・キューイングが内容を調査し、その内容に基づいてメッセージを別のキューに自動的にルーティングできます。
- 内容ベースのサブスクリプション : パブリッシュおよびサブスクライブ・システムはメッセージ・システム上に組み込まれているため、ユーザーは内容に基づいてサブスクリプションを作成できます。
- 問合せ : メッセージ内容の問合せ機能によって、メッセージ・ウェアハウスが可能です。

この機能を BooksOnLine のシナリオに適用した場合については、8-11 ページの「構造化ペイロード」を参照してください。

保存およびメッセージ履歴

システム管理者は、処理済のメッセージの保存期間を指定します。アドバンスト・キューイングは、各メッセージの履歴情報を格納し、キューおよびメッセージのプロパティ（遅延、期限切れ、およびローカルまたはリモートの受信者に対するメッセージの保存）を保存します。この情報には、エンキューおよびデキュー時刻、および各リクエストを実行したトランザクションの識別子が含まれます。これによって、ユーザーは関連するメッセージの履歴を保持できます。この履歴は、追跡、データ・ウェアハウスおよびデータ・マイニングの各操作の他に、特定の監査機能で使用できます。

この機能を BooksOnLine のシナリオに適用した場合については、8-26 ページの「保存およびメッセージ履歴」を参照してください。

追跡およびイベント・ジャーナル

保存されているメッセージは、相互に関連付けができます。たとえば、メッセージ `m1` を処理した結果、メッセージ `m2` が生成された場合、`m1` は `m2` に関連付けられます。これによって、ユーザーは関連するメッセージの順序を追跡できます。これらの順序はイベント・ジャーナルを表しており、アプリケーションによって作成される場合があります。アドバンスト・キューイングでは、アプリケーションが自動的にイベント・ジャーナルを作成できるように設計されています。

オンライン注文が発生すると、注文処理に関連する様々なアプリケーションによって複数のメッセージが生成されます。アドバンスト・キューイングには、メッセージを生成したアプリケーションとは独立して、相互に関連付けられたメッセージを追跡する機能があります。

ユーザーは、どのユーザーがメッセージをエンキューおよびデキューしたか、そのユーザーはだれか、およびどのユーザーがどの操作を行ったかを判断できます。

アドバンスト・キューイングの追跡機能を使用すると、SQL の SELECT 文および JOIN 文を使用して、AQ$< キュー表名 >、および ENQ_TRAN_ID ビュー、DEQ_TRAN_ID ビュー、USER_DATA ビュー（ペイロード）、CORR_ID ビューおよび MSG_ID ビューから注文情報を取得できます。これらのビューには、追跡に使用される次のデータが含まれます。

- エンキューおよびデキュー時に取得されたトランザクション ID（ENQ_TRAN_ID ビューおよび DEQ_TRAN_ID ビュー）
- メッセージ・プロパティの一部である相関識別子（CORR_ID ビュー）
- 追跡に使用できるメッセージ内容（USER_DATA ビュー）

キュー・レベルのアクセス制御

8.1 形式のキューの所有者は、キュー・レベルの権限を付与したり取り消すことができます。データベース管理者は、すべてのデータベース・ユーザーに、新しい AQ システム・レベル権限を付与したり取り消すことができます。また、データベース管理者は、任意のデータベース・ユーザーを AQ 管理者にすることもできます。

この機能を BooksOnLine のシナリオに適用した場合については、8-4 ページの「キュー・レベルのアクセス制御」を参照してください。

非永続キュー

アドバンスト・キューイングでは、サブスクライバに非永続メッセージを非同期に配信できます。これらのメッセージはイベント駆動方式の可能性があり、システム（またはインスタンス）に障害が発生すると保持されません。アドバンスト・キューイングは、1 つの共通 API で永続および非永続メッセージをサポートします。

この機能を BooksOnLine のシナリオに適用した場合については、8-16 ページの「非永続キュー」を参照してください。

Oracle9*i* Real Application Clusters のサポート

アプリケーションでキュー表のインスタンス・アフィニティを指定できます。アドバンスト・キューイングを Real Application Clusters および複数インスタンスで使用する場合、この情報を使用することにより、インスタンス間でキュー表をパーティション化して、キュー・モニター・スケジューリングを行うことができます。キュー表は、ユーザーが指定したインスタンスのキュー・モニターによって監視されます。インスタンス・アフィニティが指定されないと、キュー表は、使用可能なインスタンス間で任意にパーティション化されることになります。キュー表にアクセスするアプリケーションとそれを監視するキュー・モニターの間で、ping が発生する可能性があります。インスタンス・アフィニティを指定しても、そのアプリケーションが他のインスタンスから、キュー表とそのキューにアクセスできなくなることはありません。

この機能によって、キュー・モニターと別のインスタンスで実行しているアドバンスト・キューイング伝播ジョブの間で ping が発生しなくなります。互換性を Oracle8*i* リリース 8.1.5 以上に設定すると、キュー表に対してインスタンス・アフィニティ（プライマリおよびセカンダリ）を指定できます。アドバンスト・キューイングを Real Application Clusters および複数インスタンスで使用する場合、この情報を使用して、伝播用のインスタンスのみでなく、キュー・モニター・スケジューリング用のインスタンス間でもキュー表をパーティション化できます。キュー表は、常に 1 つのインスタンスと関連付けられています。明示的に指定されたアフィニティがない場合、使用可能な任意のインスタンスがキュー表の所有者となる可能性があります。キュー表の所有者がなくなると、セカンダリ・インスタンスまたはいずれかの使用可能なインスタンスがキュー表の所有権を引き継ぎます。

この機能を BooksOnLine のシナリオに適用した場合については、8-29 ページの「Oracle Real Application Clusters のサポート」を参照してください。

XMLType ペイロード

新しい不透明な型である `XMLType` を使用するキューを作成できます。これらのキューは、XML 文書であるメッセージを転送および格納するために使用できます。`XMLType` を使用すると、次の操作を実行できます。

- 任意の型のメッセージをキューに格納する。
- ドキュメントを CLOB として内部的に格納する。
- 複数の型のペイロードをキューに格納する。
- `ExistsNode()` 演算子および `SchemaMatch()` 演算子を使用して `XMLType` 列を問い合せる。
- サブスクライバ・ルールまたはデキュー条件にその演算子を指定する。

インターネット統合および Internet Data Access Presentation

Simple Object Access Protocol（SOAP）を使用してインターネット経由で AQ にアクセスできます。Internet Data Access Presentation（iDAP）は、AQ 操作に対する SOAP 仕様です。iDAP によって SOAP リクエストの本体に XML メッセージ構造が定義されます。iDAP によって構造化されたメッセージは、HTTP などの転送プロトコルを使用してインターネット上で転送されます。詳細は、1-12 ページの「インターネット経由での伝播 : HTTP」および第 17 章「AQ へのインターネット・アクセス」を参照してください。

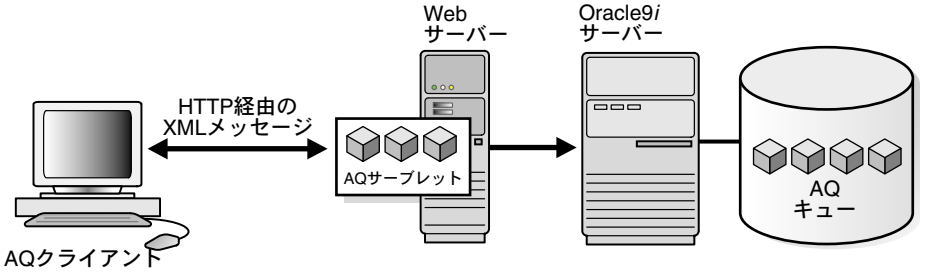
インターネット経由での伝播 : HTTP

図 1-5 に、HTTP 経由で AQ 操作を実行するためのアーキテクチャを示します。主要コンポーネントは次のとおりです。

- AQ クライアント・プログラム
- AQ サーブレットのホスト Web サーバー / サーブレット・コンテナ
- Oracle データベース・サーバー

AQ クライアント・プログラムは、XML メッセージ（iDAP 準拠）を AQ サーブレットに送信します。AQ サーブレットは、その XML メッセージを認識し、AQ 操作を実行します。Web ブラウザなど、任意の HTTP クライアントを使用できます。AQ サーブレットのホスト Web サーバー / サーブレット・コンテナは、着信 XML メッセージを解析します。これには、Apache JServ や Tomcat などがあります。AQ サーブレットは、Oracle データベース・サーバーに接続し、ユーザーのキューに対して操作を実行します。

図 1-5 HTTP を使用して AQ 操作を実行するためのアーキテクチャ



Internet Data Access Presentation（iDAP）

Internet Data Access Presentation（iDAP）は、Content-Type ヘッダーの `text/xml` を使用して、SOAP リクエストの本体を指定します。XML では、iDAP リクエストおよびレスポンス・メッセージは、次のように表現されます。

- すべてのリクエスト・タグおよびレスポンス・タグは、SOAP 名前空間のスコープにあります。
- AQ 操作は iDAP 名前空間のスコープにあります。
- 送信者は、SOAP 本体の iDAP 要素および属性に名前空間を含めます。
- 受信者は、適切な名前空間を含む iDAP メッセージを処理します。不適切な名前空間を含むリクエストの場合、受信者はリクエストが無効であるというエラーを戻します。
- SOAP 名前空間の値は `http://schemas.xmlsoap.org/soap/envelope/` です。
- iDAP 名前空間の値は `http://ns.oracle.com/AQ/schemas/access` です。

iDAP の詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

否認防止および AQ$< キュー表名 > ビュー

アドバンスト・キューイングでは、メッセージ自体とともに、メッセージ情報のすべての履歴が保持されます。AQ$< キュー表名 > ビューを使用して、履歴情報を検索できます。この情報によって、メッセージの送受信が証明され、送信者および受信者の否認防止に使用できます。AQ$< キュー表名 > ビューの詳細は、第 10 章「管理インタフェース : ビュー」を参照してください。

エンキュー時には、エンキュー元の否認防止のために次の情報が保持されます。

- エンキューを実行する AQ エージェント
- エンキューを実行するデータベース・ユーザー
- エンキュー時刻
- エンキューを実行するトランザクションの ID

デキュー時には、デキュー元の否認防止のために次の情報が保持されます。

- デキューを実行する AQ エージェント
- デキューを実行するデータベース
- デキュー時刻
- デキューを実行するトランザクションの ID

伝播後は、伝播の宛先キューの Original_Msgid フィールドが、ソース・メッセージのメッセージ ID に対応します。このフィールドは、伝播されたメッセージを相互に関連付けるために使用できます。これは、伝播されたメッセージのデキュー元の否認防止に有効です。

エンキュー時にメッセージとともに送信者のデジタル署名をエンキューし、デキュー時にデキュー元のデジタル署名を格納することによって、より強力な否認防止を実現できます。

エンキュー機能

次の機能は、メッセージのエンキューに適用されます。

相関識別子

ユーザーが各メッセージに識別子を割り当てることができるため、後で特定のメッセージを取り出せます。

サブスクリプション・リストおよび受信者リスト

単一のメッセージを、複数のコンシューマによって使用するように設計できます。キュー管理者は、キューからメッセージを取り出すことができるサブスクライバのリストを指定できます。異なるキューには異なるサブスクライバを指定でき、1 つのコンシューマ・プログラムが複数のキューのサブスクライバになることもできます。さらに、キュー内の特定のメッセージに特定の受信者（そのキューのサブスクライバであってもなくても可）を指定することで、サブスクライバ・リストをオーバーライドできます。

単一のメッセージを複数のコンシューマが様々な方法で処理するように設計できます。メッセージの取出しを許可されたコンシューマが、メッセージをエンキューしたユーザーまたはアプリケーションによって、メッセージの明示的な受信者として指定されます。明示的に指定されたすべての受信者はエージェントで、名前、アドレスおよびプロトコルによって識別されます。

キュー管理者は、特定のキューのメッセージをすべて取り出すことができる受信者のデフォルト・リストを指定できます。これらの暗黙的な受信者は、デフォルト・リストに指定されていることでキューのサブスクライバになります。受信者を明示せずにエンキューされたメッセージは、指定されているすべてのサブスクライバに配信されます。

ルールベースのサブスクライバは、デフォルトの受信者リストの中でルールを対応付けられている受信者です。受信者が明示されていないメッセージは、対応付けられたルールの評価結果が TRUE のときにのみ、ルールベースのサブスクライバに送信されます。異なるキューには異なるサブスクライバを指定でき、同じ受信者が複数のキューのサブスクライバになることも可能です。さらに、キュー内の特定のメッセージに特定の受信者（そのキューのサブスクライバであってもなくても可）を指定することで、サブスクライバ・リストをオーバーライドできます。

受信者は名前のみで指定できますが、その場合、受信者はそのメッセージがエンキューされたキューからデキューする必要があります。プロトコルの値が 0（ゼロ）の場合、名前およびアドレスで受信者を指定できます。アドレスは、同じデータベースまたは（データベース・リンクによって識別された）他の Oracle データベース中の別のキューの名前です。この場合、メッセージは指定されたキューに伝播され、指定された名前のコンシューマによってデキューできます。受信者の名前が NULL の場合、メッセージはアドレスに指定されたキューに伝播され、アドレスに指定されたキューのサブスクライバによってデキューされます。プロトコル・フィールドの値が 0（ゼロ）でない場合、名前およびアドレスのフィールドは無視され、メッセージは特定のコンシューマによってデキューできます。この機能を BooksOnLine のシナリオに適用した場合については、1-20 ページの「アドバンスト・キューイングの要素」を参照してください。

エンキューにおけるメッセージの優先順位および順序付け

エンキューされたメッセージの優先順位を指定できます。また、エンキューされたメッセージは、指定されたキュー内の正確な位置に置くことができます。つまり、ユーザーがメッセージの使用順序を指定するには、次の 3 つのオプションがあるということです。

- ソート順序は、どのプロパティを使用してキューのすべてのメッセージを順序付けるかを指定します。
- 優先順位は、各メッセージに割り当てられます。
- 順序逸脱は、メッセージの位置を他のメッセージとの相対で指定します。

さらに、複数のコンシューマが同一のキューを操作している場合、コンシューマは即時使用可能な最初のメッセージを取得します。別のコンシューマで処理中のメッセージはスキップされます。

この機能を BooksOnLine のシナリオに適用した場合については、8-36 ページの「メッセージの優先順位および順序付け」を参照してください。

メッセージのグループ化

1 つのキューに属しているメッセージをグループ化して 1 つのセットにし、1 回に 1 ユーザー以外は使用できないようにできます。このためには、メッセージのグループ化を使用可能にしたキュー表に、キューを作成する必要があります。1 つのグループに属するメッセージは、すべて同一のトランザクションで作成される必要があります。また、1 つのトランザクションで作成されるメッセージは、すべて同一のグループに属します。この機能によって、ユーザーは複雑なメッセージをセグメント化できます。たとえば、あるキュー宛てのメッセージに請求書情報が含まれている場合、そのメッセージは、ヘッダーのメッセージ、詳細情報のメッセージ、フッターのメッセージで構成されるグループとして作成できます。

この機能を BooksOnLine のシナリオに適用した場合については、8-49 ページの「メッセージのグループ化」を参照してください。

伝播

この機能によって、同じデータベースまたは同じキューに接続しなくても、アプリケーション同士が互いに通信できます。メッセージは、ローカルまたはリモートにかかわらず、ある Oracle AQ から別の Oracle AQ に伝播できます。伝播は、データベース・リンクおよび Oracle Net Services を使用して行われます。

この機能を BooksOnLine のシナリオに適用した場合については、8-102 ページの「伝播」を参照してください。

送信元の識別

アプリケーションは、送信メッセージにユーザー定義の識別マークを付加できます。Oracle は、メッセージがデキューされたキューを自動的に識別することもできます。これによって、伝播されたメッセージや同じデータベース内の文字列メッセージを、アプリケーションが追跡できます。

時間指定およびスケジューリング

エンキューされたメッセージに対して遅延間隔または期限切れを指定することで、実行枠を設定できます。メッセージをマークして、指定した時間（遅延時間）が経過しないと処理できないようにしたり、指定した期限が切れる前に必ず処理されるようにできます。

ルールベースのサブスクライバ

メッセージは、そのプロパティや内容に基づいて、複数の受信者に配信できます。ユーザーは、所定のキューに対してルールベースのサブスクリプションを定義し、関心があるメッセージの受信希望を指定するメカニズムとして使用します。ルールは、メッセージ・プロパティおよび（オブジェクト型および RAW 型ペイロードの）メッセージ・データに基づいて指定できます。サブスクライバ・ルールを使用して、メッセージ配信の受信者を評価します。

この機能を BooksOnLine のシナリオに適用した場合については、8-82 ページの「ルールベースのサブスクリプション」を参照してください。

非同期通知

非同期通知機能によって、クライアントは、関心があるメッセージがあるときに通知を受け取ることができます。このクライアントは、この機能を使用して複数のサブスクリプションを監視できます。クライアントは、自分自身のサブスクリプションに関する通知を受け取るためにデータベースと接続されている必要はありません。

クライアントは、OCI 関数 `LNOCISubcriptionRegister` または PL/SQL プロシージャ `DBMS_AQ.REGISTER` を使用して、キュー内のどのようなメッセージに関心があるかを登録します。詳細は、第 11 章「操作インタフェース : 基本操作」の「通知の登録」を参照してください。

この機能を BooksOnLine のシナリオに適用した場合については、8-93 ページの「非同期通知」を参照してください。

デキュー機能

次の機能は、メッセージのデキューに適用されます。

受信者

1 つのメッセージを、複数の受信者が取り出すことができます。同一メッセージを複数コピーする必要はありません。この機能を BooksOnLine のシナリオに適用した場合については、8-60 ページの「複数の受信者」を参照してください。

受信者をローカルまたはリモートのサイトに設定できます。この機能を BooksOnLine のシナリオに適用した場合については、8-61 ページの「ローカルおよびリモートの受信者」を参照してください。

デキューにおけるメッセージのナビゲーション

キューからメッセージを選択するにはいくつかの方法があります。最初のメッセージを選択することも、一度メッセージを選択して位置を設定してから次のメッセージを選択することもできます。選択は順序付けに影響されたり、相関識別子の指定で限定されることがあります。また、メッセージ識別子を使用して特定のメッセージを取り出すこともできます。

この機能を BooksOnLine のシナリオに適用した場合については、8-63 ページの「デキューにおけるメッセージ・ナビゲーション」を参照してください。

デキューのモード

DEQUEUE リクエストによって、メッセージの参照または取消しができます。メッセージは、ブラウズ後でも処理可能です。メッセージが取り消されると、そのメッセージに対する DEQUEUE リクエストは無効になります。キューのプロパティによっては、取り消されたメッセージがキュー表に保持されることもあります。

この機能を BooksOnLine のシナリオに適用した場合については、8-67 ページの「デキューのモード」を参照してください。

メッセージ到着待機の最適化

空のキューに対して DEQUEUE を発行できます。新しいメッセージの到着を確認するポーリングを回避するために、リクエストがメッセージ到着を待機できるようにするかどうか、およびその待機時間をユーザーが指定できます。

この機能を BooksOnLine のシナリオに適用した場合については、8-72 ページの「メッセージ到着待機の最適化」を参照してください。

遅延を伴う再試行

メッセージは、厳密には 1 回しか使用できません。メッセージをデキューしようとして失敗し、トランザクションがロールバックされた場合、ユーザーが指定した遅延が終了した後で、そのメッセージの再処理が可能になります。再処理は、ユーザーが指定した回数まで試行されます。

この機能を BooksOnLine のシナリオに適用した場合については、8-74 ページの「遅延間隔をおいた後の再試行」を参照してください。

トランザクション保護のオプション

ENQUEUE リクエストおよび DEQUEUE リクエストは、通常は複数のリクエストを含むトランザクションの一部として、必要なトランザクション動作を実現しています。ただし、特定のリクエストをそれ自身でトランザクションに指定して、そのリクエストの結果をすぐに他のトランザクションから参照できるようにできます。つまり、ENQUEUE 文または DEQUEUE 文が発行されるとすぐに、またはそのトランザクションがコミットされた直後に、メッセージを外部から参照できるようにできます。

例外処理

メッセージは、たとえば実行環境の状態や再試行回数の制限など、所定の制約内では処理できない可能性があります。このような状況が発生すると、メッセージはユーザーが指定した例外キューに移されます。

この機能を BooksOnLine のシナリオに適用した場合については、8-77 ページの「例外処理」を参照してください。

リスニング機能（複数のキューでの待機）

リスニング・コールは、複数キュー上でのメッセージの受取りを待機することができるブロック・コールです。このコールは、ゲートウェイ・アプリケーションによって一連のキューを監視するために使用できます。アプリケーションは、サブスクリプション・リスト上のメッセージを待機する目的でもリスニングを使用できます。リスニングが成功した場合は、デキューによってメッセージを取り出す必要があります。

この機能を BooksOnLine のシナリオに適用した場合については、8-86 ページの「Listen 機能」を参照してください。

ペイロードを伴わないメッセージ・ヘッダーのデキュー

デキュー・モード REMOVE_NODATA は、ペイロードを取り出さずにキューからメッセージを取り出すときに使用できます。このモードは、ペイロードが大きく、内容が無関係なメッセージを削除するときに使用します。

伝播機能

次の機能は、メッセージの伝播に適用されます。インターネット経由での伝播については、1-11 ページの「インターネット統合および Internet Data Access Presentation」を参照してください。

エンキューおよびデキューの自動調整

受信者はローカルでもリモートでもかまいません。Oracle では、分散されたオブジェクト型がサポートされていないため、標準データベース・リンクを使用したリモートのエンキューまたはデキューは正常に処理されません。ただし、AQ のメッセージ伝播を使用すると、リモート・キューにエンキューできます。たとえば、データベース X に接続して、そこにある DROPBOX キューにメッセージをエンキューできます。また、AQ を構成することで、DROPBOX キューにエンキューされたすべてのメッセージを、（ローカルまたはリモートにかかわらず）データベース Y の別のキューに自動的に伝播できます。AQ によって、データベース Y のリモート・キューのタイプが、データベース X のローカル・キューのタイプと構造的に同一であるかが自動的に確認され、メッセージが伝播されます。

伝播されたメッセージの受信者は、アプリケーションまたはキューです。受信者がキューの場合、実際の受信者は、受信者キューのサブスクリプション・リストによって決まります。キューがリモートの場合、メッセージは指定したデータベース・リンクを使用して伝播されます。サポートされるのは、AQ から AQ へのメッセージ伝播のみです。

LOB を伴うメッセージの伝播

伝播は LOB 属性を持つペイロードを処理します。この機能を BooksOnLine のシナリオに適用した場合については、8-106 ページの「LOB 属性を伴うメッセージの伝播」を参照してください。

伝播スケジュール

メッセージは、キューからローカルまたはリモートの受信者に伝播するようにスケジューリングできます。管理者は、開始時刻、伝播枠および次の伝播枠を決定する関数（定期的なスケジュールの場合）を指定できます。

伝播スケジュール機能の拡張

伝播に関する詳細なランタイム情報が収集され、伝播スケジュールごとに DBA_QUEUE_SCHEDULES ビューに格納されます。この情報はキューのデザイナおよび管理者によって、問題の解決やパフォーマンス・チューニングのために使用できます。たとえば、伝播されたメッセージ数またはバイト数の合計および平均の詳細情報が、スケジュール調整に使用できます。同様に、ビューによって報告されたエラーは、問題の診断および解決に使用できます。ビューには、伝播処理をしたセッションの ID、ジョブ・キュー・プロセスの名前などの追加情報も示されます。

この機能を BooksOnLine のシナリオに適用した場合については、8-109 ページの「拡張伝播スケジュール機能」を参照してください。

サード・パーティ・サポート

AQ では、エンキューされたキューから、サード・パーティ製のプロパゲータによって異なるメッセージ・システムにメッセージが伝播されることが可能です。受信者のプロトコル番号が 128 ～ 255 の範囲にある場合、AQ は受信者のアドレスを無視するため、メッセージが AQ システムによって伝播されることはありません。かわりに、サード・パーティ製のプロパゲータが、コンシューマ名として予約済の名前をデキュー操作に指定して、メッセージをデキューできます。予約済のコンシューマ名は、`AQ$_P#`（# は 128 ～ 255 のプロトコル番号）という形式です。たとえば、`AQ$_P128` というコンシューマ名は、プロトコル番号 128 の受信者に対してメッセージをデキューするために使用されます。特定のプロトコル番号を伴うメッセージに対する受信者リストは、デキュー時に `recipient_list` メッセージ・プロパティに戻されます。

アドバンスト・キューイングでサード・パーティのメッセージ・システムと双方向にメッセージを伝播する別の方法は、アドバンスト・キューイングの Enterprise Edition の機能であるメッセージ・ゲートウェイを介して行う方法です。メッセージ・ゲートウェイは、メッセージを AQ キューからデキューし、MQSeries などのサード・パーティのメッセージ・システムへの配信を保証します。メッセージ・ゲートウェイは、サード・パーティのメッセージ・システムからメッセージをデキューし、それらのメッセージを AQ キューにエンキューすることもできます。詳細は、第 18 章「メッセージ・ゲートウェイ」を参照してください。

アドバンスト・キューイングの要素

トランザクション処理とキューイング・テクノロジを統合することで、**アドバンスト・キューイング**という形の永続的なメッセージ機能が可能になりました。この項では、様々なアドバンスト・キューイング用語の定義を示します。

メッセージ

メッセージは、キューに挿入され、そこから取り出される情報の最小単位です。メッセージは、次の 2 つの要素で構成されます。

- 制御情報（メタデータ）
- ペイロード（データ）

制御情報は、AQ がメッセージの管理に使用するメッセージ・プロパティを表します。ペイロード・データはキューに格納される情報で、Oracle AQ に対して透過的です。1 つのメッセージは、1 つのキューにのみ常駐できます。メッセージは、エンキュー・コールによって作成され、デキュー・コールによって処理されます。

キュー

キューは、メッセージのリポジトリです。キューには、ユーザー・キュー（標準キュー）および例外キューの 2 種類があります。ユーザー・キューは、通常のメッセージ処理に使用されます。なんらかの理由で取り出して処理できないメッセージは、例外キューに転送されます。キューは、Oracle AQ 管理インタフェースを使用して作成、変更、起動、停止および削除できます。詳細は、第 9 章「管理インタフェース」を参照してください。

ユーザー・キューは、永続キュー（デフォルト）である場合と、非永続キューである場合があります。永続キューは、メッセージをデータベース表に格納します。これらのキューは、データベース表のすべての信頼性および可用性機能を提供します。非永続キューは、メッセージをメモリーに格納します。非永続キューは、通常、現在接続されているすべてのユーザーに通知を送信する非同期メカニズムを提供するために使用されます。

キュー表

キューは、キュー表に格納されます。キュー表はデータベース表で、1 つ以上のキューを含みます。各キュー表には、デフォルト例外キューがあります。7-2 ページの図 7-1「基本キュー」に、メッセージ、キューおよびキュー表間の関連を示します。

エージェント

エージェントは、キューのユーザーです。エンド・ユーザーまたはアプリケーションがエージェントになれます。エージェントには、次の 2 種類があります。

- メッセージをキューに入れる（エンキューする）プロデューサ
- メッセージを取り出す（デキューする）コンシューマ

任意の時点でキューにアクセスできるプロデューサおよびコンシューマの数に、制限はありません。エージェントは、Oracle AQ 操作インタフェースを使用して、キューへのメッセージの挿入およびキューからのメッセージの取出しができます。詳細は、第 11 章「操作インタフェース : 基本操作」を参照してください。

エージェントは、名前、アドレスおよびプロトコルで識別されます。このデータ構造の詳細は、2-3 ページの「エージェント型（aq$_agent）」を参照してください。

- エージェントの名前には、アプリケーションの名前またはアプリケーションによって割り当てられた名前を使用できます。キュー自体がエージェントになって、別のキューにエンキューまたはデキューすることがあります。
- アドレス・フィールドは、最大 1024 バイトの文字フィールドで、プロトコルに関連して解釈されます。たとえば、プロトコルのデフォルト値は 0（ゼロ）で、データベース・リンクのアドレスを示します。この場合、このプロトコルのアドレスは次のようになります。

```
queue_name@dblink
```

ここで queue_name の形式は [schema.]queue で、dblink は完全に修飾されたデータベース・リンク名、またはドメイン名がないデータベース・リンク名のどちらかです。

受信者

メッセージの受信者は名前のみで指定できますが、その場合、受信者はそのメッセージがエンキューされたキューからデキューする必要があります。プロトコルの値が 0（ゼロ）の場合、名前およびアドレスで受信者を指定できます。アドレスは、同じデータベースまたは（データベース・リンクによって識別された）他の Oracle データベース中の別のキューの名前です。この場合、メッセージは指定されたキューに伝播され、指定された名前のコンシューマによってデキューできます。受信者の名前が NULL の場合、メッセージはアドレスに指定されたキューに伝播され、アドレスに指定されたキューのサブスクライバによってデキューされます。プロトコル・フィールドの値が 0（ゼロ）でない場合、名前およびアドレスのフィールドは無視され、メッセージは特定のコンシューマによってデキューできます（1-20 ページの「サード・パーティ・サポート」を参照）。

受信者およびサブスクリプション・リスト

複数のコンシューマが単一のメッセージを使用できます。

- エンキュー元は、メッセージの受信者としてメッセージを取り出すことができるコンシューマを明示的に指定できます。受信者はエージェントで、名前、アドレスおよびプロトコルによって識別されます。
- キュー管理者は、キューからメッセージを取り出すことができる受信者のデフォルト・リストを指定できます。デフォルト・リストで指定される受信者を、サブスクライバといいます。受信者の指定がないメッセージがエンキューされると、そのメッセージはすべてのサブスクライバに送信されます。

異なるキューには異なるサブスクライバを指定でき、同じ受信者が複数のキューのサブスクライバになることも可能です。さらに、キュー内の特定のメッセージに特定の受信者（そのキューのサブスクライバであってもなくても可）を指定することで、サブスクライバ・リストをオーバーライドできます。

ルール

ルールは、サブスクライバがメッセージをサブスクライブする際に、持っている 1 つ以上の関心事項を定義するために使用します。このルール基準に適合したメッセージが、関心を持っているサブスクライバに配信されます。ルールは、SQL 問合せの WHERE 句に似た構文を使用するブール式（TRUE または FALSE という値を持つ）で定義されます。このブール式には、次のような条件を組み込むことができます。

- メッセージ・プロパティ（現行では、優先順位および相関識別子）
- ユーザー・データ・プロパティ（オブジェクト・ペイロードのみ）

- ファンクション（SQL 問合せの WHERE 句で指定）

ルールベースのサブスクライバ

ルールベースのサブスクライバとは、デフォルトの受信者リストでルールが対応付けられているサブスクライバです。メッセージの受信者が指定されていない場合でも、対応付けられたルールの評価結果が TRUE のときは、そのメッセージはルールベースのサブスクライバに送信されます。

変換

変換は、1 つの Oracle データ型から別のデータ型へのマッピングを定義します。変換は、ソース・データ型を入力として取得し、ターゲット・データ型のオブジェクトを戻す SQL ファンクションによって表現されます。エンキュー時に変換を指定して、メッセージがキューに挿入される前に適切な型に変換されるようにできます。デキュー時に変換を指定すると、メッセージが任意のフォーマットで受信されます。リモート・サブスクライバで変換が指定された場合、メッセージは宛先キューに伝播される前に変換されます。

キュー・モニター

キュー・モニター（QMNn）は、キュー内のメッセージを監視するバックグラウンド・プロセスです。キュー・モニターでは、メッセージの遅延処理、期限切れおよび再試行遅延のメカニズムを提供します。キュー・モニターは、キュー表およびその索引と索引構成表（IOT）のガベージ・コレクションも行います。たとえば、キュー・モニターは、マルチ・コンシューマ・キューのすべてのサブスクライバがメッセージを受信した時期を判断し、その後、キュー表およびその索引と IOT からメッセージを削除します。

同時に開始できるキュー・モニターは、最大で 10 個です。動的 init.ora パラメータ aq_tm_processes を設定して、キュー・モニターを開始します。キュー・モニターは、毎分、またはメッセージに期限切れや処理準備完了というマークを付加するという作業があるときに動作します。

JMS 用語

Java パッケージ oracle.jms を使用する場合は、次のことを理解しておいてください。

- JMS 用語では、エンキューを**送信**といいます。
- メッセージの宛先は、単なる**キュー**です。
- メッセージのコンテナを**トピック**といい、各アプリケーションが特定のトピックについて**パブリッシュ**、またはそのトピックを**サブスクライブ**できるようになっています。
- JMS の**トピック**は、他の AQ インタフェースの**マルチ・コンシューマ・キュー**にマップできます。

- Java パッケージ oracle.jms には、公開 JMS 標準に Oracle の拡張を実装するためのクラスおよびインタフェースがあります。

デモ

次のデモが、$ORACLE_HOME/rdbms/demo ディレクトリにあります。詳細は、このデモ・ディレクトリの aqxmlreadme.txt および aqjmsreadme.txt を参照してください。

表 1-1 デモ

デモ	トピック
aqjmsdemo01.java	テキスト・メッセージのエンキューおよびメッセージ・プロパティに基づくデキュー
aqjmsdemo02.java	メッセージ・リスナーのデモ
aqjmsdemo03.java	メッセージ・リスナーのデモ
aqjmsdemo04.java	Oracle 型ペイロード（ペイロードの内容に対するデキュー）
aqjmsdemo05.java	キュー・ブラウザの例
aqjmsdemo06.java	データベース内のキュー間でのスケジュール伝播
aqjmsdmo.sql	AQ JMS デモの設定
aqjmsREADME.txt	AQ Java API デモおよび AQ JMS デモの説明
aqorademo01.java	RAW メッセージのエンキューおよびデキュー
aqorademo02.java	Custom Datum インタフェースを使用したオブジェクト型メッセージのエンキューおよびデキュー
aqoradmo.sql	AQ Java API デモ用のファイルの設定
aqxml01.xml	AQXmlSend - 伝達（ピギーバック）コミットによるユーザー定義型シングル・コンシューマ・キューへのエンキュー
aqxml02.xml	AQXmlReceive- 伝達（ピギーバック）コミットによるユーザー定義型シングル・コンシューマ・キューからのデキュー
aqxml03.xml	AQXmlPublish -（LOB を含む）ユーザー定義型マルチ・コンシューマ・キューへのエンキュー
aqxml04.xml	AQXmlReceive- ユーザー定義型マルチ・コンシューマ・キューからのデキュー
aqxml05.xml	AQXmlCommit- 前の操作のコミット

表 1-1 デモ（続き）

デモ	トピック
aqxml06.xml	AQXmlSend- 伝達（ピギーバック）コミットによる JMS テキスト・シングル・コンシューマ・キューへのエンキュー
aqxml07.xml	AQXmlReceive- 伝達（ピギーバック）コミットによる JMS テキスト・シングル・コンシューマ・キューからのデキュー
aqxml08.xml	AQXmlPublish- 受信者を指定したマルチ・コンシューマ・キューへの JMS MAP メッセージのエンキュー
aqxml09.xml	AQXmlReceive- マルチ・コンシューマ・キューからの JMS MAP メッセージのデキュー
aqxml10.xml	AQXmlRollback- 前の操作のロールバック
aqxmlhtp.sql	HTTP 伝播
AQDemoServlet.java	AQ XML ファイルを転送するサーブレット（JVM 用）
AQPropServlet.java	AQ HTTP 伝播を行うサーブレット
newaqdemo00.sql	ユーザー、メッセージ型、表などの作成
newaqdemo01.sql	キュー表、キュー、サブスクライバおよびセットアップの設定
newaqdemo02.sql	メッセージのエンキュー
newaqdemo03.sql	デキュー・プロシージャのインストール
newaqdemo04.sql	ブロッキング・デキューの実行
newaqdemo05.sql	複数エージェントに対するリスニングの実行
newaqdemo06.sql	ユーザー、キュー表、キュー、サブスクライバのクリーンアップ（cleanup スクリプト）
ociaqdemo00.c	メッセージのエンキュー
ociaqdemo01.c	ブロッキング・デキューの実行
ociaqdemo02.c	複数エージェントに対するリスニングの実行

2

基本的なコンポーネント

この章では、次の基本的なコンポーネントについて説明します。

- データ構造
- 管理インタフェースの列挙定数
- 操作インタフェースの列挙定数
- INIT.ORA パラメータ・ファイルの考慮点

データ構造

次の章では、データ構造が使用されるアドバンスト・キューイングの管理インタフェースおよび操作インタフェースについて説明します。

- 第 9 章「管理インタフェース」
- 第 11 章「操作インタフェース : 基本操作」

オブジェクト名（object_name）

用途

データベース・オブジェクトに名前を付けます。このネーミング規則は、キュー、キュー表およびオブジェクト型に適用されます。

構文

```
object_name := VARCHAR2
object_name := [<schema_name>.]<name>
```

使用方法

オブジェクト名は、オプションのスキーマ名および名前で指定します。スキーマ名を指定しない場合は、現行のスキーマが想定されます。名前は、『Oracle9*i* SQL リファレンス』の予約語に関する説明のガイドラインに従う必要があります。スキーマ名、エージェント名およびオブジェクト型名は、それぞれ 30 バイト以下で指定可能です。ただし、キュー名およびキュー表名は 24 バイト以下で指定可能です。

型名（type_name）

用途

キュー・タイプを定義します。

構文

```
type_name := VARCHAR2
type_name := <object_type> | "RAW"
```

使用方法

表 2-1 に、type_name の使用方法を示します。

表 2-1 型名（type_name）

パラメータ	説明
<object_types>	オブジェクト型作成の詳細は、『Oracle9*i* データベース概要』を参照してください。オブジェクト型の属性数は最大で 900 です。
"RAW"	RAW 型のペイロードを格納するために、AQ ではペイロード・リポジトリとして LOB 列を持つキュー表を作成します。ペイロードのデータ・サイズは、最大で 32KB です。LOB 列は RAW ペイロードの格納に利用されるため、AQ 管理者は、キュー表の作成時に LOB 表領域を選択し、STORAGE 句パラメータに LOB 記憶域文字列を記述することで、LOB 記憶域を構成できます。

エージェント型（aq$_agent）

用途

メッセージのプロデューサまたはコンシューマを識別します。

構文

```
TYPE aq$_agent IS OBJECT (
   name            VARCHAR2(30),
   address         VARCHAR2(1024),
   protocol        NUMBER)
```

使用方法

マルチ・コンシューマ・キューにサブスクライバとして追加されたすべてのコンシューマは、AQ$_AGENT パラメータに一意の値を持つ必要があります。多くのサブスクライバを追加するには、DBMS_AQADM.ADD_SUBSCRIBER プロシージャを繰り返し使用することで、1 つのマルチ・コンシューマ・キュー当たり最大 1024 サブスクライバを追加できます。2 つのサブスクライバが AQ$_AGENT 型の NAME 属性、ADDRESS 属性および PROTOCOL 属性に同じ値を持つことはできません。この 3 つの属性のうち少なくとも 1 つは、2 つのサブスクライバに対して異なる値にする必要があります。

表 2-2 に、aq$_agent の使用方法を示します。

表 2-2 エージェント型（aq$_agent）

パラメータ	説明
name (VARCHAR2(30))	プロデューサまたはコンシューマの名前。名前は、『Oracle9*i* SQL リファレンス』の予約語に関する説明のガイドラインに従う必要があります。
address (VARCHAR2(1024))	受信者のプロトコル固有のアドレス。プロトコルが 0（デフォルト）の場合、アドレスの形式は [schema.]queue[@dblink] です。
protocol (NUMBER)	アドレスを解析してメッセージを伝播するプロトコル。デフォルト値は 0（ゼロ）です。

AQ 受信者リスト型（aq$_recipient_list_t）

用途

メッセージを受信するエージェントのリストを識別します。

構文

```
TYPE aq$_recipient_list_t IS TABLE OF aq$_agent
          INDEX BY BINARY_INTEGER;
```

AQ エージェント・リスト型（aq$_recipient_list_t）

用途

DBMS_AQ.LISTEN によってリスニングするエージェントのリストを識別します。

構文

```
TYPE aq$_agent_list_t IS TABLE OF aq$_agent
          INDEX BY BINARY_INTEGER;
```

AQ サブスクライバ・リスト型（aq$_subscriber_list_t）

用途

このキューにサブスクライブするサブスクライバのリストを識別します。

構文

```
TYPE aq$_subscriber_list_t IS TABLE OF aq$_agent
          INDEX BY BINARY INTEGER;
```

AQ 登録情報リスト型（aq$_reg_info_list）

用途

キューに対する登録リストを識別します。

構文

```
TYPE aq$_reg_info_list AS VARRAY(1024) OF sys.aq$_reg_info
```

AQ 転送情報リスト型（aq$_post_info_list）

用途

メッセージが転送される匿名サブスクリプションのリストを識別します。

構文

```
TYPE aq$_post_info_list AS VARRAY(1024) OF sys.aq$_post_info
```

AQ 登録情報型（aq$_reg_info）

用途

メッセージのプロデューサまたはコンシューマを識別します。

構文

```
TYPE sys.aq$_reg_info IS OBJECT (
   name      VARCHAR2(128),
   namespace NUMBER,
   callback  VARCHAR2(4000),
   context   RAW(2000));
```

属性

表 2-3 AQ 登録情報型の属性

属性	説明
name	サブスクリプション名を指定します。 サブスクリプション名は、シングル・コンシューマ・キューに対する登録の場合は <schema>.<queue>、マルチ・コンシューマ・キューに対する登録の場合は <schema>.<queue>:<consumer_name> という形式になります。
namespace	サブスクリプションの名前空間を指定します。 AQ キューから通知を受信するには、名前空間が DBMS_AQ.NAMESPACE_AQ である必要があります。 DBMS_AQ.POST または OCISubscriptionPost() を使用して他のアプリケーションから通知を受信するには、名前空間が DBMS_AQ.NAMESPACE_ANONYMOUS である必要があります。
callback	メッセージ通知で実行されるアクションを指定します。 電子メール通知の場合は、mailto://xyz@company.com という形式です。 AQ PL/SQL コールバックの場合は、次の形式を使用します。 RAW 型メッセージ・ペイロードの場合: plsql://<schema>.<procedure>?PR=0 XML に変換されたユーザー定義型メッセージ・ペイロードの場合: plsql://<schema>.<procedure>?PR=1
context	コールバック関数に渡すコンテキストを指定します。デフォルトは NULL です。

表 2-4 に、非永続キューに対して異なる通知のメカニズムまたは表現が指定された場合に実行されるアクションを示します。

表 2-4 非永続キュー

キュー・ペイロード型	指定の表現					
	RAW			XML		
	通知メカニズム			通知メカニズム		
	LNOCI	電子メール	PL/SQL コールバック	LNOCI	電子メール	PL/SQL コールバック
RAW 型	コールバックは、ペイロードの RAW データを受信します。	未サポート	PL/SQL コールバックは、ペイロードの RAW データを受信します。	コールバックは、ペイロードの XML データを受信します。	XML データが SOAP メッセージとしてフォーマットされ、登録されている電子メール・アドレスに送信されます。	PL/SQL コールバックは、ペイロードの XML データを受信します。
ユーザー定義型	未サポート	未サポート	未サポート	コールバックは、ペイロードの XML データを受信します。	XML データが SOAP メッセージとしてフォーマットされ、登録されている電子メール・アドレスに送信されます。	PL/SQL コールバックは、ペイロードの XML データを受信します。

AQ 通知記述子型（aq$_descriptor）

用途

通知時に AQ PL/SQL コールバックを使用して、AQ 記述子を指定します。

構文

```
TYPE sys.aq$_descriptor IS OBJECT (
  queue_name     VARCHAR2(30),
  consumer_name VARCHAR2(30),
  msg_id         RAW(16),
  msg_prop       msg_prop_t);
```

属性

表 2-5 AQ 通知記述子型

属性	説明
queue_name	メッセージがエンキューされ、通知されたキューの名前
consumer_name	マルチ・コンシューマ・キューに対するコンシューマの名前
msg_id	メッセージ ID
msg_prop	メッセージ・プロパティ

AQ 転送情報型（aq$_post_info）

用途

メッセージが転送される匿名サブスクリプションを指定します。

構文

```
TYPE sys.aq$_post_info IS OBJECT (
  name       VARCHAR2(128),
  namespace  NUMBER,
  payload    RAW(2000));
```

属性

表 2-6 AQ 転送情報型の属性

属性	説明
name	転送する匿名サブスクリプションの名前。
namespace	DBMS_AQ.POST または OCISubscriptionPost() を介して、他のアプリケーションから通知を受信するには、名前空間が DBMS_AQ.NAMESPACE_ANONYMOUS である必要があります。
payload	匿名サブスクリプションに転送されるペイロード。デフォルトは NULL です。

管理インタフェースの列挙定数

INFINITE、TRANSACTIONAL、NORMAL_QUEUE などの列挙定数を値として選択する場合、定数はそれを定義するパッケージのスコープとともに指定する必要があります。管理インタフェースに関連付けられるすべての型には、前に DBMS_AQADM を付ける必要があります。次に例を示します。

```
DBMS_AQADM.NORMAL_QUEUE
```

表 2-7 に列挙定数を示します。

表 2-7 管理インタフェースの列挙定数

パラメータ	オプション
retention	0、1、2、...INFINITE
message_grouping	TRANSACTIONAL、NONE
queue_type	NORMAL_QUEUE、EXCEPTION_QUEUE、NON_PERSISTENT_QUEUE

操作インタフェースの列挙定数

BROWSE、LOCKED、REMOVE などの列挙定数を使用する場合、PL/SQL 定数はそれを定義するパッケージのスコープとともに指定する必要があります。操作インタフェースに関連付けられるすべての型には、前に DBMS_AQ を付ける必要があります。次に例を示します。

```
DBMS_AQ.BROWSE
```

表 2-8 に列挙定数を示します。

表 2-8 操作インタフェースの列挙定数

パラメータ	オプション
visibility	IMMEDIATE、ON_COMMIT
dequeue mode	BROWSE、LOCKED、REMOVE、REMOVE_NODATA
navigation	FIRST_MESSAGE、NEXT_MESSAGE、NEXT_TRANSACTION
state	WAITING、READY、PROCESSED、EXPIRED
sequence_deviation	BEFORE、TOP
wait	FOREVER、NO_WAIT
delay	NO_DELAY
expiration	NEVER
namespace	NAMESPACE_AQ、NAMESPACE_ANONYMOUS

INIT.ORA パラメータ・ファイルの考慮点

AQ_TM_PROCESSES および JOB_QUEUE_PROCESSES パラメータを init.ora パラメータ・ファイルに指定できます。

AQ_TM_PROCESSES パラメータ

キュー・メッセージに対して時間監視を行う場合は、init.ora パラメータ・ファイルに AQ_TM_PROCESSES パラメータを指定する必要があります。これは、遅延および期限切れのプロパティが指定されたメッセージに対して使用します。このパラメータは 1 以上に設定する必要があります。このパラメータは 0（ゼロ）～ 10 の範囲内で設定できます。それ以外の数値を設定すると、エラーになります。このパラメータを 1 にすると、バックグラウンド・プロセスとして、キュー・モニター・プロセス（QMN）が 1 つ作成されます。このパラメータが未指定の場合、または 0（ゼロ）に設定した場合、キュー・モニター・プロセスは作成されません。

表 2-9 に、パラメータの情報を示します。

表 2-9 AQ_TM_PROCESSES パラメータ

パラメータ	オプション
パラメータ名	aq_tm_processes
パラメータ・タイプ	整数
パラメータ・クラス	動的
設定可能値	0 ～ 10
構文	aq_tm_processes = <0 to 10>
プロセス名	ora_qmn<n>_<oracle sid>
例	aq_tm_processes = 1

JOB_QUEUE_PROCESSES パラメータ

伝播は、ジョブ・キュー（SNP）・プロセスによって処理されます。あるインスタンスで開始されるジョブ・キュー・プロセスの数は、init.ora ファイルの JOB_QUEUE_PROCESSES パラメータで制御されます。このパラメータのデフォルト値は 0（ゼロ）です。メッセージを伝播するには、このパラメータを 2 以上にする必要があります。伝播が必要なメッセージを持っているキューが多数ある場合、伝播するメッセージの宛先が多数ある場合、またはジョブ・キューに他のジョブがある場合は、データベース管理者はこのパラメータをより高い値に設定できます。

> **参照：** JOB_QUEUE_PROCESSES の詳細は、『Oracle9*i* SQL リファレンス』を参照してください。

Java アドバンスト・キューイング API は、アドバンスト・キューイングの管理機能と操作機能の両方をサポートします。メッセージ機能アプリケーション用の Java プログラムを開発するときは、JDBC を使用してデータベースへのコネクションをオープンしてから、Java AQ API である `oracle.AQ` パッケージでメッセージ・キューイングを行います。これは、PL/SQL インタフェースを使用する必要がないことを意味します。

3

AQ プログラム環境

この章では、操作が必要な要素、および AQ アプリケーション環境の準備時に考慮する必要がある問題点を説明します。内容は次のとおりです。

- AQ にアクセスするためのプログラム環境
- PL/SQL を使用した AQ へのアクセス
- OCI を使用した AQ へのアクセス
- Visual Basic（OO4O）を使用した AQ へのアクセス
- AQ Java（oracle.AQ）クラスを使用した AQ へのアクセス
- Oracle JMS を使用した AQ へのアクセス
- AQ XML サーブレットを使用した AQ へのアクセス
- AQ プログラム環境の比較

AQ にアクセスするためのプログラム環境

Oracle のアドバンスト・キューイング機能へのアクセスには、次のプログラム環境が使用されます。

- ネイティブ AQ インタフェース
  - PL/SQL（DBMS_AQADM および DBMS_AQ）: 管理機能および操作機能をサポートします。
  - C（OCI）: 操作機能をサポートします。
  - Visual Basic（OO4O）: 操作機能をサポートします。
  - Java（JDBC を使用する oracle.AQ パッケージ）: 管理機能および操作機能をサポートします。
- AQ への JMS インタフェース
  - Java（JDBC を使用する javax.jms および oracle.jms パッケージ）: 標準 JMS 管理機能と操作機能、および Oracle の JMS 拡張機能をサポートします。
- AQ への XML インタフェース
  - AQ XML サーブレットは、XML メッセージ・フォーマットを使用して操作機能をサポートします。

表 3-1「AQ プログラム環境」に、AQ プログラム環境およびその構文の参照先を示します。

表 3-1 AQ プログラム環境

言語	プリコンパイラまたはインタフェース・プログラム	構文の参照先	この章にある参照先
PL/SQL	DBMS_AQADM および DBMS_AQ パッケージ	『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』	3-3 ページの「PL/SQL を使用した AQ へのアクセス」
C	Oracle Call Interface（OCI）	『Oracle Call Interface プログラマーズ・ガイド』	3-4 ページの「OCI を使用した AQ へのアクセス」
Visual Basic	Oracle Objects For OLE（OO4O）	Oracle Objects For OLE（OO4O）は、Oracle Client for Windows NT に含まれている Windows ベースの製品です。 この製品については、オンライン・ヘルプのみが提供されています。オンライン・ヘルプは、Oracle インストール時の「Application Development」サブメニューから使用可能です。	3-6 ページの「AQ Java（oracle.AQ）クラスを使用した AQ へのアクセス」

表 3-1 AQ プログラム環境（続き）

言語	プリコンパイラまたはインタフェース・プログラム	構文の参照先	この章にある参照先
Java（AQ）	JDBC Application Program Interface（API）を介しての oracle.AQ パッケージ	『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』	3-6 ページの「AQ Java（oracle.AQ）クラスを使用した AQ へのアクセス」
Java（JMS）	JDBC Application Program Interface（API）を介しての oracle.JMS パッケージ	『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』	3-6 ページの「AQ Java（oracle.AQ）クラスを使用した AQ へのアクセス」および 3-8 ページの「Oracle JMS を使用した AQ へのアクセス」
AQ XML サーブレット	HTTP を介しての oracle.AQ.xml.AQxmlServlet サーブレット	『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』	3-10 ページの「AQ XML サーブレットを使用した AQ へのアクセス」

PL/SQL を使用した AQ へのアクセス

PL/SQL パッケージ（DBMS_AQADM および DBMS_AQ）は、ネイティブ AQ インタフェースを使用した Oracle Advanced Queuing の管理機能および操作機能へのアクセスをサポートします。次の機能が含まれます。

- キュー、キュー表、非永続キュー、マルチ・コンシューマ・キュー / トピック、RAW メッセージ、構造化データを持つメッセージの作成
- キュー表、キュー、マルチ・コンシューマ・キュー / トピックの取得
- キュー表、キュー / トピックの変更
- キュー / トピックの削除
- キュー / トピックの開始または停止
- 権限の付与および取消し
- サブスクライバの追加、削除および変更
- AQ インターネット・エージェントの追加、削除および変更
- AQ インターネット・エージェントに対するデータベース・ユーザー権限の付与または取消し
- 伝播スケジュールの使用可能化、使用不可能化および変更
- シングル・コンシューマ・キュー（Point-to-Point モデル）へのメッセージのエンキュー
- マルチ・コンシューマ・キュー / トピック（パブリッシュ・サブスクライブ・モデル）へのメッセージのパブリッシュ

- マルチ・コンシューマ・キューのメッセージに対するサブスクライブ
- キューのメッセージのブラウズ
- キュー / トピックからのメッセージの受信
- メッセージの非同期受信登録
- 複数キュー / トピック上のメッセージのリスニング
- 匿名サブスクリプションへのメッセージの転送
- LDAP サーバーのエージェントのバインドまたはアンバインド
- LDAP サーバーの AQ オブジェクトに対する別名の追加または削除

> **参照：** パラメータ、パラメータ・タイプ、戻り値、例、DBMS_AQADM、DBMS_AQ 構文などの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

PL/SQL の DBMS_AQADM および DBMS_AQ で使用可能な機能の詳細は、表 3-2 ～表 3-9 を参照してください。

OCI を使用した AQ へのアクセス

Oracle Call Interface（OCI）は、ネイティブ AQ インタフェースを介して、Oracle Advanced Queuing の機能へのインタフェースを提供します。

OCI クライアントは、次のアクションを実行できます。

- メッセージのエンキュー
- メッセージのデキュー
- 複数キュー上のメッセージのリスニング
- メッセージ通知の受信登録

さらに、OCI クライアントは、OCISubscriptionRegister を使用して、キューの新しいメッセージの非同期通知を受信できます。

> **参照：** 構文の詳細は、『Oracle Call Interface プログラマーズ・ガイド』の「OCI およびアドバンスト・キューイング」および「パブリッシュ・サブスクライブの通知」を参照してください。

ユーザー定義のペイロード型を持つキューでは、Oracle 型の OCI マッピングを生成するために、OTT を使用する必要があります。OCI クライアントは、AQ 記述子のメモリーおよびメッセージ・ペイロードを解放する必要があります。

例

LNOCI インタフェース

OCI AQ インタフェースの例は、A-11 ページの「メッセージのエンキューおよびデキュー」を参照してください。

OCI 記述子メモリーの管理

OCI 記述子のメモリー管理の例は、A-70 ページの「AQ およびメモリーの使用」を参照してください。

Visual Basic（OO4O）を使用した AQ へのアクセス

Visual Basic（OO4O）は、ネイティブ AQ インタフェースを介して、Oracle Advaneced Queuing の操作機能へのアクセスをサポートします。

次の機能が含まれます。

- コネクション、RAW メッセージ、構造化データを持つメッセージの作成
- シングル・コンシューマ・キュー（Point-to-Point モデル）へのメッセージのエンキュー
- マルチ・コンシューマ・キュー / トピック（パブリッシュ・サブスクライブ・モデル）へのメッセージのパブリッシュ
- キューのメッセージのブラウズ
- キュー / トピックからのメッセージの受信
- メッセージの非同期受信登録

詳細情報

OO4O の詳細は、次の Web サイトを参照してください。

- http://technet.oracle.com

  「Products」>「Internet Tools」>「Programmer」を順に選択し、Oracle Objects for OLE にスクロールします。ページ最下部には、インタフェースを使用する場合に有効な記事のリストがあります。

- http://www.oracle.com/products

  OO4O または Oracle Objects for OLE に関する記事を検索します。

AQ Java（oracle.AQ）クラスを使用した AQ へのアクセス

Java AQ API は、アドバンスト・キューイングの管理機能と操作機能の両方をサポートします。メッセージ機能アプリケーション用の Java プログラムを開発するときは、JDBC を使用してデータベースに対するコネクションをオープンしてから、Java AQ API である oracle.AQ パッケージでメッセージ・キューイングを行います。

現行の PL/SQL インタフェースに基づく共通インタフェースおよびクラスについては、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

- 共通インタフェースには「**AQ**」という接頭辞が付いています。これらのインタフェースの実装は、Oracle9*i* および Oracle Lite では異なります。
- このマニュアルでは、この共通インタフェースおよびこれに対応する Oracle9*i* での実装クラスについて説明しますが、これらのクラスには「**AQOracle**」という接頭辞が付きます。

Java AQ クラスへのアクセス

Java AQ クラスは、$ORACLE_HOME/rdbms/jlib/aqapi*.jar にあります。Oracle9*i* リリース 2（9.2）では、Oracle JMS は Sun 社の JMS 1.0.2b 標準に準拠しています。これらのクラスは、Oracle9*i* 以上のすべての OracleJDBC ドライバで使用できます。

- **OCI または JDBC Thin ドライバの使用**
  - JDK 1.3 の場合は、CLASSPATH に次のパスを含めます。
    - $ORACLE_HOME/jdbc/lib/classes12.zip
    - $ORACLE_HOME/jlib/jndi.jar
    - $ORACLE_HOME/rdbms/jlib/aqapi13.jar
    - $ORACLE_HOME/rdbms/jlib/jmscommon.jar
  - JDK 1.2 の場合は、CLASSPATH に次のパスを含めます。
    - $ORACLE_HOME/jdbc/lib/classes12.zip
    - $ORACLE_HOME/jlib/jndi.jar
    - $ORACLE_HOME/rdbms/jlib/aqapi12.jar
    - $ORACLE_HOME/rdbms/jlib/jmscommon.jar
  - JDK 1.1 の場合は、CLASSPATH に次のパスを含めます。
    - $ORACLE_HOME/jdbc/lib/classes111.zip
    - $ORACLE_HOME/jlib/jndi.jar
    - $ORACLE_HOME/rdbms/jlib/aqapi11.jar

    - `$ORACLE_HOME/rdbms/jlib/jmscommon.jar`

- **Oracle JVM での Oracle サーバー・ドライバの使用 :** アプリケーションが、Oracle サーバー・ドライバを使用して、Java ストアド・プロシージャから Java AQ API にアクセスする場合、AQ 関連の Java ファイルは、通常、Java 対応のデータベースに自動的に事前にロードされています。Java ファイルがロードされていない場合、まず `loadjava` ユーティリティを使用して、`jmscommon.jar` および `aqapi.jar` ファイルをデータベースにロードする必要があります。

アドバンスト・キューイングの使用例

付録 A「Oracle Advanced Queuing の使用例」には、次の例が含まれています。

- Java を使用したオブジェクト型メッセージ（CustomDatum インタフェース）のエンキューおよびデキュー
- Java を使用したオブジェクト型メッセージ（SQLData インタフェース）のエンキューおよびデキュー
- Java を使用したキュー表およびキューの作成
- Java を使用したキュー作成およびエンキュー / デキューの開始
- Java を使用したマルチ・コンシューマ・キューの作成およびサブスクライバの追加
- Java を使用した RAW メッセージのエンキュー
- Java を使用したメッセージのデキュー
- Java を使用したブラウズ・モードでのメッセージのデキュー
- Java を使用した優先順位によるメッセージのエンキュー
- Java を使用した LOB 属性を含むオブジェクト型メッセージのエンキューおよびデキュー

Java AQ API の管理

Java AQ API の様々な実装は、`AQDriverManager` によって管理されます。Oracle Lite と Oracle9*i* の両方には、`AQDriverManager` に登録されている `AQDriver` があります。ドライバ・マネージャを使用すると、メッセージングに関するタスクを実行するために使用される `AQSession` を作成することができます。

Oracle9*i* の AQ ドライバは、`Class.forName（"oracle.AQ.AQOracleDriver"）` コマンドによって登録されます。

`AQDriverManager.createAQSession()` が起動されると、このメソッドは、`createAQSession()` コールに渡されたパラメータに基づいて、（登録済ドライバの中から）該当する `AQDriver` をコールします。

Oracle9*i* の `AQDriver` では、AQSession を作成するパラメータとして、有効な JDBC コネクションが渡されるものと想定します。ユーザーが AQ Java インタフェースを使用するには、

DBMS_AQIN パッケージに対する実行権限が必要です。これらの権限は、AQ_USER_ROLE または AQ_ADMINSTRATOR_ROLE を介しても取得できます。また、ユーザーには、8.1 形式のキュー表に対する適切なシステム権限およびキュー権限も必要です。

Oracle JMS を使用した AQ へのアクセス

JMS: JMS は、Sun 社、オラクル社、IBM 社およびその他のベンダーが定義したメッセージング機能標準です。JMS は、JMS クライアントが企業のメッセージ関連製品の機能にアクセスする方法を定義する、インタフェースおよび対応するセマンティクスの集合です。

Oracle JMS: Oracle JMS は、JMS 標準に基づいて、Oracle AQ 用に Java API を提供します。Oracle JMS は、標準 JMS インタフェースをサポートし、また、AQ 管理操作およびその他の標準ではない AQ 機能をサポートする拡張機能を持ちます。

標準 JMS 機能

標準 JMS 機能には、次のものが含まれます。

- キューを使用した Point-to-Point 通信モデル
- トピックを使用したパブリッシュ・サブスクライブ通信モデル
- ObjectMessage、StreamMessage、TextMessage、BytesMessage、MapMessage という 5 つのメッセージ型
- 同期および非同期のメッセージ配信
- メッセージ・ヘッダー・フィールド / プロパティに基づくメッセージ選択

Oracle JMS 拡張機能

Oracle JMS 拡張機能には、次のものが含まれます。

- キュー表、キューおよびトピック作成用の管理 API
- トピックの受信者リストを使用した Point-to-Multipoint 通信
- 宛先間のメッセージ伝播アプリケーションによるリモート・サブスクライバの定義を許可します。
- 1 回のアトミック・トランザクションでの JMS および SQL 操作の実行を可能にする、トランザクション属性セッションのサポート
- メッセージがデキューされた後のメッセージ保存
- メッセージ遅延
  メッセージは、一定の遅延後に参照可能にできます。
- 例外処理
  メッセージを正常に処理できない場合、メッセージは例外キューに移されます。

- 標準 JMS メッセージ型の他に、Oracle は AdtMessages もサポートします。これは、Oracle オブジェクトとしてデータベースに格納されるため、メッセージのペイロードをエンキュー後に問い合せることができます。サブスクリプションは、メッセージ・プロパティのみでなく、これらのメッセージの内容にも定義できます。
- トピックのブラウズ
  永続サブスクライバは、パブリッシュ・サブスクライブ（トピック）の宛先のメッセージ全体をブラウズしたり、オプションでブラウズしたメッセージを削除することができます（AQ は、サブスクライバに対してそれらのメッセージを保存しません）。

標準 JMS および Oracle JMS へのアクセス

Oracle JMS は JDBC を使用してデータベースに接続するため、アプリケーションは次のとおり実行できます。

- データベース外では、OCI または JDBC Thin ドライバを使用して実行します。
- Oracle JVM 内では、Oracle サーバー・ドライバを使用して実行します。

標準 JMS インタフェースは、javax.jms パッケージにあります。

Oracle JMS インタフェースは、oracle.jms パッケージにあります。

- OCI または JDBC Thin ドライバの使用 : データベース外で実行するクライアントで JMS を使用するには、CLASSPATH に適切な JDBC ドライバ、JNDI の jar ファイルおよび次の AQ の jar ファイルを指定する必要があります。
  - JDK 1.1 の場合

    ```
    $ORACLE_HOME/rdbms/jlib/jmscommon.jar
    $ORACLE_HOME/rdbms/jlib/aqapi11.jar
    $ORACLE_HOME/jlib/jndi.jar
    $ORACLE_HOME/jdbc/lib/classes111.jar
    ```

  - JDK 1.2 の場合

    ```
    $ORACLE_HOME/rdbms/jlib/jmscommon.jar
    $ORACLE_HOME/rdbms/jlib/aqapi.jar
    $ORACLE_HOME/jlib/jndi.jar
    $ORACLE_HOME/jdbc/lib/classes12.jar
    ```

- Oracle JVM での Oracle サーバー・ドライバの使用 : アプリケーションが Oracle JVM 内で実行される場合、Oracle JVM のインストール時に自動的にロードされた Oracle JMS クラスにアクセスできる必要があります。これらのクラスが使用できない場合は、$ORACLE_HOME/rdbms/admin/initjms という SQL スクリプトを使用して、jmscommon.jar の後に aqapi.jar をロードする必要があります。

権限

Oracle JMS インタフェースを使用する場合、ユーザーには、DBMS_AQIN および DBMS_AQJMS パッケージの実行権限が必要です。これらの権限は、AQ_USER_ROLE または AQ_ADMINSTRATOR_ROLE を介しても取得できます。

また、ユーザーがメッセージを送受信するには、適切なシステム、およびキュー権限またはトピック権限が必要です。

詳細情報

Oracle JMS インタフェースの詳細は、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

AQ XML サーブレットを使用した AQ へのアクセス

AQ XML サーブレットを使用すると、Simple Object Access Protocol（SOAP）および Internet Data Access Presentation（iDAP）という XML メッセージ・フォーマットを使用して、HTTP 経由で Oracle9*i* AQ にアクセスできます。

AQ サーブレットを使用すると、クライアントは次のアクションを実行できます。

- シングル・コンシューマ・キューへのメッセージの送信
- マルチ・コンシューマ・キュー / トピックへのメッセージのパブリッシュ
- キューからのメッセージの受信
- メッセージ通知の受信登録

サーブレットは、oracle.AQ.xml.AQxmlServlet または oracle.AQ.xml.AQxmlServlet20 クラスを拡張する Java クラスを定義することによって、作成できます。これらのクラスによって、javax.servlet.http.HttpServlet クラスが拡張されます。

サーブレットは、JavaSoft 社の Servlet2.0 インタフェースまたは Servlet2.2 インタフェースを実装するすべての Web サーバーまたはサーブレット・コンテナ上に配置できます。

- JavaSoft 社の Servlet2.0 インタフェースを実装する Web サーバーに AQ サーブレットを配置するには、oracle.AQ.xml.AQxmlServle20 クラスの拡張クラスを定義する必要があります。
- JavaSoft 社の Servlet2.2 インタフェースを実装する Web サーバーに AQ サーブレットを配置するには、oracle.AQ.xml.AQxmlServlet クラスの拡張クラスを定義する必要があります。

サーブレットは、JDK 1.1.x ライブラリまたは JDK 1.2.x ライブラリを使用してコンパイルできます。

- JDK 1.1.x では、CLASSPATH に次のものを指定する必要があります。

```
$ORACLE_HOME/jdbc/lib/classes111.jar
$ORACLE_HOME/jlib/jta.jar
$ORACLE_HOME/jdbc/lib/nls_charset11.jar
$ORACLE_HOME/jlib/jndi.jar
$ORACLE_HOME/lib/lclasses11.zip
$ORACLE_HOME/lib/xmlparserv2.jar
$ORACLE_HOME/lib/xschema.jar
$ORACLE_HOME/rdbms/jlib/aqapi11.jar
$ORACLE_HOME/rdbms/jlib/jmscommon.jar
$ORACLE_HOME/rdbms/jlib/aqxml.jar
$ORACLE_HOME/rdbms/jlib/xsu111.jar
$ORACLE_HOME/lib/servlet.jar
```

- JDK 1.2.x では、CLASSPATH に次のものを指定する必要があります。

```
$ORACLE_HOME/jdbc/lib/classes12.jar
$ORACLE_HOME/jlib/jta.jar
$ORACLE_HOME/jdbc/lib/nls_charset12.jar
$ORACLE_HOME/jlib/jndi.jar
$ORACLE_HOME/lib/lclasses12.zip
$ORACLE_HOME/lib/xmlparserv2.jar
$ORACLE_HOME/lib/xschema.jar
$ORACLE_HOME/rdbms/jlib/aqapi.jar
$ORACLE_HOME/rdbms/jlib/jmscommon.jar
$ORACLE_HOME/rdbms/jlib/aqxml.jar
$ORACLE_HOME/rdbms/jlib/xsu12.jar
$ORACLE_HOME/lib/servlet.jar
```

サーブレットは、JDBC OCI ドライバを使用して Oracle9*i* データベース・サーバーに接続するため、Oracle9*i* クライアント・ライブラリはサーブレットをホストするマシン上にインストールされている必要があります。LD_LIBRARY_PATH に $ORACLE_HOME/lib を含める必要があります。

アドバンスト・キューイングへのインターネット・アクセスの詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

AQ プログラム環境の比較

表 3-2 ～表 3-9 に、AQ プログラム環境で使用可能な機能を、利用方法別に示します。各利用方法の詳細は、第 9 章～第 11 章および第 13 章～第 16 章を参照してください。利用図については、第 E 章「Unified Modeling Language 図」を参照してください。

AQ 管理インタフェース

表 3-2 に、PL/SQL、Java（ネイティブ AQ）および Java（JMS）のプログラム環境における同等の AQ 管理機能を示します。

表 3-2 AQ プログラム環境の比較 : 管理インタフェース

利用方法	PL/SQL	Java（ネイティブ）	Java（JMS）
コネクション・ファクトリの作成	不可	不可	AQjmsFactory.getQueue ConnectionFactory AQjmsFactory.getTopic ConnectionFactory
LDAP サーバーでのコネクション・ファクトリの登録	不可	不可	AQjmsFactory.register ConnectionFactory
キュー表の作成	DBMS_AQADM.create_queue_table	AQQueueTableProperty の後に AQSession.createQueueTable	AQjmsSession.createQueue Table
キュー表の取得	<schema>.<queue_table_name> を使用する。	AQSession.getQueueTable	AQjmsSession.getQueue Table
キュー表の変更	DBMS_AQADM.alter_queue_table	AQQueueTable.alter	AQQueueTable.alter
キュー表の削除	DBMS_AQADM.drop_queue_table	AQQueueTable.drop	AQQueueTable.drop
キューの作成	DBMS_AQADM.create_queue	AQSession.createQueue	AQjmsSession.createQueue
キューの取得	<schema>.<queue_name> を使用する。	AQSession.getQueue	AQjmsSession.getQueue
非永続キューの作成	DBMS_AQADM.create_np_queue	未サポート	未サポート
マルチ・コンシューマ・キュー / トピックの作成	DBMS_AQADM.create_queue マルチ・コンシューマが使用可能なキュー表で使用する。	AQSession.createQueue マルチ・コンシューマが使用可能なキュー表で使用する。	AQjmsSession.createTopic マルチ・コンシューマが使用可能なキュー表で使用する。
マルチ・コンシューマ・キュー / トピックの取得	<schema>.<queue_name> を使用する。	AQSession.getQueue	AQjmsSession.getTopic
キュー / トピックの変更	DBMS_AQADM.alter_queue	AQQueue.alterQueue	AQjmsDestination.alter

表 3-2 AQ プログラム環境の比較 : 管理インタフェース（続き）

利用方法	PL/SQL	Java（ネイティブ）	Java（JMS）
キュー / トピックの開始	DBMS_AQADM.start_queue	AQQueue.start AQQueue.startEnqueue AQQueue.startDequeue	AQjmsDestination.start
キュー / トピックの停止	DBMS_AQADM.stop_queue	AQQueue.stop AQQueue.stopEnqueue AQQueue.stopDequeue	AQjmsDestination.stop
キュー / トピックの削除	DBMS_AQADM.drop_queue	AQQueue.drop AQQueueTable.dropQueue	AQjmsDestination.drop
システム権限の付与	DBMS_AQADM.grant_system_privilege	未サポート	AQjmsSession.grantSystemPrivilege
システム権限の取消し	DBMS_AQADM.revoke_system_privilege	未サポート	AQjmsSession.revokeSystemPrivilege
キュー / トピック権限の付与	DBMS_AQADM.grant_queue_privilege	AQQueue.grantQueuePrivilege	AQjmsDestination.grantQueuePrivilege AQjmsDestination.grantTopicPrivilege
キュー / トピック権限の取消し	DBMS_AQADM.revoke_queue_privilege	AQQueue.revokeQueuePrivilege	AQjmsDestination.revokeQueuePrivilege AQjmsDestination.revokeTopicPrivilege
キュー・タイプの検証	DBMS_AQADM.verify_queue_types	未サポート	未サポート
サブスクライバの追加	DBMS_AQADM.add_subscriber	AQQueue.addSubscriber	表 3-6「AQ プログラム環境の比較 : 操作インタフェース - マルチ・コンシューマ・キュー / トピックのメッセージに対するサブスクライブ（パブリッシュ・サブスクライブ・モデルでの利用方法）」を参照

表 3-2 AQ プログラム環境の比較 : 管理インタフェース（続き）

利用方法	PL/SQL	Java（ネイティブ）	Java（JMS）
サブスクライバの変更	DBMS_AQADM.alter_subscriber	AQQueue.alterSubscriber	表 3-6「AQ プログラム環境の比較 : 操作インタフェース - マルチ・コンシューマ・キュー / トピックのメッセージに対するサブスクライブ（パブリッシュ・サブスクライブ・モデルでの利用方法）」を参照
サブスクライバの削除	DBMS_AQADM.remove_subscriber	AQQueue.removeSubscriber	表 3-6「AQ プログラム環境の比較 : 操作インタフェース - マルチ・コンシューマ・キュー / トピックのメッセージに対するサブスクライブ（パブリッシュ・サブスクライブ・モデルでの利用方法）」を参照
伝播のスケジューリング	DBMS_AQADM.schedule_propagation	AQQueue.schedulePropagation	AQjmsDestination.schedulePropagation
伝播スケジュールの使用可能化	DBMS_AQADM.enable_propagation_schedule	AQQueue.enablePropagationSchedule	AQjmsDestination.enablePropagationSchedule
伝播スケジュールの変更	DBMS_AQADM.alter_propagation_schedule	AQQueue.alterPropagationSchedule	AQjmsDestination.alterPropagationSchedule
伝播スケジュールの使用不可能化	DBMS_AQADM.disable_propagation_schedule	AQQueue.disablePropagationSchedule	AQjmsDestination.disablePropagationSchedule
伝播スケジュールの解除	DBMS_AQADM.unschedule_propagation	AQQueue.unschedulePropagation	AQjmsDestination.unschedulePropagation
AQ インターネット・エージェントの作成	DBMS_AQADM.create_aq_agent	未サポート	未サポート
AQ インターネット・エージェントの変更	DBMS_AQADM.alter_aq_agent	未サポート	未サポート
AQ インターネット・エージェントの削除	DBMS_AQADM.drop_aq_agent	未サポート	未サポート
AQ インターネット・エージェントへのデータベース・ユーザー権限の付与	DBMS_AQADM.enable_db_agent	未サポート	未サポート

表 3-2 AQ プログラム環境の比較 : 管理インタフェース（続き）

利用方法	PL/SQL	Java（ネイティブ）	Java（JMS）
AQ インターネット・エージェントからのデータベース・ユーザー権限の取消し	DBMS_AQADM.disable_db_agent	未サポート	未サポート
LDAP サーバーにおけるキュー、エージェント、ConnectionFactory の別名の追加	DBMS_AQADM.add_alias_to_ldap	未サポート	未サポート
LDAP サーバーにおけるキュー、エージェント、ConnectionFactory 用の別名の削除	DBMS_AQADM.del_alias_from_ldap	未サポート	未サポート

AQ 操作インタフェース

表 3-3 〜表 3-9 に、様々な利用方法での PL/SQL、Java（ネイティブ AQ）、OCI、AQ XML サーブレットおよび JMS のプログラム環境における同等の AQ 操作機能を示します。

表 3-3 AQ プログラム環境の比較 : 操作インタフェース - コネクション、セッション、メッセージの作成での利用方法

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
コネクションの作成	不可	JDBC コネクションを作成する。	OCIServerAttach	Web サーバーを使用して、認証後に HTTP コネクションをオープンする。	AQjmsQueue ConnectionFactory. createQueue Connection AQjmsTopic ConnectionFactory. createTopic Connecion
セッションの作成	不可	AQDriver Manager.create AQSession	OCISessionBegin	HTTP サーブレットは最初の SOAP リクエストで自動的に開始される。	QueueConnection. createQueue Session TopicConnecion. createTopicSession
RAW メッセージの作成	メッセージに SQL RAW 型を使用する。	AQQueue.create Message メッセージに AQRawPayload を設定する。	メッセージに OCIRaw を使用する。	XML メッセージにメッセージ・ペイロードの 16 進表示を提供する。 例 : `<raw>023f4523 </raw>`	未サポート

表 3-3 AQ プログラム環境の比較 : 操作インタフェース - コネクション、セッション、メッセージの作成での利用方法（続き）

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
構造化データを持つメッセージの作成	メッセージに SQL ユーザー定義型を使用する。	AQQueue.create Message メッセージに AQObject Payload を設定する。	メッセージに SQL ユーザー定義型を使用する。	JMS キューではない（AQ$_JMS_* 型ではない）ユーザー定義型キューでは、<message_payload> に指定されている XML が、キュー表に対するペイロードの SQL 型にマップする必要がある。 JMS キューでは、<message_payload> に指定されている XML が、<jms_text_message>、<jms_map_message>、<jms_bytes_message> または <jms_object_message> のいずれかである必要がある。	Session.createText Message Session.create ObjectMessage Session.createMap Message Session.createBytes Message Session.create StreamMessage AQjmsSession. createAdtMessage
メッセージ・プロデューサの作成	不可	不可	不可	不可	QueueSession. createSender TopicSession.create Publisher

表 3-4 AQ プログラム環境の比較 : 操作インタフェース - シングル・コンシューマ・キューへのメッセージのエンキュー（ Point-to-Point モデルでの利用方法）

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
シングル・コンシューマ・キューへのメッセージのエンキュー	DBMS_AQ.enqueue	AQQueue.enqueue	LNOCIAQEnq	<AQXmlSend>	QueueSender.send
キューへのメッセージのエンキュー - 可視性オプションの指定	DBMS_AQ.enqueue ENQUEUE_OPTIONS の可視性を指定する。	AQQueue.enqueue AQEnqueueOption の可視性を指定する。	LNOCIAQEnq LNOCIAQEnqOptions の OCI_ATTR_VISIBILITY を指定する。	<AQXmlSend> <producer_options> の <visibility> を指定する。	未サポート
シングル・コンシューマ・キューへのメッセージのエンキュー - メッセージ・プロパティ（優先順位、期限切れ）の指定	DBMS_AQ.enqueue MESSAGE_PROPERTIES の優先順位、期限切れを指定する。	AQQueue.enqueue AQMessage Property の優先順位、期限切れを指定する。	LNOCIAQEnq LNOCIAQMsgProperties の LNOCI_ATTR_PRIORITY、LNOCI_ATTR_EXPIRATION を指定する。	<AQXmlSend> <message_header> の <priority> および <expiration> を指定する。	QueueSender.send 時に、優先順位および TimeToLive を指定する。または MessageProducer.setTimeToLive、MessageProducer.setPriority などの後に、QueueSender.send

表 3-4 AQ プログラム環境の比較 : 操作インタフェース - シングル・コンシューマ・キューへのメッセージのエンキュー（ Point-to-Point モデルでの利用方法）（続き）

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
シングル・コンシューマ・キューへのメッセージのエンキュー - メッセージ・プロパティ（相関 ID、遅延、例外キュー）の指定	DBMS_AQ.enqueue MESSAGE_PROPERTIES の相関、遅延、exception_queue（例外キュー）を指定する。	AQQueue.enqueue AQMessage Property の相関、遅延、例外キューを指定する。	LNOCIAQEnq LNOCIAQMsgProperties の OCI_ATTR_CORRELATION、OCI_ATTR_DELAY、LNOCI_ATTR_EXCEPTION_QUEUE を指定する。	<AQXmlSend> <message_header> の <correlation_id>、<delay> および <exception_queue> を指定する。	Message.setJMS CorrelationID プロバイダ固有メッセージ・プロパティとして指定された遅延および例外キュー JMS_OracleDelay、JMS_OracleExcpQ などの後に、QueueSender.send
シングル・コンシューマ・キューへのメッセージのエンキュー - メッセージ・プロパティ（ユーザー定義）の指定	未サポート プロパティはペイロードの一部である必要がある。	未サポート プロパティはペイロードの一部である必要がある。	未サポート プロパティはペイロードの一部である必要がある。	<AQXmlSend> <user_properties> の <name>、<int_value>、<string_value>、<long_value> などを指定する。	Message.setInt Property、 Message.setString Property、Message.set BooleanProperty などの後に、QueueSender.send
シングル・コンシューマ・キューへのメッセージのエンキュー - メッセージ変換の指定	DBMS_AQ.enqueue ENQUEUE_OPTIONS の変換を指定する。	AQQueue.enqueue AQDequeueOption の変換を指定する。	LNOCIAQEnq LNOCIAQEnqOptions の OCI_ATTR_TRANSFORMATION を指定する。	<AQXmlSend> <producer_options> の <transformation> を指定する。	AQjmsQueue Sender.set Transformation などの後に、QueueSender.send

表 3-5 AQ プログラム環境の比較 : 操作インタフェース - マルチ・コンシューマ・キュー / トピックへのメッセージのパブリッシュ（パブリッシュ・サブスクライブ・モデルでの利用方法）

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
マルチ・コンシューマ・キュー / トピックへのメッセージのパブリッシュ（デフォルト・サブスクリプション・リストを使用）	DBMS_AQ.enqueue MESSAGE_PROPERTIES の受信者リストを NULL に設定する。	AQQueue.enqueue AQMessage Property の受信者リストを NULL に設定する。	LNOCIAQEnq LNOCIAQMsg Properties の OCI_ATTR_RECIPIENT_LIST を NULL に設定する。	<AQXmlPublish>	TopicPublisher.publish
マルチ・コンシューマ・キュー / トピックへのメッセージのパブリッシュ（特定の受信者リストを使用） 脚注 1 を参照	DBMS_AQ.enqueue MESSAGE_PROPERTIES の受信者リストを指定する。	AQQueue.enqueue AQMessageProperty の受信者リストを指定する。	LNOCIAQEnq LNOCIAQMsg Properties の OCI_ATTR_RECIPIENT_LIST を指定する。	<AQXmlPublish> <message_header> の <recipient_list> を指定する。	AQjmsTopicPublisher.pubish AQjmsAgent の配列として受信者を指定する。
マルチ・コンシューマ・キュー / トピックへのメッセージのパブリッシュ - メッセージ・プロパティ（優先順位、期限切れ）の指定	DBMS_AQ.enqueue MESSAGE_PROPERTIES の優先順位、期限切れを指定する。	AQQueue.enqueue AQMessage Property の優先順位、期限切れを指定する。	LNOCIAQEnq LNOCIAQMsg Properties の OCI_ATTR_PRIORITY、LNOCI_ATTR_EXPIRATION を指定する。	<AQXmlPublish> <message_header> の <priority> および <expiration> を指定する。	TopicPublisher.publish 時に、優先順位および TimeToLive を指定する。または Message Producer.setTimeToLive、Message Producer.setPriority などの後に、TopicPublisher.publish

表 3-5 AQ プログラム環境の比較 : 操作インタフェース - マルチ・コンシューマ・キュー / トピックへのメッセージのパブリッシュ（パブリッシュ・サブスクライブ・モデルでの利用方法）（続き）

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
マルチ・コンシューマ・キュー / トピックへのメッセージのパブリッシュ - 送信オプション（相関 ID、遅延、例外キュー）の指定	DBMS_AQ.enqueue MESSAGE_PROPERTIES の相関、遅延、exception_queue（例外キュー）を指定する。	AQQueue.enqueue AQMessage Property の相関、遅延、例外キューを指定する。	LNOCIAQEnq LNOCIAQMsg Properties の OCI_ATTR_CORRELATION、OCI_ATTR_DELAY、LNOCI_ATTR_EXCEPTION_QUEUE を指定する。	<AQXmlPublish> <message_header> の <correlation_id>、<delay> および <exception_queue> を指定する。	Message.setJMSCorrelationID プロバイダ固有メッセージ・プロパティとして指定された遅延および例外キュー JMS_OracleDelay、JMS_OracleExcpQ などの後に、TopicPublisher.publish
トピックへのメッセージのパブリッシュ - メッセージ・プロパティ（ユーザー定義）の指定	未サポート プロパティはペイロードの一部である必要がある。	未サポート プロパティはペイロードの一部である必要がある。	未サポート プロパティはペイロードの一部である必要がある。	<AQXmlPublish> <user_properties> の <name>、<int_value>、<string_value>、<long_value> などを指定する。	Message.setIntProperty、 Message.setStringProperty、Message.setBooleanProperty などの後に、TopicPublisher.publish
トピックへのメッセージのパブリッシュ - メッセージ変換の指定	DBMS_AQ.enqueue ENQUEUE_OPTIONS の変換を指定する。	AQQueue.enqueue AQDequeueOption の変換を指定する。	LNOCIAQEnq LNOCIAQEnq Options の OCI_ATTR_TRANSFORMATION を指定する。	<AQXmlPublish> <producer_options> の <transformation> を指定する。	AQjmsTopicPublisher.setTransformation などの後に、TopicPublisher.publish

表 3-6 AQ プログラム環境の比較 : 操作インタフェース - マルチ・コンシューマ・キュー / トピックのメッセージに対するサブスクライブ（パブリッシュ・サブスクライブ・モデルでの利用方法）

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
サブスクライバの追加	管理インタフェースを参照	管理インタフェースを参照	未サポート	未サポート	TopicSession.createDurableSubscriber AQjmsSession.createDurableSubscriber
サブスクライバの変更	管理インタフェースを参照	管理インタフェースを参照	未サポート	未サポート	TopicSession.createDurableSubscriber AQjmsSession.createDurableSubscriber 新しいセレクタを使用する。
サブスクライバの削除	管理インタフェースを参照	管理インタフェースを参照	未サポート	未サポート	AQjmsSession.unsubscriber

表 3-7 AQ プログラム環境の比較 : 操作インタフェース - キューのメッセージのブラウズでの利用方法

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
キュー / トピックのメッセージのブラウズ	DBMS_AQ.dequeue DEQUEUE_OPTIONS の dequeue_mode を BROWSE に設定する。	AQQueue.dequeue AQDequeueOption の dequeue_mode を BROWSE に設定する。	LNOCIAQDeq LNOCIAQDeq Options の OCI_ATTR_DEQ_MODE を BROWSE に設定する。	<AQXmlReceive> <consumer_options> の <dequeue_mode> を BROWSE に指定する。	QueueSession.createBrowser QueueBrowser.getEnumeration トピックでは未サポート oracle.jms.AQjmsSession.createBrowser oracle.jms.TopicBrowser.getEnumeration
キュー / トピックのメッセージのブラウズ - ブラウズ中のメッセージのロック	DBMS_AQ.dequeue DEQUEUE_OPTIONS の dequeue_mode を LOCKED に設定する。	AQQueue.dequeue AQDequeueOption の dequeue_mode を LOCKED に設定する。	LNOCIAQDeq LNOCIAQDeq Options の OCI_ATTR_DEQ_MODE を LOCKED に設定する。	<AQXmlReceive> <consumer_options> の <dequeue_mode> を LOCKED に指定する。	locked を TRUE に設定した AQjmsSession.createBrowser QueueBrowser.getEnumeration トピックでは未サポート oracle.jms.AQjmsSession.createBrowser oracle.jms.TopicBrowser.getEnumeration

表 3-8 AQ プログラム環境の比較 : 操作インタフェース - キュー / トピックからのメッセージの受信での利用方法

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
メッセージ受信用のコネクションの起動	不可	不可	不可	不可	Connection.start
メッセージ・コンシューマの作成	不可	不可	不可	不可	QueueSession.create QueueReceiver TopicSession.create DurableSubscriber AQjmsSession.create TopicReceiver
キュー / トピックからのメッセージのデキュー - 可視性の指定	DBMS_AQ.dequeue ENQUEUE_OPTIONS の可視性を指定する。	AQQueue.dequeue AQDequeueOption の可視性を指定する。	LNOCIAQDeq LNOCIAQDeq Options の OCI_ATTR_VISIBILITY を指定する。	<AQXmlReceive> <consumer_options> の <visibility> を指定する。	未サポート
キュー / トピックからのメッセージのデキュー - 変換の指定	DBMS_AQ.dequeue ENQUEUE_OPTIONS の変換を指定する。	DBMS_AQ.dequeue AQDequeueOption の変換を指定する。	LNOCIAQDeq LNOCIAQDeq Options の OCI_ATTR_TRANSFORMATION を指定する。	<AQXmlReceive> <consumer_options> の <transformation> を指定する。	AQjmsQueueReceiver.setTransformation AQjmsTopicSubscriber.setTransformation AQjmsTopicReceiver.setTransformation
キュー / トピックからのメッセージのデキュー - ナビゲーション・モードの指定	DBMS_AQ.dequeue ENQUEUE_OPTIONS のナビゲーションを指定する。	DBMS_AQ.dequeue AQDequeueOption のナビゲーションを指定する。	LNOCIAQDeq LNOCIAQDeq Options の OCI_ATTR_NAVIGATION を指定する。	<AQXmlReceive> <consumer_options> の <navigation> を指定する。	AQjmsQueueReceiver.setNavigationMode AQjmsTopicSubscriber.setNavigationMode AQjmsTopicReceiver.setNavigationMode

表 3-8 AQ プログラム環境の比較 : 操作インタフェース - キュー / トピックからのメッセージの受信での利用方法（続き）

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
シングル・コンシューマ・キューからのメッセージのデキュー	DBMS_AQ.dequeue DEQUEUE_OPTIONS の dequeue_mode を REMOVE に設定する。	AQQueue.dequeue AQDequeueOption の dequeue_mode を REMOVE に設定する。	LNOCIAQDeq LNOCIAQDeq Options の OCI_ATTR_DEQ_MODE を REMOVE に設定する。	<AQXmlReceive>	QueueReceiver.receive または QueueReceiver.receiveNoWait または AQjmsQueueReceiver.receiveNoData
マルチ・コンシューマ・キュー / トピックからのメッセージのデキュー（サブスクリプション名の使用）	DBMS_AQ.dequeue DEQUEUE_OPTIONS の dequeue_mode を REMOVE に、また、consumer_name をサブスクリプション名に設定する。	AQQueue.dequeue AQDequeueOption の dequeue_mode を REMOVE に、また、consumer_name をサブスクリプション名に設定する。	LNOCIAQDeq LNOCIAQDeq Options の OCI_ATTR_DEQ_MODE を REMOVE に、また、OCI_ATTR_CONSUMER_NAME をサブスクリプション名に設定する。	<AQXmlReceive> <consumer_options> の <consumer_name> を指定する。	サブスクリプション名を使用してトピック上に永続的な TopicSubscriber を作成した後 TopicSubscriber.receive または TopicSubscriber.receiveNoWait または AQjmsTopicSubscriber.receiveNoData
マルチ・コンシューマ・キュー / トピックからのメッセージのデキュー（受信者名の使用）	DBMS_AQ.dequeue DEQUEUE_OPTIONS の dequeue_mode を REMOVE に、また、consumer_name を受信者名に設定する。	AQQueue.dequeue AQDequeueOption の dequeue_mode を REMOVE に、また、consumer_name を受信者名に設定する。	LNOCIAQDeq LNOCIAQDeq Options の OCI_ATTR_DEQ_MODE を REMOVE に、また、OCI_ATTR_CONSUMER_NAME を受信者名に設定する。	<AQXmlReceive> <consumer_options> の <consumer_name> を指定する。	受信者名を使用してトピック上に TopicReceiver を作成した後 AQjmsSession.createTopicReceiver AQjmsTopicReceiver.receive または AQjmsTopicReceiver.receiveNoWait または AQjmsTopicReceiver.receiveNoData

表 3-9 AQ プログラム環境の比較 : 操作インタフェース - キュー / トピックからメッセージを非同期受信するための登録での利用方法

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
シングル・コンシューマ・キューからのメッセージの非同期受信	PL/SQL コールバック・プロシージャを定義する。 DBMS_AQ.register を使用して PL/SQL コールバック・プロシージャを登録する。	未サポート	LNOCI Subscription Register queue_name をサブスクリプション名として指定する。 LNOCI Subscription Enable	<AQXmlRegister> <destination> のキュー名および <notify_url> の通知メカニズムを指定する。	キュー上に QueueReceiver を作成した後 QueueReceiver.set MessageListener
マルチ・コンシューマ・キュー / トピックからのメッセージの非同期受信	PL/SQL コールバック・プロシージャを定義する。 DBMS_AQ.register を使用して PL/SQL コールバック・プロシージャを登録する。	未サポート	LNOCI Subscription Register キュー OCI_ATTR_CONSUMER_NAME をサブスクリプション名として指定する。 LNOCI Subscription Enable	<AQXmlRegister> が <destination> のキュー名、<consumer_name> のコンシューマおよび <notify_url> の通知メカニズムを指定する。	トピック上に TopicSubscriber または TopicReceiver を作成した後 TopicSubscriber.set MessageListener TopicReceiver.set MessageListener

表 3-9 AQ プログラム環境の比較 : 操作インタフェース - キュー / トピックからメッセージを非同期受信するための登録での利用方法（続き）

利用方法	PL/SQL	Java（ネイティブ AQ）	OCI	AQ XML サーブレット	JMS
複数キュー / トピック上でのメッセージのリスニング	-	-	-	-	-
1 つ（または多数）のシングル・コンシューマ・キュー上でのメッセージのリスニング	DBMS_AQ.listen agent_name を agent_list のすべてのエージェントで NULL として使用する。	未サポート	LNOCIAQListen agent_name を agent_list のすべてのエージェントで NULL として使用する。	未サポート	QueueSession 上に複数の QueueReceiver を作成した後 QueueSession.set MessageListener
1 つ（または多数）のマルチ・コンシューマ・キュー / トピック上でのメッセージのリスニング	DBMS_AQ.listen agent_list のすべてのエージェントの agent_name を指定する。	未サポート	LNOCIAQListen agent_list のすべてのエージェントの agent_name を指定する。	未サポート	TopicSession 上に複数の TopicSubscriber または TopicReceiver を作成した後 TopicSession.set MessageListener

4

AQ の管理

この章では、アドバンスト・キューイングの管理について説明します。内容は次のとおりです。

- セキュリティ
- Oracle 8.1 形式のキュー
- キュー表のエクスポート / インポート
- Oracle Enterprise Manager のサポート
- XA でのアドバンスト・キューイングの使用
- キュー管理の制限事項
- 伝播の問題点
- Oracle 8.0 形式のキュー

セキュリティ

構成情報は、DBMS_AQADM パッケージ内のプロシージャを使用して管理できます。最初は、SYS および SYSTEM のみに、DBMS_AQADM および DBMS_AQ 内のプロシージャに対する実行権限が付与されています。この 2 つのパッケージに対する実行権限を付与されているユーザーは誰でも、自分自身のスキーマ内のキューを作成、管理および使用できます。他のスキーマのキューを作成および管理するには、MANAGE ANY QUEUE 権限が必要です。

JMS または Java AQ API のユーザーは、DBMS_AQJMS（AQ_ADMINSTRATOR_ROLE および AQ_USER_ROLE を介しても使用可能）および DBMS_AQIN の実行権限が必要です。

管理者ロール

AQ_ADMINISTRATOR_ROLE には、キュー管理に必要なすべての権限が付与されます。ロールに付与された権限によって、権限受領者は次のことができるようになります。

- データベースのすべてのスキーマに対するすべてのキュー管理操作（キューやキュー表の作成など）の実行
- データベースのすべてのキューに対するエンキューおよびデキュー操作の実行
- そのキューの作業負荷を監視するための統計ビューへのアクセス
- DBMS_TRANSFORM を使用した変換の作成
- DBMS_AQELM 内のすべてのプロシージャの実行
- DBMS_AQJMS 内のすべてのプロシージャの実行

ユーザー・ロール

このロールでは、Oracle9*i* または 8.1 互換のキューにエンキューまたはデキューするために必要な権限を付与することができないため、Oracle9*i* および Oracle8*i* では、AQ_USER_ROLE を付与しないでください。

データベース管理者にはデータベース・ユーザーに直接システム権限 ENQUEUE ANY QUEUE および DEQUEUE ANY QUEUE を付与したり、DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE および DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE を行使する権限があります。アプリケーション開発者としては DBMS_AQADM.GRANT_QUEUE_PRIVILEGE および DBMS_AQADM.REVOKE_QUEUE_PRIVILEGE を行使することで、ユーザーはオブジェクト・レベルの権限を付与または取り消して、キューに権限を付与します。

データベース・ユーザーとして、所有するスキーマのキューにエンキューまたはデキューするためには、DBMS_AQ の実行権限の他に、オブジェクト・レベルまたはシステム・レベルの明示的な権限は必要ありません。

AQ オブジェクト型へのアクセス

現在、すべての内部 AQ オブジェクトには、PUBLIC でアクセスできます。

Oracle 8.1 形式のキュー

互換性

8.1 形式のキューでは、次の機能を使用するために、init.ora の compatible パラメータおよびキュー表の compatible パラメータを 8.1 に設定する必要があります。

- キュー・レベルのアクセス制御
- 非永続キュー（キュー表の互換性が 8.1 のとき、自動的に作成されます。）
- Oracle Parallel Server 環境のサポート
- パブリッシュ・サブスクライブ用のルールベースのサブスクライバ
- 非同期通知
- 送信元の識別
- 履歴管理情報の記憶域の分離

セキュリティ

Oracle9*i* データベースの AQ 管理者は、8.1 形式のキューを作成できます。AQ 8.1 のすべてのセキュリティ機能は 8.1 形式のキューで可能になります。AQ 8.1 セキュリティ機能が作用するのは、8.1 形式のキューに限られることに注意してください。キューを作成する場合、DBMS_AQADM.CREATE_QUEUE_TABLE の compatible パラメータのデフォルト値は 8.1 です。

表 4-1 に、8.1 形式のキューでサポートされている AQ セキュリティ機能、および同等の権限を示します。

表 4-1 8.1 形式のキューのセキュリティ

権限	8.1.x 以上のデータベースの 8.1.x 形式のキュー
AQ_USER_ROLE	未サポート。同等の権限は次のとおりです。 ■ dbms_aq の実行権限 ■ システム権限 ENQUEUE ANY QUEUE ■ システム権限 DEQUEUE ANY QUEUE ■ dbms_transform の実行権限

表 4-1 8.1 形式のキューのセキュリティ（続き）

権限	8.1.x 以上のデータベースの 8.1.x 形式のキュー
AQ_ADMINISTRATOR_ROLE	サポート
DBMS_AQ の実行権限	すべての AQ ユーザーに DBMS_AQ の実行権限が付与される必要があります。8.1 互換のキューにエンキューまたはデキューするには、ユーザーに次の権限が必要です。 ■ DBMS_AQ の実行権限 ■ ターゲット・キューに対するエンキュー権限またはデキュー権限、またはシステム権限 ENQUEUE ANY QUEUE/DEQUEUE ANY QUEUE のいずれか

権限およびアクセス制御

8.1 形式のキューにオブジェクト・レベルの権限を付与したり、取り消すことができます。また、様々なシステムレベル権限を付与したり取り消すことができます。次の表にすべての共通 AQ 操作、およびそれらの操作を Oracle9*i* または 8.1 互換のキューで実行するために必要な権限を示します。

表 4-2 操作および必要な権限

操作	必要な権限
所有するキューに対する CREATE/DROP/MONITOR	DBMS_AQADM の実行権限が必要です。その他の権限は不要です。
すべてのキューに対する CREATE/DROP/MONITOR	DBMS_AQADM の実行権限が必要です。また、AQ_ADMINISTRATOR_ROLE を付与されている他のユーザーによって、このロールが付与される必要があります（最初は、SYS および SYSTEM が AQ_ADMINISTRATOR_ROLE を付与します）。
所有するキューに対する ENQUEUE/ DEQUEUE	DBMS_AQ の実行権限が必要です。その他の権限は不要です。
他のユーザーが所有するキューに対する ENQUEUE/ DEQUEUE	DBMS_AQ の実行権限が必要です。また、所有者から、DBMS_AQADM.GRANT_QUEUE_PRIVILEGE を使用して権限が付与される必要があります。
すべてのキューに対する ENQUEUE/ DEQUEUE	DBMS_AQ の実行権限が必要です。また、AQ 管理者から、DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE を使用してシステム権限 ENQUEUE ANY QUEUE または DEQUEUE ANY QUEUE が付与される必要があります。

LNOCI アプリケーション

OCI アプリケーションで 8.1 形式のキューにアクセスするには、そのセッション・ユーザーに、アクセス先のキューのオブジェクト権限、あるいはシステム権限 ENQUEUE ANY QUEUE または DEQUEUE ANY QUEUE が付与されている必要があります。アクセス先のキューが Oracle9*i* または 8.1 互換の場合、DBMS_AQ に対する実行権限がそのセッション・ユーザーの権限と照合して確認されることはありません。

伝播に必要なセキュリティ

AQ は、データベース・リンクを介してメッセージを伝播します。伝播ドライバは、ソース・キューの所有者としてソース・キューからデキューします。そのため、ソース・キューに対する明示的なアクセス権限が付与される必要はありません。宛先では、そのデータベース・リンクにログインしているユーザーは、ENQUEUE ANY QUEUE 権限か、その宛先キューにエンキューする権限のいずれかが付与されている必要があります。ただし、データベース・リンクのログイン・ユーザーが宛先のキュー表を所有している場合は、どのような明示的な AQ 権限も付与される必要はありません。

キュー表のエクスポート / インポート

キュー表がエクスポートされると、キュー表データおよび PL/SQL コードの無名ブロックがエクスポート・ダンプ・ファイルに書き込まれます。キュー表がインポートされると、インポート・ユーティリティがこれらの PL/SQL 無名ブロックを実行して、メタデータをデータ・ディクショナリに書き込みます。

キュー表データのエクスポート

キューをエクスポートすると、必然的に基になるキュー表および関連するディクショナリ表もエクスポートされます。キューは、キュー表単位でのみエクスポートできます。

複数の受信者がいるキュー表のエクスポート

複数の受信者をサポートするキュー表は、次の表と対応付けられています。

- デキュー索引構成表（IOT）
- 時間管理索引構成表
- サブスクライバ表（8.1 互換のキュー表用）
- 履歴索引構成表（8.1 互換のキュー表用）

全データベース・モードおよびユーザー・モード・エクスポートでは、これらの表は自動的にエクスポートされますが、表モード・エクスポートではエクスポートされません。4-6 ページの「エクスポート・モード」を参照してください。

このメタデータ表には、キュー表内の一部の行の ROWID が含まれているため、メタデータ表をインポートするときに、廃止される ROWID についての注意情報が生成されます。廃止される ROWID は、キューイング・システムのインポート操作の一部として自動的に修正されるため、このメッセージは無視してもかまいません。ただし、インポート中に他の問題（ロールバック・セグメント領域の不足など）が発生した場合は、この問題を修正し、インポートを繰り返す必要があります。

エクスポート・モード

エクスポートは、全データベース・モード、ユーザー・モードおよび表モードで、次のとおり操作されます。キュー表の増分エクスポートはサポートされていません。

- 全データベース・モード : キュー表、すべての関連表、システム・レベルの権限付与、およびプライマリ・オブジェクトとセカンダリ・オブジェクト権限付与が、自動的にエクスポートされます。
- ユーザー・モード : キュー表、すべての関連表およびプライマリ・オブジェクト権限付与が自動的にエクスポートされます。
- 表モード : お薦めできません。キュー表を表モードでエクスポートする場合、そのキュー表に属するすべての関連表をエクスポートする必要があります。たとえば、8.1 互換のマルチ・コンシューマ・キュー表 MCQ をエクスポートするとき、次の各表もエクスポートする必要があります。

```
AQ$_<queue_table>_I (the dequeue IOT)
AQ$_<queue_table>_T (the time-management IOT)
AQ$_<queue_table>_S (the subscriber table)
AQ$_<queue_table>_H (the history IOT)
```

キュー表データのインポート

エクスポートと同様に、キューをインポートすると、必ず基になるキュー表および関連するディクショナリ・データもインポートされます。キュー表データがインポートされると、インポート・ユーティリティがダンプ・ファイルの PL/SQL 無名ブロックを実行して、メタデータをデータ・ディクショナリに書き込みます。

複数の受信者がいるキュー表のインポート

複数の受信者をサポートするキュー表は、次の表と対応付けられています。

- デキュー IOT
- 時間管理 IOT
- サブスクライバ表（8.1 互換のキュー表用）
- 履歴 IOT（8.1 互換のキュー表用）

そのキュー自体のキュー表の他に、これらの表もインポートする必要があります。

インポート IGNORE パラメータ

すでにデータがあるキュー表には、キュー・データをインポートしないでください。キュー表をインポートする場合、インポート・ユーティリティの IGNORE パラメータを常に NO に設定する必要があります。IGNORE パラメータに YES が設定され、すでに存在しているキュー表がダンプ・ファイル中の表定義と互換性があるとき、ダンプ・ファイルの各行は既存の表にロードされます。それと同時に、古いキュー表定義およびキュー定義は削除および再作成されます。そのため、インポート以前に作成されたキュー表およびキュー定義は失われ、複製された行がキュー表の中に現れます。

AQ 管理者およびユーザーの作成

AQ 管理者としてのユーザーの作成

ユーザーを AQ 管理者として設定するには、次の手順が必要です。

```
CONNECT system/manager
CREATE USER aqadm IDENTIFIED BY aqadm;
GRANT AQ_ADMINISTRATOR_ROLE TO aqadm;
GRANT CONNECT, RESOURCE TO aqadm;
```

さらに、AQ パッケージの実行権限を、次のように付与します。

```
GRANT EXECUTE ON DBMS_AQADM TO aqadm;
GRANT EXECUTE ON DBMS_AQ TO aqadm;
```

これによって、ユーザー・プロシージャから AQ パッケージに含まれるプロシージャを実行できます。

ユーザー AQUSER1 および AQUSER2 の作成

所有するスキーマにキューを作成およびアクセスできる AQ ユーザーを作成する場合、AQ_ADMINISTRATOR_ROLE の付与を除いて、5-8 ページの「AQ 管理者としてのユーザーの作成」で説明した手順に従います。

```
CONNECT system/manager
CREATE USER aquser1 IDENTIFIED BY aquser1;
GRANT CONNECT, RESOURCE TO aquser1;
```

さらに、AQ パッケージの実行権限を、次のように付与します。

```
GRANT EXECUTE ON DBMS_AQADM to aquser1;
GRANT EXECUTE ON DBMS_AQ TO aquser1;
```

キューを作成しないで、別のスキーマのキューを使用する AQ ユーザーを作成する場合は、まず、前の項で説明した手順に従います。さらに、オブジェクト・レベル権限を付与しま

す。ただし、これは 8.1 互換のキュー表を使用して定義されたキューにのみ適用されることに注意してください。

```
CONNECT system/manager
CREATE USER aquser2 IDENTIFIED BY aquser2;
GRANT CONNECT, RESOURCE TO aquser2;
```

さらに、AQ パッケージの実行権限を、次のように付与します。

```
GRANT EXECUTE ON DBMS_AQADM to aquser2;
GRANT EXECUTE ON DBMS_AQ TO aquser2;
```

aquser2 が aquser1 スキーマの aquser1_q1 キューにアクセスするために、aquser1 は次の文を実行する必要があります。

```
CONNECT aquser1/aquser1
EXECUTE DBMS_AQADM.GRANT_QUEUE_PRIVILEGE(
  'ENQUEUE','aquser1_q1','aquser2',FALSE);
```

Oracle Enterprise Manager のサポート

Oracle Enterprise Manager は、ほぼすべてのアドバンスト・キューイング管理機能をサポートします。AQ 機能は、Enterprise Manager コンソールのナビゲーション・ツリーの分散ノードに表示されます。Enterprise Manager で使用可能な機能は、次のとおりです。

- スキーマ・マネージャに含まれるキュー項目を使用した各種プロパティの参照
- キューの作成、開始、停止および削除
- 伝播のスケジューリングおよびスケジュール解除
- サブスクライバの追加および削除
- データベースのすべてのキューに対する伝播スケジュールの表示
- データベースのすべてのキューに対するエラーの表示
- メッセージ・キューの表示
- 権限の付与および取消し
- 変換の作成、変更または削除

XA でのアドバンスト・キューイングの使用

AQ OCI インタフェースを使用する場合は、xa_open 文字列で「Objects=T」を指定する必要があります。これは、オブジェクト・モードで、XA にクライアント側キャッシュを初期化させます。OCI または Pro*C から、PL/SQL ラッパーを介して AQ を使用する場合は、この処理は必要ありません。AQ は（LOB として格納されるにもかかわらず）単純な RAW バッファに抽象化されるため、Pro* シリーズのマニュアルに記載されている LOB メモリー管理の概念は、RAW 型のメッセージには当てはまりません。

サービス間（xa_start 境界と xa_end 境界の間）でデキュー処理を続ける場合に、AQ ナビゲーション・オプションを使用するときは、FIRST_MESSAGE を使用してデキュー位置をリセットする必要があります。これは、xa_end が終了すると、XA がカーソル・フェッチ状態を取り消すためです。リセットしないと、「ORA-25237: ナビゲーション・オプションの指定順序が正しくありません。」というエラー・メッセージが表示されます。

キュー管理の制限事項

キュー管理の制限事項については、次の項目を参照してください。

- メッセージ・ペイロード（実際に通信される情報）内のコレクション型
- キュー表およびキューにおけるシノニム
- 表領域の Point-in-Time リカバリ
- 非永続キュー

> **注意：** キュー名およびキュー表名は、大文字に変換されます。大文字と小文字が混在しているキュー名およびキュー表名はサポートされません。

メッセージ・ペイロード（実際に通信される情報）内のコレクション型

オブジェクトに含まれていない VARRAY を使用して、メッセージ・ペイロードを組み立てることはできません。また、現時点では、メッセージ・ペイロード内で埋込みオブジェクトとしてネストした表を使用できません。ただし、1 つ以上の VARRAY を含むオブジェクト型を作成して、このオブジェクト型に基づくキュー表を作成できます。

たとえば、次の操作は有効です。

```
CREATE TYPE number_varray AS VARRAY(32) OF NUMBER;
CREATE TYPE embedded_varray AS OBJECT (col1 number_varray);
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE(
  queue_table           =>     'QT',
  queue_payload_type    =>     'embedded_varray');
```

キュー表およびキューにおけるシノニム

どの AQ PL/SQL コールも、キューおよびキュー表におけるシノニムを解決しません。ユーザーはシノニムを作成できますが、そのシノニムを AQ インタフェースに適用しないでください。

表領域の Point-in-Time リカバリ

AQ は、現時点で表領域の Point-in-Time リカバリをサポートしていません。表領域にキュー表を作成すると、Point-in-Time リカバリが使用できなくなります。

非永続キュー

現在、RAW 型およびユーザー定義型の非永続キューは作成できます。また、それらのメッセージを送信できるのは、サブスクライバおよびローカルな明示的受信者に対してのみです。非永続キューからの伝播は、サポートされていません。さらに、メッセージを取り出すときはデキュー・コールではなく、LNOCISubcriptionRegister によって通知を登録する非同期通知メカニズムを使用する必要があります。

伝播の問題点

伝播は、システム・キュー aq$_prop_notify_X（X は、スケジュールのソース・キューが常駐するインスタンスのインスタンス番号）を使用して、伝播ランタイム・イベントを操作します。このキューのメッセージは、システム表 aq$_prop_table_X（X は、スケジュールのソース・キューが常駐するインスタンスのインスタンス番号）に格納されます。

注意： 伝播が正常に動作するために、キュー aq$_prop_notify_X が中止または削除されたり、表 aq$_prop_notify_X が削除されることがないようにしてください。

伝播に必要な実行権限

伝播ジョブの所有者は SYS ですが、伝播が発生するのは、キュー表の所有者のセキュリティ・コンテキスト内です。以前は、伝播ジョブの所有者は、伝播をスケジューリングするユーザーで、伝播が発生するのは、伝播スケジュールを設定するユーザーのセキュリティ・コンテキスト内でした。キュー表の所有者には、DBMS_AQADM パッケージの実行権限が必要です。権限がない場合、Oracle スナップショット・プロセスは、エラー識別子 SYS.DBMS_AQADM が定義されていないトレース・ファイルを伝播および生成しません。伝播には、キュー表の所有者が所有するプライベート・データベース・リンクを使用できます。接続文字列に指定されたユーザーには、リモート・データベース上の DBMS_AQ パッケージおよび DBMS_AQADM パッケージに対する実行権限が必要です。

ジョブ・キュー・プロセス数

このスケジューリング・アルゴリズムには、伝播できるジョブ・キュー・プロセスを少なくとも 2 つ使用可能にする必要があるという制限事項があります。非伝播の関連ジョブがあるときは、より多くのジョブ・キュー・プロセスが必要になります。負荷が大きくなる条件（多数のアクティブ・スケジュールがあって、そのすべてに伝播が必要なメッセージがあるとき）が予測されるときは、非伝播ジョブも同時にキュー・プロセスを使用することを認識して、より多くのジョブ・キュー・プロセスを開始する必要があります。伝播ジョブのみを持っているシステムでは、2 つのジョブ・キュー・プロセスによってすべてのスケジュールを操作できます。ただし、ジョブ・キュー・プロセス数が多くなると、メッセージの伝播も速くなります。1 つのジョブ・キュー・プロセスが複数のスケジュールによってメッセージを伝播できるため、ジョブ・キュー・プロセス数はスケジュール数と同じである必要はありません。

伝播の最適化

JOB_QUEUE_PROCESSES 数を設定するとき、DBA は、この値が、伝播するメッセージの伝播元のキュー数、およびメッセージの伝播先の宛先（キューではない）の数によって決まることを理解しておく必要があります。

伝播は、スケジューリング・アルゴリズムによって処理されます。このアルゴリズムは使用可能なジョブ・キュー・プロセスを最適化し、メッセージがソース・キューにエンキューされてから宛先に現れるまでの時間を最小限にするため、OLTP とほぼ同様の動作を実現します。アルゴリズムが操作できるスケジュールの数は無制限で、様々な障害に対しても対処できます。伝播は、使用可能なジョブ・キュー・プロセスの最適使用を試みますが、開始するジョブ・キュー・プロセスの数はレプリケーション・ジョブのような非伝播関連ジョブの存在にも依存します。したがって、このスケジューリング・アルゴリズムによって最高の結果を得るためには、次のガイドラインを利用することが重要です。

スケジューリング・アルゴリズムは、次のようにジョブ・キュー・プロセスを使用します（ここでは、アクティブ・スケジュールが適切に設定されているとします）。

- アクティブ・スケジュール数がジョブ・キュー・プロセスの半数に満たない場合、アクティブ・スケジュール数と同数のジョブ・キュー・プロセスが取得されます。
- アクティブ・スケジュール数がジョブ・キュー・プロセス数の半数を超える場合、ジョブ・キュー・プロセスの半数を取得した後で、取得した各ジョブ・キュー・プロセスに複数のアクティブ・スケジュールを割り当てます。
- システムがオーバーロードになっている（すべてのスケジュールが伝播のためにビジーになっている）場合、可用性に応じて、ジョブ・キュー・プロセス総数 -1 になるまで追加のジョブ・キュー・プロセスが取得されます。
- あるプロセスによって操作されるアクティブ・スケジュールがどれも伝播の必要なメッセージを持っていない場合、そのジョブ・キュー・プロセスは解放されます。

- このアルゴリズムによって、負荷が大きいプロセスから小さいプロセスにスケジュールを転送することで自動ロード・バランスを行い、オーバーロードのプロセスがなくなるようにします。

伝播中の障害対策

スケジューリング・アルゴリズムは、様々な障害に対して強力に対処できます。様々な障害のために、メッセージが伝播されないことがあります。よくある原因としては、「データベース・リンクの障害」、「リモート・データベースが使用不可」、「リモート・キューが存在しない」、「リモート・キューが開始されていない」、「リモート・キューにエンキューしようとしたときのセキュリティ違反」などがあります。このような状況では、適切なエラー・メッセージが DBA_QUEUE_SCHEDULES ビューにレポートされます。あるスケジュールでエラーが発生すると、そのスケジュールによるメッセージ伝播は、指数バックオフ・アルゴリズムを使用して最大 16 回まで周期的に試行され、その後は使用不可になります。エラーの原因になっていた問題が解決してスケジュールが使用可能になった場合、最新のエラー日付、時刻およびメッセージ・フィールドにそのエラー情報が残ります。このフィールドがリセットされるのは、そのスケジュールによってメッセージが正常に伝播されたときです。指数バックオフの段階が進むと、伝播が試行される間隔は数時間から数日になることがあります。エラーが長時間見過ごされた場合に、このような問題が発生します。このような状況では、その伝播スケジュールを解除して、再度スケジューリングするようにしてください。

オブジェクト・キューからの伝播

AQ では、ペイロードに BFILE または REF 属性があるオブジェクト・キューからの伝播はサポートされていないことに注意してください。

AQ 伝播問題のデバッグについてのガイドライン

ここでは、ソースおよびターゲット・データベースにキュー表およびキューを作成し、接続先データベースに対するデータベース・リンクを定義していると想定します。表記法は、（カッコなしで）エンティティの実際の名前を提供するものと想定します。

デバッグを開始するには、次の手順を実行します。

1. イベント 24040、レベル 10 を使用して、伝播トレースを最高レベルで ON にします。

   伝播の発生時に、デバッグ情報がジョブ・キュー・トレース・ファイルにログされます。トレース・ファイルで、エラーがないか、またメッセージが送信されたことを示す文を確認できます。

2. データベース 2 に対するデータベース・リンクを確認します。

   これを行うには、次のように入力します。

   ```
   select count(*) from @
   ```

3. 伝播スケジュールが作成され、ジョブ・キュー・プロセスが割り当てられたことを確認します。

   dba_queue_schedules および aq$_schedules のエントリを検索します。aq$_schedules に jobno があり、job$ または dbms_jobs にその jobno のエントリがあることを確認します。

4. 2 つ以上のジョブ・キュー・プロセスが実行中であることを確認します。

5. 次のように入力して、ソース・キューのメッセージを確認します。
   ```
   select count(*) from where q_name = '<queue_name>';
   ```

6. 同じように入力して、宛先キューのメッセージを確認します。

7. 誰がジョブ・キュー・プロセスを使用しているかを確認します。

   他のジョブによって、伝播ジョブの処理時間が失われる可能性がないかどうかを確認します。

8. sys.aq$_prop_table_ が dba_queue_tables に存在し、また aq$_prop_notify_ キューが dba_queues に存在することを確認します。これらは、ジョブ・キュー・プロセス間の通信に使用されます。

9. 宛先キューからメッセージをデキューするコンシューマが、伝播されたメッセージの受信者であることを確認します。

   8.1 形式のキューでは、次のように入力します。

   ```
   select consumer_name, deq_txn_id, deq_time, deq_user_id,
     propagated_msgid from aq$
     where queue = '<queue_name>';
   ```

   8.0 形式のキューでは、次のように入力して、キュー表の履歴列から同じ情報を取得できます。

   ```
   select h.consumer, h.transaction_id, h.deq_time, h.deq_user,
   h.propagated_msgid from t, table(t.history) h
   where t.q_name = '<queue_name>';
   ```

   または、次のように入力します。

   ```
   select consumer, transaction_id, deq_time, deq_user,
   propagated_msgid from
   the(select cast(history as sys.aq$_dequeue_history_t)
   from  where q_name = '<queue_name>');
   ```

Oracle 8.0 形式のキュー

8.0 形式のキューおよび 8.1 以上の互換性を持つデータベースを使用する場合、次の機能は使用できません。

- Oracle Parallel Server 環境のサポート
- 非同期通知

これらの機能を使用するには、8.1 以上の形式のキューに移行する必要があります。

参照 :

- 4-5 ページの「伝播に必要なセキュリティ」を参照してください。
- 『Oracle9*i* データベース移行ガイド』を参照してください。

8.0 への移行および 8.0 からの移行

8.0 形式のキュー表を 8.1 形式のキュー表にアップグレードしたり、8.1 形式のキュー表を 8.0 形式のキュー表にダウングレードするには、DBMS_AQADM.MIGRATE_QUEUE_TABLE を使用します。表 4-3 に、DBMS_AQADM.MIGRATE_QUEUE_TABLE のパラメータを示します。

構文

```
DBMS_AQADM.MIGRATE_QUEUE_TABLE(
    queue_table         IN      VARCHAR2,
    compatible          IN      VARCHAR2)
```

表 4-3 DBMS_AQADM_MIGRATE_QUEUE_TABLE パラメータ

パラメータ	説明
queue_table (IN VARCHAR2)	移行するキュー表の名前を指定します。
compatible	8.0 のキュー表を 8.1 互換のキュー表にアップグレードする場合は、8.1 に設定します。8.1 のキュー表を 8.0 互換のキュー表にダウングレードする場合は、8.0 に設定します。

例 : 8.0 互換のキュー表を 8.1 互換のキュー表にアップグレードする方法

注意 : 次のようなデータ構造を設定しないと機能しない例もあります。

```
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
   queue_table            => 'qtable1',
   multiple_consumers     => TRUE,
   queue_payload_type     => 'aq.message_typ',
   compatible             =>'8.0');
```

```
EXECUTE DBMS_AQADM.MIGRATE_QUEUE_TABLE(
   queue_table  => 'qtable1',
   compatible   => '8.1');
```

8.0 形式のキューのインポートおよびエクスポート

このメタデータ表には、キュー表内の一部の行の ROWID が含まれているため、メタデータ表をインポートするときに、廃止される ROWID についての注意情報が生成されます。廃止される ROWID は、キューイング・システムのインポート操作の一部として自動的に修正されるため、このメッセージは無視してもかまいません。ただし、インポートまたはエクスポート中に他の問題（ロールバック・セグメント領域の不足など）が発生した場合は、この問題を修正し、インポートまたはエクスポートを繰り返す必要があります。

8.0 のロール

Oracle8 の AQ 操作には、AQ プロシージャに実行権限を付与するロールを通じてアクセスできます。Oracle8 を使用するときにデータベース・オブジェクト・レベルの制御がないということは、Oracle8 で AQ_USER_ROLE を持つユーザーは、システム内のすべてのキューに対してエンキューおよびデキューできるということを意味します。ファイングレイン・アクセス・コントロールを実現するには、8.1 以上の互換性を持つデータベースで、8.1 形式のキュー表を使用します。

Oracle9*i* または Oracle8*i* データベースの AQ 管理者は、8.0 互換のキューを作成できます。また、8.0 形式のキューは 8.0 互換のセキュリティ機能によって保護されます。

本来、Oracle8 のデータベース用に作成されたキューで 8.1 のセキュリティ機能を使用するときは、DBMS_AQADM.MIGRATE_QUEUE_TABLE を実行して、そのキュー表を 8.1 形式に変換する必要があります。

> **参照 :** DBMS_AQADM.MIGRATE_QUEUE_TABLE の詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

データベースのダウングレードが必要になった場合は、8.1 形式のすべてのキュー表を 8.0 互換に変換するか、ダウングレードを実行する前に削除する必要があります。変換によって、

オブジェクト権限と同様に、Oracle9*i* または Oracle8*i* のすべてのセキュリティ機能は削除されます。キューが 8.0 互換に変換されると、そのキューには Oracle8 のセキュリティ・モデルが適用され、Oracle8 のセキュリティ機能のみがサポートされます。

8.0 形式のキューのセキュリティ

表 4-4 に、8.0 形式のキューでサポートされている AQ セキュリティ機能、および同等の権限を示します。

表 4-4 8.0.x 形式のキューのセキュリティ

権限	8.0.x データベースの 8.0.x 形式のキュー	8.1.x データベースの 8.0.x 互換のキュー
AQ_USER_ROLE	サポート 権限受領者は、そのロールを介して DBMS_AQ の実行権限が付与されます。	サポート 権限受領者は、そのロールを介して DBMS_AQ の実行権限が付与されます。
AQ_ADMINISTRATOR_ROLE	サポート	サポート
DBMS_AQ の実行権限	PL/SQL で AQ アプリケーションを作成する開発者には、DBMS_AQ の実行権限が付与される必要があります。	PL/SQL で AQ アプリケーションを作成する開発者には、DBMS_AQ の実行権限が付与される必要があります。

AQ オブジェクト型へのアクセス

grant_type_access プロシージャは、8.0 形式のキューに関して、リリース 8.1.5 で廃止されました。

LNOCI アプリケーションによる 8.0 形式のキューへのアクセス

OCI アプリケーションで 8.0 形式のキューにアクセスするには、そのセッション・ユーザーに DBMS_AQ に対する実行権限が付与されている必要があります。

トランスポータブル表領域および 8.0 形式のマルチ・コンシューマ・キュー

8.0 形式のマルチ・コンシューマ・キュー表を含む表領域は、トランスポータブル表領域メカニズムでは転送されません。ただし、シングル・コンシューマ・キューや 8.1 互換のマルチ・コンシューマ・キューを含む表領域では、そのメカニズムが有効です。トランスポータブル・モードで表領域をエクスポートするには、その前に、その表領域を読込み専用モードに変更する必要があります。8.0 形式のマルチ・コンシューマ・キューを含む読込み専用表領域をインポートしようとすると、キュー表の索引をインポート時に更新できないという Oracle エラーになります。

DBMS_AQADM パッケージの自動コミット機能

DBMS_AQADM パッケージの CREATE_QUEUE_TABLE、DROP_QUEUE_TABLE、CREATE_QUEUE、DROP_QUEUE および ALTER_QUEUE コールの自動コミット・パラメータは、リリース 8.1.5 以上では使用しないでください。Oracle は、下位互換性を保つために、引き続きこのパラメータをサポートします。

5

パフォーマンスおよび拡張性

この章の内容は次のとおりです。

- パフォーマンスの概要
- 基本的なチューニングのヒント
- 伝播のチューニングのヒント

パフォーマンスの概要

キューは、データベース表に格納されます。キュー操作のパフォーマンス特性は、基になるデータベース操作のパフォーマンス特性と同等です。エンキュー操作のコード・パスは、3つの IOT を持つ複数列のキュー表に対する SELECT および INSERT と比較できます。デキュー操作のコード・パスは、同様の表に対する SELECT、DELETE および UPDATE と比較できます。

Oracle Real Application Clusters 環境における AQ

Oracle Real Application Clusters を使用することで、キュー・データに対する高可用性のアクセスを実現できます。キューの先頭および末尾は、かなりのホット・スポット（混雑点）になる可能性があります。ホット・スポットが存在すると、Oracle Real Application Clusters は十分に拡張できない場合があるため、1 つのインスタンスからキューに対する通常のアクセスを制限してください。インスタンス障害が発生した場合、障害が発生したインスタンスによって管理されていたメッセージは、障害がないインスタンスの 1 つによってすぐに処理されます。

共有サーバー環境におけるアドバンスト・キューイング

キュー操作の拡張性は、基になるデータベース操作の拡張性と同等です。共有サーバー環境で wait オプション付きのデキュー操作が発行されると、共有サーバー・プロセスでは、待機時間を含むコールの所要時間中、デキュー操作のみが行われます。このようなプロセスが多数存在すると、パフォーマンスおよび拡張性に重大な問題が発生し、共有サーバー・プロセスがデッドロック状態となる可能性があります。そのため、wait オプション付きのデキュー・リクエストは、専用サーバー・プロセスを使用して発行することをお薦めします。この制限は、強制ではありません。

基本的なチューニングのヒント

アドバンスト・キューイング表のレイアウトは、普通のデータベース表および索引のレイアウトと同様であると考えられます。

> **参照：** チューニングの推奨項目の詳細は、『Oracle9*i* データベース・パフォーマンス・チューニング・ガイドおよびリファレンス』を参照してください。

エンキュー・プロセスとデキュー・プロセスの同時実行（単一のキュー表の場合）

環境によっては、メッセージを一定の割合で処理する必要があります。そのため、エンキュー・プロセスおよびデキュー・プロセスの両方を同時に実行する必要がある場合があります。メッセージ配信システムに 1 つのキュー表および 1 つのキューしか存在しない場合、すべてのプロセスが同じセグメント領域で同時に行われる必要があるため、大量のメッセージを妥当なパフォーマンス・レベルで配信することができなくなります。

同時プロセスの最適数は、使用可能なシステム・リソースに基づいて定義する必要があります。たとえば、CPU を 4 つ持つシステムでは、2 つの同時エンキュー・プロセスおよび 2 つの同時デキュー・プロセスから始めるのが妥当です。システムを介して配信されるメッセージの最適数を実現できない場合、プロセスの数を増やすのではなく、いくつかのサブスクライバを使用してロード・バランスを行ってください。

エンキュー・プロセスとデキュー・プロセスのシリアル実行（単一のキュー表の場合）

エンキュー・プロセスとデキュー・プロセスが同時に実行していない（メッセージがエンキューされてからデキューされる）場合、同時プロセスの場合より、同一データ・セグメント上の競合は減少します。このような場合、システムがメッセージ配信に必要な合計時間は、同時プロセスの場合より長くなります。プロセス数を増加させることは、エンキュー・プロセスおよびデキュー・プロセスの両方に効果的です。プロセス数を増加すると、メッセージのスループット率はデキュー元よりエンキュー元で高くなります。通常、デキュー操作のスループットは、エンキュー操作（INSERT）のスループットより大幅に劣ります。これは、デキュー操作は SELECT、DELETE および UPDATE を実行するためです。

伝播のチューニングのヒント

伝播は、リモート（またはローカル）・キュー表に対して追加の INSERT を行う特殊なデキュー操作です。単一スケジュールからの伝播は、複数のジョブ・キュー・プロセスにわたってパラレル化されず、かわりにロード・バランス化されます。拡張性を向上させるには、使用可能なシステム・リソース（CPU）に基づいて伝播スケジュールの数を設定します。

トランザクション処理をサポートするキュー表およびトランザクション処理をサポートしない（デフォルト）キュー表からの伝播率は、ある程度変化します。これは、非トランザクション・キューのバッチ処理サイズは Oracle が決定するのに対して、トランザクション・キューのバッチ・サイズは主にユーザー・アプリケーションが決定するためです。

6

FAQ

この章では、アドバンスト・キューイングに関して、よくある質問およびそれに対する回答を示します。内容は次のとおりです。

- 一般的な質問
- JMS に関する質問
- インターネットのアクセスに関する質問
- Oracle Internet Directory に関する質問（グローバル・エージェント、グローバル・イベントおよびグローバル・キュー）
- 変換に関する質問
- パフォーマンスに関する質問
- インストールに関する質問

一般的な質問

キュー表に残っているデキュー済メッセージにアクセスする方法は？

メッセージには SQL を使用してアクセスします。キュー表のメッセージは、保存されているか、まだ処理されていません。キューごとにビューがあります（10-33 ページの「データベース全体における状態ごとのメッセージ数の選択」を参照）。

メッセージの保存とはメッセージがそこにあることですが、これらのメッセージにサブスクライバがアクセスする方法は？

通常、サブスクライバは、デキュー・インタフェースを使用してメッセージにアクセスします。ただし、処理済または待機中のメッセージを参照する場合は、メッセージ ID でデキューするか、または SQL を使用します。

キュー表作成後でもソート順序は変更できますか？

キュー表を作成した後は、メッセージのソート順序を変更することはできません。

例外キューからデキューする方法は？

マルチ・コンシューマ・キューに対する例外キューも、マルチ・コンシューマ・キューである必要があります。

マルチ・コンシューマ・キュー内の期限切れメッセージを、メッセージの対象受信者がデキューすることはできません。ただし、デキュー・オプションのコンシューマ名を NULL に指定すると、REMOVE モードで 1 回のみデキューできます。メッセージ ID を指定すると、メッセージも例外キューからデキューできます。

マルチ・コンシューマ例外キューを作成したキュー表が、互換パラメータを使用しないで作成されたか、または互換パラメータを 8.0 に設定して作成された場合、期限切れメッセージをデキューできる方法は、メッセージ ID を指定する方法のみです。

伝播のスケジューリングでは、待機時間（latency）パラメータは何を意味しますか？

伝播スケジュールで待機時間を 0（ゼロ）以外に指定すると、ジョブは指定された待機時間の後に実行されるようにスケジュールが再設定されます。ジョブが実際に実行する時間は、準備ジョブの数や job_queue_processes の数など、他の要因に依存します。また、job_queue_interval の値にも影響される場合があります。ジョブ・キューおよび SNP バックグラウンド・プロセスの詳細は、『Oracle9*i* データベース管理者ガイド』の「ジョブ・キューの管理」を参照してください。

キュー表を作成する表領域の制御方法は？

DBMS_AQADM.CREATE_QUEUE_TABLE の storage_clause パラメータを使用して、キュー表およびそのすべての補助オブジェクトを格納するための表領域を指定できます。ただし、一度表領域を指定すると、そのキュー表に対して作成されたすべての IOT および索引は、指定された表領域に作成されます。現在、これらを異なる表領域間に分割することはできません。

Oracle Parallel Server のインスタンス・アフィニティをキュー表に対応付ける方法は？

Oracle8*i* では、Oracle Parallel Server のインスタンス・アフィニティをキュー表に対応付けできます。異なるインスタンスで q1 および q2 を使用している場合、キュー表で ALTER_QUEUE_TABLE を使用（create_queue_table も可）して primary_instance に適切なインスタンス ID を設定できます。

メッセージ・プロパティ、メッセージ・データ・プロパティを含むサブスクライバ・ルールの例を教えてください。

メッセージ・プロパティを指定する場合の簡単なルールは、rule = 'priority=1' です。次に、メッセージ・プロパティとデータ属性の組合せを指定するルールの例を示します。

```
rule = 'priority 1 AND tab.userdata.sal 1000' rule = '((priority
between 0 AND 3) OR correlation = 'BACK_ORDERS') AND
tab.userdata.customer_name like ''JOHN DOE'')'
```

ユーザー・データ・プロパティまたは属性は、オブジェクト・ペイロードのみに適用され、常に接頭辞として tab.userdata を付ける必要があります。これ以外の例については、ドキュメントを参照してください。

通知登録（OCI）とリスナーの開始は同じですか？

同じではありません。登録とは、非同期通知（プッシュ）に使用される OCI クライアント・コールです。登録では、メッセージがデキュー可能になった場合に、サーバーからクライアントへの通知が提供されます。メッセージが使用可能な場合、クライアント側の関数（コールバック）が、サーバーによって起動されます。通知登録は、非ブロックで非ポーリングです。

非永続キューの使用目的は何ですか？

現在接続されているすべてのユーザーへの通知のメカニズムを提供します。非永続キューのメカニズムによって、非永続キューに対するメッセージのエンキューがサポートされます。また、OCI 通知は、通知に対して現在登録されているユーザーに、このようなメッセージを配信するために使用されます。

受信者リストの長さに制限はありますか？または、特別なキューに対するサブスクライバの数に制限はありますか？

あります。各キューに対するサブスクライバまたは受信者の数は、1024 に制限されています。

UNDELIVERABLE メッセージが表示されたキューをクリーンアウトする方法は？

これらのメッセージは、msgid でデキューできます。キュー表ビューを問い合せることで msgid を検索できます。メッセージは、最終的に例外キューに移されます（これを行うには、AQ バックグラウンド・プロセスが実行中である必要があります）。通常のデキュー方法で、例外キューからこれらのメッセージをデキューできます。

メッセージ・ペイロードは、エンキューした後で更新できますか？

メッセージをデキューし、再度エンキューすることによってのみ更新できます。メッセージ・ペイロードを変更している場合は、これは別のメッセージになります。

非同期通知を使用して、新しいメッセージを受信するたびに実行可能ファイルを起動できますか？

通知は、OCI クライアントに対してのみ可能です。クライアントは、通知を受信するためにデータベースに接続している必要はありません。クライアントは、各メッセージに対して実行されるコールバック関数を指定します。非同期通知を使用して実行可能ファイルを起動することはできませんが、コールバック関数でストアド・プロシージャを起動することはできます。

伝播はマルチ・コンシューマ・キューからシングル・コンシューマ・キューへ（およびその逆で）動作しますか？

マルチ・コンシューマ・キューからシングル・コンシューマ・キューへの伝播は可能です。その逆は不可能です（シングル・コンシューマ・キューからの伝播はできません）。

デキュー時に ORA-01555 エラーが発生する場合があるのはなぜですか？

デキューに対して NEXT_MESSAGE ナビゲーション・オプションを使用している可能性があります。このオプションでは、最初のデキュー・コール中に作成されたスナップショットが使用されます。その後、他のデキュー・コールによって、UNDO が生成されてロールバック・セグメントに書き込まれ、ORA-01555 エラーが戻されます。

これを解決するには、FIRST_MESSAGE オプションを使用してメッセージをデキューします。これによってカーソルが再実行され、新しいスナップショットが取得されます。これは、正常に実行しない場合があるため、たとえば、1 つのメッセージに FIRST_MESSAGE を使用し、次の 1000 のメッセージに NEXT_MESSAGE を使用し、再び FIRST_MESSAGE を使用するというように、メッセージをバッチでデキューすることをお薦めします。

サブスクライバ表に記録されているサブスクライバ・タイプの違いは何ですか？

サブスクライバ・タイプおよびその値は、次のとおりです。

1 - Current Subscriber。サブスクライバの名前、アドレスおよびプロトコルは同じ行にあります。

2 - Ex subscriber。サブスクライブしていないが、履歴 aq$_queuetable_h IOT にエージェント・エントリを持っていたサブスクライバです。

4 - Address。受信者のアドレスを格納するために使用されます。名前は常に NULL です。アドレスは常に NULL 以外です。

8 - Proxy for Propagation。名前は常に NULL です。

ローカル・キューへのデータベース・プロキシの場合、address=NULL、protocol=0 です。

リモート・キューへのデータベース・プロキシの場合、address=dblink アドレス、protocol=0 です。

サード・パーティ・プロキシの場合、address=NULL、protocol= サード・パーティ・プロトコルです。

メッセージが例外キューに移動した後、SQL またはその他を使用して、例外キューに移動する前にメッセージが常駐していたキューを識別する方法はありますか？

ありません。AQ では、このような情報は提供されません。この問題を回避するために、アプリケーションでそのメッセージの情報を保存できます。

多くのメッセージが同時にエンキューされている場合、メッセージはどのような順序でデキューされますか？

メッセージの enq_time が同じである場合、（同じ enq_time を持つ各メッセージに対して）一定の単位で増加する step_no という別のフィールドがあります。これが、メッセージ順序のメンテナンスに有効です。同じセッションからエンキューされた 2 つ以上のメッセージに対して、enq_time と step_no の両方が同じであることはありません。

AQ および OMB の使用時期を教えてください。

Oracle9*i* では、OMB の機能は Oracle データベースで提供されています。したがって、Oracle9*i* データベースをご使用の場合は、データベースで提供される機能を使用してください。

OMB 自体は必要ありません。

Oracle8*i* をご使用の場合、次のような場合に OMB を使用してください。

- MQ Series との統合
- HTTP フレームワークの使用

これ以外の場合は、データベースから直接 JMS 機能を使用してください。

仮想プライベート・データベースで AQ を使用できますか？

できます。AQ キュー表でセキュリティ・ポリシーを指定できます。デキュー中に、適用するポリシーのデキュー条件（deq_cond）または相関識別子を使用します。デキュー条件として「1=1」を使用できます。デキュー条件または相関識別子を使用しない場合、デキューでエラーが発生します。

保存したメッセージをクリーンアップする方法は？

アドバンスト・キューイングの保存機能を使用して、メッセージ処理後の有効期限を指定し、その期限の経過後にメッセージを自動的にクリーンアップできます。

不適切なサブスクライバに対するメッセージを挿入したアプリケーションがあります。そのようなメッセージをクリーンアップする方法は？

サブスクライバ名またはメッセージ ID を使用してデキューできます。これによってメッセージが処理され、保存期間が経過した後にクリーンアップされます。

複数の Oracle データベース間で伝播を実行していますが、なんらかの理由で接続先データベースの 1 つが長時間ダウンしています。その接続先に対するメッセージをクリーンアップする方法は？

特定のサブスクライバに対するメッセージをクリーンアップするには、そのサブスクライバを削除して、その後再び追加します。サブスクライバを削除すると、そのサブスクライバに対するすべてのメッセージが削除されます。

メッセージ・ゲートウェイに関する質問

メッセージ・ゲートウェイ・ログ・ファイルはどこにありますか？

デフォルトでは、メッセージ・ゲートウェイ・ログ・ファイルは $ORACLE_HOME/mgw/log ディレクトリにあります。この位置は、mgw.ora ファイルの log_directory パラメータで変更できます。新しいログ・ファイルは、MGW エージェントを起動すると、常に、作成されます。このログ・ファイル名のフォーマットは、「oramgw-hostname-timestamp-processid.log」です。

メッセージ・ゲートウェイ・ログ・ファイルの例外メッセージを解析する方法は？

MGW ログ・ファイルに記録された例外メッセージには、問題を特定するために有効な 1 つ以上のリンクされた例外（[Linked-exception]）が含まれる場合があります。java.sql.SQLException には Oracle エラー・メッセージが含まれる場合があり、また多くの場合、PL/SQL スタック・トレースも含まれます。

次の例では、dbms_mgwadm.db_connect_info のデータベース・パラメータに無効な値（「bad_service_name」）が指定された場合の MGW ログ・ファイルのエントリを示します。この場合、MGW エージェントではデータベース接続が確立されません。

```
>>2002-01-15 15:45:12  MGW  AdminMgr  0 LOG
Connecting to database using connect string = jdbc:oracle:oci8:@BAD_SERVICE_NAME
>>2002-01-15 15:45:15  MGW  Engine  0 3
Agent is shutdown.
oracle.mgw.admin.MgwAdminException: [241]  Failed to connect to database. SQL error:
12154, connect string: jdbc:oracle:oci8:@BAD_SERVICE_NAME
[…Java stack trace here…]

[Linked-exception]
java.sql.SQLException: ORA-12154: TNS:could not resolve service name
[…Java stack trace here…]
```

メッセージ・ゲートウェイ・エージェントが実行中であることを確認する方法は？

MGW_GATEWAY ビューを使用して、ゲートウェイのステータス情報を表示します。AGENT_STATUS フィールドおよび AGENT_PING フィールドに、現在のエージェントのステータスが表示され、エージェントがアクティブであり、ping に応答しているかどうかが示されます。MGW エージェントを起動すると、AGENT_STATUS の値は次のとおり推移します。

1. NOT_STARTED
2. START_SCHEDULED
3. STARTING
4. INITIALIZING
5. RUNNING

メッセージ・ゲートウェイ・エージェントの実行中にデータベースが停止またはクラッシュした場合、このエージェントは自動的に再起動しますか？

データベースの停止またはクラッシュ後、MGW エージェントが自動的に再起動する場合としない場合があります。MGW エージェントは、データベースを停止する前に停止する必要があります。MGW エージェントの実行中にデータベースに対して SHUTDOWN NORMAL を実行すると、MGW エージェントによってこのデータベースへの接続が保持されているため、データベースは停止しません。IMMEDIATE または ABORT では、正常に終了する時間がある場合、エージェントは再起動しません。そうでない場合、エージェントは、次回のデータベース起動時に再起動します。

メッセージ・ゲートウェイ・エージェントの実行中にデータベースが停止しないのはなぜですか？

MGW エージェントがデータベースとの接続を確立しているため、SHUTDOWN NORMAL コマンドではデータベースを停止できません。データベースを停止する前に、dbms_mgwadm.shutdown をコールして MGW エージェントを停止します。

MGW_GATEWAY ビューに常に AGENT_STATUS が START_SCHEDULED と表示されるのはなぜですか？

メッセージ・ゲートウェイは、Oracle データベース内のジョブ・キューを使用して MGW エージェント・プロセスを開始します。バックグラウンドでキューされたジョブを実行するには、1 つ以上のジョブ・キュー・プロセスが構成されている必要があります。ゲートウェイ・ジョブはすぐに実行されるようにスケジューリングされていますが、ジョブ・キュー・プロセスが使用可能になるまでは実行されません。ゲートウェイ・ステータスが長時間 START_SCHEDULED のままである場合は、ジョブ・キュー・プロセスがないか、または不足している状態でデータベース・インスタンスが起動されている可能性があります。メッセー

ジ・ゲートウェイでは、ジョブ・キュー・プロセスは、MGW エージェント・セッションの存続期間中保持されます。

十分な数のジョブ・キューがあり、そのうちの 1 つがメッセージ・ゲートウェイで使用可能な状態で、データベース・インスタンスが起動していることを確認する必要があります。推奨最小値は 2 です。

init.ora パラメータ :

JOB_QUEUE_PROCESSES で、インスタンスごとのジョブ・キュー・プロセスの数を指定します。

動的パラメータ :

```
ALTER SYSTEM SET JOB_QUEUE_PROCESSES = <number>;
```

メッセージ・ゲートウェイ・エージェントの起動後、MGW_GATEWAY ビューに AGENT_STATUS が NOT_STARTED と表示されるのはなぜですか ?

MGW_GATEWAY ビューには、ゲートウェイ・エージェントのステータス情報が表示されます。ステータスが NOT_STARTED の場合、エージェントは実行されていません。MGW エージェントの起動中または実行中に致命的エラーが発生した場合、LAST_ERROR_MSG フィールドは NULL 以外の値になります。

次の手順を実行します。

1. MGW ログ・ファイルが生成済であり、なんらかのエラーが示されていないかどうかを確認します。ログ・ファイルが存在しない場合、ゲートウェイ・エージェント・プロセスは開始されていない可能性が高いです。
2. リスナーが起動していることを確認します。
3. tnsnames.ora および listener.ora で指定されている値が適切であることを確認します。値が不適切または一致していない場合、リスナーは MGW エージェント・プロセスを開始しません。
4. mgw.ora で指定されている値が適切であることを確認します。値が不適切な場合、MGW エージェントはエラーが発生して終了することがあります。
5. エラーで示された問題を修正し、MGW エージェントを起動します。

MGW_GATEWAY ビューの LAST_ERROR_MSG に「ORA-28575: 外部プロシージャ・エージェントへの RPC 接続をオープンできません。」と表示された場合はどうすればよいですか？

- リスナーが起動していることを確認します。listener.ora が変更されている場合、その変更が有効になる前にリスナーを停止して再起動する必要があります。
- tnsnames.ora には、MGW_AGENT というネット・サービス名エントリが必要です。このエントリは、Windows NT のメッセージ・ゲートウェイには必要ありません。
- tnsnames.ora の MGW_AGENT ネット・サービス名の CONNECT_DATA に指定されている SID 値は、listener.ora の SID_DESC エントリの SID_NAME 値と一致している必要があります。
- MGW_AGENT ネット・サービス名が IPC 接続用に設定されている場合、tnsnames.ora と listener.ora の ADDRESS の KEY 値は一致している必要があります。
- tnsnames.ora または listener.ora のその他の値が適切であることを確認します。

MGW_GATEWAY ビューの LAST_ERROR_MSG に「ORA-32830: 結果コード <value> が Messaging Gateway エージェントによって戻されました。」と表示された場合はどうすればよいですか？

結果コードは次のいずれかになります。

-1：Java Virtual Machine（JVM）の起動中にエラーが発生しました。MGW ログ・ファイルに、次のいずれかの行を含むエントリがないかどうかを確認します。

- Can't create Java VM

  使用している Java のバージョンが適切であることを確認します。オペレーティング・システムのバージョンおよびパッチ・レベルが JDK のバージョンに適合していることを確認します。JVM ヒープ・サイズに適切な値が設定されていることを確認します。ヒープ・サイズは、dbms_mgwadm.alter_agent の max_memory パラメータで指定されます。

- Can't find class oracle.mgw.engine.Agent

  mgw.ora で設定されている CLASSPATH に mgw.jar が含まれているかどうか確認します。次に例を示します。

```
set CLASSPATH=<ORACLE_HOME>/mgw/classes/mgw.jar
```

-2：mgw.ora の読込み中にエラーが発生しました。ファイルが読込み可能であることを確認します。

-3：MGW ログ・ファイルの作成中にエラーが発生しました。ログ・ディレクトリが書込み可能であることを確認します。デフォルトの位置は、<ORACLE_HOME>/mgw/log です。

-100：MGW エージェントの JVM で、起動時にランタイム例外またはエラーが発生しました。

-101: MGW エージェントが致命的エラーによって停止しました。MGW ログ・ファイルを確認します。

メッセージ・ゲートウェイ・エージェントを起動しようとすると、メッセージ・ゲートウェイのログ・ファイルに「ORA-01034: Oracle は使用できません。」と表示されるのはなぜですか？

このエラーは、データベースが起動していないか、またはメッセージ・ゲートウェイ・エージェントでデータベースの接続に使用された環境が不適切であることを示している場合があります。

例 1

MGW のログ・ファイルに次の 2 つの Oracle エラーが表示されている場合

- ORA-01034: Oracle は使用できません。
- ORA-27101: 共有メモリー領域は存在しません

ゲートウェイ・エージェントはローカル IPC 接続を使用してデータベースに接続しようとしていますが、ORACLE_SID 値が不適切です。

ローカル接続は、dbms_mgwadm.db_connect_info がデータベース・パラメータに対して NULL 値でコールされたときに使用されます。ローカル接続する必要がある場合、MGW エージェント・プロセスに適切な ORACLE_SID 値を設定する必要があります。これを行うには、mgw.ora に次の行を追加します。

```
set ORACLE_SID = <sid_value>
```

dbms_mgwadm.db_connect_info がデータベース・パラメータに対して NULL 以外の値でコールされた場合は、ORACLE_SID を設定する必要はありません。この場合は、その値で tnsnames.ora のネット・サービス名が指定されます。

伝播ソースとして AQ シングル・コンシューマ・キューを使用できますか？

できません。伝播のソース・キューとして使用できるのは AQ マルチ・コンシューマ・キューのみです。

DELETE_PENDING というフラグが付いたメッセージ・ゲートウェイ・サブスクライバが削除されるのはいつですか？

サブスクライバを非強制的な方法で削除するために dbms_mgwadm.remove_subscriber がコールされ、MGW エージェントが実行中でないか、またはエージェントが実行中であってもその時点ですべての必要なクリーンアップ処理を実行できないとき、MGW サブスクライバに DELETE_PENDING というフラグが付きます。

次の場合に、MGW エージェントは DELETE_PENDING サブスクライバを削除しようとします。

1. エージェントの実行中に dbms_mgwadm.remove_subscriber がコールされたとき
2. MGW エージェントの起動時、DELETE_PENDING サブスクライバが検出されたとき

RAW ペイロードを伴う AQ キューの最大メッセージ・サイズはいくらですか？

RAW ペイロードを伴う AQ キューの場合は、MGW エージェントで 32512 バイト以下のメッセージを伝播できます。メッセージ・サイズが 32512 バイトを超える場合は、エージェントでメッセージをエンキューまたはデキューしようとすると、エラーが発生します。

メッセージ・ゲートウェイ・エージェントに使用される Oracle Real Application Clusters のインスタンスはどれですか？

DBMS_MGWADM.STARTUP プロシージャによって、実行時に MGW エージェントの外部プロセスを開始するジョブ・キューのジョブが送られます。インスタンスおよび強制実行を使用して、ジョブおよびインスタンス・アフィニティを制御できます。デフォルトでは、ジョブは、すべてのインスタンスで実行できるように設定されています。

伝播に関する質問

メッセージの伝播時期を制御する方法は？

MGW エージェントは、伝播サブスクライバおよびスケジュールが同一のソース・キュー、宛先キューおよび伝播の型に構成されている場合、メッセージを伝播します。伝播時期を制御するには、dbms_mgwadm.enable_propagation_schedule および dbms_mgwadm.disable_propagation_schedule を使用します。デフォルトでは、伝播スケジュールは、最初に作成した時に使用可能になります。

最初は無効になる伝播ジョブを作成するには、次の API を 1 から順にコールします。

1. dbms_mgwadm.schedule_propagation
2. dbms_mgwadm.disable_propagation_schedule
3. dbms_mgwadm.add_subscriber

Oracle9*i* リリース 2（9.2）では、伝播スケジュール・ウィンドウ・パラメータは使用されません。

メッセージが伝播中であるか、または例外キューへ移されたかどうかを確認する方法は？

MGW_SUBSCRIBERS ビューの PROPAGATED_MSGS フィールドには、正常に伝播されたメッセージの数が表示されます。EXCEPTIONQ_MSGS フィールドには、例外キューに移されたメッセージの数が表示されます。これらのフィールドの両方とも、MGW エージェントの起動時に、0（ゼロ）にリセットされます。

メッセージが伝播ジョブの例外キューに移されるのはいつですか？

MGW サブスクライバに例外キューが構成されている場合は、メッセージ変換障害のため MGW エージェントで最初の伝播障害が発生した時点で、MGW エージェントによってメッセージがその例外キューに移されます。メッセージ変換障害は、MGW ログ・ファイルで `oracle.mgw.common.MessageException` によって示されます。

メッセージ変換障害からリカバリする方法は？ oracle.mgw.common.MessageException が発生した場合に処理を続行する方法は？

メッセージ変換障害が発生した場合は、MGW ログ・ファイルに `oracle.mgw.common.MessageException` と記録されます。この場合、MGW エージェントは、障害の原因となったメッセージを伝播できない可能性があり、この伝播ジョブは最終的に無効になります。

ログ・ファイルに、AQ デキュー（発信伝播）または AQ エンキュー（着信伝播）に使用された変換ファンクションで例外が発生したために障害が発生したことが示されている場合は、変換ファンクションが適切かどうかを確認します。

MGW サブスクライバには、伝播例外キューを構成できます。メッセージ変換障害が発生した場合、MGW エージェントは、メッセージを例外キューに移した後で、伝播ジョブの処理を続行します。

障害が発生した伝播ジョブをリカバリする方法は？

伝播ジョブの処理中に障害が発生した場合、MGW エージェントは、ジョブを無効にする前に、指数バックオフ・スキームで最大 16 回ジョブを再試行します。

障害が発生した伝播ジョブからリカバリするには、次の手順を実行します。

1. MGW ログ・ファイルを調べ、障害の原因を判断して問題を修正します。メッセージ変換障害の場合は、MGW サブスクライバに例外キューを構成する必要がある場合があります。
2. `dbms_mgwadm.reset_subscriber` をコールして、サブスクライバの状態をリセットします。MGW エージェントは、障害が発生したジョブのリカバリを試行して、伝播を再試行します。

発信伝播ジョブで伝播に障害が発生した場合、メッセージがデフォルトの AQ 例外キューに移されるのはなぜですか？

デキューの試行時に障害が発生した場合に AQ によってメッセージが AQ 例外キューに移されるタイミングは、AQ キューの MAX_RETRIES パラメータで制御されます。デフォルト値は NULL で、Oracle 9*i* では、この値は 5 になります。

パラメータ値が小さすぎる場合は、MGW サブスクライバの処理中に MGW エージェントで連続して障害が発生すると、キュー内のメッセージが AQ 例外キューへ移される可能性があります。AQ 例外キューに移された AQ メッセージによって、関連付けられた MGW サブスクライバにリカバリ不可能な障害が発生します。伝播ソースとして使用される AQ キューの MAX_RETRIES パラメータは、16 以上（できるだけ大きい値を推奨）に設定する必要があります。

変換に関する質問

変換を使用する方法は？

MGW サブスクライバに変換を構成して、発信伝播の際の AQ デキューまたは着信伝播の際の AQ エンキューに使用できます。

次の手順を実行します。

1. 変換ファンクションを作成します。
2. MGW エージェントのユーザーまたは PUBLIC に、ファンクションおよびそのファンクションが参照するオブジェクト型への実行権限を付与します。
3. dbms_transform.create_transformation をコールして、変換を登録します。
4. dbms_mgwadm.add_subscriber をコールし、変換を使用して MGW サブスクライバを作成するか、または dbms_mgwadm.alter_subscriber をコールして既存のサブスクライバを変更します。

これらの API の変換パラメータに渡される値は、ファンクション名ではなく登録された変換名である必要があります。

変換で例外が発生した場合はどうなりますか？

変換ファンクションで例外が発生した場合は、メッセージ変換障害が発生し、MGW ログ・ファイルに oracle.mgw.common.MessageException が表示されます。

メッセージ・ゲートウェイ・ログ・ファイルに表示される変換の例外にはどのようなものがありますか？

MGW ログ・ファイルに記録される例外メッセージには、追加の情報を示すリンクされた例外が含まれています。リンクされた例外が java.sql.SQLException の場合は、Oracle エラー・メッセージが含まれる場合があり、また多くの場合、PL/SQL スタック・トレースも含まれます。

通常、ORA-25229 は、変換ファンクションによって PL/SQL 例外が発生したとき、または変換を使用しようとしてその他の Oracle エラーが発生したときに、AQ によって表示されます。

例 1

```
Errors occured during processing of subscriber SUB_MQ2AQ_2
oracle.mgw.common.GatewayException: [722]  Message transformation failed; queue:
MGWUSER.DESTQ_SIMPLEADT, transform:
MGWUSER.MGW_BASIC_MSG_TO_SIMPLEADT
[…Java stack trace here…]
[Linked-exception]
oracle.mgw.common.MessageException: [722]  Message transformation failed; queue:
MGWUSER.DESTQ_SIMPLEADT, transform:
MGWUSER.MGW_BASIC_MSG_TO_SIMPLEADT
[…Java stack trace here…]
[Linked-exception]
java.sql.SQLException: ORA-25229: error on transformation of message msgid:
9749DB80C85B0BD4E03408002086745E
ORA-00604: 再帰 SQL レベル 1 でエラーが発生しました。
ORA-00904: 列名が無効です。
[…Java stack trace here…]
```

変換の例外の原因には、次のものが考えられます。

1. MGW エージェントのユーザーが変換ファンクションに対する実行権限を持っていない場合があります。MGW_AGENT_ROLE に実行権限を付与し、その後エージェントのユーザーに MGW_AGENT_ROLE を付与するのみでは不十分です。変換ファンクションに対する実行権限を、エージェントのユーザーに直接付与するか、または PUBLIC に付与する必要があります。
2. 登録済の変換が存在するにもかかわらず、変換ファンクションが存在しない場合があります。変換ファンクションが存在しない場合は、再作成する必要があります。
3. MGW エージェントのユーザーが、例外で示されたキューのペイロード・オブジェクト型の実行権限を持っていない場合があります。MGW_AGENT_ROLE に実行権限を付与し、その後エージェント・ユーザーに MGW_AGENT_ROLE を付与するのみでは不十分です。オブジェクト型の実行権限を、エージェントのユーザーに直接付与するか、または PUBLIC に付与する必要があります。

例 2

```
Errors occured during processing of subscriber SUB_AQ2MQ_2
oracle.mgw.common.GatewayException: [703]  Failed to retrieve information for
transformation mgwuser.SAMPLEADT_TO_MGW_BASIC_MSG
[…Java stack trace here…]
```

例外に示された変換が存在しない場合があります。dbms_mgwadm.add_subscriber の変換パラメータでは、変換ファンクションではなく登録済の変換の名前が指定されることに注意してください。

例 3

```
Errors occured during processing of subscriber SUB_AQ2MQ_2
oracle.mgw.common.GatewayException: [703]  Failed to retrieve information for
transformation mgwuser.SAMPLEADT_TO_MGW_BASIC_MSG
[…Java stack trace here…]

[Linked-exception]
java.sql.SQLException: "from_type" is null
[…Java stack trace here…]
```

MGW エージェントのユーザーが、例外に示された from_type の変換に使用されるオブジェクト型に対する実行権限を持っていない場合があります。MGW_AGENT_ROLE に実行権限を付与し、その後エージェントのユーザーに MGW_AGENT_ROLE を付与するのみでは不十分です。オブジェクト型に対する実行権限を、エージェントのユーザーに直接付与するか、または PUBLIC に付与する必要があります。

例 4

```
Errors occured during processing of subscriber SUB_AQ2MQ_2
oracle.mgw.common.GatewayException: [703]  Failed to retrieve information for
transformation mgwuser.SAMPLEADT_TO_MGW_BASIC_MSG
[…Java stack trace here…]

[Linked-exception]
java.sql.SQLException: "to_type" is null
[…Java stack trace here…]
```

MGW エージェントのユーザーが、例外に示された to_type の変換に使用されるオブジェクト型に対する実行権限を持っていない場合があります。MGW_AGENT_ROLE に実行権限を付与し、その後エージェントのユーザーに MGW_AGENT_ROLE を付与するのみでは不十分です。オブジェクト型に対する実行権限を、エージェントのユーザーに直接付与するか、または PUBLIC に付与する必要があります。

JMS に関する質問

dbms_aqadm.add_subscriber および dbms_aqadm.remove_subscriber コールが発行された同じキューで、同時エンキューまたはデキューが実行された場合に、これらのコールがハングする場合があるのはなぜですか？

add_subscriber および remove_subscriber は、キューに対する管理操作です。AQ は、アプリケーションが管理コールおよび操作コールを同時に発行することを防止することはありませんが、これらの発行はシリアルに実行されます。メッセージをエンキューまたはデキューした保留トランザクションがコミットおよび保持するリソースを解放するまで、add_subscriber および remove_subscriber はブロックします。サブスクライバの追加および削除は、頻繁に発生しないと想定されています。これは、ほとんどの場合、アプリケーションの設定の一部として行われます。ご質問の動作は、ほとんどの場合問題ありません。これを解決するには、キューで他の操作が行われていないときに、設定またはクリーンアップ・フェーズで、add_subscriber および remove_subscriber へのコールを分離します。これによって、操作コールによるリソースの解放を、これらのコールがブロック状態のままで待機し続けることがなくなります。

TopicSession.createDurableSubscriber および TopicSession.unsubscribe コールによって、「ORA - 04020: オブジェクトをロックしようとしてデッドロックを検出しました。」というメッセージで JMS 例外が発生するのはなぜですか？

createDurableSubscriber および unsubscribe コールには、Topic への排他的アクセスが必要です。そのため、これらのコールが発行される前に、同じ Topic に対する保留 JMS 操作（送信 / パブリッシュ / 受信）がある場合、ORA - 04020 例外が発生します。

この問題を解決するには、次の 2 つの方法があります。

1. Topic に対して他の JMS 操作が行われていないときに、設定またはクリーンアップ・フェーズで、createDurableSubscriber および unsubscribe に対するコールを分離します。これによって、必要なリソースが他の JMS 操作コールによって保持されないようにできます。したがって、ORA - 04020 エラーは発生しません。
2. createDurableSubscriber および unsubscribe コールをコールする前に、TopicSession.commit コールを発行します。

AQ_ADMINISTRATOR_ROLE または AQ_USER_ROLE が、Java/JMS API を使用する AQ アプリケーションに対して機能しない場合があるのはなぜですか？

ロールの付与に加えて、次のように、パッケージに対する実行権限もユーザーに付与する必要があります。

- `grant execute on sys.dbms_aqin to<userid>`
- `grant execute on sys.dbms_aqjms to <userid>`

Oracle JVM 内の Java ストアド・プロシージャから JMS MessageListener を使用すると、java.security.AccessControlException が発生するのはなぜですか？

Oracle JVM 内の `MessageListener` を使用するには、次のいずれかを実行します。

1. `GRANT JAVASYSPRIV to <userid>`

```
call dbms_java.grant_permission ('JAVASYSPRIV',
'SYS:java.net.SocketPermission', '*',
'accept,connect,listen,resolve');
```

インターネットのアクセスに関する質問

iDAP とは何ですか？

iDAP は、Internet Data Access Presentation の略です。iDAP によって SOAP リクエストの本体にメッセージ構造が定義されます。iDAP メッセージでは、AQ リクエストおよびレスポンスが XML にカプセル化されます。iDAP は、エンキュー、デキュー、通知送信、通知登録、インターネットの標準転送プロトコルである HTTP や電子メールによる伝播などの AQ 操作の実行に使用されます。さらに、iDAP では、トランザクション、セキュリティ、変換、およびリクエストに対するキャラクタ・セット ID がカプセル化されます。

AQ インターネット・アクセス機能をサポートしている Web サーバーは何ですか？Apache を使用する必要がありますか？それとも、どの Web サーバーでも使用できますか？AQ インターネット・アクセスをサポートしているサーブレット・エンジンは何ですか？Tomcat は使用できますか？

AQ のインターネット・アクセス機能は Apache でサポートされます。この機能は、Tomcat または JServ サーブレット実行エンジンの他に、Apache での動作も保証されています。サーブレットは、Java Servlet 2.0 以上のインタフェースをサポートする他の Web サーバーおよびサーブレット実行エンジンでも動作します。

インターネット・エージェントを、Oracle Internet Directory に格納されている AQ エージェントと結びつける方法は？

Oracle Internet Directory（OID）内の AQ エージェントに別名を作成できます。この AQ エージェント別名を、インターネット経由で送信された iDAP ドキュメントで使用して、AQ 操作を実行できます。別名を使用することで、AQ エージェントの内部名を非公開にできます。

ユーザー定義の認証フレームワークを認証に使用できますか？

できます。AQ 操作を行う AQ サーブレットへの HTTP の `POST` リクエストが、Web サーバーによって認証される必要があります。たとえば、Apache では、次のようにして（基本的な認証を使用して）`aqserv/servlet` にインストールされたサーブレットへのアクセスを制限できます。この例では、サーブレットに `POST` リクエストを送信するすべてのユーザーは、`/apache/htdocs/userdb` のユーザー・ファイルを使用して認証されます。

```
<Location /aqserv/servlet>
    <Limit POST>
            AuthName "Restrict AQ Servlet Access"
            AuthType Basic
            AuthUserFile /apache/htdocs/userdb/users
            require valid-user
    </Limit>
</Location>
```

Oracle Internet Directory に関する質問（グローバル・エージェント、グローバル・イベントおよびグローバル・キュー）

どのイベントを Oracle Internet Directory（OID）に登録できますか？

すべてのイベント（システム・イベント、ユーザー・イベントおよびキューに関する通知）を OID に登録できます。システム・イベントは、データベースの起動、データベースの停止およびシステム・エラー・イベントです。ユーザー・イベントには、ユーザー・ログインとユーザー・ログオフ、SQL データ定義言語（DDL）文（`CREATE`、`DROP`、`ALTER`）および SQL データ操作言語（DML）文トリガーがあります。キューに関する通知には、OCI 通知、PL/SQL 通知および電子メールによる通知があります。

OID に格納されているエージェント情報の使用方法は？

OID に AQ エージェントの別名を作成できます。これらの別名は、AQ 操作（エンキュー、デキューおよび通知）の実行中に指定できます。これは、インターネット経由で AQ 操作を実行中に内部エージェント名を非公開にする必要があるときに特に有効です。別名は AQ 操作（iDAP リクエスト）で使用できます。

変換に関する質問

変換マッピングでエラーが発生した場合、エンキュー、デキューまたは伝播はどうなりますか？

メッセージのエンキューおよびデキューによって、アプリケーションにエラーが発生します。デキュー操作中にエラーが発生した場合、メッセージの再試行回数が増分されます。再試行回数が max_retries を超えた場合、メッセージは例外キューに移されます。伝播中にエラーが発生した場合、デキューと同様の方法で処理され、メッセージの伝播は正常に実行されません。伝播が再試行され、再試行回数がキューの max_retries を超えた場合、メッセージは例外キューに移されます。

XML データの変換方法は？

XML データは、次のいずれかの方法で変換できます。

- XMLType でサポートされている Extract メソッドを使用することによって、指定された XPath の式を適用した結果の XMLType オブジェクトを戻します。
- XSLPROCESSOR パッケージを使用して、XSLT 変換を適用することによって XMLType オブジェクトを変換する PL/SQL ファンクションを作成します。

パフォーマンスに関する質問

パフォーマンスに影響を及ぼさずに、表に含めることができるキューの最大数はいくらですか？

表内のキューの数によって、パフォーマンスが影響を受けることはありません。

伝播を使用してメッセージをあるキューから別のキューに移動する場合、メッセージを 1 つずつ移動するのではなく、バッチで移動するように最適化されていますか？

はい。これは最適化されており、伝播はバッチで実行されます。

リモート・キューが異なるデータベースにある場合、2 フェーズ・コミットの必要性を回避するために順序アルゴリズムを使用します。

メッセージを同じ宛先の複数キューに送信する必要がある場合、メッセージは、複数回送信されます。メッセージがその宛先の同じキューにある複数のコンシューマに送信される必要がある場合、メッセージは 1 回で送信されます。

キュー表に索引を作成するとどのような場合に有効ですか？索引の作成方法は？

キュー表に索引を作成すると、次のような場合に有効です。

a. 相関識別子を使用してデキューする場合 : デキューを迅速に処理するには、基になる `AQ$_<QueueTableName>` キュー表の `corr_id` 列に索引を作成します。

b. 条件を使用してデキューする場合 : ここで使用する条件を、基になるキュー表に対する `SELECT` 文の where 句と考えてください。`<QueueTableName>` に索引を作成して、この `SELECT` 文のパフォーマンスを向上できます。

AQ に対する Java（JMS）API と PL/SQL API のパフォーマンスを比較するとどうですか？

JMS API と PL/SQL API のパフォーマンスの比較評価は特に行っていません。一般的に、PL/SQL API は JMS API よりわずかに優れています。リリース 8.1.7 以上の JMS と PL/SQL の API のパフォーマンスは同等です。

インストールに関する質問

AQ のためのインターネット・アクセスを設定する方法は？どのようなコンポーネントが必要ですか？

詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。次に、AQ キューへのインターネット・アクセスの設定に必要な手順の概要を示します。

1. AQ サーブレットを設定します。Java Servlet 2.2 仕様をサポートするサーブレット実行エンジン（Tomcat など）を使用している場合は、`oracle.AQ.xml.AQxmlServlet` クラスを拡張するサーブレットを作成する必要があります。Java Servlet 2.0 仕様をサポートするサーブレット実行エンジン（Apache JServ など）を使用している場合は、`oracle.AQ.xml.AQxmlServlet20` クラスを拡張するサーブレットを作成する必要があります。データベース接続パラメータを指定するには、サーブレットに `init()` メソッドを実装します。

2. ユーザー認証を設定します。AQ サーブレットに POST リクエストを送信するすべてのユーザーを認証するように Web サーバーを構成します。認証されたユーザーのみが AQ サーブレットにアクセスできます。
3. ユーザー認可を設定します。DBMS_AQADM.CREATE_AQ_AGENT を使用して、AQ 操作を実行するために使用される AQ エージェント名を登録します。
DBMS_AQADM.ENABLE_DB_ACCESS を使用して、データベース・ユーザーに AQ エージェントをマップします。
4. これで、クライアントは、SOAP リクエストを作成し、HTTP の POST を使用して AQ サーブレットに送信できます。

電子メールを使用して AQ 操作を実行する方法は？

詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

インターネット経由で AQ 伝播を設定する方法は？

詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。原則として、AQ のインターネット・アクセス設定の手順に従ってください。接続先データベースは、インターネット・アクセス用に次のように設定する必要があります。

1. 接続元データベースで、dblink を作成します。プロトコルに HTTP を指定し、AQ サーブレットを実行している Web サーバーのホストとポート、および Web サーバー / サーブレット・コンテナでの認証用のユーザー名とパスワードを指定します。たとえば、Web サーバーが webdest.oracle.com のマシンで実行しており、ポート 8081 に対するリクエストをリスニングしている場合、データベースの接続文字列は次のようになります。

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=http)(HOST=webdest.oracle.com)(PO
RT=8081)))
```

Secure Sockets Layer（SSL）を使用している場合、接続文字列にプロトコルとして HTTPS を指定します。データベース・リンクは次のように作成します。

```
create public database link propdb connect to john identified by
welcome using
'(DESCRIPTION=(ADDRESS=(PROTOCOL=http)(HOST=webdest.oracle.com)(P
ORT=8081)))';
```

この場合、パスワード Welcome を持つユーザー John が Web サーバーの認証に使用されます。このユーザーは、AQ HTTP エージェントでもあります。

2. SSL が使用されている場合、接続元データベースで次の構文を入力して、Oracle Wallet を作成し、Wallet パスを指定します。

```
execute dbms_aqadm.set_aq_
propagationwallet('/home/myuid/cwallet.sso','welcome');
```

3. 接続先データベースで AQ サーブレットを配置します。
oracle.AQ.xml.AQxmlServlet20（Apache JServ などの Servlet 2.0 実行エンジンを

使用している場合）または oracle.AQ.xml.AQxmlServlet（Tomcat などの Servlet 2.2 実行エンジンを使用している場合）の拡張クラス AQPropServlet を作成します。このサーブレットは、接続先データベースに接続する必要があります。サーブレットは、Web サーバー上のパス aqserv/servlet に配置する必要があります。

注意： Oracle9*i* では、伝播サーブレット名は AQPropServlet、配置パスは aqserv/servlet に固定されています。

---

4. 接続先データベースで、伝播を実行するインターネット・ユーザー（この例の場合は John）の認可および認証を設定します。
5. dbms_aqadm.schedule_propagation('src_queue', 'propdb') をコールして、接続元サイトで伝播を開始します。

7

モデリングおよび設計

この章では、AQ のモデリングおよび設計の基礎について説明します。内容は次のとおりです。

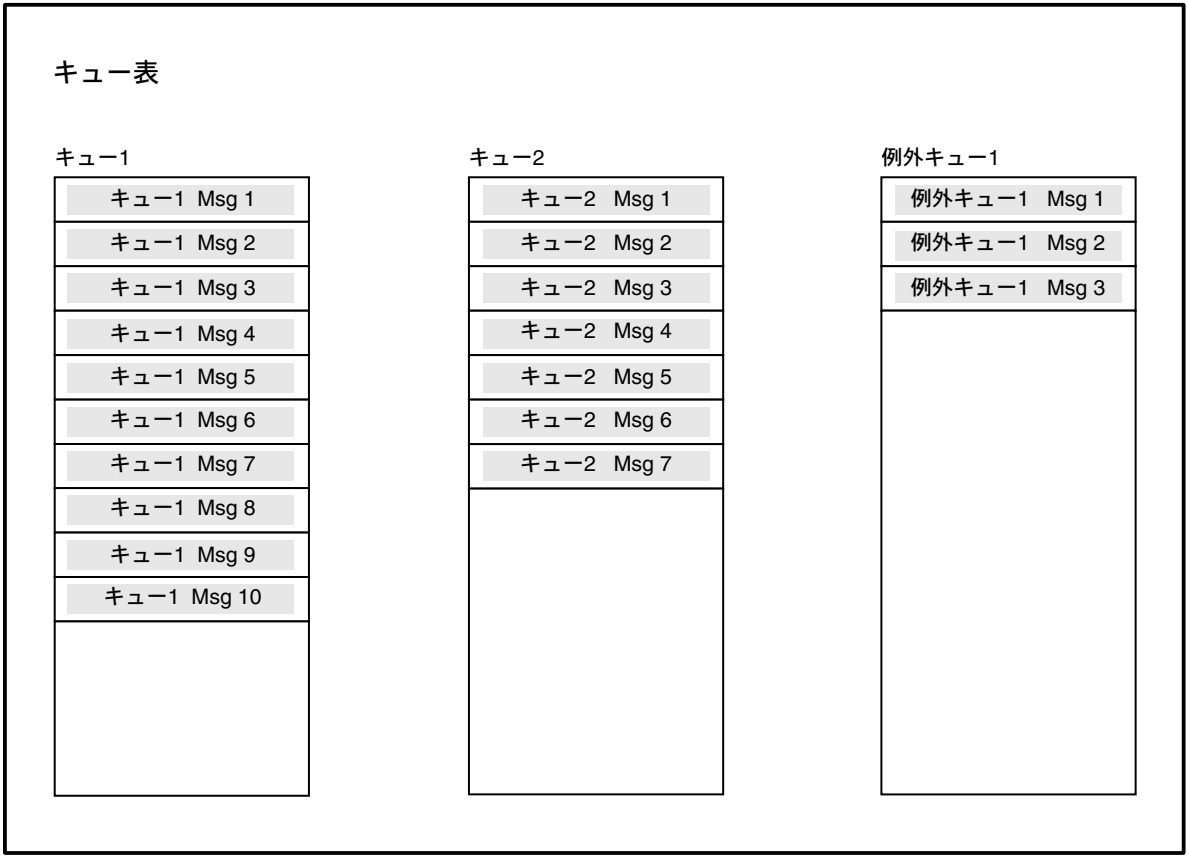
- 基本キューイング
- 基本キューイングの説明
- AQ を使用したクライアント / サーバー通信
- 複数のコンシューマによる同一メッセージのデキュー
- 指定された受信者による指定されたメッセージのデキュー
- ワークフローの AQ 実装
- パブリッシュ・サブスクライブの AQ 実装
- メッセージの伝播

キュー・エンティティのモデリング

図 7-1 に、次のキューおよびメッセージを含むキュー表を示します。

- キュー 1: 10 個のメッセージが含まれます。
- キュー 2: 7 個のメッセージが含まれます。
- 例外キュー 1: 3 個のメッセージが含まれます。

図 7-1 基本キュー



基本キューイング

基本キューイング - プロデューサ、コンシューマがともに 1 つの場合

最も基本的な場合、1 つのプロデューサが、1 つのキューに複数の異なるメッセージをエンキューします。各メッセージは、1 つのコンシューマによって 1 回のみデキューされ、処理されます。メッセージは、コンシューマによってデキューされるか、または期限が切れるまでは、キュー内に格納されたままです。プロデューサは、メッセージが使用可能になるまでの遅延および期限切れを指定できます。同様に、デキュー時に使用可能なメッセージがない場合は、コンシューマは待機できます。エージェント・プログラムまたはアプリケーションは、プロデューサおよびコンシューマの両方として動作できる点に注意してください。

基本キューイング - プロデューサが複数で、コンシューマが 1 つの場合

やや複雑な場合、多数のプロデューサがメッセージをキューにエンキューし、そのすべてのメッセージが 1 つのコンシューマによって処理されることがあります。

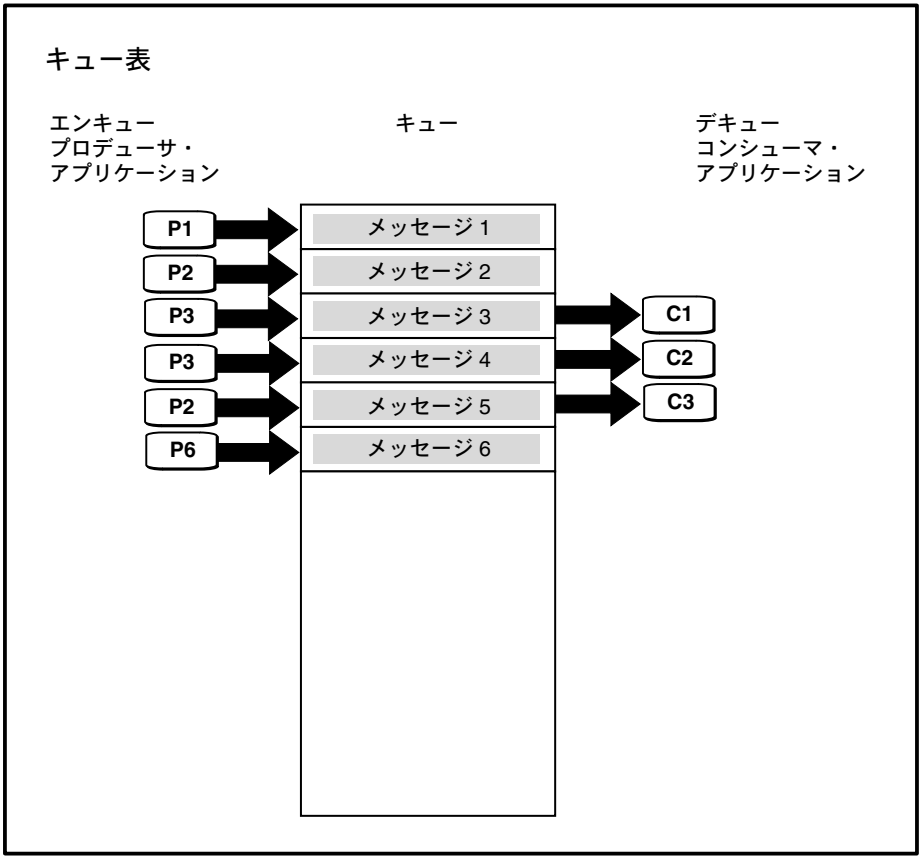
基本キューイング - 不連続なメッセージでプロデューサおよびコンシューマがともに複数の場合

次に、多数のプロデューサがメッセージをエンキューし、それぞれのメッセージを型および相関識別子によって異なるコンシューマが処理する場合を考えます。図 7-2 を参照してください。

基本キューイングの説明

図 7-2 に、メッセージがエンキューおよびデキューされる 1 つのキューを含むキュー表を示します。

図 7-2 基本キューイングのモデル



プロデューサ

この図では、メッセージのプロデューサが 6 つあり、そのうちの 4 つのみが示されています。他の 2 つのプロデューサ（P4 および P5）は、メッセージをエンキューする権利はあるが、図に示された時点ではメッセージをエンキューしていないと仮定しています。図では、次のことを示しています。

- 1 つのプロデューサが 1 つ以上のメッセージをエンキューできます。

- プロデューサは、メッセージをどの順序でもエンキューできます。

コンシューマ

この図では、メッセージのコンシューマが 3 つあり、コンシューマはこれがすべてです。図では、次のことを示しています。

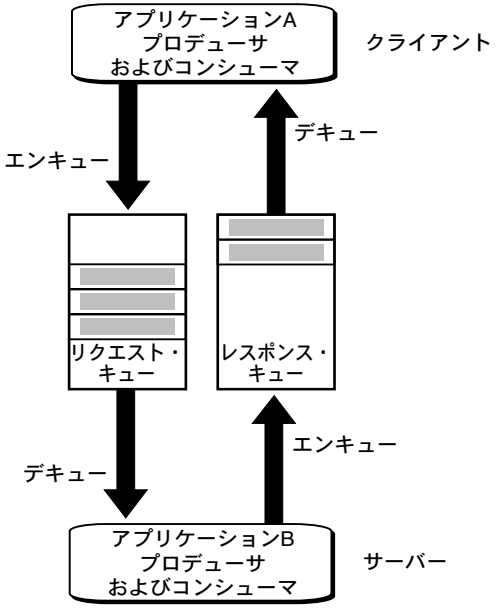
- メッセージは、エンキューされた順序でデキューする必要はありません。
- メッセージは、デキューされなくてもエンキューできます。

AQ を使用したクライアント / サーバー通信

前述の図では、一連のプロデューサによって複数のメッセージがエンキューされ、一連のコンシューマによってメッセージがデキューされることを示しました。この図で明らかにされていないのは、時間の概念および Oracle AQ を使用するメリットです。

通常、クライアント / サーバー・アプリケーションは同期方式で実行されるため、前述した密結合方式のすべてのデメリットが該当します。図 7-3 に、AQ を使用した非同期の代替案を示します。この例では、アプリケーション B（サーバー）は、リクエスト / レスポンス・キューを使用して、アプリケーション A（クライアント）にサービスを提供しています。

図 7-3 AQ を使用したクライアント / サーバー通信



1. アプリケーション A は、リクエストをリクエスト・キューにエンキューします。
2. アプリケーション B は、リクエストをデキューします。
3. アプリケーション B は、リクエストを処理します。
4. アプリケーション B は、その結果をレスポンス・キューにエンキューします。
5. アプリケーション A は、その結果をレスポンス・キューからデキューします。

このように、クライアントはサーバーに対するコネクションの確立を待つ必要はなく、サーバーは独自の処理タイミングでメッセージをデキューします。サーバーによるメッセージ処理が終了したとき、クライアントは結果を受け取るまで待つ必要はありません。このような二重遅延処理によって、クライアントおよびサーバーの両方が自由に処理できるようになります。

注意： 様々なエンキューおよびデキューの操作は、異なるトランザクションの一部です。

複数のコンシューマによる同一メッセージのデキュー

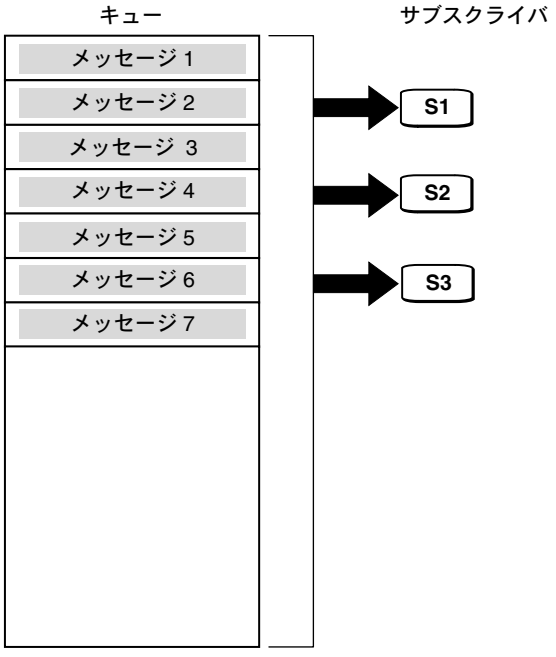
メッセージは、一度に 1 つのキューにしか入れることができません。複数のコンシューマに届けるために、プロデューサから同じメッセージを複数のキューに挿入する必要がある場合には、非常に多くのキューを管理する必要があります。Oracle AQ では、複数のコンシューマから同じメッセージをデキューできるようにするため、キューのサブスクライバおよびメッセージの受信者という 2 つのメカニズムを提供しています。これを利用する場合、キューを常駐させるキュー表は、サブスクライバ・リストおよび受信者リストの余裕を取るために、複数のコンシューマ・オプションを使用して作成する必要があります。各メッセージは、所定のすべてのコンシューマによって処理されるまで、キュー内にとどまります。

キューのサブスクライバ　この方法を使用することで、複数のコンシューマ・サブスクライバを 1 つのキューに対応付けることができます。これによって、キューにエンキューされたすべてのメッセージを、それぞれのキューのサブスクライバから使用できるようになります。キューに対応するサブスクライバは、メッセージまたはメッセージ・プロデューサを変更することなく、動的に変更できます。Oracle AQ 管理パッケージを使用することで、キューに対応するサブスクライバを追加および削除できます。図 7-4 に、メッセージが複数のプロデューサによってキューにエンキューされ、それぞれのメッセージが複数のコンシューマ・サブスクライバによって処理される過程を示します。

メッセージの受信者　メッセージ・プロデューサは、メッセージがエンキューされた時点で受信者リストを送ることができます。これによって、キュー内の各メッセージに、受信者の一意の集合を対応付けることができます。メッセージに対応付けられた受信者リストは、キューに対応付けられたサブスクライバ・リストが存在する場合には、それをオーバーライドします。受信者は、サブスクライバ・リストに含まれている必要はありません。ただし、受信者はサブスクライバの中から選択できます。

図 7-4 では、3 つすべてのコンシューマがキューのサブスクライバとしてリストされている場合を示しています。これは、3 つのコンシューマが、そのキューにエンキューされるすべてのメッセージをサブスクライブしているということです。

図 7-4 マルチ・コンシューマによる同一メッセージのデキュー



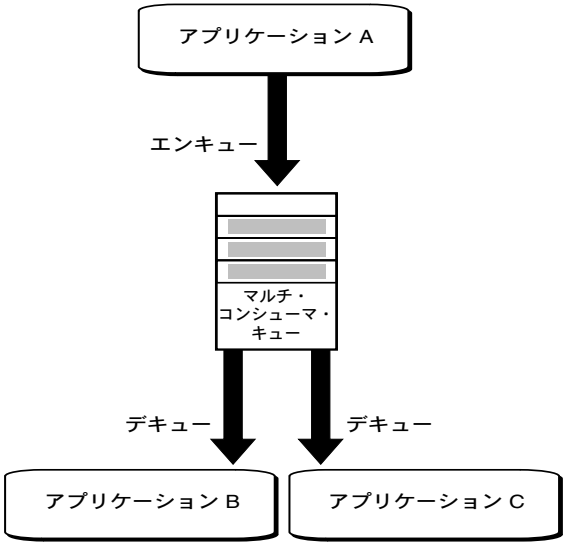
図には、いくつかの重要な点が示されています。

- 3 つのコンシューマが、すでにエンキューされている 7 つのメッセージに対するサブスクライバであり、まだエンキューされていないメッセージのサブスクライバになる可能性があります。
- どのメッセージも、最終的にそれぞれのサブスクライバによってデキューされます。
- サブスクライバの間に優先順位はありません。これは、どのサブスクライバがどのメッセージをどのような順番でデキューするかはわからないということです。サブスクライバによるデキュー順序は決まっていません。

- この図からは、メッセージがすでにデキューされたのか、そしてキューから削除されたのかについては判断できません。

図 7-5 では、同じテクノロジを動的な観点から示しています。この例では、あるアプリケーションの作成結果が複数のアプリケーションで必要になるシナリオを示しています。アプリケーション A によってエンキューされたすべてのメッセージは、アプリケーション B およびアプリケーション C によってデキューされます。これは、アプリケーション B および C をキュー・サブスクライバとするマルチ・コンシューマ・キューが、特別に設定されて可能になります。結果的に、これらのアプリケーションが、暗黙的にキューに格納されるすべてのメッセージの受信者になります。

図 7-5 マルチ・コンシューマ・キューを使用した通信

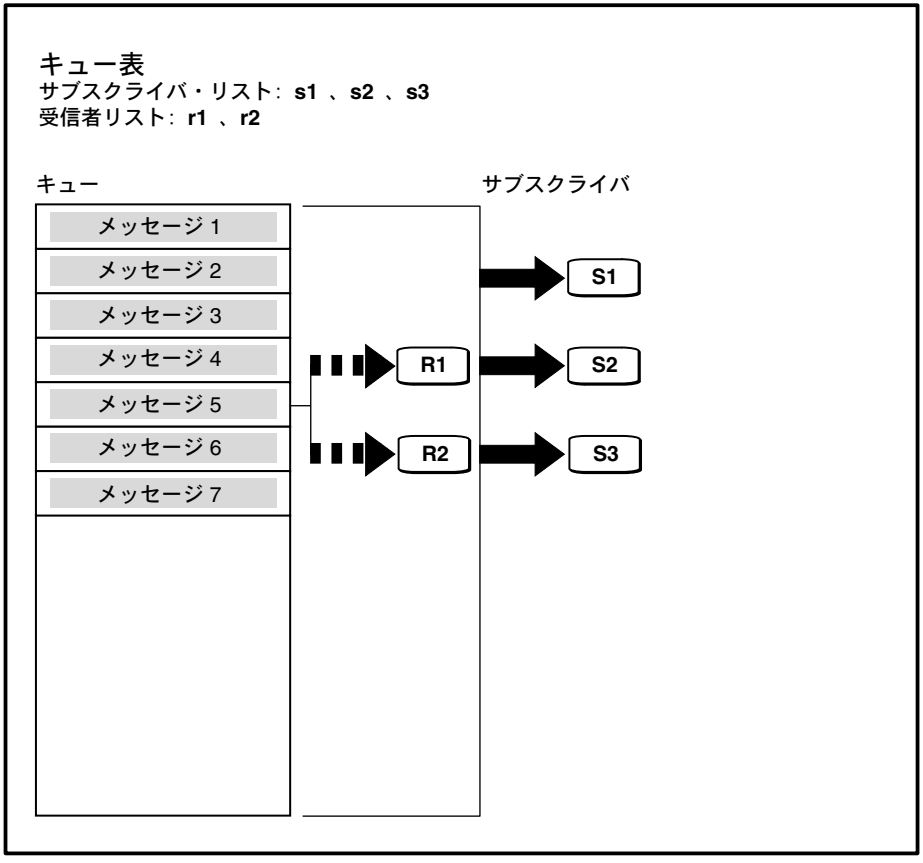


注意： アプリケーションまたは他のキューが、キュー・サブスクライバになることができます。

指定された受信者による指定されたメッセージのデキュー

図 7-6 に、メッセージを 1 人以上の受信者に指定する方法を示します。この場合、メッセージ 5 は、受信者 1 および受信者 2 によってデキューされるように指定されています。いずれの受信者も、キューに対するサブスクライバではありません。

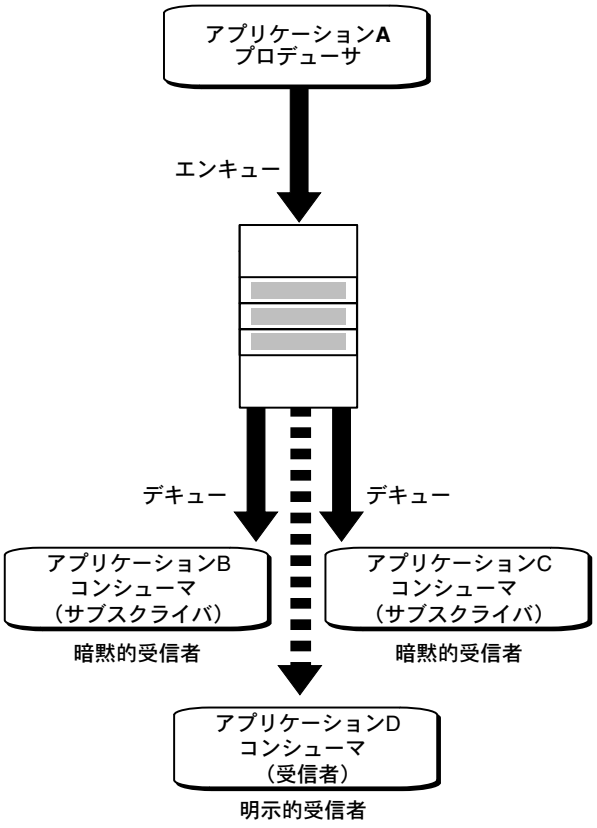
図 7-6 指定された受信者による指定されたメッセージのデキュー



前述のように、サブスクライバは、特定のキューに格納されたすべてのメッセージをデキューできるという点から、暗黙的な受信者といわれます。これは、雑誌の購読契約をすることで暗黙的にすべての記事にアクセスできることと似ています。受信者といわれるコンシューマのカテゴリは、特定のメッセージのターゲットに指定されるという点から、明示的な受信者といわれます。

図 7-7 に、Oracle AQ が動的にどのように調整され、両方の種類のコンシューマに対応するかを示します。この使用例では、アプリケーション B および C は、暗黙的な受信者（サブスクライバ）です。ただし、メッセージは、キューのサブスクライバであるかどうかにかかわらず、特定のコンシューマ（サブスクライバ）に明示的に転送することもできます。このような受信者リストは、そのメッセージのエンキュー・コールで指定され、そのキューのサブスクライバのリストをオーバーライドします。図では、アプリケーション D は、アプリケーション A によってエンキューされたメッセージの唯一の受信者に指定されています。

図 7-7 メッセージの明示的および暗黙的な受信者

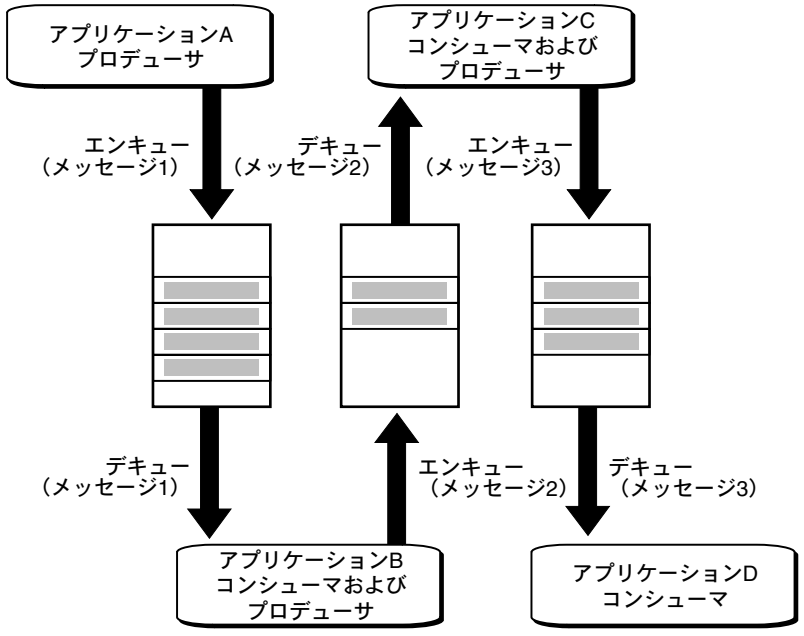


注意： 複数のプロデューサによって、異なる受信者に宛てられたメッセージが同時にエンキューされることがあります。

ワークフローの AQ 実装

図 7-8 では、ワークフロー（連鎖したアプリケーション・トランザクションともいう）を実装するために AQ が使用されています。ここでは、アプリケーション A、B、C および D によって実行されるワークフローのそれぞれの手順を示しています。キューは、各業務処理の各段階間での情報の流れをバッファするために使用されます。メッセージの遅延間隔および期限切れを指定することで、各アプリケーションに実行枠を設定できます。

図 7-8 AQ を使用したワークフローの実装



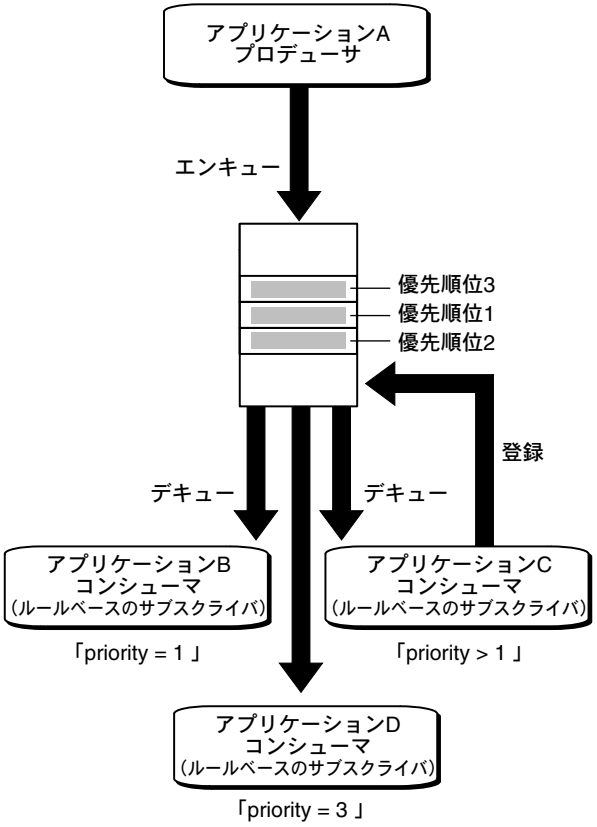
ワークフローの観点からみると、メッセージの受渡しは、ペイロード・データの価値以上に業務資産となります。そのため、AQ では、過去の傾向を分析して将来の動向を予測するためにメッセージの保存がオプションとしてサポートされています。

注意： メッセージ 1、2 および 3 の内容は同一の場合も異なる場合もあります。異なる場合でも、メッセージに前のメッセージの内容が一部含まれることがあります。

パブリッシュ・サブスクライブの AQ 実装

図 7-9 では、アプリケーション間のメッセージのパブリッシュ・サブスクライブ・スキームの実装のために AQ が使用されています。アプリケーション A は、メッセージをキューにパブリッシュするパブリッシャ・アプリケーションです。アプリケーション B、C および D は、サブスクライバ・アプリケーションです。アプリケーション A は、匿名でメッセージをキューにパブリッシュします。すると、それらのメッセージは、アプリケーションごとに指定されたルールに従ってサブスクライバ・アプリケーションに配信されます。サブスクライバ・アプリケーションは、メッセージ・プロパティおよびメッセージ・データの内容に対してルールを定義して、関心のあるメッセージを指定できます。

図 7-9 AQ を使用したパブリッシュ・サブスクライブの実装



この例では、アプリケーション B は「priority=1」というルールでサブスクライブし、アプリケーション C は「priority > 1」というルールでサブスクライブし、アプリケーション D は「priority = 3」というルールでサブスクライブしています。アプリケーション A は、3 個のメッセージ（優先順位はそれぞれ 3、1、2）をエンキューします。アプリケーション B は、1 個のメッセージ（優先順位が 1）を受信し、アプリケーション C は、2 個のメッセージ（優先順位が 2、3）を受信し、アプリケーション D は、1 個のメッセージ（優先順位が 3）を受信します。このように、メッセージ受信者はメッセージ・プロパティおよび内容に基づいて動的に計算されます。また、この図には、アプリケーション C がどのようにメッセージ配信の非同期式通知を使用するかも示されています。アプリケーション C はキューのメッセージに登録を行っています。メッセージが届くと、そのことが通知されるため、アプリケーション C はメッセージをデキューできます。

メッセージの伝播

メッセージのファンアウト

AQ では、メッセージの受信者はコンシューマまたは他のキューのどちらかです。メッセージの受信者がキューの場合には、実際の受信者は、そのキューに対するサブスクライバ（これが他のキューの場合もある）によって決定されます。これによって、全員が 1 つのキューからメッセージのデキューをしなくても、多くの受信者にメッセージをファンアウトできます。

たとえば、ソース・キューが、サブスクライバ・キューとして *dispatch1@dest1* および *dispatch2@dest2* を持っているとします。次にキュー *dispatch1@dest1* が、サブスクライバとしてキュー *outerreach1@dest3* および *outerreach2@dest4* を持ち、キュー *dispatch2@dest2* のサブスクライバとしてキュー *outerreach3@dest21* および *outerreach4@dest4* を持つことができます。このようにして、ソース・キューにエンキューされたメッセージは、4 つの異なるキューのサブスクライバに伝播します。

コンポジット（ファネリング）

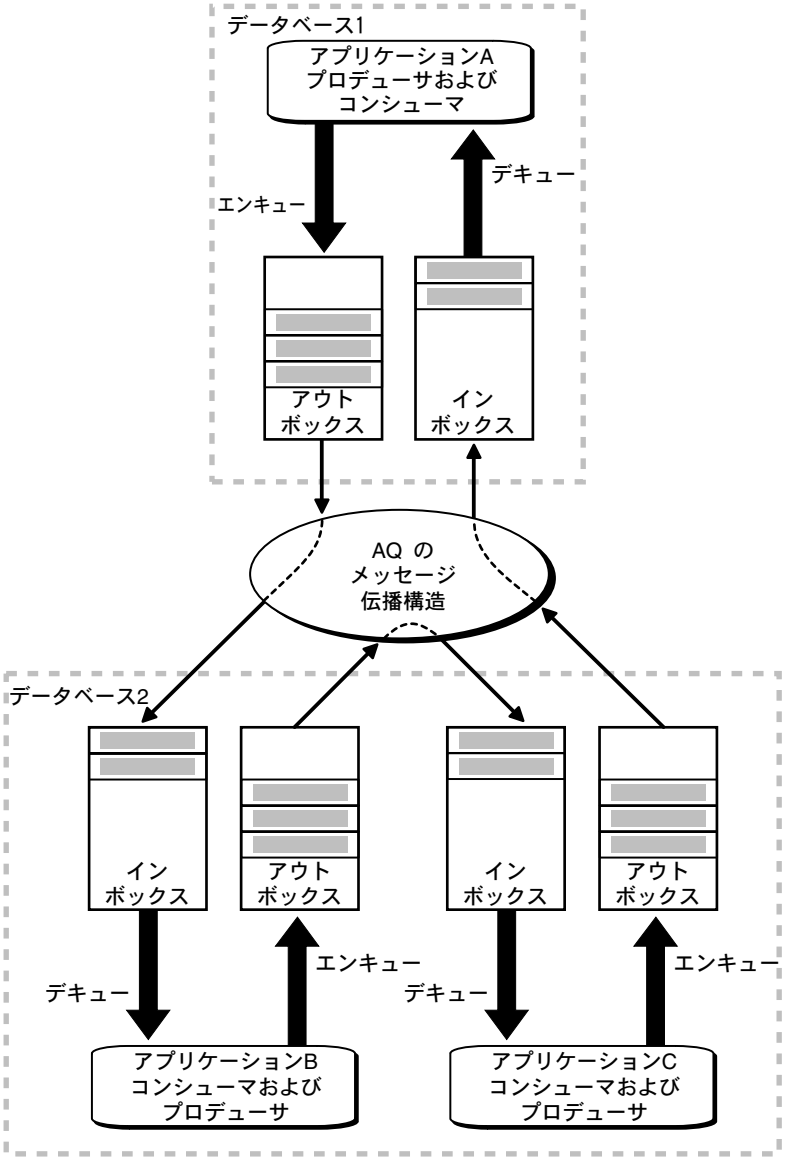
異なるキューのメッセージを単一のキューに結合することもできます。これは、コンポジットと呼ばれることもあります。たとえば、キュー *composite@endpoint* が *funnel1@source1* および *funnel2@source2* の両方に対するサブスクライバである場合、*composite@endpoint* に対応するサブスクライバは、そのキュー自身に直接エンキューされるメッセージのみでなく、残り 2 つのキューにエンキューされるすべてのメッセージを取得できます。

伝播およびアドバンスト・キューイング

図 7-10 に、異なるデータベース上のアプリケーションが AQ を使用して通信する過程を示します。各アプリケーションには、受信メッセージおよび送信メッセージを処理するためのインボックスおよびアウトボックスがあります。アプリケーションは、メッセージが送信されるアプリケーションがローカル（同一ノード上）かまたはリモート（異なるノード上）かにかかわらず、メッセージをアウトボックスにエンキューします。アプリケーションは、メッセージがローカルで作成されたか、またはリモートで作成されたかにかかわらず、その

インボックスにあるメッセージをデキューします。AQ では、メッセージが同じ基準で扱われるため、すべての交換を簡単に実行できます。

図 7-10 メッセージの伝播



8

AQ を使用したサンプル・アプリケーション

第 1 章「Oracle Advanced Queuing の概要」では、架空の会社 BooksOnLine のメッセージ・システムについて説明しました。この章では、BooksOnLine のサンプル・アプリケーションにおける AQ の機能について説明します。内容は次のとおりです。

- サンプル・アプリケーション
- アドバンスト・キューイングの一般機能
- エンキュー機能
- デキュー機能
- 非同期通知
- 伝播機能

サンプル・アプリケーション

大型の書籍販売店である BooksOnLine の運営は、販売にかかわる様々な部署にまたがるアクティビティを自動化する、オンライン書籍発注システムが基礎となっています。システムのフロント・エンドは、新しい注文を入力するための注文入力アプリケーションです。入力された注文は、注文処理アプリケーションによって妥当性チェックおよび記録が行われます。地域倉庫に置かれた出荷部門は、発注された書籍を時間どおりに発送する責任があります。地域倉庫は 3 箇所で、それぞれが東部地域、西部地域、そして海外からの発注に対応しています。発注された書籍の発送完了後、発注情報は支払処理のために中央の請求部門に送られます。各地の顧客サービス部門は、発注情報の状況を管理し、発注に関する問合せを処理する責任があります。

BooksOnLine のシナリオでは、AQ の機能の例を示し、AQ テクノロジの可能性を実証します。サンプル・コードのスクリプトについては、付録 C「BooksOnLine 用スクリプト」を参照してください。

アドバンスト・キューイングの一般機能

この項の内容は次のとおりです。

- システム・レベルのアクセス制御
- キュー・レベルのアクセス制御
- メッセージ・フォーマットの変換
- 構造化ペイロード
- XMLType キューのペイロード
- 非永続キュー
- 保存およびメッセージ履歴
- パブリッシュ・サブスクライブ・サポート
- Oracle Real Application Clusters のサポート
- 統計ビューのサポート

システム・レベルのアクセス制御

Oracle は、すべてのキューイング操作に対してシステム・レベルのアクセス制御をサポートします。この機能によって、アプリケーション設計者または DBA は、ユーザーをキュー管理者に指定できます。キュー管理者は、データベースのどのキューに対しても、AQ インタフェース（管理および操作）を起動できます。これによって、データベース上のキュー全体に対するすべての管理スクリプトを 1 つのスキーマで管理できるため、管理作業が簡単になります。詳細は、4-8 ページの「Oracle Enterprise Manager のサポート」を参照してください。

PL/SQL（DBMS_AQADM）：シナリオおよびコード

BooksOnLine アプリケーションでは、DBA がデータベースのキュー管理者として BOLADM（BooksOnLine の管理者アカウント）を作成します。これによって、BOLADM はデータベース上のキューを作成、削除、管理および監視できるようになります。アプリケーションからエンキューおよびデキューするために BOLADM スキーマに PL/SQL パッケージが必要な場合、DBA は ENQUEUE_ANY および DEQUEUE_ANY システム権限を、BOLADM に付与する必要があります。

```
CREATE USER BOLADM IDENTIFIED BY BOLADM;
GRANT CONNECT, RESOURCE, aq_administrator_role TO BOLADM;
GRANT EXECUTE ON dbms_aq TO BOLADM;
GRANT EXECUTE ON dbms_aqadm TO BOLADM;
EXECUTE dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','BOLADM',FALSE);
EXECUTE dbms_aqadm.grant_system_privilege('DEQUEUE_ANY','BOLADM',FALSE);
```

Java AQ API を使用する場合、BOLADM には DBMS_AQIN パッケージの実行権限が必要です。

```
GRANT EXECUTE ON DBMS_AQIN to BOLADM;
```

このアプリケーションでは、AQ のプロパゲータは注文入力（OE）スキーマから西部売上（WS）、東部売上（ES）および海外売上（OS）スキーマにメッセージを移入します。次に、WS、ES および OS スキーマから顧客請求（CB）および顧客サービス（CS）スキーマにメッセージを移入します。したがって、OE、WS、ES および OS スキーマのすべてに、プロパゲータのソース・キューとなるキューがあります。

メッセージが宛先キューに届くと、ソース・キューのスキーマ名に基づいたセッションによって、新しく届いたメッセージが宛先キューにエンキューされます。つまり、ソース・キューのスキーマに、宛先キューに対するエンキュー権限を付与する必要があります。

管理を単純化するために、BooksOnLine アプリケーションでソース・キューを持つすべてのスキーマに ENQUEUE_ANY システム権限を付与します。

```
EXECUTE dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','OE',FALSE);
EXECUTE dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','WS',FALSE);
EXECUTE dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','ES',FALSE);
EXECUTE dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','OS',FALSE);
```

リモートの宛先キューに伝播するために、エージェント構造体のアドレス・フィールドのデータベース・リンクに指定されたログイン・ユーザーには、ENQUEUE ANY QUEUE 権限を付与するか、または宛先キューに対するエンキュー権限を付与する必要があります。データベース・リンクのログイン・ユーザーが宛先のキュー表を所有している場合は、どのような明示的な権限も付与する必要はありません。

Visual Basic（OO4O）：サンプル・コード

この機能については、データベースの dbexecutesql インタフェースを使用します。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

キュー・レベルのアクセス制御

Oracle は、エンキューおよびデキュー操作に対してキュー・レベルのアクセス制御をサポートします。この機能によって、アプリケーション設計者は、あるスキーマに作成されたキューを、他のスキーマで実行中のアプリケーションから保護することができます。アプリケーション設計者は、キューのスキーマの外で実行中のアプリケーションに対して、最小限のアクセス権限のみを付与する必要があります。キューに対するアクセス権限でサポートされているのは、ENQUEUE、DEQUEUE および ALL です。詳細は、4-8 ページの「Oracle Enterprise Manager のサポート」を参照してください。

シナリオ

BooksOnLine アプリケーションでは、顧客請求情報を CB および CBADM スキーマで処理します。CB スキーマには顧客請求情報アプリケーションがあり、CBADM スキーマには、関連する請求情報データがキュー表の形で格納されています。

請求情報データを保護するために、請求処理アプリケーションおよび請求情報データは別のスキーマに常駐しています。請求処理アプリケーションは、出荷済の注文情報キューである CBADM_shippedorders_que からメッセージをデキューする以外の処理はできません。そこでメッセージを処理し、請求済の注文情報キューである CBADM_billedorders_que に新しいメッセージをエンキューします。

他のアプリケーションの不正な操作からキューを保護するために、次の 2 つの grant コールを行う必要があります。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/* Grant dequeue privilege on the shopped orders queue to the Customer
   Billing application. The CB application retrieves orders that are shipped but
   not billed from the shipped orders queue. */
EXECUTE dbms_aqadm.grant_queue_privilege(
   'DEQUEUE','CBADM_shippedorders_que', 'CB', FALSE);

/* Grant enqueue privilege on the billed orders queue to Customer Billing
   application. The CB application is allowed to put billed orders into this
   queue after processing the orders. */

EXECUTE dbms_aqadm.grant_queue_privilege(
   'ENQUEUE', 'CBADM_billedorders_que', 'CB', FALSE);
```

Visual Basic（OO4O）：サンプル・コード

この機能については、データベースの dbexecutesql インタフェースを使用します。

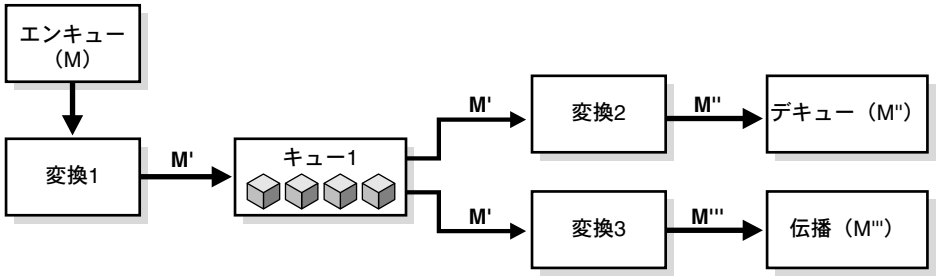
Java（JDBC）：サンプル・コード

```
public static void grantQueuePrivileges(Connection db_conn)
{
    AQSession  aq_sess;
    AQQueue    sh_queue;
    AQQueue    bi_queue;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        /* Grant dequeue privilege on the shipped orders queue to the Customer
           Billing application. The CB application retrieves orders that are
           shipped but not billed from the shipped orders queue. */

        sh_queue = aq_sess.getQueue("CBADM", "CBADM_shippedorders_que");

        sh_queue.grantQueuePrivilege("DEQUEUE", "CB", false);

        /* Grant enqueue privilege on the billed orders queue to Customer
           Billing application.The CB application is allowed to put billed
           orders into this queue after processing the orders. */

        bi_queue = aq_sess.getQueue("CBADM", "CBADM_billedorders_que");

        bi_queue.grantQueuePrivilege("ENQUEUE", "CB", false);
    }
    catch (AQException ex)
    {
        System.out.println("AQ Exception: " + ex);
    }
}
```

メッセージ・フォーマットの変換

異なるメッセージ・ペイロード型間の変換マッピングを定義できます。変換マッピングは、PL/SQL ファンクション（コールアウトを含む）および Java ストアド・プロシージャを含めることが可能な SQL 式として定義されます。1 対 1 のメッセージ変換のみがサポートされます。変換エンジンがアドバンスト・キューイングと緊密に統合されているため、データベース・メッセージ・システム内で移動するメッセージを簡単に変換できます。図 8-1 に、変換とアドバンスト・キューイングとの統合状態を示します。

図 8-1 アドバンスト・キューイングに統合された変換



変換1 (m, m')
変換2 (m', m'')
変換3 (m', m'')
M、M′、M′′およびM′′′は、それぞれm型、
m′型、m′′型およびm′′′型のメッセージ

変換マッピングは、エンキュー、デキューおよび伝播操作中に使用できます。エンキューで変換を使用するには、マッピングをエンキュー・オプションで指定します。デキューで変換を使用するには、マッピングをデキュー・オプションで指定するか、またはサブスクライバの追加時に指定します。デキュー・オプションで指定されたマッピングは、ADD_SUBSCRIBER で指定されたマッピングより優先されます。伝播で変換を使用するには、マッピングをサブスクライバの追加時に指定します。

PL/SQL（DBMS_TRANSFORM）：シナリオおよびコード

BooksOnLine アプリケーションで、注文タイプが、注文入力アプリケーションと出荷処理アプリケーションで異なって表現されると想定します。

注文入力アプリケーション（スキーマ OE 内）の注文タイプは、次のとおりです。

```
create or replace type order_typ as object (
        orderno         number,
        status          varchar2(30),
        ordertype       varchar2(30),
        orderregion     varchar2(30),
        custno          number,
        paymentmethod   varchar2(30),
        items           orderitemlist_vartyp,
        ccnumber        varchar2(20),
        order_date      date);

create or replace type customer_typ as object (
        custno          number,
        custid          varchar2(20),
        name            varchar2(100),
```

```
        street          varchar2(100),
        city            varchar2(30),
        state           varchar2(2),
        zip             number,
        country         varchar2(100));

create or replace type book_typ as object (
        title           varchar2(100),
        authors         varchar2(100),
        ISBN            varchar2(20),
        price           number);

create or replace type orderitem_typ as object (
        quantity        number,
        item            book_typ,
        subtotal        number);

create or replace type orderitemlist_vartyp as varray (20) of
orderitem_typ;
```

出荷処理アプリケーションの注文項目は、次のとおり定義されます。

```
create or replace type order_typ_sh as object (
        orderno         number,
        status          varchar2(30),
        ordertype       varchar2(30),
        orderregion     varchar2(30),
        customer        customer_typ_sh,
        paymentmethod   varchar2(30),
        items           orderitemlist_vartyp,
        ccnumber        varchar2(20),
        order_date      date);

create or replace type customer_typ_sh as object (
        custno          number,
        name            varchar2(100),
        street          varchar2(100),
        city            varchar2(30),
        state           varchar2(2),
        zip             number);

create or replace type book_typ_sh as object (
        title           varchar2(100),
        authors         varchar2(100),
        ISBN            varchar2(20),
        price           number);
```

```
create or replace type orderitem_typ_sh as object (
        quantity        number,
        item            book_typ,
        subtotal        number);

create or replace type orderitemlist_vartyp_sh as varray (20) of
orderitem_typ_sh;
```

海外向け出荷アプリケーションでは、sys.XMLType 属性が使用されます。

変換の作成

変換は、次の方法で作成できます。

- ターゲット型のオブジェクトを戻す単一の PL/SQL ファンクションまたはターゲット型のコンストラクタを作成します。

  この表現は、単純な変換、または属性ごとに独立した変換を作成するのが困難な変換に適しています。

  ```
  execute dbms_transform.create_transformation(
            schema => 'OE', name => 'OE2WS',
            from_schema => 'OE', from_type => 'order_typ',
            to_schema => 'WS', to_type => 'order_typ_sh',
            transformation(
                'WS.order_typ_sh(source.user_data.orderno,
                                 source.user_data.status,
                                 source.user_data.ordertype,
                                 source.user_data.orderregion,

  WS.get_customer_info(source.user_data.custno),
                                 source.user_data.paymentmethod,
                                 source.user_data.items,
                                 source.user_data.ccnumber,
                                 source.user_data.order_date)');
  ```

  BooksOnLine アプリケーションの海外向け出荷サイトでは、注文が、XMLType ペイロードとして表現されると想定します。注文入力サイトでは、注文が Oracle オブジェクト ORDER_TYP として表現されます。海外向け出荷サイトでは、OE_BOOKEDORDERS_QUE キューのメッセージがサブスクライブされるため、メッセージが注文入力サイトから海外向け出荷サイトに伝播される前に、変換が適用されます。

変換は、次のとおり定義されます。

```
CREATE OR REPLACE FUNCTION CONVERT_TO_ORDER_XML(input_order TYPE OE.ORDER_TYP)
RETURN SYS.XMLType AS
   new_order SYS.XMLType;
BEGIN
   select SYS_XMLGEN(input_order) into new_order from dual;
   RETURN new_order;
END CONVERT_TO_ORDER_XML;

execute dbms_transform.create_transformation(
   schema =>         'OS',
   name   =>         'OE2XML',
   from_schema =>    'OE',
   from_type =>      'ORDER_TYP',
   to_schema =>      'SYS',
   to_type =>        'XMLTYPE',
   transformation => 'CONVERT_TO_ORDER_XML(source.user_data)');

/*  Add a rule-based subscriber for Overseas Shipping to the Booked orders
queues with Transformation. Overseas Shipping handles all non-US orders: */
DECLARE
 subscriber     aq$_agent;
BEGIN
 subscriber := aq$_agent('Overseas_Shipping','OS.OS_bookedorders_que',null);

 dbms_aqadm.add_subscriber(
        queue_name     => 'OE.OE_bookedorders_que',
        subscriber     => subscriber,
        rule           => 'tab.user_data.orderregion = ''INTERNATIONAL'''
        transformation => 'OS.OE2XML');
END;
```

- ターゲット型の属性ごとに別々の式を作成します。この表現によって、宛先タイプの各属性に対して変換マッピングを簡単に作成および管理できるようになります。これは、宛先タイプに多数の属性がある場合に有効です。

```
/* first create the transformation without any transformation expression*/
execute dbms_transform.create_transformation(
          schema => 'OE', name => 'OE2WS',
          from_schema => 'OE', from_type => 'order_typ',
          to_schema => 'WS', to_type => 'order_typ_sh');

/* specify each attribute of the target type as a function of the source type*/
execute dbms_transform.modify_transformation(
          schema => 'OE', name => 'OE2WS',
          attribute_number => 1,
          transformation => 'source.user_data.orderno');
```

```
execute dbms_transform.modify_transformation(
           schema => 'OE', name => 'OE2WS',
           attribute_number => 1,
           transformation => 'source.user_data.status');

execute dbms_transform.modify_transformation(
           schema => 'OE', name => 'OE2WS',
           attribute_number => 1,
           transformation => 'source.user_data.ordertype');

execute dbms_transform.modify_transformation(
           schema => 'OE', name => 'OE2WS',
           attribute_number => 1,
           transformation => 'source.user_data.orderregion');

execute dbms_transform.modify_transformation(
           schema => 'OE', name => 'OE2WS',
           attribute_number => 1,
           transformation =>
'WS.get_customer_info(source.user_data.custno)');

execute dbms_transform.modify_transformation(
           schema => 'OE', name => 'OE2WS',
           attribute_number => 1,
           transformation => 'source.user_data.payment_method');

execute dbms_transform.modify_transformation(
           schema => 'OE', name => 'OE2WS',
           attribute_number => 1,
           transformation => 'source.user_data.orderitemlist_vartyp');

execute dbms_transform.modify_transformation(
           schema => 'OE', name => 'OE2WS',
           attribute_number => 1,
           transformation => 'source.user_data.ccnumber');

execute dbms_transform.modify_transformation(
           schema => 'OE', name => 'OE2WS',
           attribute_number => 1,
           transformation => 'source.user_data.order_date');
```

Visual Basic（OO4O）：サンプル・コード

今回のリリースでは、例は記載していません。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

構造化ペイロード

Oracle AQ では、メッセージのペイロードを構造化して管理するためにオブジェクト型を使用できます。Oracle のオブジェクト・リレーショナル機能は、以前からのリレーショナル・データ型からユーザー定義型にいたるまでの豊富なデータ型を備えています。

強力な型指定を持つ内容（Oracle オブジェクト型指定システムによって定義されるフォーマットを持つ内容）によって、次の機能が使用できます。

- 内容ベースのルーティング : アドバンスト・キューイングで内容を調査し、その内容に基づいて自動的にメッセージを別のキューにルーティングできます。
- 内容ベースのサブスクリプション : パブリッシュ・サブスクライブ・システムをメッセージ・システム上に組み込んで、内容に基づいてサブスクリプションを作成できます。
- XML: AQ メッセージで XML の柔軟性および拡張性を使用します。XMLType の持ついくつかのメソッドによって XML データを簡単に使用できます。このメソッドには XMLType.existsNode( ) および XMLType.extract( ) が含まれます。

また、XMLType 属性を持つ Oracle オブジェクトを含むペイロード型の作成もできます。これらのペイロードは、XML 文書を含むメッセージの転送および格納に使用できます。XMLType 属性を持つ Oracle オブジェクトを定義すると、次のことができます。

- 複数のスキーマを持つ XML 文書を同じキュー内に格納します。このドキュメントは、CLOB として内部的に格納されます。
- XMLType.existsNode()、XMLType.extract() などのメソッドを使用して、ペイロード内の XMLType 属性を問い合せます。

PL/SQL（DBMS_AQADM）：シナリオおよびコード

BooksOnLine アプリケーションでは、書籍注文データをメッセージ内容としてモデル化するために豊富なデータ型セットが利用されます。

- 顧客は、customer_typ というオブジェクト型にモデル化されます。

```
CREATE OR REPLACE TYPE customer_typ AS OBJECT (
        custno          NUMBER,
        name            VARCHAR2(100),
        street          VARCHAR2(100),
        city            VARCHAR2(30),
        state           VARCHAR2(2),
        zip             NUMBER,
        country         VARCHAR2(100));
```

- 書籍は、book_typ というオブジェクト型にモデル化されます。

```
CREATE OR REPLACE TYPE book_typ AS OBJECT (
        title           VARCHAR2(100),
        authors         VARCHAR2(100),
        ISBN            NUMBER,
        price           NUMBER);
```

- 注文明細項目を表す注文項目は、orderitem_typ というオブジェクト型にモデル化されます。注文項目は、書籍型を含むネストされた型になります。

```
CREATE OR REPLACE TYPE orderitem_typ AS OBJECT (
        quantity        NUMBER,
        item            BOOK_TYP,
        subtotal        NUMBER);
```

- 注文項目リストは、注文明細項目のリストとして使用され、注文項目の VARRAY としてモデル化されます。

```
CREATE OR REPLACE TYPE orderitemlist_vartyp AS VARRAY (20) OF orderitem_typ;
```

- 注文は、order_typ というオブジェクト型にモデル化されます。これはコンポジット型で、ネストされたオブジェクト型（定義済）を含んでいます。この型は注文、顧客情報および注文項目リストの詳細を格納します。

```
CREATE OR REPLACE TYPE order_typ as object (
        orderno         NUMBER,
        status          VARCHAR2(30),
        ordertype       VARCHAR2(30),
        orderregion     VARCHAR2(30),
        customer        CUSTOMER_TYP,
        paymentmethod   VARCHAR2(30),
        items           ORDERITEMLIST_VARTYP,
        total           NUMBER);
```

- BooksOnLine アプリケーションには、SYS.XMLType ペイロードを使用して注文をモデル化するキューもあります。

Visual Basic（OO4O）：サンプル・コード

この機能については、データベースの dbexecutesql インタフェースを使用します。

Java（JDBC）：サンプル・コード

型を作成した後、JPublisher を使用して、SQL 型にマップする Java クラスを生成します。

1. 次のように入力して、JPublisher に対する入力ファイル `jaqbol.typ` を作成します。

```
TYPE boladm.customer_typ as Customer
TYPE boladm.book_typ as Book
TYPE boladm.orderitem_typ AS OrderItem
TYPE boladm.orderitemlist_vartyp AS OrderItemList
TYPE boladm.order_typ AS Order
```

2. 次の引数を使用して、JPublisher を実行します。

```
jpub -input=jaqbol.typ -user=boladm/boladm -case=mixed -methods=false
-compatible=CustomDatum
```

これによって、前述の項で作成された SQL オブジェクト型にマップする Java クラス（`Customer`、`Book`、`OrderItem` および `OrderItemList`）が作成されます。

3. Java AQ ドライバをロードし、JDBC コネクションを作成します。

```
public static Connection loadDriver(String user, String passwd)
{
   Connection db_conn = null;
   try
   {
         Class.forName("oracle.jdbc.driver.OracleDriver");

      /* your actual hostname, port number, and SID will
      vary from what follows. Here we use 'dlsun736,' '5521,'
      and 'test,' respectively: */

      db_conn =
               DriverManager.getConnection(
               "jdbc:oracle:thin:@dlsun736:5521:test",
               user, passwd);

      System.out.println("JDBC Connection opened ");
      db_conn.setAutoCommit(false);

      /* Load the Oracle8i AQ driver: */
      Class.forName("oracle.AQ.AQOracleDriver");

      System.out.println("Successfully loaded AQ driver ");
   }
   catch (Exception ex)
   {
      System.out.println("Exception: " + ex);
```

```
      ex.printStackTrace();
    }
    return db_conn;
```

XMLType キューのペイロード

XMLType ペイロードを持つキューを作成できます。これらのペイロードは、XML 文書を含むメッセージの転送および格納に使用できます。XMLType 属性を持つ Oracle オブジェクトを定義すると、次のことができます。

- 複数のスキーマを持つ XML 文書を同じキュー内に格納します。このドキュメントは、CLOB として内部的に格納されます。
- XMLType.existsNode()、XMLType.extract() などのメソッドを使用して、ペイロード内の XMLType 属性を持つメッセージを選択的にデキューします。

  参照： XMLType 演算子の詳細は、『Oracle9*i* XML データベース開発者ガイド -Oracle XML DB』を参照してください。

- 変換を定義して、Oracle オブジェクトを XMLType に変換します。
- XMLType.existsNode() や XMLType.extract() などの XMLType メソッドを使用して、メッセージ内容を問い合せるルールベースのサブスクライバを定義します。

BooksOnLine アプリケーションの海外向け出荷サイトでは、注文が、SYS.XMLType として表現されると想定します。注文入力サイトでは、注文が Oracle オブジェクト ORDER_TYP として表現されます。

海外向けキュー表およびキューは、次のとおり作成されます。

```
BEGIN
dbms_aqadm.create_queue_table(
   queue_table        => 'OS_orders_pr_mqtab',
   comment            => 'Overseas Shipping MultiConsumer Orders queue table',
   multiple_consumers => TRUE,
   queue_payload_type => 'SYS.XMLTtype',
   compatible         => '8.1');
END;

BEGIN
dbms_aqadm.create_queue (
   queue_name   => 'OS_bookedorders_que',
   queue_table  => 'OS_orders_pr_mqtab');
END;
```

海外向け出荷サイトでの注文の表現は、注文入力サイトでの注文の表現と異なるため、メッセージが注文入力サイトから海外向け出荷サイトに伝播される前に、変換が適用されます。

```
/*  Add a rule-based subscriber (for Overseas Shipping) to the Booked orders queues
with Transformation. Overseas Shipping handles all non-US orders: */
DECLARE
  subscriber     aq$_agent;
BEGIN
  subscriber := aq$_agent('Overseas_Shipping','OS.OS_bookedorders_que',null);

  dbms_aqadm.add_subscriber(
   queue_name     => 'OE.OE_bookedorders_que',
   subscriber     => subscriber,
   rule           => 'tab.user_data.orderregion = ''INTERNATIONAL''',
   transformation => 'OS.OE2XML');
END;
```

注文入力アプリケーションで使用される型から、海外向け出荷アプリケーションで使用される型への変換の定義の詳細は、8-8 ページの「変換の作成」を参照してください。

アプリケーションが、カナダの顧客の注文を処理すると想定します。このアプリケーションは、次のプロシージャを使用してメッセージをデキューできます。

```
/*  Create procedures to enqueue into single-consumer queues: */
create or replace procedure get_canada_orders() as
deq_msgid                RAW(16);
dopt                     dbms_aq.dequeue_options_t;
mprop                    dbms_aq.message_properties_t;
deq_order_data           SYS.XMLTtype;
no_messages              exception;
pragma exception_init    (no_messages, -25228);
new_orders               BOOLEAN := TRUE;

begin
        dopt.wait := 1;

/* Specify dequeue condition to select Orders for Canada */
        dopt.deq_condition := 'tab.user_data.extract(
''/ORDER_TYP/CUSTOMER/COUNTRY/text()'').getStringVal()=''CANADA''';

            dopt.consumer_name : = 'Overseas_Shipping';

        WHILE (new_orders) LOOP
          BEGIN
            dbms_aq.dequeue(
                queue_name         => 'OS.OS_bookedorders_que',
                dequeue_options    => dopt,
                message_properties => mprop,
```

```
            payload             => deq_order_data,
            msgid               => deq_msgid);
        commit;

        dbms_output.put_line(' Order for Canada - Order: ' ||
                             deq_order_data.getStringVal());

      EXCEPTION
        WHEN no_messages THEN
             dbms_output.put_line (' ---- NO MORE ORDERS  ---- ');
             new_orders := FALSE;
      END;
    END LOOP;
end;
```

非永続キュー

非永続キューにあるメッセージは、データベース表には格納されません。ユーザーは、シングル・コンシューマ・タイプまたはマルチ・コンシューマ・タイプの非永続キューを作成できます。このキューは、create_np_queue コマンドで指定されたスキーマ内の、システムによって作成されたキュー表（シングル・コンシューマ・キューは AQ$_MEM_SC、マルチ・コンシューマ・キューは AQ$_MEM_MC）内に作成されます。マルチ・コンシューマ・キューには、サブスクライバを追加できます（9-27 ページの「非永続キューの作成」を参照）。非永続キューは、伝播の宛先になることができます。

ユーザーは、エンキュー・インタフェースを使用して、通常の方法で非永続キューにメッセージをエンキューします。非永続キュー内部に、RAW 型およびオブジェクト型（ユーザー定義型）メッセージをエンキューできます。関心があるキューに（LNOCISubcriptionRegister または DBMS_AQADM.REGISTER を使用して）通知登録を行い、非同期式通知メカニズムを使用して、非永続キューからメッセージを取り出します（11-55 ページの「通知の登録」を参照）。

あるキューにメッセージがエンキューされると、そのキューに対してアクティブな登録を持っているクライアントに配信されます。さらにそのメッセージは、データベースへの格納というオーバーヘッドなしで、関心を持っているクライアントにパブリッシュされます。

参照： DBMS_AQADM.REGISTER については、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』、LNOCISubscriptionRegister については、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

シナリオ

注文入力システムには、ユーザー・リクエストに対処するアプリケーション・プロセスが 3 つあると想定します。コネクション・ディスパッチャは、アプリケーション・プロセスからの接続要求を分配します。注文入力システムにログイン中のユーザー数、およびアプリケーション・プロセス当たりのユーザー数をメンテナンスします。アプリケーション・プロセスの名前は、APP_1、APP_2、APP_3 です（この例では、アプリケーション・プロセス障害は想定していません）。

非永続キューを使用することで、このシナリオでの要件を満たすことができます。ユーザーがデータベースにログインするときに、アプリケーション・プロセスはマルチ・コンシューマの非永続キューである LOGIN_LOGOUT にエンキューし、アプリケーション名をコンシューマ名にします。ユーザーがログアウトするときも、同じように処理されます。2 つのイベントを区別するために、ログイン・メッセージの相関識別子は「LOGIN」、ログアウト・メッセージの相関識別子は「LOGOUT」にします。

コールバック関数は、アプリケーション・プロセスごとにログイン / ログアウトのイベント件数を数えます。ディスパッチャ・プロセスがデータベースに接続する必要があるのは、サブスクリプションを登録する場合のみということに注意してください。通知そのものは、プロセスがデータベースから切断されている間にも受け取ることができます。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CONNECT oe/oe;
/* Create the Object Type/ADT adtmsg */
CREATE OR REPLACE TYPE adtmsg AS OBJECT (id NUMBER, data VARCHAR2(4000));

/* Create the multiconsumer nonpersistent queue in OE schema: */
EXECUTE dbms_aqadm.create_np_queue(queue_name         => 'LOGON_LOGOFF',
                                   multiple_consumers => TRUE);

/* Enable the queue for enqueue and dequeue: */
EXECUTE dbms_aqadm.start_queue(queue_name => 'LOGON_LOGOFF');

/* Nonpersistent Queue Scenario - procedure to be executed upon logon: */
CREATE OR REPLACE PROCEDURE  User_Logon(app_process IN VARCHAR2)
AS
  msgprop        dbms_aq.message_properties_t;
  enqopt         dbms_aq.enqueue_options_t;
  enq_msgid      RAW(16);
  payload        RAW(1);
BEGIN
  /* visibility must always be immediate for NonPersistent queues */
  enqopt.visibility:=dbms_aq.IMMEDIATE;
  msgprop.correlation:= 'LOGON';
  msgprop.recipient_list(0) := aq$_agent(app_process, NULL, NULL);
  /* payload is NULL */
  dbms_aq.enqueue(
```

```
        queue_name         => 'LOGON_LOGOFF',
        enqueue_options    => enqopt,
        message_properties => msgprop,
        payload            => payload,
        msgid              => enq_msgid);

END;

/* Nonpersistent queue scenario - procedure to be executed upon logoff: */
CREATE OR REPLACE PROCEDURE  User_Logoff(app_process IN VARCHAR2)
AS
  msgprop        dbms_aq.message_properties_t;
  enqopt         dbms_aq.enqueue_options_t;
  enq_msgid      RAW(16);
  payload        adtmsg;
BEGIN
  /* Visibility must always be immediate for NonPersistent queues: */
  enqopt.visibility:=dbms_aq.IMMEDIATE;
  msgprop.correlation:= 'LOGOFF';
  msgprop.recipient_list(0) := aq$_agent(app_process, NULL, NULL);
  /* Payload is NOT NULL: */
payload := adtmsg(1, 'Logging Off');

dbms_aq.enqueue(
        queue_name         => 'LOGON_LOGOFF',
        enqueue_options    => enqopt,
        message_properties => msgprop,
        payload            => payload,
        msgid              => enq_msgid);
 END;
/


/* If there is a login at APP1, enqueue a message into 'login_logoff' with
   correlation 'LOGIN': */
EXECUTE User_logon('APP1');

/* If there is a logout at APP3, enqueue a message into 'login_logoff' with
   correlation 'LOGOFF' and payload adtmsg(1, 'Logging Off'): */
EXECUTE User_logoff('App3');

 /* The OCI program which waits for notifications: */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>
#ifdef WIN32COMMON
```

```
#define sleep(x)   Sleep(1000*(x))
#endif

/* LOGON / password:  */
static text *username = (text *) "OE";
static text *password = (text *) "OE";

/* The correlation strings of messages: */
static char  *logon = "LOGON";
static char  *logoff = "LOGOFF";

/* The possible consumer names of queues: */
static char *applist[] = {"APP1", "APP2","APP3"};

static OCIEnv *envhp;
static OCIServer *srvhp;
static OCIError *errhp;
static OCISvcCtx *svchp;

static void checkerr(/*_ OCIError *errhp, sword status _*/);

struct process_statistics
{
  ub4  logon;
  ub4  logoff;
};

typedef struct process_statistics process_statistics;

int main(/*_ int argc, char *argv[] _*/);

/* Notify Callback: */
ub4 notifyCB(ctx, subscrhp, pay, payl, desc, mode)
dvoid *ctx;
LNOCISubscription *subscrhp;
dvoid *pay;
ub4    payl;
dvoid *desc;
ub4    mode;
{
 text               *subname;   /* subscription name */
 ub4                 lsub;      /* length of subscription name */
 text               *queue;     /* queue name */
 ub4                *lqueue;    /* queue name */
 text               *consumer;  /* consumer name */
 ub4                 lconsumer;
 text               *correlation;
```

```
ub4                  lcorrelation;
ub4                  size;
ub4                  appno;
OCIRaw              *msgid;
OCIAQMsgProperties  *msgprop;   /* message properties descriptor */
process_statistics   *user_count = (process_statistics *)ctx;

OCIAttrGet((dvoid *)subscrhp, OCI_HTYPE_SUBSCRIPTION,
                           (dvoid *)&subname, &lsub,
                           OCI_ATTR_SUBSCR_NAME, errhp);

/* Extract the attributes from the AQ descriptor: */
/* Queue name: */
OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&queue, &size,
          OCI_ATTR_QUEUE_NAME, errhp);

/* Consumer name: */
OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&consumer, &lconsumer,
          OCI_ATTR_CONSUMER_NAME, errhp);

/* Message properties: */
OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&msgprop, &size,
          OCI_ATTR_MSG_PROP, errhp);

/* Get correlation from message properties: */
 checkerr(errhp, OCIAttrGet(msgprop, OCI_DTYPE_AQMSG_PROPERTIES,
                           (dvoid *)&correlation, &lcorrelation,
                           OCI_ATTR_CORRELATION, errhp));

 if (lconsumer == strlen(applist[0]))
 {
   if (!memcmp((dvoid *)consumer, (dvoid *)applist[0], strlen(applist[0])))
    appno = 0;
   else if (!memcmp((dvoid *)consumer, (dvoid *)applist[1],
strlen(applist[1])))
    appno = 1;
   else if (!memcmp((dvoid *)consumer, (dvoid *)applist[2],
strlen(applist[2])))
    appno = 2;
   else
   {
    printf("Wrong consumer in notification");
    return;
   }
 }
 else
 {  /* consumer name must be "APP1", "APP2" or "APP3"  */
```

```
    printf("Wrong consumer in notification");
    return;
  }

  if (lcorrelation == strlen(logon) &&                    /* logon event */
       !memcmp((dvoid *)correlation, (dvoid *)logon, strlen(logon)))
  {
     user_count[appno].logon++;
                          /* increment logon count for the app process */
        printf("Logon by APP%d \n", (appno+1));
        printf("Logon Payload length = %d \n", pay1);
   }
  else if  (lcorrelation == strlen(logoff) &&           /* logoff event */
       !memcmp((dvoid *)correlation,(dvoid *)logoff, strlen(logoff)))
  {
     user_count[appno].logoff++;
                         /* increment logoff count for the app process */
     printf("Logoff by APP%d \n", (appno+1));
     printf("Logoff Payload length = %d \n", pay1);
  }
  else                              /* correlation is "LOGON" or "LOGOFF" */
    printf("Wrong correlation in notification");

  printf("Total  : \n");

  printf("App1 : %d \n", user_count[0].logon-user_count[0].logoff);
  printf("App2 : %d \n", user_count[1].logon-user_count[1].logoff);
  printf("App3 : %d \n", user_count[2].logon-user_count[2].logoff);

}

int main(argc, argv)
int argc;
char *argv[];
{
  OCISession *authp = (OCISession *) 0;
  OCISubscription *subscrhp[3];
  ub4 namespace = OCI_SUBSCR_NAMESPACE_AQ;
  process_statistics  ctx[3] = {{0,0}, {0,0}, {0,0}};
  ub4 sleep_time = 0;

  printf("Initializing OCI Process\n");

  /* Initialize OCI environment with OCI_EVENTS flag set: */
  (void) OCIInitialize((ub4) OCI_EVENTS|OCI_OBJECT, (dvoid *)0,
                       (dvoid * (*)(dvoid *, size_t)) 0,
                       (dvoid * (*)(dvoid *, dvoid *, size_t))0,
```

```
                        (void (*)(dvoid *, dvoid *)) 0 );

  printf("Initialization successful\n");

  printf("Initializing OCI Env\n");
  (void) OCIEnvInit( (OCIEnv **) &envhp, OCI_DEFAULT, (size_t) 0, (dvoid **) 0
);
  printf("Initialization successful\n");

  checkerr(errhp, OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &errhp,
LNOCI_HTYPE_ERROR,
                   (size_t) 0, (dvoid **) 0));

  checkerr(errhp, OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &srvhp,
LNOCI_HTYPE_SERVER,
                   (size_t) 0, (dvoid **) 0));

  checkerr(errhp, OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &svchp,
LNOCI_HTYPE_SVCCTX,
                   (size_t) 0, (dvoid **) 0));

  printf("connecting to server\n");
  checkerr(errhp, OCIServerAttach( srvhp, errhp, (text *)"inst1_alias",
          strlen("inst1_alias"), (ub4) OCI_DEFAULT));
  printf("connect successful\n");

  /* Set attribute server context in the service context: */
  checkerr(errhp, OCIAttrSet( (dvoid *) svchp, OCI_HTYPE_SVCCTX, (dvoid *)srvhp,
                   (ub4) 0, OCI_ATTR_SERVER, (OCIError *) errhp));

  checkerr(errhp, OCIHandleAlloc((dvoid *) envhp, (dvoid **)&authp,
                       (ub4) OCI_HTYPE_SESSION, (size_t) 0, (dvoid **) 0));

  /* Set username and password in the session handle: */
  checkerr(errhp, OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION,
                  (dvoid *) username, (ub4) strlen((char *)username),
                  (ub4) OCI_ATTR_USERNAME, errhp));

  checkerr(errhp, OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION,
                  (dvoid *) password, (ub4) strlen((char *)password),
                  (ub4) OCI_ATTR_PASSWORD, errhp));

  /* Begin session: */
  checkerr(errhp, OCISessionBegin ( svchp,  errhp, authp, OCI_CRED_RDBMS,
                        (ub4) OCI_DEFAULT));

  (void) OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX,
```

```
                    (dvoid *) authp, (ub4) 0,
                    (ub4) OCI_ATTR_SESSION, errhp);

  /* Register for notification: */
  printf("allocating subscription handle\n");
 subscrhp[0] = (OCISubscription *)0;
 (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&subscrhp[0],
                       (ub4) OCI_HTYPE_SUBSCRIPTION,
                       (size_t) 0, (dvoid **) 0);

 /* For application process APP1: */
 printf("setting subscription name\n");
 (void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) "OE.LOGON_LOGOFF:APP1",
                (ub4) strlen("OE.LOGON_LOGOFF:APP1"),
                (ub4) OCI_ATTR_SUBSCR_NAME, errhp);

 printf("setting subscription callback\n");
 (void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) notifyCB, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_CALLBACK, errhp);

(void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *)&ctx, (ub4)sizeof(ctx),
                (ub4) OCI_ATTR_SUBSCR_CTX, errhp);

 printf("setting subscription namespace\n");
 (void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) &namespace, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_NAMESPACE, errhp);

printf("allocating subscription handle\n");
 subscrhp[1] = (OCISubscription *)0;
 (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&subscrhp[1],
                       (ub4) OCI_HTYPE_SUBSCRIPTION,
                       (size_t) 0, (dvoid **) 0);

 /* For application process APP2: */
 printf("setting subscription name\n");
 (void) OCIAttrSet((dvoid *) subscrhp[1], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) "OE.LOGON_LOGOFF:APP2",
                (ub4) strlen("OE.LOGON_LOGOFF:APP2"),
                (ub4) OCI_ATTR_SUBSCR_NAME, errhp);

 printf("setting subscription callback\n");
 (void) OCIAttrSet((dvoid *) subscrhp[1], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) notifyCB, (ub4) 0,
```

```
                (ub4) OCI_ATTR_SUBSCR_CALLBACK, errhp);

(void) OCIAttrSet((dvoid *) subscrhp[1], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *)&ctx, (ub4)sizeof(ctx),
                (ub4) OCI_ATTR_SUBSCR_CTX, errhp);

printf("setting subscription namespace\n");
(void) OCIAttrSet((dvoid *) subscrhp[1], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) &namespace, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_NAMESPACE, errhp);

 printf("allocating subscription handle\n");
subscrhp[2] = (OCISubscription *)0;
(void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&subscrhp[2],
                      (ub4) OCI_HTYPE_SUBSCRIPTION,
                      (size_t) 0, (dvoid **) 0);

/* For application process APP3: */
printf("setting subscription name\n");
(void) OCIAttrSet((dvoid *) subscrhp[2], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) "OE.LOGON_LOGOFF:APP3",
                (ub4) strlen("OE.LOGON_LOGOFF:APP3"),
                (ub4) OCI_ATTR_SUBSCR_NAME, errhp);

printf("setting subscription callback\n");
(void) OCIAttrSet((dvoid *) subscrhp[2], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) notifyCB, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_CALLBACK, errhp);

(void) OCIAttrSet((dvoid *) subscrhp[2], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *)&ctx, (ub4)sizeof(ctx),
                (ub4) OCI_ATTR_SUBSCR_CTX, errhp);

printf("setting subscription namespace\n");
(void) OCIAttrSet((dvoid *) subscrhp[2], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) &namespace, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_NAMESPACE, errhp);

printf("Registering fornotifications \n");
checkerr(errhp, OCISubscriptionRegister(svchp, subscrhp, 3, errhp,
                                        OCI_DEFAULT));

sleep_time = (ub4)atoi(argv[1]);
printf ("waiting for %d s \n", sleep_time);
sleep(sleep_time);

printf("Exiting");
```

```
  exit(0);
}

void checkerr(errhp, status)
LNOCIError *errhp;
sword status;
{
  text errbuf[512];
  sb4 errcode = 0;

  switch (status)
  {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    (void) printf("Error - OCI_SUCCESS_WITH_INFO\n");
    break;
  case OCI_NEED_DATA:
    (void) printf("Error - OCI_NEED_DATA\n");
    break;
  case OCI_NO_DATA:
    (void) printf("Error - OCI_NODATA\n");
    break;
  case OCI_ERROR:
    (void) OCIErrorGet((dvoid *)errhp, (ub4) 1, (text *) NULL, &errcode,
                       errbuf, (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    (void) printf("Error - %.*s\n", 512, errbuf);
    break;
  case OCI_INVALID_HANDLE:
    (void) printf("Error - OCI_INVALID_HANDLE\n");
    break;
  case OCI_STILL_EXECUTING:
    (void) printf("Error - OCI_STILL_EXECUTE\n");
    break;
  case OCI_CONTINUE:
    (void) printf("Error - OCI_CONTINUE\n");
    break;
  default:
    break;
  }
}

/* End of file tkaqdocn.c */
```

Visual Basic（OO4O）：サンプル・コード

この機能は、現在サポートされていません。

Java（JDBC）：サンプル・コード

この機能は、Java API ではサポートされません。

保存およびメッセージ履歴

アドバンスト・キューイングでは、処理終了後にメッセージ履歴を保存できます。メッセージおよびその履歴は、SQL を使用して問合せできます。これによって、統合システムの業務上の分析が可能になります。場合によっては、メッセージの追跡が必要になります。たとえば、あるメッセージを処理した結果、他のメッセージが生成された場合、両者は関連付けられています。アプリケーション設計者としては、そのような関連を追跡することが必要な場合があります。保存、メッセージ識別子および SQL 問合せの協調によって、強力なメッセージ・ウェアハウスを構築できます。

シナリオ

注文の平均処理時間を判断する必要があると想定します。この時間には、backed_order キューでの待機時間も含まれます。backed_order キューでの平均待機時間を確認する必要があります。出荷キューの保存を TRUE に指定し、メッセージの相関フィールドに注文番号を指定すると、SQL 問合せによって、出荷処理アプリケーション中の注文の待機時間を判断できます。

単純化のために、処理済の注文のみを分析することにします。出荷処理アプリケーションで注文の処理にかかる時間は、WS_bookedorders_que にエンキューされる時刻と WS_shipped_orders_que にエンキューされる時刻の差です（C-2 ページの「tkaqdoca.sql: ユーザー、オブジェクト、キュー表、キューおよびサブスクライバ作成用のスクリプト」を参照）。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
SELECT  SUM(SO.enq_time - BO.enq_time) / count (*) AVG_PRCS_TIME
   FROM WS.AQ$WS_orders_pr_mqtab BO , WS.AQ$WS_orders_mqtab SO
   WHERE SO.msg_state = 'PROCESSED' and BO.msg_state = 'PROCESSED'
   AND SO.corr_id = BO.corr_id and SO.queue = 'WS_shippedorders_que';

/* Average waiting time in the backed order queue: */
SELECT SUM(BACK.deq_time - BACK.enq_time)/count (*) AVG_BACK_TIME
   FROM WS.AQ$WS_orders_mqtab BACK
   WHERE BACK.msg_state = 'PROCESSED' AND BACK.queue = 'WS_backorders_que';
```

Visual Basic（OO4O）：サンプル・コード

この機能については、データベースの dbexecutesql インタフェースを使用します。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

パブリッシュ・サブスクライブ・サポート

アドバンスト・キューイングは、アプリケーション統合のパブリッシュ・サブスクライブ・モデルをサポートします。このモデルでは、パブリッシュ側のアプリケーションによってメッセージがキューに挿入されます。サブスクライブ側のアプリケーションでは、メッセージがキューからサブスクライブされます。既存のパブリッシュ側アプリケーションおよびサブスクライブ側アプリケーションを変更せずに、多くのパブリッシュ側アプリケーションおよびサブスクライブ側アプリケーションを動的に追加できます。アドバンスト・キューイングは、内容ベースのサブスクリプションもサポートします。サブスクライバは、メッセージ・プロパティおよびメッセージの内容に基づいて、キューに挿入されたメッセージのサブセットをサブスクライブできます。キューのサブスクライバは、別のキューまたは別のキューのコンシューマになることもできます。

アドバンスト・キューイング使用して、パブリッシュ・サブスクライブ・モデルの通信を次のように実装できます。

- メッセージを保持するために 1 つまたは複数のキューを設定します。このキューは、関心がある領域またはサブジェクトを表しています。たとえば、あるキューは請求済注文を表すために使用される可能性があります。
- ルールベースのサブスクライバを 1 組設定します。各サブスクライバは、受信を希望するメッセージ仕様を表すルールを指定できます。NULL ルールは、サブスクライバがすべてのメッセージの受信を希望することを意味します。
- パブリッシャ・アプリケーションが、エンキュー・コールをコールしてキューにメッセージをパブリッシュします。
- サブスクライバ・アプリケーションは、次のような方法でメッセージを受信できます。
  - `DEQUEUE` コールが、サブスクリプションの基準に合うメッセージを取り出します。
  - `LISTEN` コールを使用して、様々なキューに対するサブスクリプションを複数のキューで監視できます。これは、サブスクライバ・アプリケーションが多数のキューにサブスクライブしていて、どのキューに届いたメッセージでも受信する場合には、非常にスケーラブルなソリューションです。
  - OCI 通知メカニズムを使用します。これによって、プッシュ・モードのメッセージ配信が可能になります。サブスクライバ・アプリケーションは、メッセージの受信元となるキュー（およびサブスクライブ・エージェントとして指定されるサブスクリプション）を登録します。これによって、サブスクリプションに一致するメッセージが届いたときに、コールされるコールバックが登録されます。

シナリオ

BooksOnLine アプリケーションは、アプリケーション間通信のパブリッシュ・サブスクライブ・モデルの使用方法を示しています。次の項で、いくつか例を示します。

キューの定義 注文入力アプリケーションでは、様々なアプリケーションに入力済注文および通信するためのキュー（OE_booked_orders_que）が定義されます。注文入力アプリケーションは、複数のサブスクライバ・アプリケーションを認識しないため、注文入力（パブリッシャ）アプリケーションの設定またはロジックを混乱させることなく、新しいサブスクライバ・アプリケーションを追加できます。

サブスクリプションの設定 様々な出荷処理アプリケーションおよび顧客サービス・アプリケーション（東部向け出荷、西部向け出荷、海外向け出荷および顧客サービス）は、注文入力アプリケーションの booked_orders キューに対するサブスクライバとして定義されます。ルールは、関心があるメッセージを様々なサブスクライバに送るために使用されます。たとえば、東部向け出荷は東海岸地区向けの注文および合衆国全域向けの至急注文を扱い、次のようなサブスクリプション・ルールの記述になります。

```
rule  => 'tab.user_data.orderregion = ''EASTERN'' OR
(tab.user_data.ordertype = ''RUSH'' AND
tab.user_data.customer.country = ''USA'') '
```

各サブスクライバは、メッセージが配信されるローカル・キューを指定できます。東部向け出荷処理アプリケーションでは、サブスクライバ・アドレスを次のように指定して、メッセージ配信用のローカル・キュー（ES_booked_orders_que）を指定します。

```
subscriber := aq$_agent('East_Shipping', 'ES.ES_bookedorders_que', null);
```

伝播の設定 各パブリッシャ・アプリケーション・キューから伝播できます。サブスクライブされたメッセージをリモート・キューに配信できるようにするために、注文入力アプリケーションでは次のように指定して伝播を使用可能にします。

```
execute dbms_aqadm.schedule_propagation(queue_name => 'OE.OE_bookedorders_que');
```

メッセージのパブリッシュ 注文入力アプリケーションが注文を妥当性チェックし、出荷準備完了後に（OE_booked_order_que に）エンキューしたときに、入力済注文がパブリッシュされます。このメッセージは、次に、サブスクライブしている各アプリケーションに送られます。メッセージは、各サブスクライバ・アプリケーションのローカル・キュー（指定されている場合）に配信されます。

メッセージの受信 出荷処理アプリケーションおよび顧客サービス・アプリケーションは、それぞれローカル・キューのメッセージを受信します。たとえば、東部向け出荷処理アプリケーションは、東海岸向けまたは RUSH というマークのある合衆国内向けの入力済注文のみを受信します。その後、このアプリケーションはメッセージをデキューして、その注文を出荷のために処理します。

Oracle Real Application Clusters のサポート

Real Application Clusters を使用すると、異なるキューを別々のインスタンスで管理できるようにして AQ パフォーマンスを改善できます。このためには、キューを格納するキュー表に様々なインスタンス・アフィニティ（作業環境）を指定します。これによって、様々なキューに対する操作（エンキューおよびデキュー）をパラレルで行うことができるようになります。

AQ のキュー・モニター・プロセスは、キュー表のインスタンス・アフィニティを継続的に監視します。キュー・モニターは指定されたプライマリ・インスタンスが使用可能な場合はそれにキュー表の所有権を割り当て、失敗した場合は指定されたセカンダリ・インスタンスに割り当てます。

キュー表を所有しているインスタンスが終了する場合、キュー・モニターはプライマリ・インスタンスなどの適したインスタンスに、所有者を変更します。

AQ の伝播は Real Application Clusters でも使用可能になりますが、これはユーザーにとっては透過的です。伝播スケジュールのジョブ・アフィニティは、それぞれのキュー表のアフィニティと同じ値に設定されます。このように、キュー表を所有するインスタンスに対応付けられたジョブ・キュー・プロセスは、そのキュー表に格納されているキューからの伝播を処理し、ping 操作を最小限に抑えます。詳細は、9-72 ページの「キューの伝播のスケジューリング」を参照してください。

> **参照：**『Oracle9*i* Real Application Clusters セットアップおよび構成』を参照してください。

シナリオ

BooksOnLine の例では、注文入力（OE）サイトでの new_orders_queue および booked_order_queue の操作は、この 2 つのキューが異なるインスタンスと対応付けられている場合は高速化できます。そのために、2 つのキューを別々のキュー表に作成し、create_queue_table() コマンドでそれらのキュー表に別々のアフィニティを指定します。

この例では、キュー表 OE_orders_sqtab にキュー new_orders_queue を格納し、プライマリおよびセカンダリ・インスタンスはそれぞれインスタンス 1 および 2 です。キュー表 OE_orders_mqtab にはキュー booked_order_queue を格納し、プライマリおよびセカンダリ・インスタンスはそれぞれインスタンス 2 および 1 です。目的は、インスタンス 1 および 2 に 2 つのキューをパラレルで管理させることです。デフォルトでは、1 つのインスタンスのみが使用可能であり、この場合、両方のキュー表の所有者はインスタンス 1 に設定されます。ただし、Real Application Clusters が正しく設定され、インスタンス 1 および 2 が両方とも使用可能な場合は、キュー表 OE_orders_sqtab がインスタンス 1 によって所有され、他のキュー表はインスタンス 2 によって所有されます。キュー表に対するプライマリおよびセカンダリ・インスタンスの指定は、次の例に示すとおり、alter_queue_table() コマンドを使用して動的に変更できます。キュー表のプライマリ、

セカンダリおよび所有者インスタンスに関する情報は、USER_QUEUE_TABLES ビューを問い合せて取得できます（10-21 ページの「ユーザー・スキーマのキュー表の選択」を参照）。

> **注意：** キュー名およびキュー表名は、大文字に変換されます。大文字と小文字が混在しているキュー名およびキュー表名はサポートされません。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/* Create queue tables, queues for OE  */
CONNECT OE/OE;
EXECUTE dbms_aqadm.create_queue_table( \
        queue_table        => 'OE_orders_sqtab',\
        comment            => 'Order Entry Single-Consumer Orders queue table',\
        queue_payload_type => 'BOLADM.order_typ',\
        compatible         => '8.1',\
        primary_instance   => 1,\
        secondary_instance => 2);

EXECUTE dbms_aqadm.create_queue_table(\
        queue_table        => 'OE_orders_mqtab',\
        comment            => 'Order Entry Multi Consumer Orders queue table',\
        multiple_consumers => TRUE,\
        queue_payload_type => 'BOLADM.order_typ',\
        compatible         => '8.1',\
        primary_instance   => 2,\
        secondary_instance => 1);

EXECUTE dbms_aqadm.create_queue ( \
        queue_name         => 'OE_neworders_que',\
        queue_table        => 'OE_orders_sqtab');

EXECUTE dbms_aqadm.create_queue ( \
        queue_name         => 'OE_bookedorders_que',\
        queue_table        => 'OE_orders_mqtab');

/* Check instance affinity of OE queue tables from AQ administrative view: */
SELECT queue_table, primary_instance, secondary_instance, owner_instance
FROM user_queue_tables;

/* Alter instance affinity of OE queue tables: */
EXECUTE dbms_aqadm.alter_queue_table( \
        queue_table        => 'OE.OE_orders_sqtab',\
        primary_instance   => 2,\
        secondary_instance => 1);

EXECUTE dbms_aqadm.alter_queue_table(  \
```

```
        queue_table        => 'OE.OE_orders_mqtab', \
        primary_instance   => 1,\
        secondary_instance => 2);

/* Check instance affinity of OE queue tables from AQ administrative view: */
SELECT queue_table, primary_instance, secondary_instance, owner_instance
FROM user_queue_tables;
```

Visual Basic（OO4O）：サンプル・コード

この機能は、現在サポートされていません。

Java（JDBC）：サンプル・コード

```
public static void createQueueTablesAndQueues(Connection db_conn)
{
    AQSession            aq_sess;
    AQQueueTableProperty sqt_prop;
    AQQueueTableProperty mqt_prop;
    AQQueueTable         sq_table;
    AQQueueTable         mq_table;
    AQQueueProperty      q_prop;
    AQQueue              neworders_q;
    AQQueue              bookedorders_q;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        /* Create a single-consumer orders queue table */
        sqt_prop = new AQQueueTableProperty("BOLADM.order_typ");
        sqt_prop.setComment("Order Entry Single-Consumer Orders queue table");
        sqt_prop.setCompatible("8.1");
        sqt_prop.setPrimaryInstance(1);
        sqt_prop.setSecondaryInstance(2);

        sq_table = aq_sess.createQueueTable("OE", "OE_orders_sqtab", sqt_prop);

        /* Create a multiconsumer orders queue table */
        mqt_prop = new AQQueueTableProperty("BOLADM.order_typ");
        mqt_prop.setComment("Order Entry Multi Consumer Orders queue table");
        mqt_prop.setCompatible("8.1");
        mqt_prop.setMultiConsumer(true);
        mqt_prop.setPrimaryInstance(2);
        mqt_prop.setSecondaryInstance(1);
```

```
        mq_table = aq_sess.createQueueTable("OE", "OE_orders_mqtab", mqt_prop);


        /* Create Queues in these queue tables */
        q_prop = new AQQueueProperty();

        neworders_q = aq_sess.createQueue(sq_table, "OE_neworders_que",
                                          q_prop);

        bookedorders_q = aq_sess.createQueue(mq_table, "OE_bookedorders_que",
                                             q_prop);

    }
    catch (AQException ex)
    {
        System.out.println("AQ Exception: " + ex);
    }
}

public static void alterInstanceAffinity(Connection db_conn)
{
    AQSession            aq_sess;
    AQQueueTableProperty sqt_prop;
    AQQueueTableProperty mqt_prop;
    AQQueueTable         sq_table;
    AQQueueTable         mq_table;
    AQQueueProperty      q_prop;

    try
    {

        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        /* Check instance affinities */
        sq_table = aq_sess.getQueueTable("OE", "OE_orders_sqtab");

        sqt_prop = sq_table.getProperty();
        System.out.println("Current primary instance for OE_orders_sqtab: " +
                           sqt_prop.getPrimaryInstance());

        mq_table = aq_sess.getQueueTable("OE", "OE_orders_mqtab");
        mqt_prop = mq_table.getProperty();
        System.out.println("Current primary instance for OE_orders_mqtab: " +
                           mqt_prop.getPrimaryInstance());

        /* Alter queue table affinities */
```

```
            sq_table.alter(null, 2, 1);

            mq_table.alter(null, 1, 2);

            sqt_prop = sq_table.getProperty();
            System.out.println("Current primary instance for OE_orders_sqtab: " +
                               sqt_prop.getPrimaryInstance());

            mq_table = aq_sess.getQueueTable("OE", "OE_orders_mqtab");
            mqt_prop = mq_table.getProperty();
            System.out.println("Current primary instance for OE_orders_mqtab: " +
                               mqt_prop.getPrimaryInstance());

        }
        catch (AQException ex)
        {
            System.out.println("AQ Exception: " + ex);
        }
    }
```

統計ビューのサポート

各インスタンスは、それぞれの AQ 統計情報をシステム・グローバル領域（SGA）に所有し、他のインスタンスによって収集された統計については認識しません。ただし、あるインスタンスで GV$AQ ビューに対して問い合せると、その他のすべてのインスタンスからそれぞれの AQ 統計情報が問合せ元のインスタンスに集計されます。

シナリオ

GV$ ビューでは、待機中、準備完了または期限切れの各メッセージ数を必要なときにいつでも問い合せることができます。また、メッセージが処理されるまでの平均待機秒数も表示します。注文処理アプリケーションでは、これを使用して注文処理プロセスの数を動的にチューニングできます（10-33 ページの「データベース全体における状態ごとのメッセージ数の選択」を参照）。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CONNECT oe/oe

/* Count the number as messages and the average time for which the messages have
   been waiting: */
SELECT READY, AVERAGE_WAIT FROM gv$aq Stats, user_queues Qs
  WHERE Stats.qid = Qs.qid and Qs.Name = 'OE_neworders_que';
```

Visual Basic（OO4O）：サンプル・コード

この機能については、データベースの dbexecutesql インタフェースを使用します。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

インターネット・アクセス

アドバンスト・キューイング機能へのインターネット・アクセスの詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

エンキュー機能

この項の内容は次のとおりです。

- サブスクリプションおよび受信者リスト
- メッセージの優先順位および順序付け
- 時間指定 : 遅延
- 時間指定 : 期限切れ
- メッセージのグループ化
- 遅延間隔をおいた後の再試行
- エンキュー中のメッセージ変換
- AQ XML サーブレットを使用したエンキュー

サブスクリプションおよび受信者リスト

デキューされて処理済になったメッセージは、retention_time に指定された期間保存されます。retention_time が期限切れになると、メッセージはタイム・マネージャ・プロセスによって削除されます。

処理が済むと、キューの retention_time が 0（ゼロ）の場合はそのメッセージは削除され、そうでない場合は指定された期間保存されます。メッセージが保存されている間は、キュー表ビューに対する SQL 問合せでそのメッセージを参照したり、BROWSE（ブラウズ）モードで処理済メッセージの ID を指定してデキューすることができます。

アドバンスト・キューイングによって、単一のメッセージを複数のコンシューマが処理できます。この機能を使用するには、マルチ・コンシューマ・キューを作成し、それらのマルチ・コンシューマ・キューにメッセージをエンキューする必要があります。アドバンスト・キューイングでは、メッセージに対してコンシューマ・リストを識別する 2 つの方法（サブスクリプション・リストおよび受信者リスト）が使用できます。

サブスクリプション

PL/SQL プロシージャ DBMS_AQADM.ADD_SUBSCRIBER を使用して、キューにサブスクリプションを追加できます（9-59 ページの「サブスクライバの追加」を参照）。これによって、エンキューされたメッセージに対する AQ$_AGENT パラメータでコンシューマを指定できます。多くのサブスクライバを追加するには、DBMS_AQADM.ADD_SUBSCRIBER プロシージャを繰り返し使用することで、1 つのマルチ・コンシューマ・キュー当たり最大 1024 サブスクライバを追加できます。

マルチ・コンシューマ・キューにサブスクライバとして追加されたすべてのコンシューマは、AQ$_AGENT パラメータに一意の値を持つ必要があります。すなわち、2 つのサブスクライバが AQ$_AGENT 型の NAME、ADDRESS および PROTOCOL 属性に同じ値を持つことはできません。この 3 つの属性のうち 1 つ以上を、2 つのサブスクライバに対して異なる値にする必要があります（このデータ構造の詳細は、2-3 ページの「エージェント型（aq$_agent）」を参照）。

シングル・コンシューマ・キューまたは例外キューには、サブスクリプションを追加できません。あるキューにサブスクライバとして追加されたコンシューマは、DBMS_AQADM.ADD_SUBSCRIBER プロシージャが完了した後にエンキューされたメッセージしかデキューできません。つまり、このプロシージャが実行される前にエンキューされたメッセージは、このコンシューマからはデキューできません。

サブスクリプションを削除するには、DBMS_AQADM.REMOVE_SUBSCRIBER プロシージャを使用します（9-69 ページの「サブスクライバの削除」を参照）。AQ は、AQ$_AGENT パラメータによって識別されるコンシューマに対応するすべてのデータを、自動的にキューから削除します。つまり、そのコンシューマがデキューできるが保留しているメッセージがあるときに、REMOVE_SUBSCRIBER プロシージャを実行してもエラーにはなりません。このメッセージは、REMOVE_SUBSCRIBER プロシージャが実行されると自動的にデキューに対して使用不可になります。8.1 以上に設定された互換パラメータで作成されたキュー表では、コンシューマによってデキューされなかったメッセージが、AQ$<queue_table> ビューに「UNDELIVERABLE」と表示されます。互換パラメータなしで作成されたマルチ・コンシューマ・キュー表、または 8.0 に設定された互換パラメータで作成されたマルチ・コンシューマ・キュー表は、コンシューマごとのメッセージ状態を表示せず、全体的なメッセージの状態のみを表示します。

受信者リスト

メッセージをエンキューしたプロデューサが、コンシューマの受信者リストを提供している場合、マルチ・コンシューマ・キューにサブスクリプションを指定する必要はありません。状況によっては、デフォルトのサブスクライバ・リストのかわりに、特定の一連のコンシューマをターゲットに指定したメッセージをエンキューする方が望ましいこともあります。このような場合は、メッセージをエンキューするときに受信者リストを指定します。

- PL/SQL では、message_properties レコードの recipient_list フィールドに要素を追加して、受信者リストを指定します。
- OCI では、LNOCISetAttr プロシージャを使用し、LNOCI_DTYPE_AQAGENT 記述子の配列を LNOCI_DTYPE_AQMSG_PROPERTIES メッセージ・プロパティ記述子の受信者リ

スト（LNOCI_ATTR_RECIPIENT_LIST 属性）として指定することによって、受信者リストを指定します。

エンキュー中に指定された受信者リストは、サブスクリプション・リストをオーバーライドします。つまり、受信者リストが指定されているメッセージは、そのキューのサブスクライバからはデキューできません。受信者リストで指定されたコンシューマは、そのキューのサブスクライバの場合もあれば、そうでない場合もあります。サブスクライバがないキューに受信者リストを指定しないでエンキューすると、エラーになります（11-4 ページの「メッセージのエンキュー」を参照）。

メッセージの優先順位および順序付け

メッセージの順序付けとは、キューからメッセージがデキューされる順序を示します。キューに対する順序付けの方法は、キュー表の作成時に指定されます（9-4 ページの「キュー表の作成」を参照）。

メッセージの優先順位による順序付けは、優先順位およびエンキュー時刻でメッセージのソート順序を指定することによって実行されます。優先順位による順序付けを選択すると、各メッセージがエンキューされるときにエンキュー元によって優先順位が割り当てられます。デキューのときには、割り当てられた優先順位でデキューされます。2 つのメッセージに同じ優先順位が割り当てられた場合、両者のデキュー順序はエンキュー時刻によって決定されます。先入れ先出し（FIFO）優先順位のキューも、エンキュー時刻および優先順位でメッセージのソート順序を指定するこによって作成できます。

シナリオ

BooksOnLine アプリケーションでは、顧客は次の方法を要求できます。

- FedEx による出荷（優先順位 1）
- 優先扱い航空便による出荷（優先順位 2）
- 通常地上便による出荷（優先順位 3）

注文入力アプリケーションは、入力済注文を FIFO 優先順位のキューに格納します。入力済注文は、地区の入力済注文キューに伝播されます。各地区では、その地区の入力済注文キューにある注文が、出荷の優先順位に従って処理されます。

次のコールで、注文入力アプリケーションに優先順位キューを作成します。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/* Create a priority queue table for OE: */
EXECUTE dbms_aqadm.create_queue_table( \
   queue_table         => 'OE_orders_pr_mqtab', \
   sort_list           =>'priority,enq_time', \
   comment             => 'Order Entry Priority  \
                          MultiConsumer Orders queue table',\
   multiple_consumers  => TRUE, \
   queue_payload_type  => 'BOLADM.order_typ', \
   compatible          => '8.1', \
   primary_instance    => 2, \
   secondary_instance  => 1);

EXECUTE dbms_aqadm.create_queue ( \
   queue_name          => 'OE_bookedorders_que', \
   queue_table         => 'OE_orders_pr_mqtab');

/* When an order arrives, the order entry application can use the following
   procedure to enqueue the order into its booked orders queue. A shipping
   priority is specified for each order: */
CREATE OR REPLACE procedure order_enq(book_title        IN VARCHAR2,
                                      book_qty          IN NUMBER,
                                      order_num         IN NUMBER,
                                      shipping_priority IN NUMBER,
                                      cust_state        IN VARCHAR2,
                                      cust_country      IN VARCHAR2,
                                      cust_region       IN VARCHAR2,
                                      cust_ord_typ      IN VARCHAR2) AS

OE_enq_order_data        BOLADM.order_typ;
OE_enq_cust_data         BOLADM.customer_typ;
OE_enq_book_data         BOLADM.book_typ;
OE_enq_item_data         BOLADM.orderitem_typ;
OE_enq_item_list         BOLADM.orderitemlist_vartyp;
enqopt                   dbms_aq.enqueue_options_t;
msgprop                  dbms_aq.message_properties_t;
enq_msgid                RAW(16);

BEGIN
   msgprop.correlation := cust_ord_typ;
   OE_enq_cust_data    := BOLADM.customer_typ(NULL, NULL, NULL, NULL,
                                 cust_state, NULL, cust_country);
   OE_enq_book_data    := BOLADM.book_typ(book_title, NULL, NULL, NULL);
   OE_enq_item_data    := BOLADM.orderitem_typ(book_qty,
                                 OE_enq_book_data, NULL);
   OE_enq_item_list    := BOLADM.orderitemlist_vartyp(
                                 BOLADM.orderitem_typ(book_qty,
```

```
                                OE_enq_book_data, NULL));
   OE_enq_order_data   := BOLADM.order_typ(order_num, NULL,
                                cust_ord_typ, cust_region,
                                OE_enq_cust_data, NULL,
                                OE_enq_item_list, NULL);

   /*Put the shipping priority into message property before enqueuing
     the message: */
   msgprop.priority    := shipping_priority;
   dbms_aq.enqueue('OE.OE_bookedorders_que', enqopt, msgprop,
                        OE_enq_order_data, enq_msgid);
        COMMIT;
  END;
  /


/* At each region, similar booked order queues are created. The orders are
   propagated from the central Order Entry's booked order queues to the regional
   booked order queues.For example, at the western region, the booked orders
   queue is created.
   Create a priority queue table for WS shipping: */
EXECUTE dbms_aqadm.create_queue_table( \
   queue_table        =>  'WS_orders_pr_mqtab',
   sort_list          =>'  priority,enq_time',  \
   comment            =>  'West Shipping Priority  \
                           MultiConsumer Orders queue table',\
   multiple_consumers => TRUE, \
   queue_payload_type => 'BOLADM.order_typ', \
   compatible         => '8.1');

/* Booked orders are stored in the priority queue table: */
EXECUTE dbms_aqadm.create_queue ( \
   queue_name         => 'WS_bookedorders_que', \
   queue_table        => 'WS_orders_pr_mqtab');

/* At each region, the shipping application dequeues orders from the regional
   booked order queue according to the orders' shipping priorities, processes
   the orders, and enqueues the processed orders into the shipped orders queues
   or the back orders queues. */
```

Visual Basic（OO4O）：サンプル・コード

```
Dim OraSession as object
Dim OraDatabase as object
Dim OraAq as object
Dim OraMsg as Object
Dim OraOrder,OraCust,OraBook,OraItem,OraItemList as Object
Dim Msgid as String

   Set OraSession = CreateObject("OracleInProcServer.XOraSession")
   Set OraDatabase = OraSession.DbOpenDatabase("dbname", "user/pwd", 0&)
   set oraaq = OraDatabase.CreateAQ("OE.OE_bookedorders_que")
   Set OraMsg = OraAq.AQMsg(ORATYPE_OBJECT, "BOLADM.order_typ")
   Set OraOrder = OraDatabase.CreateOraObject("BOLADM.order_typ")
   Set OraCust = OraDatabase.CreateOraObject("BOLADM.Customer_typ")
   Set OraBook = OraDatabase.CreateOraObject("BOLADM.book_typ")
   Set OraItem = OraDatabase.CreateOraObject("BOLADM.orderitem_typ")
   Set OraItemList = OraDatabase.CreateOraObject("BOLADM.orderitemlist_vartyp")

   ' Get the values of cust_state,cust_country etc from user(form_based
   ' input) and then a cmd_click event for Enqueue
   ' will execute the subroutine order_enq.
   Private Sub Order_enq()

   OraMsg.correlation = txt_correlation
   'Initialize the customer details
        OraCust("state") = txt_cust_state
   OraCust("country") = txt_cust_country
        OraBook("title") = txt_book_title
   OraItem("quantity") = txt_book_qty
   OraItem("item") = OraBook
   OraItemList(1) = OraItem
   OraOrder("orderno") = txt_order_num
   OraOrder("ordertype") = txt_cust_order_typ
   OraOrder("orderregion") = cust_region
   OraOrder("customer") = OraCust
   OraOrder("items") = OraItemList

   'Put the shipping priority into message property before enqueuing
   '  the message:
   OraMsg.priority = priority
   OraMsg = OraOrder
   Msgid = OraAq.enqueue

   'Release all allocations
   End Sub
```

Java（JDBC）：サンプル・コード

```
public static void createPriorityQueueTable(Connection db_conn)
{
    AQSession            aq_sess;
    AQQueueTableProperty mqt_prop;
    AQQueueTable         pr_mq_table;
    AQQueueProperty      q_prop;
    AQQueue              bookedorders_q;

    try
    {

        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        /* Create a priority queue table for OE */
        mqt_prop = new AQQueueTableProperty("BOLADM.order_typ");
        mqt_prop.setComment("Order Entry Priority " +
                            "MultiConsumer Orders queue table");
        mqt_prop.setCompatible("8.1");
        mqt_prop.setMultiConsumer(true);

        mqt_prop.setSortOrder("priority,enq_time");

        pr_mq_table = aq_sess.createQueueTable("OE", "OE_orders_pr_mqtab",
                                               mqt_prop);

        /* Create a Queue in this queue table */
        q_prop = new AQQueueProperty();

        bookedorders_q = aq_sess.createQueue(pr_mq_table,
                                             "OE_bookedorders_que", q_prop);

        /* Enable enqueue and dequeue on the queue */
        bookedorders_q.start(true, true);

    }
    catch (AQException ex)
    {
        System.out.println("AQ Exception: " + ex);
    }
}


/* When an order arrives, the order entry application can use the following
   procedure to enqueue the order into its booked orders queue. A shipping
   priority is specified for each order
```

```
 */
public static void order_enqueue(Connection db_conn, String book_title,
                                 double book_qty, double order_num,
                                 int ship_priority, String cust_state,
                                 String cust_country, String cust_region,
                                 String cust_order_type)
{
    AQSession         aq_sess;
    AQQueue           bookedorders_q;
    Order             enq_order;
    Customer          cust_data;
    Book              book_data;
    OrderItem         item_data;
    OrderItem[]       items;
    OrderItemList     item_list;
    AQEnqueueOption   enq_option;
    AQMessageProperty m_property;
    AQMessage         message;
    AQObjectPayload   obj_payload;
    byte[]            enq_msg_id;

    try
    {

        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        cust_data = new Customer();
        cust_data.setCountry(cust_country);
        cust_data.setState(cust_state);

        book_data = new Book();
        book_data.setTitle(book_title);

        item_data = new OrderItem();
        item_data.setQuantity(new BigDecimal(book_qty));
        item_data.setItem(book_data);

        items = new OrderItem[1];
        items[0] = item_data;

        item_list = new OrderItemList(items);

        enq_order = new Order();
        enq_order.setCustomer(cust_data);
        enq_order.setItems(item_list);
        enq_order.setOrderno(new BigDecimal(order_num));
```

```
            enq_order.setOrdertype(cust_order_type);

            bookedorders_q = aq_sess.getQueue("OE", "OE_bookedorders_que");

            message = bookedorders_q.createMessage();

            /* Put the shipping priority into message property before enqueuing */
            m_property = message.getMessageProperty();

            m_property.setPriority(ship_priority);

            obj_payload = message.getObjectPayload();

            obj_payload.setPayloadData(enq_order);

            enq_option = new AQEnqueueOption();

            /* Enqueue the message */
            enq_msg_id = bookedorders_q.enqueue(enq_option, message);

            db_conn.commit();

        }
        catch (AQException aq_ex)
        {
            System.out.println("AQ Exception: " + aq_ex);
        }
        catch (SQLException sql_ex)
        {
            System.out.println("SQL Exception: " + sql_ex);
        }

}

/* At each region, similar booked order queues are created. The orders are
   propagated from the central Order Entry's booked order queues to the
   regional booked order queues.
   For example, at the western region, the booked orders queue is created.
   Create a priority queue table for WS shipping
 */
public static void createWesternShippingQueueTable(Connection db_conn)
{
    AQSession            aq_sess;
    AQQueueTableProperty mqt_prop;
    AQQueueTable         mq_table;
    AQQueueProperty      q_prop;
    AQQueue              bookedorders_q;
```

```
    try
    {

        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);


        /* Create a priority queue table for WS */
        mqt_prop = new AQQueueTableProperty("BOLADM.order_typ");
        mqt_prop.setComment("Western Shipping Priority " +
                            "MultiConsumer Orders queue table");
        mqt_prop.setCompatible("8.1");
        mqt_prop.setMultiConsumer(true);
        mqt_prop.setSortOrder("priority,enq_time");

        mq_table = aq_sess.createQueueTable("WS", "WS_orders_pr_mqtab",
                                            mqt_prop);

        /* Booked orders are stored in the priority queue table: */
        q_prop = new AQQueueProperty();

        bookedorders_q = aq_sess.createQueue(mq_table, "WS_bookedorders_que",
                                             q_prop);

        /* Start the queue */
        bookedorders_q.start(true, true);

    }
    catch (AQException ex)
    {
        System.out.println("AQ Exception: " + ex);
    }

  /* At each region, the shipping application dequeues orders from the
     regional booked order queue according to the orders' shipping priorities,
     processes the orders, and enqueues the processed orders into the shipped
     orders queues or the back orders queues.
   */
}
```

時間指定 : 遅延

AQ は、メッセージをエンキューするエンキュー元に遅延間隔を指定させることで、メッセージの遅延配信をサポートします。つまり、デキュー・コールによってメッセージを取り出せるようになるまでの時間を指定できます（11-10 ページの「メッセージのエンキュー（メッセージ・プロパティの指定）」を参照）。エンキューされたメッセージに、デキュー元で使用可能というマークを付ける時刻が、遅延間隔によって決定されます。

あるメッセージが遅延時間指定付きでエンキューされると、そのメッセージには WAIT 状態というマークが付きます。WAIT 状態のメッセージは、デフォルトのデキュー・コールからはマスクされます。バックグラウンドのタイム・マネージャ・デーモンは定期的にアクティブになり、内部索引を使用して WAIT 状態にあるすべてのメッセージでスキャンし、遅延時間が過ぎたメッセージには READY 状態というマークを付けます。次に、タイム・マネージャは、メッセージが使用可能になったキューで待機しているすべてのフォアグラウンド・プロセスに、そのことを通知します。

シナリオ

BooksOnLine アプリケーションでは、遅延は遅延請求処理を実装するために使用できます。請求処理アプリケーションはキューを定義して、その繰延べ請求処理キューの中には出荷済ですぐには請求されない注文が遅延とともに入力されるようにできます。たとえば、法人顧客のような顧客アカウントの中には、15 日間経過するまで請求が発行されないケースがあります。請求処理アプリケーションは、受け取った出荷済注文メッセージを（shippedorders キューから）デキューし、それが法人顧客からの注文である場合は、遅延とともに繰延べ請求処理キューにエンキューします。

PL/SQL（DBMS_AQADM）: サンプル・コード

```
/* Enqueue an order to implement deferred billing so that the order is not made
   visible again until delay has expired: */
CREATE OR REPLACE PROCEDURE defer_billing(deferred_billing_order order_typ)
AS
  defer_bill_queue_name    VARCHAR2(62);
  enqopt                   dbms_aq.enqueue_options_t;
  msgprop                  dbms_aq.message_properties_t;
  enq_msgid                RAW(16);
BEGIN

/* Enqueue the order into the deferred billing queue with a delay of 15 days: */
  defer_bill_queue_name := 'CBADM.deferbilling_que';
  msgprop.delay := 15*60*60*24;
  dbms_aq.enqueue(defer_bill_queue_name, enqopt, msgprop,
                  deferred_billing_order, enq_msgid);
END;
/
```

Visual Basic（OO4O）：サンプル・コード

```
set oraaq = OraDatabase.CreateAQ("CBADM.deferbilling_que")
Set OraMsg = OraAq.AQMsg(ORATYPE_OBJECT, "BOLADM.order_typ")
Set OraOrder = OraDatabase.CreateOraObject("BOLADM.order_typ")

Private Sub defer_billing

OraMsg = OraOrder
OraMsg.delay = 15*60*60*24
OraMsg = OraOrder 'OraOrder contains the order details
Msgid = OraAq.enqueue

End Sub
```

Java（JDBC）：サンプル・コード

```
public static void defer_billing(Connection db_conn, Order deferred_order)
{
    AQSession         aq_sess;
    AQQueue           def_bill_q;
    AQEnqueueOption   enq_option;
    AQMessageProperty m_property;
    AQMessage         message;
    AQObjectPayload   obj_payload;
    byte[]            enq_msg_id;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        def_bill_q = aq_sess.getQueue("CBADM", "deferbilling_que");

        message = def_bill_q.createMessage();

        /* Enqueue the order into the deferred billing queue with a delay
           of 15 days */
        m_property = message.getMessageProperty();
        m_property.setDelay(15*60*60*24);

        obj_payload = message.getObjectPayload();
        obj_payload.setPayloadData(deferred_order);

        enq_option = new AQEnqueueOption();

        /* Enqueue the message */
        enq_msg_id = def_bill_q.enqueue(enq_option, message);
```

```
            db_conn.commit();
        }
        catch (Exception ex)
        {
            System.out.println("Exception " + ex);
        }

    }
```

時間指定 : 期限切れ

メッセージをエンキューするときに、そのメッセージがいつまで使用可能かという期限切れを指定できます。期限切れ処理を行うためには、キュー・モニターが起動されている必要があることに注意してください。また、プロデューサは、メッセージが期限切れになる時刻も指定できます。期限切れになると、メッセージは例外キューに移されます。

シナリオ

BooksOnLine アプリケーションでは、期限切れは受注残処理にあてられる時間を制御するために使用できます。出荷処理アプリケーションは、処理できない書籍注文を受注残キューに送ります。すべての受注残を 1 週間以内に必ず出荷するという方針であれば、1 週間という期限切れとともにメッセージを受注残キューにエンキューできます。このケースでは、1 週間以内に処理されなかった受注残は例外キューに移されて、EXPIRED というメッセージ状態になります。これは、受注残の出荷方針に従って未出荷になっている注文情報にフラグを付けるために使用できます。

PL/SQL（DBMS_AQADM）: サンプル・コード

```
CONNECT BOLADM/BOLADM
/* Req-enqueue a back order into a back order queue and set a delay of 7 days;
   all back orders must be processed in 7 days or they are moved to the
   exception queue: */
CREATE OR REPLACE PROCEDURE requeue_back_order(sale_region varchar2,
                                               backorder order_typ)
AS
  back_order_queue_name    VARCHAR2(62);
  enqopt                   dbms_aq.enqueue_options_t;
  msgprop                  dbms_aq.message_properties_t;
  enq_msgid                RAW(16);
BEGIN
  /* Look up a back order queue based the the region by means of a directory
     service: */
  IF sale_region = 'WEST' THEN
    back_order_queue_name := 'WS.WS_backorders_que';
  ELSIF sale_region = 'EAST' THEN
    back_order_queue_name := 'ES.ES_backorders_que';
```

```
  ELSE
    back_order_queue_name := 'OS.OS_backorders_que';
  END IF;

  /* Enqueue the order with expiration set to 7 days: */
  msgprop.expiration := 7*60*60*24;
  dbms_aq.enqueue(back_order_queue_name, enqopt, msgprop,
                  backorder, enq_msgid);
END;
/
```

Visual Basic（OO4O）：サンプル・コード

```
   set oraaq1 = OraDatabase.CreateAQ("WS.WS_backorders_que")
   set oraaq2 = OraDatabase.CreateAQ("ES.ES_backorders_que")
   set oraaq3 = OraDatabase.CreateAQ("CBADM.deferbilling_que")
   Set OraMsg = OraAq.AQMsg(ORATYPE_OBJECT, "BOLADM.order_typ")
   Set OraBackOrder = OraDatabase.CreateOraObject("BOLADM.order_typ")

Private Sub Requeue_backorder
   Dim q as oraobject
   If sale_region = WEST then
      q = oraaq1
   else if sale_region = EAST then
      q = oraaq2
   else
      q = oraaq3
   end if

   OraMsg.delay = 7*60*60*24
   OraMsg = OraBackOrder 'OraOrder contains the order details
   Msgid = q.enqueue

End Sub
```

Java（JDBC）：サンプル・コード

```
/* Re-enqueue a back order into a back order queue and set a delay of 7 days;
   all back orders must be processed in 7 days or they are moved to the
   exception queue */
public static void requeue_back_order(Connection db_conn,
                                      String sale_region, Order back_order)
{
    AQSession         aq_sess;
    AQQueue           back_order_q;
```

```
    AQEnqueueOption   enq_option;
    AQMessageProperty m_property;
    AQMessage         message;
    AQObjectPayload   obj_payload;
    byte[]            enq_msg_id;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        /* Look up a back order queue based on the region */
        if(sale_region.equals("WEST"))
        {
            back_order_q = aq_sess.getQueue("WS", "WS_backorders_que");
        }
        else if(sale_region.equals("EAST"))
        {
            back_order_q = aq_sess.getQueue("ES", "ES_backorders_que");
        }
        else
        {
            back_order_q = aq_sess.getQueue("OS", "OS_backorders_que");
        }

        message = back_order_q.createMessage();

        m_property = message.getMessageProperty();

        /* Enqueue the order with expiration set to 7 days: */
        m_property.setExpiration(7*60*60*24);

        obj_payload = message.getObjectPayload();
        obj_payload.setPayloadData(back_order);

        enq_option = new AQEnqueueOption();

        /* Enqueue the message */
        enq_msg_id = back_order_q.enqueue(enq_option, message);

        db_conn.commit();
    }
    catch (Exception ex)
    {
        System.out.println("Exception :" + ex);
    }
}
```

メッセージのグループ化

1 つのキューに属しているメッセージをグループ化して 1 つのセットにし、一度に 1 ユーザーしか使用できないようにできます。そのためには、トランザクション処理でのメッセージのグループ化に対応したキュー表に、そのキューを作成する必要があります（9-4 ページの「キュー表の作成」を参照）。1 つのグループに属するメッセージは、すべて同一のトランザクションで作成される必要があります。また、1 つのトランザクションで作成されるメッセージは、すべて同一のグループに属します。この機能によって、複雑なメッセージを、複数の単純なメッセージにセグメント化できます。

たとえば、あるキュー宛てのメッセージに請求書情報が含まれている場合、そのメッセージは、ヘッダーのメッセージ、詳細情報のメッセージ、フッターのメッセージで構成されるグループとして作成できます。小さいオブジェクトにセグメント化できるイメージやビデオなどの複合ラージ・オブジェクトがメッセージ・ペイロードにある場合は、メッセージのグループ化が有効です。

グループに含まれるメッセージの一般的なメッセージ・プロパティ（優先順位、遅延、期限切れ）は、単にグループの最初のメッセージ（ヘッダー）のプロパティによってのみ判断され、グループの他のメッセージのプロパティは無視されます。

グループ化メッセージのプロパティは、伝播されても保持されます。ただし、メッセージが伝播される宛先キューも、トランザクション処理でグループ化可能であることが必要な点に注意してください。トランザクション処理でグループ化可能なキューからメッセージをデキューするときに、グループ化メッセージのプロパティを保持する場合、他にも認識しておく必要がある制限があります（詳細は、8-56 ページの「デキューの方法」および 8-67 ページの「デキューのモード」を参照）。

シナリオ

BooksOnLine アプリケーションでは、メッセージのグループ化は新規の注文を扱うために使用できます。各注文には、注文された多数の書籍が 1 つずつ連続して入っています。Web を経由して注文された項目も同様です。

次の例では、各エンキューが注文の中の個々の書籍に対応し、グループ / トランザクションが 1 つの完全な注文を表します。最初のエンキューにのみ、顧客情報が含まれています。`OE_neworders_que` は、グループをトランザクション処理化できる表 `OE_orders_sqtab` に格納されることに注意してください。プロシージャ `new_order_enq()` および `same_order_enq()` の説明は、サンプル・コードを参照してください。

注意： キュー名およびキュー表名は、大文字に変換されます。大文字と小文字が混在しているキュー名およびキュー表名はサポートされません。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
connect OE/OE;

/* Create queue table for OE:  */
EXECUTE dbms_aqadm.create_queue_table( \
        queue_table        => 'OE_orders_sqtab',\
        comment            => 'Order Entry Single-Consumer Orders queue table',\
        queue_payload_type => 'BOLADM.order_typ',\
        message_grouping   => DBMS_AQADM.TRANSACTIONAL, \
        compatible         => '8.1',  \
        primary_instance   => 1,\
        secondary_instance => 2);

/* Create neworders queue for OE: */
EXECUTE dbms_aqadm.create_queue ( \
        queue_name         => 'OE_neworders_que',
        queue_table        => 'OE_orders_sqtab');

/* Login into OE account :*/
CONNECT OE/OE;
SET serveroutput on;
/* Enqueue some orders using message grouping into OE_neworders_que,
   First Order Group: */
EXECUTE BOLADM.new_order_enq('My First   Book', 1, 1001, 'CA');
EXECUTE BOLADM.same_order_enq('My Second  Book', 2);
COMMIT;
/
/* Second Order Group: */
EXECUTE BOLADM.new_order_enq('My Third   Book', 1, 1002, 'WA');
COMMIT;
/
/* Third Order Group: */
EXECUTE BOLADM.new_order_enq('My Fourth  Book', 1, 1003, 'NV');
EXECUTE BOLADM.same_order_enq('My Fifth   Book', 3);
EXECUTE BOLADM.same_order_enq('My Sixth   Book', 2);
COMMIT;
/
/* Fourth Order Group: */
EXECUTE BOLADM.new_order_enq('My Seventh Book', 1, 1004, 'MA');
EXECUTE BOLADM.same_order_enq('My Eighth  Book', 3);
EXECUTE BOLADM.same_order_enq('My Ninth   Book', 2);
COMMIT;
/
```

Visual Basic（OO4O）：サンプル・コード

この機能は、現在使用できません。

Java（JDBC）：サンプル・コード

```
public static void createMsgGroupQueueTable(Connection db_conn)
{
    AQSession            aq_sess;
    AQQueueTableProperty sqt_prop;
    AQQueueTable         sq_table;
    AQQueueProperty      q_prop;
    AQQueue              neworders_q;

    try
    {

        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        /* Create a single-consumer orders queue table */
        sqt_prop = new AQQueueTableProperty("BOLADM.order_typ");
        sqt_prop.setComment("Order Entry Single-Consumer Orders queue table");
        sqt_prop.setCompatible("8.1");
        sqt_prop.setMessageGrouping(AQQueueTableProperty.TRANSACTIONAL);

        sq_table = aq_sess.createQueueTable("OE", "OE_orders_sqtab", sqt_prop);

        /* Create new orders queue for OE */
        q_prop = new AQQueueProperty();

        neworders_q = aq_sess.createQueue(sq_table, "OE_neworders_que",
                                          q_prop);

    }
    catch (AQException ex)
    {
        System.out.println("AQ Exception: " + ex);
    }
}
```

エンキュー中のメッセージ変換

8-5 ページの「メッセージ・フォーマットの変換」のシナリオで説明したとおり、注文入力アプリケーションと出荷処理アプリケーションでは、注文項目の表現が異なります。注文入力アプリケーションでは、ユーザー定義型の OE.order_typ フォーマットで注文項目が表現されます。西部向け出荷処理アプリケーションでは、ユーザー定義型の WS.order_typ_sh フォーマットで注文項目が表現されます。したがって、OE スキーマ内のキューは、OE.orders_typ のペイロード型で、WS スキーマ内のキューは、WS.orders_typ_sh のペイロード型です。

メッセージ変換は、エンキュー中に使用できます。これは、エンキュー中のメッセージの検証および変換に特に有効です。アプリケーションでは、アプリケーション独自のデータ・モデルに基づいてメッセージを生成できます。メッセージは、変換マッピングを使用してエンキューする前に、キューのデータ型に変換できます。

シナリオ

エンキュー時に、アプリケーションは、OE_booked_orders_topic からメッセージを伝播するかわりに注文をデキューし、そのデキューされた注文が西部向け出荷である場合は、そのデキューされた注文を WS_booked_orders_topic にパブリッシュすると想定します。

PL/SQL（DBMS_AQ）：サンプル・コード

アプリケーションは、次のとおりエンキュー時に変換を使用できます。

```
CREATE OR REPLACE FUNCTION
  fwd_message_to_ws_shipping(booked_order  OE.order_typ)
   RETURNS boolean AS

 enq_opt   dbms_aq.enqueue_options_t;
 msg_prp   dbms_aq.message_properties_t;
BEGIN

 IF (booked_order.order_region = 'WESTERN' and
     booked_order.order_type != 'RUSH') THEN
   enq_opt.transformation := 'OE.OE2WS';
   msg_prp.recipient_list(0) := aq$_agent('West_shipping', null, null);

   dbms_aq.enqueue('WS.ws_bookedorders_topic',
                   enq_opt, msg_prp, booked_order);

   RETURN true;
 ELSE
    RETURN false;
 END IF;
END;
```

Visual Basic（OO4O）：サンプル・コード

今回のリリースでは、例は記載していません。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

AQ XML サーブレットを使用したエンキュー

iDAP を使用して、インターネット経由でエンキュー・リクエストを実行できます。iDAP を使用した AQ リクエストの送信の詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

シナリオ

BooksOnLine アプリケーションでは、顧客は次の方法を要求できます。

- FedEx による出荷（優先順位 1）
- 優先扱い航空便による出荷（優先順位 2）
- 通常地上便による出荷（優先順位 3）

注文入力アプリケーションは、入力済注文を FIFO 優先順位のキューに格納します。入力済注文は、地区の入力済注文キューに伝播されます。各地区では、その地区の入力済注文キューにある注文が、出荷の優先順位に従って処理されます。

次のコールで、注文入力アプリケーションに優先順位キューを作成します。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/* Create a priority queue table for OE: */
EXECUTE dbms_aqadm.create_queue_table( \
   queue_table         => 'OE_orders_pr_mqtab', \
   sort_list           =>'priority,enq_time', \
   comment             => 'Order Entry Priority  \
                          MultiConsumer Orders queue table',\
   multiple_consumers  => TRUE, \
   queue_payload_type  => 'BOLADM.order_typ', \
   compatible          => '8.1', \
   primary_instance    => 2, \
   secondary_instance  => 1);

EXECUTE dbms_aqadm.create_queue ( \
   queue_name          => 'OE_bookedorders_que', \
   queue_table         => 'OE_orders_pr_mqtab');
```

顧客 John が、SOAP を使用してエンキュー・リクエストを送信すると想定します。XML メッセージのフォーマットは、次のとおりです。

```
<?xml version="1.0"?>
   <Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
        <AQXmlSend xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <producer_options>
            <destination>OE.OE_bookedorders_que</destination>
          </producer_options>
```

```
          <message_set>
            <message_count>1</message_count>

            <message>
              <message_number>1</message_number>
              <message_header>
                <correlation>ORDER1</correlation>
<priority>1</priority>
 <sender_id>
   <agent_name>john</agent_name>
         </sender_id>
              </message_header>

              <message_payload>

               <ORDER_TYP>
                     <ORDERNO>100</ORDERNO>
                     <STATUS>NEW</STATUS>
                     <ORDERTYPE>URGENT</ORDERTYPE>
                     <ORDERREGION>EAST</ORDERREGION>
                     <CUSTOMER>
                        <CUSTNO>1001233</CUSTNO>
                        <CUSTID>JOHN</CUSTID>
                        <NAME>JOHN DASH</NAME>
                        <STREET>100 EXPRESS STREET</STREET>
                        <CITY>REDWOOD CITY</CITY>
                        <STATE>CA</STATE>
                        <ZIP>94065</ZIP>
                        <COUNTRY>USA</COUNTRY>
                     </CUSTOMER>
                     <PAYMENTMETHOD>CREDIT</PAYMENTMETHOD>
                     <ITEMS>
                        <ITEMS_ITEM>
                           <QUANTITY>10</QUANTITY>
                           <ITEM>
                              <TITLE>Perl handbook</TITLE>
                              <AUTHORS>Randal</AUTHORS>
                              <ISBN>345620200</ISBN>
                              <PRICE>19</PRICE>
                           </ITEM>
                           <SUBTOTAL>190</SUBTOTAL>
                        </ITEMS_ITEM>
                        <ITEMS_ITEM>
                           <QUANTITY>10</QUANTITY>
                           <ITEM>
                              <TITLE>JDBC guide</TITLE>
```

```
                                    <AUTHORS>Taylor</AUTHORS>
                                    <ISBN>123420212</ISBN>
                                    <PRICE>59</PRICE>
                                  </ITEM>
                                  <SUBTOTAL>590</SUBTOTAL>
                                </ITEMS_ITEM>
                              </ITEMS>
                              <CCNUMBER>NUMBER01</CCNUMBER>
                              <ORDER_DATE>08/23/2000 12:45:00</ORDER_DATE>
                            </ORDER_TYP>
                          </message_payload>
                        </message>
                      </message_set>

                        <AQXmlCommit/>
                      </AQXmlSend>
                    </Body>
</Envelope>
```

デキュー機能

シングル・コンシューマ・キューからデキューするプロセスまたはマルチ・コンシューマ・キューから同じコンシューマでデキューするプロセスが複数ある場合、同時プロセスで処理中のメッセージは別のプロセスではスキップされます。このため、複数のプロセスが同一コンシューマの異なるメッセージを同時に処理できます。

この項の内容は次のとおりです。

- デキューの方法
- 複数の受信者
- ローカルおよびリモートの受信者
- デキューにおけるメッセージ・ナビゲーション
- デキューのモード
- メッセージ到着待機の最適化
- 遅延間隔をおいた後の再試行
- 例外処理
- ルールベースのサブスクリプション
- Listen 機能
- デキュー中のメッセージ変換
- AQ XML サーブレットを使用したデキュー

デキューの方法

次のいずれかの方法で、メッセージをデキューできます。

- 相関識別子
- メッセージ識別子
- デキューの条件
- デフォルトのデキュー

相関識別子はユーザー定義のメッセージ・プロパティ（VARCHAR2 データ型）で、メッセージ識別子はシステムによって割り当てられる値（RAW データ型）です。1 つのキューに同じ相関識別子を持つメッセージが複数存在することがありますが、同じメッセージ識別子を持つメッセージが複数存在することはありません。同じ相関識別子を持つメッセージが複数ある場合、メッセージ同士の順序付け（エンキュー順序）がデキュー・コールでは変更されることがあります。一連のデキュー・コールの間は、first message ナビゲーション・オプションを指定しないと相関識別子を変更できません。

デキュー条件は、SQL 問合せの WHERE 句の構文に似た式です。デキュー条件は、メッセージ・プロパティまたはメッセージ内容を表すペイロードの属性を使用して表現されます。キュー内のメッセージは、条件に対して評価され、指定した条件を満たすメッセージが戻されます。

デフォルトのデキューとは、マルチ・コンシューマ・キューのコンシューマで使用可能な最初のメッセージまたはシングル・コンシューマ・キューで使用可能な最初のメッセージがデキューされることを意味します。

また、相関識別子、メッセージ識別子またはデキュー条件を使用してメッセージをデキューすると、グループ化メッセージのプロパティが変更されることに注意してください（詳細は、8-49 ページの「メッセージのグループ化」および 8-63 ページの「デキューにおけるメッセージ・ナビゲーション」を参照）。

シナリオ

BooksOnLine の例では、東部向け出荷サイトに届いた至急（RUSH）注文が、最初に処理されます。そのために、注文タイプ（至急 / 普通）を含めて定義されている相関識別子を使用してメッセージをデキューします。メッセージ識別子を使用したデキューの詳細は、8-67 ページの「デキューのモード」の例で説明した get_northamerican_orders プロシージャを参照してください。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CONNECT boladm/boladm;

/*  Create procedures to dequeue RUSH orders */
create or replace procedure get_rushtitles(consumer in varchar2) as

deq_cust_data            BOLADM.customer_typ;
deq_book_data            BOLADM.book_typ;
deq_item_data            BOLADM.orderitem_typ;
deq_msgid                RAW(16);
dopt                     dbms_aq.dequeue_options_t;
mprop                    dbms_aq.message_properties_t;
deq_order_data           BOLADM.order_typ;
qname                    varchar2(30);
no_messages              exception;
pragma exception_init    (no_messages, -25228);
new_orders               BOOLEAN := TRUE;

begin

        dopt.consumer_name := consumer;
        dopt.wait := 1;
        dopt.correlation := 'RUSH';

        IF (consumer = 'West_Shipping') THEN
                qname := 'WS.WS_bookedorders_que';
        ELSIF (consumer = 'East_Shipping') THEN
                qname := 'ES.ES_bookedorders_que';
        ELSE
                qname := 'OS.OS_bookedorders_que';
        END IF;

        WHILE (new_orders) LOOP
          BEGIN
            dbms_aq.dequeue(
                queue_name => qname,
                dequeue_options => dopt,
                message_properties => mprop,
                payload => deq_order_data,
                msgid => deq_msgid);
            commit;

            deq_item_data := deq_order_data.items(1);
            deq_book_data := deq_item_data.item;

            dbms_output.put_line(' rushorder book_title: ' ||
                                 deq_book_data.title ||
```

```
                    ' quantity: ' || deq_item_data.quantity);
        EXCEPTION
          WHEN no_messages THEN
               dbms_output.put_line (' ---- NO MORE RUSH TITLES ---- ');
               new_orders := FALSE;
        END;
      END LOOP;

end;
/

CONNECT EXECUTE on get_rushtitles to ES;

/* Dequeue the orders: */
CONNECT ES/ES;

/*  Dequeue all rush order titles for East_Shipping: */
EXECUTE BOLADM.get_rushtitles('East_Shipping');
```

Visual Basic（OO4O）：サンプル・コード

```
   set oraaq1 = OraDatabase.CreateAQ("WS.WS_backorders_que")
   set oraaq2 = OraDatabase.CreateAQ("ES.ES_backorders_que")
   set oraaq3 = OraDatabase.CreateAQ("CBADM.deferbilling_que")
   Set OraMsg = OraAq.AQMsg(ORATYPE_OBJECT, "BOLADM.order_typ")
   Set OraBackOrder = OraDatabase.CreateOraObject("BOLADM.order_typ")

Private Sub Requeue_backorder
   Dim q as oraobject
   If sale_region = WEST then
      q = oraaq1
   else if sale_region = EAST then
      q = oraaq2
   else
      q = oraaq3
   end if

   OraMsg.delay = 7*60*60*24
   OraMsg = OraBackOrder 'OraOrder contains the order details
   Msgid = q.enqueue

End Sub
```

Java（JDBC）：サンプル・コード

```
public static void getRushTitles(Connection db_conn, String consumer)
{
    AQSession         aq_sess;
    Order             deq_order;
    byte[]            deq_msgid;
    AQDequeueOption   deq_option;
    AQMessageProperty msg_prop;
    AQQueue           bookedorders_q;
    AQMessage         message;
    AQObjectPayload   obj_payload;
    boolean           new_orders = true;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        deq_option = new AQDequeueOption();

        deq_option.setConsumerName(consumer);
        deq_option.setWaitTime(1);
        deq_option.setCorrelation("RUSH");

        if(consumer.equals("West_Shipping"))
        {
            bookedorders_q = aq_sess.getQueue("WS", "WS_bookedorders_que");
        }
        else if(consumer.equals("East_Shipping"))
        {
            bookedorders_q = aq_sess.getQueue("ES", "ES_bookedorders_que");
        }
        else
        {
            bookedorders_q = aq_sess.getQueue("OS", "OS_bookedorders_que");
        }

        while(new_orders)
        {
            try
            {
              /* Dequeue the message */
              message = bookedorders_q.dequeue(deq_option, Order.getFactory());

              obj_payload = message.getObjectPayload();

              deq_order = (Order)(obj_payload.getPayloadData());
```

```
                System.out.println("Order number " + deq_order.getOrderno() +
                                   " is a rush order");

              }
              catch (AQException aqex)
              {
                new_orders = false;
                System.out.println("No more rush titles");
                System.out.println("Exception-1: " + aqex);
              }
          }
      }
      catch (Exception ex)
      {
          System.out.println("Exception-2: " + ex);
      }

}
```

複数の受信者

コンシューマは、DBMS_AQADM.ADD_SUBSCRIBER プロシージャの AQ$_AGENT 型で使用された名前、またはメッセージ・プロパティの受信者リストを指定することで、マルチ・コンシューマの通常キューからメッセージをデキューできます（9-59 ページの「サブスクライバの追加」または 11-10 ページの「メッセージのエンキュー（メッセージ・プロパティの指定）」を参照）。

- PL/SQL では、コンシューマ名は dequeue_options_t レコードの consumer_name フィールドを使用して提供されます。
- OCI では、コンシューマ名は、LNOCISetAttr プロシージャを使用し、テキスト文字列を LNOCI_DTYPE_AQDEQ_OPTIONS 記述子の LNOCI_ATTR_CONSUMER_NAME として指定して提供されます。
- OO4O では、コンシューマ名は OraAQ オブジェクトのコンシューマ・プロパティを設定することで提供されます。

複数のプロセスまたはスレッドが、1 つのキューから同じ consumer_name を使用して同時にデキューできます。そのような場合、AQ はキューの先頭にあるロックされていない最初のメッセージをコンシューマに提供します。デキュー中に特定のメッセージのメッセージ ID が指定されないかぎり、コンシューマは READY 状態のメッセージをデキューできます。

メッセージが PROCESSED（処理済）とみなされるのは、所定のコンシューマ全員がメッセージのデキューに成功したときのみです。メッセージが EXPIRED（期限切れ）とみなされるのは、1 つまたは複数のコンシューマが EXPIRATION 時刻までにそのメッセージをデキューしなかったときです。期限切れになったメッセージは、例外キューに移されます。

例外キューも必ずマルチ・コンシューマ・キューです。マルチ・コンシューマ・キューから移された期限切れメッセージは、受信者を指定してデキューできません。ただし、デキュー・オプションのコンシューマ名に NULL を指定することによって、REMOVE モードで 1 回のみデキューできます。したがって、デキューの観点からすると、マルチ・コンシューマの例外キューはシングル・コンシューマ・キューのように動作します。これは、それぞれの期限切れメッセージは、NULL コンシューマ名を使用して 1 回しかデキューできないためです。マルチ・コンシューマ例外キューが、互換パラメータを「8.0」に設定してキュー表に作成された場合は、期限切れメッセージはメッセージ ID を指定する方法でしかデキューできないことに注意してください。

リリース 8.1.6 以上では、マルチ・コンシューマ・キューからメッセージを削除できるのはキュー・モニターのみです。これによって、デキュー元はキュー表のメッセージをロックすることなくデキュー操作を完了できます。キュー・モニターは、すべてのコンシューマが処理を完了したメッセージをマルチ・コンシューマ・キューから削除する作業を毎分約 1 回行うため、メッセージが完全に処理された後キューから物理的に削除されるまでの遅延があります。

ローカルおよびリモートの受信者

マルチ・コンシューマ・キュー内のメッセージのコンシューマ（そのキューのサブスクライバであるか、エンキュー元の受信者リストに含まれている受信者）は、ローカルでもリモートでもかまいません。

- ローカル・コンシューマは、プロデューサがそのメッセージをエンキューしたキューからデキューします。ローカル・コンシューマの場合は、AQ$_AGENT 型の NAME は NULL 以外で、ADDRESS および PROTOCOL フィールドは NULL です（2-3 ページの「エージェント型（aq$_agent）」を参照）。
- リモート・コンシューマは、メッセージがエンキューされたものとは別のキューからデキューします。ユーザーは、このリモート・コンシューマの特徴を理解し、AQ 伝播機能を使用してリモート・コンシューマを使用する必要があります。リモート・コンシューマは次の 3 つのカテゴリに分類されます。

  a. ADDRESS フィールドが同一データベース内のキューを参照しているもの。この場合、コンシューマは同一データベースの別のキューからメッセージをデキューすることになります。アドレスの形式は [schema].queue_name で、queue_name（オプションで、スキーマ名によって修飾されることもある）がターゲット・キューです。スキーマが指定されない場合は、ADD_SUBSCRIBER プロシージャを実行している現行のユーザーのスキーマまたはエンキューされたスキーマが使用されます（9-59 ページの「サブスクライバの追加」または 11-4 ページの「メッセージのエンキュー」を参照）。このようなリモート・コンシューマへの伝播のスケジューリングには、宛先に NULL（デフォルト値）を指定した DBMS_AQADM.SCHEDULE_PROPAGATION コマンドを使用します（9-72 ページの「キューの伝播のスケジューリング」を参照）。

b. ADDRESS フィールドが別のデータベース内のキューを参照しているもの。この場合の別のデータベースは、データベース・リンクによって到達可能で、PROTOCOL が NULL または 0（ゼロ）である必要があります。アドレスの形式は、[schema].queue_name@dblink になります。スキーマが指定されないと、ADD_SUBSCRIBER プロシージャを実行している現行のユーザーのスキーマまたはエンキューされたスキーマが使用されます。データベース・リンクが完全な修飾名でない場合（ドメイン名が指定されない場合）、db_domaininit.ora パラメータで指定されたデフォルトのドメインが使用されます。データベース・リンクを宛先とした DBMS_AQADM.SCHEDULE_PROPAGATION プロシージャを使用して伝播をスケジューリングします。AQ は、シノニムを使用してキューまたはデータベース・リンクを参照する方法はサポートしていません。

c. ADDRESS フィールドがサード・パーティのプロトコルによって到達できる宛先を参照しているもの。サード・パーティ・ソフトウェアのドキュメントを参照してデータベース・リンクの ADDRESS および PROTOCOL をどのように指定するか、またどのように伝播をスケジューリングするかを決定する必要があります。

コンシューマがリモートの場合、伝播されたメッセージがリモート・キューからデキューされていないにもかかわらず、ソース・キューでは、伝播された直後に PROCESSED とマークされます。同様に、伝播されたメッセージがリモート・キューで期限切れになると、そのメッセージはローカル・キューの例外キューではなく、リモート・キューのキュー表の DEFAULT 例外キューに移されます。この 2 つのケースからわかるように、現状では、AQ は例外をソース・キューに伝播しません。キュー表のビュー（AQ$<queue_table>）の MSGID 列および ORIGINAL_MSGID 列を使用して、伝播されたメッセージを連鎖させることができます。メッセージ ID が m1 のメッセージがリモート・キューに伝播されると、m1 はリモート・キューの ORIGINAL_MSGID 列に格納されます。

DELAY、EXPIRATION および PRIORITY パラメータは、ローカル・コンシューマにもリモート・コンシューマにも同様に適用されます。AQ は伝播による遅延を見込んで、DELAY および EXPIRATION パラメータを調整します。たとえば、EXPIRATION が 1 時間で、メッセージが 15 分後に伝播された場合、リモート・キューでの期限切れは 45 分に設定されます。

データベースがメッセージ伝播を処理するため、OO4O はリモートおよびローカルの受信者を区別しません。メッセージをデキューするには、ローカルおよびリモートの受信者に対して、コール / ステップの順序が同じである必要があります。

デキューにおけるメッセージ・ナビゲーション

キューからメッセージを選択するには、いくつかのオプションがあります。first message（最初のメッセージ）を選択できます。または、メッセージを選択してキュー内での位置を（たとえば 4 番目のメッセージとして）確立してから、next message（次のメッセージ）を取り出せます。

最初のメッセージ・ナビゲーションでは、キューに SELECT が実行されます。次のメッセージ・ナビゲーションでは、最初のメッセージ・ナビゲーションで実行された SELECT の結果からフェッチが実行されます。このように、後続のデキューでは SELECT 全体を再度実行する必要がないため、パフォーマンスが最適化されます。

そのキューがトランザクション処理でグループ化できる場合は、前述の選択は少し違ったものになります。

- first message が要求されると、デキュー位置はキューの最初にリセットされます。
- next message が要求されると、位置は同一トランザクションの次のメッセージに設定されます。
- next transaction（次のトランザクション）が要求されると、位置は次のトランザクションの最初のメッセージに設定されます。

相関識別子を指定してデキューするか、メッセージ識別子を指定してデキューするか、または任意のトランザクション・メッセージをデキュー後にコミットすると、グループ化トランザクションのプロパティは無効になることに注意してください（8-56 ページの「デキューの方法」を参照）。

next message または next transaction オプションを使用したプログラムがキューの中をナビゲートしてキューの最後に到達し、待機時間を 0（ゼロ）以外に指定していた場合、ナビゲート位置は自動的にそのキューの先頭に変更されます。待機時間を 0（ゼロ）に指定した場合、キューの最後に到達すると例外が発生することがあります。

シナリオ

次の BooksOnLine のシナリオでは、エンキューに関連してすでに説明したメッセージのグループ化の例をさらに続けます（8-56 ページの「デキューの方法」を参照）。

get_orders() プロシージャは、OE_neworders_que から注文をデキューします。トランザクションごとに注文が参照され、その注文に含まれる個々の書籍に、各メッセージが対応します。get_orders() プロシージャは、メッセージをループして書籍注文をデキューします。最初のデキューの前に、first message オプションでキューの先頭にリセットします。それから、next message ナビゲーション・オプションで、注文（トランザクション）から次の書籍（メッセージ）を取り出します。現行のグループ / トランザクションのすべてのメッセージがフェッチされたというエラー・メッセージが戻された場合は、ナビゲーション・オプションを next transaction に変更して、次の注文の最初の書籍を取り出します。次に、再度 next message オプションに戻して、同一トランザクションの次のメッセージをフェッチします。すべての注文（トランザクション）がフェッチされるまで、この処理を繰り返します。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CONNECT boladm/boladm;

create or replace procedure get_new_orders as

deq_cust_data            BOLADM.customer_typ;
deq_book_data            BOLADM.book_typ;
deq_item_data            BOLADM.orderitem_typ;
deq_msgid                RAW(16);
dopt                     dbms_aq.dequeue_options_t;
mprop                    dbms_aq.message_properties_t;
deq_order_data           BOLADM.order_typ;
qname                    VARCHAR2(30);
no_messages              exception;
end_of_group             exception;
pragma exception_init    (no_messages, -25228);
pragma exception_init    (end_of_group, -25235);
new_orders               BOOLEAN := TRUE;

BEGIN

        dopt.wait := 1;
        dopt.navigation := DBMS_AQ.FIRST_MESSAGE;
        qname := 'OE.OE_neworders_que';
        WHILE (new_orders) LOOP
          BEGIN
            LOOP
                BEGIN
                    dbms_aq.dequeue(
                        queue_name          => qname,
                        dequeue_options     => dopt,
                        message_properties  => mprop,
                        payload             => deq_order_data,
                        msgid               => deq_msgid);

                    deq_item_data := deq_order_data.items(1);
                    deq_book_data := deq_item_data.item;
                    deq_cust_data := deq_order_data.customer;

                    IF (deq_cust_data IS NOT NULL) THEN
                      dbms_output.put_line(' **** NEXT ORDER **** ');
                      dbms_output.put_line('order_num: ' ||
                                deq_order_data.orderno);
                      dbms_output.put_line('ship_state: ' ||
                                deq_cust_data.state);
                    END IF;
                    dbms_output.put_line(' ---- next book ---- ');
```

```
                    dbms_output.put_line(' book_title: ' ||
                                deq_book_data.title ||
                                ' quantity: ' || deq_item_data.quantity);
                EXCEPTION
                    WHEN end_of_group THEN
                      dbms_output.put_line ('*** END OF ORDER ***');
                      commit;
                      dopt.navigation := DBMS_AQ.NEXT_TRANSACTION;
                END;
              END LOOP;
            EXCEPTION
              WHEN no_messages THEN
                   dbms_output.put_line (' ---- NO MORE NEW ORDERS ---- ');
                   new_orders := FALSE;
            END;
          END LOOP;

END;
/

CONNECT EXECUTE ON get_new_orders to OE;

/*  Dequeue the orders: */
CONNECT OE/OE;
EXECUTE BOLADM.get_new_orders;
```

Visual Basic（OO4O）：サンプル・コード

```
Dim OraSession as object
Dim OraDatabase as object
Dim OraAq as object
Dim OraMsg as Object
Dim OraOrder,OraItemList,OraItem,OraBook,OraCustomer  as Object
Dim Msgid as String

   Set OraSession = CreateObject("OracleInProcServer.XOraSession")
   Set OraDatabase = OraSession.DbOpenDatabase("", "boladm/boladm", 0&)
   set oraaq = OraDatabase.CreateAQ("OE.OE_neworders_que")
   Set OraMsg = OraAq.AQMsg(ORATYPE_OBJECT, "BOLADM.order_typ")
       OraAq.wait = 1
   OraAq.Navigation = ORAAQ_DQ_FIRST_MESSAGE

private sub get_new_orders
   Dim MsgIsDequeued as Boolean
   On Error goto ErrHandler
   MsgIsDequeued = TRUE
       msgid = q.Dequeue
```

```
        if MsgIsDequeued then
      set OraOrder = OraMsg
      OraItemList = OraOrder("items")
      OraItem = OraItemList(1)
      OraBook = OraItem("item")
      OraCustomer = OraOrder("customer")

         ' Populate the textboxes with the values
      if( OraCustomer ) then
         if OraAq.Navigation <> ORAAQ_DQ_NEXT_MESSAGE then
            MsgBox " ********* NEXT ORDER *******"
         end if
         txt_book_orderno = OraOrder("orderno")
         txt_book_shipstate = OraCustomer("state")
      End if
      OraAq.Navigation = ORAAQ_DQ_NEXT_MESSAGE
      txt_book_title = OraBook("title")
      txt_book_qty  = OraItem("quantity")
    Else
      MsgBox " ********* END OF  ORDER *******"
   End if

ErrHandler :
   'Handle error case, like no message etc
   If OraDatabase.LastServerErr = 25228 then
      OraAq.Navigation = ORAAQ_DQ_NEXT_TRANSACTION
      MsgIsDequeued = FALSE
      Resume Next
   End If
   'Process other errors
end sub
```

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

デキューのモード

デキュー・リクエストによって、メッセージを参照または削除できます（11-44 ページの「メッセージのデキュー」を参照）。

- メッセージを参照するときは、BROWSE（ブラウズ）モードまたは LOCKED（ロック）モードを使用します。
- メッセージを削除するときは、REMOVE（削除）モードまたは REMOVE WITH NO DATA（データ削除を伴わない）モードを使用します。

メッセージは、ブラウズ後でも処理可能です。同様に、ロック後でも、トランザクションのコミットまたはロールバックによってそのロックが解除されると、メッセージは処理可能です。どちらかの削除モードで一度メッセージが削除されると、デキュー・リクエストは使用できません。

REMOVE_NODATA モードでメッセージがデキューされた場合、メッセージのペイロードは取り出されません。このモードは、ユーザーが先に BROWSE モードでデキューしてペイロードを調べてあるときに有効です。このようにして、ペイロードの取出しというオーバーヘッドを回避できます。

キューに保存時間が指定されている場合、メッセージは削除された後でもキュー表に保存されます。例外キューのメッセージは保存できません（詳細は、9-82 ページの「例外処理」を参照）。一般に、データを伴わないメッセージの削除は、（あらかじめ BROWSE/LOCKED モードでデキューしていて）ペイロード内容がわかっていたり、メッセージが使用される見込みがないときに使用されます。

メッセージをブラウズした後は、そのメッセージを再度デキューできる保証がないことに注意してください。これは、同時ユーザーのデキュー・コールによってそのメッセージが削除される可能性があるためです。一度参照したメッセージが同時ユーザーによってデキューされないようにするには、LOCKED モードでメッセージを参照する必要があります。

通常、BROWSE モードを使用する場合は注意が必要です。待機時間が 0（ゼロ）以外に指定されていて、ナビゲート位置がキューの最後に到達した場合、デキュー位置は自動的にそのキューの先頭に変更されます。したがって、BROWSE モードで next message ナビゲーション・オプションおよび 0（ゼロ）以外の待機時間を指定してデキューを繰り返すと、何度も同一メッセージをデキューすることがあります。キューに対する最初のデキューに待機時間を 0（ゼロ）以外で指定して、それ以降のデキュー・コールには待機時間 0（ゼロ）の next message ナビゲーション・オプションを使用することをお薦めします。デキュー・コールで「end of queue」エラー・メッセージが戻された場合は、デキュー位置を first message ナビゲーション・オプションを使用して明示的にキューの先頭に設定できます。このようにすると、そのキューのメッセージを再度ブラウズできます。

シナリオ

次の BooksOnLine のシナリオでは、海外からの注文情報（メキシコおよびカナダ向け）が、通商政策および送料割引の理由で別々に処理されます。したがって、（他の同時ユーザーが削除できないように）メッセージをロック・モードで参照し、顧客の国（メッセージ・ペイロード）を確認します。顧客の国がメキシコまたはカナダの場合、（ペイロードはすでにわかっているため）REMOVE_NODATA でメッセージをキューから削除します。これを実行しないと、そのメッセージのロックはコミット・コールによって解除されるためです。削除モードのデキュー・コールには、ロック・モードのデキュー・コールによって取得したメッセージ識別子を使用することに注意してください。shipping_bookedorder_deq コールは、BROWSE モードの使用方法を示します（このプロシージャについては、サンプル・コードを参照）。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CONNECT boladm/boladm;

create or replace procedure get_northamerican_orders as

deq_cust_data             BOLADM.customer_typ;
deq_book_data             BOLADM.book_typ;
deq_item_data             BOLADM.orderitem_typ;
deq_msgid                 RAW(16);
dopt                      dbms_aq.dequeue_options_t;
mprop                     dbms_aq.message_properties_t;
deq_order_data            BOLADM.order_typ;
deq_order_nodata          BOLADM.order_typ;
qname                     VARCHAR2(30);
no_messages               exception;
pragma exception_init     (no_messages, -25228);
new_orders                BOOLEAN := TRUE;

begin

        dopt.consumer_name := consumer;
        dopt.wait := DBMS_AQ.NO_WAIT;
        dopt.navigation := dbms_aq.FIRST_MESSAGE;
        dopt.dequeue_mode := DBMS_AQ.LOCKED;

        qname := 'OS.OS_bookedorders_que';

        WHILE (new_orders) LOOP
          BEGIN
            dbms_aq.dequeue(
                queue_name => qname,
                dequeue_options => dopt,
                message_properties => mprop,
```

```
                payload => deq_order_data,
                msgid => deq_msgid);

            deq_item_data := deq_order_data.items(1);
            deq_book_data := deq_item_data.item;
            deq_cust_data := deq_order_data.customer;

            IF (deq_cust_data.country = 'Canada' OR
                deq_cust_data.country = 'Mexico' ) THEN

                dopt.dequeue_mode := dbms_aq.REMOVE_NODATA;
                dopt.msgid := deq_msgid;
                dbms_aq.dequeue(
                        queue_name => qname,
                        dequeue_options => dopt,
                        message_properties => mprop,
                        payload => deq_order_nodata,
                        msgid => deq_msgid);
                commit;

                dbms_output.put_line(' **** next booked order **** ');
                dbms_output.put_line('order_no: ' || deq_order_data.orderno ||
                        ' book_title: ' || deq_book_data.title ||
                        ' quantity: ' || deq_item_data.quantity);
                dbms_output.put_line('ship_state: ' || deq_cust_data.state ||
                        ' ship_country: ' || deq_cust_data.country ||
                        ' ship_order_type: ' || deq_order_data.ordertype);

            END IF;

            commit;
            dopt.dequeue_mode := DBMS_AQ.LOCKED;
            dopt.msgid := NULL;
            dopt.navigation := dbms_aq.NEXT_MESSAGE;
          EXCEPTION
            WHEN no_messages THEN
                 dbms_output.put_line (' ---- NO MORE BOOKED ORDERS ---- ');
                 new_orders := FALSE;
          END;
        END LOOP;

end;
/

CONNECT EXECUTE on get_northamerican_orders to OS;

CONNECT ES/ES;
```

```
/*  Browse all booked orders for East_Shipping: */
EXECUTE BOLADM.shipping_bookedorder_deq('East_Shipping', DBMS_AQ.BROWSE);

CONNECT OS/OS;

/*  Dequeue all international North American orders for Overseas_Shipping: */
EXECUTE BOLADM.get_northamerican_orders;
```

Visual Basic（OO4O）：サンプル・コード

OO4O は、前述したデキューのすべてのモードをサポートします。可能な値は次のとおりです。

- ORAAQ_DQ_BROWSE (1) - デキュー時にロックしません。
- ORAAQ_DQ_LOCKED (2) - メッセージを読み込んで、書込みロックを取得します。
- ORAAQ_DQ_REMOVE (3)（デフォルト）- メッセージを読み込んで、更新または削除します。

```
Dim OraSession as object
Dim OraDatabase as object
Dim OraAq as object
Dim OraMsg as Object
Dim OraOrder,OraItemList,OraItem,OraBook,OraCustomer  as Object
Dim Msgid as String

   Set OraSession = CreateObject("OracleInProcServer.XOraSession")
   Set OraDatabase = OraSession.DbOpenDatabase("", "boladm/boladm", 0&)
   set oraaq = OraDatabase.CreateAQ("OE.OE_neworders_que")
       OraAq.DequeueMode = ORAAQ_DQ_BROWSE
```

Java（JDBC）：サンプル・コード

```
public static void get_northamerican_orders(Connection db_conn)
{

    AQSession         aq_sess;
    Order             deq_order;
    Customer          deq_cust;
    String            cust_country;
    byte[]            deq_msgid;
    AQDequeueOption   deq_option;
    AQMessageProperty msg_prop;
    AQQueue           bookedorders_q;
    AQMessage         message;
    AQObjectPayload   obj_payload;
```

```
boolean           new_orders = true;

try
{
    /* Create an AQ Session: */
    aq_sess = AQDriverManager.createAQSession(db_conn);

    deq_option = new AQDequeueOption();

    deq_option.setConsumerName("Overseas_Shipping");
    deq_option.setWaitTime(AQDequeueOption.WAIT_NONE);
    deq_option.setNavigationMode(AQDequeueOption.NAVIGATION_FIRST_MESSAGE);
    deq_option.setDequeueMode(AQDequeueOption.DEQUEUE_LOCKED);

    bookedorders_q = aq_sess.getQueue("OS", "OS_bookedorders_que");

    while(new_orders)
    {
        try
        {
          /* Dequeue the message - browse with lock */
          message = bookedorders_q.dequeue(deq_option, Order.getFactory());

          obj_payload = message.getObjectPayload();

          deq_msgid = message.getMessageId();
          deq_order = (Order)(obj_payload.getPayloadData());

          deq_cust = deq_order.getCustomer();

          cust_country = deq_cust.getCountry();

          if(cust_country.equals("Canada") ||
             cust_country.equals("Mexico"))
          {
            deq_option.setDequeueMode(
                              AQDequeueOption.DEQUEUE_REMOVE_NODATA);
            deq_option.setMessageId(deq_msgid);


            /* Delete the message */
            bookedorders_q.dequeue(deq_option, Order.getFactory());

            System.out.println("----  next booked order ------");
            System.out.println("Order no: " + deq_order.getOrderno());
            System.out.println("Ship state: " + deq_cust.getState());
            System.out.println("Ship country: " + deq_cust.getCountry());
```

```
                System.out.println("Order type: " + deq_order.getOrdertype());

              }

              db_conn.commit();

              deq_option.setDequeueMode(AQDequeueOption.DEQUEUE_LOCKED);
              deq_option.setMessageId(null);
              deq_option.setNavigationMode(
                                  AQDequeueOption.NAVIGATION_NEXT_MESSAGE);
            }
            catch (AQException aqex)
            {
              new_orders = false;
              System.out.println("--- No more booked orders ----");
              System.out.println("Exception-1: " + aqex);
            }
        }

    }
    catch (Exception ex)
    {
        System.out.println("Exception-2: " + ex);
    }

}
```

メッセージ到着待機の最適化

AQ では、新しくエンキューされるメッセージまたは READY 状態になるメッセージのどちらかに、1 つ以上のキューでアプリケーションを待機させることができます。DEQUEUE 操作によって、キューへのメッセージ到着を待機でき（11-44 ページの「メッセージのデキュー」を参照）、LISTEN 操作によって複数のキューへのメッセージ到着を待機できます（11-24 ページの「1 個以上のシングル・コンシューマ・キューのリスニング」を参照）。

ブロック（待機）している DEQUEUE コールが戻るとき、メッセージ・プロパティおよびメッセージ・ペイロードが戻されます。それに対して、ブロックしている LISTEN コールが戻るときは、メッセージが届いたキューの名前のみが戻ります。メッセージをデキューするためには、その後に DEQUEUE 操作をする必要があります。

アプリケーションは、必要に応じて AQ のメッセージ到着待機時間を示すタイムアウトを 0（ゼロ）または任意の秒数に指定できます。デフォルトでは、そのキューにメッセージが到着するまで、待機することになっています。この最適化は、2 つの点で重要です。まず、アプリケーションからメッセージをポーリングし続けるという負担がなくなります。また、アプリケーションは、新しいメッセージがエンキューされるか、DELAY 時間が過ぎて READY 状態になるまでブロックし続けるため、CPU およびネットワーク・リソースの節約になり

ます。アプリケーションは例外キューのブロッキング・デキューを実行して、EXPIRED メッセージを待機することもできます。

デキューによってブロックされたプロセスまたはスレッドは、新しいメッセージに DELAY が指定されていない場合はエンキュー元が直接アクティブにし、DELAY または EXPIRATION 時間が経過した場合はキュー・モニター・プロセスがアクティブにします。アプリケーションは、エンキュー元がエンキューするキューに届くメッセージを待機する他に、DBMS_AQADM.SCHEDULE_PROPAGATION による伝播がスケジューリングされている場合にはリモート・キューに届くメッセージも待機できます。この場合、AQ のプロパゲータは、メッセージの伝播後に、ブロックされたデキュー元をアクティブにします。

シナリオ

BooksOnLine の例では、8-56 ページの「デキューの方法」で説明した get_rushtitles プロシージャがデキュー・コールの引数 dequeue_options の待機時間を 1 秒に指定しています。待機時間は、次のサンプル・コードに示すとおり、別の方法でも指定できます。

- 待機時間を 10 秒に指定すると、そのキューのメッセージが使用可能になるまで、デキュー・コールは 10 秒のタイムアウト指定でブロックされます。つまり、10 秒が過ぎてもそのキューにメッセージがない場合は、デキュー・コールはメッセージなしで戻ります。事前定義された定数をこの待機時間に割り当てることもできます。
- 待機時間を DBMS_AQ.NO_WAIT に指定すると、待機時間には 0（ゼロ）秒が設定されます。この場合、そのキューにメッセージがない場合でも、デキュー・コールはすぐに戻ります。
- 待機時間を DBMS_AQ.FOREVER に指定すると、そのキューのメッセージが使用可能になるまで、デキュー・コールはタイムアウト指定なしでブロックされます。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/* dopt is a variable of type dbms_aq.dequeue_options_t.
   Set the dequeue wait time to 10 seconds: */
dopt.wait := 10;

/* Set the dequeue wait time to 0 seconds: */
dopt.wait := DBMS_AQ.NO_WAIT;

/* Set the dequeue wait time to infinite (forever): */
dopt.wait := DBMS_AQ.FOREVER;
```

Visual Basic（OO4O）：サンプル・コード

OO4O は、メッセージの非同期デキューをサポートします。まず、モニターが特定のキューに対して開始されます。ユーザー基準に合うメッセージがデキューされると、ユーザーのコールバック・オブジェクトが通知されます。

Java（JDBC）：サンプル・コード

```
AQDequeueOption deq-opt;

deq-opt = new AQDequeueOption ();
```

遅延間隔をおいた後の再試行

キューからメッセージをデキューするトランザクションが失敗した場合、メッセージ削除が正常に行われなかったと考えられます。AQ は、メッセージ削除に失敗した数をメッセージ履歴に記録します。アプリケーションは、キュー表ビューの RETRY_COUNT 列を問い合せて、メッセージに対する試行の失敗回数を参照できます。さらに、AQ では、アプリケーションがキュー内のメッセージに対する再試行の最大回数を、キュー・レベルで指定できます。メッセージ削除がこの数より多く失敗した場合、メッセージは例外キューに移動されるか、またはアプリケーションで使用できなくなります。

再試行遅延

条件が不適切な場合、メッセージを受信するトランザクションが終了する場合があります。AQ では、事前に指定された間隔で、不適切なメッセージを隠すことができます。再試行の最大回数とともに再試行遅延を指定できます。つまり、試行が失敗したメッセージは、再試行遅延間隔後に、デキュー用にキューで参照できます。それまでは、WAITING 状態になります。AQ バックグラウンド・プロセスであるタイム・マネージャが再試行遅延を施行します。再試行の最大回数のデフォルトの値は 5 であり、再試行遅延のデフォルト値は 0（ゼロ）です。再試行の最大回数および再試行遅延は、8.0 互換のマルチ・コンシューマ・キューでは使用できないことに注意してください。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/*  Create a package that enqueue with delay set to one day: /*
     CONNECT BOLADM/BOLADM
    >
     /* queue has max retries = 4 and retry delay = 12 hours */
     execute dbms_aqadm.alter_queue(queue_name = 'WS.WS_BOOKED_ORDERS_QUE',
     max_retr
     ies = 4,
                                    retry_delay = 3600*12);
    >
     /* processes the next order available in the booked_order_queue */
     CREATE OR REPLACE PROCEDURE  process_next_order()
     AS
       dqqopt                    dbms_aq.dequeue_options_t;
       msgprop                   dbms_aq.message_properties_t;
       deq_msgid                 RAW(16);
       book                      BOLADM.book_typ;
       item                      BOLADM.orderitem_typ;
       BOLADM.order_typ          order;
```

```
 BEGIN
>
   dqqopt.dequeue_option := DBMS_AQ.FIRST_MESSAGE;
   dbms_aq.dequeue('WS.WS_BOOKED_ORDERS_QUEUE', dqqopt, msgprop, order,
 deq_msgid
 );
>
   /* for simplicity, assume order has a single item */
  item = order.items(1);
   book = the_orders.item;
>
   /* assume search_inventory searches inventory for the book */
   /* if we don't find the book in the warehouse, abort transaction */
   IF  (search_inventory(book) != TRUE)
       rollback;
   ELSE
      process_order(order);
   END IF;
>
 END;
 /
```

Visual Basic（OO4O）：サンプル・コード

この機能については、データベースの dbexecutesql インタフェースを使用します。

Java（JDBC）：サンプル・コード

```
public static void setup_queue(Connection db_conn)
{
    AQSession         aq_sess;
    AQQueue           bookedorders_q;
    AQQueueProperty   q_prop;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        bookedorders_q = aq_sess.getQueue("WS", "WS_bookedorders_que");

        /* Alter queue - set max retries = 4 and retry delay = 12 hours */
        q_prop = new AQQueueProperty();
        q_prop.setMaxRetries(4);

        q_prop.setRetryInterval(3600*12);  // specified in seconds
```

```
            bookedorders_q.alterQueue(q_prop);


        }
        catch (Exception ex)
        {
            System.out.println("Exception: " + ex);
        }

    }
    public static void process_next_order(Connection db_conn)
    {
        AQSession         aq_sess;
        Order             deq_order;
        OrderItem         order_item;
        Book              book;
        AQDequeueOption   deq_option;
        AQMessageProperty msg_prop;
        AQQueue           bookedorders_q;
        AQMessage         message;
        AQObjectPayload   obj_payload;

        try
        {
            /* Create an AQ Session: */
            aq_sess = AQDriverManager.createAQSession(db_conn);

            deq_option = new AQDequeueOption();

            deq_option.setNavigationMode(AQDequeueOption.NAVIGATION_FIRST_MESSAGE);

            bookedorders_q = aq_sess.getQueue("WS", "WS_bookedorders_que");


            /* Dequeue the message */
            message = bookedorders_q.dequeue(deq_option, Order.getFactory());

            obj_payload = message.getObjectPayload();

            deq_order = (Order)(obj_payload.getPayloadData());

            /* for simplicity, assume order has a single item */
            order_item = deq_order.getItems().getElement(0);
            book = order_item.getItem();

            /* assume search_inventory searches inventory for the book
             * if we don't find the book in the warehouse, abort transaction
```

```
         */
        if(search_inventory(book) != true)
          db_conn.rollback();
        else
          process_order(deq_order);

    }
    catch (AQException aqex)
    {
        System.out.println("Exception-1: " + aqex);
    }
    catch (Exception ex)
    {
        System.out.println("Exception-2: " + ex);
    }
}
```

例外処理

AQ は、EXCEPTION_QUEUES、EXPIRATION、MAX_RETRIES および RETRY_DERAY の 4 つの統合化メカニズムによって、アプリケーションの例外処理をサポートします。

exception_queue（例外キュー）は、期限切れまたは処理できないすべてのメッセージのリポジトリになります。アプリケーションから例外キューには直接エンキューできません。また、マルチ・コンシューマの例外キューに、サブスクライバを対応付けることはできません。ただし、期限切れまたは処理できないメッセージを処理するアプリケーションは、例外キューからデキューできます。マルチ・コンシューマ・キューのメッセージを受け入れるために作成された例外キューは、それ自体がマルチ・コンシューマ・キューである必要があります。他のキューと同様に、例外キューも DBMS_AQADM.START_QUEUE プロシージャを使用してデキューできる必要があります。例外キューをエンキュー可能に設定しようとすると、Oracle エラーになります。

期限切れになったメッセージは、例外キューに移されます。マルチ・コンシューマ・キューのメッセージを受け入れる例外キューも、マルチ・コンシューマ・キューである必要があります。マルチ・コンシューマ・キューから移された期限切れメッセージは、受信者を指定してデキューできません。ただし、デキュー・オプションのコンシューマ名に NULL を指定することによって、REMOVE モードで 1 回のみデキューできます。したがって、デキューの観点からすると、マルチ・コンシューマの例外キューはシングル・コンシューマ・キューのように動作します。これは、それぞれの期限切れメッセージは、NULL コンシューマ名を使用して 1 回しかデキューできないためです。メッセージ ID を指定すると、メッセージも例外キューからデキューできます。

例外キューは、エンキュー時に指定可能なメッセージ・プロパティです（11-10 ページの「メッセージのエンキュー（メッセージ・プロパティの指定）」を参照）。PL/SQL では、DBMS_AQ.MESSAGE_PROPERTIES_T レコードの exception_queue 属性を使用して、例外キューを指定できます。OCI では、LNOCISetAttr プロシージャを使用して、LNOCIAQMsgProperties 記述子の LNOCI_ATTR_EXCEPTION_QUEUE 属性を設定します。

例外キューが指定されていないと、デフォルトの例外キューが使用されます。キューがキュー表（たとえばQTAB）の中に作成されている場合、デフォルトの例外キューはAQ$_QTAB_Eとなります。デフォルトの例外キューは、キュー表が作成されるときに自動的に作成されます。メッセージは、次の条件が成立するときにAQによって例外キューに移されます。

- そのメッセージが、指定された期限内にデキューされなかった場合。複数の受信者を指定したメッセージの場合、指定されていながら指定された期限切れまでにそのメッセージをデキューできない受信者が1つでもあると、例外キューに移されます。デフォルトの期限切れはDBMS_AQ.NEVERで、そのメッセージは期限切れになりません。
- そのメッセージは正常にデキューされているが、そのメッセージの処理中にエラーが発生したために、メッセージをデキューするアプリケーションがトランザクションをロールバックした場合。この場合、メッセージはキューに戻され、同一キューからのデキューを待機しているアプリケーションであればどれでも使用可能になります。アプリケーションがトランザクション全体にわたって、またはデキュー以前のセーブポイントまでロールバックしたときは、デキュー処理がロールバックまたは取り消されたとみなされます。メッセージがデキューされてロールバックされることが再試行制限の指定を超えて繰り返されると、そのメッセージは例外キューに移されます。

  複数の受信者を指定したメッセージの場合、各メッセージが受信者ごとにそれぞれの再試行回数を保持しています。すべての受信者の再試行回数が再試行制限の指定を超えたときのみ、そのメッセージは例外キューに移されます。シングル・コンシューマ・キューと8.1互換のマルチ・コンシューマ・キューの場合、デフォルトの再試行制限は5回です。8.0互換のマルチ・コンシューマ・キューでは、再試行制限はサポートされていません。
- クライアントによって実行された文に含まれているデキューは成功したが、文自体は後で例外処理のために取り消された場合。これがどのような場合かを理解するために、DBMS_AQ.DEQUEUEコールを含むPL/SQLプロシージャを考えてみます。デキュー・プロシージャが成功したが、PL/SQLプロシージャから例外が発生した場合、AQはデキュー・プロシージャから戻されたメッセージのRETRY_COUNTを増分します。
- クライアント・プログラムはメッセージのデキューに成功したが、トランザクションをコミットする前に終了した場合。

8.1互換のマルチ・コンシューマ・キューを指定したメッセージは、一度例外キューに移されると、指定された受信者はデキューできません。ただし、REMOVEまたはBROWSEモードで、デキュー・オプションのコンシューマ名にNULLを指定することによって、1回のみデキューできます。メッセージIDを指定してデキューすることもできます。

シングル・コンシューマ・キューまたは8.0互換のマルチ・コンシューマ・キューを指定したメッセージが例外キューに移された場合は、メッセージIDを指定してデキューできます。

ユーザーは、キューにRETRY_DELAYを対応付けることができます。このパラメータのデフォルト値は0（ゼロ）で、RETRY_COUNTが増分されるとすぐにそのメッセージがデキューできるようになるという意味です。それ以外の値の場合、そのメッセージはRETRY_DELAY秒間は使用できません。RETRY_DELAY秒が過ぎると、キュー・モニターがそのメッセージにREADYマークを付けます。

マルチ・コンシューマ・キューでは、RETRY_DELAY は各サブスクライバに固有です。

シナリオ

BooksOnLine アプリケーションでは、各出荷地域のビジネス・ルールによって、すぐに応じることができない注文情報は受注残キューに移されます。受注残アプリケーションは、毎日 1 回その注文情報を出荷しようとします。5 日以内に出荷できない場合、その注文は例外キューに移されて特別に処理されます。AQ の再試行および例外処理機能を使用して、この処理を実装できます。

次の例では、キューを作成するときに、再試行回数の最大値および再試行の遅延間隔を設定する方法を示します。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/* Example for creating a back order queue in Western Region which allows a
   maximum of 5 retries and 1 day delay between each retry. */
CONNECT BOLADM/BOLADM
BEGIN
  dbms_aqadm.create_queue (
        queue_name              => 'WS.WS_backorders_que',
        queue_table             => 'WS.WS_orders_mqtab',
        max_retries             => 5,
        retry_delay             => 60*60*24);
END;
/

/* Create an exception queue for the back order queue for Western Region. */
CONNECT BOLADM/BOLADM
BEGIN
  dbms_aqadm.create_queue (
        queue_name              => 'WS.WS_backorders_excpt_que',
        queue_table             => 'WS.WS_orders_mqtab',
        queue_type              => DBMS_AQADM.EXCEPTION_QUEUE);
end;
/

/* Enqueue a message to WS_backorders_que and specify WS_backorders_excpt_que as the
exception queue for the message: */
CONNECT BOLADM/BOLADM
CREATE OR REPLACE PROCEDURE enqueue_WS_unfilled_order(backorder order_typ)
 AS
   back_order_queue_name    varchar2(62);
   enqopt                   dbms_aq.enqueue_options_t;
   msgprop                  dbms_aq.message_properties_t;
   enq_msgid                raw(16);
 BEGIN
```

```
   /* Set back order queue name for this message: */
   back_order_queue_name := 'WS.WS_backorders_que';

   /* Set exception queue name for this message: */
   msgprop.exception_queue := 'WS.WS_backorders_excpt_que';

   dbms_aq.enqueue(back_order_queue_name, enqopt, msgprop,
                   backorder, enq_msgid);
END;
/
```

Visual Basic（OO4O）：サンプル・コード

例外キューは、メッセージをエンキューするときに提供されるメッセージ・プロパティです。このプロパティが設定されていない場合、キューのデフォルト例外キューがすべてのエラー条件に対して使用されます。

```
set oraaq = OraDatabase.CreateAQ("CBADM.deferbilling_que")
Set OraMsg = OraAq.AQMsg(ORATYPE_OBJECT, "BOLADM.order_typ")
Set OraOrder = OraDatabase.CreateOraObject("BOLADM.order_typ")
OraMsg = OraOrder
   OraMsg.delay = 15*60*60*24
   OraMsg.ExceptionQueue = "WS.WS_backorders_que"
   'Fill up the order values
   OraMsg = OraOrder 'OraOrder contains the order details
   Msgid = OraAq.enqueue
```

Java（JDBC）：サンプル・コード

```
public static void createBackOrderQueues(Connection db_conn)
{
    AQSession         aq_sess;
    AQQueue           backorders_q;
    AQQueue           backorders_excp_q;
    AQQueueProperty   q_prop;
    AQQueueProperty   q_prop2;
    AQQueueTable      mq_table;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        mq_table = aq_sess.getQueueTable("WS", "WS_orders_mqtab");


        /* Create a back order queue in Western Region which allows a
```

```
            maximum of 5 retries and 1 day delay between each retry. */

        q_prop = new AQQueueProperty();
        q_prop.setMaxRetries(5);
        q_prop.setRetryInterval(60*24*24);

        backorders_q = aq_sess.createQueue(mq_table, "WS_backorders_que",
                                           q_prop);

        backorders_q.start(true, true);

        /* Create an exception queue for the back order queue for
           Western Region. */
        q_prop2 = new AQQueueProperty();
        q_prop2.setQueueType(AQQueueProperty.EXCEPTION_QUEUE);

        backorders_excp_q = aq_sess.createQueue(mq_table,
                                           "WS_backorders_excpt_que", q_prop2);

    }
    catch (Exception ex)
    {
        System.out.println("Exception " + ex);
    }

}

/* Enqueue a message to WS_backorders_que and specify WS_backorders_excpt_que
   as the exception queue for the message: */
public static void enqueue_WS_unfilled_order(Connection db_conn,
                                             Order back_order)
{
    AQSession         aq_sess;
    AQQueue           back_order_q;
    AQEnqueueOption   enq_option;
    AQMessageProperty m_property;
    AQMessage         message;
    AQObjectPayload   obj_payload;
    byte[]            enq_msg_id;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        back_order_q = aq_sess.getQueue("WS", "WS_backorders_que");
```

```
            message = back_order_q.createMessage();

            /* Set exception queue name for this message: */
            m_property = message.getMessageProperty();

            m_property.setExceptionQueue("WS.WS_backorders_excpt_que");

            obj_payload = message.getObjectPayload();
            obj_payload.setPayloadData(back_order);

            enq_option = new AQEnqueueOption();

            /* Enqueue the message */
            enq_msg_id = back_order_q.enqueue(enq_option, message);

            db_conn.commit();
        }
        catch (Exception ex)
        {
            System.out.println("Exception: " + ex);
        }

    }
```

ルールベースのサブスクリプション

メッセージは、そのメッセージ・プロパティまたは内容に基づいて、様々な受信者に送られます。ユーザーは、与えられたキューに対して、特定の条件に合うメッセージのみを受信するために関心を指定するルールベースのサブスクリプションを定義します。

ルールは、評価の結果 TRUE または FALSE となるブール式です。SQL 問合せの WHERE 句の構文と同様、ルールはメッセージ・プロパティまたはメッセージ内容を示す属性によって表現されます。このようなサブスクライバ・ルールが、着信メッセージに対して評価され、一致したルールを使用してメッセージ受信者が判断されます。この機能によって、メッセージの内容ベースのサブスクリプションおよびルーティングがサポートされます。

サブスクリプション・ルールは、ExistsNode などの XML 演算子を使用して、XMLType の属性にも定義できます。

シナリオ

BooksOnLine アプリケーションでは、内容ベースのサブスクリプションおよびルーティングを利用するパブリッシュ・サブスクライブ・パラダイムを実装するために、どのようにルールベースのサブスクリプションが使用されているかを説明します。注文入力アプリケーションと各出荷処理アプリケーションの相互作用が、次のようにモデル化されています。

- 西部向け出荷は、合衆国の西部地区の注文を扱います。
- 東部向け出荷は、合衆国の東部地区の注文を扱います。
- 海外向け出荷は、合衆国以外からの注文を扱います。
- 海外向け出荷では、XMLType 属性を確認して、特別な処理が必要かどうかを識別します。
- 東部向け出荷は、合衆国全域の至急注文を扱います。

各出荷処理アプリケーションは、OE の入力済注文キューにサブスクライブします。次に示すルールベースのサブスクリプションは、注文入力アプリケーションから各出荷処理アプリケーションに入力済注文をルーティングするために、OE のユーザーによって定義されます。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CONNECT OE/OE;
```

西部向け出荷は、エージェント・アドレス（メッセージが配信される宛先キュー）として WS 入力済注文キューを設定した West_Shipping といわれるエージェントを定義します。このエージェントは OE の入力済注文キューに、orderregion および ordertype 属性を指定するルールを使用してサブスクライブします。

```
/*  Add a rule-based subscriber for West Shipping -
    West Shipping handles Western region U.S. orders,
    Rush Western region orders are handled by East Shipping: */
DECLARE
  subscriber     aq$_agent;
BEGIN
  subscriber := aq$_agent('West_Shipping', 'WS.WS_bookedorders_que', null);
  dbms_aqadm.add_subscriber(
               queue_name => 'OE.OE_bookedorders_que',
               subscriber => subscriber,
               rule       => 'tab.user_data.orderregion =
                   ''WESTERN'' AND tab.user_data.ordertype != ''RUSH''');
END;
/
```

東部向け出荷は、エージェント・アドレス（メッセージが配信される宛先キュー）として ES 入力済注文キューを設定した East_Shipping といわれるエージェントを定義します。このエージェントは、orderregion、ordertype および customer 属性に指定されたルールを使用して OE の入力済注文キューを定義します。

```
/*  Add a rule-based subscriber for East Shipping -
    East shipping handles all Eastern region orders,
    East shipping also handles all U.S. rush orders: */
DECLARE
  subscriber     aq$_agent;
BEGIN
  subscriber := aq$_agent('East_Shipping', 'ES.ES_bookedorders_que', null);
  dbms_aqadm.add_subscriber(
        queue_name => 'OE.OE_bookedorders_que',
        subscriber => subscriber,
        rule       => 'tab.user_data.orderregion = ''EASTERN'' OR
                      (tab.user_data.ordertype = ''RUSH'' AND
                       tab.user_data.customer.country = ''USA'') ');
END;
```

海外向け出荷は、エージェント・アドレス（メッセージが配信される宛先キュー）として OS 入力済注文キューを設定した Overseas_Shipping といわれるエージェントを定義します。このエージェントは、orderregion 属性に指定したルールを使用して OE の入力済注文キューを定義します。海外向け出荷サイトでの注文の表現は、注文入力サイトでの注文の表現と異なるため、メッセージが注文入力サイトから海外向け出荷サイトに伝播される前に、変換が適用されます。

```
/*  Add a rule-based subscriber (for Overseas Shipping) to the Booked orders queues
with Transformation. Overseas Shipping handles all non-US orders: */
DECLARE
  subscriber     aq$_agent;
BEGIN
  subscriber := aq$_agent('Overseas_Shipping','OS.OS_bookedorders_que',null);

  dbms_aqadm.add_subscriber(
        queue_name     => 'OE.OE_bookedorders_que',
        subscriber     => subscriber,
        rule           => 'tab.user_data.orderregion = ''INTERNATIONAL'''',
        transformation => 'OS.OE2XML');
END;
```

変換の定義の詳細は、8-5 ページの「メッセージ・フォーマットの変換」を参照してください。

海外向け出荷サイトには、RUSH 注文を処理するために、Overseas_DHL サブスクライバがあると想定します。OS_bookedorders_que には XMLType として表現された注文の詳細があるため、ルールには XPath 構文が使用されます。

```
DECLARE
  subscriber     aq$_agent;
BEGIN
  subscriber := aq$_agent('Overseas_DHL', null, null);
```

```
  dbms_aqadm.add_subscriber(
        queue_name     => 'OS.OS_bookedorders_que',
        subscriber     => subscriber,
        rule           => 'tab.user_data.extract(''/ORDER_TYP/ORDERTYPE/
                              text()'').getStringVal()=''RUSH''');

END;
```

Visual Basic（OO4O）：サンプル・コード

この機能は、現在使用できません。

Java（JDBC）：サンプル・コード

```
public static void addRuleBasedSubscribers(Connection db_conn)
{

    AQSession         aq_sess;
    AQQueue           bookedorders_q;
    String            rule;
    AQAgent           agt1, agt2, agt3;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

        bookedorders_q = aq_sess.getQueue("OE", "OE_booked_orders_que");

        /* Add a rule-based subscriber for West Shipping -
           West Shipping handles Western region U.S. orders,
           Rush Western region orders are handled by East Shipping: */

        agt1 = new AQAgent("West_Shipping", "WS.WS_bookedorders_que");

        rule = "tab.user_data.orderregion = 'WESTERN' AND " +
               "tab.user_data.ordertype != 'RUSH'";

        bookedorders_q.addSubscriber(agt1, rule);

        /*  Add a rule-based subscriber for East Shipping -
            East shipping handles all Eastern region orders,
            East shipping also handles all U.S. rush orders: */

        agt2 = new AQAgent("East_Shipping", "ES.ES_bookedorders_que");
        rule = "tab.user_data.orderregion = 'EASTERN' OR " +
               "(tab.user_data.ordertype = 'RUSH' AND " +
```

```
                "tab.user_data.customer.country = 'USA')";

            bookedorders_q.addSubscriber(agt2, rule);

            /*  Add a rule-based subscriber for Overseas Shipping
                Intl Shipping handles all non-U.S. orders: */

            agt3 = new AQAgent("Overseas_Shipping", "OS.OS_bookedorders_que");
            rule = "tab.user_data.orderregion = 'INTERNATIONAL'";

            bookedorders_q.addSubscriber(agt3, rule);
        }
        catch (Exception ex)
        {
            System.out.println("Exception: " + ex);
        }
    }
```

Listen 機能

アドバンスト・キューイングは、1 つの LISTEN コールで複数のキューのメッセージを監視できます。アプリケーションは、LISTEN を使用して複数のサブスクリプションに対するメッセージを待機できます。また、ゲートウェイ・アプリケーションでも、この機能を使用して複数のキューを監視できます。LISTEN コールが正常に戻された場合は、デキューしてメッセージを取り出す必要があります（11-24 ページの「1 個以上のシングル・コンシューマ・キューのリスニング」を参照）。

LISTEN コールがない場合、一連のキューからデキューするアプリケーションは、それらのキューにメッセージがあるかどうかを判断するために継続的にポーリングする必要があります。または、各キューに対するデキュー処理が互いに独立するようにアプリケーションを設計することもできます。ただし、長期間そのキューにメッセージがない場合は、これらの方法によって、許容できないオーバーヘッドが発生します。LISTEN コールはそのようなアプリケーションに適しています。

エージェント・リストの複数のエージェントに向けたメッセージがいくつかある場合、LISTEN はメッセージの宛先になっている最初のエージェントとともに戻ります。その意味で、LISTEN のキュー監視は公正ではありません。アプリケーション設計者はこの仕様を認識してこのコールを使用する必要があります。あるエージェントが他のエージェントに対してメッセージの欠乏状態を引き起こさないように、アプリケーションはエージェント・リストのエージェント順序を変更できます。

シナリオ

BooksOnLine シナリオの顧客サービス・コンポーネントでは、異なるデータベースから顧客サービス・キューに届いたメッセージにはその状態が示されます。顧客サービス・アプリケーションはそのキューを監視し、顧客の注文に関するメッセージがあるときはいつでも order_status_table の注文状態を更新します。アプリケーションは listen コールを使用して様々なキューを監視します。それらのキューのどれかにメッセージがあるときはいつでも、メッセージをデキューし、それに応じて注文状態を更新します。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CODE (in tkaqdocd.sql)

/* Update the status of the order in the order status table: */
CREATE OR REPLACE PROCEDURE update_status(
                                new_status    IN VARCHAR2,
                                order_msg     IN BOLADM.ORDER_TYP)
IS
 old_status    VARCHAR2(30);
 dummy         NUMBER;
BEGIN

  BEGIN
    /* Query old status from the table: */
    SELECT st.status INTO old_status FROM order_status_table st
       WHERE st.customer_order.orderno = order_msg.orderno;

  /* Status can be 'BOOKED_ORDER', 'SHIPPED_ORDER', 'BACK_ORDER'
     and   'BILLED_ORDER': */

   IF new_status = 'SHIPPED_ORDER' THEN
      IF old_status = 'BILLED_ORDER' THEN
        return;             /* message about a previous state */
      END IF;
   ELSIF new_status = 'BACK_ORDER' THEN
      IF old_status = 'SHIPPED_ORDER' OR old_status = 'BILLED_ORDER' THEN
        return;             /* message about a previous state */
      END IF;
   END IF;

   /* Update the order status:  */
     UPDATE order_status_table st
        SET st.customer_order = order_msg, st.status = new_status;

   COMMIT;

  EXCEPTION
```

```
  WHEN OTHERS  THEN     /* change to no data found */
    /* First update for the order: */
    INSERT INTO order_status_table(customer_order, status)
    VALUES (order_msg, new_status);
    COMMIT;

  END;
END;
/


/* Dequeues message from 'QUEUE' for 'CONSUMER': */
CREATE OR REPLACE PROCEDURE DEQUEUE_MESSAGE(
                         queue      IN   VARCHAR2,
                         consumer   IN   VARCHAR2,
                         message    OUT  BOLADM.order_typ)
IS

dopt                     dbms_aq.dequeue_options_t;
mprop                    dbms_aq.message_properties_t;
deq_msgid                RAW(16);
BEGIN
  dopt.dequeue_mode := dbms_aq.REMOVE;
  dopt.navigation := dbms_aq.FIRST_MESSAGE;
  dopt.consumer_name := consumer;

  dbms_aq.dequeue(
               queue_name => queue,
               dequeue_options => dopt,
               message_properties => mprop,
               payload => message,
               msgid => deq_msgid);
  commit;
END;
/

/* Monitor the queues in the customer service databse for 'time' seconds: */
 CREATE OR REPLACE PROCEDURE MONITOR_STATUS_QUEUE(time  IN  NUMBER)
IS
  agent_w_message   aq$_agent;
  agent_list        dbms_aq.agent_list_t;
  wait_time         INTEGER := 120;
  no_message        EXCEPTION;
  pragma EXCEPTION_INIT(no_message, -25254);
  order_msg         boladm.order_typ;
  new_status        VARCHAR2(30);
  monitor           BOOLEAN := TRUE;
```

```
  begin_time        NUMBER;
  end_time          NUMBER;
BEGIN

 begin_time :=  dbms_utility.get_time;
 WHILE (monitor)
 LOOP
 BEGIN

  /* Construct the waiters list: */
  agent_list(1) := aq$_agent('BILLED_ORDER', 'CS_billedorders_que', NULL);
  agent_list(2) := aq$_agent('SHIPPED_ORDER', 'CS_shippedorders_que',
NULL);
  agent_list(3) := aq$_agent('BACK_ORDER', 'CS_backorders_que', NULL);
  agent_list(4) := aq$_agent('Booked_ORDER', 'CS_bookedorders_que', NULL);

   /* Wait for order status messages: */
   dbms_aq.listen(agent_list, wait_time, agent_w_message);

   dbms_output.put_line('Agent' || agent_w_message.name || ' Address '||
agent_w_message.address);
   /* Dequeue the message from the queue: */
   dequeue_message(agent_w_message.address, agent_w_message.name, order_msg);

   /* Update the status of the order depending on the type of the message,
    * the name of the agent contains the new state: */
   update_status(agent_w_message.name, order_msg);

  /* Exit if we have been working long enough: */
   end_time := dbms_utility.get_time;
   IF  (end_time - begin_time > time)   THEN
     EXIT;
   END IF;

  EXCEPTION
  WHEN  no_message  THEN
    dbms_output.put_line('No messages in the past 2 minutes');
       end_time := dbms_utility.get_time;
    /* Exit if we have done enough work: */
    IF  (end_time - begin_time > time)   THEN
      EXIT;
    END IF;
  END;

  END LOOP;
END;
/
```

Visual Basic（OO4O）：サンプル・コード

この機能は、現在使用できません。

Java（JDBC）：サンプル・コード

```
public static void monitor_status_queue(Connection db_conn)
{
    AQSession         aq_sess;
    AQAgent[]         agt_list = null;
    AQAgent           ret_agt  = null;
    Order             deq_order;
    AQDequeueOption   deq_option;
    AQQueue           orders_q;
    AQMessage         message;
    AQObjectPayload   obj_payload;
    String            owner = null;
    String            queue_name = null;
    int               idx = 0;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

/* Construct the waiters list: */
agt_list = new AQAgent[4];

agt_list[0] = new AQAgent("BILLED_ORDER", "CS_billedorders_que", 0);
agt_list[1] = new AQAgent("SHIPPED_ORDER", "CS_shippedorders_que", 0);
agt_list[2] = new AQAgent("BACK_ORDER", "CS_backorders_que", 0);
agt_list[3] = new AQAgent("BOOKED_ORDER", "CS_bookedorders_que", 0);

/* Wait for order status messages for 120 seconds: */
ret_agt = aq_sess.listen(agt_list, 120);

System.out.println("Message available for agent: " +
   ret_agt.getName()  + "   "  + ret_agt.getAddress());

/* Get owner, queue where message is available */
idx = ret_agt.getAddress().indexOf(".");

if(idx != -1)
{
  owner = ret_agt.getAddress().substring(0, idx);
  queue_name = ret_agt.getAddress().substring(idx + 1);

/* Dequeue the message */
```

```
deq_option = new AQDequeueOption();

deq_option.setConsumerName(ret_agt.getName());
deq_option.setWaitTime(1);

orders_q = aq_sess.getQueue(owner, queue_name);

/* Dequeue the message */
message = orders_q.dequeue(deq_option, Order.getFactory());

obj_payload = message.getObjectPayload();

deq_order = (Order)(obj_payload.getPayloadData());

   System.out.println("Order number " + deq_order.getOrderno() + " retrieved");

   }
   catch (AQException aqex)
   {
   System.out.println("Exception-1: " + aqex);
   }
   catch (Exception ex)
   {
    System.out.println("Exception-2: " + ex);
    }

}
```

デキュー中のメッセージ変換

8-5 ページの「メッセージ・フォーマットの変換」および 8-51 ページの「エンキュー中のメッセージ変換」のシナリオで説明したとおり、OE スキーマ内のキューは OE.orders_typ のペイロード型で、WS スキーマ内のキューは WS.orders_typ_sh のペイロード型です。

シナリオ

デキュー時に、アプリケーションは、デキューに対する選択条件を使用してメッセージを OE_booked_orders_topic から WS_booked_orders_topic に移動し、order_region が「WESTERN」で order_type が「RUSH」ではない注文のみをデキューできます。また、変換が適用され、ws.order_typ_sh 型の注文が取り出されます。その後、メッセージは WS.ws_booked_orders キューにエンキューされます。

PL/SQL（DBMS_AQ）：サンプル・コード

```
CREATE OR REPLACE PROCEDURE   fwd_message_to_ws_shipping AS

  enq_opt   dbms_aq.enqueue_options_t;
  deq_opt   dbms_aq.dequeue_options_t;
  msg_prp   dbms_aq.message_properties_t;
  booked_order WS.order_typ_sh;
BEGIN

/* First dequeue the message from OE booked orders topic */
    deq_opt.transformation := 'OE.OE2WS';
    deq_opt.condition := 'tab.user_data.order_region = ''WESTERN'' and tab.user_
data.order_type != ''RUSH''';

    dbms_aq.dequeue('OE.oe_bookedorders_topic', deq_opt,
                    msg_prp, booked_order);

/* enqueue the message in the WS booked orders topic */
    msg_prp.recipient_list(0) := aq$_agent('West_shipping', null, null);

    dbms_aq.enqueue('WS.ws_bookedorders_topic',
                    enq_opt, msg_prp, booked_order);

END;
```

Visual Basic（OO4O）：サンプル・コード

今回のリリースでは、例は記載していません。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

AQ XML サーブレットを使用したデキュー

SOAP を使用して、インターネット経由でデキュー・リクエストを実行できます。SOAP を使用した AQ メッセージの受信の詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

BooksOnLine のシナリオで、東部向け出荷処理アプリケーションが、相関識別子 RUSH の AQ メッセージを、インターネット経由で受信すると想定します。デキュー・リクエストのフォーマットは、次のとおりです。

```
<?xml version="1.0"?>
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
        <AQXmlReceive xmlns = "http://ns.oracle.com/AQ/schemas/access">
```

```
            <consumer_options>
              <destination>ES_ES_bookedorders_que</destination>
              <consumer_name>East_Shipping</consumer_name>
              <wait_time>0</wait_time>
              <selector>
                   <correlation>RUSH</correlation>
              </selector>
            </consumer_options>

            <AQXmlCommit/>

          </AQXmlReceive>
        </Body>
</Envelope>
```

非同期通知

この機能によって、クライアントは、関心があるメッセージの通知を受信できます。非同期通知は、通知を受信するための複数のメカニズムをサポートします。クライアントは、PL/SQL、JMS または OCI コールバック関数を使用してプロシージャで通知を受信するか、または電子メールまたは HTTP の POST を介して通知を受信できます。

永続キューの場合、JMS 通知以外の通知にはメッセージ・プロパティのみが含まれます。メッセージを受信するには、クライアントが明示的にデキューする必要があります。JMS では、通知の一部としてデキューが行われるため、明示的デキューの必要はありません。非永続キューの場合、メッセージは通知の一部として配信されます。

クライアントは、通知の表現を `RAW` または `XML` のいずれかに指定することもできます。

シナリオ

BooksOnLine アプリケーションでは、顧客は FedEx による出荷（優先順位 1）、優先扱い航空便による出荷（優先順位 2）または通常地上便による出荷（優先順位 3）のいずれかの方法を要求できます。

出荷処理アプリケーションは、注文された書籍をユーザー・リクエストに従って出荷します。BooksOnLine は、それぞれの出荷方法が毎日何件要求されるかに関心があります。アプリケーションは、非同期通知機能をこの目的に使用します。`WS.WS_bookedorders_que` に対して通知を登録し、そのキューに新しいメッセージがあることを通知されると、アプリケーションはメッセージの優先順位に従って適切な出荷方法の件数を更新します。

Visual Basic（OO4O）: サンプル・コード

OO4O のオンライン・ヘルプの「Monitoring Message」を参照してください。

Java（JDBC）：サンプル・コード

この機能は、Java API ではサポートされません。

C（OCI）：サンプル・コード

この例では OCIRegister の使用方法を示します。この OCI クライアント・プログラムは、出荷サイトで、FEDEX、AIR および GROUND の各出荷方法の注文が何件あるかを追跡します。メッセージの優先順位フィールドによって、希望の出荷方法を判断できます。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>
#ifdef WIN32COMMON
#define sleep(x)   Sleep(1000*(x))
#endif
static text *username = (text *) "WS";
static text *password = (text *) "WS";

static OCIEnv *envhp;
static OCIServer *srvhp;
static OCIError *errhp;
static OCISvcCtx *svchp;

static void checkerr(/*_ OCIError *errhp, sword status _*/);

struct ship_data
{
  ub4  fedex;
  ub4  air;
  ub4  ground;
};

typedef struct ship_data ship_data;

int main(/*_ int argc, char *argv[] _*/);


/* Notify callback: */
ub4 notifyCB(ctx, subscrhp, pay, payl, desc, mode)
dvoid *ctx;
LNOCISubscription *subscrhp;
dvoid *pay;
ub4    payl;
dvoid *desc;
ub4    mode;
{
```

```
text                *subname;
ub4                 size;
ship_data           *ship_stats = (ship_data *)ctx;
text                *queue;
text                *consumer;
OCIRaw              *msgid;
ub4                 priority;
OCIAQMsgProperties  *msgprop;

OCIAttrGet((dvoid *)subscrhp, OCI_HTYPE_SUBSCRIPTION,
                            (dvoid *)&subname, &size,
                            OCI_ATTR_SUBSCR_NAME, errhp);

/* Extract the attributes from the AQ descriptor.
   Queue name: */
OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&queue, &size,
           OCI_ATTR_QUEUE_NAME, errhp);

/* Consumer name: */
OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&consumer, &size,
           OCI_ATTR_CONSUMER_NAME, errhp);

/* Msgid: */
OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&msgid, &size,
           OCI_ATTR_NFY_MSGID, errhp);

/* Message properties: */
OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&msgprop, &size,
           OCI_ATTR_MSG_PROP, errhp);

/* Get priority from message properties: */
checkerr(errhp, OCIAttrGet(msgprop, OCI_DTYPE_AQMSG_PROPERTIES,
                            (dvoid *)&priority, 0,
                            OCI_ATTR_PRIORITY, errhp));

 switch (priority)
 {
 case 1:  ship_stats->fedex++;
          break;
 case 2 : ship_stats->air++;
          break;
 case 3:  ship_stats->ground++;
          break;
 default:
          printf(" Error priority %d", priority);
 }
}
```

```
int main(argc, argv)
int argc;
char *argv[];
{
  OCISession *authp = (OCISession *) 0;
  OCISubscription *subscrhp[8];
  ub4 namespace = OCI_SUBSCR_NAMESPACE_AQ;
  ship_data  ctx = {0,0,0};
  ub4 sleep_time = 0;

  printf("Initializing OCI Process\n");

  /* Initialize OCI environment with OCI_EVENTS flag set: */
  (void) OCIInitialize((ub4) OCI_EVENTS|OCI_OBJECT, (dvoid *)0,
                       (dvoid * (*)(dvoid *, size_t)) 0,
                       (dvoid * (*)(dvoid *, dvoid *, size_t))0,
                       (void (*)(dvoid *, dvoid *)) 0 );

  printf("Initialization successful\n");

  printf("Initializing OCI Env\n");
  (void) OCIEnvInit( (OCIEnv **) &envhp, OCI_DEFAULT, (size_t) 0, (dvoid **) 0 );
  printf("Initialization successful\n");

  checkerr(errhp, OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &errhp, OCI_HTYPE_
ERROR,
                   (size_t) 0, (dvoid **) 0));

  checkerr(errhp, OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &srvhp, OCI_HTYPE_
SERVER,
                   (size_t) 0, (dvoid **) 0));

  checkerr(errhp, OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &svchp, OCI_HTYPE_
SVCCTX,
                   (size_t) 0, (dvoid **) 0));


  printf("connecting to server\n");
  checkerr(errhp, OCIServerAttach( srvhp, errhp, (text *)"inst1_alias",
           strlen("inst1_alias"), (ub4) OCI_DEFAULT));
  printf("connect successful\n");

  /* Set attribute server context in the service context: */
  checkerr(errhp, OCIAttrSet( (dvoid *) svchp, OCI_HTYPE_SVCCTX, (dvoid *)srvhp,
                    (ub4) 0, OCI_ATTR_SERVER, (OCIError *) errhp));
```

```
checkerr(errhp, OCIHandleAlloc((dvoid *) envhp, (dvoid **)&authp,
                     (ub4) OCI_HTYPE_SESSION, (size_t) 0, (dvoid **) 0));

/* Set username and password in the session handle: */
checkerr(errhp, OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION,
                (dvoid *) username, (ub4) strlen((char *)username),
                (ub4) OCI_ATTR_USERNAME, errhp));

checkerr(errhp, OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION,
                (dvoid *) password, (ub4) strlen((char *)password),
                (ub4) OCI_ATTR_PASSWORD, errhp));

/* Begin session: */
checkerr(errhp, OCISessionBegin ( svchp,  errhp, authp, OCI_CRED_RDBMS,
                        (ub4) OCI_DEFAULT));

(void) OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX,
                 (dvoid *) authp, (ub4) 0,
                 (ub4) OCI_ATTR_SESSION, errhp);


/* Register for notification: */
printf("allocating subscription handle\n");
subscrhp[0] = (OCISubscription *)0;
(void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&subscrhp[0],
                      (ub4) OCI_HTYPE_SUBSCRIPTION,
                      (size_t) 0, (dvoid **) 0);

printf("setting subscription name\n");
(void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
               (dvoid *) "WS.WS_BOOKEDORDERS_QUE:BOOKED_ORDERS",
               (ub4) strlen("WS.WS_BOOKEDORDERS_QUE:BOOKED_ORDERS"),
               (ub4) OCI_ATTR_SUBSCR_NAME, errhp);

printf("setting subscription callback\n");
(void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
               (dvoid *) notifyCB, (ub4) 0,
               (ub4) OCI_ATTR_SUBSCR_CALLBACK, errhp);

(void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
               (dvoid *)&ctx, (ub4)sizeof(ctx),
               (ub4) OCI_ATTR_SUBSCR_CTX, errhp);

printf("setting subscription namespace\n");
(void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
               (dvoid *) &namespace, (ub4) 0,
```

```
                    (ub4) OCI_ATTR_SUBSCR_NAMESPACE, errhp);

  printf("Registering \n");
  checkerr(errhp, OCISubscriptionRegister(svchp, subscrhp, 1, errhp,
                                          OCI_DEFAULT));

  sleep_time = (ub4)atoi(argv[1]);
  printf ("waiting for %d s", sleep_time);
  sleep(sleep_time);

  printf("Exiting");
  exit(0);
}

void checkerr(errhp, status)
LNOCIError *errhp;
sword status;
{
  text errbuf[512];
  sb4 errcode = 0;

  switch (status)
  {
  case OCI_SUCCESS:
    break;
  case OCI_SUCCESS_WITH_INFO:
    (void) printf("Error - OCI_SUCCESS_WITH_INFO\n");
    break;
  case OCI_NEED_DATA:
    (void) printf("Error - OCI_NEED_DATA\n");
    break;
  case OCI_NO_DATA:
    (void) printf("Error - OCI_NODATA\n");
    break;
  case OCI_ERROR:
    (void) OCIErrorGet((dvoid *)errhp, (ub4) 1, (text *) NULL, &errcode,
                       errbuf, (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
    (void) printf("Error - %.*s\n", 512, errbuf);
    break;
  case OCI_INVALID_HANDLE:
    (void) printf("Error - OCI_INVALID_HANDLE\n");
    break;
  case OCI_STILL_EXECUTING:
    (void) printf("Error - OCI_STILL_EXECUTE\n");
    break;
  case OCI_CONTINUE:
    (void) printf("Error - OCI_CONTINUE\n");
```

```
    break;
  default:
    break;
  }
}
```

PL/SQL（DBMS_AQ）：サンプル・コード

この例では、DBMS_AQ.REGISTER プロシージャの使用方法について説明します。

BooksOnLine のシナリオで、サブスクライバ BOOKED_ORDER で WS.WS_BOOKED_ORDERS_QUE キューのメッセージが受信されると、PL/SQL コールバック WS.notifyCB() をコールするとします。また、注文がサブスクライバ BOOKED_ORDERS のキューにエンキューされると、john@company.com に電子メールを送信するとします。さらに、サーブレット http://xyz.company.com/servlets/NofifyServlet をコールするとします。これは、次のとおり実行できます。

まず、通知に対してコールされる PL/SQL プロシージャを定義します。

```
connect ws/ws;
set echo on;
set serveroutput on;

-- notifyCB callback
create or replace procedure notifyCB(
  context raw, reginfo sys.aq$_reg_info, descr sys.aq$_descriptor,
  payload raw, payloadl number)
AS
  dequeue_options   DBMS_AQ.dequeue_options_t;
  message_properies DBMS_AQ.message_properties_t;
  message_handle    RAW(16);
  message           BOLADM.order_typ;
BEGIN
  -- get the consumer name and msg_id from the descriptor
  dequeue_options.msgid := descr.msg_id;
  dequeue_options.consumer_name := descr.consumer_name;

  -- Dequeue the message
  DBMS_AQ.DEQUEUE(queue_name => descr.queue_name,
                  dequeue_options => dequeue_options,
                  message_properties => message_properties,
                  payload => message,
                  msgid => message_handle);

  commit;

  DBMS_OUTPUT.PUTLINE('Received Order: ' || message.orderno);
```

```
END;
/
```

PL/SQL プロシージャ、電子メールおよび HTTP URL は、次のとおり登録できます。

```
connect ws/ws;
set echo on;
set serveroutput on;

DECLARE
  reginfo1     sys.aq$_reg_info;
  reginfo2     sys.aq$_reg_info;
  reginfo3     sys.aq$_reg_info;
  reginfolist  sys.aq$_reg_info_list;

BEGIN
   -- register for the pl/sql procedure notifyCB to be called on notification
  reginfo1 := sys.aq$_reg_info('WS.WS_BOOKEDORDERS_QUE:BOOKED_ORDERS',
                   DBMS_AQ.NAMESPACE_AQ, 'plsql://WS.notifyCB',
                   HEXTORAW('FF'));

  -- register for an e-mail to be sent to john@company.com on notification
  reginfo2 := sys.aq$_reg_info('WS.WS_BOOKEDORDERS_QUE:BOOKED_ORDERS',
                        DBMS_AQ.NAMESPACE_AQ, 'mailto://john@company.com',
                          HEXTORAW('FF'));

  -- register for an HTTP servlet to be invoked for notification
  reginfo3 := sys.aq$_reg_info('WS.WS_BOOKEDORDERS_QUE:BOOKED_ORDERS',
                        DBMS_AQ.NAMESPACE_AQ,
                        'http://xyz.oracle.com/servlets/NotifyServlet',
                          HEXTORAW('FF'));

  -- Create the registration info list
  reginfolist := sys.aq$_reg_info_list(reginfo1);
  reginfolist.EXTEND;
  reginfolist(2) := reginfo2;

  reginfolist.EXTEND;
  reginfolist(3) := reginfo3;

-- do the registration
  sys.dbms_aq.register(reginfolist, 3);

END;
```

AQ XML サーブレットを使用した通知登録

クライアントはインターネット経由で AQ 通知を登録できます。SOAP を使用した AQ 通知の登録の詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

登録リクエストのフォーマットは次のとおりです。

```
?xml version="1.0"?>
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>

        <AQXmlRegister xmlns = "http://ns.oracle.com/AQ/schemas/access">

          <register_options>
            <destination>WS.WS_BOOKEDORDERS_QUE</destination>
            <consumer_name>BOOKED_ORDERS</consumer_name>
            <notify_url>mailto://john@company.com</notify_url>
          </register_options>

          <AQXmlCommit/>
        </AQXmlRegister>
      </Body>
</Envelope>
```

john@company.com に送信された電子メールの通知には次のフォーマットがあります。

```
<?xml version="1.0"?>
<Envelope xmlns="http://www.oracle.com/schemas/IDAP/envelope">
    <Body>
        <AQXmlNotification xmlns="http://www.oracle.com/schemas/AQ/access">
            <notification_options>
                <destination>WS.WS_BOOKEDORDERS_QUE</destination>
            </notification_options>
            <message_set>
                <message>
                    <message_header>
                        <message_id>81128B6AC46D4B15E03408002092AA15</message_id>
                        <correlation>RUSH</correlation>
                        <priority>1</priority>
                        <delivery_count>0</delivery_count>
                        <sender_id>
                             <agent_name>john</agent_name>
                        </sender_id>
                        <message_state>0</message_state>
                    </message_header>
                </message>
            </message_set>
        </AQXmlNotification>
```

```
    </Body>
</Envelope>
```

伝播機能

この項の内容は次のとおりです。

- 伝播
- 伝播スケジュール
- シナリオ
- 拡張伝播スケジュール機能
- 伝播中の例外処理
- 伝播中のメッセージ・フォーマットの変換

伝播

この機能によって、同じデータベースまたは同じキューに接続しなくても、アプリケーション同士が互いに通信できます。メッセージは、あるキューから別のキューに伝播できます。宛先キューは、同じデータベースまたはリモート・データベースに設定できます。伝播は、ジョブ・キュー・バックグラウンド・プロセスによって実行されます。リモート・キューへの伝播は、Oracle Net Services または HTTP(S) でデータベース・リンクを使用して行われます。

伝播機能は次のように使用されます。まず、メッセージ伝播の送信元のキューに、1 つ以上のサブスクライバが定義されます（8-34 ページの「サブスクリプションおよび受信者リスト」を参照）。次に、そのキューからメッセージを伝播する宛先ごとに、スケジュールが定義されます。エンキューされたメッセージは伝播されて、自動的に宛先キューでデキューできるようになります。

インターネット経由での伝播の場合、データベース・リンクにリモート・インターネット・ユーザーを指定する必要があります。リモート・インターネット・ユーザーには、宛先キューでエンキューする権限が必要です。

伝播を使用するためには、複数のジョブ・キュー・バックグラウンド・プロセスを実行中にする必要があります。その他に、伝播しない関連ジョブを扱うために必要な数のジョブ・キュー・バックグラウンド・プロセスが必要です。また、リモート伝播を実行する場合は、そのスケジュールのために指定されたデータベース・リンクが有効で、宛先キューにエンキューするための適切な権限を持っていることを確認する必要があります。伝播スケジュールを管理するための管理コマンドの詳細は、8-103 ページの「伝播スケジュール」を参照してください。

伝播には、障害に対処するためのメカニズムもあります。たとえば、指定されたデータベース・リンクが無効な場合、適切なエラー・メッセージがレポートされます。

最後に、伝播されたメッセージおよびそのスケジュールに関する詳細な統計を取得する機能があります。この情報は、最高のパフォーマンスが得られるように適切にスケジュールを調整するために使用できます。伝播の障害対処およびエラー・レポート機能および伝播統計については、8-109 ページの「拡張伝播スケジュール機能」を参照してください。

伝播スケジュール

伝播スケジュールは、1 組のソース（伝播元のキュー）および宛先データベース・リンクに対して定義されます。あるキューに複数のメッセージがあり、いくつかのキューに伝播されることになっている場合、宛先データベース・リンクごとにスケジュールを定義する必要があります。スケジュールは時間の枠を示し、メッセージはその枠内にソース・キューから伝播されます。この時間枠は、ネットワーク通信量、ソース・データベースの負荷、接続先データベースの負荷などの多くの要因によって左右されます。したがって、スケジュールはその特定のソースおよび宛先にあわせて作成する必要があります。スケジュールが作成されると、ジョブは自動的にジョブ・キューに発行され、伝播が処理されます。

伝播スケジュールのための管理コールによって、スケジュール管理を柔軟に行うことができます（9-72 ページの「キューの伝播のスケジューリング」を参照）。スケジュールの存続時間または伝播枠パラメータによって、伝播が開始される時間枠が指定されます。存続時間が指定されない場合、時間枠は無制限の単一枠になります。枠を定期的に繰り返す必要がある場合、連続する枠の間の周期的間隔を定義する `next_time` 関数を使用して有限の存続時間を指定します。

スケジュールの待機時間パラメータが関係するのは、キューに伝播する必要があるメッセージがないときのみです。このパラメータは、キューを再チェックする時間間隔を指定します。待機時間が 5 秒未満の場合、ジョブ・キュー・プロセス用の `job_queue_interval` パラメータは、待機時間パラメータ以下である必要があることに注意してください。

あるキューに定義された伝播スケジュールは、そのキューの有効期間中いつでも変更または削除できます。さらに、（スケジュールを削除するかわりに）一時的に使用不可にするコール、および使用不可のスケジュールを使用可能にするコールがあります。メッセージがスケジュール内で伝播されているとき、そのスケジュールはアクティブです。すべての管理コールは、スケジュールがアクティブかどうかに関係なく実行されます。アクティブなスケジュールでは、コールが実行されるまでに数秒かかります。

シナリオ

BooksOnLine の例では、`OE_bookedorders_que` にあるメッセージが別の出荷サイトに伝播されます。スケジュールを指定し管理するために利用できる様々な管理コールを、次のサンプル・コードで示します。また、ソース・キューにメッセージをエンキューしたり、宛先サイトでメッセージをデキューするためのコールも示します。カタログ・ビュー `USER_QUEUE_SCHEDULES` によって、任意のスケジュールに関連するすべての情報が得られます（10-25 ページの「ユーザー・スキーマの伝播スケジュールの選択」を参照）。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CONNECT OE/OE;

/* Schedule Propagation from bookedorders_que to shipping: */
EXECUTE dbms_aqadm.schedule_propagation( \
   queue_name      => 'OE.OE_bookedorders_que');

/* Check if a schedule has been created: */
SELECT * FROM user_queue_schedules;

/* Enqueue some orders into OE_bookedorders_que: */
EXECUTE BOLADM.order_enq('My First   Book', 1, 1001, 'CA', 'USA', \
   'WESTERN', 'NORMAL');
EXECUTE BOLADM.order_enq('My Second  Book', 2, 1002, 'NY', 'USA', \
   'EASTERN', 'NORMAL');
EXECUTE BOLADM.order_enq('My Third   Book', 3, 1003, '',   'Canada', \
   'INTERNATIONAL', 'NORMAL');
EXECUTE BOLADM.order_enq('My Fourth  Book', 4, 1004, 'NV', 'USA', \
   'WESTERN', 'RUSH');
EXECUTE BOLADM.order_enq('My Fifth   Book', 5, 1005, 'MA', 'USA', \
   'EASTERN', 'RUSH');
EXECUTE BOLADM.order_enq('My Sixth   Book', 6, 1006, ''  , 'UK', \
   'INTERNATIONAL', 'NORMAL');
EXECUTE BOLADM.order_enq('My Seventh Book', 7, 1007, '',   'Canada', \
   'INTERNATIONAL', 'RUSH');
EXECUTE BOLADM.order_enq('My Eighth  Book', 8, 1008, '',   'Mexico', \
   'INTERNATIONAL', 'NORMAL');
EXECUTE BOLADM.order_enq('My Ninth   Book', 9, 1009, 'CA', 'USA', \
   'WESTERN', 'RUSH');
EXECUTE BOLADM.order_enq('My Tenth   Book', 8, 1010, ''  , 'UK', \
   'INTERNATIONAL', 'NORMAL');
EXECUTE BOLADM.order_enq('My Last    Book', 7, 1011, ''  , 'Mexico', \
   'INTERNATIONAL', 'NORMAL');

/* Wait for propagation to happen: */
EXECUTE dbms_lock.sleep(100);

/* Connect to shipping sites and check propagated messages: */
CONNECT WS/WS;
set serveroutput on;

/*  Dequeue all booked orders for West_Shipping: */
EXECUTE BOLADM.shipping_bookedorder_deq('West_Shipping', DBMS_AQ.REMOVE);

CONNECT ES/ES;
SET SERVEROUTPUT ON;
```

```
/* Dequeue all remaining booked orders (normal order) for East_Shipping: */
EXECUTE BOLADM.shipping_bookedorder_deq('East_Shipping', DBMS_AQ.REMOVE);

CONNECT OS/OS;
SET SERVEROUTPUT ON;

/* Dequeue all international North American orders for Overseas_Shipping: */
EXECUTE BOLADM.get_northamerican_orders('Overseas_Shipping');

/* Dequeue rest of the booked orders for Overseas_Shipping: */
EXECUTE BOLADM.shipping_bookedorder_deq('Overseas_Shipping', DBMS_AQ.REMOVE);

/* Disable propagation schedule for booked orders
EXECUTE dbms_aqadm.disable_propagation_schedule(   \
   queue_name   => 'OE_bookedorders_que');

/* Wait for some time for call to be effected: */
EXECUTE dbms_lock.sleep(30);

/*  Check if the schedule has been disabled: */
SELECT schedule_disabled FROM user_queue_schedules;

/* Alter propagation schedule for booked orders to execute every
   15 mins (900 seconds) for a window duration of 300 seconds: */
EXECUTE dbms_aqadm.alter_propagation_schedule( \
   queue_name     => 'OE_bookedorders_que', \
   duration       => 300, \
   next_time      => 'SYSDATE + 900/86400',\
   latency        => 25);

/* Wait for some time for call to be effected: */
EXECUTE dbms_lock.sleep(30);

/*  Check if the schedule parameters have changed: */
SELECT next_time, latency, propagation_window FROM user_queue_schedules;

/* Enable propagation schedule for booked orders:
EXECUTE dbms_aqadm.enable_propagation_schedule( \
   queue_name     => 'OE_bookedorders_que');

/* Wait for some time for call to be effected: */
EXECUTE dbms_lock.sleep(30);

/* Check if the schedule has been enabled: */
SELECT schedule_disabled FROM user_queue_schedules;

/* Unschedule propagation for booked orders: */
```

```
EXECUTE dbms_aqadm.unschedule_propagation(   \
   queue_name      => 'OE.OE_bookedorders_que');

/* Wait for some time for call to be effected: */
EXECUTE dbms_lock.sleep(30);

/*  Check if the schedule has been dropped
SELECT *  FROM user_queue_schedules;
```

Visual Basic（OO4O）：サンプル・コード

この機能は、現在使用できません。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

LOB 属性を伴うメッセージの伝播

AQ を使用したラージ・オブジェクトの伝播には、2 通りの方法があります。

- RAW キューからの伝播。RAW キューでは、メッセージ・ペイロードはバイナリ・ラージ・オブジェクト（BLOB）として保存されます。これによって、PL/SQL インタフェースを使用したときには 32KB までのデータを格納でき、OCI を使用したときには、クライアントが同じ容量のデータを連続して割り当てることができます。この方法は、リリース 8.0.4 以上でサポートされています。
- LOB 属性を伴うオブジェクト・キューからの伝播。ユーザーは、Oracle の LOB 操作ルーチンを使用して、LOB を移入することも LOB から読み込むこともできます。LOB 属性は、BLOB または CLOB（NCLOB ではなく）です。属性が CLOB の場合、AQ はソース・キューと宛先キューの間で必要なすべてのキャラクタ・セット変換を自動的に実行します。この方法は、リリース 8.1.5 以上でサポートされています。

  参照：『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』を参照してください。

AQ では、ペイロードに BFILE または REF 属性があるオブジェクト・キューからの伝播はサポートされていないことに注意してください。

シナリオ

BooksOnLine アプリケーションでは、書籍注文とともに販売促進用の割引券を送ることがあります。割引券を送るかどうかは、注文内容およびその他の顧客情報によって決定されます。その割引券のイメージは、いくつかのマルチメディア・データベースで生成され、LOB として保存されます。

注文情報が出荷倉庫に送られるときに、割引券の内容も倉庫に送られます。次に示すコードでは、LOB 型の割引券属性を含むように order_typ が拡張されています。このコードは、注文が発生したときに OE_bookedorders_que にエンキューされるメッセージに LOB 内容を挿入する方法を示しています。メッセージ・ペイロードは、最初に空の LOB によって組み立てられます。プレース・ホルダ（LOB ロケータ）情報はキュー表から取得され、DBMS_LOB.WRITE() などの LOB 操作ルーチンと組み合せて LOB 内容を充填するために使用されます。さらに、ペイロードの一部に LOB があるメッセージのエンキューおよびデキューに関する例も示します。

COMMIT は、LOB 内容が適切なイメージ・データによって充填されてから発行されます。伝播は、LOB 内容をそれ以外のメッセージ内容とともに自動的に移動させる処理をします。次のコードには、宛先キューで伝播されたメッセージの LOB 内容を読み込むデキューについても示します。LOB 内容は、バッファに読み込まれ、そこから割引券を印刷するためにプリンタに送られることもあります。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/* Enhance the type order_typ to contain coupon field (lob field): */
CREATE OR REPLACE TYPE order_typ AS OBJECT (
        orderno         NUMBER,
        status          VARCHAR2(30),
        ordertype       VARCHAR2(30),
        orderregion     VARCHAR2(30),
        customer        customer_typ,
        paymentmethod   VARCHAR2(30),
        items           orderitemlist_vartyp,
        total           NUMBER,
        coupon          BLOB);
/

/* lob_loc is a variable of type BLOB,
   buffer is a variable of type RAW,
   length is a variable of type NUMBER. */

/* Complete the order data and perform the enqueue using the order_enq()
   procedure: */
dbms_aq.enqueue('OE.OE_bookedorders_que', enqopt, msgprop,
                OE_enq_order_data, enq_msgid);

/* Get the lob locator in the queue table after enqueue: */
SELECT t.user_data.coupon INTO lob_loc
```

```
FROM   OE.OE_orders_pr_mqtab t
WHERE  t.msgid = enq_msgid;

/* Generate a sample LOB of 100 bytes: */
buffer := hextoraw(rpad('FF',100,'FF'));

/* Fill in the lob using LOB routines in the dbms_lob package: */
dbms_lob.write(lob_loc, 90, 1, buffer);

/* Issue a commit only after filling in lob contents: */
COMMIT;

/* Sleep until propagation is complete: */

/* Perform dequeue at the Western Shipping warehouse: */
dbms_aq.dequeue(
        queue_name         => qname,
        dequeue_options    => dopt,
        message_properties => mprop,
        payload            => deq_order_data,
        msgid              => deq_msgid);

/* Get the LOB locator after dequeue: */
lob_loc := deq_order_data.coupon;

/* Get the length of the LOB: */
length := dbms_lob.getlength(lob_loc);

/* Read the LOB contents into the buffer: */
dbms_lob.read(lob_loc, length, 1, buffer);
```

Visual Basic（OO4O）：サンプル・コード

この機能は、現在使用できません。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

拡張伝播スケジュール機能

スケジュールに関する詳細情報は、伝播のために定義されたカタログ・ビューから取得できます。アクティブ・スケジュールに関する情報、たとえば、そのスケジュールを処理しているバックグラウンド・プロセス名、伝播を処理するセッションの SID（セッションのシリアル番号）、スケジュールを処理する Oracle インスタンス（Real Application Clusters が使用されている場合）などの情報は、そのカタログ・ビューから取得できます。同じカタログ・ビューによって、先行して正常に実行されたスケジュール（最後に正常に伝播されたメッセージ）および次に実行されるスケジュールに関する情報も取得できます。

各スケジュールには、次の伝播の詳細情報が保持されます。

- スケジュールの中で伝播された合計メッセージ数
- スケジュールの中で伝播された合計バイト数
- 伝播枠の中で伝播されたメッセージの最大数
- 伝播枠の中で伝播されたバイトの最大値
- 伝播枠の中で伝播されたメッセージの平均数
- 伝播済メッセージの平均サイズ
- 伝播済メッセージの平均時間

これには、スケジュールの中で伝播されたメッセージの合計数とバイトの合計数、伝播枠の中で伝播されたメッセージの最大数と平均数、伝播されたメッセージの平均サイズと平均時間が含まれます。このような統計は、キュー管理者が最も効率的なスケジュール調整に役立てることができるように設計されています。

伝播機能には、障害対処およびエラー・レポートが組み込まれています。たとえば、指定されたデータベース・リンクが無効な場合、リモート・データベースが使用できない場合、またはリモート・キューにエンキューできない場合、適切なエラー・メッセージがレポートされます。伝播は指数バックオフ・スキームを使用して、障害が発生したスケジュールからの伝播を再試行します。

あるスケジュールが続けて障害を発生したときは、最初の再試行は 30 秒後、次の再試行は 60 秒後、3 回目の再試行は 120 秒後、というように続きます。再試行時間が現行の伝播枠の期限切れ時刻を超える場合は、次の再試行は、次の伝播枠の開始時刻に行われます。最大 16 回の再試行が行われた後、そのスケジュールは自動的に使用不可になります。障害のためにスケジュールが自動的に使用不可になると、関連情報がアラート・ログに書き込まれます。

スケジュール障害の確認項目は次のとおりです。

- 連続的な障害の発生回数
- 障害の原因を示すエラー・メッセージ
- 直前の障害の発生時刻

この情報を調べることで、キュー管理者は障害を回復し、スケジュールを使用可能にできます。再試行の間に伝播が成功したときは、障害の数は 0（ゼロ）にリセットされます。

伝播機能には Oracle Real Application Clusters サポートが組み込まれていますが、ユーザーおよびキュー管理者には透過的です。伝播を処理するジョブは、キューが常駐しているキュー表の所有者と同じインスタンスに送られます。

あるインスタンスに障害があって、キューを保存しているキュー表が他のインスタンスに移行される場合は、伝播ジョブも必ず新しいインスタンスに移されます。これによって、インスタンス間の ping 操作は最小限に抑えられ、パフォーマンスが向上します。伝播は、同時スケジュールをいくつでも処理できるように設計されています。ジョブ・キュー・プロセスの数は、最大 1000 に制限され、その中のいくつかは伝播以外の関連ジョブを処理するために使用されることに注意してください。このように、伝播にはマルチタスキングおよびロード・バランスのサポートが組み込まれています。

伝播アルゴリズムは、複数スケジュールが単一スナップショット（ジョブ・キュー）・プロセスによって処理できるように設計されています。ジョブ・キュー・プロセスに対する伝播の負荷は、異なるソース・キューからのメッセージ到着割合に基づいて偏りが発生する場合があります。

あるプロセスが数個のアクティブ・スケジュールによって過負荷になっている一方で、別のプロセスは受動的なスケジュールが多いために余力があるというとき、伝播は負荷が均等になるようにスケジュールを自動的に再分散します。

シナリオ

BooksOnLine の例では、OE_bookedorders_que は、そこに含まれるメッセージが別の出荷サイトに伝播されるため、ビジー・キューになります。エラー・チェックおよびスケジュール監視のための拡張伝播スケジュールをサポートしているコールを、次のサンプル・コードで示します。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
CONNECT OE/OE;

/*  get averages
select avg_time, avg_number, avg_size from user_queue_schedules;

/*  get totals
select total_time, total_number, total_bytes from user_queue_schedules;

/*  get maximums for a window
select max_number, max_bytes from user_queue_schedules;

/*  get current status information of schedule
select process_name, session_id, instance, schedule_disabled
   from user_queue_schedules;
```

```
/*  get information about last and next execution
select last_run_date, last_run_time, next_run_date, next_run_time
   from user_queue_schedules;

/*  get last error information if any
select failures, last_error_msg, last_error_date, last_error_time
   from user_queue_schedules;
```

Visual Basic（OO4O）：サンプル・コード

この機能は、現在使用できません。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

伝播中の例外処理

ネットワーク障害のようなシステム・エラーが発生した場合、アドバンスト・キューイングは指数バックオフ・アルゴリズムを使用してメッセージを伝播する試みを継続します。状況がアプリケーション・エラーを示しているとき、メッセージの伝播でエラーが発生した場合は、AQ はそのメッセージに UNDELIVERABLE というマークを付けます。

このようなエラーは、リモート・キューが存在しない場合、またはソース・キューおよびリモート・キューの型が一致しない場合に発生します。このような状況では、ユーザーは DBA_SCHEDULES ビューを問い合せて、特定の宛先向けの伝播中に発生した最新のエラーを調べます。$ORACLE_HOME/log ディレクトリのトレース・ファイルには、そのエラーに関する追加情報があります。

シナリオ

BooksOnLine の例では、東部向け出荷の ES_bookedorders_que が stop_queue() コールを使用して故意に中止されます。少し時間が経過すると、OE_bookedorders_que の伝播スケジュールには、リモート・キュー ES_bookedorders_que にはエンキューできないというエラーが表示されます。start_queue() コールを使用して ES_bookedorders_que を開始すると、そのキューへの伝播が再開され、OE_bookedorders_que のスケジュールに関連したエラー・メッセージはなくなります。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
/*  Intentionally stop the eastern shipping queue : */
connect BOLADM/BOLADM
EXECUTE dbms_aqadm.stop_queue(queue_name => 'ES.ES_bookedorders_que');

/* Wait for some time before error shows up in dba_queue_schedules: */
EXECUTE dbms_lock.sleep(100);

/* This query will return an ORA-25207 enqueue failed error: */
SELECT qname, last_error_msg from dba_queue_schedules;

/* Start the eastern shipping queue: */
EXECUTE dbms_aqadm.start_queue(queue_name => 'ES.ES_bookedorders_que');

/* Wait for Propagation to resume for eastern shipping queue: */
EXECUTE dbms_lock.sleep(100);

/* This query will indicate that there are no errors with propagation:
SELECT qname, last_error_msg from dba_queue_schedules;
```

Visual Basic（OO4O）：サンプル・コード

この機能は、データベースでは処理されません。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

伝播中のメッセージ・フォーマットの変換

伝播時には、西部向け出荷注文の OE_bookedorders_topic に対してルール・ベースのサブスクライバを追加するときに、変換を指定できます。変換は注文に対して適用され、注文は、WS_bookedorders_topic に伝播される前に WS.order_typ_sh 型に変換されます。

PL/SQL（DBMS_AQADM）：サンプル・コード

```
declare
subscriber     sys.aq$_agent;
begin
  subscriber :=sys.aq$_agent('West_Shipping','WS.WS_bookedorders_topic',null);
  dbms_aqadm.add_subscriber(queue_name => 'OE.OE_bookedorders_topic',
       subscriber     => subscriber,
       rule           => 'tab.user_data.orderregion =''WESTERN''
                          AND tab.user_data.ordertype != ''RUSH''',
       transformation => 'OE.OE2WS');
end;
```

Visual Basic（OO4O）：サンプル・コード

今回のリリースでは、例は記載していません。

Java（JDBC）：サンプル・コード

今回のリリースでは、例は記載していません。

HTTP を使用した伝播

Oracle9*i* では、HTTP および HTTPS（SSL による HTTP）経由のアドバンスト・キューイング伝播を設定できます。HTTP 伝播では、インターネット・アクセス・インフラストラクチャが使用され、接続先データベースに接続する AQ サーブレットを配置する必要があります。データベース・リンクは、接続文字列に Web サーバーのアドレスとポートを指定し、HTTP をプロトコルとすることを指定して作成する必要があります。Java および XML を実行するためのソース・データベースを作成する必要があります。作成しないと、HTTP 伝播の設定は、Oracle Net Services（以前の Net8）伝播の設定と同じになります。

シナリオ

BooksOnLine の例では、`OE_bookedorders_que` にあるメッセージが別の出荷サイトに伝播されます。このシナリオでは、西部向け出荷処理アプリケーションが別のデータベース `dest-db` で実行しており、メッセージを `WS_bookedorders_que` に伝播するとします。

伝播の設定

1. AQ サーブレットを配置します。

   HTTP 伝播は、接続先データベースへのインターネット・アクセスに依存します。AQxmlServlet を拡張する `AQPropServlet` クラスを作成します。

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.AQ.*;
import oracle.AQ.xml.*;
import java.sql.*;
import oracle.jms.*;
import javax.jms.*;
import java.io.*;
import oracle.jdbc.pool.*;

/* This is an AQ Propagation Servlet. */
public class AQPropServlet extends oracle.AQ.xml.AQxmlServlet

/* getDBDrv - specify the database to which the servlet will connect */
public AQxmlDataSource createAQDataSource() throws AQxmlException
{
```

```
  AQxmlDataSource  db_drv = null;
  db_drv = new AQxmlDataSource("aqadm", "aqadm", "dest-db", "dest-host",
      5521);
    return db_drv;
  }

  public void init()
  {
      try {
        AQxmlDataSource axds = this.createAQDataSource();
        setAQDataSource(axds) ;
        setSessionMaxInactiveTime(180) ;

      } catch (Exception e) {
         System.err.println("Error in init : " +e) ;
      }
  }
}
```

このサーブレットは、接続先データベースに接続する必要があります。サーブレットは、Web サーバーのパス `aqserv/servlet` に配置される必要があります。Oracle9*i* では、伝播のサーブレット名は `AQPropServlet`、配置パスは `aqserv/servlet` に固定されています。

Web サーバーのホストが `webdest.oracle.com`、ポートが `8081` であると想定します。

2. データベース・リンク dba を作成します。

   - HTTP をプロトコルとして指定します。
   - 認証に使用されるユーザー名およびパスワードを指定し、Web サーバー / サーブレット・コンテナを、AQ サーブレットを実行する Web サーバーのホストおよびポートとして指定します。

   この例では、データベース・リンクの接続文字列は次のようになります。

   ```
   (DESCRIPTION=(ADDRESS=(PROTOCOL=http)(HOST=webdest.oracle.com)(PORT=8081)))
   ```

   SSL を使用する場合、HTTPS をプロトコルとして接続文字列に指定します。

   次のとおり入力して、データベース・リンクを作成します。

   ```
   create public database link dba connect to john identified by welcome
   using
   '(DESCRIPTION=(ADDRESS=(PROTOCOL=http)(HOST=webdest.oracle.com)(PORT=8081)))';
   ```

   SSL を使用する場合、接続文字列に HTTPS をプロトコルとして指定します。

   次のとおり入力して、データベース・リンクを作成します。

```
create public database link dba connect to john identified by welcome
using
'(DESCRIPTION=(ADDRESS=(PROTOCOL=http)(HOST=webdest.oracle.com)(PORT=8081)))';
```

john は、AQ（伝播）サーブレットにアクセスするために使用される AQ HTTP エージェントです。Welcome は、Web サーバーでの認証に使用されるパスワードです。

3. AQ HTTP エージェントである John に、AQ 操作の実行権限があることを確認します。接続先データベースで、次の操作を実行します。

   a. AQ エージェントを登録します。

```
dbms_aqadm.create_aq_agent(agent_name => 'John', enable_http => true);
```

   b. AQ エージェントをデータベース・ユーザーにマップします。

```
dbms_aqadm.enable_db_access(agent_name =>'John', db_username =>'CBADM')'
```

4. リモート・サブスクリプションを OE.OE_bookedorders_que に設定します。

```
execute dbms_aqadm.add_subscriber('OE.OE_bookedorders_que',
aq$_agent(null, 'WS.WS_bookedorders_que', null));
```

5. ソース・データベースで dbms_aqdm.schedule_propagation をコールすることによって、伝播を開始します。

```
dbms_aqadm.schedule_propagation('OE.OE_bookedorders_que', 'dba');
```

他のすべての伝播管理 API は、HTTP 伝播と同様に機能します。DBA_QUEUE_SCHEDULES 伝播ビューを使用して、HTTP を使用した伝播スケジュールの伝播統計を確認します。

9

管理インタフェース

この章では、Oracle Advanced Queuing 用の管理インタフェースについて説明します。それぞれの操作（キュー表の作成など）を、その操作名ごとに利用方法に沿って説明します。個々の利用方法は、次の形式で説明されています。

- **利用図 :** 利用方法を表す図
- **用途 :** この利用方法の用途
- **使用上の注意 :** 実装に有効なガイドライン
- **構文 :** このアクティビティの実行に使用する主な構文
- **例 :** 各プログラム環境での利用例

利用モデル : 管理インタフェース - 基本操作

表 9-1「利用モデル : 管理インタフェース - 基本操作」の「+」は、その利用方法で、プログラム環境の例が記載されていることを示します。

この表では、プログラム環境を次の略称で表しています。

- **P** - DBMS_AQADM および DBMS_AQ パッケージを使用した PL/SQL
- **V** - Oracle Objects for OLE（OO4O）を使用した Visual Basic
- **J** - Java Databasae Connectivity（JDBC）を使用した Java（ネイティブ AQ）
- **JMS** - Java Databasae Connectivity（JDBC）を使用した Java（JMS 標準）

表 9-1 利用モデル : 管理インタフェース - 基本操作

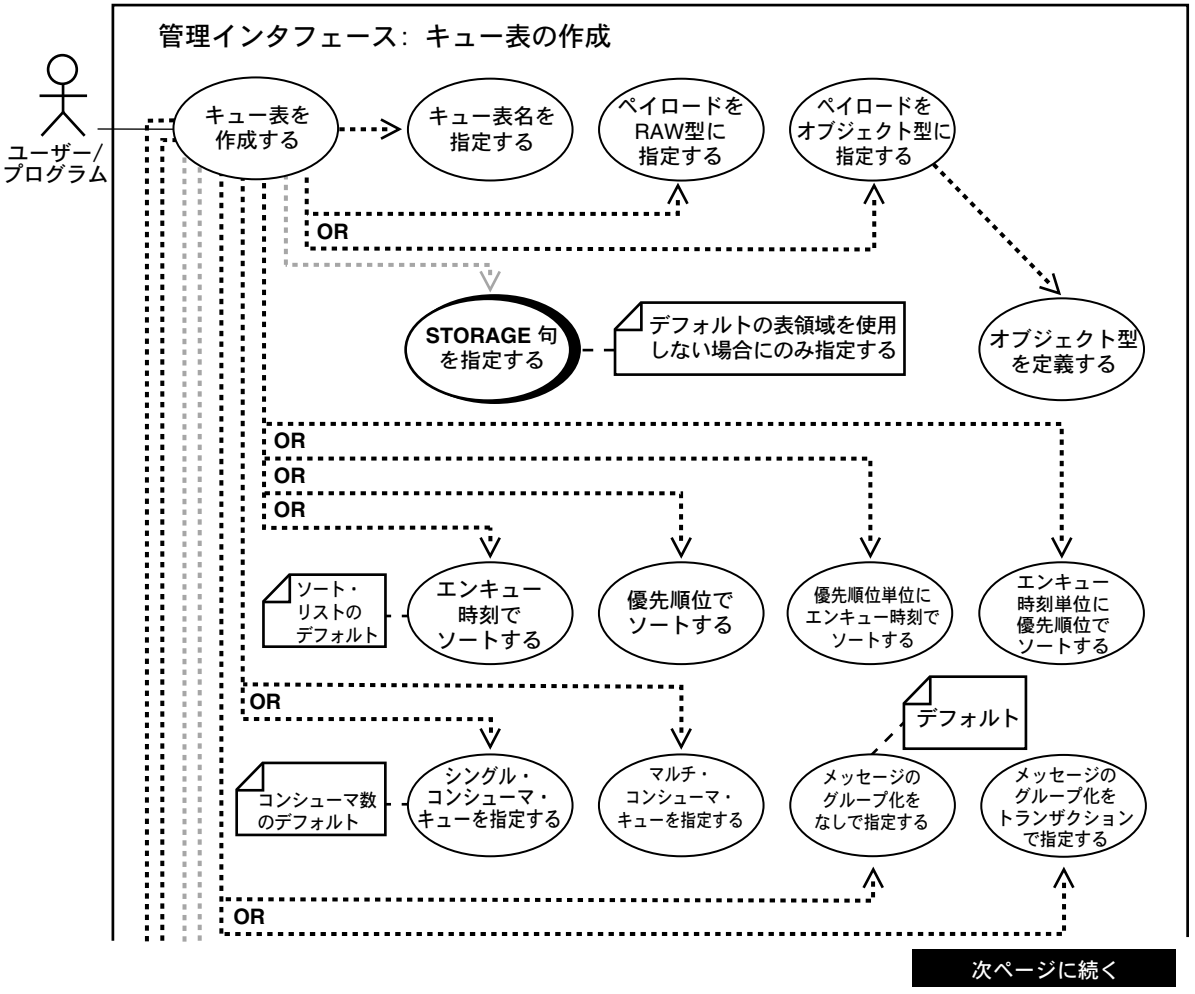
利用方法	P	V	J	JMS
キュー表の作成（9-4 ページ）	+	+	+	-
キュー表の作成（STORAGE 句の設定）（9-13 ページ）	+	-	+	-
キュー表の変更（9-15 ページ）	+	-	+	-
キュー表の削除（9-18 ページ）	+	-	+	-
キューの作成（9-21 ページ）	+	-	+	-
非永続キューの作成（9-27 ページ）	+	-	-	-
キューの変更（9-29 ページ）	+	-	+	-
キューの削除（9-32 ページ）	+	-	+	-
変換の作成（9-35 ページ）	+	-	-	-
変換の変更（9-38 ページ）	+	-	-	-
変換の適用（9-40 ページ）	+	-	-	-
変換の削除（9-41 ページ）	+	-	-	-
キューの開始（9-43 ページ）	+	-	+	-
キューの停止（9-46 ページ）	+	-	+	-
システム権限の付与（9-49 ページ）	+	-	+	-
システム権限の取消し（9-52 ページ）	+	-	-	-
キュー権限の付与（9-54 ページ）	+	-	+	-
キュー権限の取消し（9-56 ページ）	+	-	+	-

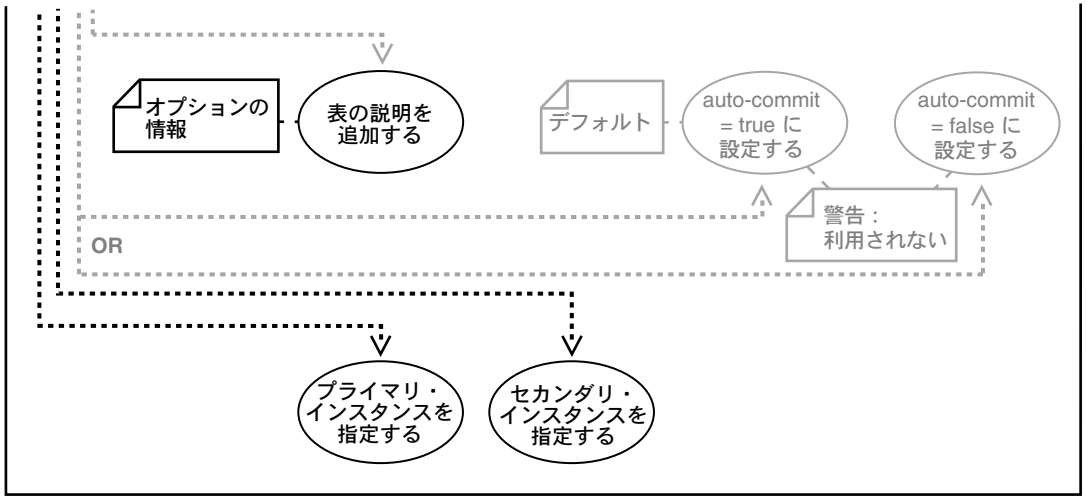
表 9-1 利用モデル : 管理インタフェース - 基本操作（続き）

利用方法	P	V	J	JMS
サブスクライバの追加（9-59 ページ）	+	-	+	-
サブスクライバの変更（9-65 ページ）	+	-	+	-
サブスクライバの削除（9-69 ページ）	+	-	+	-
キューの伝播のスケジューリング（9-72 ページ）	+	-	+	-
キューの伝播スケジュールの解除（9-76 ページ）	+	-	+	-
キュー・タイプの検証（9-79 ページ）	+	-	-	-
伝播スケジュールの変更（9-82 ページ）	+	-	+	-
伝播スケジュールの使用可能化（9-85 ページ）	+	-	+	-
伝播スケジュールの使用不可能化（9-88 ページ）	+	-	+	-
AQ エージェントの作成（9-91 ページ）	+	-	-	-
AQ エージェントの変更（9-93 ページ）	+	-	-	-
AQ エージェントの削除（9-95 ページ）	+	-	-	-
データベース・アクセスの許可（9-97 ページ）	+	-	-	-
データベース・アクセスの禁止（9-99 ページ）	+	-	-	-
LDAP サーバーへの別名の追加（9-101 ページ）	+	-	-	-
LDAP サーバーからの別名の削除（9-103 ページ）	+	-	-	-

キュー表の作成

図 9-1 キュー表の作成





参照：

- 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。
- 9-13 ページの「キュー表の作成（STORAGE 句の設定）」も参照してください。

用途

事前定義された型のメッセージを持つキュー表を作成します。

使用上の注意

- キュー名およびキュー表名は、大文字に変換されます。大文字と小文字の組合せはサポートされていません。
- デキュー順序付け用のソート・キーを使用する場合は、表の作成時に定義する必要があります。この時点では、次のオブジェクトが作成されます。
  - キュー表に対応付けられた aq$_<queue_table_name>_e という名前のデフォルトの例外キュー
  - AQ アプリケーションでキュー・データの問合せに使用される aq$<queue_table_name> という名前の読込み専用ビュー
  - aq$_<queue_table_name>_t という名前のキュー・モニター操作の索引
  - マルチ・コンシューマ・キューの場合に使用される、aq$_<queue_table_name>_i という名前のデキュー操作用の索引または索引構成表（IOT）
- 8.1 互換のマルチ・コンシューマ・キュー表の場合、次の追加オブジェクトが作成されます。
  - aq$_<queue_table_name>_s という名前の表。この表にはサブスクライバの情報が格納されます。
  - aq$_<queue_table_name>_r という名前の表。この表にはサブスクリプション・ルールの情報が格納されます。
  - aq$_<queue_table_name>_h という名前の索引構成表（IOT）。この表にはデキュー履歴データが格納されます。
- AQ メッセージでは、CLOB、BLOB または BFILE オブジェクトが有効です。Oracle リリース 8.1.x 以上では、AQ 伝播を使用してこれらのオブジェクト型を伝播できます。LOB を持つオブジェクト型をエンキューするには、LOB_attribute を EMPTY_BLOB() に設定してからエンキューを実行する必要があります。これによって、キュー表のビューから生成された LOB ロケータを選択でき、標準の LOB 操作を使用できます。詳細は、『Oracle9*i* アプリケーション開発者ガイド - ラージ・オブジェクト』を参照してください。
- 8.1 互換モードの場合のみ、primary_instance および secondary_instance を指定および変更できます。
- プライマリ・インスタンスがない場合は、セカンダリ・インスタンスを指定できません。
- キュー、キュー表またはサブスクライバが、作成、変更または削除された場合、GLOBAL_TOPIC_ENABLED が TRUE であれば、対応する LDAP エントリも作成されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の CREATE_QUEUE_TABLE プロシージャ
- Visual Basic（OO4O）：参照マニュアルはありません。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQSession」の createQueueTable

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

- PL/SQL（DBMS_AQADM）：キュー表の作成（9-7 ページ）
- Visual Basic（OO4O）：キュー表の作成（9-9 ページ）
- Java（JDBC）：キュー表の作成（9-10 ページ）

PL/SQL（DBMS_AQADM）：キュー表の作成

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
CONNECT system/manager;
DROP USER aqadm CASCADE;
GRANT CONNECT, RESOURCE TO aqadm;
CREATE USER aqadm IDENTIFIED BY aqadm;
GRANT EXECUTE ON DBMS_AQADM TO aqadm;
GRANT Aq_administrator_role TO aqadm;
DROP USER aq CASCADE;
CREATE USER aq IDENTIFIED BY aq;
GRANT CONNECT, RESOURCE TO aq;
GRANT EXECUTE ON dbms_aq TO aq;
```

オブジェクト型のメッセージを持つキューのキュー表を作成する

```
CREATE type aq.Message_typ as object (
   Subject               VARCHAR2(30),
   Text                  VARCHAR2(80));

/* Note: if you do not stipulate a schema, you default to the user's schema. */
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table           => 'aq.ObjMsgs_qtab',
   Queue_payload_type    => 'aq.Message_typ');
```

RAW 型のメッセージを持つキューのキュー表を作成する

```
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table           => 'aq.RawMsgs_qtab',
   Queue_payload_type    => 'RAW');
```

XMLType のメッセージを持つキューのキュー表を作成する

```
execute dbms_aqadm.create_queue_table(
        queue_table     => 'OS_orders_pr_mqtab',
        comment         => 'Overseas Shipping MultiConsumer Orders queue table',
        multiple_consumers  => TRUE,
        queue_payload_type  => 'SYS.XMLType',
        compatible          => '8.1');
```

優先メッセージのキュー表を作成する

```
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table           => 'aq.PriorityMsgs_qtab',
   Sort_list             => 'PRIORITY,ENQ_TIME',
   Queue_payload_type    => 'aq.Message_typ');
```

マルチ・コンシューマのキュー表を作成する

```
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table           => 'aq.MultiConsumerMsgs_qtab',
   Multiple_consumers    => TRUE,
   Queue_payload_type    => 'aq.Message_typ');
```

8.1 互換のマルチ・コンシューマのキュー表を作成する

```
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table            => 'aq.Multiconsumermsgs8_1qtab',
   Multiple_consumers     =>  TRUE,
   Compatible             => '8.1',
   Queue_payload_type     => 'aq.Message_typ');
```

指定された表領域にキュー表を作成する

```
EXECUTE dbms_aqadm.create_queue_table(
        queue_table        => 'aq.aq_tbsMsg_qtab',
        queue_payload_type => 'aq.Message_typ',
        storage_clause     => 'tablespace aq_tbs');
```

空きリストまたは空きリスト・グループがあるキュー表を作成する

```
BEGIN
dbms_aqadm.create_queue_table (
queue_table=> 'AQ_ADMIN.TEST',
queue_payload_type=> 'RAW',
storage_clause=> 'STORAGE (FREELISTS 4 FREELIST GROUPS 2)',
compatible => '8.1');
COMMIT;
END;
```

Visual Basic（OO4O）：キュー表の作成

OO4O は、この操作に対してデータベース機能を使用します。

Java（JDBC）：キュー表の作成

Java を使用したキュー表の作成方法の 3 つの例を、次に示します。

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
CONNECT system/manager;
DROP USER aqadm CASCADE;
CREATE USER aqadm IDENTIFIED BY aqadm;
GRANT CONNECT, RESOURCE TO aqadm;
GRANT EXECUTE ON DBMS_AQADM TO aqadm;
GRANT Aq_administrator_role TO aqadm;
DROP USER aq CASCADE;
CREATE USER aq IDENTIFIED BY aq;
GRANT CONNECT, RESOURCE TO aq;
GRANT EXECUTE ON dbms_aq TO aq;

CREATE type aq.Message_typ as object (
   Subject                VARCHAR2(30),
   Text                   VARCHAR2(80));
```

オブジェクト型のメッセージを持つキューのキュー表を作成する

```
public static void example(AQSession aq_sess) throws AQException
{
    AQQueueTableProperty     qtable_prop;
    AQQueueProperty          queue_prop;
    AQQueueTable             q_table;
    AQQueue                  queue;

    /* Create a AQQueueTableProperty object (payload type Message_typ): */
    qtable_prop = new AQQueueTableProperty("AQ.MESSAGE_TYP");

    /* Create a queue table in aq schema */
    q_table = aq_sess.createQueueTable ("aq", "ObjMsgs_qtab", qtable_prop);

    System.out.println("Successfully created ObjMsgs_qtab in aq schema");
}
```

RAW 型のメッセージを持つキューのキュー表を作成する

```
public static void example(AQSession aq_sess) throws AQException
{
    AQQueueTableProperty     qtable_prop;
    AQQueueProperty          queue_prop;
    AQQueueTable             q_table;
    AQQueue                  queue;

    /* Create a AQQueueTableProperty object (payload type RAW): */
    qtable_prop = new AQQueueTableProperty("RAW");

    /* Create a queue table in aq schema */
    q_table = aq_sess.createQueueTable ("aq", "RawMsgs_qtab", qtable_prop);

    System.out.println("Successfully created RawMsgs_qtab in aq schema");
}
```

マルチ・コンシューマおよび優先メッセージのキュー表を作成する

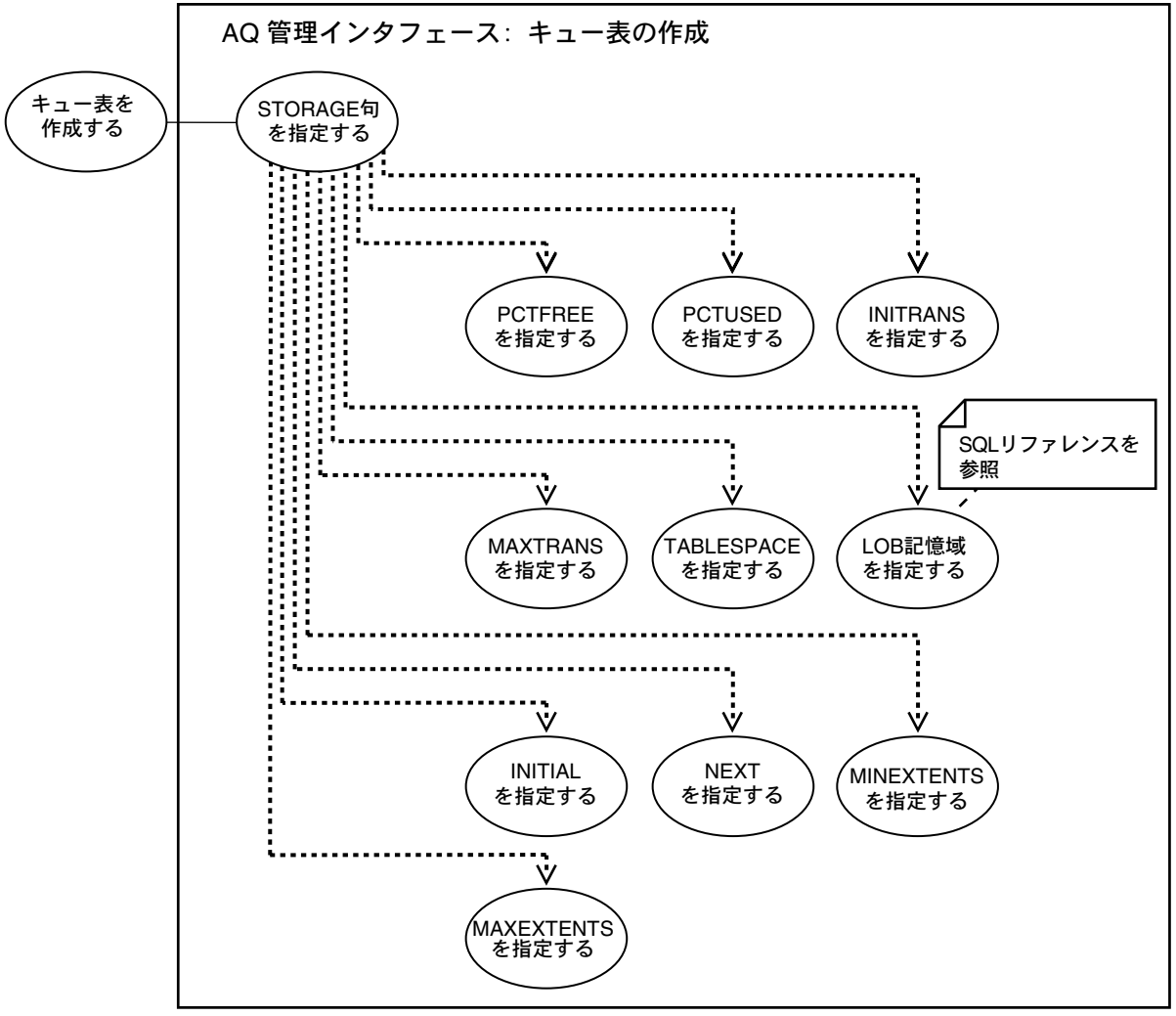
```
public static void example(AQSession aq_sess) throws AQException
{
    AQQueueTableProperty     qtable_prop;
    AQQueueProperty          queue_prop;
    AQQueueTable             q_table;
    AQQueue                  queue;

    qtable_prop = new AQQueueTableProperty("RAW");

    /* Enable multiple consumers */
    qtable_prop.setMultiConsumer(true);
    qtable_prop.setCompatible("8.1");

    /* Specify sort order as priority,enqueue_time */
    qtable_prop.setSortOrder("PRIORITY,ENQ_TIME");

    /* Create a queue table in aq schema */
    q_table = aq_sess.createQueueTable ("aq", "PriorityMsgs_qtab",
qtable_prop);

    System.out.println("Successfully created PriorityMsgs_qtab in aq schema");
}
```

指定された表領域にキュー表を作成する

```
public static void example(AQSession aq_sess) throws AQException
{
    AQQueueTableProperty     qtable_prop;
    AQQueueProperty          queue_prop;
    AQQueueTable             q_table;
    AQQueue                  queue;

    /* Create a AQQueueTableProperty object (payload type Message_typ): */
    qtable_prop = new AQQueueTableProperty("AQ.MESSAGE_TYP");

    /* Specify tablespace for queue table */
    qtable_prop.setStorageClause("tablespace aq_tbs");

    /* Create a queue table in aq schema */
    q_table = aq_sess.createQueueTable ("aq", "aq_tbsMsg_qtab", qtable_prop);
}
```

キュー表の作成（STORAGE 句の設定）

図 9-2 キュー表の作成（STORAGE 句の設定）



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

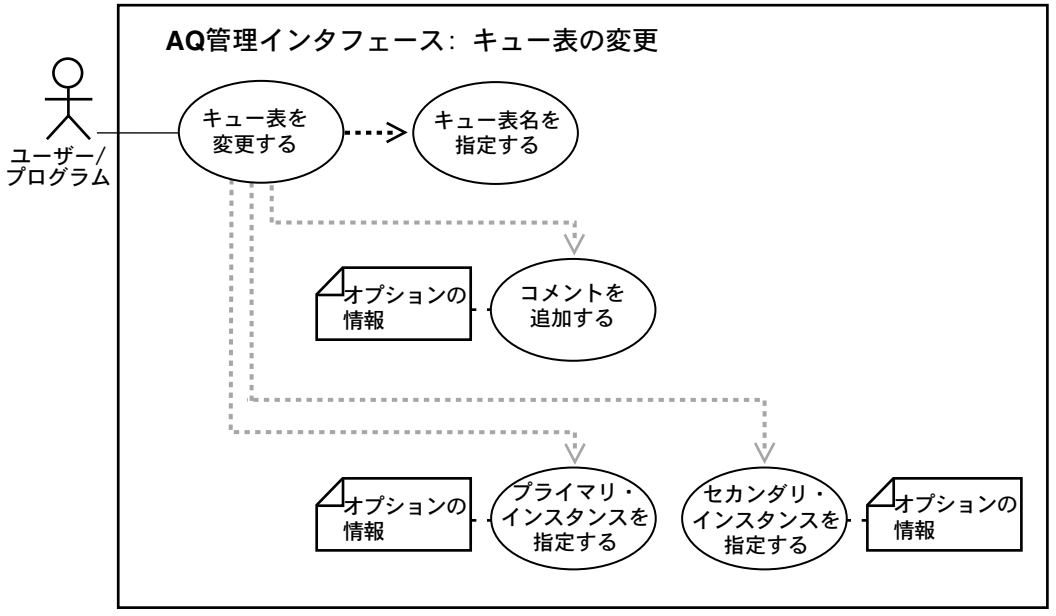
構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の `CREATE_QUEUE_TABLE` プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQSession」の `createQueueTable`

キュー表の変更

図 9-3 キュー表の変更



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

キュー表の既存のプロパティを変更します。

使用上の注意

キュー、キュー表またはサブスクライバが、作成、変更または削除された場合、GLOBAL_TOPIC_ENABLED が TRUE であれば、対応する LDAP エントリも作成されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の ALTER_QUEUE_TABLE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQQueueAdmin」の alterQueue

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キュー表の変更（9-16 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キュー表の変更（9-17 ページ）

PL/SQL（DBMS_AQADM）: キュー表の変更

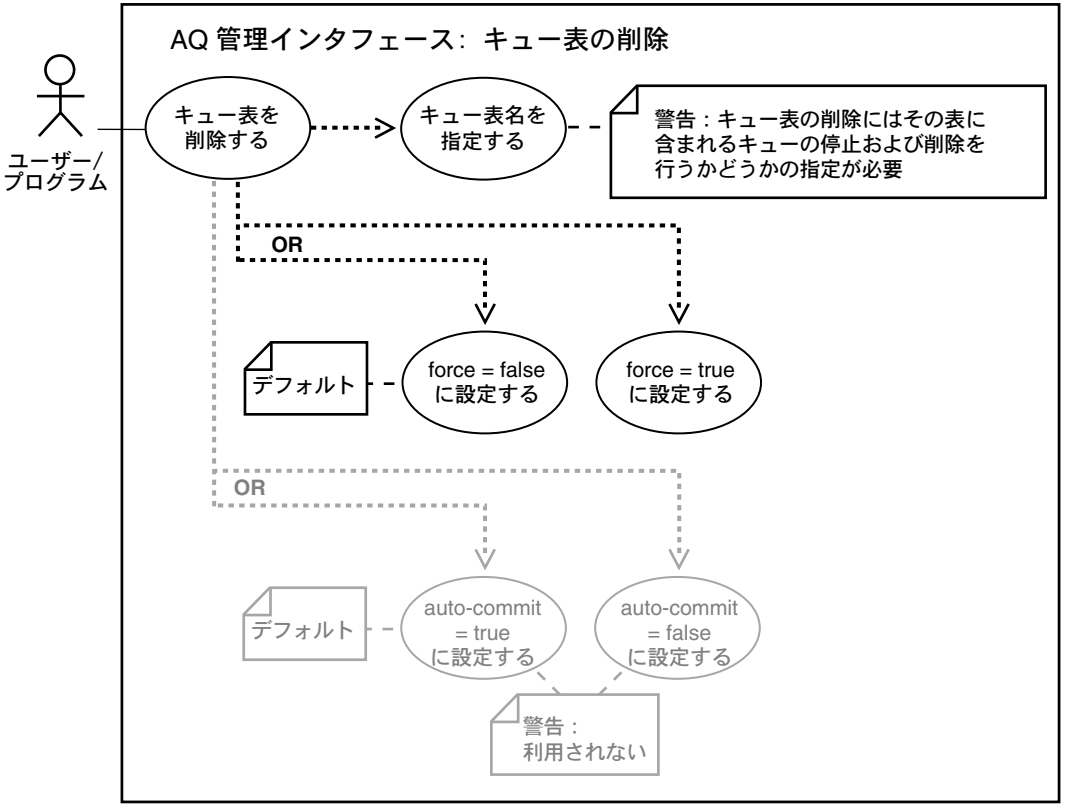
```
/* Altering the table to change the primary, secondary instances for queue owner
(only applies to Real Application Clusters environments). The primary instance is
the instance number of the primary owner of the queue table. The secondary instance
is the instance number of the secondary owner of the queue table. */
 EXECUTE dbms_aqadm.alter_queue_table (
   Queue_table          => 'aq.ObjMsgs_qtab',
   Primary_instance     => 3,
   Secondary_instance   => 2);

/* Altering the table to change the comment for a queue table: */
EXECUTE dbms_aqadm.alter_queue_table (
   Queue_table          => 'aq.ObjMsgs_qtab',
   Comment              => 'revised usage for queue table');

/* Altering the table to change the comment for a queue table and use
nonrepudiation: */
EXECUTE dbms_aqadm.alter_queue_table (
   Queue_table          => 'aq.ObjMsgs_qtab',
   Comment              => 'revised usage for queue table',
```

Java（JDBC）：キュー表の変更

```
/* Alter a queue table */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueueTableProperty    qtable_prop;
     AQQueueTable            q_table;

     q_table = aq_sess.getQueueTable ("aq", "ObjMsgs_qtab");

     /* Get queue table properties: */
     qtable_prop = q_table.getProperty();

     /* Alter the queue table comment and instance affinity */
     q_table.alter("altered queue table", 3, 2);
}
```

キュー表の削除

図 9-4 キュー表の削除



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

既存のキュー表を削除します。キュー表を削除する前に、そのキュー表内のすべてのキューを停止する必要があることに注意してください。`force` オプションを使用して自動的に行わないかぎり、この操作は明示的に行う必要があります。

使用上の注意

キュー、キュー表またはサブスクライバが、作成、変更または削除された場合、GLOBAL_TOPIC_ENABLED が TRUE であれば、対応する LDAP エントリも作成または削除されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の DROP_QUEUE_TABLE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQQueueTable」の drop

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キュー表の削除（9-19 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キュー表の削除（9-20 ページ）

PL/SQL（DBMS_AQADM）: キュー表の削除

```
/* Drop the queue table (for which all queues have been previously dropped by
   the user) */
EXECUTE dbms_aqadm.drop_queue_table (
   queue_table        => 'aq.Objmsgs_qtab');
```

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

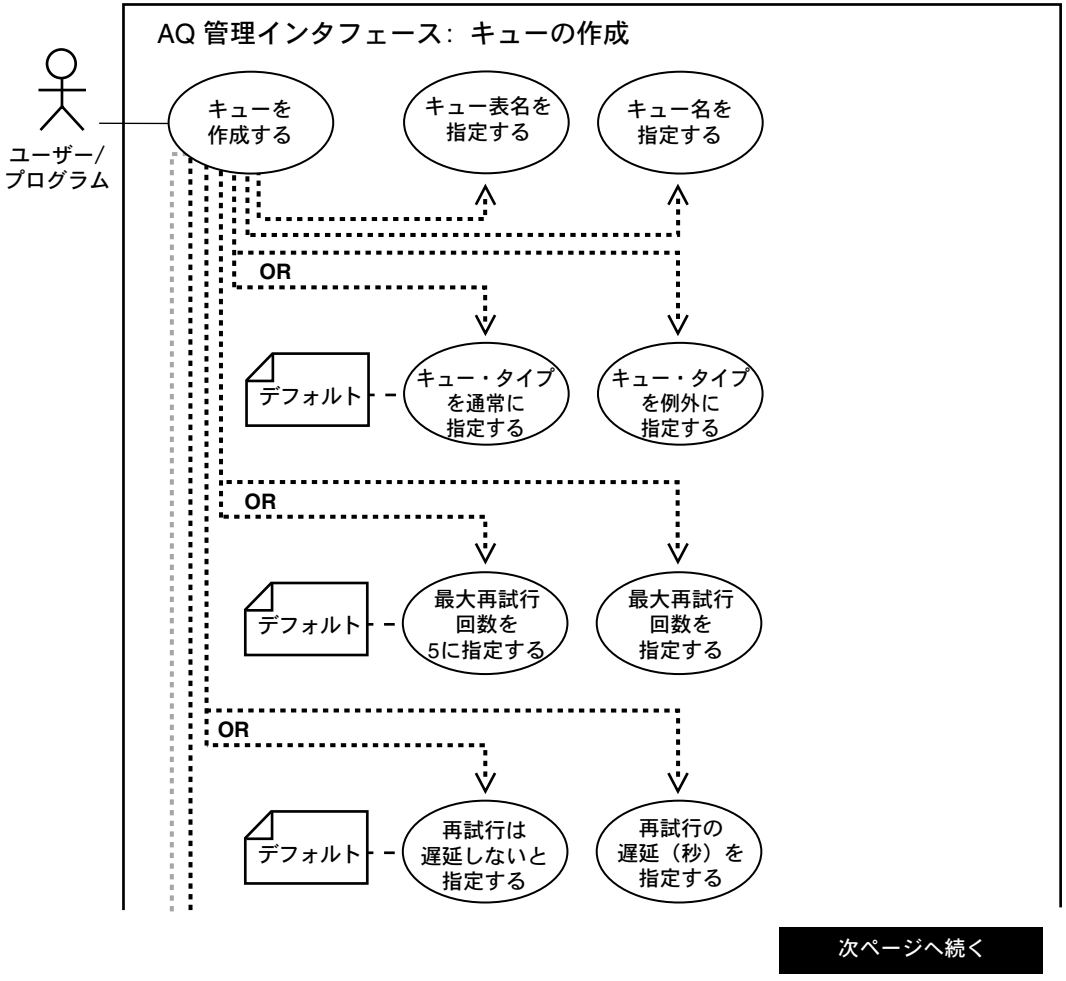
```
/* Drop the queue table and force all queues to be stopped and dropped by the
   system */
EXECUTE dbms_aqadm.drop_queue_table (
   queue_table        => 'aq.Objmsgs_qtab',
   force              => TRUE);
```

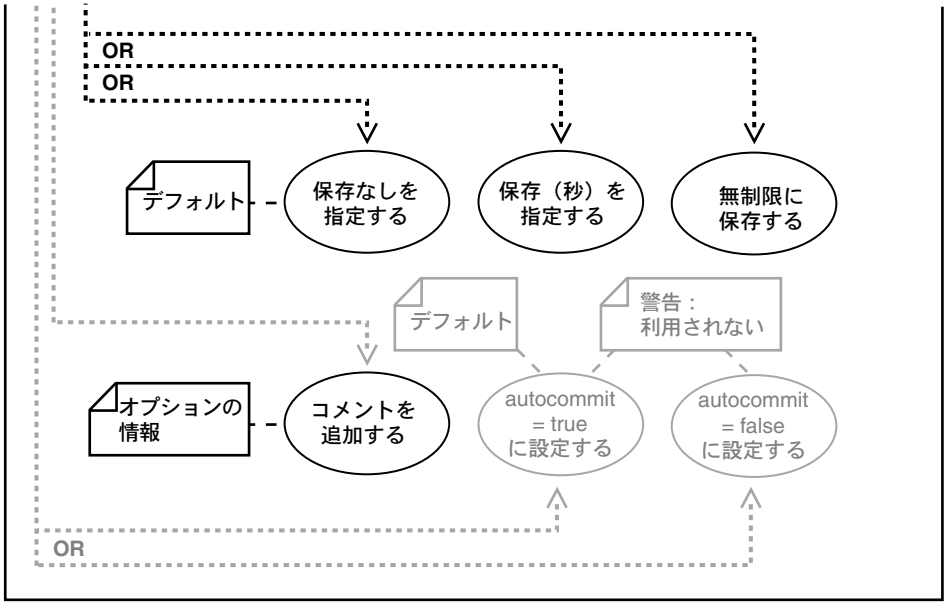
Java（JDBC）：キュー表の削除

```
/* Drop a queue table - for which all queues have already been dropped by
   the user */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueueTable            q_table;

     q_table = aq_sess.getQueueTable ("aq", "ObjMsgs_qtab");

     /* Drop the queue table*/
     q_table.drop(false);
     System.out.println("Successful drop");
}


/* Drop the queue table (and force all queues to be stopped and dropped by
   the user */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueueTable            q_table;

     q_table = aq_sess.getQueueTable ("aq", "ObjMsgs_qtab");

     /* Drop the queue table (and automatically drop all queues inside it */
     q_table.drop(true);
     System.out.println("Successful drop");
}
```

キューの作成

図 9-5 キューの作成





参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

指定したキュー表にキューを作成します。

使用上の注意

- キュー名およびキュー表名は、大文字に変換されます。大文字と小文字の組合せはサポートされていません。
- すべてのキュー名は、スキーマ内において一意である必要があります。キューは、CREATE_QUEUE で作成した後、START_QUEUE をコールすると有効になります。デフォルトでは、キューはエンキューとデキューの両方を無効にして作成されます。
- 保存されているメッセージを参照するには、メッセージ ID によってデキューするか、または SQL を使用します。
- キュー、キュー表またはサブスクライバが作成された場合、GLOBAL_TOPIC_ENABLED が TRUE であれば、対応する LDAP エントリも作成されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の CREATE_QUEUE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQSession」の createQueue

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キューの作成（9-23 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キューの作成（9-25 ページ）

PL/SQL（DBMS_AQADM）: キューの作成

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

オブジェクト型のメッセージ用のキュー表にキューを作成する

```
/* Create a message type: */
CREATE type aq.Message_typ as object (
   Subject     VARCHAR2(30),
   Text        VARCHAR2(80));

/* Create a object type queue table and queue: */
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table        => 'aq.ObjMsgs_qtab',
   Queue_payload_type => 'aq.Message_typ');

EXECUTE dbms_aqadm.create_queue (
   Queue_name         =>  'msg_queue',
   Queue_table        => 'aq.ObjMsgs_qtab');
```

RAW 型のメッセージ用のキュー表にキューを作成する

```
/* Create a RAW type queue table and queue: */
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table          => 'aq.RawMsgs_qtab',
   Queue_payload_type   => 'RAW');

/* Create queue: */
EXECUTE dbms_aqadm.create_queue (
   Queue_name           => 'raw_msg_queue',
   Queue_table          => 'aq.RawMsgs_qtab');
  Create a prioritized message queue table and queue

/* Create a queue table for priortized messages: */
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table        => 'aq.PriorityMsgs_qtab',
   Sort_list          => 'PRIORITY,ENQ_TIME',
   Queue_payload_type => 'aq.Message_typ');
/* Create queue: */
EXECUTE dbms_aqadm.create_queue (
   Queue_name         => 'priority_msg_queue',
   Queue_table        => 'aq.PriorityMsgs_qtab');
```

マルチ・コンシューマのキュー表およびキューを作成する

```
/* Create a queue table for multi-consumers: */
EXECUTE dbms_aqadm.create_queue_table (
   queue_table        => 'aq.MultiConsumerMsgs_qtab',
   Multiple_consumers => TRUE,
   Queue_payload_type => 'aq.Message_typ');

/* Create queue: */
EXECUTE dbms_aqadm.create_queue (
   Queue_name         => 'MultiConsumerMsg_queue',
   Queue_table        => 'aq.MultiConsumerMsgs_qtab');
```

伝播を実証するためのキュー表およびキューを作成する

```
/* Create queue: */
EXECUTE dbms_aqadm.create_queue (
   Queue_name        => 'AnotherMsg_queue',
   queue_table       => 'aq.MultiConsumerMsgs_qtab');
```

8.1 互換のマルチ・コンシューマのキュー表およびキューを作成する

```
/* Create a queue table for multi-consumers compatible with Release 8.1: */
EXECUTE dbms_aqadm.create_queue_table (
   Queue_table        => 'aq.MultiConsumerMsgs81_qtab',
   Multiple_consumers => TRUE,
   Compatible         => '8.1',
   Queue_payload_type => 'aq.Message_typ');

EXECUTE dbms_aqadm.create_queue (
   Queue_name         => 'MultiConsumerMsg81_queue',
   Queue_table        => 'aq.MultiConsumerMsgs81_qtab');
```

Java（JDBC）：キューの作成

オブジェクト型のメッセージ用のキュー表にキューを作成する

```
public static void example(AQSession aq_sess) throws AQException
{
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;

     q_table = aq_sess.getQueueTable ("aq", "ObjMsgs_qtab");

     /* Create a new AQQueueProperty object: */
     queue_prop = new AQQueueProperty();

     queue = aq_sess.createQueue (q_table, "msg_queue", queue_prop);
     System.out.println("Successful createQueue");

}
```

RAW 型のメッセージ用のキュー表にキューを作成する

```
public static void example(AQSession aq_sess) throws AQException
{
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;

     q_table = aq_sess.getQueueTable ("aq", "RawMsgs_qtab");

     /* Create a new AQQueueProperty object: */
     queue_prop = new AQQueueProperty();

     queue = aq_sess.createQueue (q_table, "msg_queue", queue_prop);
```

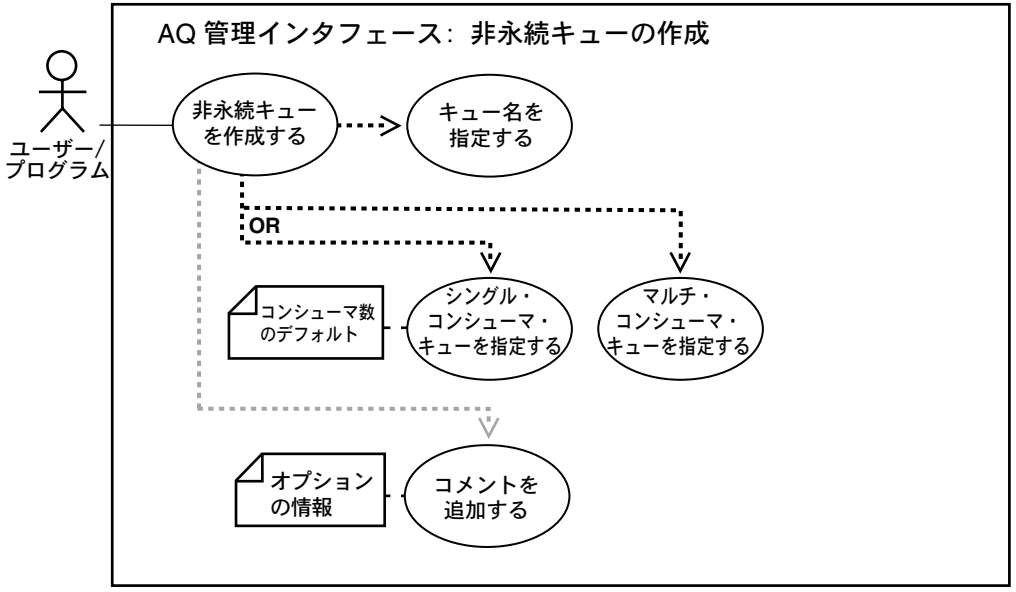
```
     System.out.println("Successful createQueue");

}
```

優先メッセージを持つマルチ・コンシューマ・キューを作成する

```
public static void example(AQSession aq_sess) throws AQException
{
     AQQueueTableProperty    qtable_prop;
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;
     AQAgent                 agent;

     qtable_prop = new AQQueueTableProperty("RAW");
     qtable_prop.setMultiConsumer(true);

     qtable_prop.setSortOrder("priority,enq_time");
     q_table = aq_sess.createQueueTable ("aq", "PriorityMsgs_qtab",
qtable_prop);

     queue_prop = new AQQueueProperty();
     queue = aq_sess.createQueue (q_table, "priority_msg_queue", queue_prop);

}
```

非永続キューの作成

図 9-6 非永続キューの作成



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

非永続キューを作成します。

使用上の注意

このキューは、シングル・コンシューマ・キューまたはマルチ・コンシューマ・キューのいずれかです。すべてのキュー名は、スキーマ内において一意である必要があります。このキューは、キュー名によって指定された同じスキーマ内にシステムが作成した 8.1 互換のキュー表（AQ$_MEM_SC または AQ$_MEM_MC）に作成されます。キュー名にスキーマ名が指定されていないときは、ログイン・ユーザーのスキーマに作成されます。キューは、CREATE_NP_QUEUE で作成した後、START_QUEUE をコールすると有効になります。デフォルトでは、キューはエンキューとデキューの両方を無効にして作成されます。

非永続キュー内部に、RAW 型およびオブジェクト型（ユーザー定義型）メッセージをエンキューできます。ユーザーは、非永続キューからはデキューできません。非永続キューから

メッセージを取り出す唯一の方法は、OCI 通知メカニズムを使用する方法です（11-55 ページの「通知の登録」を参照）。

非永続キューには、listen コールを起動できません（11-24 ページの「1 個以上のシングル・コンシューマ・キューのリスニング」を参照）。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の CREATE_NP_QUEUE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）: 参照マニュアルはありません。

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 非永続キューの作成（9-28 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 非永続キューの作成（9-28 ページ）

PL/SQL（DBMS_AQADM）: 非永続キューの作成

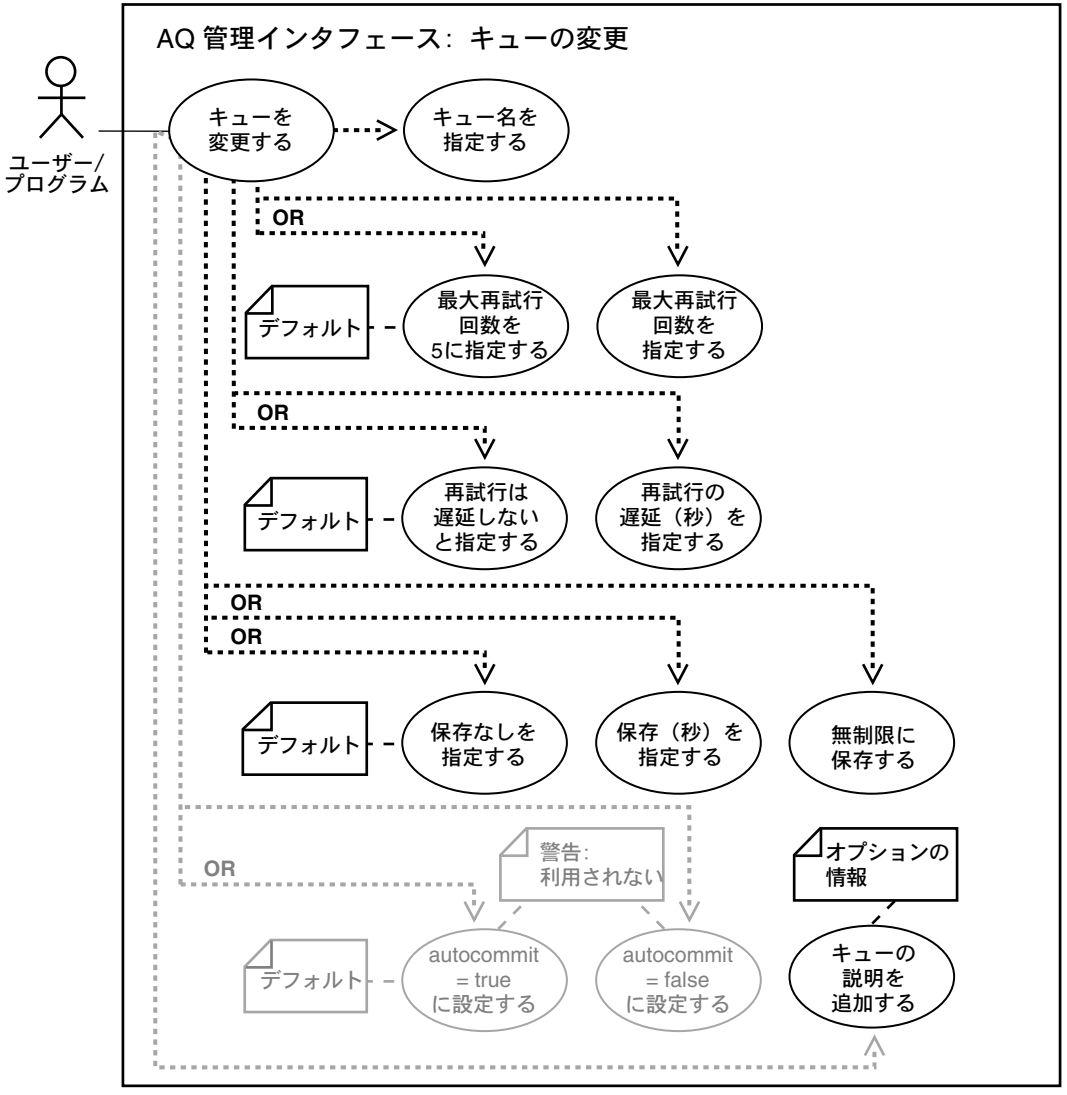
```
/* Create a nonpersistent single-consumer queue (Note: this is not preceded by
   creation of a queue table) */
EXECUTE dbms_aqadm.create_np_queue(
   Queue_name          => 'Singleconsumersmsg_npque',
   Multiple_consumers  => FALSE);

/* Create a nonpersistent multi-consumer queue (Note: this is not preceded by
   creation of a queue table) */
EXECUTE dbms_aqadm.create_np_queue(
   Queue_name          => 'Multiconsumersmsg_npque',
   Multiple_consumers  => TRUE);
```

Java（JDBC）: 非永続キューの作成

この機能は、Java API を介しては使用できません。

キューの変更

図 9-7 キューの変更



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

キューの既存のプロパティを変更します。変更できるのは、max_retries、comment、retry_delay および retention_time のみです。

使用上の注意

保存されているメッセージを参照するには、メッセージ ID によってデキューするか、または SQL を使用します。

キュー、キュー表またはサブスクライバが、作成、変更または削除された場合、GLOBAL_TOPIC_ENABLED が TRUE であれば、対応する LDAP エントリも作成されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の ALTER_QUEUE プロシージャ
- Visual Basic（OO4O）：参照マニュアルはありません。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の alterQueue

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）：キューの変更（9-30 ページ）
- Visual Basic（OO4O）：例はありません。
- Java（JDBC）：キューの変更（9-31 ページ）

PL/SQL（DBMS_AQADM）：キューの変更

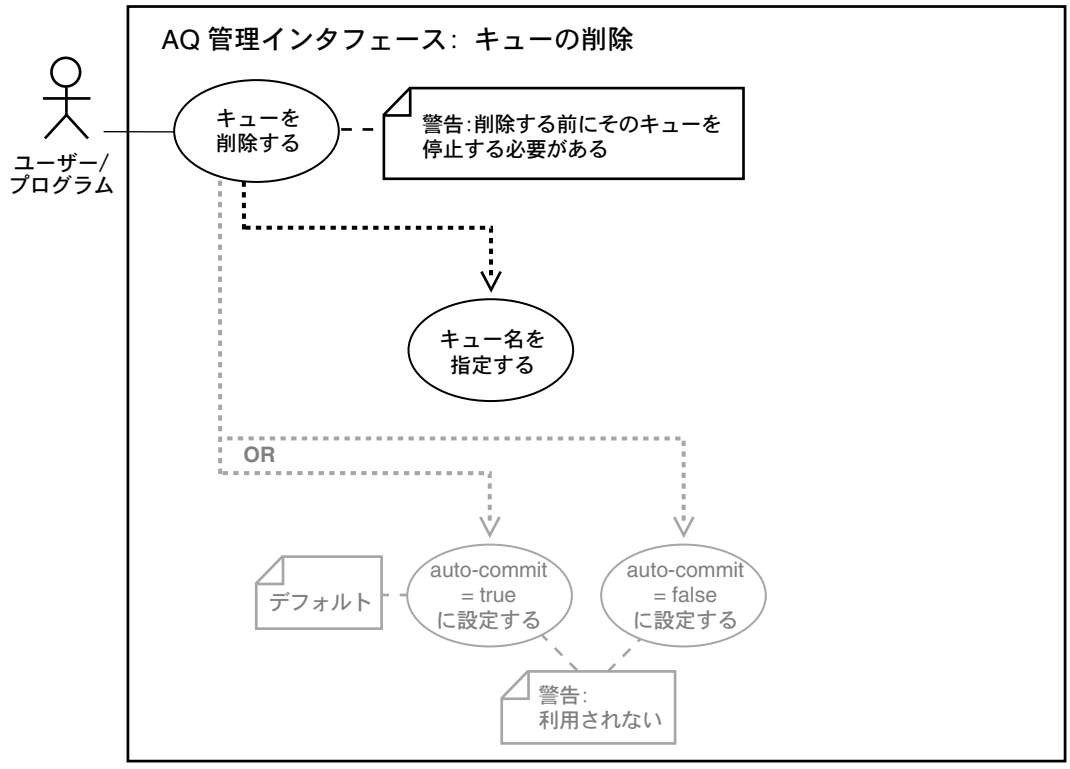
```
/* Alter queue to change retention time, saving messages for 1 day after
   dequeueing:  */
EXECUTE dbms_aqadm.alter_queue (
   queue_name        => 'aq.Anothermsg_queue',
   retention_time    => 86400);
```

Java（JDBC）：キューの変更

```
/* Alter a queue to change retention time, saving messages for 1 day
   after dequeuing */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueueProperty         queue_prop;
     AQQueue                 queue;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "Anothermsg_queue");

     /* Create a new AQQueueProperty object: */
     queue_prop = new AQQueueProperty();

     /* Change retention time to 1 day */
     queue_prop.setRetentionTime(new Double(86400));

     /* Alter the queue */
     queue.alterQueue(queue_prop);

}
```

キューの削除

図 9-8 キューの削除



参照: 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

既存のキューを削除します。あらかじめ STOP_QUEUE がコールされ、キューがエンキューおよびデキューの両方に対して無効にされていないかぎり、DROP_QUEUE は許可されません。すべてのキュー・データは、削除操作の一部として削除されます。

使用上の注意

キュー、キュー表またはサブスクライバが、作成、変更または削除された場合、GLOBAL_TOPIC_ENABLED が TRUE であれば、対応する LDAP エントリも作成されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の DROP_QUEUE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の dropQueue

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キューの削除（9-33 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キューの削除（9-34 ページ）

PL/SQL（DBMS_AQADM）: キューの削除

標準キューを削除する

```
/* Stop the queue preparatory to dropping it (a queue may be dropped only after
   it has been succesfully stopped for enqueing and dequeing): */
EXECUTE dbms_aqadm.stop_queue (
   Queue_name         => 'aq.Msg_queue');

/* Drop queue: */
EXECUTE dbms_aqadm.drop_queue (
   Queue_name          => 'aq.Msg_queue');
```

非永続キューを削除する

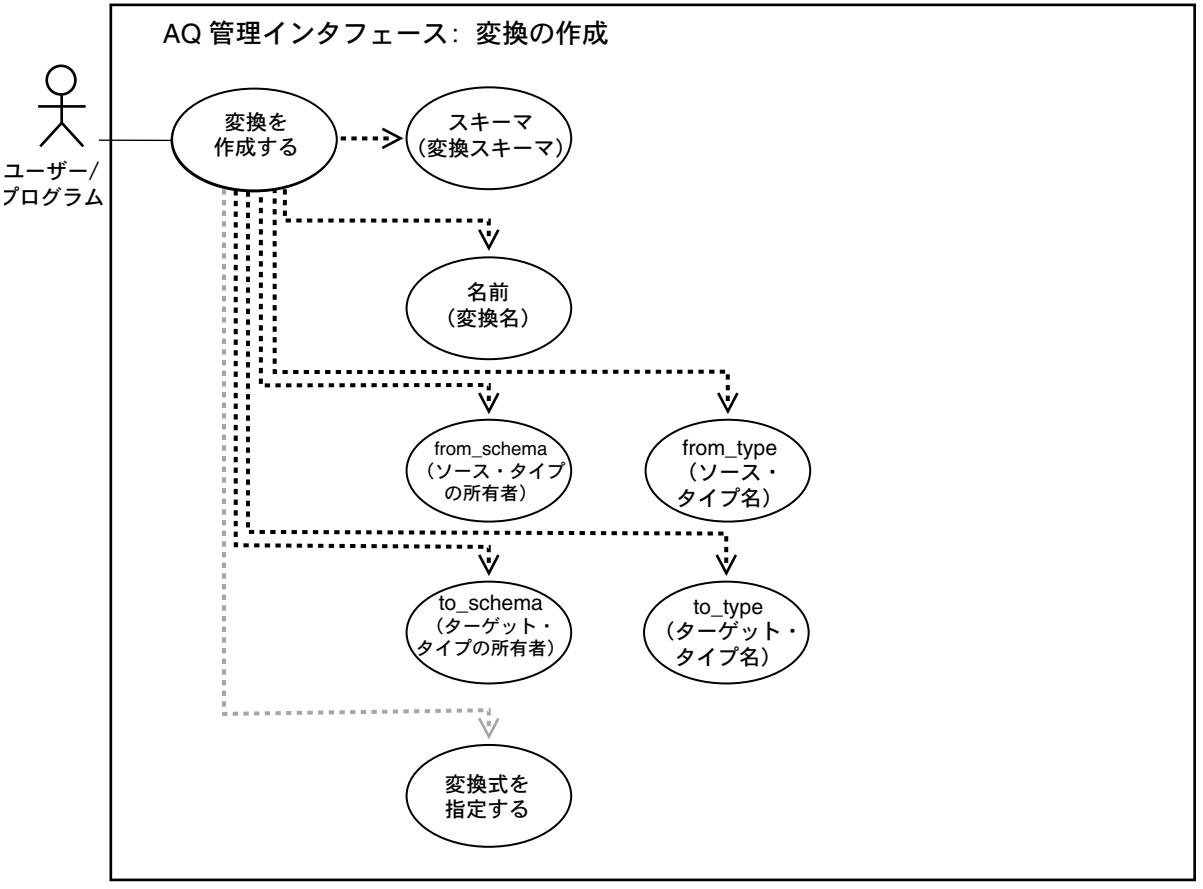
```
EXECUTE DBMS_AQADM.DROP_QUEUE( queue_name => 'Nonpersistent_singleconsumerq1');
EXECUTE DBMS_AQADM.DROP_QUEUE( queue_name => 'Nonpersistent_multiconsumerq1');
```

Java（JDBC）：キューの削除

```
/* Drop a queue */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue                queue;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "Msg_queue");

     /* Stop the queue first */
     queue.stop(true);

     /* Drop the queue */
     queue.drop();
}
```

変換の作成

図 9-9 変換の作成



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

メッセージ・フォーマットの変換を作成します。この変換は、from_type 型の入力を持つ SQL ファンクションである必要があり、to_type 型のオブジェクトを戻します。to_type 型で、from_type を参照する SQL 式も使用できます。from_type に対するすべての参照は、source.user_data というフォーマットにします。

使用上の注意

この機能を使用するには、dbms_transform に対する実行権限が必要です。また、変換のソース・タイプおよび宛先タイプであるユーザー定義型に対する実行権限と、変換ファンクションで使用するすべての PL/SQL ファンクションに対する実行権限も必要です。変換では、データベースの状態の書込み（DML の実行）、カレント・トランザクションのコミットまたはロールバックはできません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_TRANSFORM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 76 章「DBMS_TRANSFORM」の CREATE_TRANSFORMATION プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_TRANSFORM）: 変換の作成（9-37 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

PL/SQL（DBMS_TRANSFORM）：変換の作成

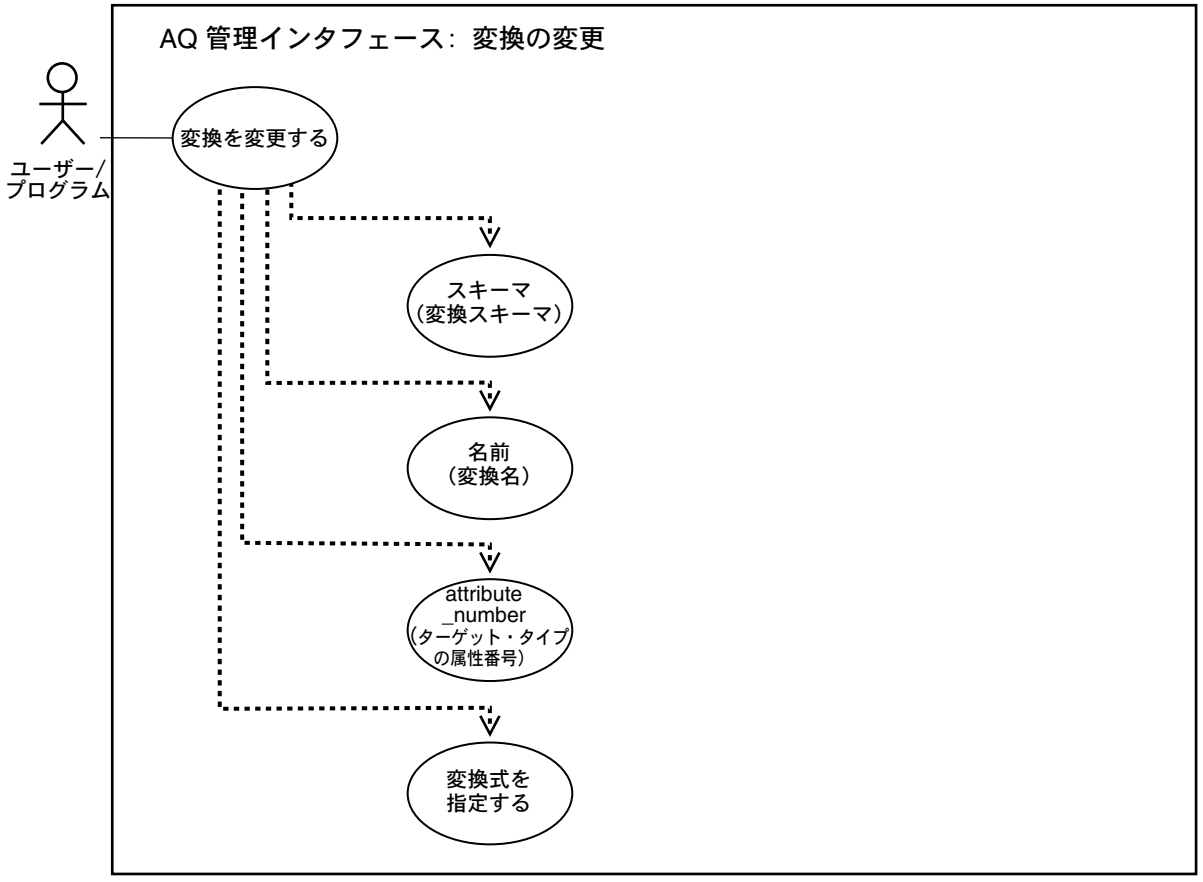
```
dbms_transform.create_transformation(schema => 'scott',
    name           => 'test_transf', from_schema => 'scott',
    from_type      => 'type1', to_schema => 'scott',
    to_type        => 'type2',
    transformation => 'scott.trans_func(source.user_data)');
```

次のようにも実行できます。

```
dbms_transform.create_transformation(schema => 'scott',
    name           => 'test_transf',
    from_schema    => 'scott',
    from_type      => 'type1,
    to_schema      => 'scott',
    to_type        => 'type2',
    transformation => 'scott.type2(source.user_data.attr2,
        source.user_data.attr1)');
```

変換の変更

図 9-10 変換の変更



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

変換ファンクションを変更し、ターゲット・タイプの属性ごとに変換を指定します。属性の番号が 0（ゼロ）に指定された場合、変換式は単純にソース・タイプからターゲット・タイプへの変換を定義します。`from_type` に対するすべての参照は、`source.user_data` という形式にします。ソース・タイプの属性に対するすべての参照には、`source.user_data` という接頭辞が付きます。

使用上の注意

この機能を使用するには、dbms_transform に対する実行権限が必要です。また、変換のソース・タイプおよび宛先タイプであるユーザー定義型に対する実行権限と、変換ファンクションで使用するすべての PL/SQL ファンクションに対する実行権限も必要です。

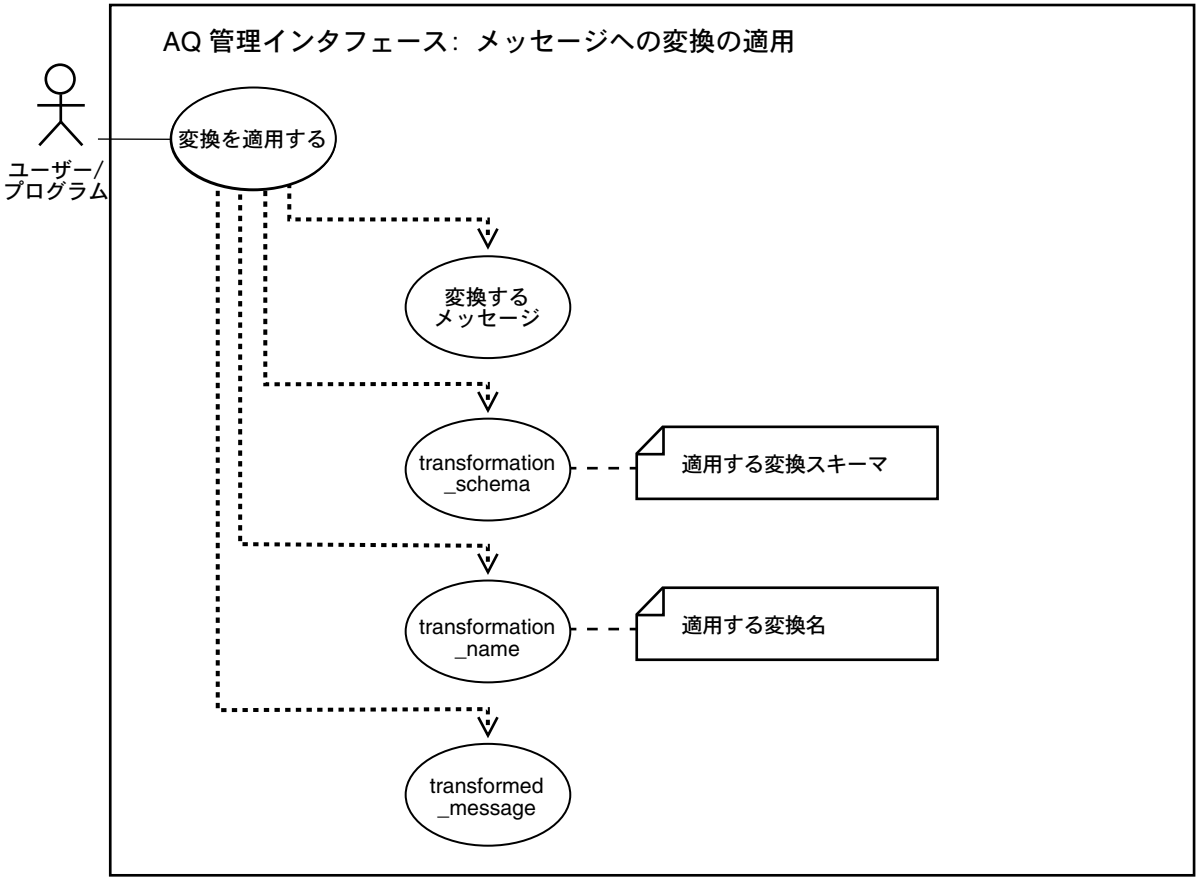
構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_TRANSFORM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 76 章「DBMS_TRANSFORM」の MODIFY_TRANSFROMATION プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

変換の適用

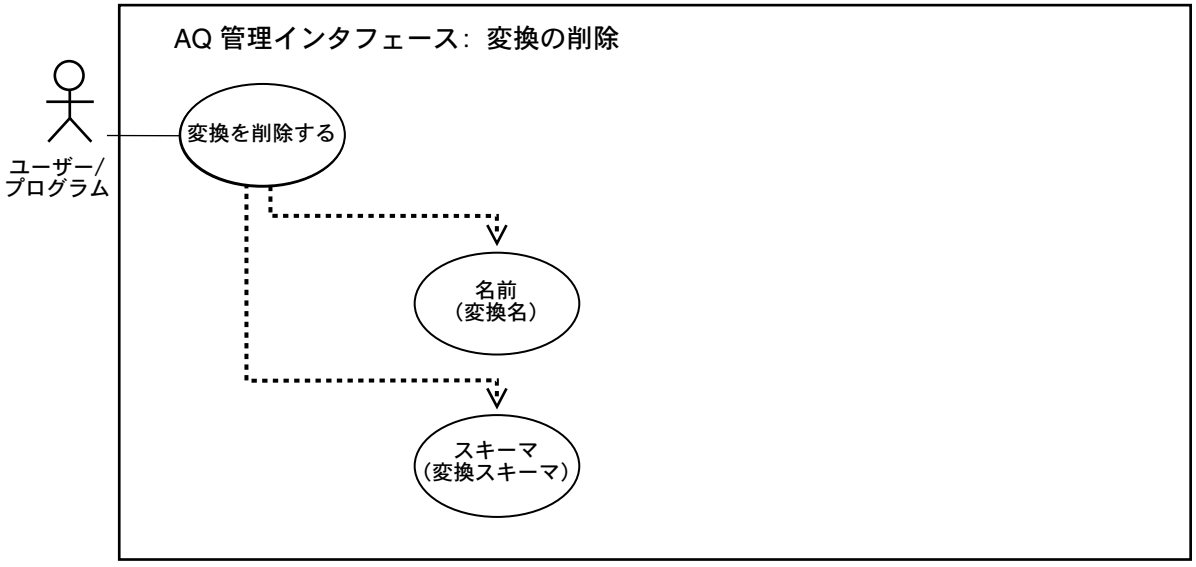
図 9-11 変換の適用



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

変換の削除

図 9-12 変換の削除



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

変換を削除します。

使用上の注意

この機能を使用するには、dbms_transform に対する実行権限が必要です。また、変換のソース・タイプおよび宛先タイプであるユーザー定義型に対する実行権限と、変換ファンクションで使用するすべての PL/SQL ファンクションに対する実行権限も必要です。

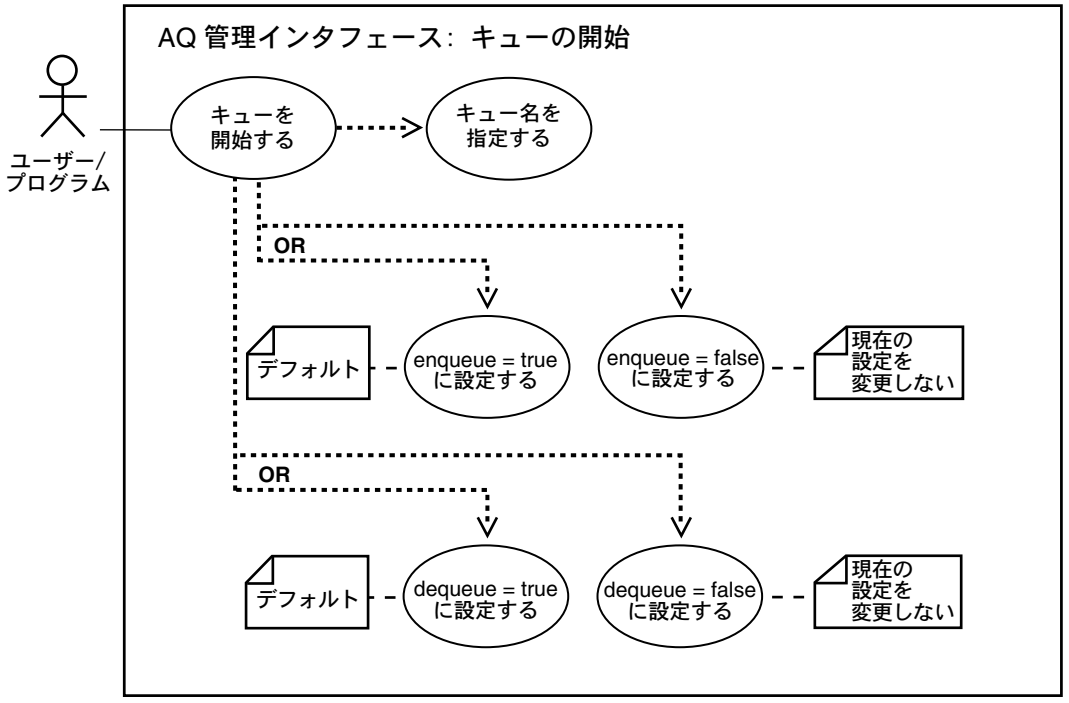
構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_TRANSFORM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 76 章「DBMS_TRANSFORM」の `DROP_TRANSFORMATION` プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』

キューの開始

図 9-13 キューの開始



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

指定したキューに対するエンキューまたはデキューを有効にします。

使用上の注意

管理者は、キューを作成した後、START_QUEUE を使用してそのキューを有効にする必要があります。デフォルトでは、ENQUEUE および DEQUEUE の両方を有効にします。例外キューに対しては、デキュー操作のみが可能です。この操作は、コールが完了し、コールにトランザクションの特性がない場合にのみ有効になります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の START_QUEUE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQQueueAdmin」の start

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キューの開始（9-44 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キューの開始（9-45 ページ）

PL/SQL（DBMS_AQADM）: キューの開始

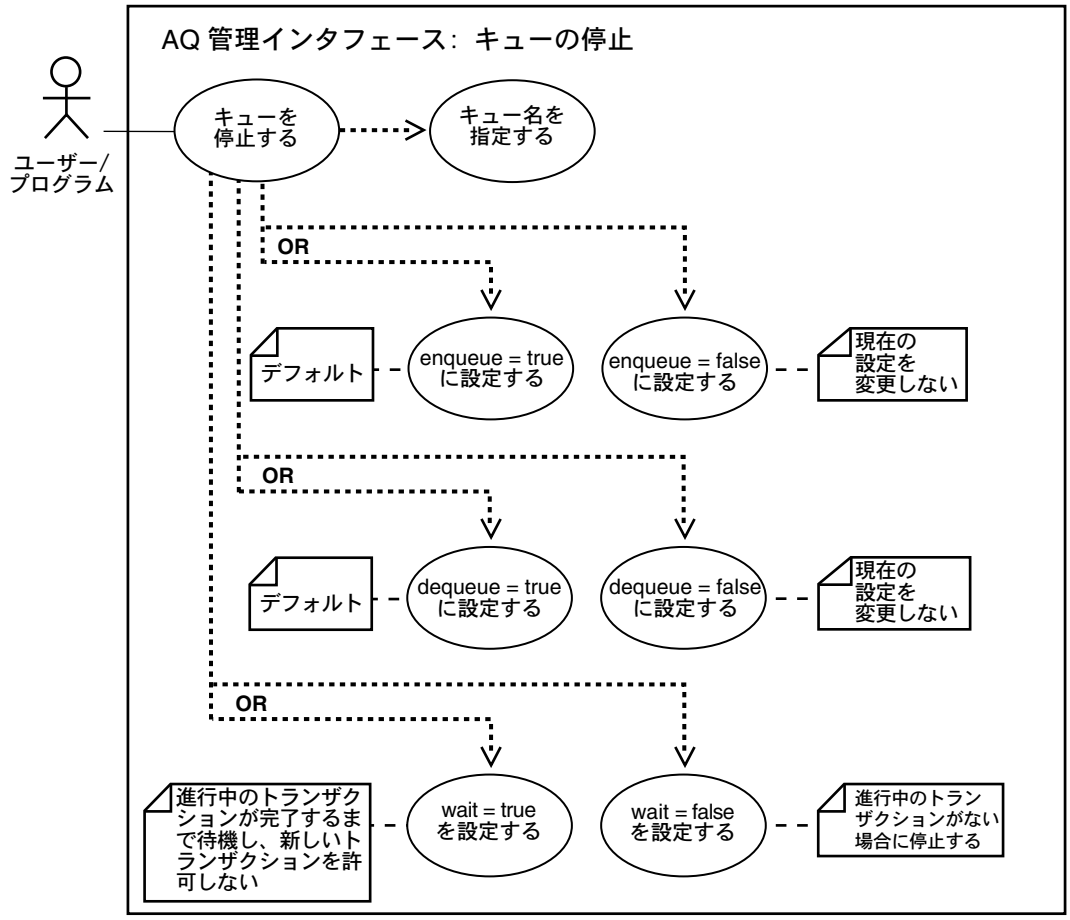
```
/* Start a queue and enable both enqueue and dequeue: */
EXECUTE dbms_aqadm.start_queue (
   queue_name         => 'Msg_queue');

/* Start a previously stopped queue for dequeue only */
EXECUTE dbms_aqadm.start_queue (
   queue_name         => 'aq.msg_queue',
   dequeue            => TRUE,
   enqueue            => FALSE);
```

Java（JDBC）：キューの開始

```
/* Start a queue - enable both enqueue and dequeue */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue                 queue;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "Msg_queue");

     /* Enable enqueue and dequeue */
     queue.start();
}


/* Start a previously stopped queue for dequeue only */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue                 queue;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "Msg_queue");

     /* Enable enqueue and dequeue */
     queue.start(false, true);
}
```

キューの停止

図 9-14 キューの停止



参照: 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

指定したキューに対するエンキューまたはデキューを無効にします。

使用上の注意

デフォルトでは、このコールによって ENQUEUE および DEQUEUE の両方が無効になります。キューは、未完了のトランザクションが存在する場合には停止できません。この操作は、コールが完了し、コールにトランザクションの特性がない場合にのみ有効になります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の STOP_QUEUE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQQueueAdmin」の stop

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キューの停止（9-47 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キューの停止（9-48 ページ）

PL/SQL（DBMS_AQADM）: キューの停止

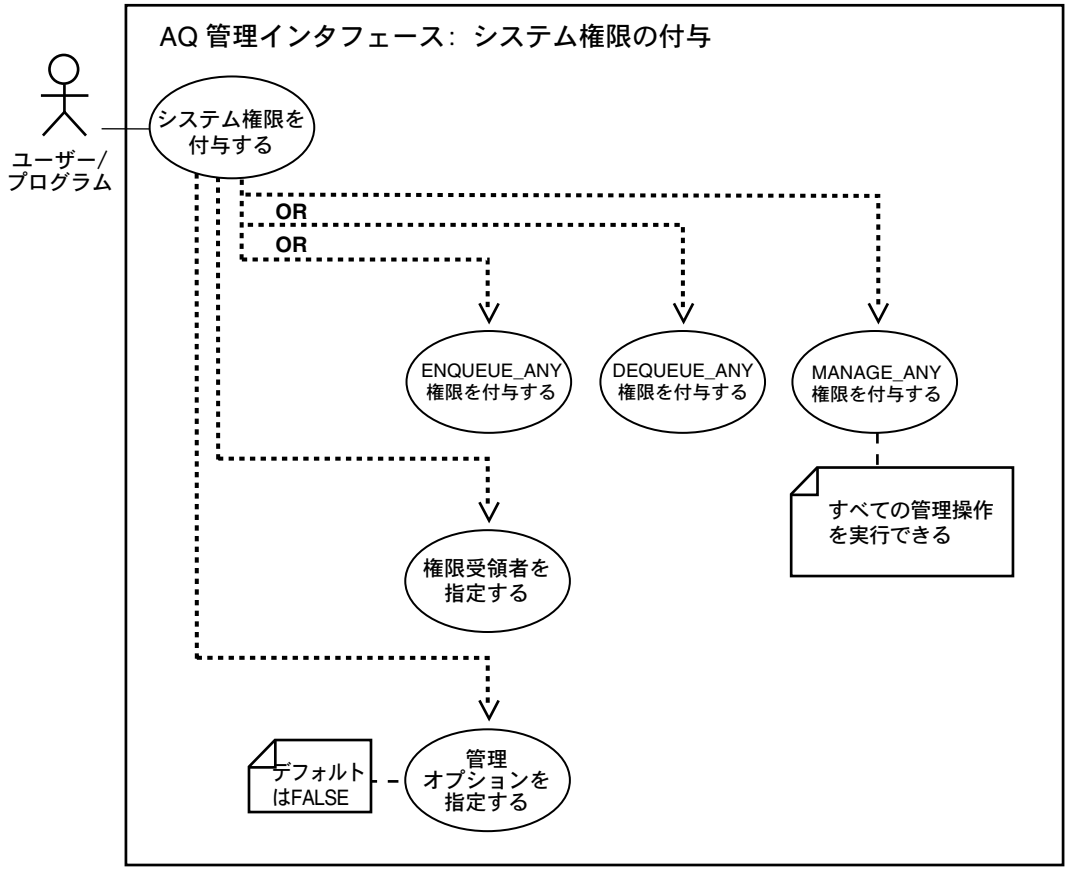
```
/* Stop the queue: */
EXECUTE dbms_aqadm.stop_queue (
   queue_name         => 'aq.Msg_queue');
```

Java（JDBC）：キューの停止

```
/* Stop a queue - wait for oustanding transactions */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue                queue;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "Msg_queue");

     /* Enable enqueue and dequeue */
     queue.stop(true);
}
```

システム権限の付与

図 9-15 システム権限の付与



参照: 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

ユーザーおよびロールに AQ システム権限を付与します。この権限とは、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY です。最初は、SYS および SYSTEM のみがこのプロシージャを正常に使用できます。

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の GRANT_SYSTEM_PRIVILEGE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）: 参照マニュアルはありません。

使用上の注意

ありません。

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: システム権限の付与（9-50 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: システム権限の付与（9-51 ページ）

PL/SQL（DBMS_AQADM）: システム権限の付与

```
/* User AQADM grants the rights to enqueue and dequeue to ANY queues: */
```

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
CONNECT system/manager;
CREATE USER aqadm IDENTIFIED BY aqadm;
GRANT CONNECT, RESOURCE TO aqadm;
GRANT EXECUTE ON DBMS_AQADM TO aqadm;
GRANT Aq_administrator_role TO aqadm;
```

```
CONNECT aqadm/aqadm;
EXECUTE DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE(
   privilege          =>    'ENQUEUE_ANY',
   grantee            =>    'Jones',
```

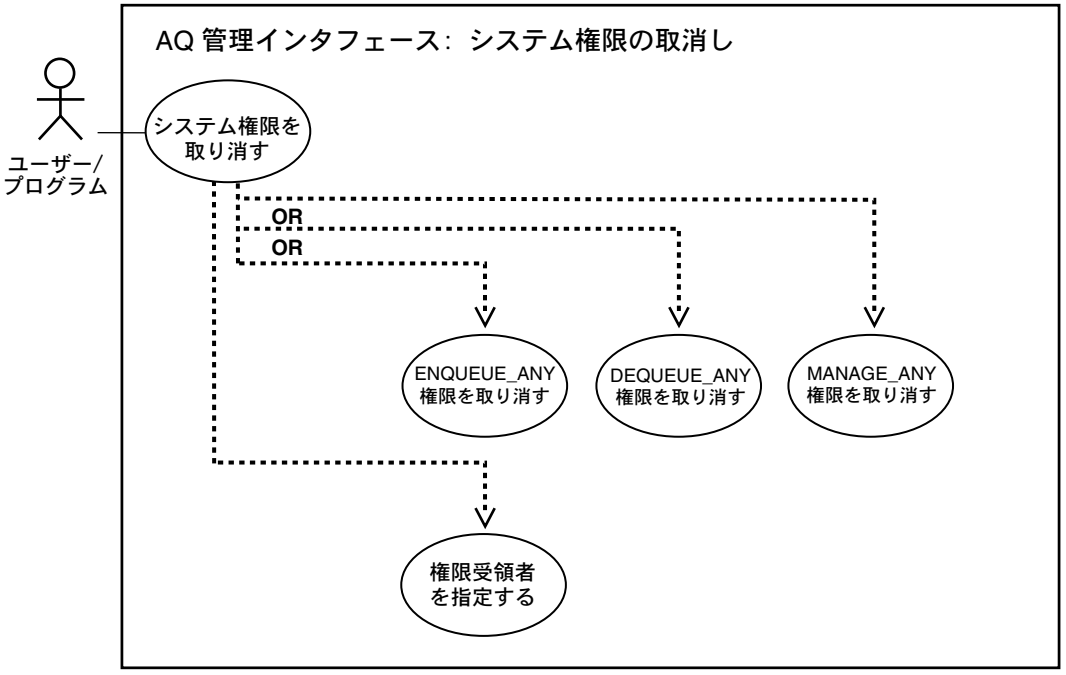
```
   admin_option       =>     FALSE);
EXECUTE DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE(
   privilege          =>     'DEQUEUE_ANY',
   grantee            =>     'Jones',
   admin_option       =>      FALSE);
```

Java（JDBC）：システム権限の付与

この機能は、Java API を介しては使用できません。

システム権限の取消し

図 9-16 システム権限の取消し



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

ユーザーおよびロールの AQ システム権限を取り消します。この権限とは、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY です。システム権限の ADMIN オプションを選択的に取り消すことはできません。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の REVOKE_SYSTEM_PRIVILEGE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）: 参照マニュアルはありません。

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

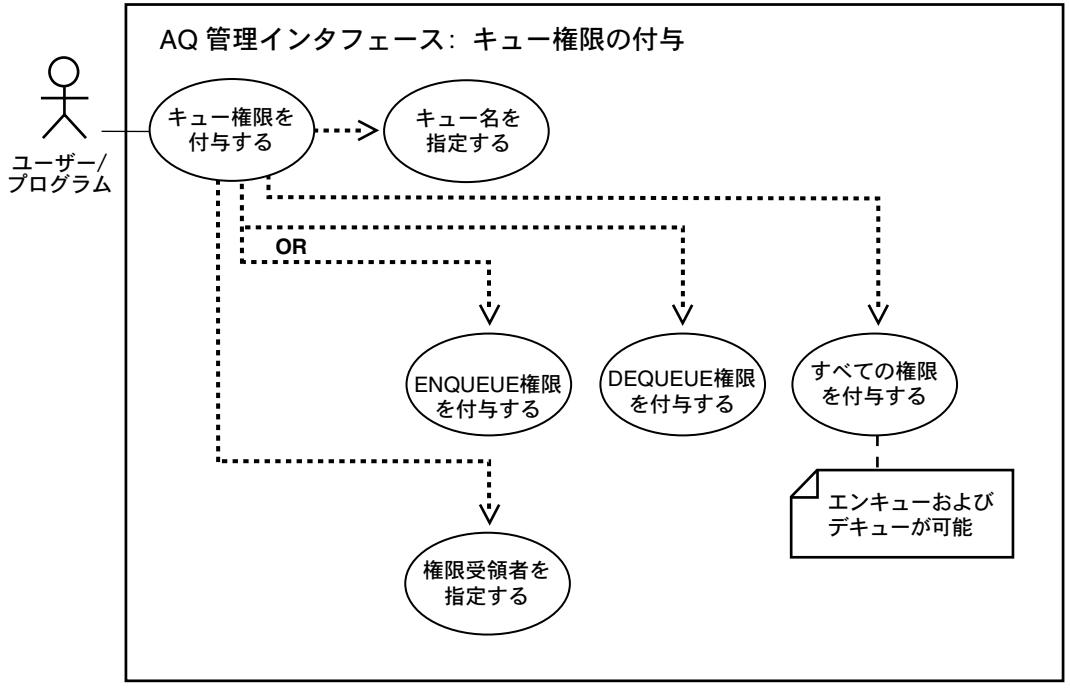
- PL/SQL（DBMS_AQADM）の使用 : システム権限の取消し（9-53 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

PL/SQL（DBMS_AQADM）の使用 : システム権限の取消し

```
/* To revoke the DEQUEUE_ANY system privilege from Jones. */
CONNECT system/manager;
   execute DBMS_AQADM.REVOKE_SYSTEM_PRIVILEGE(privilege=>'DEQUEUE_ANY',
                                grantee=>'Jones');
```

キュー権限の付与

図 9-17 キュー権限の付与



参照: 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

キューに関する権限をユーザーおよびロールに付与します。この権限とは、ENQUEUE または DEQUEUE です。最初は、キュー表の所有者のみがキューを付与するプロシージャを使用できます。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の GRANT_QUEUE_PRIVILEGE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQQueueAdmin」の grantQueuePrivilege

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キュー権限の付与（9-55 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キュー権限の付与（9-55 ページ）

PL/SQL（DBMS_AQADM）: キュー権限の付与

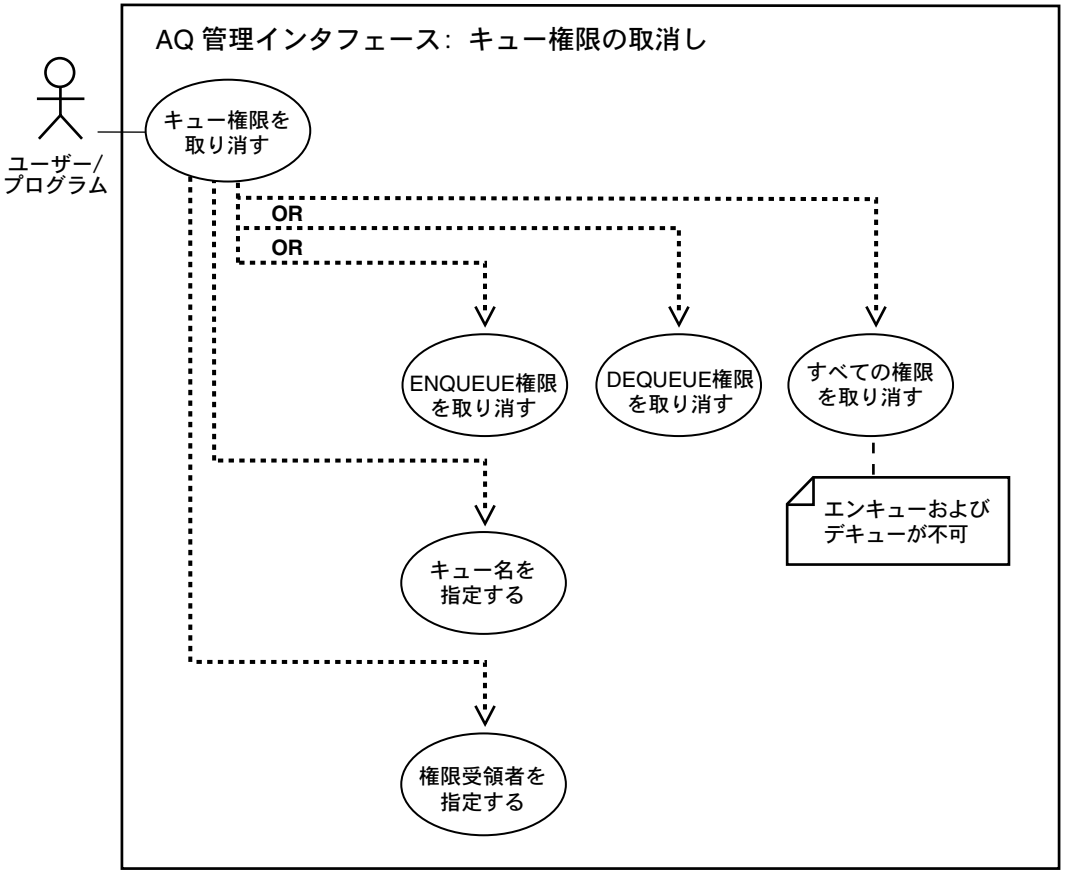
```
/* User grants the access right for both enqueue and dequeue rights using
   DBMS_AQADM.GRANT. */
EXECUTE DBMS_AQADM.GRANT_QUEUE_PRIVILEGE (
   privilege     =>     'ALL',
   queue_name    =>     'aq.multiconsumermsg81_queue',
   grantee       =>     'Jones',
   grant_option  =>      TRUE);
```

Java（JDBC）: キュー権限の付与

```
/* Grant enqueue and dequeue privileges on queue to user 'Jones' */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue                  queue;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "multiconsumermsg81_queue");
     /* Enable enqueue and dequeue */
     queue.grantQueuePrivilege("ALL", "Jones", true);
}
```

キュー権限の取消し

図 9-18 キュー権限の取消し



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

ユーザーおよびロールのキュー権限を取り消します。この権限とは、ENQUEUE または DEQUEUE です。

使用上の注意

権限を取り消すユーザーは、取消し対象となる権限の付与者である必要があります。GRANT オプションによって伝播された権限は、付与者の権限が取り消されたときに取り消されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の REVOKE_QUEUE_PRIVILEGE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の revokeQueuePrivilege

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キュー権限の取消し（9-57 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キュー権限の取消し（9-58 ページ）

PL/SQL（DBMS_AQADM）: キュー権限の取消し

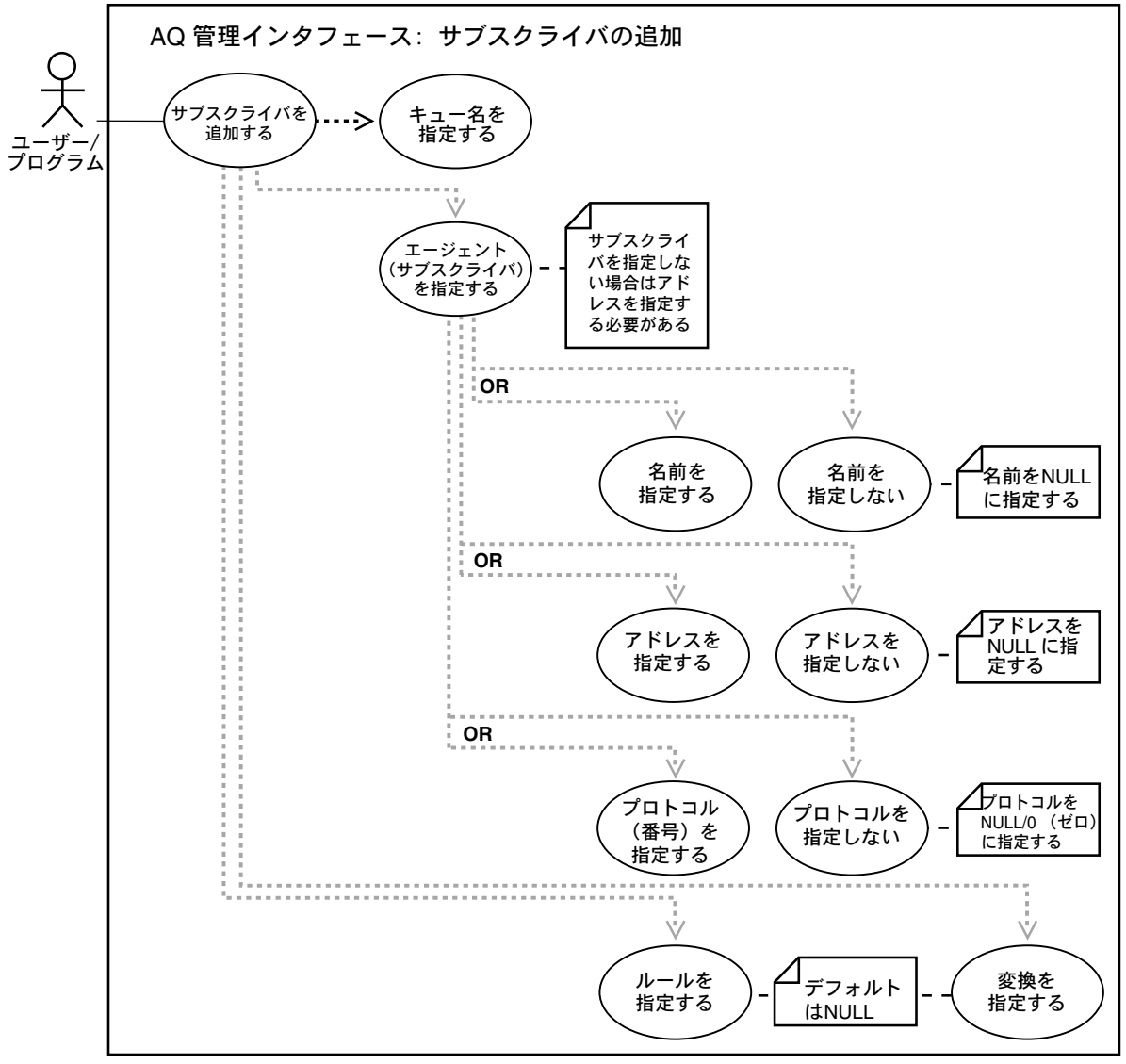
```
/* User can revoke the dequeue right of a grantee on a specific queue
   leaving the grantee with only the enqueue right: */
CONNECT scott/tiger;
EXECUTE DBMS_AQADM.REVOKE_QUEUE_PRIVILEGE(
   privilege     =>     'DEQUEUE',
   queue_name    =>     'scott.ScottMsgs_queue',
   grantee       =>     'Jones');
```

Java（JDBC）：キュー権限の取消し

```
/* User can revoke the dequeue right of a grantee on a specific
   queue, leaving only the enqueue right */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue                queue;

     /* Get the queue object */
     queue = aq_sess.getQueue("SCOTT", "ScottMsgs_queue");

     /* Enable enqueue and dequeue */
     queue.revokeQueuePrivilege("DEQUEUE", "Jones");
}
```

サブスクライバの追加

図 9-19 サブスクライバの追加



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

デフォルトのサブスクライバをキューに追加します。

使用上の注意

- プログラムから特定の受信者リストまたはデフォルトのサブスクライバ・リストに、メッセージをエンキューできます。この操作は、マルチ・コンシューマに対応したキューに対してのみ正常に実行できます。この操作はすぐに有効になり、この操作を含むトランザクションはコミットされます。このコールが完了した後に実行されるエンキュー・リクエストには、新しい動作が反映されます。
- 次に示すように、ルール内のすべての文字列は引用符で囲む必要があります。

```
rule   => 'PRIORITY <= 3 AND CORRID =  ''FROM JAPAN'''
```

  すべて一重引用符を使用することに注意してください。

- キュー、キュー表またはサブスクライバが作成された場合、GLOBAL_TOPIC_ENABLED が TRUE であれば、対応する LDAP エントリも作成されます。
- デキューまたは伝播中に適用する変換の名前を指定します。変換は、DBMS_TRANSFORM パッケージを使用して作成します。詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。
- XMLType 属性を持つペイロードを含むキューに対しては、XMLType.existsNode() メソッドおよび XMLType.extract() メソッドを含むルールを指定できます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の ADD_SUBSCRIBER プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の addSubscriber

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）：サブスクライバの追加（9-61 ページ）
- Visual Basic（OO4O）：例はありません。
- Java（JDBC）：サブスクライバの追加（9-61 ページ）

PL/SQL（DBMS_AQADM）：サブスクライバの追加

```
/* Anonymous PL/SQL block for adding a subscriber at a designated queue in a
designated schema at a database link: */
DECLARE
   subscriber          sys.aq$_agent;
BEGIN
   subscriber := sys.aq$_agent ('subscriber1', 'aq2.msg_queue2@london', null);
   DBMS_AQADM.ADD_SUBSCRIBER(
     queue_name         => 'aq.multi_queue',
      subscriber            =>  subscriber);
 END;

/* Add a subscriber with a rule: */
DECLARE
   subscriber       sys.aq$_agent;
BEGIN
   subscriber := sys.aq$_agent('subscriber2', 'aq2.msg_queue2@london', null);
   DBMS_AQADM.ADD_SUBSCRIBER(
     queue_name         =>  'aq.multi_queue',
      subscriber            =>    subscriber,
      rule                  =>    'priority < 2');
END;
```

サブスクライバを追加して変換を指定する

```
/* Add a subscriber with a rule and specify a transformation */
DECLARE
   subscriber       sys.aq$_agent;
BEGIN
   subscriber := sys.aq$_agent('subscriber2', 'aq2.msg_queue2@london', null);
   DBMS_AQADM.ADD_SUBSCRIBER(
     queue_name         =>  'aq.multi_queue',
      subscriber            =>    subscriber,
      transformation        => 'AQ.msg_map');
/* Where the transformation was created as */
EXECUTE DBMS_TRANSFORM.CREATE_TRANSFORMATION
```

```
 ( schema => 'AQ',
   name => 'msg_map',
   from_schema => 'AQ',
   from_type => 'purchase_order1',
   to_schema => 'AQ',
   to_type => 'purchase_order2',
   transformation => 'AQ.transform_PO(source.user_data)');
END;
```

PL/SQL（DBMS_AQADM）：ルールベースのサブスクライバの追加

```
DECLARE
   subscriber          sys.aq$_agent;
BEGIN
   subscriber :=  sys.aq$_agent('East_Shipping','ES.ES_bookedorders_que',null);
   DBMS_AQADM.ADD_SUBSCRIBER(
      queue_name           => 'OE.OE_bookedorders_que',
      subscriber           => subscriber,
      rule                 => 'tab.user_data.orderregion =  ''EASTERN'' OR
                              (tab.user_data.ordertype =  ''RUSH'' AND
                               tab.user_data.customer.country = ''USA'') ');
END;

/* Add a rule-based subscriber for Overseas Shipping */
DECLARE
  subscriber     aq$_agent;
BEGIN
  subscriber := aq$_agent('Overseas_DHL', null, null);

  dbms_aqadm.add_subscriber(
        queue_name     => 'OS.OS_bookedorders_que',
        subscriber     => subscriber,
        rule           => 'tab.user_data.xdata.extract(''/ORDER_
TYP/ORDERTYPE/text()'').getStringVal()=''RUSH''');

END;
```

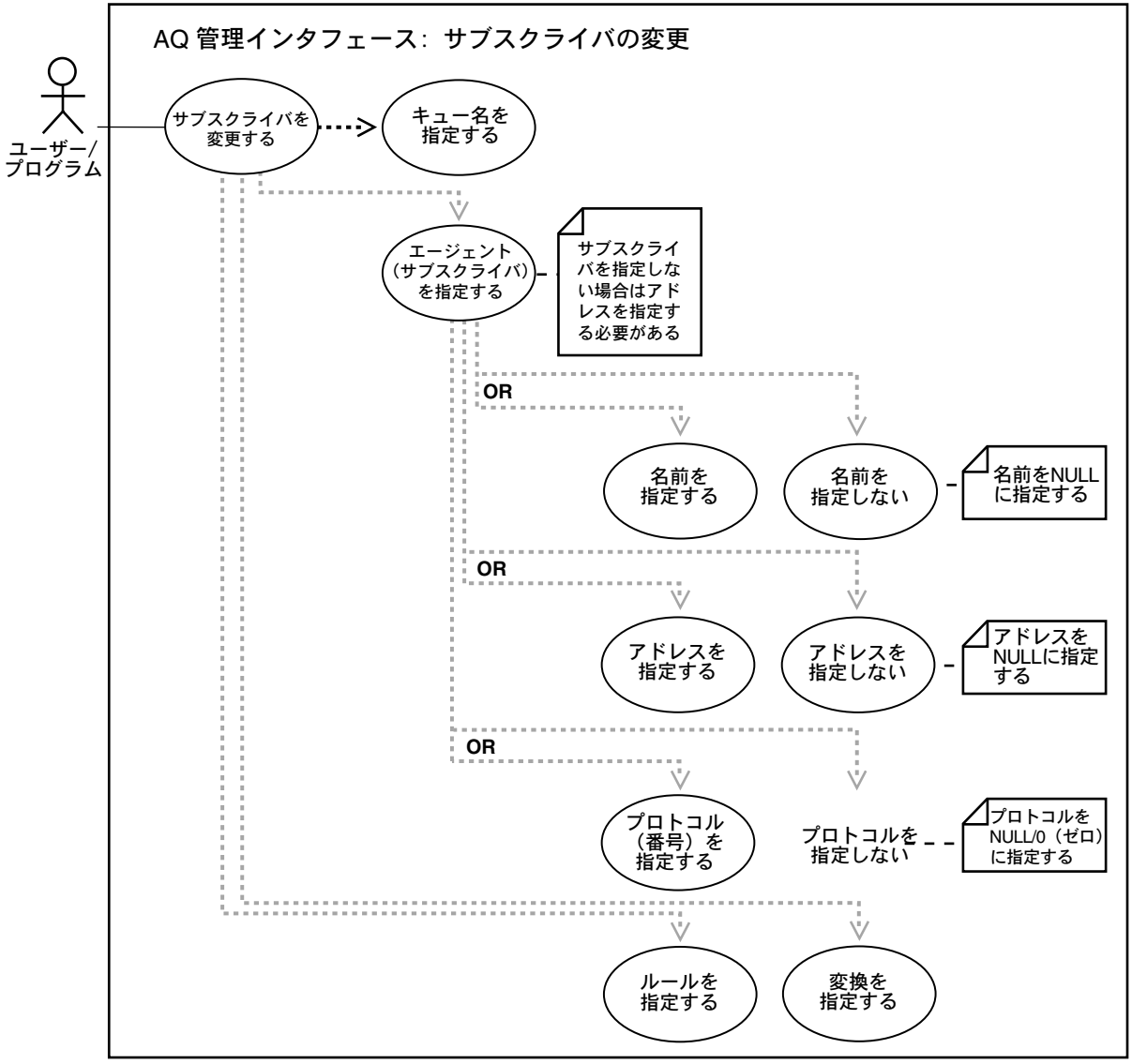
Java（JDBC）：サブスクライバの追加

```
/* Setup */
public static void setup(AQSession aq_sess) throws AQException
{
     AQQueueTableProperty    qtable_prop;
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;

     /* Create a AQQueueTable property object */
     qtable_prop = new AQQueueTableProperty("AQ.MESSAGE_TYP");
     qtable_prop.setMultiConsumer(true);

     q_table = aq_sess.createQueueTable ("aq", "multi_qtab", qtable_prop);

     /* Create a new AQQueueProperty object: */
     queue_prop = new AQQueueProperty();
     queue = aq_sess.createQueue (q_table, "multi_queue", queue_prop);

}

/* Add subscribers to a queue */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "multi_queue");

     /* add a subscriber */
     agent1 = new AQAgent("subscriber1", "aq2.msg_queue2@london");
     queue.addSubscriber(agent1, null);

     /* add a subscriber with a rule */
     agent2 = new AQAgent("subscriber2", "aq2.msg_queue2@london");

     queue.addSubscriber(agent2, "priority < 2");
}

/* Add a subscriber with a rule */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
```

```
     /* Get the queue object */
     queue = aq_sess.getQueue("OE", "OE_bookedorders_que");

     /* add a subscriber */
     agent1 = new AQAgent("East_Shipping", "ES.ES_bookedorders_que");

     queue.addSubscriber(agent1,
"tab.user_data.orderregion='EASTERN' OR " +
"(tab.user_data.ordertype='RUSH' AND " +
"tab.user_data.customer.country='USA')");
}
```

サブスクライバの変更

図 9-20 サブスクライバの変更



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

指定されたキューのサブスクライバの既存のプロパティを変更します。ルールのみを変更できます。

使用上の注意

ルールまたは変換（あるいはその両方）を変更できます。属性、ルールまたはサブスクライバの変換のいずれか 1 つを変更する場合は、他の属性の既存の値を変更コールに指定します。

キュー、キュー表またはサブスクライバが、作成、変更または削除された場合、`GLOBAL_TOPIC_ENABLED` が `TRUE` であれば、対応する LDAP エントリも作成されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の `ALTER_SUBSCRIBER` プロシージャ
- Visual Basic（OO4O）：参照マニュアルはありません。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の `alterSubscriber`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）：サブスクライバの変更（9-67 ページ）
- Visual Basic（OO4O）：例はありません。
- Java（JDBC）：サブスクライバの変更（9-68 ページ）

PL/SQL（DBMS_AQADM）：サブスクライバの変更

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
   queue_table            => 'aq.multi_qtab',
   multiple_consumers     => TRUE,
   queue_payload_type     => 'aq.message_typ',
   compatible             => '8.1.5');
EXECUTE DBMS_AQADM.CREATE_QUEUE (
   queue_name             =>  'multi_queue',
   queue_table            => 'aq.multi_qtab');
```

```
/* Add a subscriber with a rule: */
DECLARE
   subscriber       sys.aq$_agent;
BEGIN
   subscriber := sys.aq$_agent('SUBSCRIBER1', 'aq2.msg_queue2@london', null);
   DBMS_AQADM.ADD_SUBSCRIBER(
      queue_name         =>    'aq.msg_queue',
      subscriber         =>     subscriber,
      rule               =>    'priority < 2');
END;
/* Change rule for subscriber: */
DECLARE
   subscriber          sys.aq$_agent;
BEGIN
   subscriber := sys.aq$_agent('SUBSCRIBER1', 'aq2.msg_queue2@london', null);
   DBMS_AQADM.ALTER_SUBSCRIBER(
      queue_name         =>    'aq.msg_queue',
      subscriber         =>     subscriber,
      rule              =>     'priority = 1');
END;
```

変換を使用するサブスクライバを追加する

```
/* Add a subscriber with transformation */
EXECUTE DBMS_AQADM.ADD_SUBSCRIBER
   ('aq.msg_queue',
     aq$_agent('subscriber1',
        'aq2.msg_queue2@london',
            null),
'AQ.MSG_MAP1');
/* Alter the subscriber*/
```

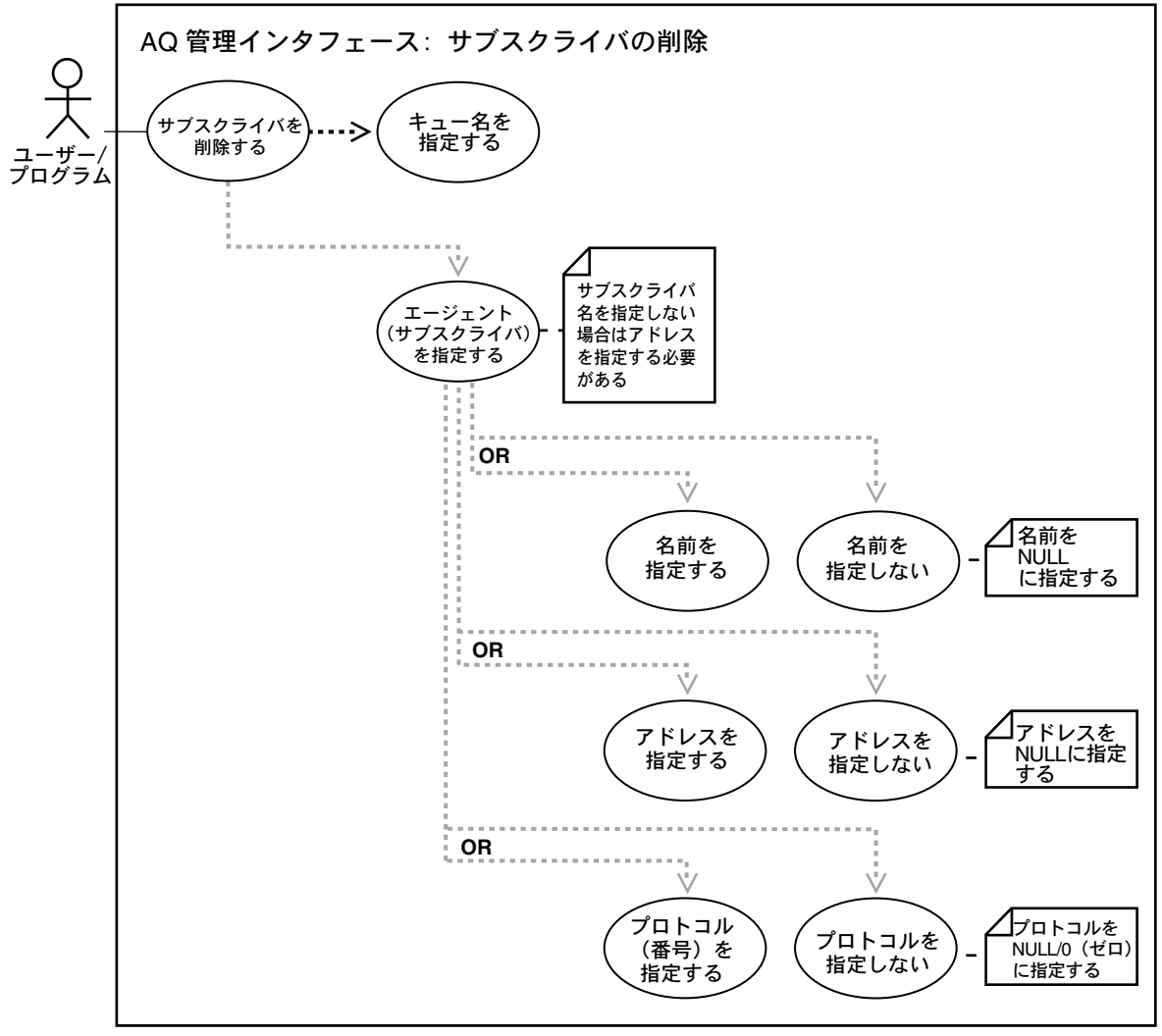
```
EXECUTE DBMS_AQADM.ALTER_SUBSCRIBER
   ('aq.msg_queue',
     aq$_agent ('subscriber1',
      'aq2.msg_queue2@london',
               null),
       'AQ.MSG.MAP2');
```

Java（JDBC）：サブスクライバの変更

```
/* Alter the rule for a subscriber */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "multi_queue");

     /* add a subscriber */
     agent1 = new AQAgent("subscriber1", "aq2.msg_queue2@london");

     queue.alterSubscriber(agent1, "priority=1");
}
```

サブスクライバの削除

図 9-21 サブスクライバの削除



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

デフォルトのサブスクライバをキューから削除します。

使用上の注意

この操作はすぐに有効になり、この操作を含むトランザクションはコミットされます。既存メッセージ内のこのサブスクライバに対するすべての参照は、操作の一部として削除されます。

キュー、キュー表またはサブスクライバが、作成、変更または削除された場合、GLOBAL_TOPIC_ENABLED が TRUE であれば、対応する LDAP エントリも作成されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の REMOVE_SUBSCRIBER プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の removeSubscriber

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: サブスクライバの削除（9-71 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: サブスクライバの削除（9-71 ページ）

PL/SQL（DBMS_AQADM）：サブスクライバの削除

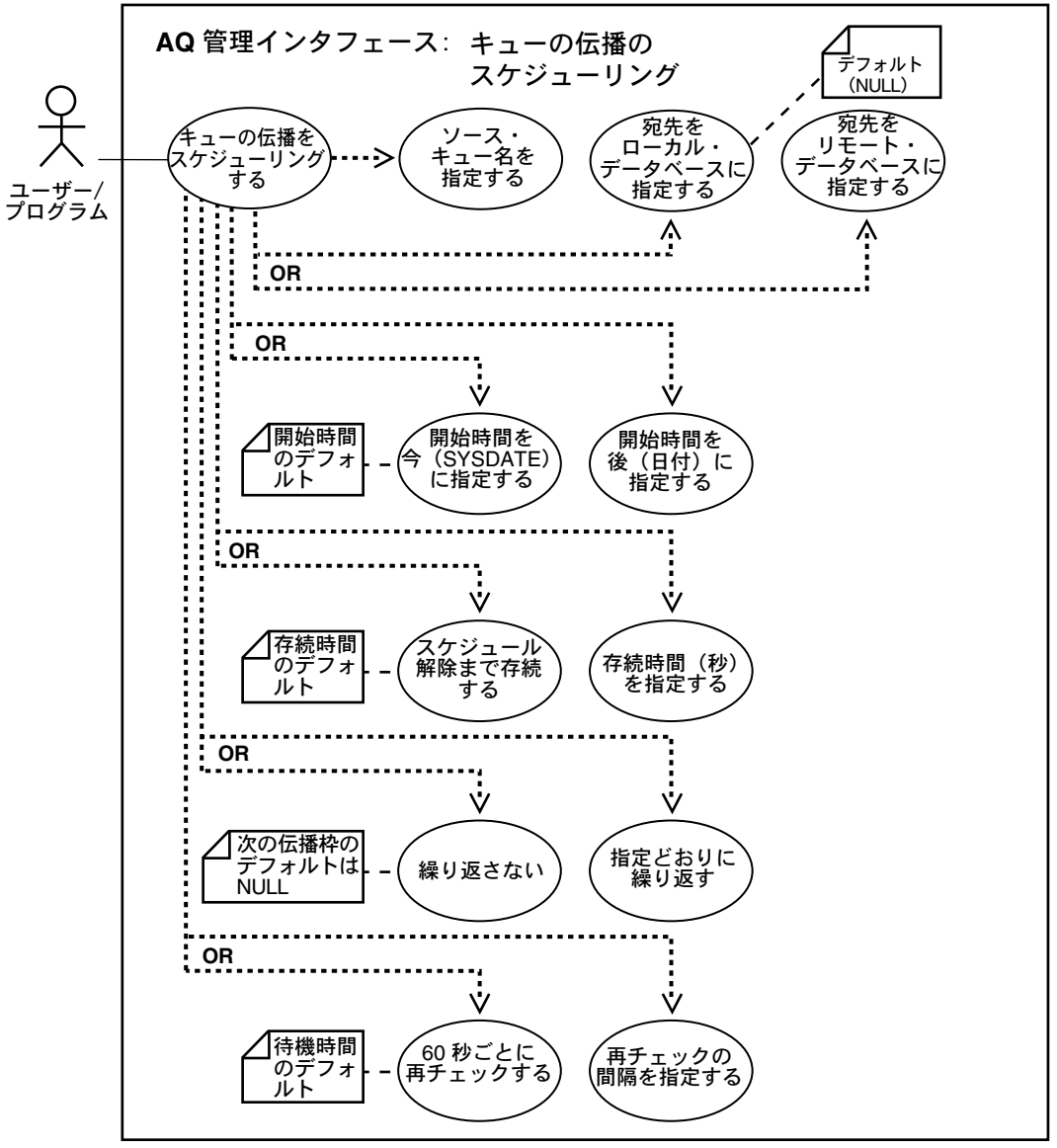
```
DECLARE
   subscriber       sys.aq$_agent;
BEGIN
   subscriber := sys.aq$_agent('subscriber1','aq2.msg_queue2', NULL);
   DBMS_AQADM.REMOVE_SUBSCRIBER(
      queue_name => 'aq.multi_queue',
      subscriber => subscriber);
END;
```

Java（JDBC）：サブスクライバの削除

```
/* Remove a subscriber */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "multi_queue");

     /* add a subscriber */
     agent1 = new AQAgent("subscriber1", "aq2.msg_queue2@london");

     queue.removeSubscriber(agent1);
}
```

キューの伝播のスケジューリング

図 9-22 キューの伝播のスケジューリング



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

あるキューから特定の dblink で識別される宛先へのメッセージ伝播をスケジューリングします。

使用上の注意

宛先に NULL を指定すると、メッセージは同じデータベース内の他のキューにも伝播されます。同じ宛先に複数の受信者を持つ場合、（キューが同じかどうかにかかわらず）メッセージは、すべての受信者に同時に伝播されます。

HTTP または HTTPS を経由したメッセージの伝播の詳細は、第 17 章「AQ へのインターネット・アクセス」を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の SCHEDULE_PROPAGATION プロシージャ
- Visual Basic（OO4O）：参照マニュアルはありません。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の schedulePropagation

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）：キューの伝播のスケジューリング（9-74 ページ）
- Visual Basic（OO4O）：例はありません。
- Java（JDBC）：キューの伝播のスケジューリング（9-74 ページ）

PL/SQL（DBMS_AQADM）：キューの伝播のスケジューリング

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
   queue_table        => 'aq.objmsgs_qtab',
   queue_payload_type => 'aq.message_typ',
   multiple_consumers => TRUE);
EXECUTE DBMS_AQADM.CREATE_QUEUE (
   queue_name         => 'aq.q1def',
   queue_table        => 'aq.objmsgs_qtab');
```

あるキューから同じデータベース内の他のキューへの伝播をスケジューリングする

```
/* Schedule propagation from queue aq.q1def to other queues in the same
   database  */
EXECUTE DBMS_AQADM.SCHEDULE_PROPAGATION(
   Queue_name    =>    'aq.q1def');
```

あるキューから別のデータベース内の他のキューへの伝播をスケジューリングする

```
/* Schedule a propagation from queue aq.q1def to other queues in another
   database */
EXECUTE DBMS_AQADM.SCHEDULE_PROPAGATION(
   Queue_name    =>    'aq.q1def',
   Destination   =>    'another_db.world');
```

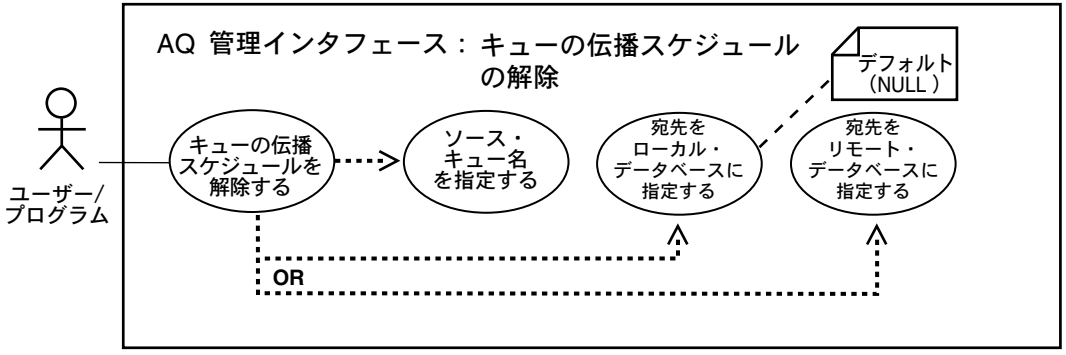
Java（JDBC）：キューの伝播のスケジューリング

```
/* Setup  */
public static void setup(AQSession aq_sess) throws AQException
{
     AQQueueTableProperty    qtable_prop;
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;

     qtable_prop = new AQQueueTableProperty("AQ.MESSAGE_TYP");
     qtable_prop.setMultiConsumer(true);

     q_table = aq_sess.createQueueTable ("aq", "objmsgs_qtab", qtable_prop);
```

```
     /* Create a new AQQueueProperty object: */
     queue_prop = new AQQueueProperty();
     queue = aq_sess.createQueue (q_table, "q1def", queue_prop);
}

/* Schedule propagation from a queue to other queues in the same database */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.scheduleePropagation(null, null, null, null, null);
}

/* Schedule propagation from a queue to other queues in another database */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.scheduleePropagation("another_db.world", null, null, null, null);
}
```

キューの伝播スケジュールの解除

図 9-23 キューの伝播スケジュールの解除



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

あるキューから特定の dblink で識別される宛先に対して設定されていたメッセージ伝播スケジュールを解除します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の UNSCHEDULE_PROPAGATION プロシージャ
- Visual Basic（OO4O）：参照マニュアルはありません。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の schedulePropagation

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 伝播スケジュールの解除（9-77 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: キューの伝播スケジュールの解除（9-77 ページ）

PL/SQL（DBMS_AQADM）: 伝播スケジュールの解除

あるキューから同じデータベース内の他のキューへの伝播スケジュールを解除する

```
/* Unschedule propagation from queue aq.q1def to other queues in the same
   database */
EXECUTE DBMS_AQADM.UNSCHEDULE_PROPAGATION(queue_name => 'aq.q1def');
```

あるキューから別のデータベース内の他のキューへの伝播スケジュールを解除する

```
/* Unschedule propagation from queue aq.q1def to other queues in another
   database reached by the database link another_db.world */
EXECUTE DBMS_AQADM.UNSCHEDULE_PROPAGATION(
   Queue_name     =>   'aq.q1def',
   Destination    =>   'another_db.world');
```

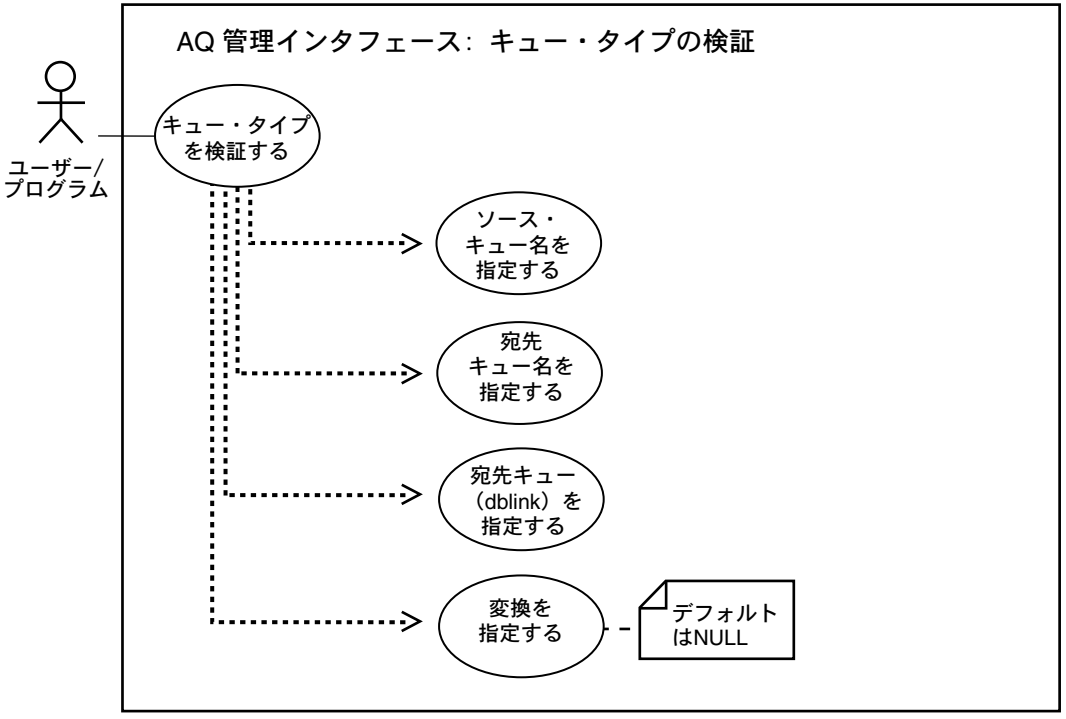
Java（JDBC）: キューの伝播スケジュールの解除

```
/* Unschedule propagation from a queue to other queues in the same database */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.unschedulePropagation(null);
}

/* Unschedule propagation from a queue to other queues in another database */
public static void example(AQSession aq_sess) throws AQException
{
```

```
        AQQueue         queue;
        AQAgent         agent1;
        AQAgent         agent2;

        /* Get the queue object */
        queue = aq_sess.getQueue("AQ", "q1def");

        queue.unschedulePropagation("another_db.world");
    }
```

キュー・タイプの検証

図 9-24 キュー・タイプの検証



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

ソースおよび宛先が同じ型を持っているかどうかを検証します。検証の結果は、SYS.AQ$_MESSAGE_TYPES 表に格納され、以前にこのコマンドから出力されたすべての結果は上書きされます。

使用上の注意

ソースおよび宛先に同じ型のキューがない場合に指定された変換では、ソース・キューの型を宛先キューの型にマップする必要があります。

注意： `sys.aq$_message_types` 表は、同じソース・キュー、宛先キューおよび `dblink` に対して複数のエントリを持ちますが、変換は異なります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の `VERIFY_QUEUE_TYPES` プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）: 参照マニュアルはありません。

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: キュー・タイプの検証（9-81 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

PL/SQL（DBMS_AQADM）：キュー・タイプの検証

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
EXECUTE DBMS_AQADM.CREATE_QUEUE (
   queue_name         => 'aq.q2def',
   queue_table        => 'aq.objmsgs_qtab');
```

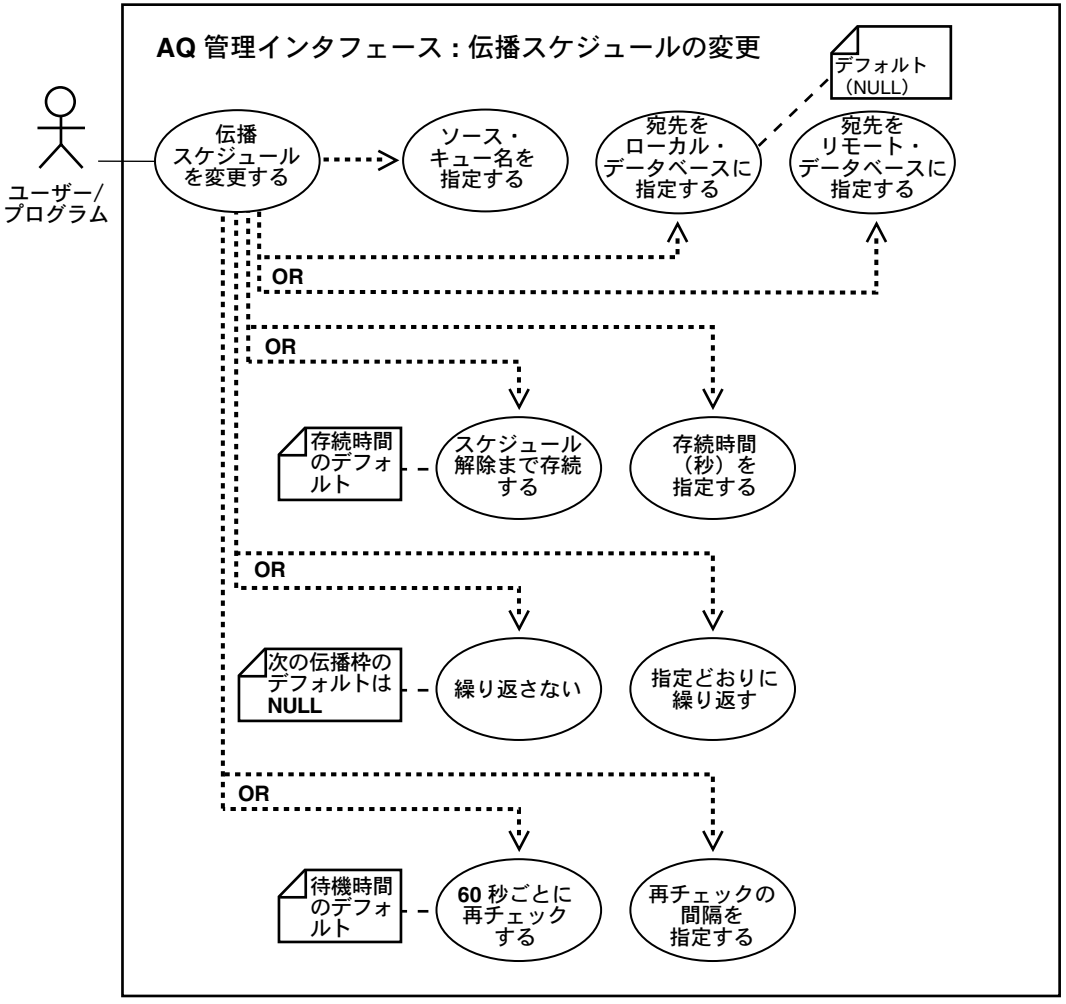
```
/*  Verify if the source and destination queues have the same type. The
    function has the side effect of inserting/updating the entry for the source
    and destination queues in the dictionary table AQ$_MESSAGE_TYPES */
DECLARE
rc      BINARY_INTEGER;
BEGIN
/* Verify if the queues aquser.q1def and aquser.q2def in the local database
   have the same payload type */
   DBMS_AQADM.VERIFY_QUEUE_TYPES(
      src_queue_name  => 'aq.q1def',
      dest_queue_name => 'aq.q2def',
      rc              => rc);
   DBMS_OUTPUT.PUT_LINE(rc);
END;
```

Java（JDBC）：キュー・タイプの検証

この機能は、Java API を介しては使用できません。

伝播スケジュールの変更

図 9-25 伝播スケジュールの変更



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途
伝播スケジュールのパラメータを変更します。

使用上の注意
ありません。

構文
各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の ALTER_PROPAGATION_SCHEDULE プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の alterPropagationSchedule

例
各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 伝播スケジュールの変更（9-83 ページ）
- Visual Basic（OO4O）: 例はありません。
- PL/SQL（DBMS_AQADM）: 伝播スケジュールの変更（9-84 ページ）

PL/SQL（DBMS_AQADM）: 伝播スケジュールの変更

あるキューから同じデータベース内の他のキューへの伝播スケジュールを変更する

```
/* Alter schedule from queue aq.q1def to other queues in the same database */
EXECUTE DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE(
   Queue_name     =>    'aq.q1def',
   Duration       =>    '2000',
   Next_time      =>    'SYSDATE + 3600/86400',
   Latency        =>    '32');
```

あるキューから別のデータベース内の他のキューへの伝播スケジュールを変更する

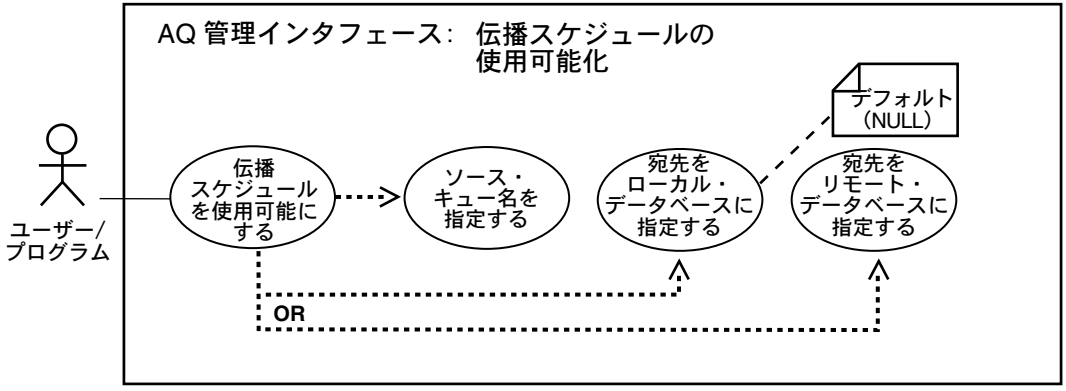
```
/* Alter schedule from queue aq.q1def to other queues in another database
reached by the database link another_db.world */
EXECUTE DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE(
   Queue_name    =>    'aq.q1def',
   Destination   =>    'another_db.world',
   Duration      =>    '2000',
   Next_time     =>    'SYSDATE + 3600/86400',
   Latency       =>    '32');
```

Java（JDBC）：伝播スケジュールの変更

```
/* Alter propagation schedule from a queue to other queues
   in the same database */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.alterPropagationSchedule(null, new Double(2000),
   "SYSDATE + 3600/86400", new Double(32));
}

/* Unschedule propagation from a queue to other queues in another database */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.alterPropagationSchedule("another_db.world", new Double(2000),
   "SYSDATE + 3600/86400", new Double(32));
}
```

伝播スケジュールの使用可能化

図 9-26 伝播スケジュールの使用可能化



参照: 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

以前に使用不可にされた伝播スケジュールを使用可能にします。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の `ENABLE_PROPAGATION_SCHEDULE` プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の `enablePropagationSchedule`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 伝播の使用可能化（9-86 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 伝播スケジュールの使用可能化（9-86 ページ）

PL/SQL（DBMS_AQADM）: 伝播の使用可能化

あるキューから同じデータベース内の他のキューへの伝播を使用不可能にする

```
/* Enable propagation from queue aq.q1def to other queues in the same
   database */
EXECUTE DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE(
   Queue_name    =>   'aq.q1def');
```

あるキューから別のデータベース内の他のキューへの伝播を使用可能にする

```
/* Enable propagation from queue aq.q1def to other queues in another
  database reached by the database link another_db.world */
EXECUTE DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE(
   Queue_name     =>    'aq.q1def',
   Destination    =>    'another_db.world');
```

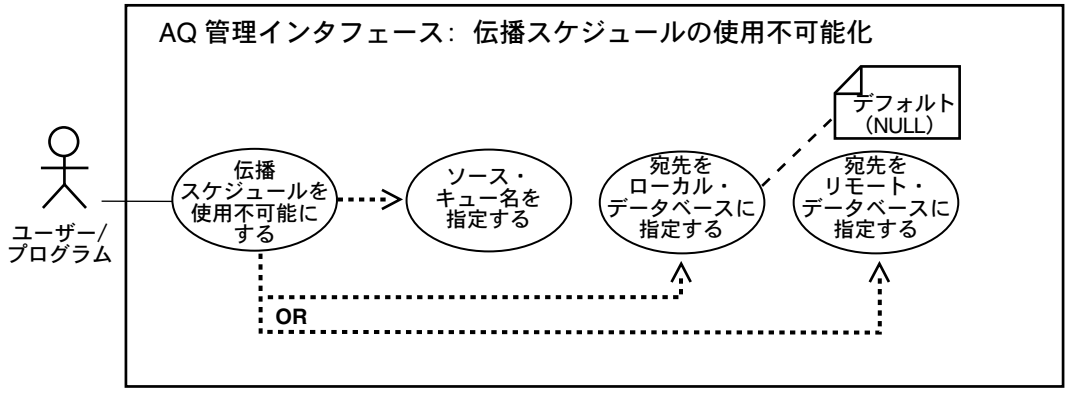
Java（JDBC）: 伝播スケジュールの使用可能化

```
/* Enable propagation from a queue to other queues in the same database */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue          queue;
     AQAgent          agent1;
     AQAgent          agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.enablePropagationSchedule(null);
}

/* Enable propagation from a queue to other queues in another database */
```

```
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue         queue;
     AQAgent         agent1;
     AQAgent         agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.enablePropagationSchedule("another_db.world");
}
```

伝播スケジュールの使用不可能化

図 9-27 伝播スケジュールの使用不可能化



参照: 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

以前に使用可能にされた伝播スケジュールを使用不可能にします。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の `DISABLE_PROPAGATION_SCHEDULE` プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の `disablePropagationSchedule`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 伝播スケジュールの使用不可能化（9-89 ページ）
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 伝播スケジュールの使用不可能化（9-89 ページ）

PL/SQL（DBMS_AQADM）: 伝播スケジュールの使用不可能化

あるキューから同じデータベース内の他のキューへの伝播を使用不可能にする

```
/* Disable a propagation from queue aq.q1def to other queues in the same
   database */
EXECUTE DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE(
   Queue_name    =>   'aq.q1def');
```

あるキューから別のデータベース内の他のキューへの伝播を使用不可能にする

```
/* Disable a propagation from queue aq.q1def to other queues in another
  database reached by the database link another_db.world */
EXECUTE DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE(
   Queue_name     =>    'aq.q1def',
   Destination    =>    'another_db.world');
```

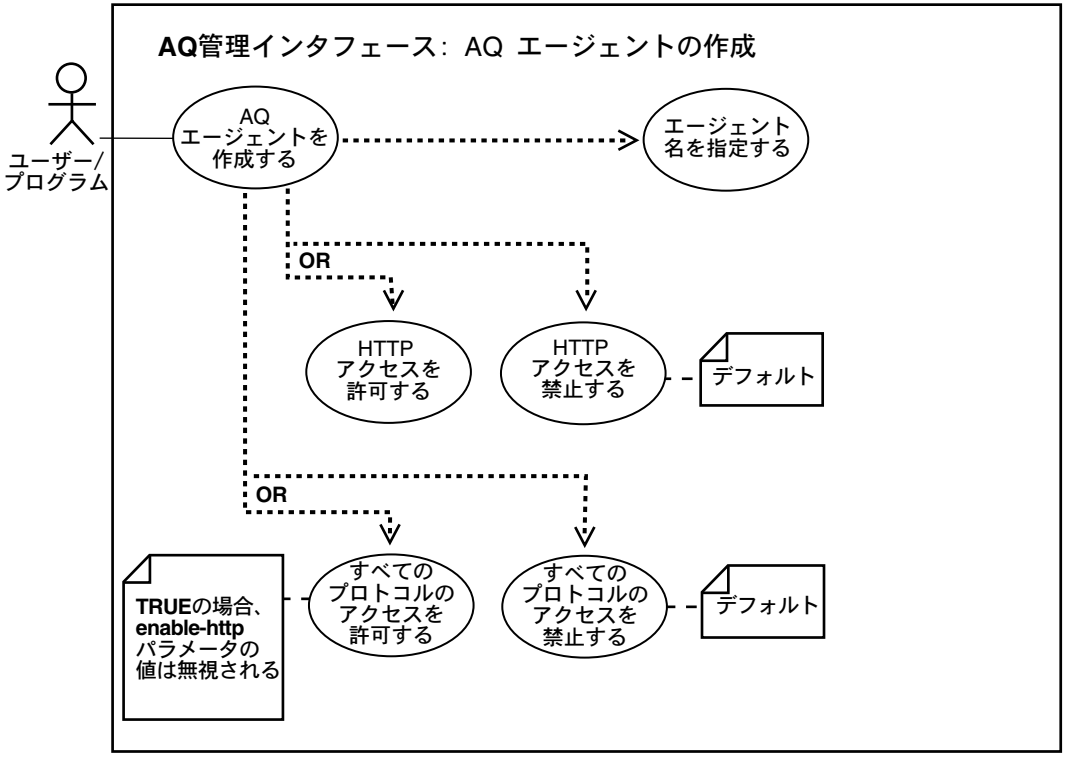
Java（JDBC）: 伝播スケジュールの使用不可能化

```
/* Disable propagation from a queue to other queues in the same database */
public static void example(AQSession aq_sess) throws AQException
{
     AQQueue          queue;
     AQAgent          agent1;
     AQAgent          agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.disablePropagationSchedule(null);
}

/* Disable propagation from a queue to other queues in another database */
public static void example(AQSession aq_sess) throws AQException
```

```
{
     AQQueue          queue;
     AQAgent          agent1;
     AQAgent          agent2;

     /* Get the queue object */
     queue = aq_sess.getQueue("AQ", "q1def");

     queue.disablePropagationSchedule("another_db.world");
}
```

AQ エージェントの作成

図 9-28 AQ エージェントの作成



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

HTTP プロトコルを使用して、AQ インターネット・アクセスのエージェントを登録します。

使用上の注意

SYS.AQ$INTERNET_USERS ビューには、AQ インターネット・エージェントのすべてのリストがあります。

AQ エージェントが作成、変更または削除された場合、次の条件が満たされていれば、そのエージェントに対して LADP エントリが作成されます。

- GLOBAL_TOPIC_ENABLED が TRUE である。
- certificate_location が指定されている。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の CREATE_AQ_AGENT プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」

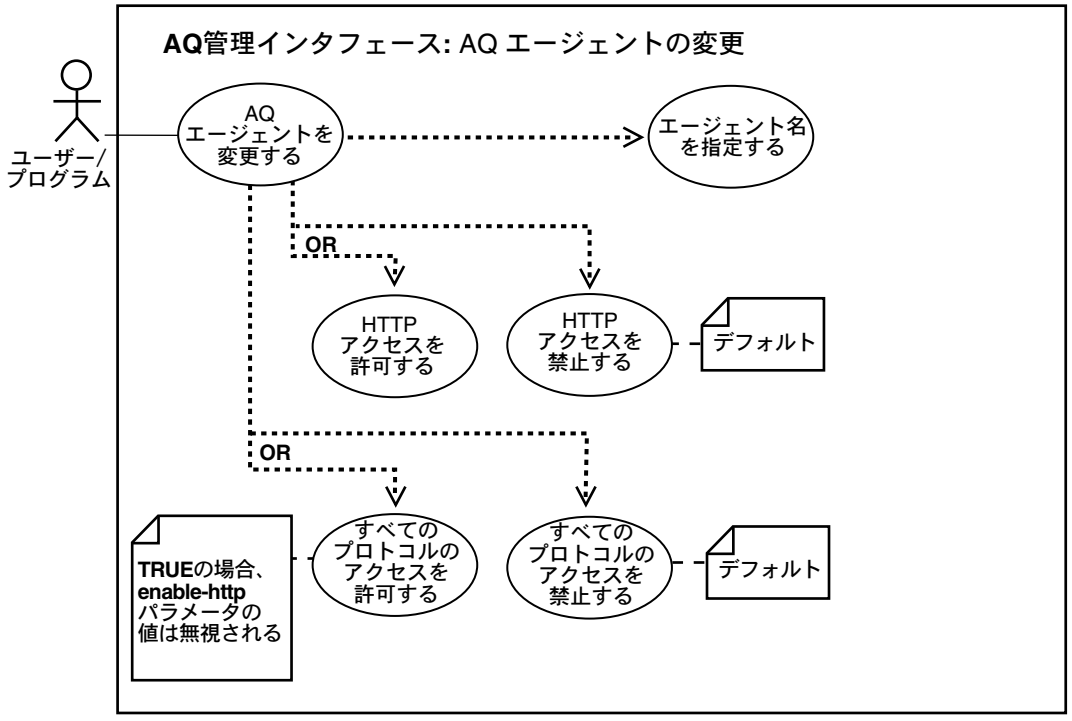
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 例はありません。
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

AQ エージェントの変更

図 9-29 AQ エージェントの変更



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

AQ インターネット・アクセスに登録されたエージェントを変更します。

使用上の注意

AQ エージェントが作成、変更または削除された場合、次の条件が満たされていれば、そのエージェントに対して LADP エントリが作成されます。

- `GLOBAL_TOPIC_ENABLED` が `TRUE` である。
- `certificate_location` が指定されている。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の ALTER_AQ_AGENT プロシージャ

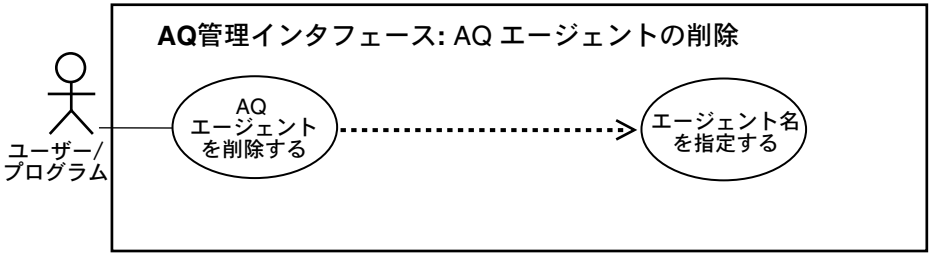
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 例はありません。
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

AQ エージェントの削除

図 9-30 AQ エージェントの削除



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

以前 AQ インターネット・アクセスに登録されたエージェントを削除します。

使用上の注意

AQ エージェントが作成、変更または削除された場合、次の条件が満たされていれば、そのエージェントに対して LADP エントリが作成されます。

- `GLOBAL_TOPIC_ENABLED` が `TRUE` である。
- `certificate_location` が指定されている。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の `DROP_AQ_AGENT` プロシージャ

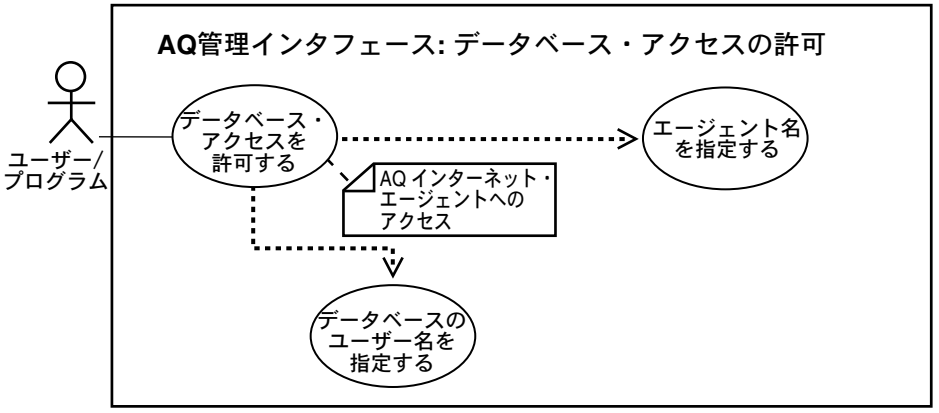
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 例はありません。
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

データベース・アクセスの許可

図 9-31 データベース・アクセスの許可



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

AQ インターネット・エージェントに、特定のデータベース・ユーザーに対する権限を付与します。この AQ インターネット・エージェントは、CREATE_AQ_AGENT プロシージャを使用して事前に作成されています。

使用上の注意

SYS.AQ$INTERNET_USERS ビューに、すべての AQ インターネット・エージェントおよび権限を付与されているデータベース・ユーザーのリストがあります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の ENABLE_DB_ACCESS プロシージャ

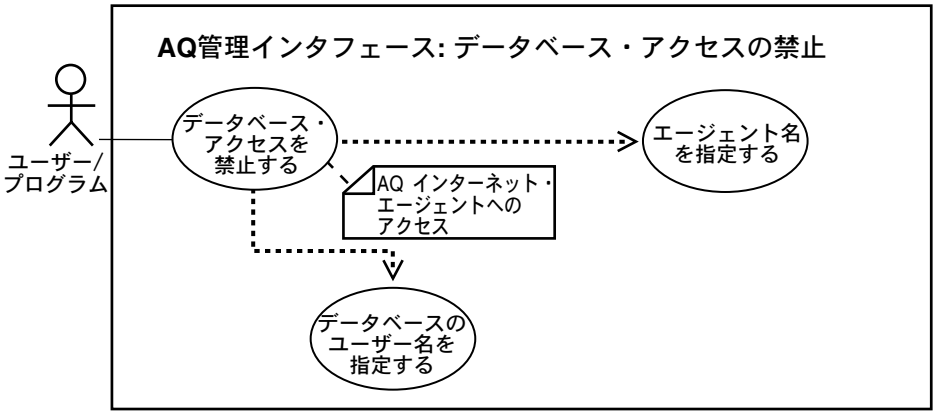
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 例はありません。
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

データベース・アクセスの禁止

図 9-32 データベース・アクセスの禁止



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

AQ インターネット・エージェントから、特定のデータベース・ユーザーに対する権限を取り消します。これらの権限は、AQ インターネット・エージェントが ENABLE_DB_ACCESS プロシージャを使用して事前に付与しています。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の DISABLE_DB_ACCESS プロシージャ

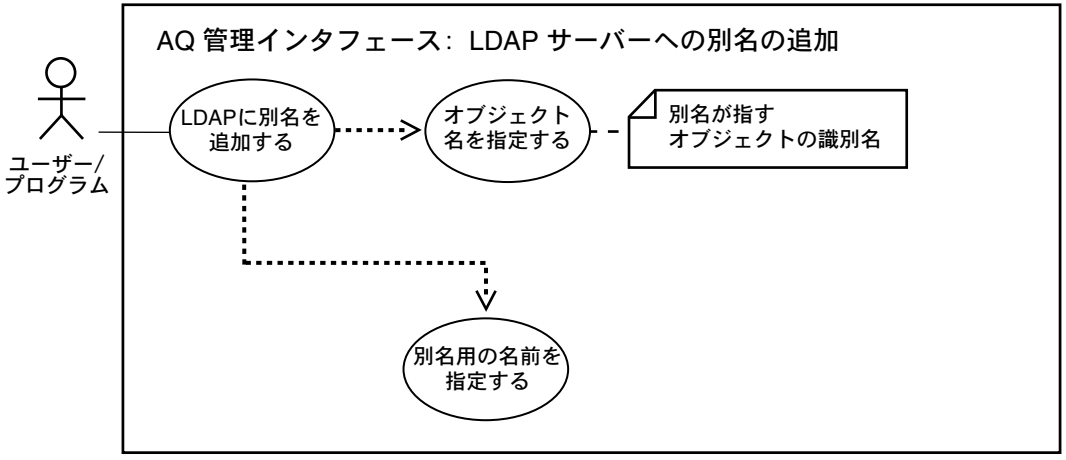
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 例はありません。
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

LDAP サーバーへの別名の追加

図 9-33 LDAP サーバーへの別名の追加



参照: 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

LDAP サーバーに別名を追加します。

使用上の注意

このコールは、LDAP の AQ オブジェクトの別名および識別名を使用して、AQ オブジェクトを指す別名を作成します。この別名は、データベース・サーバーの識別名の直下に配置されます。別名が指すオブジェクトは、キュー、エージェントまたはコネクション・ファクトリです。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の ADD_ALIAS_TO_LDAP プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」

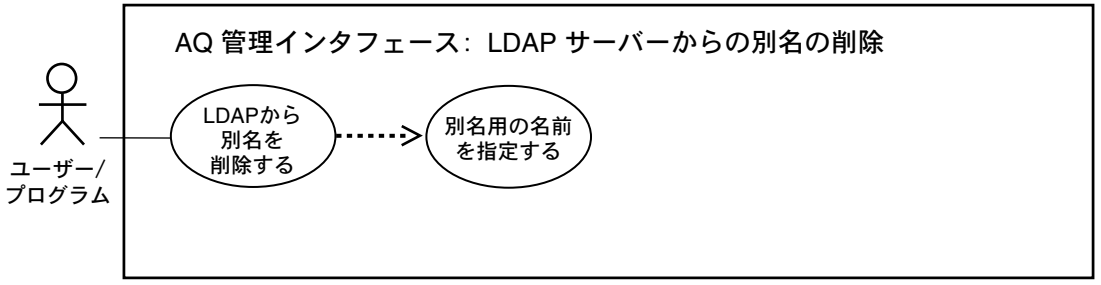
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 例はありません。
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

LDAP サーバーからの別名の削除

図 9-34 LDAP サーバーからの別名の削除



参照： 管理インタフェースの基本操作の詳細は、表 9-1 を参照してください。

用途

LDAP サーバーから別名を削除します。

使用上の注意

このコールは、引数に別名を使用して、LDAP サーバー内の別名エントリを削除します。別名は、LDAP ディレクトリのデータベース・サーバーの直下に配置されているとします。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQADM）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 6 章「DBMS_AQADM」の `DEL_ALIAS_FROM_LDAP` プロシージャ
- Visual Basic（OO4O）: 参照マニュアルはありません。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQADM）: 例はありません。
- Visual Basic（OO4O）: 例はありません。
- Java（JDBC）: 例はありません。

10

管理インタフェース：ビュー

この章では、それぞれの操作（データベース内のすべてのキュー表の選択など）を、その操作名ごとに利用方法に沿って説明します。表 10-1 に、すべての利用方法をまとめて示します。

利用方法は、次の形式で説明されています。

- **利用図：**利用方法と状態図を組み合せた複合図に沿ってビューの管理インタフェースを説明します。利用方法の説明では、各ビューを代表的な操作（データベース内のすべてのキュー表の選択など）に沿って説明します。状態図では、各ビューの各属性の状態を表します。どの属性（列）も、状態は可視または不可視のいずれかです。
- **構文：**このアクティビティの実行に使用する構文です。

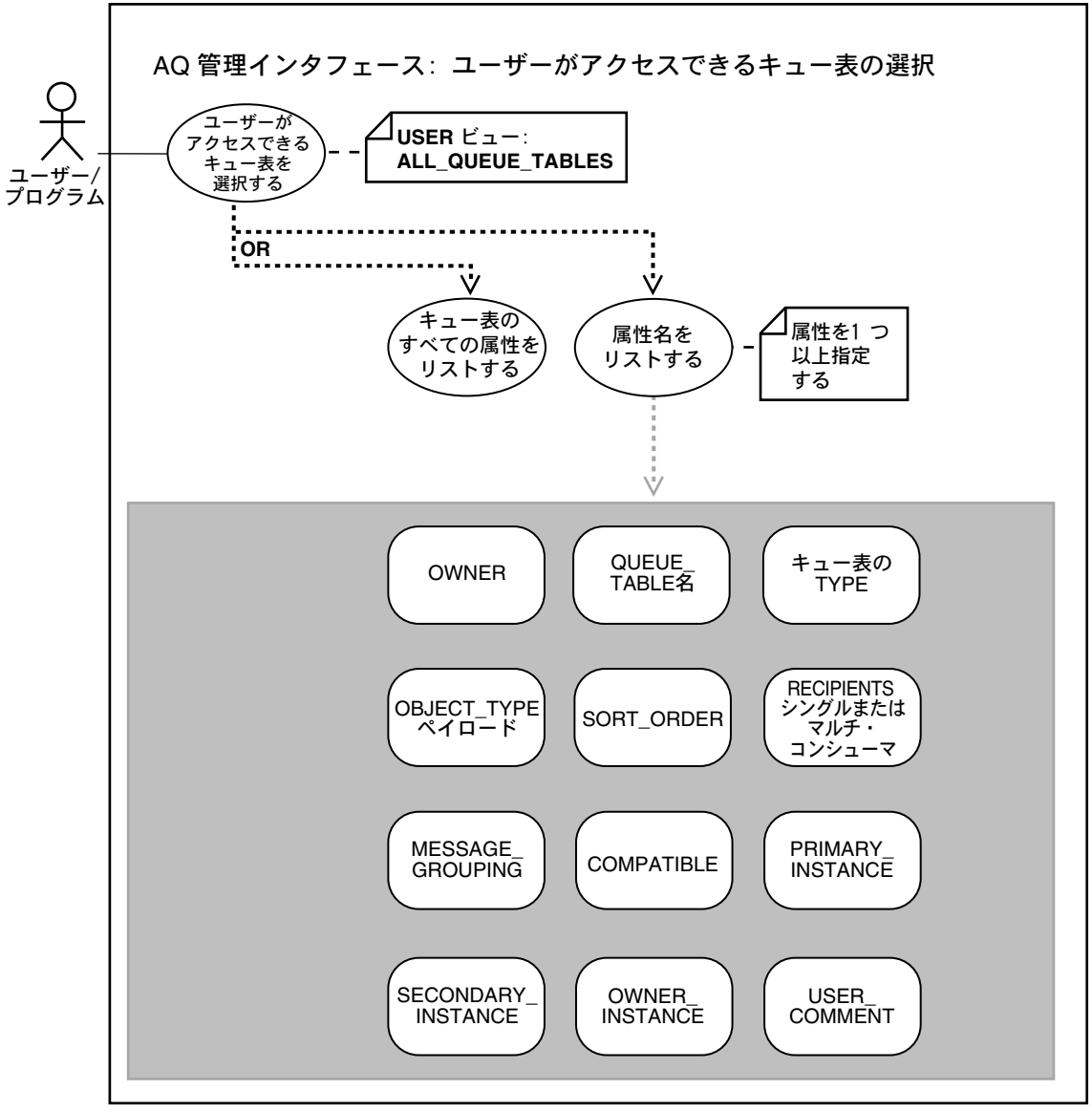
利用モデル : 管理インタフェース - ビュー

表 10-1 利用モデル : 管理インタフェース - ビュー

利用方法	ビュー名
データベース内のすべてのキュー表の選択（10-3 ページ）	DBA_QUEUE_TABLES
ユーザーのキュー表の選択（10-5 ページ）	ALL_QUEUE_TABLES
データベース内のすべてのキューの選択（10-7 ページ）	DBA_QUEUES
すべての伝播スケジュールの選択（10-9 ページ）	DBA_QUEUE_SCHEDULES
ユーザーがなんらかの権限を持っているキューの選択（10-13 ページ）	ALL_QUEUES
ユーザーがキュー権限を持っているキューの選択（10-15 ページ）	QUEUE_PRIVILEGES
キュー表のメッセージの選択（10-17 ページ）	AQ$< キュー表名 >
ユーザー・スキーマのキュー表の選択（10-21 ページ）	USER_QUEUE_TABLES
ユーザー・スキーマのキューの選択（10-23 ページ）	USER_QUEUES
ユーザー・スキーマの伝播スケジュールの選択（10-25 ページ）	USER_QUEUE_SCHEDULES
キューのサブスクライバの選択（10-29 ページ）	AQ$< キュー表名 >_S
キューのサブスクライバおよびそのルールの選択（10-31 ページ）	AQ$< キュー表名 >_R
データベース全体における状態ごとのメッセージ数の選択（10-33 ページ）	GV$AQ
特定のインスタンスにおける状態ごとのメッセージ数の選択（10-35 ページ）	V$AQ
インターネット・アクセスに登録された AQ エージェントの選択（10-37 ページ）	AQ$INTERNET_USERS
ユーザー変換の選択（10-39 ページ）	USER_TRANSFORMATIONS
ユーザー変換ファンクションの選択（10-40 ページ）	USER_ATTRIBUTE_TRANSFORMATIONS
すべての変換の選択（10-41 ページ）	DBA_TRANSFORMATIONS
すべての変換ファンクションの選択（10-43 ページ）	DBA_ATTRIBUTE_TRANSFORMATIONS

データベース内のすべてのキュー表の選択

図 10-1 データベース内のすべてのキュー表の選択



参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

DBA_QUEUE_TABLES

用途

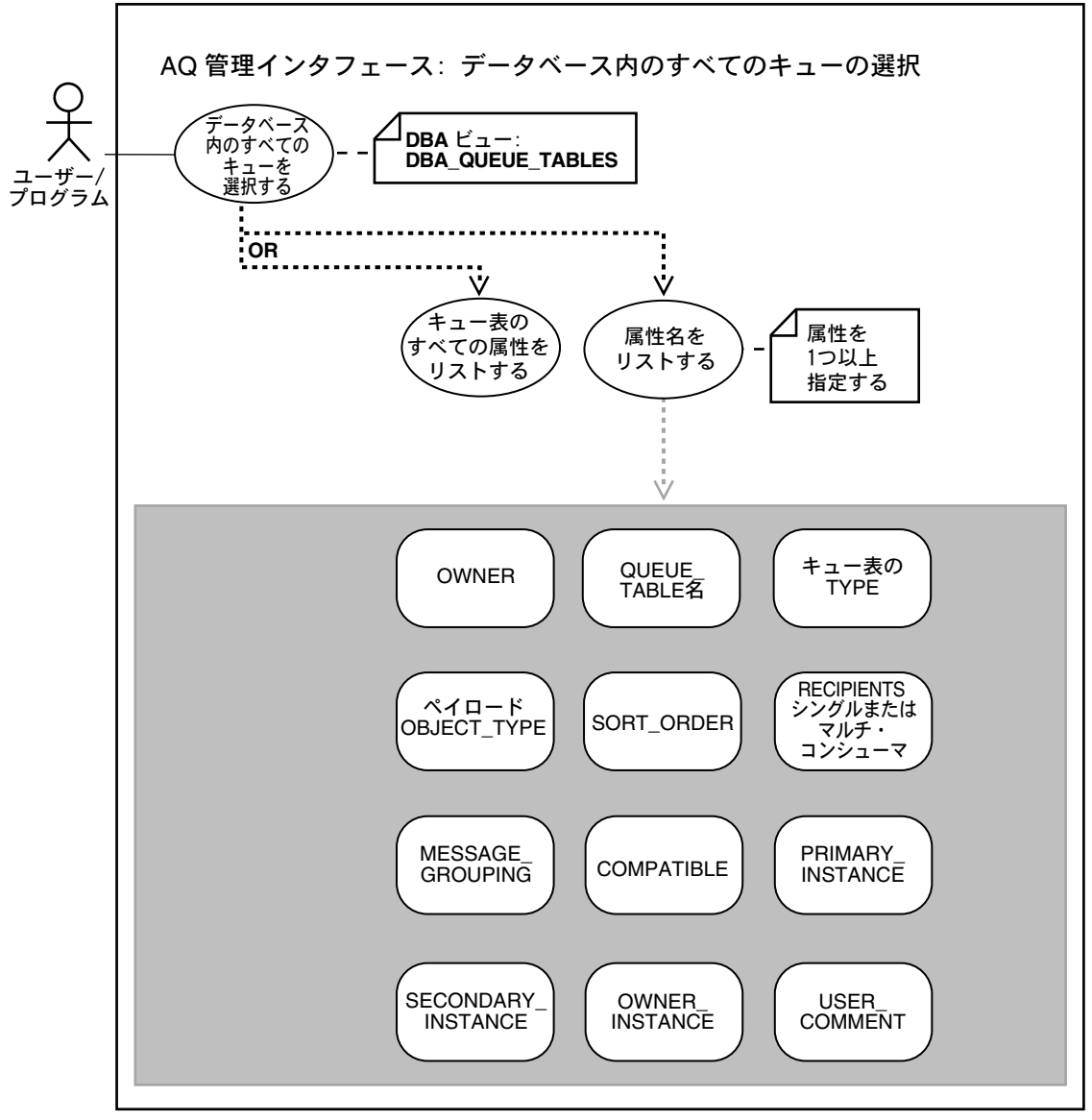
このビューは、データベース内で作成されたすべてのキュー表の名前および型を示します。

表 10-2 DBA_QUEUE_TABLES

列名および説明	NULL かどうか	型
OWNER - キュー表スキーマ	-	VARCHAR2(30)
QUEUE_TABLE - キュー表名	-	VARCHAR2(30)
TYPE - ペイロード型	-	VARCHAR2(7)
OBJECT_TYPE - オブジェクト型が定義されている場合、その型名	-	VARCHAR2(61)
SORT_ORDER - ユーザー指定のソート順	-	VARCHAR2(22)
RECIPIENTS - SINGLE または MULTIPLE	-	VARCHAR2(8)
MESSAGE_GROUPING - NONE または TRANSACTIONAL	-	VARCHAR2(13)
COMPATIBLE - そのキュー表が互換性を持つ最も低いバージョン	-	VARCHAR2(5)
PRIMARY_INSTANCE - そのキュー表の 1 次所有者であるインスタンス。値が 0（ゼロ）のときは、1 次所有者が指定されていません。	-	NUMBER
SECONDARY_INSTANCE - そのキュー表の 2 次所有者であるインスタンス。1 次所有者が存在しないときにキュー表の所有者になります。値が 0（ゼロ）のときは、2 次所有者が指定されていません。	-	NUMBER
OWNER_INSTANCE - 現在のキュー表の所有者インスタンス	-	NUMBER
USER_COMMENT - キュー表に対するユーザー・コメント	-	VARCHAR2(50)

ユーザーのキュー表の選択

図 10-2 ユーザーのキュー表の選択



参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

ALL_QUEUE_TABLES

用途

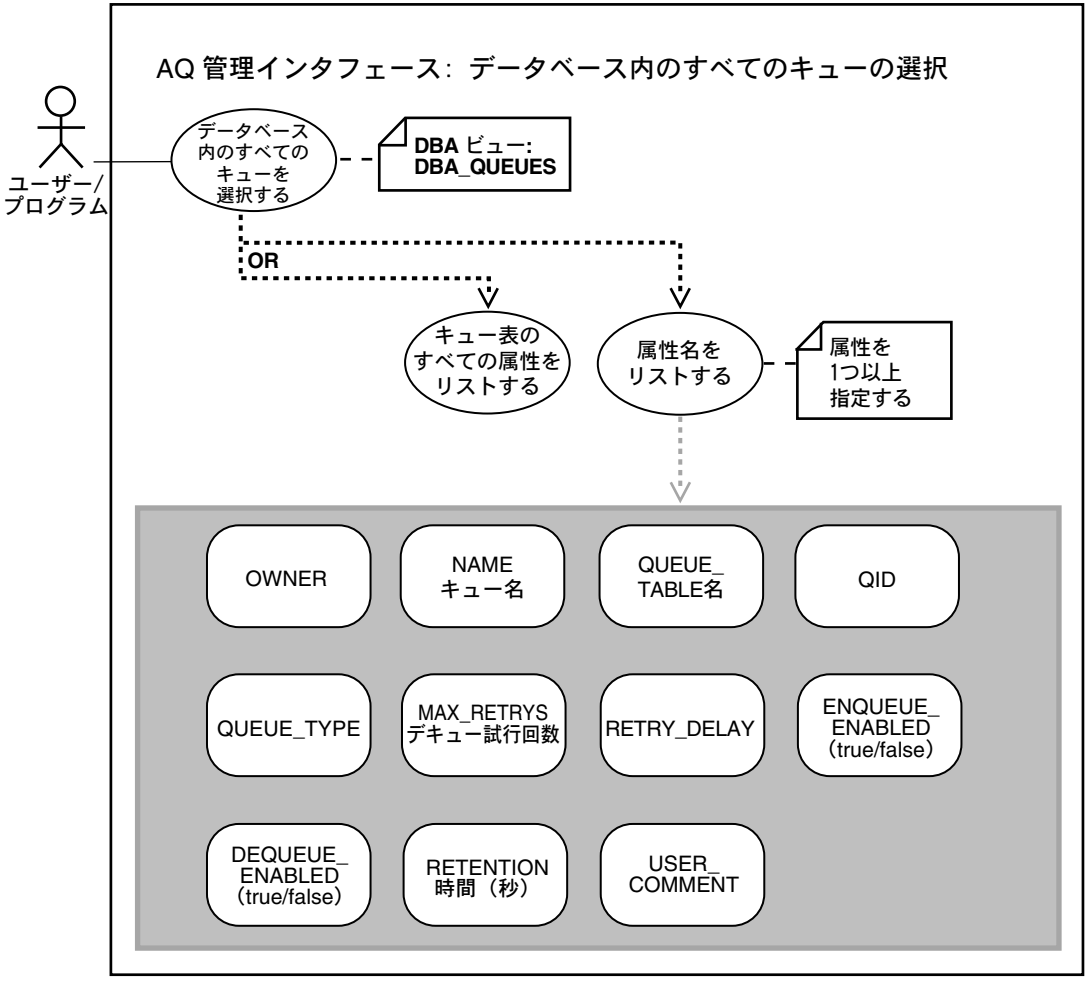
このビューは、ユーザーがアクセスできるキュー表を示します。

表 10-3 DBA_QUEUE_TABLES

列名および説明	NULL かどうか	型
OWNER - キュー表の所有者	-	VARCHAR2(30)
QUEUE_TABLE - キュー表名	-	VARCHAR2(30)
TYPE - ペイロード型	-	VARCHAR2(7)
OBJECT_TYPE - オブジェクト型が定義されている場合、その型名	-	VARCHAR2(61)
SORT_ORDER - ユーザー指定のソート順	-	VARCHAR2(22)
RECIPIENTS - SINGLE または MULTIPLE の受信者のキュー	-	VARCHAR2(8)
MESSAGE_GROUPING - NONE または TRANSACTIONAL	-	VARCHAR2(13)
COMPATIBLE - そのキュー表が互換性を持つ最も低いバージョン	-	VARCHAR2(5)
PRIMARY_INSTANCE - そのキュー表の 1 次所有者であるインスタンス。値が 0（ゼロ）のときは、1 次所有者が指定されていません。	-	NUMBER
SECONDARY_INSTANCE - そのキュー表の 2 次所有者であるインスタンス。1 次所有者が存在しないときにキュー表の所有者になります。値が 0（ゼロ）のときは、2 次所有者が指定されていません。	-	NUMBER
OWNER_INSTANCE - 現在のキュー表の所有者インスタンス	-	NUMBER
USER_COMMENT - キュー表に対するユーザー・コメント	-	VARCHAR2(50)

データベース内のすべてのキューの選択

図 10-3 データベースのすべてのキューの選択



参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

DBA_QUEUES

用途

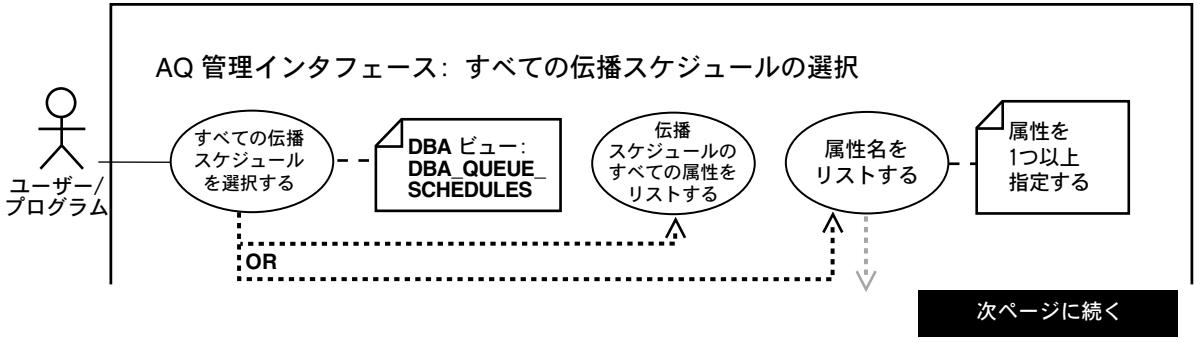
ユーザーは、個々のキューについて操作上の特性を指定できます。DBA_QUEUES には、データベース内のすべてのキューについての関連情報を含むビューが含まれます。

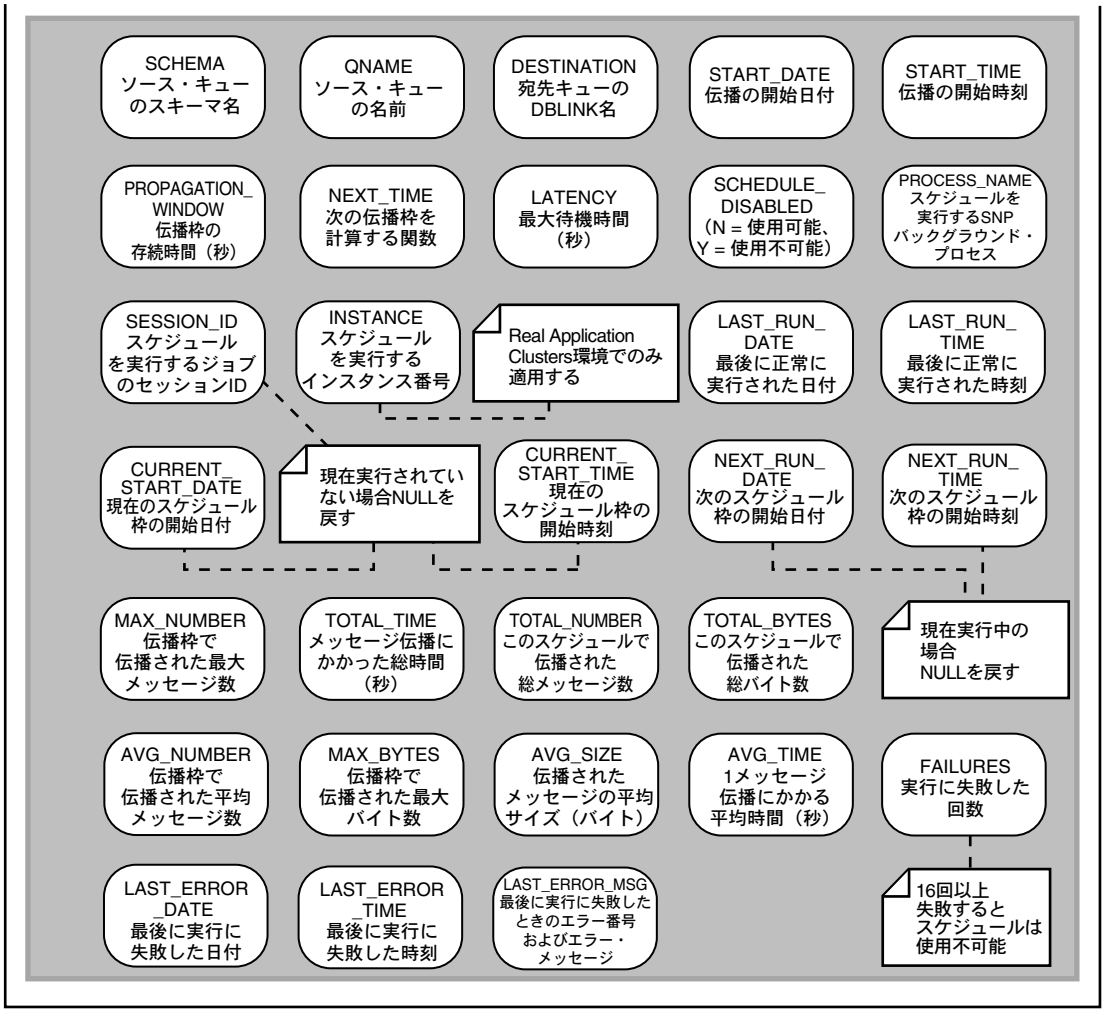
表 10-4 DBA_QUEUES

列名および説明	NULL かどうか	型
OWNER - キューのスキーマ名	NOT NULL	VARCHAR2(30)
NAME - キュー名	NOT NULL	VARCHAR2(30)
QUEUE_TABLE - このキューが格納されるキュー表名	NOT NULL	VARCHAR2(30)
QID - 一意のキュー識別子	NOT NULL	NUMBER
QUEUE_TYPE - キューの型	-	VARCHAR2(15)
MAX_RETRIES - デキューの許容試行回数	-	NUMBER
RETRY_DELAY - 再試行までの秒数	-	NUMBER
ENQUEUE_ENABLED - YES/NO	-	VARCHAR2(7)
DEQUEUE_ENABLED - YES/NO	-	VARCHAR2(7)
RETENTION - メッセージがデキュー後に保存される秒数	-	VARCHAR2(40)
USER_COMMENT - キューに対するユーザー・コメント	-	VARCHAR2(50)

すべての伝播スケジュールの選択

図 10-4 すべての伝播スケジュールの選択





参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

DBA_QUEUE_SCHEDULES

用途

このビューは、現在のメッセージ伝播スケジュールを示します。

表 10-5 DBA_QUEUE_SCHEDULES

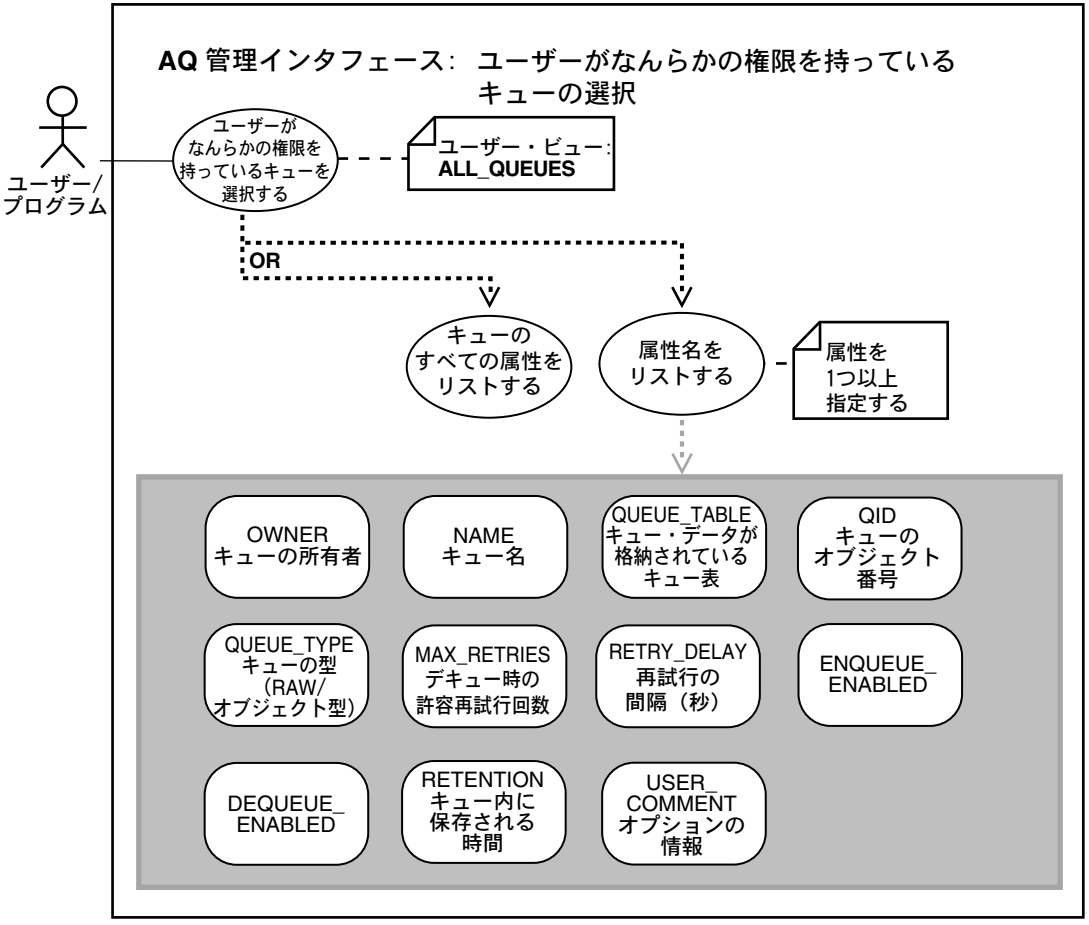
列名および説明	NULL かどうか	型
SCHEMA - ソース・キューのスキーマ名	NOT NULL	VARCHAR2(30)
QNAME - ソース・キューの名前	NOT NULL	VARCHAR2(30)
DESTINATION - 宛先名。現在は DBLINK 名に制限されています。	NOT NULL	VARCHAR2(128)
START_DATE - デフォルトの日付書式による伝播の開始日付	-	DATE
START_TIME - HH:MI:SS フォーマットによる伝播の開始時刻	-	VARCHAR2(8)
PROPAGATION_WINDOW - 伝播枠の存続期間（秒）	-	NUMBER
NEXT_TIME - 次の伝播枠の開始を計算する関数	-	VARCHAR2(200)
LATENCY - 伝播枠内で、メッセージを伝播するまでの最大待機時間	-	NUMBER
SCHEDULE_DISABLED - 使用可能のときは N、使用不可能でスケジュールが実行されていないときは Y	-	VARCHAR(1)
PROCESS_NAME - このスケジュールを実行する SNP バックグラウンド・プロセス。実行されていないときは NULL になります。	-	VARCHAR2(8)
SESSION_ID - このスケジュールを実行しているジョブのセッション ID（SID,SERIAL#）。実行されていないときは NULL になります。	-	NUMBER
INSTANCE - このスケジュールを実行している Real Application Clusters インスタンス番号	-	NUMBER
LAST_RUN_DATE - 最後に正常に実行された日付	-	DATE
LAST_RUN_TIME - 最後に正常に実行された時刻（HH:MI:SS フォーマット）	-	VARCHAR2(8)
CURRENT_START_DATE - このスケジュール枠が開始された日付	-	DATE

表 10-5 DBA_QUEUE_SCHEDULES（続き）

列名および説明	NULL かどうか	型
CURRENT_START_TIME - このスケジュール枠が開始された時刻（HH:MI:SS フォーマット）	-	VARCHAR2(8)
NEXT_RUN_DATE - 次のスケジュール枠が開始される日付	-	DATE
NEXT_RUN_TIME - 次のスケジュール枠が開始される時刻（HH:MI:SS フォーマット）	-	VARCHAR2(8)
TOTAL_TIME - このスケジュールでメッセージの伝播に費やされた総時間（秒）	-	NUMBER
TOTAL_NUMBER - このスケジュールで伝播された総メッセージ数	-	NUMBER
TOTAL_BYTES - このスケジュールで伝播された総バイト数	-	NUMBER
MAX_NUMBER - 伝播枠で伝播される最大メッセージ数	-	NUMBER
MAX_BYTES - 伝播枠で伝播された最大バイト数	-	NUMBER
AVG_NUMBER - 伝播枠で伝播された平均メッセージ数	-	NUMBER
AVG_SIZE - 伝播されたメッセージの平均サイズ（バイト）	-	NUMBER
AVG_TIME - 1 つのメッセージ伝播にかかる平均時間（秒）	-	NUMBER
FAILURES - 実行が失敗した回数。16 になった場合、そのスケジュールは使用不可能になります。	-	NUMBER
LAST_ERROR_DATE - 最後に実行に失敗した日付	-	DATE
LAST_ERROR_TIME - 最後に実行に失敗した時刻	-	VARCHAR2(8)
LAST_ERROR_MSG - 最後に実行に失敗したときのエラー番号およびエラー・メッセージ	-	VARCHAR2(4000)

ユーザーがなんらかの権限を持っているキューの選択

図 10-5 ユーザーがなんらかの権限を持っているキューの選択



参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

ALL_QUEUES

用途

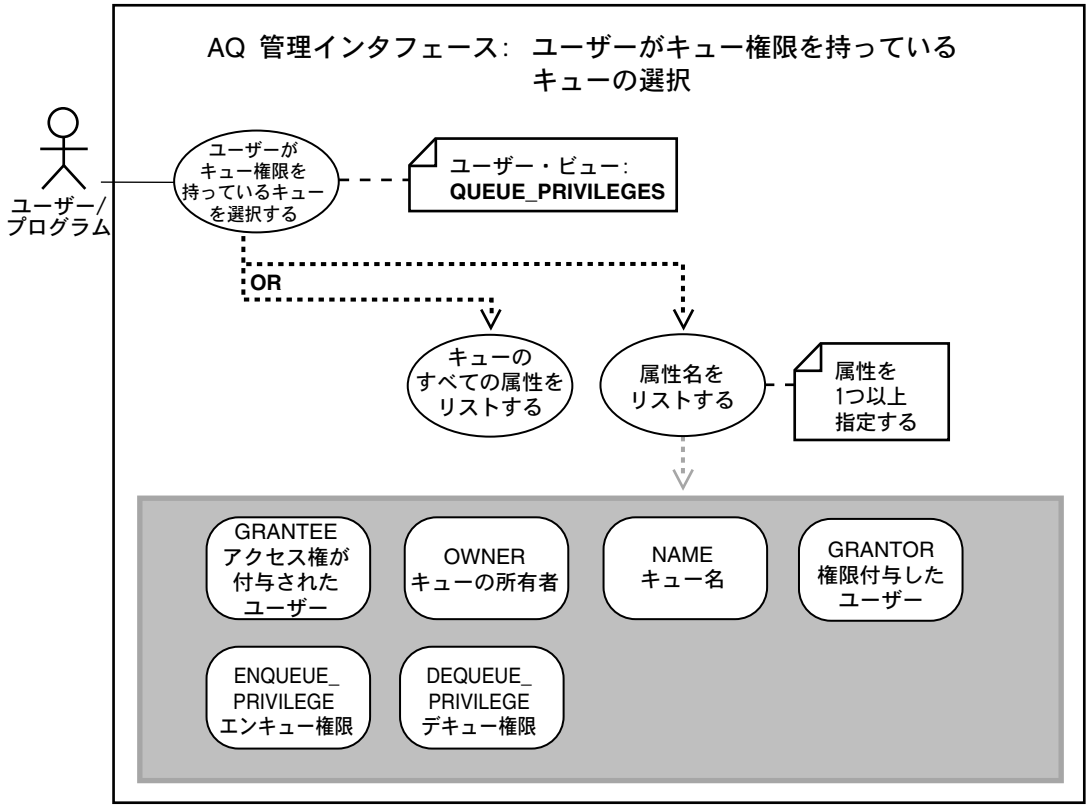
このビューは、ユーザーがアクセスできるすべてのキューを示します。

表 10-6 ALL_QUEUES

列名および説明	NULL かどうか	型
OWNER - キューの所有者	NOT NULL	VARCHAR2(30)
NAME - キュー名	NOT NULL	VARCHAR2(30)
QUEUE_TABLE - そのキュー・データが格納されているキュー表の名前	NOT NULL	VARCHAR2(30)
QID - そのキューのオブジェクト番号	NOT NULL	NUMBER
QUEUE_TYPE - キューの型	-	VARCHAR2(15)
MAX_RETRIES - そのキューからデキューするときの許容再試行回数	-	NUMBER
RETRY_DELAY - 再試行の間隔	-	NUMBER
ENQUEUE_ENABLED - キューにエンキュー可能かどうか	-	VARCHAR2(7)
DEQUEUE_ENABLED - キューからデキュー可能かどうか	-	VARCHAR2(7)
RETENTION - 処理済メッセージがキュー内に保存される時間	-	VARCHAR2(40)
USER_COMMENT - ユーザー指定のコメント	-	VARCHAR2(50)

ユーザーがキュー権限を持っているキューの選択

図 10-6 ユーザーがキュー権限を持っているキューの選択



参照: 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

QUEUE_PRIVILEGES

用途

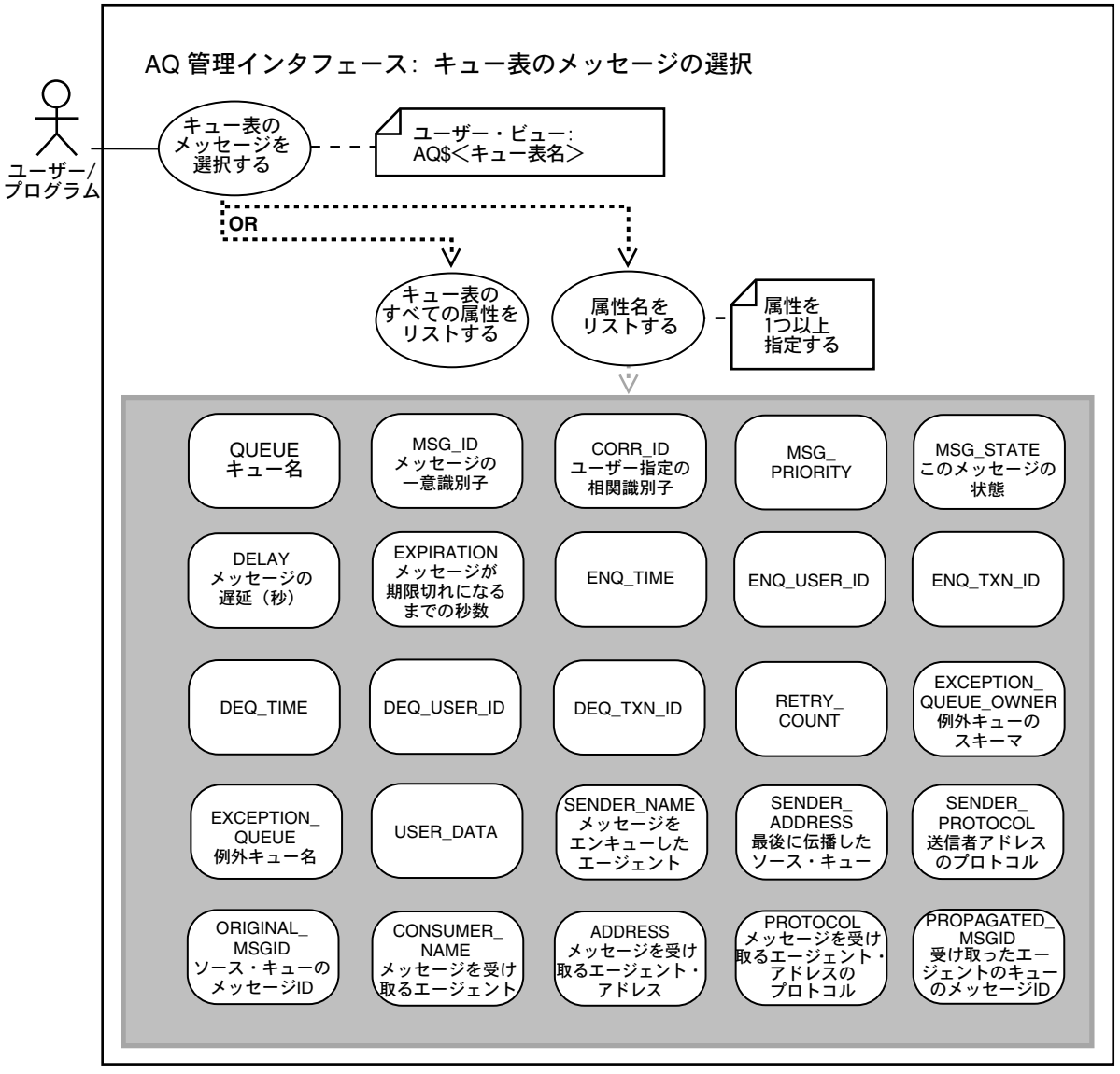
このビューは、ユーザーが権限付与者、権限受領者、所有者または使用可能なロールになっているキュー、あるいは PUBLIC に付与されているキューを示します。

表 10-7 QUEUE_PRIVILEGES

列名および説明	NULL かどうか	型
GRANTEE - アクセス権限が付与されたユーザー	NOT NULL	VARCHAR2(30)
OWNER - キューの所有者	NOT NULL	VARCHAR2(30)
NAME - キュー名	NOT NULL	VARCHAR2(30)
GRANTOR - 権限付与を実行したユーザー	NOT NULL	VARCHAR2(30)
ENQUEUE_PRIVILEGE - そのキューに対するエンキュー権限	-	NUMBER（付与されているときは 1、付与されていないときは 0（ゼロ））
DEQUEUE_PRIVILEGE - そのキューに対するデキュー権限	-	NUMBER（付与されているときは 1、付与されていないときは 0（ゼロ））

キュー表のメッセージの選択

図 10-7 キュー表のメッセージの選択



参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

AQ$< キュー表名 >

用途

このビューは、メッセージ・データが格納されているキュー表を示します。このビューはキュー表ごとに自動的に作成され、キュー・データの問合せに使用されます。デキュー履歴データ（時間、ユーザー ID およびトランザクション ID）は、シングル・コンシューマ・キューについてのみ有効です。

表 10-8 AQ$< キュー表名 >

列名および説明	NULL かどうか	型
QUEUE - キュー名	-	VARCHAR2(30)
MSG_ID - メッセージの一意識別子	-	RAW(16)
CORR_ID - ユーザー指定の相関識別子	-	VARCHAR2(128)
MSG_PRIORITY - メッセージの優先順位	-	NUMBER
MSG_STATE - メッセージの状態	-	VARCHAR2(9)
DELAY - メッセージが遅延処理されるまでの秒数	-	DATE
EXPIRATION - メッセージが READY 状態になってから、期限切れになるまでの秒数	-	NUMBER
ENQ_TIME - エンキュー時刻	-	DATE
ENQ_USER_ID - エンキューしたユーザーのユーザー ID	-	NUMBER
ENQ_TXN_ID - エンキューしたトランザクションのトランザクション ID	NOT NULL	VARCHAR2(30)
DEQ_TIME - デキュー時刻	-	DATE
DEQ_USER_ID - デキューしたユーザーのユーザー ID	-	NUMBER
DEQ_TXN_ID - デキューしたトランザクションのトランザクション ID	-	VARCHAR2(30)
RETRY_COUNT - 再試行回数	-	NUMBER
EXCEPTION_QUEUE_OWNER - 例外キューのスキーマ	-	VARCHAR2(30)

表 10-8 AQ$< キュー表名 >（続き）

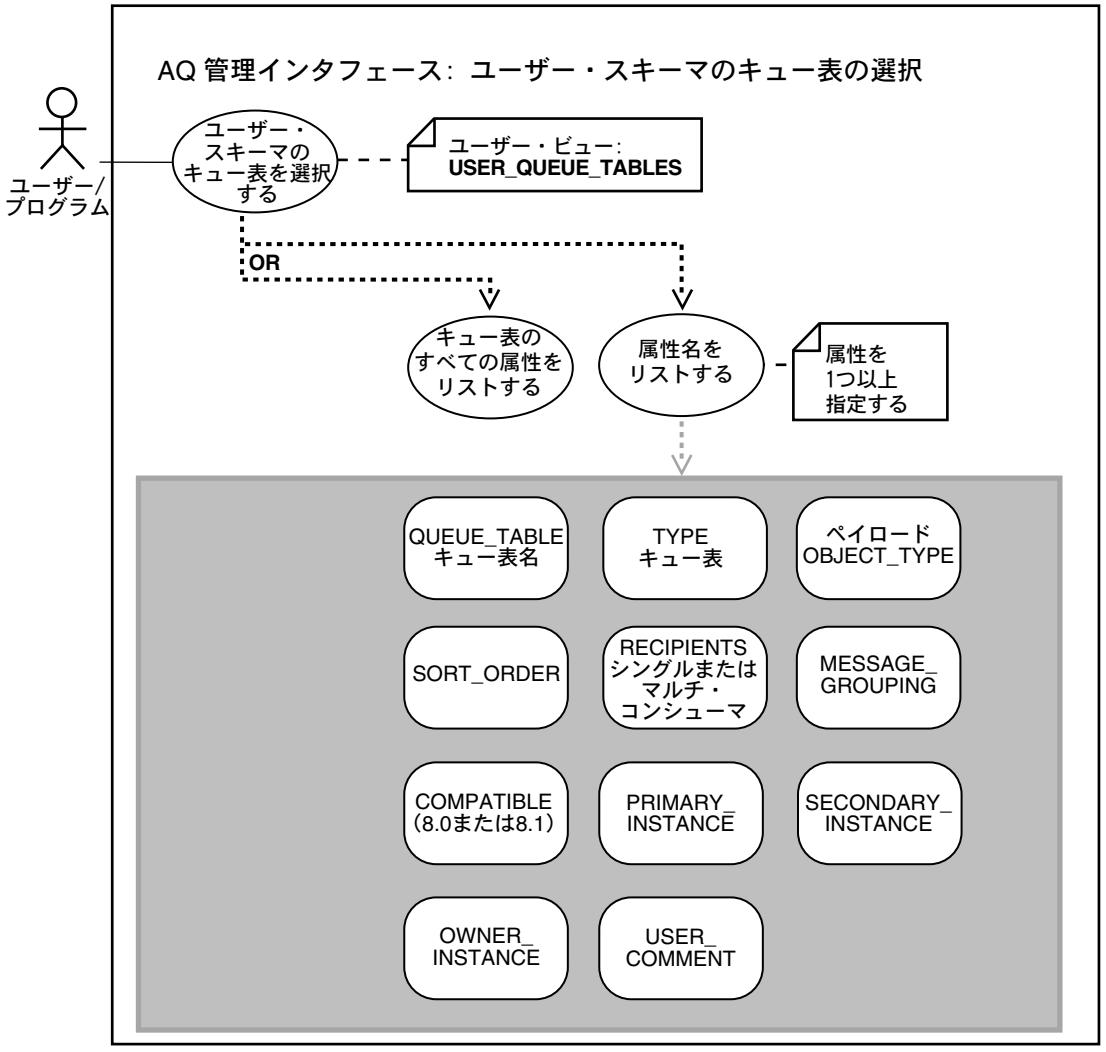
列名および説明	NULL かどうか	型
EXCEPTION_QUEUE - 例外キュー名	-	VARCHAR2(30)
USER_DATA - ユーザー・データ	-	BLOB
SENDER_NAME - メッセージをエンキューしたエージェント名（8.1 互換のキュー表でのみ有効）	-	VARCHAR2(30)
SENDER_ADDRESS - 最後に伝播を行ったソース・キュー名およびデータベース名。ソース・キューがローカル・データベースにあるときは、データベース名は指定されません（8.1 互換のキュー表でのみ有効）。	-	VARCHAR2(1024)
SENDER_PROTOCOL - 送信者アドレスのプロトコル。将来の使用のために確保されています（8.1 互換のキュー表でのみ有効）。	-	NUMBER
ORIGINAL_MSGID - ソース・キューにおけるメッセージ ID（8.1 互換のキュー表でのみ有効）	-	RAW(16)
CONSUMER_NAME - メッセージを受け取るエージェント名（8.1 互換のマルチ・コンシューマ・キュー表でのみ有効）	-	VARCHAR2(30)
ADDRESS - メッセージを受け取るエージェントのアドレス（キュー名およびデータベース・リンク名）。ソース・キューがローカルデータベースにあるときは、データベース・リンク名は指定されません。そのキューのローカル・エージェントがメッセージを受け取るとき、アドレスは NULL になります（8.1 互換のマルチ・コンシューマ・キュー表でのみ有効）。	-	VARCHAR2(1024)
PROTOCOL - メッセージを受け取るエージェント・アドレスのプロトコル（8.1 互換のキュー表でのみ有効）	-	NUMBER
PROPAGATED_MSGID - 受け取ったエージェントのキューにおけるメッセージ ID（8.1 互換のキュー表でのみ有効）	NULL	RAW(16)
ORIGINAL_QUEUE_NAME - メッセージを送信する側のキューの名前	-	-
ORIGINAL_QUEUE_OWNER - メッセージを送信する側のキューの所有者	-	-

表 10-8 AQ$< キュー表名 >（続き）

列名および説明	NULL かどうか	型
EXPIRATION_REASON - メッセージが例外キューに格納された理由。可能性のある値は次のとおり。 ■ TIME_EXPIRATION（メッセージが指定された時間の後に期限切れになった） ■ MAX_RETRY_EXCEEDED（再試行回数が最大値を超えた） ■ PROPAGATION_FAILURE（メッセージが伝播の途中で配信不可能になった）	-	-

ユーザー・スキーマのキュー表の選択

図 10-8 ユーザー・スキーマのキュー表の選択



参照: 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

USER_QUEUE_TABLES

構文

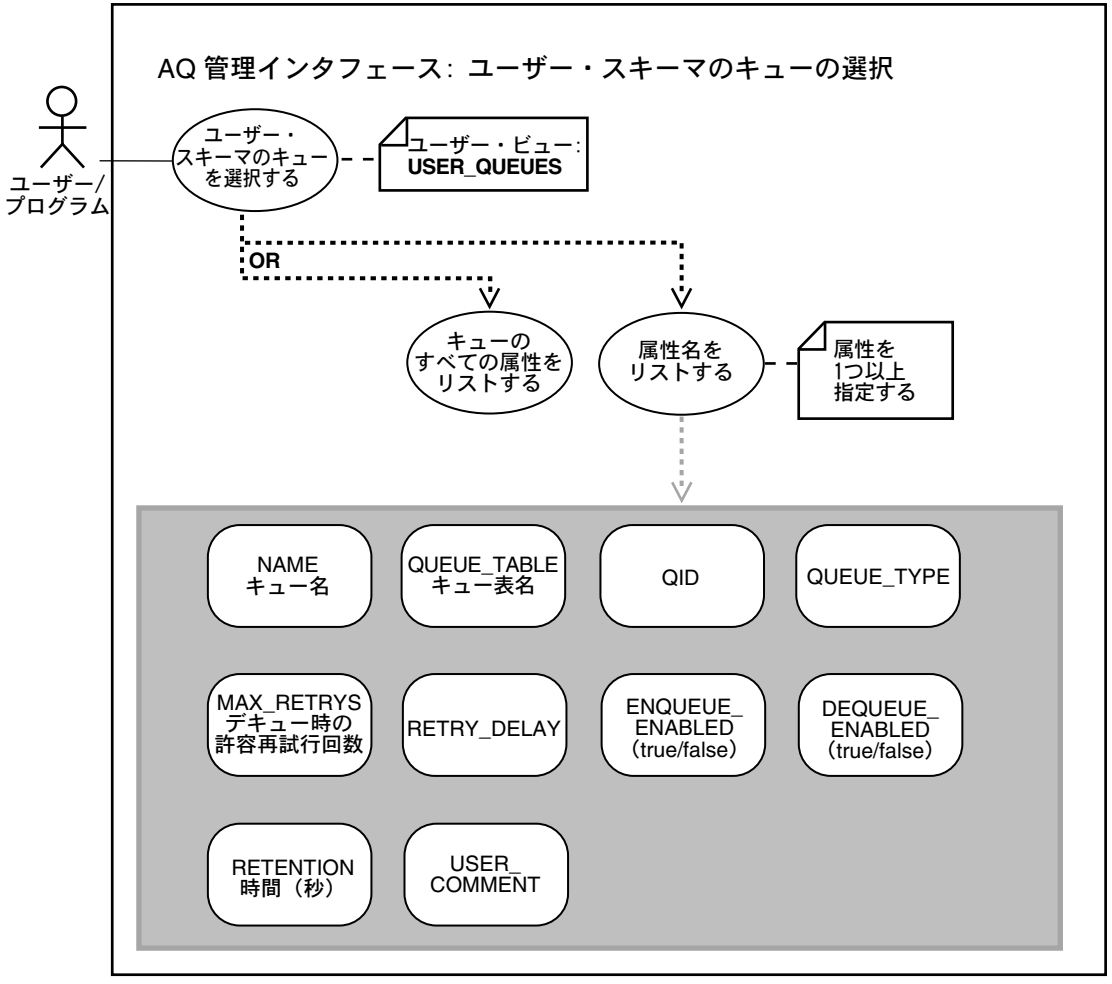
このビューは、ユーザー・スキーマ内のキュー表のみを表示する点を除いて、DBA_QUEUE_TABLES と同じです。このビューには、OWNER の列は含まれません。

表 10-9 USER_QUEUE_TABLES

列名および説明	NULL かどうか	型
QUEUE_TABLE - キュー表名	-	VARCHAR2(30)
TYPE - ペイロード型	-	VARCHAR2(7)
OBJECT_TYPE - オブジェクト型が定義されている場合、その型名	-	VARCHAR2(61)
SORT_ORDER - ユーザー指定のソート順	-	VARCHAR2(22)
RECIPIENTS - SINGLE または MULTIPLE	-	VARCHAR2(8)
MESSAGE_GROUPING - NONE または TRANSACTIONAL	-	VARCHAR2(13)
COMPATIBLE - そのキュー表が互換性を持つ最も低いバージョン	-	VARCHAR2(5)
PRIMARY_INSTANCE - そのキュー表の 1 次所有者であるインスタンス。値が 0（ゼロ）のときは、1 次所有者が指定されていません。	-	NUMBER
SECONDARY_INSTANCE - そのキュー表の 2 次所有者であるインスタンス。1 次所有者が存在しないときにキュー表の所有者になります。値が 0（ゼロ）のときは、2 次所有者が指定されていません。	-	NUMBER
OWNER_INSTANCE - 現在のキュー表の所有者インスタンス	-	NUMBER
USER_COMMENT - キュー表に対するユーザー・コメント	-	VARCHAR2(50)

ユーザー・スキーマのキューの選択

図 10-9 ユーザー・スキーマのキューの選択



参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

USER_QUEUES

用途

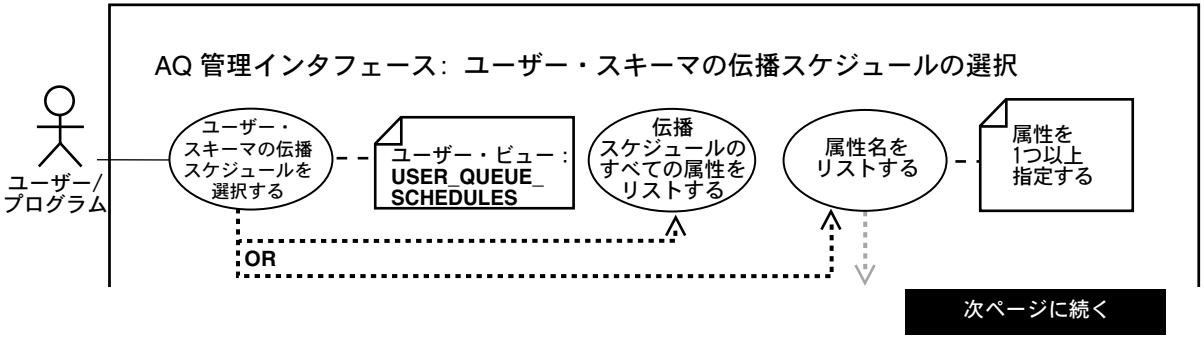
このビューは、ユーザー・スキーマ内のキューのみを表示する点を除いて、DBA_QUEUES と同じです。

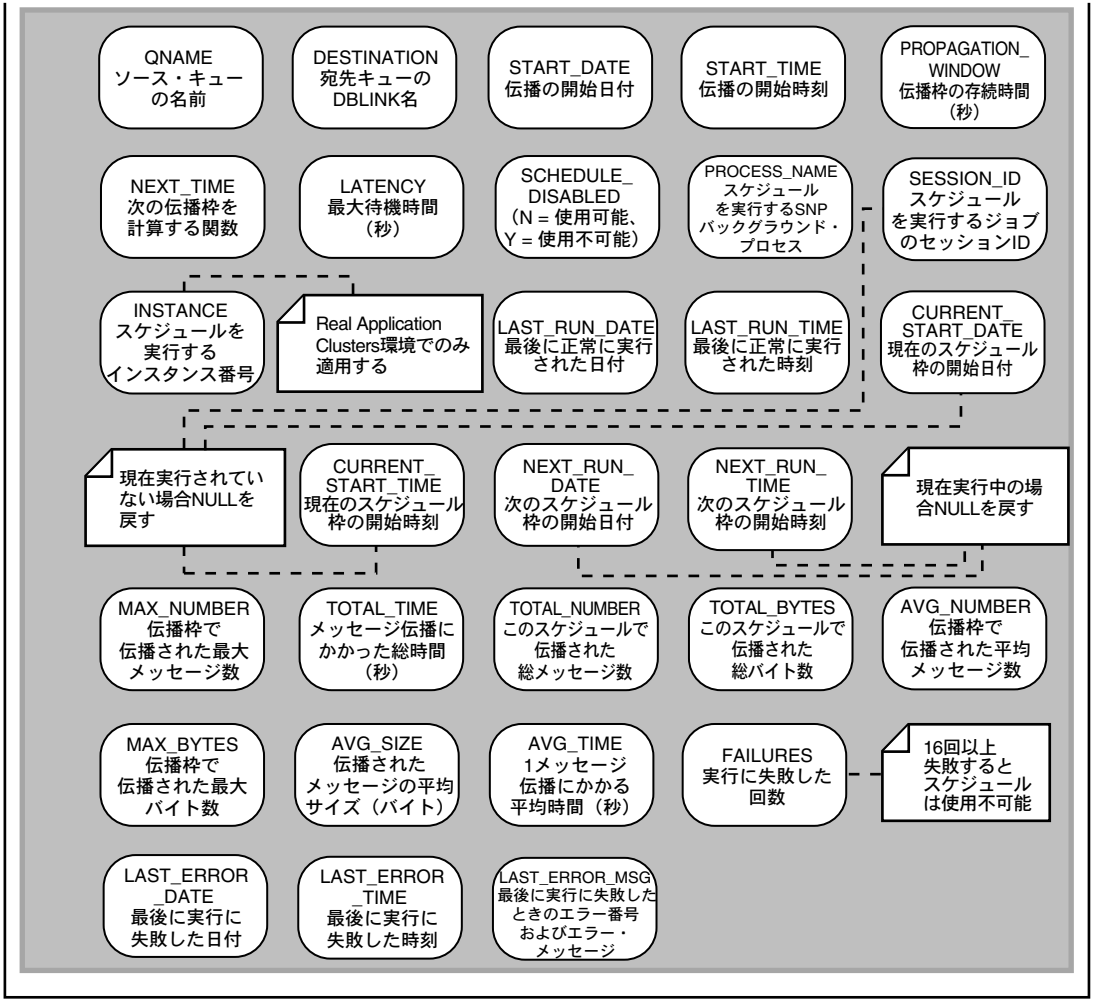
表 10-10 USER_QUEUES

列名および説明	NULL かどうか	型
NAME - キュー名	NOT NULL	VARCHAR2(30)
QUEUE_TABLE - このキューが格納されるキュー表名	NOT NULL	VARCHAR2(30)
QID - 一意のキュー識別子	NOT NULL	NUMBER
QUEUE_TYPE - キューの型	-	VARCHAR2(15)
MAX_RETRIES - デキューするときの許容試行回数	-	NUMBER
RETRY_DELAY - 再試行までの秒数	-	NUMBER
ENQUEUE_ENABLED - YES/NO	-	VARCHAR2(7)
DEQUEUE_ENABLED - YES/NO	-	VARCHAR2(7)
RETENTION - メッセージがデキュー後に保存される秒数	-	VARCHAR2(40)
USER_COMMENT - キューに対するユーザー・コメント	-	VARCHAR2(50)

ユーザー・スキーマの伝播スケジュールの選択

図 10-10 ユーザー・スキーマの伝播スケジュールの選択





参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

USER_QUEUE_SCHEDULES

用途

このビューは、現在のメッセージ伝播スケジュールを示します。

表 10-11 USER_QUEUE_SCHEDULES

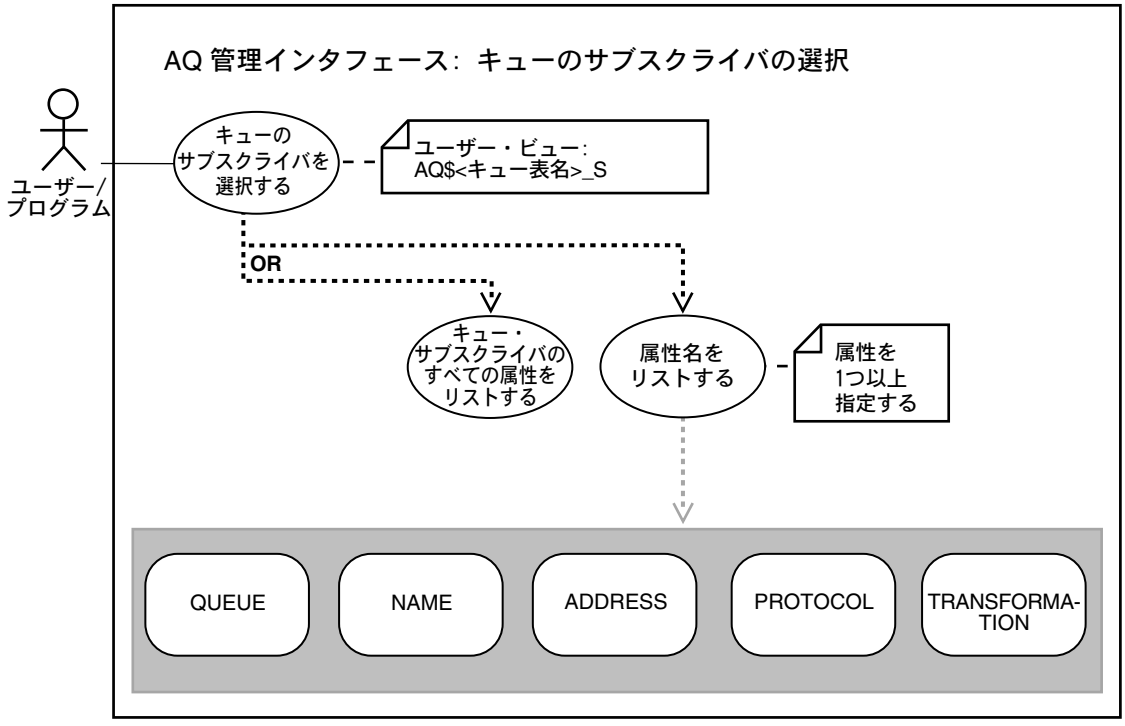
列名および説明	NULL かどうか	型
QNAME - ソース・キューの名前	NOT NULL	VARCHAR2(30)
DESTINATION - 宛先名。現在は DBLINK 名に制限されています。	NOT NULL	VARCHAR2(128)
START_DATE - デフォルトの日付書式による伝播の開始日付	-	DATE
START_TIME - HH:MI:SS フォーマットによる伝播の開始時刻	-	VARCHAR2(8)
PROPAGATION_WINDOW - 伝播枠の存続期間（秒）	-	NUMBER
NEXT_TIME - 次の伝播枠の開始を計算する関数	-	VARCHAR2(200)
LATENCY - 伝播枠内で、メッセージを伝播するまでの最大待機時間	-	NUMBER
SCHEDULE_DISABLED - 使用可能のときは N、使用不可能でスケジュールが実行されていないときは Y	-	VARCHAR(1)
PROCESS_NAME - このスケジュールを実行する SNP バックグラウンド・プロセス。実行されていないときは NULL になります。	-	VARCHAR2(8)
SESSION_ID - このスケジュールを実行しているジョブのセッション ID(SID,SERIAL#)。実行されていないときは NULL になります。	-	VARCHAR2(82)
INSTANCE - このスケジュールを実行している Real Application Clusters インスタンス番号	-	NUMBER
LAST_RUN_DATE - 最後に正常に実行された日付	-	DATE
LAST_RUN_TIME - 最後に正常に実行された時刻（HH:MI:SS フォーマット）	-	VARCHAR2(8)
CURRENT_START_DATE - このスケジュール枠が開始された日付	-	DATE
CURRENT_START_TIME - このスケジュール枠が開始された時刻（HH:MI:SS フォーマット）	-	VARCHAR2(8)
NEXT_RUN_DATE - 次のスケジュール枠が開始される日付	-	DATE
NEXT_RUN_TIME - 次のスケジュール枠が開始される時刻（HH:MI:SS フォーマット）	-	VARCHAR2(8)

表 10-11 USER_QUEUE_SCHEDULES（続き）

列名および説明	NULL かどうか	型
TOTAL_TIME - このスケジュールでメッセージの伝播に費やされた総時間（秒）	-	NUMBER
TOTAL_NUMBER - このスケジュールで伝播された総メッセージ数	-	NUMBER
TOTAL_BYTES - このスケジュールで伝播された総バイト数	-	NUMBER
MAX_NUMBER - 伝播枠で伝播される最大メッセージ数	-	NUMBER
MAX_BYTES - 伝播枠で伝播された最大バイト数	-	NUMBER
AVG_NUMBER - 伝播枠で伝播された平均メッセージ数	-	NUMBER
AVG_SIZE - 伝播されたメッセージの平均サイズ（バイト）	-	NUMBER
AVG_TIME - 1 つのメッセージ伝播にかかる平均時間（秒）	-	NUMBER
FAILURES - 実行が失敗した回数。16 になった場合、そのスケジュールは使用不可能になります。	-	NUMBER
LAST_ERROR_DATE - 最後に実行に失敗した日付	-	DATE
LAST_ERROR_TIME - 最後に実行に失敗した時刻	-	VARCHAR2(8)
LAST_ERROR_MSG - 最後に実行に失敗したときのエラー番号およびエラー・メッセージ	-	VARCHAR2(4000)

キューのサブスクライバの選択

図 10-11 キューのサブスクライバの選択



参照: 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

AQ$< キュー表名 >_S

用途

このビューは、指定されたキュー表の、すべてのキューに対するすべてのサブスクライバを示します。このビューは、キュー表が作成されたときに AQ$< キュー表名 >_S という名前で生成されます。キュー表内の一部またはすべてのキューに対するサブスクライバを問い合せるために使用します。このビューは、8.1 互換のキュー表でのみ作成されることに注意してください。また、サブスクライバの変換が作成された場合は、その変換も示します。

表 10-12 AQ$< キュー表名 >_S

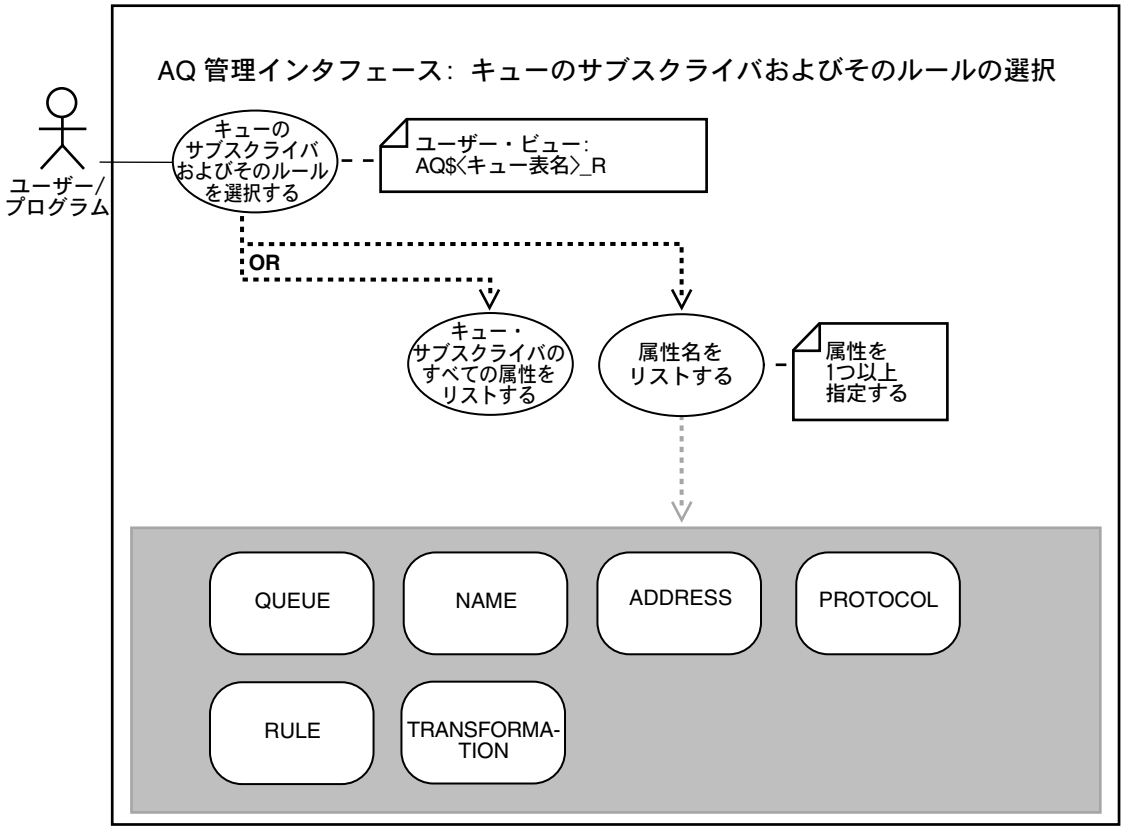
列名および説明	NULL かどうか	型
QUEUE - サブスクライバが定義されたキューの名前	NOT NULL	VARCHAR2(30)
NAME - エージェント名	-	VARCHAR2(30)
ADDRESS - エージェントのアドレス	-	VARCHAR2(1024)
PROTOCOL - エージェントのプロトコル	-	NUMBER
TRANSFORMATION-NULL となる可能性のある変換の名前	-	VARCHAR2(61)

使用上の注意

このビューは、8.1 互換のキュー表に作成されたキューに対して、dbms_aqadm.queue_subscribers() プロシージャと等価の機能を提供します。これらのキューのサブスクライバを参照する場合、プロシージャではなくビューを使用することをお薦めします。

キューのサブスクライバおよびそのルールの選択

図 10-12 キューのサブスクライバおよびそのルールの選択



参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

AQ$< キュー表名 >_R

用途

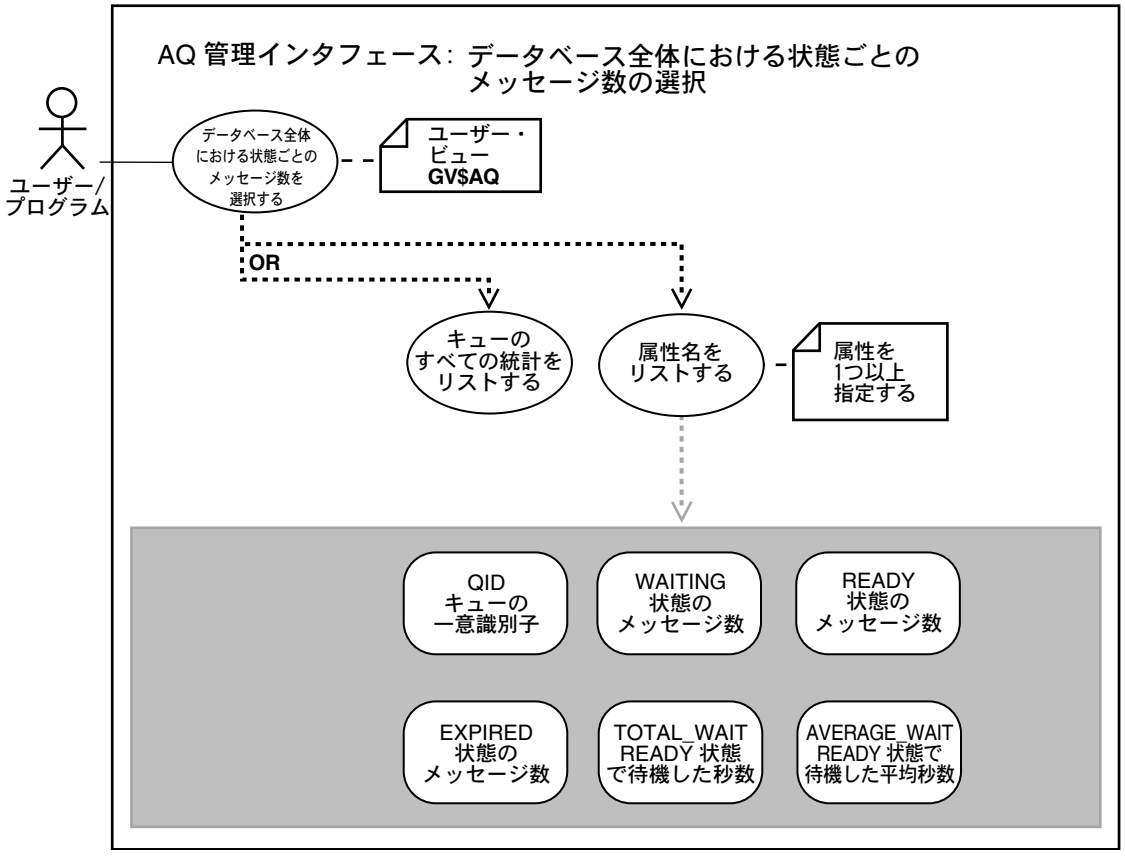
このビューは、指定されたキュー表のすべてのキューに対するルールベースのサブスクライバのみ（各サブスクライバによって定義されたルールのテキストを含む）を示します。このビューでは、指定されたキュー表のいずれかのキューに対して、ルールを定義されたサブスクライバを示します。このビューは、キュー表が作成されたときに AQ$< キュー表名 >_R という名前で生成されます。キュー表内の一部またはすべてのキューに対するサブスクライバを問い合せるために使用します。このビューは、8.1 互換のキュー表でのみ作成されることに注意してください。また、サブスクライバの変換が作成された場合は、その変換も示します。

表 10-13 AQ$< キュー表名 >_R

列名および説明	NULL かどうか	型
QUEUE - サブスクライバが定義されたキューの名前	NOT NULL	VARCHAR2(30)
NAME - エージェント名	-	VARCHAR2(30)
ADDRESS - エージェントのアドレス	-	VARCHAR2(1024)
PROTOCOL - エージェントのプロトコル	-	NUMBER
RULE - 定義されたルールのテキスト	-	VARCHAR2(30)
TRANSFORMATION - 指定された変換の名前（NULL となる可能性がある）	-	VARCHAR2(61)

データベース全体における状態ごとのメッセージ数の選択

図 10-13 データベース全体における状態ごとのメッセージ数の選択



参照: 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

GV$AQ

用途

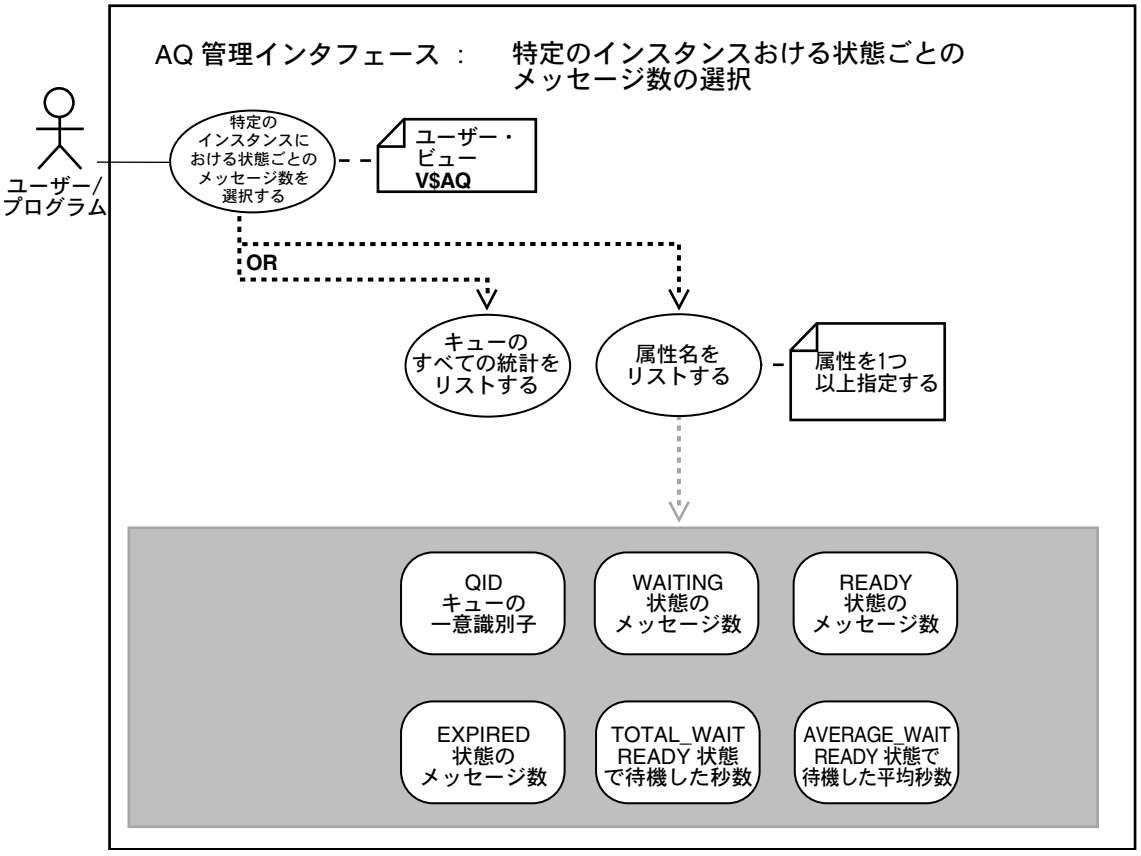
このビューは、データベース全体における状態ごとのメッセージ数の情報を示します。

表 10-14 GV$AQ

列名および説明	NULL かどうか	型
QID - キューの ID。user_queues および dba_queues の qid と同じです。	-	NUMBER
WAITING - WAITING 状態にあるメッセージ数	-	NUMBER
READY - READY 状態にあるメッセージ数	-	NUMBER
EXPIRED - EXPIRED 状態にあるメッセージ数	-	NUMBER
TOTAL_WAIT - READY 状態でメッセージがキューに待機している秒数	-	NUMBER
AVERAGE_WAIT - READY 状態で、メッセージがデキューを待機している平均秒数	-	NUMBER

特定のインスタンスにおける状態ごとのメッセージ数の選択

図 10-14 特定のインスタンスにおける状態ごとのメッセージ数の選択



参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

V$AQ

用途

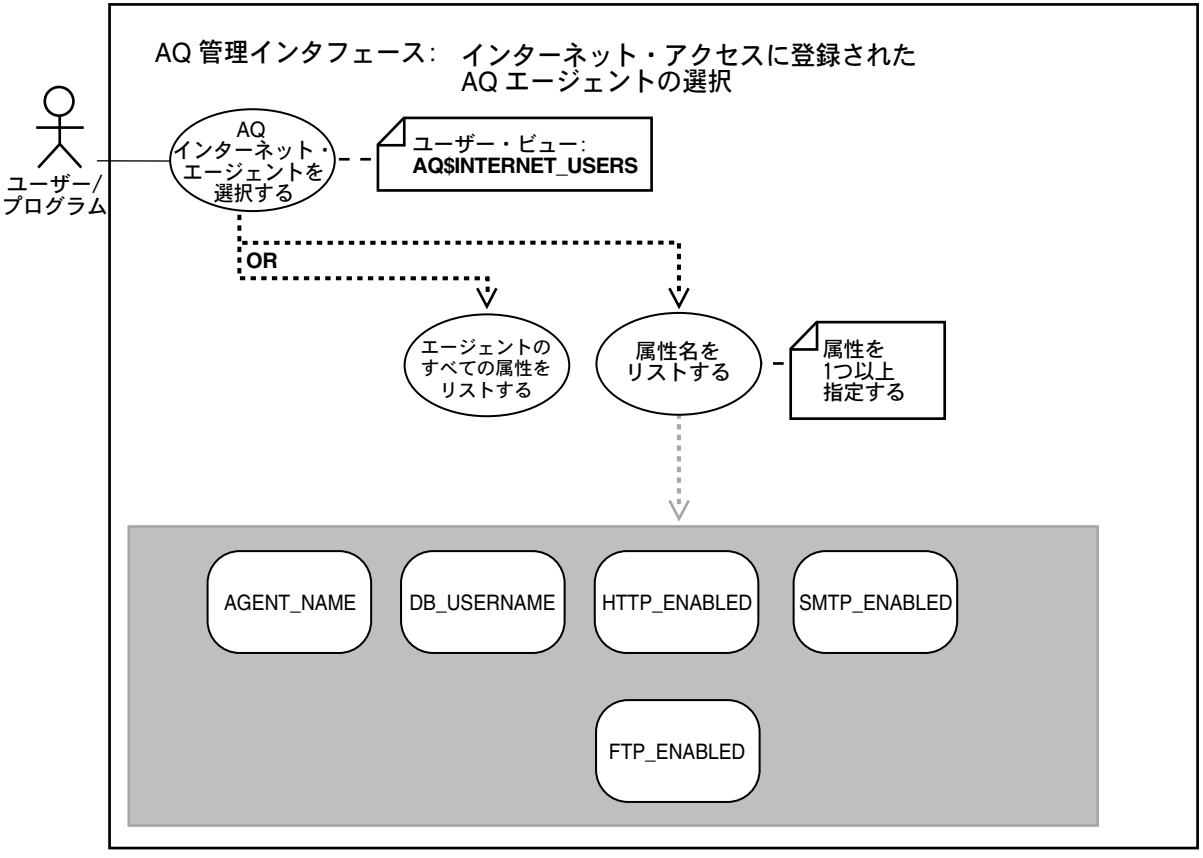
このビューは、特定のインスタンスにおける様々な状態にあるメッセージ数の情報を示します。

表 10-15 V$AQ

列名および説明	NULL かどうか	型
QID - キューの ID。user_queues および dba_queues の qid と同じです。	-	NUMBER
WAITING - WAITING 状態にあるメッセージ数	-	NUMBER
READY - READY 状態にあるメッセージ数	-	NUMBER
EXPIRED - EXPIRED 状態にあるメッセージ数	-	NUMBER
TOTAL_WAIT - READY 状態でメッセージがキューに待機している秒数	-	NUMBER
AVERAGE_WAIT - READY 状態でメッセージがデキューを待機している平均秒数	-	NUMBER

インターネット・アクセスに登録された AQ エージェントの選択

図 10-15 インターネット・アクセスに登録された AQ エージェントの選択



参照: 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

AQ$INTERNET_USERS

用途

AQ へのインターネット・アクセス用に登録されたエージェントの情報を示します。また、各インターネット・エージェントがマップするデータベース・ユーザーのリストも示します。

表 10-16 AQ$INTERNET_USERS

列名および説明	NULL かどうか	型
AGENT_NAME - AQ インターネット・エージェントの名前	NOT NULL	VARCHAR2(30)
DB_USERNAME - インターネット・エージェントがマップするデータベース・ユーザーの名前	NOT NULL	VARCHAR2(30)
HTTP_ENABLED - エージェントが HTTP を介して AQ へアクセス可能かどうか。値には YES または NO があります。	-	VARCHAR2(4)
FTP_ENABLED - エージェントが FTP を介して AQ へアクセス可能かどうか。今回のリリースでは、値は常に NO となります。	-	VARCHAR2(4)

ユーザー変換の選択

参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

USER_TRANSFORMATIONS

用途

このビューは、ユーザーが所有するすべての変換を示します。変換の定義を参照するときは、USER_ATTRIBUTE_TRANSFORMATIONS を問い合せます。

表 10-17 USER_TRANSFORMATIONS

列名および説明	NULL かどうか	型
TRANSFORMATION_ID - 変換の一意識別子	-	NUMBER
NAME - 変換名	-	VARCHAR2(30)
FROM_TYPE - ソース・タイプ名	-	VARCHAR2(61)
TO_TYPE - ターゲット・タイプ名	-	VARCHAR2(61)

ユーザー変換ファンクションの選択

参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

USER_ATTRIBUTE_TRANSFORMATIONS

用途

このビューは、ユーザーのすべての変換に対する変換ファンクションを示します。

表 10-18 USER_ATTRIBUTE_TRANSFORMATIONS

列名および説明	NULL かどうか	型
TRANSFORMATION_ID - 変換の一意識別子	-	NUMBER
NAME - 変換名	-	VARCHAR2(30)
FROM_TYPE - ソース・タイプ名	-	VARCHAR2(61)
TO_TYPE - ターゲット・タイプ名	-	VARCHAR2(61)
ATTRIBUTE - ターゲット・タイプの属性番号	-	NUMBER
ATRIBUTE_TRANSFORMATION - 属性の変換ファンクション	-	VARCHAR2(4000)

すべての変換の選択

参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

DBA_TRANSFORMATIONS

用途

このビューは、データベース内のすべての変換を示します。これらの変換は、エンキュー、デキュー、サブスクライブなどのアドバンスト・キューイング操作で指定でき、AQ メッセージ機能における変換を自動的に統合します。このビューは、DBA 権限を持つユーザーのみがアクセスできます。

表 10-19 DBA_TRANSFORMATIONS

列名および説明	NULL かどうか	型
TRANSFORMATION_ID - 変換の一意識別子	-	NUMBER
OWNER - 変換を所有するユーザー	-	VARCHAR2(30)
NAME - 変換名	-	VARCHAR2(30)
FROM_TYPE - ソース・タイプ名	-	VARCHAR2(61)
TO_TYPE - ターゲット・タイプ名	-	VARCHAR2(61)
Namespace - Oracle 変換エンジンが作成した変換用の名前空間。サード・パーティの変換エンジンによる変換では、名前空間が異なります。	-	-
From_type_schema - ソース・タイプの所有ユーザー	-	-
From_type_name - 変換のソース・タイプ	-	-
To_type_Schema - 宛先タイプの所有ユーザー	-	-
To_type_name - 変換の宛先タイプ。変換は、ソース・タイプのオブジェクトを取得し、宛先タイプのオブジェクトを返します。	-	-
Transformation_type - 変換タイプ。値は SQL および XSL です。	-	-
Attribute_Name - 変換に指定する宛先タイプの属性名	-	-

表 10-19 DBA_TRANSFORMATIONS（続き）

列名および説明	NULL かどうか	型
Transformation_Expression - SQL 式、PL/SQL ファンクションまたは XSL 文	-	-
Comment - ユーザー指定のコメント	-	-

すべての変換ファンクションの選択

参照： 管理インタフェースのビューのリストは、表 10-1 を参照してください。

ビュー名

DBA_ATTRIBUTE_TRANSFORMATIONS

用途

このビューは、データベース内のすべての変換に対する変換ファンクションを示します。

表 10-20 DBA_ATTRIBUTE_TRANSFORMATIONS

列名および説明	NULL かどうか	型
TRANSFORMATION_ID - 変換の一意識別子	-	NUMBER
OWNER - 変換の所有者	-	VARCHAR2(30)
NAME - 変換名	-	VARCHAR2(30)
FROM_TYPE - ソース・タイプ名	-	VARCHAR2(61)
TO_TYPE - ターゲット・タイプ名	-	VARCHAR2(61)
ATTRIBUTE - ターゲット・タイプの属性番号	-	NUMBER
ATRIBUTE_TRANSFORMATION - 属性の変換ファンクション	-	VARCHAR2(4000)

11

操作インタフェース : 基本操作

この章では、Oracle Advanced Queuing の操作インタフェースを利用方法に沿って説明します。それぞれの操作（メッセージのエンキューなど）を、その操作名ごとに利用方法に沿って説明します。この章の先頭に、すべての利方法を示します（11-2 ページの「利用モデル : 操作インタフェース - 基本操作」を参照）。

図 11-1 に、すべての利用方法を 1 つの図にまとめています。HTML 版のマニュアルをご使用の場合、この図の中の利用方法のタイトルをクリックして、関心のある利用方法に移動できます。

個々の利用方法は、次の形式で説明されています。

- **利用図 :** 利用方法を表す図
- **用途 :** この利用方法の用途
- **使用上の注意 :** 実装に有効なガイドライン
- **構文 :** このアクティビティの実行に使用する主な構文
- **例 :** 各プログラム環境での利用例

利用モデル : 操作インタフェース - 基本操作

表 11-1「利用モデル : 操作インタフェース」の「+」は、その利用方法で、プログラム環境の例が記載されていることを示します。

この表では、プログラム環境を次の略称で表しています。

- **P** - DBMS_AQADM および DBMS_AQ パッケージを使用した PL/SQL
- **O** - Oracle Call Interface（OCI）を使用した C 言語
- **V** - Oracle Objects for OLE（OO4O）を使用した Visual Basic
- **J** - Java Databasae Connectivity（JDBC）を使用した Java（ネイティブ AQ）
- **JMS** - Java Databasae Connectivity（JDBC）を使用した Java（JMS 標準）

表 11-1 利用モデル : 操作インタフェース

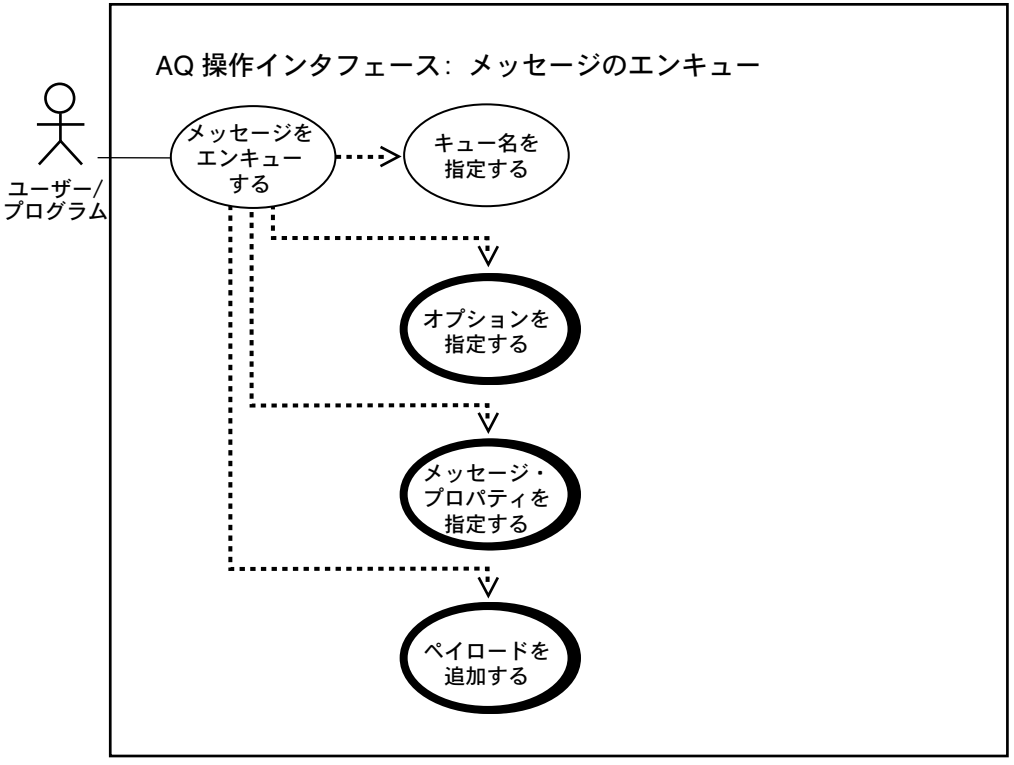
利用方法	P	O	V	J	JM
メッセージのエンキュー（11-4 ページ）	-	-	-	-	-
メッセージのエンキュー（オプションの指定）（11-7 ページ）	+	-	+	-	+
メッセージのエンキュー（メッセージ・プロパティの指定）（11-10 ページ）	+	-	+	-	+
メッセージのエンキュー（メッセージ・プロパティの指定（送信者 ID の指定））（11-13 ページ）	+	-	+	-	+
メッセージのエンキュー（ペイロードの追加）（11-15 ページ）	+	-	+	-	+
1 個以上のシングル・コンシューマ・キューのリスニング（11-24 ページ）	+	+	+	-	-
1 個以上のマルチ・コンシューマ・キューのリスニング（11-36 ページ）	+	+	+	-	-
メッセージのデキュー（11-44 ページ）	-	-	-	-	-
シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定）（11-47 ページ）	+	-	+	-	+
マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定）（11-52 ページ）	+	-	+	-	+
通知の登録（11-55 ページ）	-	-	-	-	-
通知の登録（サブスクリプション名の指定 - シングル・コンシューマ・キュー）（11-58 ページ）	-	+	-	-	-
通知の登録（サブスクリプション名の指定 - マルチ・コンシューマ・キュー）（11-59 ページ）	-	+	-	-	-

表 11-1 利用モデル : 操作インタフェース（続き）

利用方法	P	O	V	J	JM
サブスクライバの通知の転送（11-66 ページ）	+	+	-	-	-
エージェントの LDAP サーバーへの追加（11-69 ページ）	-	-	-	-	-
エージェントの LDAP サーバーからの削除（11-71 ページ）	-	-	-	-	-

メッセージのエンキュー

図 11-1 メッセージのエンキュー



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-7 ページの「メッセージのエンキュー（オプションの指定）」も参照してください。
- 11-10 ページの「メッセージのエンキュー（メッセージ・プロパティの指定）」も参照してください。
- 11-13 ページの「メッセージのエンキュー（メッセージ・プロパティの指定（送信者 ID の指定））」も参照してください。
- 11-15 ページの「メッセージのエンキュー（ペイロードの追加）」も参照してください。

用途

メッセージを、指定したキューに追加します。

使用上の注意

メッセージが受信者を指定しないでマルチ・コンシューマ・キューにエンキューされ、そのキューにサブスクライバが指定されていない（またはそのメッセージに一致するルールベースのサブスクライバが存在しない）場合、ORA-24033 エラーになります。これは、配信先になる受信者またはサブスクライバが存在しないメッセージは廃棄されるという警告です。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の `ENQUEUE` プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）：「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ」を選択してください。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQQueue」の `enqueue`

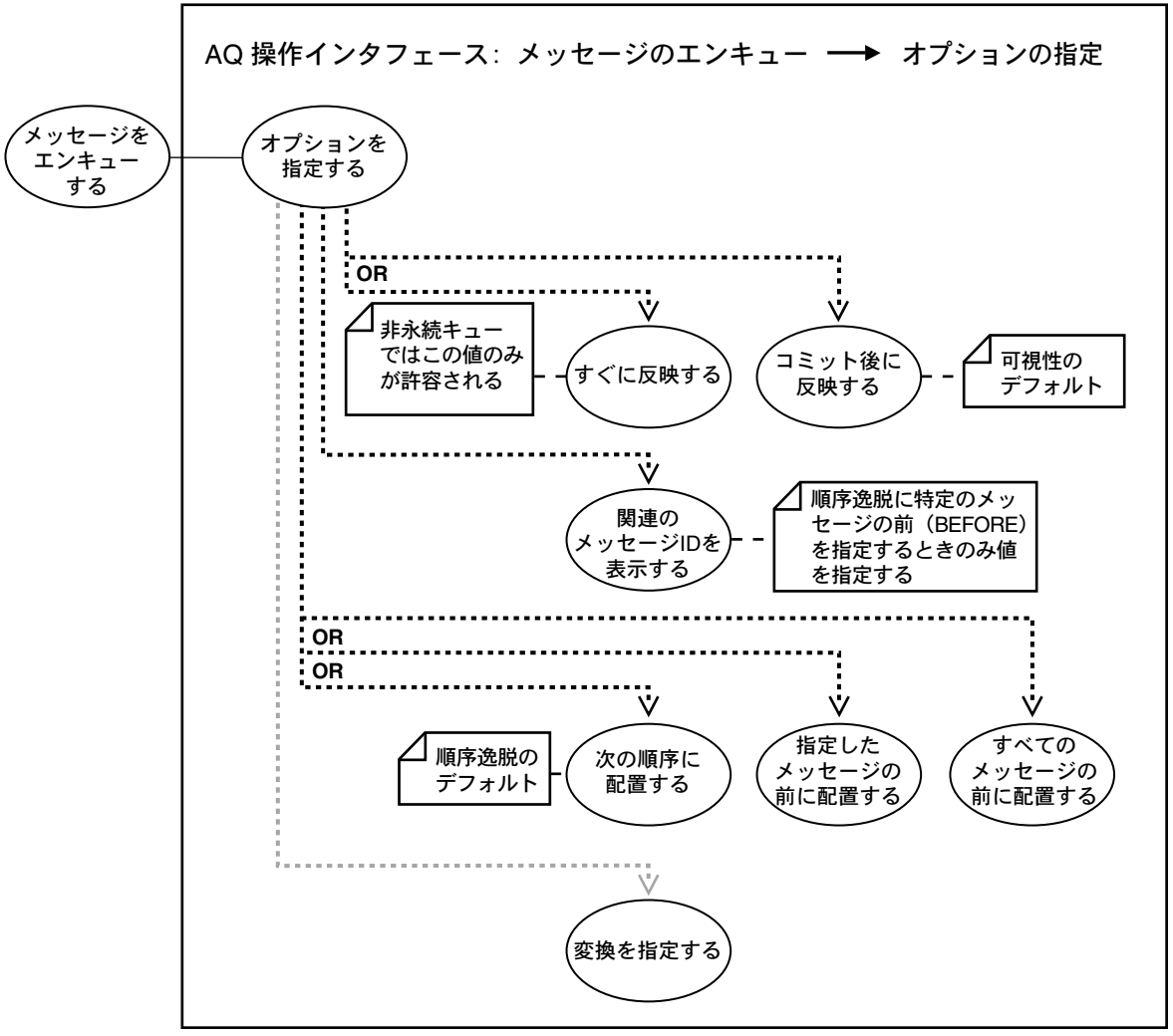
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）：オブジェクト型メッセージのエンキュー（11-17 ページ）
- Java（JDBC）：メッセージのエンキュー（ペイロードの追加）（11-19 ページ）
- Visual Basic（OO4O）：メッセージのエンキュー（11-22 ページ）

メッセージのエンキュー（オプションの指定）

図 11-2 メッセージのエンキュー（オプションの指定）



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-4 ページの「メッセージのエンキュー」も参照してください。
- 11-10 ページの「メッセージのエンキュー（メッセージ・プロパティの指定）」も参照してください。
- 11-13 ページの「メッセージのエンキュー（メッセージ・プロパティの指定（送信者 ID の指定））」も参照してください。
- 11-15 ページの「メッセージのエンキュー（ペイロードの追加）」も参照してください。

用途

エンキュー操作で使用できるオプションを指定します。

使用上の注意

LOB ロケータは、トランザクションの存続期間中にのみ有効であるため、LOB ロケータを使用するときは、IMMEDIATE オプションを使用しないでください。IMMEDIATE オプションを使用すると、トランザクションが自動的にコミットされるため、ロケータは有効になりません。

- エンキュー・オプションの sequence deviation パラメータを使用すると、2 つのメッセージ間の処理順序を変更できます。他のメッセージが存在する場合は、その識別情報をエンキュー・オプションのパラメータ関連 msgid で指定します。関連が、Sequence deviation パラメータによって識別されます。

  メッセージに Sequence deviation を指定すると、そのメッセージに指定できる遅延および優先順位の値が制限されます。このメッセージの遅延の値は、このメッセージより前にエンキューされるメッセージの遅延の値以下にする必要があります。このメッセージの優先順位の値は、このメッセージより前にエンキューされるメッセージの優先順位以上にする必要があります。
- 非永続キューでは、可視性オプションを IMMEDIATE にする必要があります。
- 非永続キューでは、ローカル受信者のみがサポートされています。
- 変換が指定されている場合は、メッセージがキューにエンキューされる前にそのメッセージに適用されます。変換では、キューのユーザー定義型と同じ型のオブジェクトにメッセージをマップする必要があります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の ENQUEUE プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）：「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ」を選択してください。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQEnqueueOption」

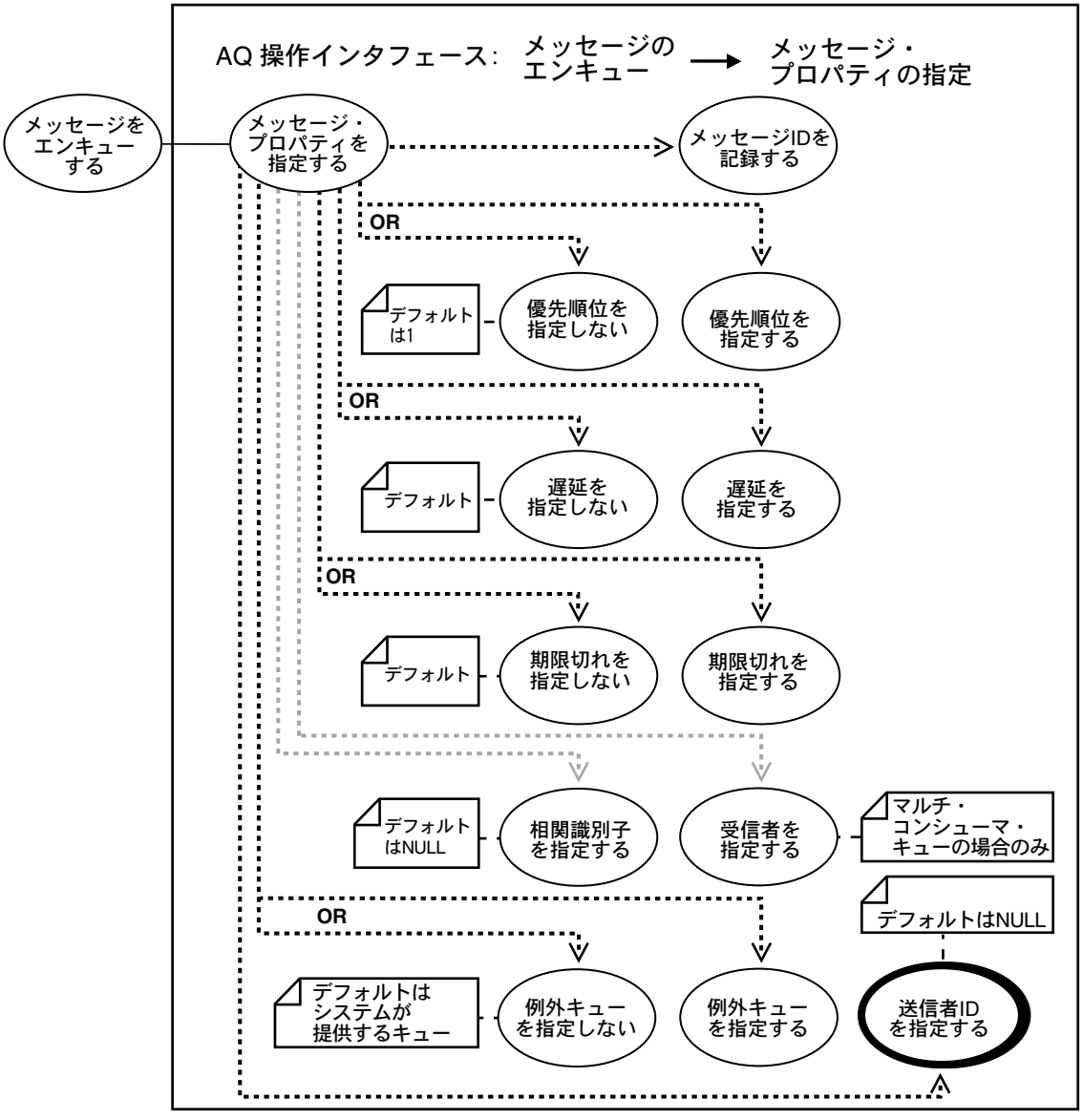
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）：オブジェクト型メッセージのエンキュー（11-17 ページ）
- Java（JDBC）：メッセージのエンキュー（ペイロードの追加）（11-19 ページ）
- Visual Basic（OO4O）：メッセージのエンキュー（11-22 ページ）

メッセージのエンキュー（メッセージ・プロパティの指定）

図 11-3 メッセージのエンキュー（メッセージ・プロパティの指定）



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-4 ページの「メッセージのエンキュー」も参照してください。
- 11-7 ページの「メッセージのエンキュー（オプションの指定）」も参照してください。
- 11-13 ページの「メッセージのエンキュー（メッセージ・プロパティの指定（送信者 ID の指定））」も参照してください。
- 11-15 ページの「メッセージのエンキュー（ペイロードの追加）」も参照してください。

用途

メッセージ・プロパティには、AQ によって個々のメッセージを管理するために使用される情報が記述されています。これはエンキュー時に設定され、デキュー時にその値が戻されます。

使用上の注意

- 待機中または処理済状態のメッセージを参照するには、メッセージ ID によってデキューまたはブラウズするか、または SELECT 文を使用します。
- メッセージの遅延および期限切れの処理は、キュー・モニター・バックグラウンド・プロセスによって行われます。AQ の遅延および期限切れの機能を使用するときは、データベースの QMN プロセスを開始する必要があります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の ENQUEUE プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）：「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ」を選択してください。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQMessageProperty」

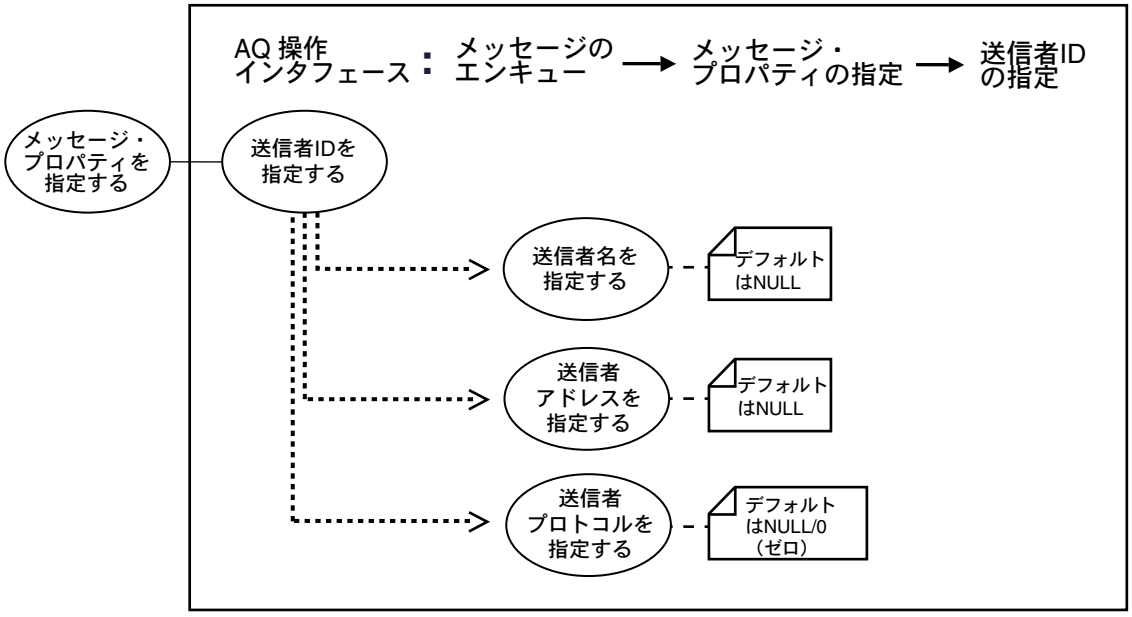
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）：オブジェクト型メッセージのエンキュー（11-17 ページ）
- Java（JDBC）：メッセージのエンキュー（ペイロードの追加）（11-19 ページ）
- Visual Basic（OO4O）：メッセージのエンキュー（11-22 ページ）

メッセージのエンキュー（メッセージ・プロパティの指定（送信者 ID の指定））

図 11-4 メッセージのエンキュー（メッセージ・プロパティの指定（送信者 ID の指定））



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-4 ページの「メッセージのエンキュー」も参照してください。
- 11-7 ページの「メッセージのエンキュー（オプションの指定）」も参照してください。
- 11-10 ページの「メッセージのエンキュー（メッセージ・プロパティの指定）」も参照してください。
- 11-15 ページの「メッセージのエンキュー（ペイロードの追加）」も参照してください。

用途

メッセージの送信者（プロデューサ）を識別します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の `ENQUEUE` プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）：「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ」を選択してください。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQMessageProperty」の `setSender`

> **参照：** エージェントの詳細は、2-3 ページの「エージェント型（aq$_agent）」を参照してください。

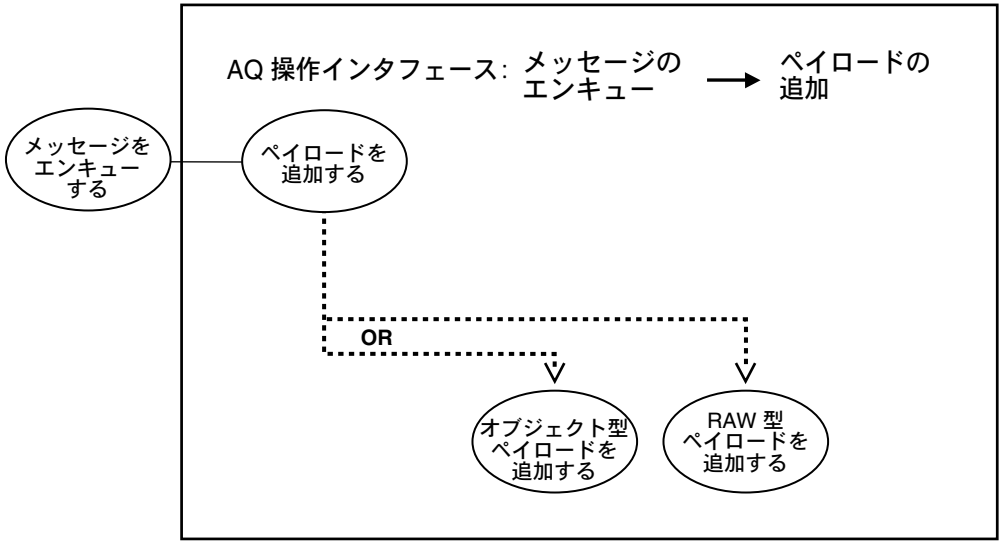
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）：オブジェクト型メッセージのエンキュー（11-17 ページ）
- Java（JDBC）：メッセージのエンキュー（ペイロードの追加）（11-19 ページ）
- Visual Basic（OO4O）：メッセージのエンキュー（11-22 ページ）

メッセージのエンキュー（ペイロードの追加）

図 11-5 メッセージのエンキュー（ペイロードの追加）



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-4 ページの「メッセージのエンキュー」も参照してください。
- 11-7 ページの「メッセージのエンキュー（オプションの指定）」も参照してください。
- 11-10 ページの「メッセージのエンキュー（メッセージ・プロパティの指定）」も参照してください。
- 11-13 ページの「メッセージのエンキュー（メッセージ・プロパティの指定（送信者 ID の指定））」も参照してください。

使用上の注意

RAW 型のペイロードを格納するために、AQ ではペイロード・リポジトリとして LOB 列を持つキュー表を作成します。ペイロードの最大サイズは、AQ にアクセスするために使用しているプログラム環境によって決定されます。PL/SQL、Java およびプリコンパイラの場合は 32KB、OCI の場合は 4GB に制限されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の `ENQUEUE` プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）：「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ」を選択してください。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQQueue」の `enqueue`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）：オブジェクト型メッセージのエンキュー（11-17 ページ）
- Java（JDBC）：メッセージのエンキュー（ペイロードの追加）（11-19 ページ）
- Visual Basic（OO4O）：メッセージのエンキュー（11-22 ページ）

PL/SQL（DBMS_AQ）：オブジェクト型メッセージのエンキュー

注意：次のようなデータ構造を設定しないと機能しない例もあります。

```
CONNECT system/manager
CREATE USER aq IDENTIFIED BY aq;
GRANT Aq_administrator_role TO aq;
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
   Queue_table            =>  'aq.objmsgs_qtab',
   Queue_payload_type     =>  'aq.message_typ');
EXECUTE DBMS_AQADM.CREATE_QUEUE (
   Queue_name            =>  'aq.msg_queue',
   Queue_table           =>  'aq.objmsgs_qtab');
EXECUTE DBMS_AQADM.START_QUEUE (
   Queue_name         => 'aq.msg_queue',
   Enqueue            => TRUE);
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
   Queue_table            => 'aq.prioritymsgs_qtab',
   Sort_list              => 'PRIORITY,ENQ_TIME',
   Queue_payload_type     => 'aq.message_typ');
EXECUTE DBMS_AQADM.CREATE_QUEUE (
   Queue_name             => 'aq.priority_msg_queue',
   Queue_table            => 'aq.prioritymsgs_qtab');
EXECUTE DBMS_AQADM.START_QUEUE (
   Queue_name             => 'aq.priority_msg_queue',
   Enqueue                => TRUE);
```

単一メッセージをエンキューし、キュー名およびペイロードを指定する

```
/* Enqueue to msg_queue: */
DECLARE
   Enqueue_options     DBMS_AQ.enqueue_options_t;
   Message_properties  DBMS_AQ.message_properties_t;
   Message_handle      RAW(16);
   Message             aq.message_typ;

BEGIN
   Message := aq.message_typ('NORMAL MESSAGE',
      'enqueued to msg_queue first.');

   DBMS_AQ.ENQUEUE(queue_name => 'msg_queue',
   Enqueue_options            => enqueue_options,
   Message_properties         => message_properties,
   Payload                    => message,
   Msgid                      => message_handle);
```

```
   COMMIT;
END;
```

単一メッセージをエンキューし、優先順位を指定する

```
/* The queue name priority_msg_queue is defined as an object type queue table.
   The payload object type is message. The schema of the queue is aq.  */

 /* Enqueue a message with priority 30: */
DECLARE
   Enqueue_options       dbms_aq.enqueue_options_t;
   Message_properties    dbms_aq.message_properties_t;
   Message_handle        RAW(16);
   Message               aq.Message_typ;

BEGIN
   Message := Message_typ('PRIORITY MESSAGE', 'enqued at priority 30.');

   message_properties.priority := 30;

   DBMS_AQ.ENQUEUE(queue_name => 'priority_msg_queue',
   enqueue_options            => enqueue_options,
   message_properties         => message_properties,
   payload                    => message,
   msgid                      => message_handle);

   COMMIT;
END;
```

単一メッセージをエンキューし、変換を指定する

```
/* Enqueue to msg_queue: */
DECLARE
   Enqueue_options     DBMS_AQ.enqueue_options_t;
   Message_properties  DBMS_AQ.message_properties_t;
   Message_handle      RAW(16);
   Message             aq.message_typ;

BEGIN
   Message := aq.message_typ('NORMAL MESSAGE',
      'enqueued to msg_queue first.');

   DBMS_AQ.ENQUEUE(queue_name => 'msg_queue',
   Enqueue_options            => enqueue_options,
   Message_properties         => message_properties,
   transformation             => 'AQ.MSG_MAP',
```

```
   Payload                    => message,
   Msgid                      => message_handle);

   COMMIT;
END;
```

ここで、MSG_MAP は次のように作成されます。

```
BEGIN
   DBMS.TRANSFORM.CREATE_TRANSFORMATION
   (
      schema => 'AQ',
      name =>   'MSG_MAP',
      from_schema => 'AQ',
      from_type => 'PO_ORDER1',
      to_schema => 'AQ',
      to_type => 'PO_ORDER2',
      transformation => 'AQ.MAP_PO_ORDER (source.user_data)'),
END;
```

Java（JDBC）：メッセージのエンキュー（ペイロードの追加）

```
/* Setup  */
connect system/manager
create user aq identified by aq;
grant aq_administrator_role to aq;

public static void setup(AQSession aq_sess) throws AQException
{
     AQQueueTableProperty    qtable_prop;
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;
     AQAgent                 agent;

     qtable_prop = new AQQueueTableProperty("RAW");

     q_table = aq_sess.createQueueTable ("aq", "rawmsgs_qtab", qtable_prop);

     queue_prop = new AQQueueProperty();
     queue = aq_sess.createQueue (q_table, "msg_queue", queue_prop);

     queue.start();

     qtable_prop = new AQQueueTableProperty("RAW");
     qtable_prop.setMultiConsumer(true);
```

```
     qtable_prop.setSortOrder("priority,enq_time");
     q_table = aq_sess.createQueueTable ("aq", "rawmsgs_qtab2",
    qtable_prop);

     queue_prop = new AQQueueProperty();
     queue = aq_sess.createQueue (q_table, "priority_msg_queue", queue_prop);

     queue.start();

     agent = new AQAgent("subscriber1", null);

     queue.addSubscriber(agent, null);
}


/* Enqueue a message */
public static void example(AQSession aq_sess) throws AQException, SQLException
{
     AQQueue                  queue;
     AQMessage                message;
     AQRawPayload             raw_payload;
     AQEnqueueOption          enq_option;
     String                   test_data = "new message";
     byte[]                   b_array;
     Connection               db_conn;

     db_conn = ((AQOracleSession)aq_sess).getDBConnection();

     /* Get a handle to the queue */
     queue = aq_sess.getQueue ("aq", "msg_queue");

     /* Create a message to contain raw payload: */
     message = queue.createMessage();

     /* Get handle to the AQRawPayload object and populate it with raw data: */
     b_array = test_data.getBytes();

     raw_payload = message.getRawPayload();

     raw_payload.setStream(b_array, b_array.length);

     /* Create a AQEnqueueOption object with default options: */
     enq_option = new AQEnqueueOption();

     /* Enqueue the message: */
     queue.enqueue(enq_option, message);
```

```
     db_conn.commit();
}


/* Enqueue a message with priority = 5 */
public static void example(AQSession aq_sess) throws AQException, SQLException
{
     AQQueue                  queue;
     AQMessage                message;
     AQMessageProperty        msg_prop;
     AQRawPayload             raw_payload;
     AQEnqueueOption          enq_option;
     String                   test_data = "priority message";
     byte[]                   b_array;
     Connection               db_conn;

     db_conn = ((AQOracleSession)aq_sess).getDBConnection();

     /* Get a handle to the queue */
     queue = aq_sess.getQueue ("aq", "msg_queue");

     /* Create a message to contain raw payload: */
     message = queue.createMessage();

     /* Get Message property */
     msg_prop = message.getMessageProperty();

     /* Set priority */
     msg_prop.setPriority(5);

     /* Get handle to the AQRawPayload object and populate it with raw data: */
     b_array = test_data.getBytes();

     raw_payload = message.getRawPayload();

     raw_payload.setStream(b_array, b_array.length);

     /* Create a AQEnqueueOption object with default options: */
     enq_option = new AQEnqueueOption();

     /* Enqueue the message: */
     queue.enqueue(enq_option, message);

     db_conn.commit();
}
```

Visual Basic（OO4O）：メッセージのエンキュー

オブジェクト型メッセージのエンキュー

```
'Prepare the message. MESSAGE_TYPE is a user defined type
' in the "AQ" schema
Set OraMsg = Q.AQMsg(1, "MESSAGE_TYPE")
Set OraObj = DB.CreateOraObject("MESSAGE_TYPE")

OraObj("subject").Value = "Greetings from OO4O"
OraObj("text").Value = "Text of a message originated from OO4O"

Set OraMsg.Value = OraObj
Msgid = Q.Enqueue
```

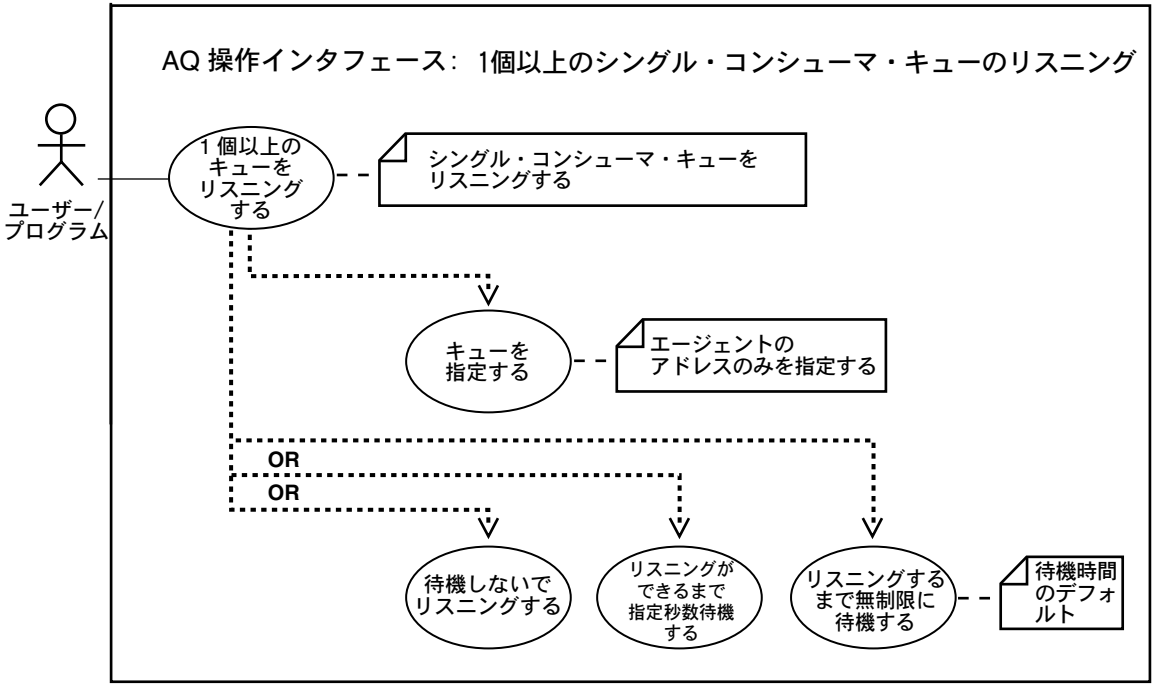
RAW 型メッセージのエンキュー

```
'Create an OraAQ object for the queue "DBQ"
Dim Q as object
Dim Msg as object
Dim OraSession as object
Dim DB as object

Set OraSession = CreateObject("OracleInProcServer.XOraSession")
Set OraDatabase = OraSession.OpenDatabase(mydb, "scott/tiger" 0&)
Set Q = DB.CreateAQ("DBQ")

'Get a reference to the AQMsg object
Set Msg = Q.AQMsg
Msg.Value = "Enqueue the first message to a RAW queue."

'Enqueue the message
Q.Enqueue()

'Enqueue another message.

Msg.Value = "Another message"
Q.Enqueue()

'Enqueue a message with nondefault properties.
Msg.Priority = ORAQMSG_HIGH_PRIORITY
Msg.Delay = 5
Msg.Value = "Urgent message"
Q.Enqueue()
Msg.Value = "The visibility option used in the enqueue call is
             ORAAQ_ENQ_IMMEDIATE"
Q.Visible = ORAAQ_ENQ_IMMEDIATE
```

```
Msgid = Q.Enqueue

'Enqueue Ahead of message Msgid_1
Msg.Value = "First Message to test Relative Message id"
Msg.Correlation = "RELATIVE_MESSAGE_ID"

Msgid_1 = Q.Enqueue
Msg.Value = "Second message to test RELATIVE_MESSAGE_ID is queued
             ahead of the First Message "
OraAq.relmsgid = Msgid_1
Msgid = Q.Enqueue
```

1 個以上のシングル・コンシューマ・キューのリスニング

図 11-6 1 個以上のシングル・コンシューマ・キューのリスニング



参照:

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-36 ページの「1 個以上のマルチ・コンシューマ・キューのリスニング」も参照してください。

使用上の注意

コールの引数に、エージェント・リストがあります。各エージェント・リストのアドレス・フィールドには、監視するキューを指定します。マルチ・コンシューマ・キューを監視するときは、エージェント名も指定する必要があります。シングル・コンシューマ・キューの場合は、エージェント名を指定しないでください。アドレスでサポートされているのは、ローカル・キューのみです。プロトコルは、将来の使用に備えて確保されています。

このコールは、リストにあるエージェントによって処理可能なメッセージがある場合に戻されるブロッキング・コールです。処理可能なエージェントが複数ある場合、最初にリストされたエージェントのみが戻されます。待機期限切れになった時点で 1 つもメッセージがない場合は、エラーになります。

Listen コールから正常に戻っても、指定されたキューの 1 つに指定されたエージェントの 1 つによって処理できるメッセージがあることを意味しているに過ぎません。処理するエージェントは、その関連メッセージをデキューする必要があります。

非永続キューには Listen をコールできないことに注意してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の LISTEN プロシージャ
- C（OCI）:『Oracle Call Interface プログラマーズ・ガイド』の第 16 章「リレーショナル機能」の OCIAQListen プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）:「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ Object」>「Monitoring Messages」を選択してください。
- Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQSession」の listen

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）: キューのリスニング（11-26 ページ）
- Java（JDBC）: キューのリスニング（11-26 ページ）
- C（OCI）: シングル・コンシューマ・キューのリスニング（11-27 ページ）

PL/SQL（DBMS_AQ）：キューのリスニング

```
/* The listen call allows you to monitor a list of queues for messages for
   specific agents. You need to have dequeue privileges for all the queues
   you wish to monitor. */
```

タイムアウト 0（ゼロ）でシングル・コンシューマ・キューをリスニングする

```
DECLARE
   Agent_w_msg      aq$_agent;
   My_agent_list    dbms_aq.agent_list_t;

BEGIN
   /* NOTE:  MCQ1, MCQ2, MCQ3 are multiconsumer queues  in SCOTT's schema
   *        SCQ1, SCQ2, SCQ3 are single-consumer queues in SCOTT's schema
   */

   Qlist(1):= aq$_agent(NULL, 'scott.SCQ1',  NULL);
   Qlist(2):= aq$_agent(NULL, 'SCQ2', NULL);
   Qlist(3):= aq$_agent(NULL, 'SCQ3', NULL);

   /* Listen with a time-out of zero: */
   DBMS_AQ.LISTEN(
      Agent_list   =>   My_agent_list,
      Wait         =>   0,
      Agent        =>   agent_w_msg);
   DBMS_OUTPUT.PUT_LINE('Message in Queue :- ' ||  agent_w_msg.address);
   DBMS_OUTPUT.PUT_LINE('');
END;
```

Java（JDBC）：キューのリスニング

```
public static void monitor_status_queue(Connection db_conn)
{
    AQSession         aq_sess;
    AQAgent[]         agt_list = null;
    AQAgent           ret_agt  = null;

    try
    {
        /* Create an AQ Session: */
        aq_sess = AQDriverManager.createAQSession(db_conn);

/* Construct the waiters list: */
agt_list = new AQAgent[3];
```

```
agt_list[0] = new AQAgent(null, "scott.SCQ1",0);
agt_list[1] = new AQAgent (null, "SCQ2",0);
agt_list[2] = new AQAgent (null, "SCQ3",0);

/* Wait for order status messages for 120 seconds: */
ret_agt = aq_sess.listen(agt_list, 120);

System.out.println("Message available for agent: " +
   ret_agt.getName()  + "   "  + ret_agt.getAddress());

    }
    catch (AQException aqex)
    {
System.out.println("Exception-1: " + aqex);
    }
    catch (Exception ex)
    {
        System.out.println("Exception-2: " + ex);
    }

}
```

C（OCI）: シングル・コンシューマ・キューのリスニング

タイムアウト 0（ゼロ）でシングル・コンシューマ・キューをリスニングする

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>

static void checkerr(errhp, status)
LNOCIError *errhp;
sword status;
{
    text errbuf[512];
    ub4 buflen;
    sb4 errcode;

    switch (status)
    {
   case OCI_SUCCESS:
      break;
   case OCI_SUCCESS_WITH_INFO:
```

```
        printf("Error - OCI_SUCCESS_WITH_INFO\n");
        break;
    case OCI_NEED_DATA:
        printf("Error - OCI_NEED_DATA\n");
        break;
    case OCI_NO_DATA:
        printf("Error - OCI_NO_DATA\n");
        break;
    case OCI_ERROR:
        OCIErrorGet ((dvoid *) errhp, (ub4) 1, (text *) NULL, &errcode,
        errbuf, (ub4) sizeof(errbuf), (ub4) OCI_HTYPE_ERROR);
        printf("Error - %s\n", errbuf);
        break;
    case OCI_INVALID_HANDLE:
        printf("Error - OCI_INVALID_HANDLE\n");
        break;
    case OCI_STILL_EXECUTING:
        printf("Error - OCI_STILL_EXECUTE\n");
        break;
    case OCI_CONTINUE:
        printf("Error - OCI_CONTINUE\n");
        break;
    default:
    break;
     }
}

/* set agent into descriptor */
void SetAgent(agent, appname, queue,errhp)

LNOCIAQAgent  *agent;
text        *appname;
text        *queue;
LNOCIError    *errhp;
{

  OCIAttrSet(agent, OCI_DTYPE_AQAGENT,
     appname ? (dvoid *)appname : (dvoid *)"",
     appname ? strlen((const char *)appname) : 0,
        OCI_ATTR_AGENT_NAME, errhp);

  OCIAttrSet(agent, OCI_DTYPE_AQAGENT,
     queue ? (dvoid *)queue : (dvoid *)"",
     queue ? strlen((const char *)queue) : 0,
        OCI_ATTR_AGENT_ADDRESS, errhp);

  printf("Set agent name to %s\n", appname ? (char *)appname : "NULL");
```

```
  printf("Set agent address to %s\n", queue ? (char *)queue : "NULL");
}

/* get agent from descriptor */
void GetAgent(agent, errhp)
LNOCIAQAgent *agent;
LNOCIError   *errhp;
{
text      *appname;
text      *queue;
ub4       appsz;
ub4       queuesz;

  if (!agent )
  {
    printf("agent was NULL \n");
    return;
  }
  checkerr(errhp, OCIAttrGet(agent, OCI_DTYPE_AQAGENT,
     (dvoid *)&appname, &appsz, OCI_ATTR_AGENT_NAME, errhp));
  checkerr(errhp, OCIAttrGet(agent, OCI_DTYPE_AQAGENT,
     (dvoid *)&queue, &queuesz, OCI_ATTR_AGENT_ADDRESS, errhp));
  if (!appsz)
     printf("agent name: NULL\n");
  else printf("agent name: %.*s\n", appsz, (char *)appname);
  if (!queuesz)
     printf("agent address: NULL\n");
  else printf("agent address: %.*s\n", queuesz, (char *)queue);
}

int main()
{
  OCIEnv *envhp;
  OCIServer *srvhp;
  OCIError *errhp;
  OCISvcCtx *svchp;
  OCISession *usrhp;
  OCIAQAgent *agent_list[3];
  OCIAQAgent *agent = (OCIAQAgent *)0;
  /* added next 2 121598 */
  int i;

 /* Standard OCI Initialization */

  OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
     (dvoid * (*)()) 0,  (void (*)()) 0 );
```

```
OCIHandleAlloc( (dvoid *) NULL, (dvoid **) &envhp,
           (ub4) OCI_HTYPE_ENV, 0, (dvoid **) 0);

OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 0, (dvoid **) 0);

OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
   0, (dvoid **) 0);

OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
   0, (dvoid **) 0);

OCIServerAttach( srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
   0, (dvoid **) 0);

/* set attribute server context in the service context */
OCIAttrSet( (dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
   (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

/* allocate a user context handle */
OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
    (size_t) 0, (dvoid **) 0);

/* allocate a user context handle */
OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
   (size_t) 0, (dvoid **) 0);

OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
   (dvoid *)"scott", (ub4)strlen("scott"), OCI_ATTR_USERNAME, errhp);

OCIAttrSet((dvoid *) usrhp, (ub4) OCI_HTYPE_SESSION,
    (dvoid *) "tiger", (ub4) strlen("tiger"),
    (ub4) OCI_ATTR_PASSWORD, errhp);

OCISessionBegin (svchp, errhp, usrhp, OCI_CRED_RDBMS, OCI_DEFAULT);

OCIAttrSet((dvoid *)svchp, (ub4)OCI_HTYPE_SVCCTX,
   (dvoid *)usrhp, (ub4)0, OCI_ATTR_SESSION, errhp);

/* AQ LISTEN Initialization - allocate agent handles */
for (i = 0; i < 3; i++)
{
   agent_list[i] = (OCIAQAgent *)0;
   OCIDescriptorAlloc(envhp, (dvoid **)&agent_list[i],
       OCI_DTYPE_AQAGENT, 0, (dvoid **)0);
}
```

```
  /*
   *   SCQ1, SCQ2, SCQ3 are single-consumer queues in SCOTT's schema
   */

  SetAgent(agent_list[0], (text *)0, "SCOTT.SCQ1", errhp);
  SetAgent(agent_list[1], (text *)0, "SCOTT.SCQ2", errhp);
  SetAgent(agent_list[2], (text *)0, "SCOTT.SCQ3", errhp);

  checkerr(errhp,OCIAQListen(svchp, errhp, agent_list, 3, 0, &agent, 0));

  printf("MESSAGE for :- \n");
  GetAgent(agent, errhp);
  printf("\n");

}
```

タイムアウト 120 秒でシングル・コンシューマ・キューをリスニングする

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>

static void checkerr(errhp, status)
LNOCIError *errhp;
sword status;
{
    text errbuf[512];
    ub4 buflen;
    sb4 errcode;

    switch (status)
    {
   case OCI_SUCCESS:
       break;
   case OCI_SUCCESS_WITH_INFO:
       printf("Error - OCI_SUCCESS_WITH_INFO\n");
       break;
   case OCI_NEED_DATA:
       printf("Error - OCI_NEED_DATA\n");
       break;
   case OCI_NO_DATA:
       printf("Error - OCI_NO_DATA\n");
       break;
   case OCI_ERROR:
```

```
        OCIErrorGet ((dvoid *) errhp, (ub4) 1, (text *) NULL, &errcode,
        errbuf, (ub4) sizeof(errbuf), (ub4) OCI_HTYPE_ERROR);
        printf("Error - %s\n", errbuf);
        break;
    case OCI_INVALID_HANDLE:
        printf("Error - OCI_INVALID_HANDLE\n");
        break;
    case OCI_STILL_EXECUTING:
        printf("Error - OCI_STILL_EXECUTE\n");
        break;
    case OCI_CONTINUE:
        printf("Error - OCI_CONTINUE\n");
        break;
    default:
    break;
     }
}

/* set agent into descriptor */
/* void SetAgent(agent, appname, queue) */
void SetAgent(agent, appname, queue,errhp)

LNOCIAQAgent  *agent;
text        *appname;
text        *queue;
LNOCIError    *errhp;
{

  OCIAttrSet(agent, OCI_DTYPE_AQAGENT,
     appname ? (dvoid *)appname : (dvoid *)"",
     appname ? strlen((const char *)appname) : 0,
        OCI_ATTR_AGENT_NAME, errhp);

  OCIAttrSet(agent, OCI_DTYPE_AQAGENT,
     queue ? (dvoid *)queue : (dvoid *)"",
     queue ? strlen((const char *)queue) : 0,
        OCI_ATTR_AGENT_ADDRESS, errhp);

  printf("Set agent name to %s\n", appname ? (char *)appname : "NULL");
  printf("Set agent address to %s\n", queue ? (char *)queue : "NULL");
}

/* get agent from descriptor */
void GetAgent(agent, errhp)
LNOCIAQAgent *agent;
LNOCIError   *errhp;
{
```

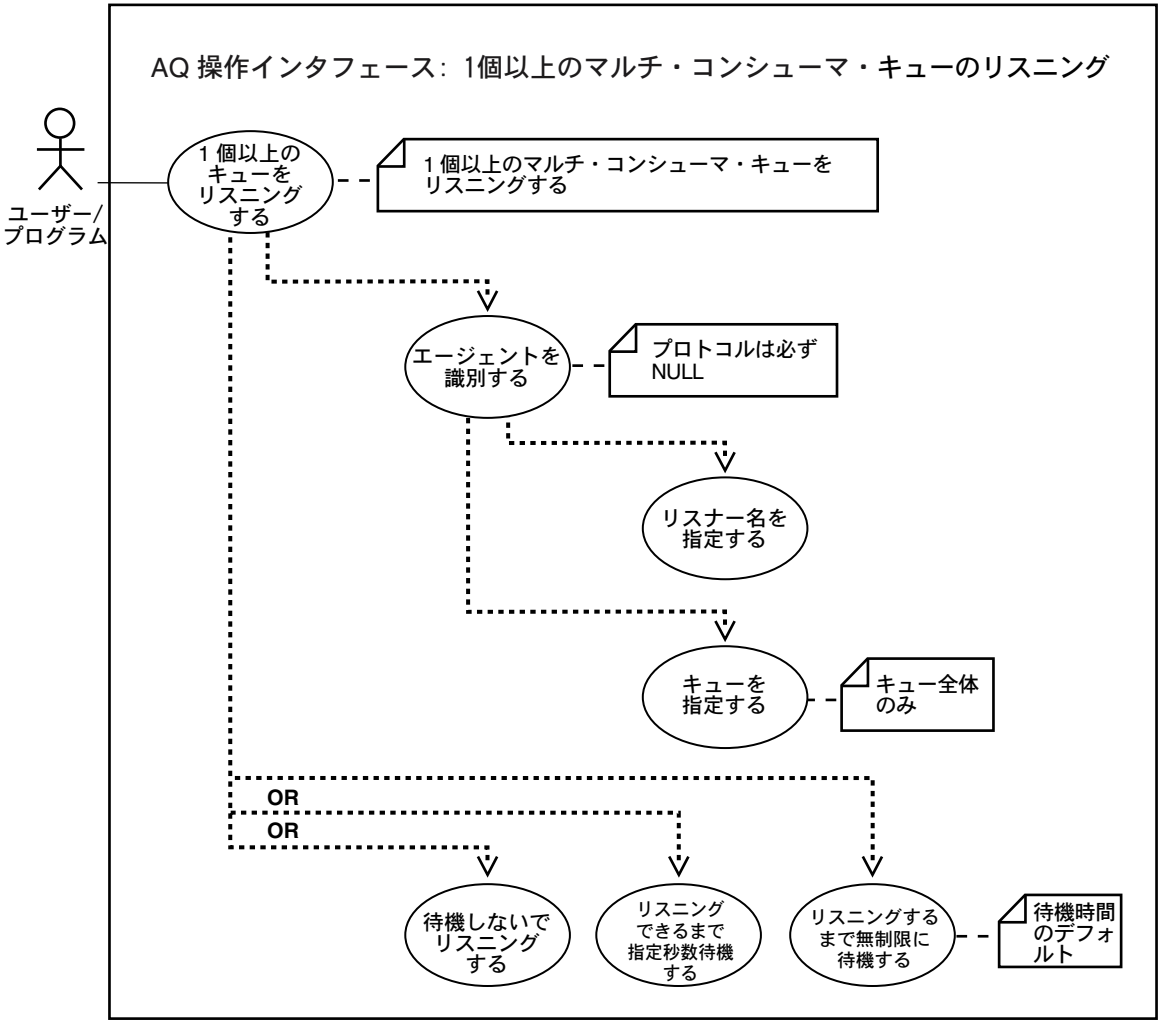
```
text      *appname;
text      *queue;
ub4       appsz;
ub4       queuesz;

  if (!agent )
  {
    printf("agent was NULL \n");
    return;
  }
  checkerr(errhp, OCIAttrGet(agent, OCI_DTYPE_AQAGENT,
     (dvoid *)&appname, &appsz, OCI_ATTR_AGENT_NAME, errhp));
  checkerr(errhp, OCIAttrGet(agent, OCI_DTYPE_AQAGENT,
     (dvoid *)&queue, &queuesz, OCI_ATTR_AGENT_ADDRESS, errhp));
  if (!appsz)
     printf("agent name: NULL\n");
  else printf("agent name: %.*s\n", appsz, (char *)appname);
  if (!queuesz)
     printf("agent address: NULL\n");
  else printf("agent address: %.*s\n", queuesz, (char *)queue);
}

int main()
{
  OCIEnv *envhp;
  OCIServer *srvhp;
  OCIError *errhp;
  OCISvcCtx *svchp;
  OCISession *usrhp;
  OCIAQAgent *agent_list[3];
  OCIAQAgent *agent = (OCIAQAgent *)0;
  /* added next 2 121598 */
  int i;

 /* Standard OCI Initialization */

  OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
     (dvoid * (*)()) 0,  (void (*)()) 0 );

  OCIHandleAlloc( (dvoid *) NULL, (dvoid **) &envhp,
             (ub4) OCI_HTYPE_ENV, 0, (dvoid **) 0);

  OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 0, (dvoid **) 0);

  OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
     0, (dvoid **) 0);
```

```
OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
   0, (dvoid **) 0);

OCIServerAttach( srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
   0, (dvoid **) 0);

/* set attribute server context in the service context */
OCIAttrSet( (dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
   (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

/* allocate a user context handle */
OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
    (size_t) 0, (dvoid **) 0);

/* allocate a user context handle */
OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
   (size_t) 0, (dvoid **) 0);

OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
   (dvoid *)"scott", (ub4)strlen("scott"), OCI_ATTR_USERNAME, errhp);

OCIAttrSet((dvoid *) usrhp, (ub4) OCI_HTYPE_SESSION,
    (dvoid *) "tiger", (ub4) strlen("tiger"),
    (ub4) OCI_ATTR_PASSWORD, errhp);

OCISessionBegin (svchp, errhp, usrhp, OCI_CRED_RDBMS, OCI_DEFAULT);

OCIAttrSet((dvoid *)svchp, (ub4)OCI_HTYPE_SVCCTX,
   (dvoid *)usrhp, (ub4)0, OCI_ATTR_SESSION, errhp);

/* AQ LISTEN Initialization - allocate agent handles */
for (i = 0; i < 3; i++)
{
   agent_list[i] = (OCIAQAgent *)0;
   OCIDescriptorAlloc(envhp, (dvoid **)&agent_list[i],
       OCI_DTYPE_AQAGENT, 0, (dvoid **)0);
}

/*
 *   SCQ1, SCQ2, SCQ3 are single-consumer queues in SCOTT's schema
 */

SetAgent(agent_list[0], (text *)0, "SCOTT.SCQ1", errhp);
SetAgent(agent_list[1], (text *)0, "SCOTT.SCQ2", errhp);
SetAgent(agent_list[2], (text *)0, "SCOTT.SCQ3", errhp);
```

```
  checkerr(errhp,OCIAQListen(svchp, errhp, agent_list, 3, 120, &agent, 0));

  printf("MESSAGE for :- \n");
  GetAgent(agent, errhp);
  printf("\n");

}
```

1 個以上のマルチ・コンシューマ・キューのリスニング

図 11-7 1 個以上のマルチ・コンシューマ・キューのリスニング



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-24 ページの「1 個以上のシングル・コンシューマ・キューのリスニング」も参照してください。

使用上の注意

11-24 ページの「1 個以上のシングル・コンシューマ・キューのリスニング」の「使用上の注意」を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の `LISTEN` プロシージャ
- C（OCI）：『Oracle Call Interface プログラマーズ・ガイド』の第 16 章「リレーショナル機能」の `OCIAQListen` プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）：「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ Object」>「Monitoring Messages」を選択してください。
- この機能は、Java API を介しては使用できません。

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）：キューのリスニング（11-37 ページ）
- C（OCI）：キューのリスニング（11-39 ページ）

PL/SQL（DBMS_AQ）：キューのリスニング

```
/* The listen call allows you to monitor a list of queues for messages for
   specific agents. You need to have dequeue privileges for all the queues
   you wish to monitor. */
```

タイムアウト 0（ゼロ）でマルチ・コンシューマ・キューをリスニングする

```
DECLARE
   Agent_w_msg      aq$_agent;
   My_agent_list    dbms_aq.agent_list_t;

BEGIN
   /* NOTE:  MCQ1, MCQ2, MCQ3 are multiconsumer queues  in SCOTT's schema
   *        SCQ1, SCQ2, SCQ3 are single-consumer queues in SCOTT's schema
   */
    Qlist(1):= aq$_agent('agent1', 'MCQ1',  NULL);
    Qlist(2):= aq$_agent('agent2', 'scott.MCQ2', NULL);
    Qlist(3):= aq$_agent('agent3', 'scott.MCQ3', NULL);

   /* Listen with a time-out of zero: */
   DBMS_AQ.LISTEN(
      agent_list   =>    My_agent_list,
      wait         =>    0,
      agent        =>    agent_w_msg);
   DBMS_OUTPUT.PUT_LINE('Message in Queue :- ' ||  agent_w_msg.address);
   DBMS_OUTPUT.PUT_LINE('');
END;
   /
```

タイムアウト 100 秒でマルチ・コンシューマ・キューのグループをリスニングする

```
DECLARE
   Agent_w_msg      aq$_agent;
   My_agent_list    dbms_aq.agent_list_t;

BEGIN
   /* NOTE:  MCQ1, MCQ2, MCQ3 are multiconsumer queues  in SCOTT's schema
   *        SCQ1, SCQ2, SCQ3 are single-consumer queues in SCOTT's schema
   */
   Qlist(1):= aq$_agent('agent1', 'MCQ1',  NULL);
   Qlist(2):= aq$_agent(NULL, 'scott.SQ1', NULL);
   Qlist(3):= aq$_agent('agent3', 'scott.MCQ3', NULL);
   /* Listen with a time-out of 100 seconds */
   DBMS_AQ.LISTEN(
      Agent_list   =>   My_agent_list,
      Wait         =>   100,
      Agent        =>    agent_w_msg);
      DBMS_OUTPUT.PUT_LINE('Message in Queue :- ' ||  agent_w_msg.address
                           || 'for agent' || agent_w_msg.name);
      DBMS_OUTPUT.PUT_LINE('');
```

```
END;
/
```

C（OCI）：キューのリスニング

タイムアウト 0（ゼロ）、120 秒および 100 秒でマルチ・コンシューマ・キューをリスニングする

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>

static void checkerr(errhp, status)
LNOCIError *errhp;
sword status;
{
    text errbuf[512];
    ub4 buflen;
    sb4 errcode;

    switch (status)
    {
   case OCI_SUCCESS:
       break;
   case OCI_SUCCESS_WITH_INFO:
       printf("Error - OCI_SUCCESS_WITH_INFO\n");
       break;
   case OCI_NEED_DATA:
       printf("Error - OCI_NEED_DATA\n");
       break;
   case OCI_NO_DATA:
       printf("Error - OCI_NO_DATA\n");
       break;
   case OCI_ERROR:
       OCIErrorGet ((dvoid *) errhp, (ub4) 1, (text *) NULL, &errcode,
       errbuf, (ub4) sizeof(errbuf), (ub4) OCI_HTYPE_ERROR);
       printf("Error - %s\n", errbuf);
       break;
   case OCI_INVALID_HANDLE:
       printf("Error - OCI_INVALID_HANDLE\n");
       break;
   case OCI_STILL_EXECUTING:
       printf("Error - OCI_STILL_EXECUTE\n");
       break;
   case OCI_CONTINUE:
```

```
        printf("Error - OCI_CONTINUE\n");
        break;
    default:
    break;
     }
}

void SetAgent(OCIAQAgent *agent,
         text       *appname,
         text       *queue,
         OCIError   *errhp,
         OCIEnv     *envhp);

void GetAgent(OCIAQAgent *agent,
         OCIError   *errhp);

/*-----------------------------------------------------------------*/
/* OCI Listen examples for multiconsumers                          */
/*                                                                  */
void SetAgent(agent, appname, queue, errhp)
LNOCIAQAgent    *agent;
text          *appname;
text          *queue;
LNOCIError      *errhp;
{
  OCIAttrSet(agent,
        OCI_DTYPE_AQAGENT,
        appname ? (dvoid *)appname : (dvoid *)"",
        appname ? strlen((const char *)appname) : 0,
             OCI_ATTR_AGENT_NAME,
        errhp);

  OCIAttrSet(agent,
        OCI_DTYPE_AQAGENT,
        queue ? (dvoid *)queue : (dvoid *)"",
        queue ? strlen((const char *)queue) : 0,
             OCI_ATTR_AGENT_ADDRESS,
        errhp);

  printf("Set agent name to %s\n", appname ? (char *)appname : "NULL");
  printf("Set agent address to %s\n", queue ? (char *)queue : "NULL");
}

/* get agent from descriptor */
void GetAgent(agent, errhp)
LNOCIAQAgent *agent;
LNOCIError   *errhp;
```

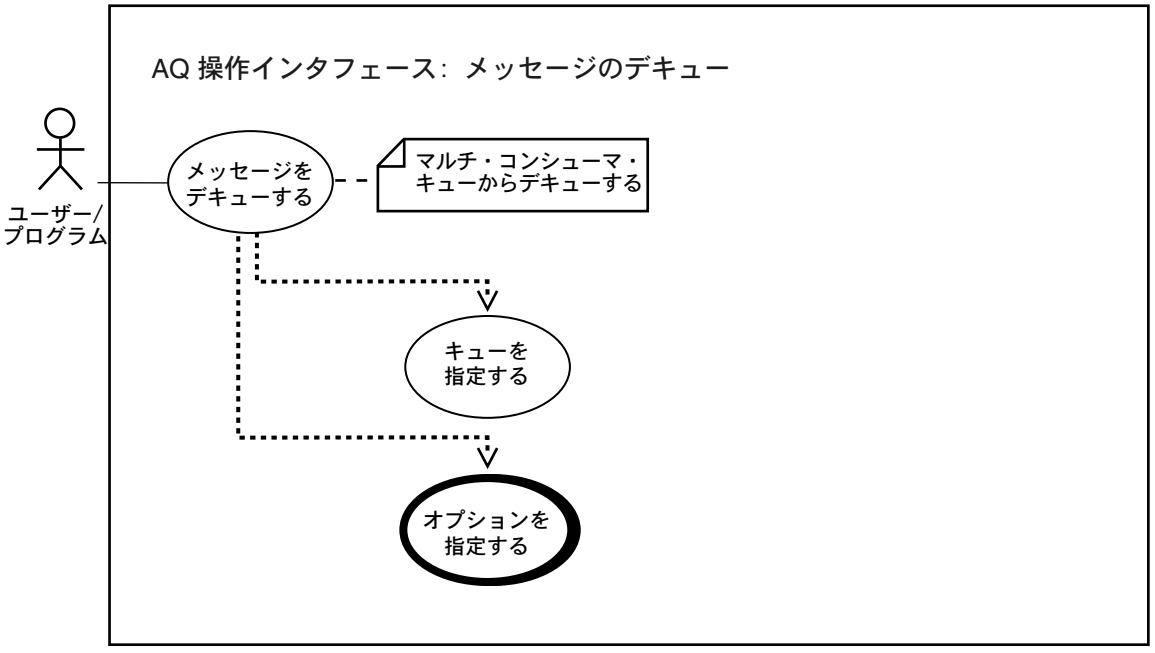
```
{
   text      *appname;
   text      *queue;
   ub4       appsz;
   ub4       queuesz;

   if (!agent )
  {
     printf("agent was NULL \n");
     return;
  }
  checkerr(errhp, OCIAttrGet(agent, OCI_DTYPE_AQAGENT,
     (dvoid *)&appname, &appsz, OCI_ATTR_AGENT_NAME, errhp));
  checkerr(errhp, OCIAttrGet(agent, OCI_DTYPE_AQAGENT,
     (dvoid *)&queue, &queuesz, OCI_ATTR_AGENT_ADDRESS, errhp));
  if (!appsz)
     printf("agent name: NULL\n");
  else printf("agent name: %.*s\n", appsz, (char *)appname);
  if (!queuesz)
     printf("agent address: NULL\n");
  else printf("agent address: %.*s\n", queuesz, (char *)queue);
}

/* main from AQ Listen to multiconsumer Queues */

/*  int main() */
int main(char *argv, int argc)
{
    OCIEnv     *envhp;
    OCIServer  *srvhp;
    OCIError   *errhp;
    OCISvcCtx  *svchp;
    OCISession *usrhp;
    OCIAQAgent *agent_list[3];
    OCIAQAgent *agent;
    int         i;

 /* Standard OCI Initialization */

  OCIInitialize((ub4) OCI_OBJECT,
      (dvoid *)0,
      (dvoid * (*)()) 0,
      (dvoid * (*)()) 0,
      (void (*)()) 0 );

  OCIHandleAlloc( (dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
     0, (dvoid **) 0);
```

```
OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 0, (dvoid **)0);

OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
   0, (dvoid **) 0);

OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
   0, (dvoid **) 0);

OCIServerAttach( srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
   0, (dvoid **) 0);

/* set attribute server context in the service context */
OCIAttrSet( (dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
   (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

 /* allocate a user context handle */
OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
   (size_t) 0, (dvoid **) 0);

 /* allocate a user context handle */
OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
   (size_t) 0, (dvoid **) 0);


OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
   (dvoid *)"scott", (ub4)strlen("scott"), OCI_ATTR_USERNAME, errhp);


 OCIAttrSet((dvoid *) usrhp, (ub4) OCI_HTYPE_SESSION,
    (dvoid *) "tiger", (ub4) strlen("tiger"),
    (ub4) OCI_ATTR_PASSWORD, errhp);

OCISessionBegin (svchp, errhp, usrhp, OCI_CRED_RDBMS, OCI_DEFAULT);

OCIAttrSet((dvoid *)svchp, (ub4)OCI_HTYPE_SVCCTX,
   (dvoid *)usrhp, (ub4)0, OCI_ATTR_SESSION, errhp);

/* AQ LISTEN Initialization - allocate agent handles */
for (i = 0; i < 3; i++)
{
   OCIDescriptorAlloc(envhp, (dvoid **)&agent_list[i],
       OCI_DTYPE_AQAGENT, 0, (dvoid **)0);
}
```

```
  /*
   *  MCQ1, MCQ2, MCQ3 are multiconsumer queues in SCOTT's schema
   */
  /* Listening to Multiconsumer Queues with Zero Timeout */

  SetAgent(agent_list[0], "app1", "MCQ1", errhp);
  SetAgent(agent_list[1], "app2", "MCQ2", errhp);
  SetAgent(agent_list[2], "app3", "MCQ3", errhp);

  checkerr(errhp, OCIAQListen(svchp, errhp, agent_list, 3, 0, &agent, 0));

  printf("MESSAGE for :- \n");
  GetAgent(agent, errhp);
  printf("\n");


  /* Listening to Multiconsumer Queues with Timeout of 120 Seconds */

  SetAgent(agent_list[0], "app1", "SCOTT.MCQ1", errhp);
  SetAgent(agent_list[1], "app2", "SCOTT.MCQ2", errhp);
  SetAgent(agent_list[2], "app3", "SCOTT.MCQ3", errhp);

  checkerr(errhp, OCIAQListen(svchp, errhp, agent_list, 3, 120, &agent, 0));

  printf("MESSAGE for :- \n");
  GetAgent(agent, errhp);
  printf("\n");


  /* Listening to a Mixture of Single and Multiconsumer Queues
   * with a Timeout of 100 Seconds
   */

  SetAgent(agent_list[0], "app1", "SCOTT.MCQ1", errhp);
  SetAgent(agent_list[1], "app2", "SCOTT.MCQ2", errhp);
  SetAgent(agent_list[2], (text *)0, "SCOTT.SCQ3", errhp);

  checkerr(errhp, OCIAQListen(svchp, errhp, agent_list, 3, 100, &agent, 0));

  printf("MESSAGE for :- \n");
  GetAgent(agent, errhp);
  printf("\n");

}
```

メッセージのデキュー

図 11-8 メッセージのデキュー



参照:

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-47 ページの「シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定）」も参照してください。
- 11-52 ページの「マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定）」も参照してください。

用途

メッセージを、指定したキューからデキューします。

使用上の注意

メッセージの検索基準およびデキュー順序

- デキューするメッセージの検索基準は、デキュー・オプションのコンシューマ名、msgid および correlation パラメータによって決定されます。msgid によって、デキューされるメッセージが一意に識別されます。相関識別子は、AQ では解釈されない、アプリケーション定義の識別子です。
- msgid を指定しないかぎり、READY 状態のメッセージのみがデキューされます。
- デキューの順序は、デキュー・オプションの msgid および相関識別子でオーバーライドされないかぎり、キュー表の作成時に指定されたソート順によって決定されます。
- キュー操作には、データベース読込み一貫性メカニズムが適用されます。たとえば、トランザクションによる参照が始まってからエンキューされたメッセージを、BROWSE コールでは参照できません。

キューのナビゲート

デキュー中のデフォルト NAVIGATION パラメータは、NEXT MESSAGE です。つまり、後続のデキューでは、最初のデキューで取得したスナップショットを基にするキューからメッセージが取り出されます。特別なケースとして、最初のデキュー・コマンドの後にエンキューされたメッセージは、キューに残っているメッセージがすべて処理されてから処理されます。すべてのメッセージがすでにキューにエンキューされている場合、またはキューに優先順位が設定されていない場合にはこのような処理で十分です。ただし、デキュー・コマンドごとにキューの最初のメッセージを処理する必要がある場合は、アプリケーションで FIRST MESSAGE ナビゲーション・オプションを使用する必要があります。通常は、すでにエンキューされたメッセージの処理中に、優先順位の高いメッセージがキューに到着する場合に、このような処理が必要になります。

注意： 同時にエンキューされるメッセージがある場合には、FIRST_MESSAGE ナビゲーション・オプションを使用するとより効率的です。FIRST MESSAGE オプションが指定されないと、AQ では最初のデキュー・コマンドと同様のスナップショットを継続して生成する必要があり、パフォーマンスの低下につながります。FIRST MESSAGE オプションが指定されると、AQ では、すべてのデキュー・コマンドに新規のスナップショットを使用します。

メッセージのグループ化によるデキュー

- メッセージをグループ化できるキューに同一トランザクションのメッセージがエンキューされると、グループが形成されます。トランザクションに 1 つのメッセージしかエンキューされていない場合でも、事実上 1 つのグループが形成されます。1 つのトランザクションでグループ化できるメッセージの数に、上限はありません。
- メッセージをグループ化できないキューでは、LOCKED または REMOVE モードのデキューによって 1 つのメッセージのみがロックされます。それに対して、グループの一部であるメッセージをデキューする操作では、グループ全体がロックされます。これは、グループのメッセージ全体を基本単位として処理する必要がある場合に有効です。
- グループのメッセージ全体をデキューした後でデキュー操作をすると、グループのすべてのメッセージが処理されたことを示すエラーが戻されます。その後、アプリケーションは NEXT TRANSACTION を使用して、次の使用可能なグループからのメッセージのデキューを開始します。使用可能なグループがない場合、指定された WAIT 時間の経過後、デキューはタイムアウトになります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の DEQUEUE プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）：「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ」を選択してください。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQQueue」の dequeue

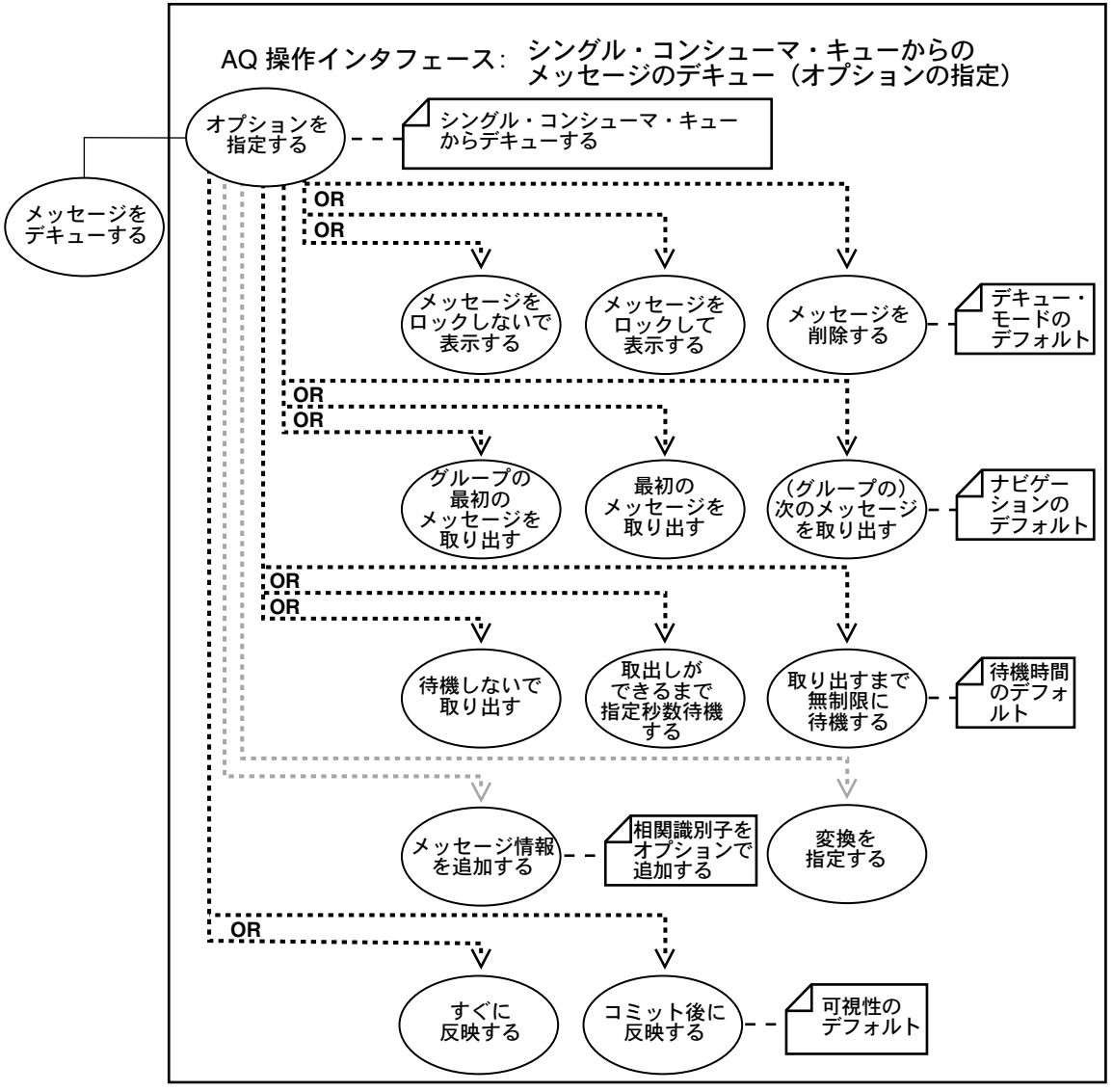
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）：オブジェクト型メッセージのデキュー（11-49 ページ）
- Java（JDBC）：シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定）（11-49 ページ）
- Visual Basic（OO4O）：メッセージのデキュー（11-50 ページ）

シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定）

図 11-9 シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定）



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-44 ページの「メッセージのデキュー」も参照してください。
- 11-52 ページの「マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定）」も参照してください。

用途

デキュー操作で使用可能なオプションを指定します。

使用上の注意

一般的には、メッセージのコンシューマはデキュー・インタフェースによってメッセージにアクセスすると考えられます。処理済または処理が終了していないメッセージは、メッセージ ID または SELECT 文を使用して参照できます。

変換を指定した場合は、コール側にメッセージを戻す前に適用されます。変換は、キューのユーザー定義型がコール側に戻す型にマップされるように定義する必要があります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の DEQUEUE プロシージャ
- Visual Basic（OO4O オンライン・ヘルプ）：「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ」を選択してください。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQDequeueOption」

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- PL/SQL（DBMS_AQ）：オブジェクト型メッセージのデキュー（11-49 ページ）
- Java（JDBC）：シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定）（11-49 ページ）
- Visual Basic（OO4O）：メッセージのデキュー（11-50 ページ）

PL/SQL（DBMS_AQ）：オブジェクト型メッセージのデキュー

```
/* Dequeue from msg_queue: */
DECLARE
dequeue_options     dbms_aq.dequeue_options_t;
message_properties  dbms_aq.message_properties_t;
message_handle      RAW(16);
message             aq.message_typ;

BEGIN
   DBMS_AQ.DEQUEUE(
      queue_name          =>     'msg_queue',
      dequeue_options      =>    dequeue_options,
      message_properties  =>     message_properties,
      payload             =>     message,
      msgid               =>     message_handle);

   DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
                                       ' ... ' || message.text );
   COMMIT;
END;
```

Java（JDBC）：シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定）

```
/* Dequeue a message with correlation id = 'RUSH' */
public static void example(AQSession aq_sess) throws AQException, SQLException
{
     AQQueue                 queue;
     AQMessage               message;
     AQRawPayload            raw_payload;
     AQDequeueOption         deq_option;
     byte[]                  b_array;
     Connection              db_conn;

     db_conn = ((AQOracleSession)aq_sess).getDBConnection();

     queue = aq_sess.getQueue ("aq", "msg_queue");

     /* Create a AQDequeueOption object with default options: */
     deq_option = new AQDequeueOption();

     deq_option.setCorrelation("RUSH");

     /* Dequeue a message */
     message = queue.dequeue(deq_option);
```

```
        System.out.println("Successful dequeue");

        /* Retrieve raw data from the message: */
        raw_payload = message.getRawPayload();

        b_array = raw_payload.getBytes();

        db_conn.commit();
    }
```

Visual Basic（OO4O）：メッセージのデキュー

RAW 型メッセージのデキュー

```
'Dequeue the first message available
Q.Dequeue()
Set Msg = Q.QMsg

'Display the message content
MsgBox Msg.Value

'Dequeue the first message available without removing it
' from the queue
Q.DequeueMode = ORAAQ_DEQ_BROWSE

'Dequeue the first message with the correlation identifier
' equal to "RELATIVE_MSG_ID"
Q.Navigation = ORAAQ_DQ_FIRST_MSG
Q.correlate = "RELATIVE_MESSAGE_ID"
Q.Dequeue

'Dequeue the next message with the correlation identifier

' of "RELATIVE_MSG_ID"
Q.Navigation = ORAAQ_DQ_NEXT_MSG
Q.Dequeue()

'Dequeue the first high priority message
Msg.Priority = ORAQMSG_HIGH_PRIORITY
Q.Dequeue()

'Dequeue the message enqueued with message id of Msgid_1
Q.DequeueMsgid = Msgid_1
Q.Dequeue()

'Dequeue the message meant for "ANDY"
```

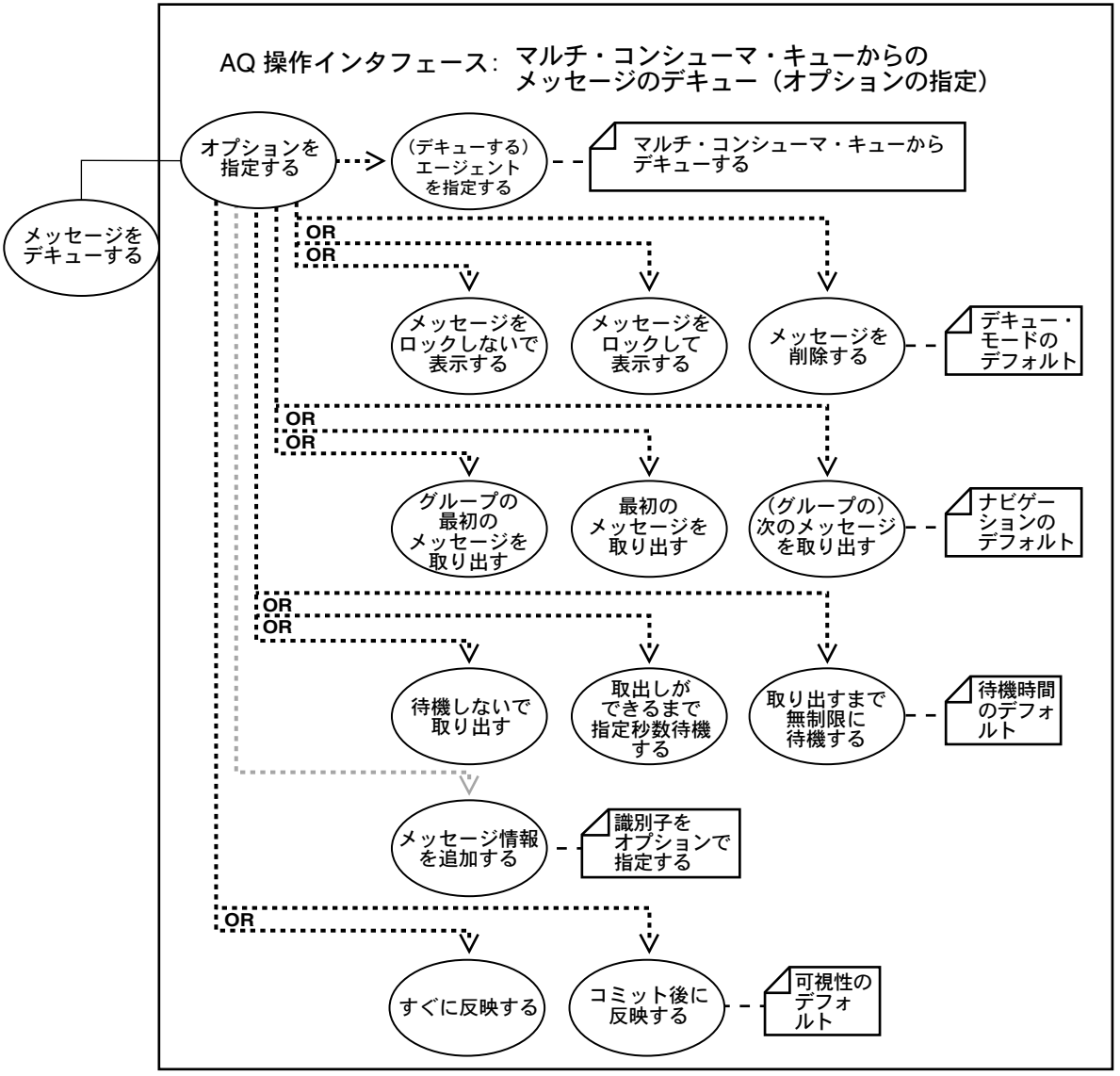
```
Q.consumer = "ANDY"
Q.Dequeue()

'Return immediately if there is no message on the queue
Q.wait = ORAAQ_DQ_NOWAIT
Q.Dequeue()
```

オブジェクト型メッセージのデキュー

```
Set OraObj = DB.CreateOraObject("MESSAGE_TYPE")
Set QMsg = Q.AQMsg(1, "MESSAGE_TYPE")

'Dequeue the first message available without removing it
Q.Dequeue()
OraObj = QMsg.Value

'Display the subject and data
MsgBox OraObj!subject & OraObj!Data
```

マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定）

図 11-10 マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定）



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-44 ページの「メッセージのデキュー」も参照してください。
- 11-47 ページの「シングル・コンシューマ・キューからのメッセージのデキュー（オプションの指定）」も参照してください。

用途

デキュー操作で使用可能なオプションを指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の DEQUEUE プロシージャ
- Visual Basic（OO4O）：参照マニュアルはありません。
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「AQDequeueOption」

例

ここでは、次のプログラム環境の例を示します。

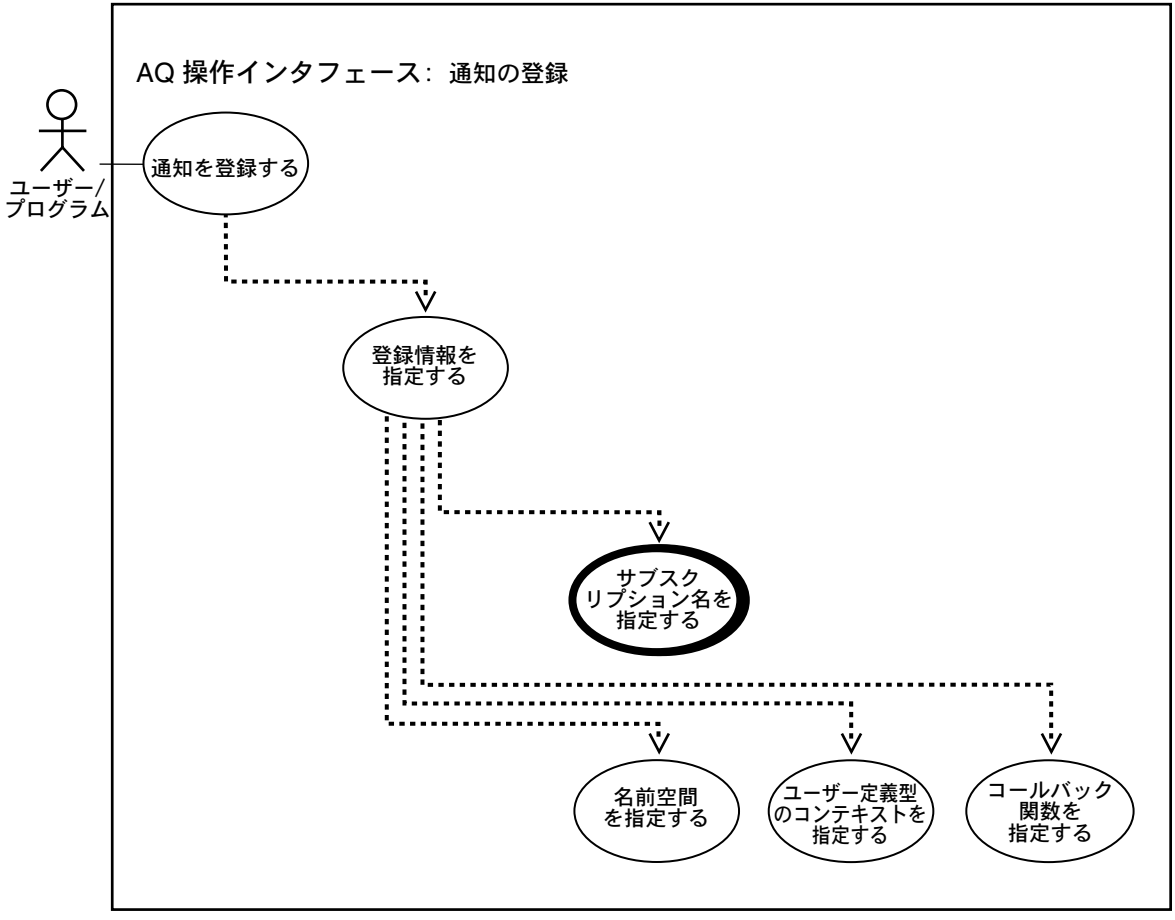
- Java（JDBC）：マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定）（11-54 ページ）

Java（JDBC）：マルチ・コンシューマ・キューからのメッセージのデキュー（オプションの指定）

```
/* Dequeue a message for subscriber1 in browse mode*/
public static void example(AQSession aq_sess) throws AQException, SQLException
{
     AQQueue                  queue;
     AQMessage                message;
     AQRawPayload             raw_payload;
     AQDequeueOption          deq_option;
     byte[]                   b_array;
     Connection               db_conn;

     db_conn = ((AQOracleSession)aq_sess).getDBConnection();

     queue = aq_sess.getQueue ("aq", "priority_msg_queue");

     /* Create a AQDequeueOption object with default options: */
     deq_option = new AQDequeueOption();

     /* Set dequeue mode to BROWSE */
     deq_option.setDequeueMode(AQDequeueOption.DEQUEUE_BROWSE);

     /* Dequeue messages for subscriber1 */
     deq_option.setConsumerName("subscriber1");

     /* Dequeue a message: */
     message = queue.dequeue(deq_option);

     System.out.println("Successful dequeue");

     /* Retrieve raw data from the message: */
     raw_payload = message.getRawPayload();

     b_array = raw_payload.getBytes();

     db_conn.commit();
}
```

通知の登録

図 11-11 通知の登録



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-58 ページの「通知の登録（サブスクリプション名の指定 - シングル・コンシューマ・キュー）」も参照してください。
- 11-59 ページの「通知の登録（サブスクリプション名の指定 - マルチ・コンシューマ・キュー）」も参照してください。

用途

メッセージ通知のコールバックを登録します。

使用上の注意

- このコールは、関心があるサブスクリプション名、および起動されるように対応付けられたコールバックを識別するサブスクリプション登録に対して起動されます。一度に、複数のサブスクリプションを登録できます。
- このインタフェースは、メッセージ配信が非同期モードで行われるときのみ有効です。このモードで、サブスクライバはコールバックを指定する登録呼出しを発行します。サブスクリプションの基準に一致するメッセージが届くと、そのコールバックが起動します。次に、そのコールバックは、メッセージを取り出す明示的な `message_receive`（デキュー）を発行できます。
- ユーザーは登録の際に、`LNOCI_SUBSCR_NAMESPACE_AQ` に設定された名前空間属性によって、サブスクリプション・ハンドルを指定する必要があります。
- サブスクリプション名は、シングル・コンシューマ・キューに対する登録の場合は「`schema.queue`」という文字列で、マルチ・コンシューマ・キューに対する登録の場合は「`schema.queue:consumer_name`」になります。
- 関連する関数は、`LNOCIAQListen()`、`LNOCISubscriptionDisable()`、`LNOCISubscriptionEnable()` および `LNOCISubscriptionUnRegister()` です。

参照： OCI の通知登録操作の詳細は、『Oracle Call Interface プログラマーズ・ガイド』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）: 使用できません。
- C（OCI）:『Oracle Call Interface プログラマーズ・ガイド』の第 9 章「OCI プログラミングの高度なトピック」の「パブリッシュ・サブスクライブの通知」
- Visual Basic（OO4O オンライン・ヘルプ）:「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ Object」>「Monitoring Messages」を選択してください。
- Java（JDBC）: 使用できません。

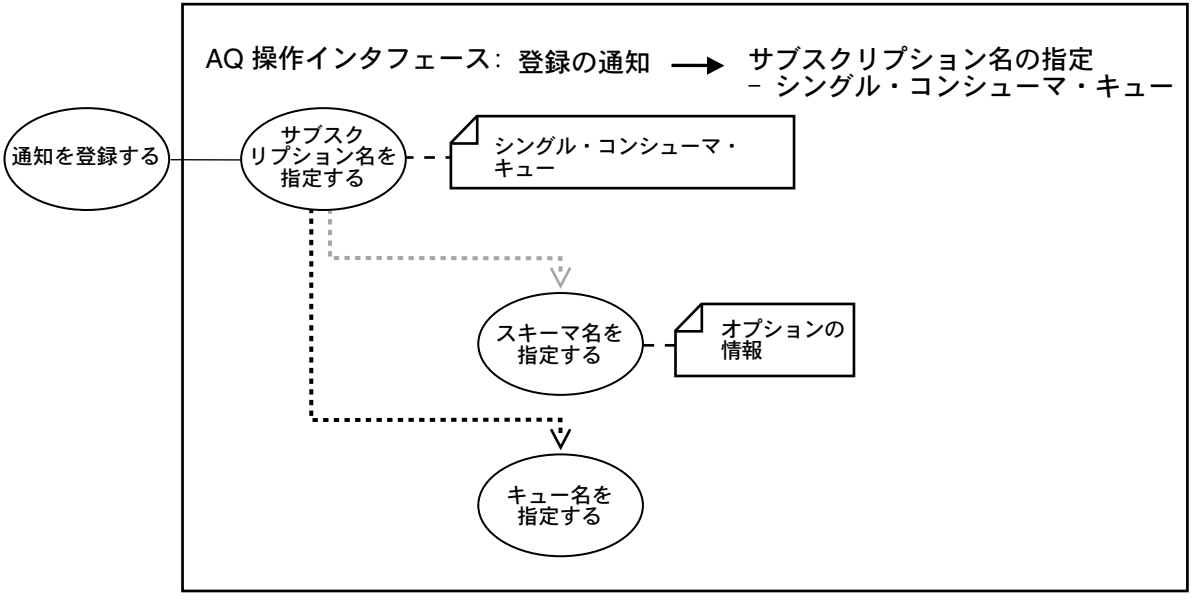
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。ここでは、次のプログラム環境の例を示します。

- C（OCI）: シングル・コンシューマおよびマルチ・コンシューマ・キューへの通知登録 （11-60 ページ）

通知の登録（サブスクリプション名の指定 - シングル・コンシューマ・キュー）

図 11-12 通知の登録（サブスクリプション名の指定 - シングル・コンシューマ・キュー）

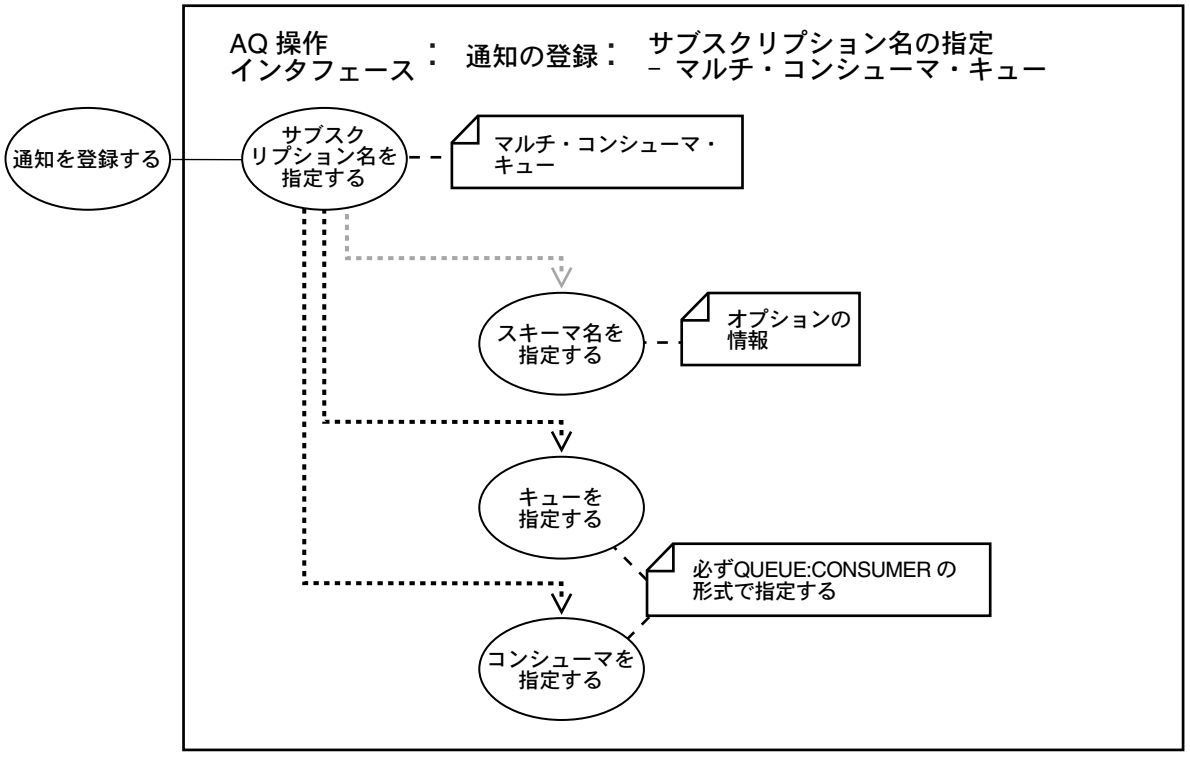


参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-55 ページの「通知の登録」も参照してください。
- 11-59 ページの「通知の登録（サブスクリプション名の指定 - マルチ・コンシューマ・キュー）」も参照してください。

通知の登録（サブスクリプション名の指定 - マルチ・コンシューマ・キュー）

図 11-13 通知の登録（サブスクリプション名の指定 - マルチ・コンシューマ・キュー）



参照：

- 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。
- 11-55 ページの「通知の登録」も参照してください。
- 11-58 ページの「通知の登録（サブスクリプション名の指定 - シングル・コンシューマ・キュー）」も参照してください。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）: 使用できません。
- C（OCI）:『Oracle Call Interface プログラマーズ・ガイド』の第 9 章「OCI プログラミングの高度なトピック」の「パブリッシュ・サブスクライブの通知」
- Visual Basic（OO4O オンライン・ヘルプ）:「Help」の「Constents」タブから、「OO4O Automation Server」>「OBJECTS」>「OraAQ Object」>「Monitoring Messages」を選択してください。
- Java（JDBC）: 使用できません。

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

C（OCI）: シングル・コンシューマおよびマルチ・コンシューマ・キューへの通知登録

```
/* OCIRegister can be used by the client to register to receive notifications
   when messages are enqueued into non-persistent and normal queues. */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>


static OCIEnv     *envhp;
static OCIServer  *srvhp;
static OCIError   *errhp;
static OCISvcCtx  *svchp;


/* The callback that gets invoked on notification */
ub4 notifyCB(ctx, subscrhp, pay, payl, desc, mode)
dvoid *ctx;
LNOCISubscription *subscrhp;      /* subscription handle */
dvoid           *pay;           /* payload  */
ub4              payl;          /* payload length */
dvoid           *desc;          /* the AQ notification descriptor */
ub4              mode;
{
```

```
 text                *subname;
 ub4                  size;
 ub4                 *number = (ub4 *)ctx;
 text                *queue;
 text                *consumer;
 OCIRaw              *msgid;
 OCIAQMsgProperties  *msgprop;

  (*number)++;

  /* Get the subscription name */
  OCIAttrGet((dvoid *)subscrhp, OCI_HTYPE_SUBSCRIPTION,
                             (dvoid *)&subname, &size,
                             OCI_ATTR_SUBSCR_NAME, errhp);
 printf("got notification number %d for %.*s  %d  \n",
         *number, size, subname, payl);

 /* Get the queue name from the AQ notify descriptor */
 OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&queue, &size,
             OCI_ATTR_QUEUE_NAME, errhp);

 /* Get the consumer name for which this notification was received */
 OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&consumer, &size,
       OCI_ATTR_CONSUMER_NAME, errhp);

 /* Get the message id of the message for which we were notified */
 OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&msgid, &size,
       OCI_ATTR_NFY_MSGID, errhp);

 /* Get the message properties of the message for which we were notified */
 OCIAttrGet(desc, OCI_DTYPE_AQNFY_DESCRIPTOR, (dvoid *)&msgprop, &size,
       OCI_ATTR_MSG_PROP, errhp);

}


int main(argc, argv)
int argc;
char *argv[];
{
  OCISession *authp = (OCISession *) 0;

  /* The subscription handles */
  OCISubscription *subscrhp[5];

  /* Registrations are for AQ namespace */
  ub4 namespace = OCI_SUBSCR_NAMESPACE_AQ;
```

```
/* The context fot the callback */
ub4 ctx[5] = {0,0,0,0,0};

printf("Initializing OCI Process\n");

/* The OCI Process Environment must be initialized  with OCI_EVENTS */
/* OCI_OBJECT flag is set to enable us dequeue */
(void) OCIInitialize((ub4) OCI_EVENTS|OCI_OBJECT, (dvoid *)0,
                     (dvoid * (*)(dvoid *, size_t)) 0,
                     (dvoid * (*)(dvoid *, dvoid *, size_t))0,
                     (void (*)(dvoid *, dvoid *)) 0 );

printf("Initialization successful\n");

/* The standard OCI setup */
printf("Initializing OCI Env\n");
(void) OCIEnvInit((OCIEnv **) &envhp, OCI_DEFAULT, (size_t) 0,
              (dvoid **) 0 );

(void) OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &errhp, OCI_HTYPE_ERROR,
                (size_t) 0, (dvoid **) 0);

/* Server contexts */
(void) OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &srvhp, OCI_HTYPE_SERVER,
                (size_t) 0, (dvoid **) 0);

(void) OCIHandleAlloc( (dvoid *) envhp, (dvoid **) &svchp, OCI_HTYPE_SVCCTX,
                (size_t) 0, (dvoid **) 0);


printf("connecting to server\n");
(void) OCIServerAttach( srvhp, errhp, (text *)"", strlen(""), 0);
printf("connect successful\n");

/* Set attribute server context in the service context */
(void) OCIAttrSet( (dvoid *) svchp, OCI_HTYPE_SVCCTX, (dvoid *)srvhp,
        (ub4) 0, OCI_ATTR_SERVER, (OCIError *) errhp);

(void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&authp,
           (ub4) OCI_HTYPE_SESSION, (size_t) 0, (dvoid **) 0);

(void) OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION,
               (dvoid *) "scott", (ub4) strlen("scott"),
               (ub4) OCI_ATTR_USERNAME, errhp);

(void) OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION,
```

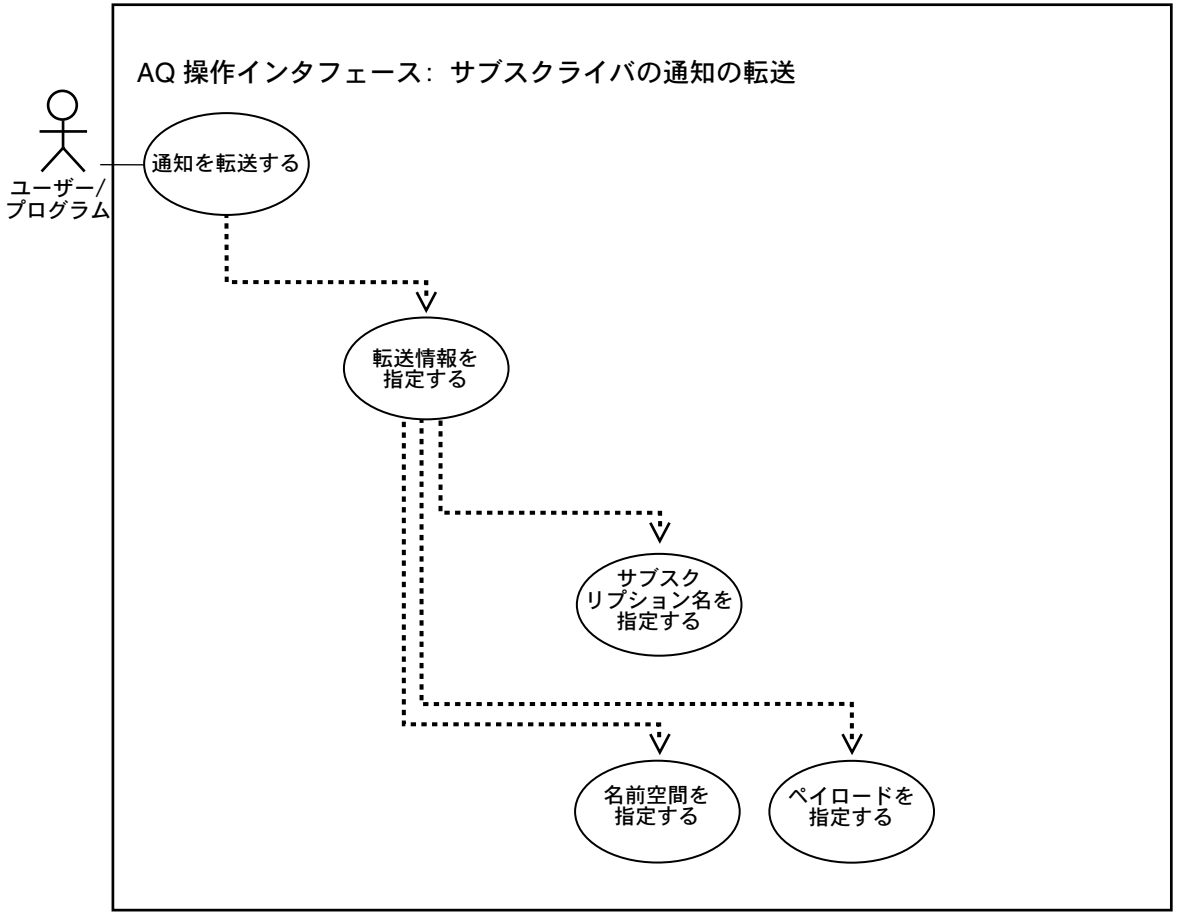
```
                (dvoid *) "tiger", (ub4) strlen("tiger"),
                (ub4) OCI_ATTR_PASSWORD, errhp);

 checkerr(errhp, OCISessionBegin ( svchp,  errhp, authp, OCI_CRED_RDBMS,
          (ub4) OCI_DEFAULT));

 (void) OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX,
                  (dvoid *) authp, (ub4) 0,
                  (ub4) OCI_ATTR_SESSION, errhp);

/* Setting the subscription handle for notification on
   a NORMAL single-consumer queue */
 printf("allocating subscription handle\n");
 subscrhp[0] = (OCISubscription *)0;
 (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&subscrhp[0],
        (ub4) OCI_HTYPE_SUBSCRIPTION,
        (size_t) 0, (dvoid **) 0);

 printf("setting subscription name\n");
 (void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
               (dvoid *) "SCOTT.SCQ1", (ub4) strlen("SCOTT.SCQ1"),
               (ub4) OCI_ATTR_SUBSCR_NAME, errhp);

 printf("setting subscription callback\n");
 (void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) notifyCB, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_CALLBACK, errhp);

printf("setting subscription context \n");
(void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *)&ctx[0], (ub4)sizeof(ctx[0]),
                (ub4) OCI_ATTR_SUBSCR_CTX, errhp);

 printf("setting subscription namespace\n");
 (void) OCIAttrSet((dvoid *) subscrhp[0], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) &namespace, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_NAMESPACE, errhp);

/* Setting the subscription handle for notification on a NORMAL multiconsumer
   consumer queue */
 subscrhp[1] = (OCISubscription *)0;
 (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&subscrhp[1],
        (ub4) OCI_HTYPE_SUBSCRIPTION,
        (size_t) 0, (dvoid **) 0);

 (void) OCIAttrSet((dvoid *) subscrhp[1], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) "SCOTT.MCQ1:APP1",
```

```
                (ub4) strlen("SCOTT.MCQ1:APP1"),
                (ub4) OCI_ATTR_SUBSCR_NAME, errhp);

 (void) OCIAttrSet((dvoid *) subscrhp[1], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) notifyCB, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_CALLBACK, errhp);

 (void) OCIAttrSet((dvoid *) subscrhp[1], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *)&ctx[1], (ub4)sizeof(ctx[1]),
                (ub4) OCI_ATTR_SUBSCR_CTX, errhp);

 (void) OCIAttrSet((dvoid *) subscrhp[1], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) &namespace, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_NAMESPACE, errhp);


/* Setting the subscription handle for notification on a non-persistent
  single-consumer queue */
 subscrhp[2] = (OCISubscription *)0;
 (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&subscrhp[2],
        (ub4) OCI_HTYPE_SUBSCRIPTION,
        (size_t) 0, (dvoid **) 0);

 (void) OCIAttrSet((dvoid *) subscrhp[2], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) "SCOTT.NP_SCQ1",
                (ub4) strlen("SCOTT.NP_SCQ1"),
                (ub4) OCI_ATTR_SUBSCR_NAME, errhp);

 (void) OCIAttrSet((dvoid *) subscrhp[2], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) notifyCB, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_CALLBACK, errhp);

 (void) OCIAttrSet((dvoid *) subscrhp[2], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *)&ctx[2], (ub4)sizeof(ctx[2]),
                (ub4) OCI_ATTR_SUBSCR_CTX, errhp);

 (void) OCIAttrSet((dvoid *) subscrhp[2], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) &namespace, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_NAMESPACE, errhp);


/* Setting the subscription handle for notification on
   a non-persistent multi consumer queue */
/* Waiting on user specified recipient */
 subscrhp[3] = (OCISubscription *)0;
 (void) OCIHandleAlloc((dvoid *) envhp, (dvoid **)&subscrhp[3],
        (ub4) OCI_HTYPE_SUBSCRIPTION,
```

```
        (size_t) 0, (dvoid **) 0);

  (void) OCIAttrSet((dvoid *) subscrhp[3], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) "SCOTT.NP_MCQ1",
                (ub4) strlen("SCOTT.NP_MCQ1"),
                (ub4) OCI_ATTR_SUBSCR_NAME, errhp);

  (void) OCIAttrSet((dvoid *) subscrhp[3], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) notifyCB, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_CALLBACK, errhp);

 (void) OCIAttrSet((dvoid *) subscrhp[3], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *)&ctx[3], (ub4)sizeof(ctx[3]),
                (ub4) OCI_ATTR_SUBSCR_CTX, errhp);

  (void) OCIAttrSet((dvoid *) subscrhp[3], (ub4) OCI_HTYPE_SUBSCRIPTION,
                (dvoid *) &namespace, (ub4) 0,
                (ub4) OCI_ATTR_SUBSCR_NAMESPACE, errhp);


  printf("Registering for all the subscriptiosn \n");
  checkerr(errhp, OCISubscriptionRegister(svchp, subscrhp, 4, errhp,
                OCI_DEFAULT));

  printf("Waiting for notifcations \n");

  /* wait for minutes for notifications */
  sleep(300);

  printf("Exiting\n");
}
```

サブスクライバの通知の転送

図 11-14 サブスクライバの通知の転送



参照： 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。

用途

匿名サブスクリプションのリストに通知を転送し、サブスクリプションに登録されているユーザーが通知を取得するようにします。

使用上の注意

一度に、複数のサブスクリプションを転送できます。サブスクリプションの転送時には、サブスクリプション名、および必要に応じてペイロードが識別されます。このコールにペイロードを対応付けなくてもかまいません。このコールは、最も効果的な転送を保証します。通知は、登録されたクライアントにすぐに届きます。

このコールは主に軽量な通知に使用され、いくつかのシステム・イベントに対して有効です。アプリケーションがより確実な保証を必要とする場合は、キューをエンキューすることによって AQ 機能を使用できます。

OCI を使用する場合、登録時に、OCI_SUBSCR_NAMESPACE_ANONYMOUS に設定された名前空間属性によって、サブスクリプション・ハンドルを指定する必要があります。

PL/SQL を使用する場合は、aq$_post_info の名前空間属性を DBMS_AQ.NAMESPACE_ANONYMOUS に設定する必要があります。

関連する関数は、LNOCIAQListen()、OCISvcCtxToLda()、LNOCISubscriptionEnable()、OCISubscriptionRegister()、LNOCISubscriptionUnRegister()、dbms_aq.register および dbms_aq.unregister です。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の POST プロシージャ
- C（OCI）:『Oracle Call Interface プログラマーズ・ガイド』の第 9 章「OCI プログラミングの高度なトピック」の「パブリッシュ・サブスクライブの通知」
- Visual Basic（OO4O オンライン・ヘルプ）: 使用できません。
- Java（JDBC）: 使用できません。

例

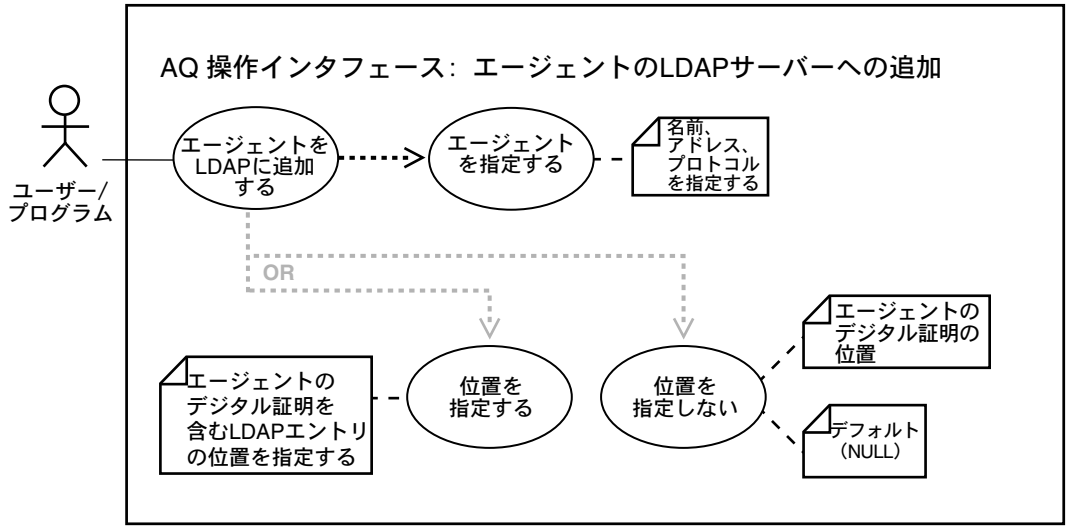
各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

PL/SQL（DBMS_AQ）：オブジェクト型メッセージの転送

```
-- Register for notification
DECLARE
  reginfo             sys.aq$_reg_info;
  reginfolist         sys.aq$_reg_info_list;

BEGIN
  -- Register for anonymous subscription PUBSUB1.ANONSTR, consumer_name ADMIN
  -- The PL/SQL callback pubsub1.mycallbk will be invoked
  -- when a notification is received
  reginfo := sys.aq$_reg_info('PUBSUB1.ANONSTR:ADMIN',
     DBMS_AQ.NAMESPACE_ANONYMOUS,
      'plsql://PUBSUB1.mycallbk', HEXTORAW('FF'));

  reginfolist := sys.aq$_reg_info_list(reginfo);

  sys.dbms_aq.register(reginfolist, 1);

  commit;
END;
/

-- Post to an anonymous subscription
DECLARE

  postinfo            sys.aq$_post_info;
  postinfolist        sys.aq$_post_info_list;

BEGIN

  -- Post to the anonymous subscription PUBSUB1.ANONSTR, consumer_name ADMIN
  postinfo := sys.aq$_post_info('PUBSUB1.ANONSTR:ADMIN',0,HEXTORAW('FF'));
  postinfolist := sys.aq$_post_info_list(postinfo);

  sys.dbms_aq.post(postinfolist, 1);

  commit;

END;
/
```

エージェントの LDAP サーバーへの追加

図 11-15 エージェントの LDAP への追加



参照: 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。

用途

エージェントを LDAP サーバーに追加します。

使用上の注意

このコールは、エージェントおよびオプションで証明書の位置を引数としてとり、エージェントのエントリを LDAP サーバーに追加します。この証明書の位置パラメータは、エージェントが使用するデジタル証明を含む LDAP エントリの識別名です。エージェントがデジタル証明を持たない場合は、このパラメータはデフォルトで NULL に設定されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

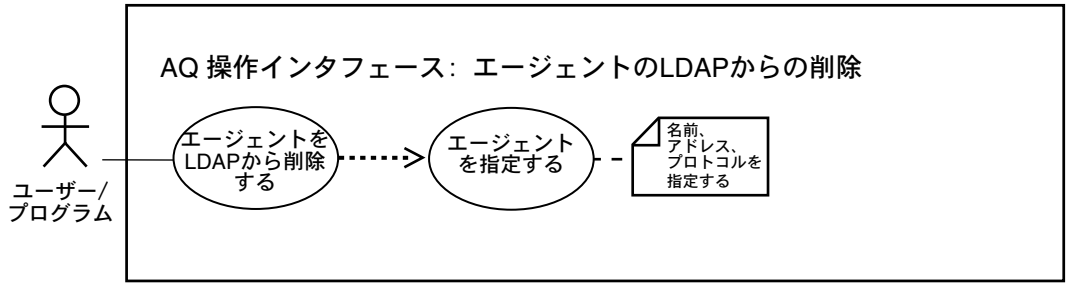
- PL/SQL（DBMS_AQ）：『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の BIND_AGENT プロシージャ
- C（OCI）：『Oracle Call Interface プログラマーズ・ガイド』の第 9 章「OCI プログラミングの高度なトピック」
- Visual Basic（OO4O オンライン・ヘルプ）：使用できません。
- Java（JDBC）：使用できません。

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

エージェントの LDAP サーバーからの削除

図 11-16 エージェントの LDAP からの削除



参照： 操作インタフェースの基本操作の詳細は、表 11-1 を参照してください。

用途

エージェントを LDAP サーバーから削除します。

使用上の注意

このコールは、エージェントを引数としてとり、LDAP サーバー内の対応するエージェントのエントリを削除します。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- PL/SQL（DBMS_AQ）:『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の第 5 章「DBMS_AQ」の UNBIND_AGENT プロシージャ
- C（OCI）:『Oracle Call Interface プログラマーズ・ガイド』の第 9 章「OCI プログラミングの高度なトピック」
- Visual Basic（OO4O オンライン・ヘルプ）: 使用できません。
- Java（JDBC）: 使用できません。

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

12

JMS を使用したアプリケーションの作成

第 1 章「Oracle Advanced Queuing の概要」では、架空の会社 BooksOnLine を基にメッセージ・システムを説明しました。この章では、シナリオに基づいたサンプル・アプリケーションに沿って、AQ の機能について説明します。内容は次のとおりです。

- JMS を使用したサンプル・アプリケーション
- JMS の一般的な機能
- JMS での Point-to-Point モデル機能
- JMS パブリッシュ・サブスクライブ・モデル機能
- JMS メッセージ・プロデューサ機能
- メッセージ・コンシューマ機能
- JMS 伝播
- JMS AQ のメッセージ変換

JMS を使用したサンプル・アプリケーション

大型の書籍販売店である BooksOnLine の運営は、販売プロセス全体にかかわる様々な部署にまたがるアクティビティを自動化する、オンライン書籍発注システムが基礎となっています。システムのフロント・エンドは、新しい発注を入力するための注文入力アプリケーションです。入力された注文は、注文処理アプリケーションによって妥当性チェックおよび記録が行われます。地域倉庫に置かれた出荷部門は、発注された書籍をすぐに確実に発送する責任があります。地域倉庫は 3 箇所で、それぞれが東部地域、西部地域および海外からの発注に対応しています。発注された書籍の発送完了後、発注情報は支払処理のために中央の請求部門に送られます。各地の顧問サービス部門は、発注情報の状況を管理し、発注に関する問合せを処理する責任があります。

第 1 章「Oracle Advanced Queuing の概要」では、架空の会社 BooksOnLine を基にメッセージ・システムの概要を示しました。この章では、そのシナリオに基づいたサンプル・アプリケーションを使用して、AQ への JMS インタフェースの機能について検討します。このサンプル・アプリケーションは、Oracle AQ の機能を説明する目的でのみ作成されたものです。この統合化シナリオを作成するにあたって、単一の使用方法に絞った説明をすることで、このテクノロジの可能性を理解しやすくするよう努めました。ただし、考えられる AQ のすべての応用例を、比較的小さい単一のコード・サンプルの範囲内で示すことは不可能なことを理解しておいてください。

JMS の一般的な機能

この項の内容は次のとおりです。

- J2EE 準拠
- JMS コネクションおよびセッション
- JMS 宛先 - キューおよびトピック
- JMS でのシステム・レベルのアクセス制御
- JMS での宛先レベルのアクセス制御
- JMS での保存およびメッセージ履歴
- JMS での Oracle Real Application Clusters のサポート
- JMS での統計ビューのサポート
- JMS での構造化ペイロード / メッセージの型

J2EE 準拠

Oracle9*i* リリース 2（9.2）の Oracle JMS は、Sun 社の JMS 1.0.2b 標準に準拠しています。実行時に、OJMS クライアントに対して J2EE 準拠モードを定義できます。準拠モードにするには、コマンドライン・オプションとして Java プロパティ `oracle.jms.j2eeCompliant` を `TRUE` に設定します。非準拠モードにするには、処理は必要はありません。デフォルトは `FALSE` です。

Oracle9*i* リリース 2（9.2）の新機能は、J2EE 準拠をサポートしていますが、非準拠モードでも使用できます。これらの新機能では、次のものがサポートされます。

- 非トランザクション処理のセッション
- 非永続サブスクライバ
- 一時キューおよびトピック
- 非永続配信モード
- (`AQ$_JMS_MESSAGE` 型の AQ キューを使用した)1 つの JMS キューまたはトピックに対する複数の JMS メッセージ型
- 永続サブスクライバに対する `noLocal` オプション

> **参照：** Sun 社の『Java Message Service Specification, version 1.0.2b』を参照してください。

`JMSPriority`、`JMSExpiration` および非永続サブスクライバの機能は、使用するモードによって異なります。

JMSPriority

`JMSPriority` の値は、デフォルトの非準拠モードで実行するか、または準拠フラグを `TRUE` に設定して準拠モードで実行するかによって異なります。

- 非準拠モードでは、`java.lang.Integer.MAX_VALUE` が最も低い優先順位で、`java.lang.Integer.MIN_VALUE` が最も高い優先順位です。デフォルトの優先順位は `1` です。
- 準拠モードでは、`0` が最も低い優先順位で、`9` が最も高い優先順位です。デフォルトの優先順位は `4` です。

JMSExpiration

JMSExpiration 値は、デフォルトの非準拠モードで実行するか、または準拠フラグを TRUE に設定して準拠モードで実行するかによって異なります。

- 非準拠モードでは、JMSExpiration ヘッダー値は、メッセージのエンキュー時に JMS 仕様で指定されたエンキュー時間と Time-To-Live の合計になります。メッセージが受信されると、（期限切れ時間ではなく）期限切れの存続期間が戻されます。メッセージに期限がない場合、-1 が戻されます。
- 準拠モードでは、デキューされたメッセージの JMSExpiration ヘッダー値は、メッセージがエンキューされたときの JMS タイムスタンプ（グリニッジ標準時で 1000 分の 1 秒単位）と Time-To-Live（1000 分の 1 秒単位）の合計になります。メッセージに期限がない場合、0 が戻されます。

永続サブスクライバ

永続サブスクライバが同じ名前を使用するときの動作は、デフォルトの非準拠モードで実行するか、または準拠フラグを TRUE に設定して準拠モードで実行するかによって異なります。

- 非準拠モードでは、同じ名前の 2 つの永続的な TopicSubscriber が、2 つの異なるトピックに対してアクティブになることができます。
- 準拠モードでは、複数の永続サブスクライバが同じ名前を持つことは許可されません。次のケースが発生する可能性があります。

  ケース 1 - 同じトピックに対して作成された 2 つのサブスクライバが同じ名前を使用する場合、各サブスクライバに使用されるセレクタが異なると、DBMS_AQJMS.ALTER_SUBSCRIBER() の内部コールを使用して、基になる AQ サブスクリプションが変更されます。

  ケース 2 - 2 つの異なるトピックに対して作成された 2 つのサブスクライバが同じ名前を使用する場合 :

  - 最初に作成されたサブスクリプションと同じサブスクリプション名を同じクライアントが使用する場合、既存のサブスクリプションは削除され、新しいサブスクリプションが作成されます。
  - 別の（最初にそのサブスクリプション名を作成したクライアントではない）クライアントが既存のサブスクリプション名を使用する場合、そのサブスクリプションは削除されず、エラーが発生します。サブスクリプションが JMS または PL/SQL のどちらで作成されたかが不明であるため、その他のトピックについてのサブスクリプションは削除しないでください。

JMS コネクションおよびセッション

コネクション・ファクトリ

ConnectionFactory は、管理者によって定義された接続構成パラメータの集合をカプセル化します。クライアントは、これを使用して JMS プロバイダとの**コネクション**を確立します。Oracle JMS では、Oracle9*i* が JMS プロバイダです。

ConnectionFactory オブジェクトには、次の 2 種類があります。

- QueueConnectionFactory
- TopicConnectionFactory

ConnectionFactory オブジェクトを取得するには、次のいずれかの方法を使用します。

1. AQjmsFactory の static メソッドで取得する。
2. Java Naming and Directory Interface（JNDI）検索で、LDAP ディレクトリ・サーバーから取得する。

AQjmsFactory を使用した ConnectionFactory オブジェクトの取得

AQjmsFactory クラスを使用すると、QueueConnectionFactory または TopicConnectionFactory オブジェクトに対するハンドルを取得できます。

- QueueConnectionFactory を取得するには、AQjmsFactory.getQueueConnectionFactory() メソッドを使用します。

  キュー・コネクション・ファクトリは、ホスト名、ポート番号、SID を使用するか、JDBC URL およびプロパティを使用して作成できます。

- TopicConnectionFactory を取得するには、AQjmsFactory.getTopicConnectionFactory() メソッドを使用します。

  トピック・コネクション・ファクトリは、ホスト名、ポート番号、SID を使用するか、JDBC URL およびプロパティを使用して作成できます。

例

```
public static void get_Factory() throws JMSException
{
  QueueConnectionFactory    qc_fact   = null;
 /* get queue connection factory for database "aqdb", host "sun-123", port     5521,
driver "thin" */
 qc_fact = AQjmsFactory.getQueueConnectionFactory("sun-123", "aqdb", 5521,
                                                  "thin");
}
```

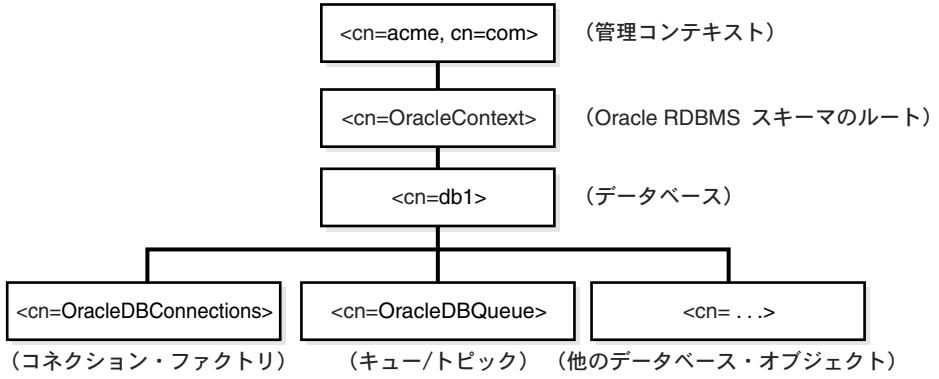
JNDI を使用した ConnectionFactory オブジェクトの検索

JMS 管理者は、LDAP サーバーに ConnectionFactory オブジェクトを登録できます。

JMS で JNDI 検索を使用可能にするには、次のとおり設定する必要があります。

1. Oracle9*i* サーバーのインストール時に、データベースを LDAP サーバーに登録する必要があります。これを行うには、Database Configuration Assistant（DBCA）を使用します。

   LDAP サーバーの AQ エントリの構造を次に示します。



   コネクション・ファクトリ情報は、<cn=OracleDBConnections> に格納されています。トピックおよびキューは、<cn=OracleDBQueues> に格納されています。

2. データベースの GLOBAL_TOPIC_ENABLED システム・パラメータを TRUE に設定する必要があります。これによって、AQ で作成されたすべてのキューおよびトピックが自動的に LDAP サーバーに登録されます。

   このパラメータは、次のようにして設定できます。

   ```
   ALTER SYSTEM SET GLOBAL_TOPICS_ENABLED = TRUE
   ```

3. LDAP サーバーを使用するようにデータベースを設定すると、JMS 管理者は、AQjmsFactory.registerConnectionFactory() メソッドを使用して、LDAP に QueueConnectionFactory オブジェクトおよび TopicConnectionFactory オブジェクトを登録できます。

   登録には、次のいずれかの方法を使用します。

   - LDAP サーバーに直接接続します。LDAP にコネクション・ファクトリを登録するには、ユーザーに GLOBAL_AQ_USER_ROLE が必要です。

     LDAP に直接接続するには、registerConnectionFactory メソッドのパラメータに、LDAP コンテキスト、QueueConnectionFactory/TopicConnectionFactory の名前、ホスト名、

データベース SID、ポート番号、（Thin または OCI8）JDBC ドライバおよびファクトリ・タイプ（キューまたはトピック）を含める必要があります。

- データベース・サーバー経由で LDAP に接続します。ユーザーは、まず、Oracle9*i* データーベースにログインして、データベースによって LADP エントリを更新します。ユーザーが、データベースにログインしてこの操作を実行するには、AQ_ADMINISTRATOR_ROLE を取得する必要があります。

  データベース・サーバーを経由して LDAP に直接接続するには、registerConnectionFactory メソッドのパラメータに、（AQ_ADMINISTRATOR_ROLE を取得しているユーザーとの）JDBC コネクション、QueueConnectionFactory/TopicConnectionFactory の名前、ホスト名、データベース SID、ポート番号、（Thin または OCI8）JDBC ドライバおよびファクトリ・タイプ（キューまたはトピック）を含める必要があります。

JMS 管理者が LDAP に ConnectionFactory オブジェクトを登録すると、JNDI を使用してそのオブジェクトを検索できます。

例

JMS 管理者が、注文入力キュー・コネクション・ファクトリ oe_queue_factory を登録すると想定します。これは、次のとおり LDAP に登録されます。

```
public static void register_Factory_in_LDAP() throws Exception
{
    Hashtable env = new Hashtable(5, 0.75f);
    env.put(Context.INITIAL_CONTEXT_FACTORY, AQjmsConstants.INIT_CTX_FACTORY);

    // aqldapserv is your LDAP host and 389 is your port
    env.put(Context.PROVIDER_URL, "ldap://aqldapserv:389);

    // now authentication info
    // username/password scheme, user is OE, password is OE
    env.put(Context.SECURITY_AUTHENTICATION, "simple");
    env.put(Context.SECURITY_PRINCIPAL, "cn=oe,cn=users,cn=acme,cn=com");
    env.put(Context.SECURITY_CREDENTIALS, "oe");

    /* register queue connection factory for database "aqdb", host "sun-123",
       port 5521,  driver "thin" */
    AQjmsFactory.registerConnectionFactory(env, "oe_queue_factory", "sun-123",
                                           "aqdb", 5521, "thin", "queue");
   }
```

注文を入力すると、キュー・コネクション・ファクトリ oe_queue_factory が LDAP に登録されます。このファクトリは、次のとおり検索できます。

```
public static void get_Factory_from_LDAP() throws Exception
{
    Hashtable env = new Hashtable(5, 0.75f);
    env.put(Context.INITIAL_CONTEXT_FACTORY, AQjmsConstants.INIT_CTX_FACTORY);

    // aqldapserv is your LDAP host and 389 is your port
    env.put(Context.PROVIDER_URL, "ldap://aqldapserv:389);

    // now authentication info
    // username/password scheme, user is OE, password is OE
    env.put(Context.SECURITY_AUTHENTICATION, "simple");
    env.put(Context.SECURITY_PRINCIPAL, "cn=oe,cn=users,cn=acme,cn=com");
    env.put(Context.SECURITY_CREDENTIALS, "oe");

    DirContext inictx = new InitialDirContext(env);
    // initialize context with the distinguished name of the database server
    inictx=(DirContext)inictx.lookup("cn=db1,cn=OracleContext,cn=acme,cn=com");

    //go to the connection factory holder cn=OraclDBConnections
    DirContext connctx = (DirContext)inictx.lookup("cn=OracleDBConnections");

    // get connection factory "oe_queue_factory"
    QueueConnectionFactory qc_fact =
                (QueueConnectionFactory)connctx.lookup("cn=oe_queue_factory");
}
```

コネクション

JMS コネクションは、クライアントによる JMS プロバイダへのアクティブなコネクションを表します。コネクションによって、次に示すいくつかの重要なサービスが実行されます。

- JMS プロバイダとのオープン・コネクションまたはコネクション・プールのいずれかをカプセル化します。
- クライアントとプロバイダのサービス・デーモン間のオープンな TCP/IP ソケットを表します。
- コネクション確立時に、クライアントを認証するための構造を提供します。
- セッションを作成します。
- コネクション・メタデータを提供します。
- オプションの ExceptionListener をサポートします。

データベースへの JMS コネクションを確立するには、createQueueConnection() または createTopicConnection() を起動して、QueueConnectionFactory オブジェクトおよび TopicConnectionFactory オブジェクトに、それぞれパラメータのユーザー名とパスワードを渡します。

コネクションの設定

JMS クライアントは、1 つのコネクション、1 つのセッション、および多数のメッセージ・プロデューサとメッセージ・コンシューマを作成します。現在のバージョンでは、次の場合を除いて、1 つのコネクションにつき 1 つのオープン・セッションのみが許可されています。

- JDBC OCI8 ドライバを使用して JMS コネクションを作成する場合
- ユーザーが、JMS コネクション作成時に、OracleOCIConnectionPool インスタンスを提供する場合

コネクションは、確立時は停止モードです。この状態では、メッセージをコネクションに配信できません。通常、設定が完了するまで、コネクションは停止モードのままです。設定が完了した時点で、コネクションの start() メソッドがコールされ、メッセージがコネクション・コンシューマに届き始めます。この設定規則によって、設定処理中であるにもかかわらずメッセージが非同期配信されることによって発生するクライアントの混乱を、最小限に抑えることができます。

コネクションを開始してから設定を行うことは可能ですが、これを行うクライアントは、設定処理中でもメッセージの非同期配信を処理できるように準備しておく必要があります。メッセージ・プロデューサは、コネクションが停止中でもメッセージを送信できます。

次に、コネクション・オブジェクトでサポートされているメソッドのいくつかを示します。

- start() - コネクションによる受信メッセージの配信を開始または再開します。
- stop() - コネクションによる受信メッセージの配信を一時的に停止します。配信が停止されると、コネクションのすべてのメッセージ・コンシューマへの配信が禁止されます。また、同期受信のブロックおよびメッセージは、メッセージ・リスナーに配信されません。
- close() - JMS セッションを閉じて、対応付けられたすべてのリソースを解放します。
- createQueueSession(true,0) - キュー・セッションを作成します。
- createTopicSession(true,0) - トピック・セッションを作成します。
- setExceptionListener(ExceptionListener) - コネクション用の例外リスナーを設定します。これによって、問題がクライアントに非同期に通知されます。メッセージを処理するのみのコネクションもあるため、コネクションが失敗したことを知るための他の方法がありません。
- getExceptionListener() - このコネクション用の ExceptionListener を取得します。

セッション

コネクションは、JMS プロバイダへの基本コネクションを使用してメッセージを作成および処理するセッションのファクトリです。JMS セッションは、メッセージの作成および処理用の単一スレッドのコンテキストです。これは、Java Virtual Machine（JVM）の外でプロバイダ・リソースを割り当てますが、軽量な JMS オブジェクトとみなされます。

セッションには、次の役割があります。

- メッセージ・プロデューサおよびメッセージ・コンシューマのファクトリを構成します。
- 宛先オブジェクト（キュー / トピック）に対するハンドルの取得方法を提供します。
- プロバイダが最適化したメッセージ・ファクトリを提供します。
- このセッションのプロデューサおよびコンシューマにまたがる作業を組み合せる一連のトランザクションをサポートし、これらを基本単位に編成します。
- セッションが処理および作成するメッセージのシリアル順序を定義します。
- セッションに登録されたメッセージ・リスナーの実行をシリアル化します。

OCI JDBC ドライバの使用時は、コネクションごとに複数のセッションを作成できます。その他の JDBC ドライバの使用時は、1 つのコネクションから作成できるのは 1 つのセッションのみです。

プロバイダが、JVM 外のセッションにかわってリソースを割り当てる場合があるため、クライアントは、リソースが必要ないときは、これらを閉じる必要があります。ガベージ・コレクションによる最終的なリソースの解放を待つ必要はありません。セッションが作成したメッセージ・プロデューサおよびメッセージ・コンシューマについても、同じことがいえます。

セッション・オブジェクトのメソッドには、次のものが含まれます。

- `commit()` - このトランザクションで実行されるすべてのメッセージをコミットして、現在保持されているロックを解放します。
- `rollback()` - トランザクションで実行されたすべてのメッセージをロールバックして、現在保持されているロックを解放します。
- `close()` - セッションを閉じます。
- `getDBConnection()` - 基礎となる JDBC コネクションに対するハンドルを取得します。このハンドルを使用して、他の SQL DML 操作を同じセッションの一部として実行できます。このメソッドは、Oracle JMS 固有のメソッドです。
- `acknowledge()` - 非トランザクション処理のセッションでメッセージの受信を認識します。
- `recover()` - 非トランザクション処理のセッションでメッセージの配信を再開します。セッション中に配信された一連のメッセージは、最後に認識されたメッセージの後の時点にリセットされます。

次に、JMS に対する Oracle の拡張機能をいくつか示します。どの拡張機能を使用する場合も、セッション・オブジェクトを AQjmsSession にキャストする必要があります。

- キュー表、キューおよびトピックは、セッション・オブジェクトから作成できます。
- createQueueTable() - キュー表を作成します。
- getQueueTable() - 既存のキュー表に対するハンドルを取得します。
- createQueue() - キューを作成します。
- getQueue() - 既存のキューに対するハンドルを取得します。
- createTopic() - トピックを作成します。
- getTopic() - 既存のトピックに対するハンドルを取得します。

次に、前述のコールの使用方法を示します。

サンプル・コード

```
public static void bol_example(String ora_sid, String host, int port,
                               String driver)
{

 QueueConnectionFactory    qc_fact   = null;
 QueueConnection           q_conn    = null;
 QueueSession              q_sess    = null;
 AQQueueTableProperty      qt_prop   = null;
 AQQueueTable              q_table   = null;
 AQjmsDestinationProperty  dest_prop = null;
 Queue                     queue     = null;
 BytesMessage              bytes_msg = null;

 try
 {
   /* get queue connection factory */
   qc_fact = AQjmsFactory.getQueueConnectionFactory(host, ora_sid,
                     port, driver);

   /* create queue connection */
   q_conn = qc_fact.createQueueConnection("boluser", "boluser");

   /* create queue session */
   q_sess = q_conn.createQueueSession(true, Session.CLIENT_ACKNOWLEDGE);

   /* start the queue connection */
   q_conn.start();

   qt_prop = new AQQueueTableProperty("SYS.AQ$_JMS_BYTES_MESSAGE");
```

```
    /* create a queue table */
    q_table = ((AQjmsSession)q_sess).createQueueTable("boluser",
                                                      "bol_ship_queue_table",
                                                      qt_prop);

    dest_prop = new AQjmsDestinationProperty();

    /* create a queue */
    queue = ((AQjmsSession)q_sess).createQueue(q_table, "bol_ship_queue",
                                               dest_prop);

    /* start the queue */
    ((AQjmsDestination)queue).start(q_sess, true, true);

    /* create a bytes message */
    bytes_msg = q_sess.createBytesMessage();

    /* close session */
    q_sess.close();

    /* close connection */
    q_conn.close();
  }
  catch (Exception ex)
  {
    System.out.println("Exception: " + ex);
  }
}
```

JMS 宛先 - キューおよびトピック

宛先は、クライアントがメッセージの送信先および受信元の指定に使用するオブジェクトです。

宛先オブジェクトには、キューおよびトピックの 2 種類があります。AQ では、これらは特定のデータベースで <schema>.<queue> にマップします。キューは AQ のシングル・コンシューマ・キューに、トピックは AQ のマルチ・コンシューマ・キューにマップします。

宛先オブジェクトを取得するには、次のいずれかの方法を使用します。

1. JMS セッションでドメイン固有のメソッドを使用します。
2. LDAP ディレクトリ・サーバーから Java Naming and Directory Interface（JNDI）検索を行います。

JMS セッションを使用した宛先オブジェクトの取得

宛先オブジェクトは、セッション・オブジェクトから、ドメイン固有のセッション・メソッドを使用して作成されます。

- `AQjmsSession.getQueue(queue_owner, queue_name)` - JMS キューに対するハンドルを取得します。
- `AQjmsSession.getTopic(topic_owner, topic_name)` - JMS トピックに対するハンドルを取得します。

サンプル・コード

BooksOnLine アプリケーションでは、新規の注文が OE スキーマの `neworders_queue` に送信されます。JMS コネクションおよびセッションを作成すると、次のようにキューに対するハンドルを取得できます。

```
public Queue get_queue_example(QueueSession jms_session)
{
   QueueSender   sender;
   Queue         queue = null;

   try
   {
       /* get a handle to the OE.oe_new_orders queue */
       queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");
   }
   catch (JMSException ex){
      System.out.println("Exception: " + ex); }
   return queue;
}
```

JNDI を使用した宛先オブジェクトの検索

12-5 ページの「コネクション・ファクトリ」に示すとおり、LDAP サーバーにスキーマ・オブジェクトを登録するようにデータベースを設定できます。データベースが LDAP を使用できるように設定され、`GLOBAL_TOPIC_ENABLED` パラメータが `TRUE` に設定されている場合、すべての JMS キューおよびトピックは、作成時に自動的に LDAP サーバーに登録されます。

また、管理者は、PL/SQL プロシージャ `DBMS_AQAQDM.add_alias_to_ldap` を使用して、LDAP に登録されているキューおよびトピックの別名を作成することもできます。

LDAP に登録されているキューおよびトピックは、キュー / トピック名またはその別名の 1 つを使用して、JNDI を介して検索できます。

サンプル・コード

LDAP に新しい注文キュー queue OE.OE_neworders_que が格納されていると想定します。次のとおりに検索できます。

```
public static void get_Factory_from_LDAP() throws Exception
{
    Hashtable env = new Hashtable(5, 0.75f);
    env.put(Context.INITIAL_CONTEXT_FACTORY, AQjmsConstants.INIT_CTX_FACTORY);

    // aqldapserv is your LDAP host and 389 is your port
    env.put(Context.PROVIDER_URL, "ldap://aqldapserv:389);

    // now authentication info
    // username/password scheme, user is OE, password is OE
    env.put(Context.SECURITY_AUTHENTICATION, "simple");
    env.put(Context.SECURITY_PRINCIPAL, "cn=oe,cn=users,cn=acme,cn=com");
    env.put(Context.SECURITY_CREDENTIALS, "oe");

    DirContext inictx = new InitialDirContext(env);
    // initialize context with the distinguished name of the database server
    inictx=(DirContext)inictx.lookup("cn=db1,cn=OracleContext,cn=acme,cn=com");

   // go to the destination holder
    DirContext destctx = (DirContext)inictx.lookup("cn=OracleDBQueues");

    // get the destination OE.OE_new_orders queue
    Queue myqueue = (Queue)destctx.lookup("cn=OE.OE_new_orders_que");

}
```

宛先オブジェクトのメソッドには、次のものが含まれます。

- alter() - キューまたはトピックを変更します。
- schedulePropagation() - ソースから宛先への伝播をスケジューリングします。
- unschedulePropagation() - 設定されていた伝播スケジュールを解除します。
- enablePropagationSchedule() - 伝播スケジュールを使用可能にします。
- disablePropagationSchedule() - 伝播スケジュールを使用不可能にします。
- start() - キューまたはトピックを開始します。キューは、エンキューまたはデキューを可能にするために開始されます。トピックは、パブリッシュまたはサブスクライブを可能にするために開始できます。
- stop() - キューまたはトピックを停止します。キューは、エンキューまたはデキューを実行不可にするために停止できます。トピックは、パブリッシュまたはサブスクライブを実行不可にするために停止できます。

- drop() - キューまたはトピックを削除します。

サンプル・コード

```
public static void setup_example(TopicSession t_sess)
{
  AQQueueTableProperty     qt_prop   = null;
  AQQueueTable             q_table   = null;
  AQjmsDestinationProperty dest_prop = null;
  Topic                    topic     = null;
  TopicConnection          t_conn    = null;

  try
  {
    qt_prop = new AQQueueTableProperty("SYS.AQ$_JMS_BYTES_MESSAGE");
    /* create a queue table */
    q_table = ((AQjmsSession)t_sess).createQueueTable("boluser",
                          "bol_ship_queue_table",
                          qt_prop);
    dest_prop = new AQjmsDestinationProperty();
    /* create a topic */
    topic = ((AQjmsSession)t_sess).createTopic(q_table, "bol_ship_queue",
                        dest_prop);

    /* start the topic */
    ((AQjmsDestination)topic).start(t_sess, true, true);

    /* schedule propagation from topic "boluser" to the destination
      dblink "dba" */
    ((AQjmsDestination)topic).schedulePropagation(t_sess, "dba", null,
                  null, null, null);
    /*
       some processing done here
    */
    /* Unschedule propagation */
    ((AQjmsDestination)topic).unschedulePropagation(t_sess, "dba");
    /* stop the topic */
    ((AQjmsDestination)topic).stop(t_sess, true, true, true);
    /* drop topic */
    ((AQjmsDestination)topic).drop(t_sess);
    /* drop queue table */
    q_table.drop(true);
    /* close session */
    t_sess.close();
    /* close connection */
    t_conn.close();
  }
```

```
  catch(Exception ex)
  {
    System.out.println("Exception: " + ex);
  }
}
```

JMS でのシステム・レベルのアクセス制御

Oracle9*i* は、すべてのキューイング操作に対してシステム・レベルのアクセス制御をサポートします。この機能によって、アプリケーション設計者または DBA は、ユーザーをキュー管理者にできます。キュー / トピック管理者は、データベース上のどのキューに対しても、すべての JMS インタフェース（管理および操作）を起動できます。これによって、データベース上のキュー全体に対するすべての管理スクリプトを 1 つのスキーマで管理できるため、管理作業が簡単になります。詳細は、4-8 ページの「Oracle Enterprise Manager のサポート」を参照してください。

サンプル・シナリオおよびコード

BooksOnLine アプリケーションでは、DBA がデータベースのキュー管理者として BOLADM（BooksOnLine の管理者アカウント）を作成します。これによって、BOLADM はデータベース上のすべてのキューを作成、削除、管理および監視できるようになります。どのアプリケーションからでも BOLADM スキーマにエンキューまたはデキューできるように PL/SQL パッケージを作成する場合は、BOLADM にシステム権限 ENQUEUE_ANY および DEQUEUE_ANY を付与する必要があります。

```
CREATE USER BOLADM IDENTIFIED BY BOLADM; GRANT CONNECT, RESOURCE, aq_administrator_
role TO BOLADM;
((AQjmsSession)t_sess).grantSystemPrivilege("ENQUEUE_ANY", "BOLADM", false);
((AQjmsSession)t_sess).grantSystemPrivilege("DEQUEUE_ANY", "BOLADM", false)
;where t_sess is the session object.
```

このアプリケーションでは、AQ のプロパゲータは OE（注文入力）スキーマから WS（西部売上）、ES（東部売上）および OS（海外売上）スキーマにメッセージを移入します。次に、WS、ES および OS スキーマから CB（顧客請求）および CS（顧客サービス）スキーマにメッセージを移入します。したがって、OE、WS、ES および OS スキーマのすべてに、プロパゲータのソース・キューとなるキューがあります。

メッセージが宛先キューに届くと、ソース・キューのスキーマ名に基づいたセッションによって、新しく届いたメッセージが宛先キューにエンキューされます。つまり、ソース・キューのスキーマに、宛先キューに対するエンキュー権限を付与する必要があります。

管理を単純化するために、BooksOnLine アプリケーションでソース・キューを持つすべてのスキーマに ENQUEUE_ANY システム権限を付与します。

```
((AQjmsSession)t_sess).grantSystemPrivilege("ENQUEUE_ANY", "OE", false);
((AQjmsSession)t_sess).grantSystemPrivilege("ENQUEUE_ANY", "WS", false);
((AQjmsSession)t_sess).grantSystemPrivilege("ENQUEUE_ANY", "ES", false);
```

```
((AQjmsSession)t_sess).grantSystemPrivilege("ENQUEUE_ANY", "OS", false);
where t_sess is the session object
```

リモートの宛先キューに伝播するために、エージェント構造体のアドレス・フィールドのデータベース・リンクに指定されたログイン・ユーザーには、ENQUEUE ANY QUEUE 権限を付与するか、または宛先キューに対するエンキュー権限を付与する必要があります。ただし、データベース・リンクのログイン・ユーザーが宛先のキュー表を所有している場合は、どのような明示的な権限も付与する必要はありません。

JMS での宛先レベルのアクセス制御

Oracle9*i* は、エンキューおよびデキュー操作に対してキュー・レベルまたはトピック・レベルのアクセス制御をサポートします。この機能によって、アプリケーション設計者は、あるスキーマに作成されたキューまたはトピックを、他のスキーマで実行中のアプリケーションから保護できます。そのキューまたはトピックが属するスキーマの外で実行しているアプリケーションには、最小限のアクセス権限のみを付与します。キューまたはトピックに対するアクセス権限として、ENQUEUE、DEQUEUE および ALL がサポートされています。詳細は、第 4 章「AQ の管理」の「Oracle Enterprise Manager のサポート」を参照してください。

サンプル・シナリオおよびコード

BooksOnLine アプリケーションでは、顧客請求情報を CB および CBADM スキーマで処理します。CB（顧客請求情報）スキーマには顧客請求情報アプリケーションがあり、CBADM スキーマには、すべての関連する請求情報データがキュー表の形で格納されています。請求情報データを保護するために、請求処理アプリケーションと請求情報データは別のスキーマに常駐しています。請求処理アプリケーションは、出荷済の注文情報トピックである CBADM_shippedorders_topic からメッセージをデキューする以外の処理はできません。そこでメッセージを処理し、請求済の注文情報トピックである CBADM_billedorders_topic に新しいメッセージをエンキューします。

他のアプリケーションの不正な操作からキューを保護するために、次の 2 つの grant コールを行います。

```
/* Grant dequeue privilege on the shipped orders queue to the Customer
   Billing application. The CB application retrieves orders that are shipped
   but not billed from the shipped orders queue. */

((AQjmsDestination)cbadm_shippedorders_topic).grantTopicPrivilege(t_sess, "DEQUEUE",
"CB", false);
where t_sess is the session

/* Grant enqueue privilege on the billed orders queue to Customer Billing
   application.The CB application is allowed to put billed orders into this
   queue after processing the orders. */

((AQjmsDestination)cbadm_billedorders_topic).grantTopicPrivilege(t_sess, "ENQUEUE",
"CB", false);
```

JMS での保存およびメッセージ履歴

AQ では、ユーザーがメッセージをキュー表に保存できます。これによって、SQL を使用してメッセージを問い合せて分析できます。メッセージは、相互に関連していることがよくあります。たとえば、あるメッセージを処理した結果、他のメッセージが生成された場合、両者は関連付けられています。アプリケーション設計者としては、そのような関連を追跡することが必要な場合があります。保存機能およびメッセージ識別子とともに、メッセージ・ジャーナルが AQ によって自動作成され、追跡ジャーナルまたはイベント・ジャーナルをコールできます。保存、メッセージ識別子および SQL 問合せの協調によって、強力なメッセージ・ウェアハウスが構築できます。

サンプル・シナリオおよびコード

出荷処理アプリケーションで、注文の平均処理時間を判断する必要があると仮定します。この時間には、backed_order トピックでの待機時間も含まれます。出荷キューの保存を TRUE に指定し、メッセージの相関フィールドの注文番号を指定した SQL 問合せを作成すると、出荷処理アプリケーション中の注文情報の待機時間を判断できます。

単純化のために、すでに処理された注文情報のみを分析することにします。出荷処理アプリケーション中で注文情報の処理にかかる時間は、WS_bookedorders_topic にエンキューされる時刻と WS_shipped_orders_topic にエンキューされる時刻の差です。

```
SELECT  SUM(SO.enq_time - BO.enq_time) / count (*) AVG_PRCS_TIME
   FROM WS.AQ$WS_orders_pr_mqtab BO , WS.AQ$WS_orders_mqtab SO
   WHERE SO.msg_state = 'PROCESSED' and BO.msg_state = 'PROCESSED'
   AND SO.corr_id = BO.corr_id and SO.queue = 'WS_shippedorders_topic';

/* Average waiting time in the backed order queue: */
SELECT SUM(BACK.deq_time - BACK.enq_time)/count (*) AVG_BACK_TIME
   FROM WS.AQ$WS_orders_mqtab BACK
   WHERE BACK.msg_state = 'PROCESSED' AND BACK.queue = 'WS_backorders_topic';
```

JMS での Oracle Real Application Clusters のサポート

Oracle Real Application Clusters を使用すると、異なるキューを別々のインスタンスによって管理できるようにして AQ パフォーマンスを改善できます。このためには、キューを格納するキュー表に様々なインスタンス・アフィニティ（作業環境）を指定します。これによって、様々なキュー / トピックに対するキュー操作（エンキューまたはデキュー）またはトピック操作（パブリッシュ・サブスクライブ）を並行して行うことができるようになります。

AQ のキュー・モニター・プロセスは、キュー表のインスタンス・アフィニティを継続的に監視します。キュー・モニターは指定されたプライマリ・インスタンスが使用可能な場合はそれにキュー表の所有権を割り当て、失敗した場合は指定されたセカンダリ・インスタンスに割り当てます。

キュー表を所有しているインスタンスが停止した場合、キュー・モニターはそのキュー・インスタンスの所有権をセカンダリ・インスタンスなどの適切なインスタンスに割り当てます。

AQ の伝播は Real Application Clusters でも使用可能になりますが、これはユーザーにとっては透過的です。伝播スケジュールのジョブ・アフィニティは、それぞれのキュー表のアフィニティと同じ値に設定されます。このように、キュー表を所有するインスタンスに対応付けられたジョブ・キュー・プロセスは、そのキュー表に格納されているキューからの伝播を処理し、ping 操作を最小限に抑えます。詳細は、9-72 ページの「キューの伝播のスケジューリング」および『Oracle9*i* Real Application Clusters セットアップおよび構成』を参照してください。

サンプル・シナリオおよびコード

BooksOnLine の例では、注文入力（OE）サイトでの OE_neworders_que および booked_order_topic の操作は、この 2 つのトピックが異なるインスタンスと対応付けられている場合は高速化できます。そのために、2 つのトピックを別々のキュー表に作成し、CreateQueueTable() コマンドでそれらのキュー表に別々のアフィニティを指定します。

この例では、キュー表 OE_orders_sqtab にキュー OE_neworders_que を格納し、プライマリおよびセカンダリ・インスタンスはそれぞれインスタンス 1 および 2 です。キュー表 OE_orders_mqtab にはキュー booked_order_topic を格納し、プライマリおよびセカンダリ・インスタンスはそれぞれインスタンス 2 および 1 です。目的は、インスタンス 1 および 2 に 2 つのキューをパラレルで管理させることです。デフォルトでは、使用可能なインスタンスは 1 つのみです。この場合、両方のキュー表を所有しているインスタンスはインスタンス 1 に設定されます。ただし、Oracle Real Application Clusters が正しく設定され、インスタンス 1 および 2 が使用可能な場合は、キュー表 OE_orders_sqtab がインスタンス 1 によって所有され、他のキュー表はインスタンス 2 によって所有されます。キュー表に対するプライマリおよびセカンダリ・インスタンスの指定は、次の例に示すように、alter_queue_table() コマンドを使用して動的に変更できます。キュー表のプライマリ、セカンダリおよび所有者インスタンスに関する情報は、USER_QUEUE_TABLES ビューを問い合せて取得できます。10-21 ページの「ユーザー・スキーマのキュー表の選択」を参照してください。

```
/* Create queue tables, topics for OE */

/* createing a queue table to hold queues */
qt_prop = new AQQueueTableProperty("SYS.AQ$_JMS_OBJECT_MESSAGE");
qt_prop.setPrimaryInstance(1);
qt_prop.setSecondaryInstance(2);
q_table = createQueueTable("OE", "OE_orders_sqtab", qt_prop);

/* creating a queue table to hold topics */
qt1_prop = new AQQueueTableProperty("SYS.AQ$_JMS_OBJECT_MESSAGE");
qt1_prop.setMultiConsumer(TRUE);
qt1_prop.setPrimaryInstance(2);
qt1_prop.setSecondaryInstance(1);
```

```
q_table1 = createQueueTable("OE", "OE_orders_mqtab", qt1_prop);

dest_prop = new AQjmsDestinationProperty();
queue = ((AQjmsSession)q_sess).createQueue(q_table. "OE_neworders_que",
                                           dest_prop);

dest_prop1 = new AQjmsDestinationProperty();
topic = ((AQjmsSession)q_sess).createTopic(q_table1, "OE_bookedorders_topic",
                                           dest_prop1);


/* Check instance affinity of OE queue tables from AQ administrative view: */
SELECT queue_table, primary_instance, secondary_instance, owner_instance
FROM user_queue_tables;

/* Alter Instance Affinity of OE queue tables */
q_table.alter("OE_orders_sqtab", 2, 1);
q_table1.alter("OE_orders_mqtabl", 1, 2);
```

JMS での統計ビューのサポート

各インスタンスは、それぞれの AQ 統計情報をシステム・グローバル領域（SGA）に保持しますが、他のインスタンスによって収集された統計については保持しません。そのため、あるインスタンスが GV$AQ ビューに対して問い合せると、その他のすべてのインスタンスからそれぞれのその時点の AQ 統計情報が問合せ元のインスタンスに集まります。

サンプル・シナリオおよびコード

GV$ ビューは、待機中、準備完了または期限切れの各メッセージ数を必要なときにいつでも問い合せることができます。また、メッセージが処理されるまでの平均待機秒数も表示します。注文処理アプリケーションは、これを使用して注文処理プロセスの数を動的にチューニングできます（10-33 ページの「データベース全体における状態ごとのメッセージ数の選択」を参照）。

```
CONNECT oe/oe

/* Count the number as messages and the average time for which the messages
   have been waiting: */
SELECT READY, AVERAGE_WAIT
FROM gv$aq Stats, user_queues Qs
WHERE Stats.qid = Qs.qid and Qs.Name = 'OE_neworders_que';
```

JMS での構造化ペイロード / メッセージの型

JMS メッセージは、次の部分で構成されます。

- ヘッダー - すべてのメッセージは、同じヘッダー・フィールドの集合をサポートします。ヘッダー・フィールドには、クライアントおよびプロバイダの両方がメッセージの識別およびルーティングに使用する値が含まれます。
- プロパティ - 標準ヘッダー・フィールドの他に、オプションのヘッダー・フィールドをメッセージに追加できます。
  - 標準プロパティ - JMS は、実質的にはオプションのヘッダー・フィールドであるいくつかの標準プロパティを定義します。
  - プロバイダ固有プロパティ - 各 JMS プロバイダが、特定のプロバイダ固有のプロパティをメッセージに追加できます。
  - アプリケーション固有プロパティ - メッセージにアプリケーション固有ヘッダー・フィールドを追加できます。
- 本体 - メッセージ・ペイロードです。JMS は、様々な型のメッセージ・ペイロードおよび JMS で指定されたメッセージ型の JMS メッセージを格納できる型を定義します。

メッセージ・ヘッダー

ヘッダーのみの JMS メッセージを使用できます。メッセージ本体は必要ありません。メッセージ・ヘッダーには、次のフィールドが含まれます。

- `JMSDestination` - このフィールドには、メッセージの送信先が含まれます。AQ では、これは宛先キュー / トピックに対応します。
- `JMSDeliveryMode` - JMS は、`PERSISTENT` および `NON_PERSISTENT` の 2 つのメッセージ配信モードをサポートします。`PERSISTENT` では、メッセージは決まった記憶域に記録され、`NON_PERSISTENT` では、メッセージのログが取られません。Oracle AQ は、`PERSISTENT` メッセージ配信をサポートします。
- `JMSMessageID` - この値は、プロバイダのメッセージを一意に識別します。すべてのメッセージ ID は、ID: で始まる必要があります。
- `JMSTimeStamp` - メッセージが送信先のプロバイダに渡された時刻が含まれます。これは、AQ のメッセージのエンキュー時刻に対応します。
- `JMSCorrelationID` - クライアントは、このフィールドを使用して、あるメッセージを別のメッセージにリンクできます。
- `JMSReplyTo` - このフィールドには、メッセージ送信時にクライアントが指定する宛先が含まれます。クライアントは、`ReplyTo` 宛先の指定に `oracle.jms.AQjmsAgent`、`javax.jms.Queue`、`javax.jms.Topic` の型を使用できます。
- `JMSType` - このフィールドには、送信時にクライアントが指定するメッセージ型識別子が含まれます。移植性の問題から、`JMSType` を記号値にすることをお薦めします。

- JMSExpiration - J2EE 非準拠モードでは、JMSExpiration ヘッダー値は、エンキュー時間と Time-To-Live の合計になります。準拠モードでは、デキューされたメッセージの JMSExpiration ヘッダー値は、メッセージがエンキューされたときの JMS タイムスタンプ（グリニッジ標準時で、1000 分の 1 秒単位）と Time-To-Live（1000 分の 1 秒単位）の合計になります。詳細は、12-3 ページの「J2EE 準拠」を参照してください。
- JMSPriority - このフィールドには、メッセージの優先順位が含まれます。J2EE 準拠モードで指定できる優先順位の値は 0 ～ 9 です。Sun 社の JMS 1.0.2 標準準拠では、9 が最も高い優先順位で、4 がデフォルトです。デフォルトは非準拠モードです。詳細は、12-3 ページの「J2EE 準拠」を参照してください。JMS では、クライアントが JMSDeliveryMode、JMSExpiration および JMSPriority に指定した値を管理者がオーバーライドするように JMS を構成できます。

メッセージ・プロパティ

プロパティは、メッセージにオプションのヘッダー・フィールドを追加する機能です。プロパティによって、クライアントは、メッセージ・セレクタを使用して、クライアントのかわりにプロバイダにアプリケーション固有基準を使用してメッセージを選択させることができます。プロパティ名は文字列であり、値は Boolean、Byte、Short、Int、Long、Float、Double および String にすることができます。

JMS 定義済プロパティは、「JMSX」で始まります。

- JMSXUserID - メッセージを送信するユーザーの識別情報です。
- JMSXAppID - メッセージを送信するアプリケーションの識別情報です。
- JMSXDeliveryCount - メッセージ配信の試行回数です。
- JMSXGroupid - このフィールドには、このメッセージが属するメッセージ・グループの識別情報へのクライアント参照が設定されます。
- JMSXGroupSeq - グループ内のメッセージの順序番号です。
- JMSXRcvTimeStamp - メッセージがコンシューマに配信された時刻（デキュー時刻）です。
- JMSXState - プロバイダによって設定されたメッセージ状態です。メッセージは、WAITING、READY、EXPIRED または RETAINED の状態に設定できます。

Oracle JMS 固有プロパティは、JMS_Oracle で始まります。次のプロパティは、Oracle 固有です。

- JMS_OracleExcpQ - メッセージを元の宛先に配信できない場合に、そのメッセージの送信先となるキュー名です。JMS_OracleExcpQ プロパティに指定できる宛先キューのタイプは、EXCEPTION のみです。
- JMS_OracleDelay - メッセージ配信の遅延秒数です。これは、メッセージが配信されるときの順序に影響します。

- JMS_OracleOriginalMessageId - メッセージが 1 つの宛先から別の宛先に伝播される場合、このプロパティは元のメッセージのメッセージ ID に設定されます。メッセージが伝播されない場合、このプロパティは JMSMessageId と同じ値になります。

クライアントは、プロパティを定義することによって、メッセージにヘッダー・フィールドを追加できます。これらのプロパティをメッセージ・セレクタで使用して特定のメッセージを選択できます。

JMS プロパティまたはヘッダー・フィールドは、クライアントによって明示的に設定されるか、または JMS プロバイダによって自動的に設定されます（通常、これらは読込み専用です）。JMS プロパティには、送受信操作に指定されたパラメータを使用して設定されるものもあります。

表 12-1 メッセージ・ヘッダー・フィールド

メッセージ・ヘッダー・フィールド	型	設定方法	使用目的
JMSDestination	Destination	Send メソッド完了後に JMS によって設定されます。	メッセージの送信宛先
JMSDeliveryMode	Int	Send メソッド完了後に JMS によって設定されます。	配信モード -PERSISTENT
JMSExpiration	Long	Send メソッド完了後に JMS によって設定されます。	期限切れ時刻。メッセージ・プロデューサに対して指定、あるいは送信またはパブリッシュ中に明示的に指定できます。
JMSPriority	Int	Send メソッド完了後に JMS によって設定されます。	メッセージの優先順位。メッセージ・プロデューサに対して指定、あるいは送信またはパブリッシュ中に明示的に指定できます。
JMSMessageID	String	Send メソッド完了後に JMS によって設定されます。	プロバイダによって送信された各メッセージを一意に識別する値
JMSTimeStamp	Long	Send メソッド完了後に JMS によって設定されます。	送信先のプロバイダにメッセージが渡された時刻
JMSCorrelationID	String	JMS クライアントによって設定されます。	1 つのメッセージを別のメッセージにリンクするために使用できるフィールド

表 12-1 メッセージ・ヘッダー・フィールド（続き）

メッセージ・ヘッダー・フィールド	型	設定方法	使用目的
JMSReplyTo	Destination	JMS クライアントによって設定されます。	メッセージへの応答が送信される、クライアントによって設定された宛先。AQjsAgent、javax.jms.Queue または javax.jms.Topic 型として指定される必要があります。
JMSType	String	JMS クライアントによって設定されます。	メッセージの型を表す識別子
JMSRedelivered	Boolean	JMS プロバイダによって設定されます。	配信済の可能性があるメッセージが、配信時にクライアントに認識されませんでした。

表 12-2 JMS 定義のメッセージ・プロパティ

JMS 定義のメッセージ・プロパティ	型	設定方法	使用目的
JMSXUserID	String	Send メソッド完了後に JMS によって設定されます。	メッセージを送信するユーザーの識別情報
JMSAppID	String	Send メソッド完了後に JMS によって設定されます。	メッセージを送信するアプリケーションの識別情報
JMSDeliveryCount	Int	Receive メソッド完了後に JMS によって設定されます。	メッセージ配信の試行回数。1 回目は 1、2 回目以降は 2、3、4 と続きます。
JMSXGroupID	String	JMS クライアントによって設定されます。	そのメッセージが属するメッセージ・グループの識別情報
JMSXGroupSeq	Int	JMS クライアントによって設定されます。	グループ内のメッセージの順序番号。1 番目のメッセージは 1、2 番目以降は 2、3、4 と続きます。
JMSXRcvTimeStamp	String	Receive メソッド完了後に JMS によって設定されます。	JMS がメッセージをコンシューマに配信した時刻
JMSXState	Int	JMS プロバイダによって設定されます。	プロバイダによって設定されたメッセージ状態

表 12-3 Oracle 定義のメッセージ・プロパティ

ヘッダー・フィールド / プロパティ	型	設定方法	使用目的
JMS_OracleExcpQ	String	JMS クライアントによって設定されます。	例外キュー名を指定します。
JMS_OracleDelay	Int	JMS クライアントによって設定されます。	コンシューマがメッセージを使用できるようになるまでの時間（秒数）を指定します。
JMS_OracleOriginalMessageID	String	JMS プロバイダによって設定されます。	メッセージが 1 つの宛先から別の宛先に伝播されるとき、ソースのメッセージのメッセージ ID を指定します。

メッセージ本体

JMS では、次の 5 つのフォーマットのメッセージ本体が提供されます。

- **StreamMessage** - 本体に Java 基本データ型の値がストリームとして含まれるメッセージです。このメッセージ本体には順次書込み / 読込みが可能です。
- **BytesMessage** - 本体にバイト列のストリームが含まれるメッセージです。これは、本体を直接エンコードして既存のメッセージ・フォーマットに一致させるためのメッセージ型です。
- **MapMessage** - 本体に名前と値の組の集合を含むメッセージです。名前は文字列であり、値は Java 基本データ型です。エントリには、Enumeration オブジェクトを使用して順次アクセスするか、名前でランダムにアクセスできます。
- **TextMessage** - 本体に `java.lang.String` が含まれるメッセージです。
- **ObjectMessage** - シリアル化可能な Java オブジェクトが含まれるメッセージです。
- **AdtMessage** - 本体に Oracle のユーザー定義型オブジェクトが含まれるメッセージです（Oracle JMS では、`AdtMessage` 型が追加されています）。

AQ$_JMS_MESSAGE 型

この型は、JMS によって指定されたすべてのメッセージ型（`JMSStream`、`JMSBytes`、`JMSMap`、`JMSText` および `JMSObject`）の JMS メッセージを格納できます。`AQ$_JMS_MESSAGE` 型のキュー表を作成することにより、すべてのメッセージ型を使用できます。

StreamMessage

StreamMessage は、Java 基本データ型の値をストリームとして送信する場合に使用します。このメッセージ本体には順次書込み / 読込みが可能です。StreamMessage は Message から拡張され、StreamMessage 本体を追加します。このメソッドは、主に java.io.DataInputStream および java.io.DataOutputStream のメソッドに基づいています。

基本データ型の値は、それぞれの型ごとのメソッドを使用して、明示的に読込みまたは書込みができます。また、抽象的なオブジェクトとして読込みまたは書込みが行われる場合もあります。StreamMessage を使用するには、ペイロード型 SYS.AQ$_JMS_STREAM_MESSAGE または AQ$_JMS_MESSAGE を持つキュー表を作成します。

StreamMessage は、次の変換表をサポートします。行の型として書き込まれる値は、列の型として読み込むことができます。

表 12-4 StreamMessage 変換

	Boolean	Byte	Short	Char	Int	Long	Float	Double	String	Byte[]
Boolean	X	-	-	-	-	-	-	-	X	-
Byte	-	X	X	-	X	X	-	-	X	-
Short	-	-	X	-	X	X	-	-	X	-
Char	-	-	-	X	-	-	-	-	X	-
Int	-	-	-	-	X	X	-	-	X	-
Long	-	-	-	-	-	X	-	-	X	-
Float	-	-	-	-	-	-	X	X	X	-
Double	-	-	-	-	-	-	-	X	X	-
String	X	X	X	X	X	X	X	X	X	-
Byte[]	-	-	-	-	-	-	-	-	-	X

BytesMessage

BytesMessage は、未解析バイトの 1 つのストリームを含むメッセージを送信する場合に使用します。BytesMessage は Message から拡張され、BytesMessage 本体を追加します。メッセージの受信者が、そのバイト列を解析します。このメソッドは、主に java.io.DataInputStream および java.io.DataOutputStream のメソッドに基づいています。

これは、クライアントが既存のメッセージ・フォーマットをコード化するためのメッセージ型です。可能であれば、かわりに他の自己定義メッセージ型を使用してください。

Java の基本データ型の値は、それぞれの型のメソッドを使用して、明示的に書込みができます。また、抽象的なオブジェクトとして書き込まれる場合もあります。BytesMessage を使

用するには、ペイロード型 SYS.AQ$_JMS_BYTES_MESSAGE または AQ$_JMS_MESSAGE を持つキュー表を作成します。

MapMessage

MapMessage は、名前が文字列で値が Java 基本データ型である名前と値の組の集合を送信する場合に使用します。エントリには、名前で順次またはランダムにアクセスできます。エントリの順序は未定義です。MapMessage は Message から拡張され、MapMessage 本体を追加します。基本データ型の値は、それぞれの型のメソッドを使用して、明示的に読込みまたは書込みできます。また、抽象的なオブジェクトとして読込みまたは書込みが行われる場合もあります。

MapMessage を使用するには、ペイロード型 SYS.AQ$_JMS_MAP_MESSAGE または AQ$_JMS_MESSAGE を持つキュー表を作成します。MapMessage は、次の変換表をサポートします。行の型として書き込まれる値は、列の型として読み込むことができます。

表 12-5 MapMessage 変換

	Boolean	Byte	Short	Char	Int	Long	Float	Double	String	Byte[]
Boolean	X	-	-	-	-	-	-	-	X	-
Byte	-	X	X	-	X	X	-	-	X	-
Short	-	-	X	-	X	X	-	-	X	-
Char	-	-	-	X	-	-	-	-	X	-
Int	-	-	-	-	X	X	-	-	X	-
Long	-	-	-	-	-	X	-	-	X	-
Float	-	-	-	-	-	-	X	X	X	-
Double	-	-	-	-	-	-	-	X	X	-
String	X	X	X	X	X	X	X	X	X	-
Byte[]	-	-	-	-	-	-	-	-	-	X

TextMessage

TextMessage は java.lang.StringBuffer を含むメッセージを送信する場合に使用します。TextMessage は Message から継承され、TextMessage 本体を追加します。テキスト情報は、getText() および setText(...) メソッドを使用して読込みまたは書込みができます。TextMessage を使用するには、ペイロード型 SYS.AQ$_JMS_TEXT_MESSAGE または AQ$_JMS_MESSAGE を持つキュー表を作成します。

ObjectMessage

ObjectMessage は、シリアル化可能な Java オブジェクトを含むメッセージを送信する場合に使用します。ObjectMessage は、Message から拡張され、単一の Java オブジェクトへの参照を含む本体を追加します。シリアル化可能な Java オブジェクトのみを使用できます。Java オブジェクトのコレクションの送信が必要な場合は、JDK 1.2 で提供されたコレクション・クラスの 1 つを使用できます。このオブジェクトは、`getObject()` および `setObject(...)` メソッドを使用して読込みまたは書込みができます。ObjectMessage を使用するには、ペイロード型 `SYS.AQ$_JMS_OBJECT_MESSAGE` または `AQ$_JMS_MESSAGE` を持つキュー表を作成します。

サンプル・コード

```
public void enqueue_new_orders(QueueSession jms_session, BolOrder new_order)
{
   QueueSender   sender;
   Queue         queue;
   ObjectMessage obj_message;

   try
   {
       /* get a handle to the new_orders queue */
       queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");
       sender = jms_session.createSender(queue);
       obj_message = jms_session.createObjectMessage();
       obj_message.setJMSCorrelationID("RUSH");
       obj_message.setObject(new_order);
       jms_session.commit();
    }
    catch (JMSException ex)
    {
      System.out.println("Exception: " + ex);
    }

}
```

AdtMessage

AdtMessage は、Oracle のオブジェクト型に対応する Java オブジェクトを含むメッセージを送信する場合に使用します。これらのオブジェクトは Message から拡張され、`CustomDatum` または `ORAData` インタフェースを実装する Java オブジェクトを含む本体を追加します。

> **参照：** `CustomDatum` インタフェースおよび `ORAData` インタフェースの詳細は、『Oracle9*i* Java Developer's Guide』を参照してください。

AdtMessage を使用するには、Oracle のオブジェクト型としてのペイロード型を持つキュー表を作成します。AdtMessage のペイロードは、getAdtPayload および setAdtPayload メソッドを使用して、読込みおよび書込みができます。

AdtMessage を使用して SYS.XMLType 型のキューへメッセージを送信することもできます。oracle.xdb.XMLType クラスを使用してメッセージを作成する必要があります。

異なるメッセージ型でのメッセージ・プロパティの使用

- クライアントが setProperty コールを使用して設定できる JMS プロパティは、次のとおりです。
  - StreamMessage、BytesMessage、ObjectMessage、TextMessage、MapMessage の場合

    JMSXAppID

    JMSXGroupID

    JMSXGroupSeq

    JMS_OracleExcpQ

    JMS_OracleDelay
  - AdtMessage の場合

    JMS_OracleExcpQ

    JMS_OracleDelay
- クライアントが getProperty コールを使用して取得できる JMS プロパティは、次のとおりです。
  - StreamMessage、BytesMessage、ObjectMessage、TextMessage、MapMessage の場合

    JMSXuserID

    JMSXAppID

    JMSXDeliveryCount

    JMSXGroupID

    JMSXGroupSeq

    JMSXRecvTimeStamp

    JMSXState

    JMS_OracleExcpQ

    JMS_OracleDelay

JMS_OracleOriginalMessageID

– AdtMessage の場合

JMSXDeliveryCount

JMSXRecvTimeStamp

JMSXState

JMS_OracleExcpQ

JMS_OracleDelay

- メッセージ・セレクタに含めることができる JMS プロパティ / ヘッダー・フィールドは、次のとおりです。
  – （JMS 型ペイロードを含むキューの）キュー・レシーバ、トピック・サブスクライバおよびトピック・レシーバの場合は、次のものを含む文字列があるすべての SQL92 の WHERE 句

    JMSPriority(Int)

    JMSCorrelationID(String)

    JMSMessageID(String) - キュー・レシーバおよびトピック・レシーバに対してのみ

    JMSTimestamp(Date)

    JMSType(String)

    JMSXUserID(String)

    JMSXAppID(String)

    JMSXGroupID(String)

    JMSXGroupSeq(Int)

    JMS メッセージでのすべてのユーザー定義のプロパティ

  – （ユーザー定義型ペイロードを含むキューの）キュー・レシーバ、トピック・サブスクライバおよびトピック・レシーバの場合は、次のものを含む文字列があるすべての SQL92 の WHERE 句に対して AQ 規則の構文を使用します。

    * corrid
    * priority
    * tab.user_data.<adt_field_name>

JMS サンプルで使用したペイロード

```
/*
 *  BooksOrder - payload for BooksOnline example
 *
 */

import java.lang.*;
import java.io.*;
import java.util.*;

public class BolOrder implements Serializable
{

  int             orderno;
  String          status;
  String          type;
  String          region;
  BolCustomer     customer;
  String          paymentmethod;
  BolOrderItem[]  itemlist;
  String          ccnumber;
  Date            orderdate;

  public BolOrder(int orderno, BolCustomer customer)
  {
    this.customer   = customer;
    this.orderno    = orderno;
  }

  public int getOrderNo()
  {
    return orderno;
  }

  public String getStatus()
  {
    return status;
  }


  public void setStatus(String new_status)
  {
    status = new_status;
  }
```

```
public String getRegion()
{
  return region;
}


public void setRegion(String region)
{
  this.region = region;
}

public BolCustomer getCustomer()
{
  return customer;
}


public String getPaymentmethod()
{
  return paymentmethod;
}

public void setPaymentmethod(String paymentmethod)
{
  this.paymentmethod = paymentmethod;
}

public BolOrderItem[] getItemList()
{
  return itemlist;
}

public void setItemList(BolOrderItem[] itemlist)
{
  this.itemlist = itemlist;
}

public String getCCnumber()
{
  return ccnumber;
}

public void setCCnumber(String ccnumber)
{
  this.ccnumber = ccnumber;
}
```

```
  public Date getOrderDate()
  {
    return orderdate;
  }

  public void setOrderDate(Date orderdate)
  {
    this.orderdate = orderdate;
  }

}

/*
 *  BolOrderItem - order item type  for BooksOnline example
 *
 */

import java.lang.*;
import java.io.*;
import java.util.*;

public class BolOrderItem implements Serializable
{

  BolBook     item;
  int         quantity;

  public BolOrderItem(BolBook book, int quantity)
  {
    item          = book;
    this.quantity = quantity;

  }

  public BolBook getItem()
  {
    return item;
  }

  public int getQuantity()
  {
    return quantity;
  }
}


/*
```

```
 *  BolBook - book type  for BooksOnline example
 *
 */

import java.lang.*;
import java.io.*;
import java.util.*;

public class BolBook implements Serializable
{

  String    title;
  String    authors;
  String    isbn;
  float     price;

  public BolBook(String title)
  {
    this.title   = title;
  }

  public BolBook(String title, String authors, String isbn, float price)
  {
    this.title   = title;
    this.authors = authors;
    this.isbn    = isbn;
    this.price   = price;

  }

  public String getISBN()
  {
    return isbn;
  }

  public String getTitle()
  {
    return title;
  }

  public String getAuthors()
  {
    return authors;
  }

  public float getPrice()
  {
```

```
    return price;
  }

}
/*
 *  BolCustomer - customer type  for BooksOnline example
 *
 */

import java.lang.*;
import java.io.*;
import java.util.*;

public class BolCustomer implements Serializable
{

  int          custno;
  String       custid;
  String       name;
  String       street;
  String       city;
  String       state;
  int          zip;
  String       country;

  public BolCustomer(int custno, String name)
  {

    this.custno  = custno;
    this.name    = name;
  }

  public BolCustomer(int custno, String custid, String name, String  street,
          String city, String state, int zip, String country)
  {

    this.custno  = custno;
    this.custid  = custid;
    this.name    = name;
    this.street  = street;
    this.city    = city;
    this.state   = state;
    this.zip     = zip;
    this.country = country;

  }
```

```
  public int getCustomerNo()
  {
    return custno;
  }

  public String getCustomerId()
  {
    return custid;
  }

  public String getName()
  {
    return name;
  }

  public String getStreet()
  {
    return street;
  }

  public String getCity()
  {
    return city;
  }

  public String getState()
  {
    return state;
  }

  public int getZipcode()
  {
    return zip;
  }

  public String getCountry()
  {
    return country;
  }
}
```

JMS での Point-to-Point モデル機能

- キュー
- キュー・センダー
- キュー・レシーバ
- キュー・ブラウザ

キュー

Point-to-Point モデルでは、クライアントは、1 つのポイントから別のポイントへ、キューを使用してメッセージを交換します。これらのキューは、メッセージ・プロデューサおよびコンシューマによるメッセージの送受信に使用されます。

管理者は、`AQjmsSession` の `createQueue` メソッドを使用して、シングル・コンシューマ・キューを作成します。クライアントは、`AQjmsSession` の `getQueue` メソッドを使用して、事前に作成されたキューに対するハンドルを取得する場合があります。

これらのキューは、単一のコンシューマのみがメッセージを処理できるため、**シングル・コンシューマ・キュー**といわれます。つまり、メッセージは一度に 1 つのコンシューマでしか処理できません。同じキューから、複数のプロセスまたはオペレーティング・システム・スレッドが同時にデキューしようとする場合を考えてみます。ロックされたメッセージをデキューできるのは、ロックを作成したプロセスのみであるとすると、各プロセスは、キューの先頭にあるロックされていない最初のメッセージをデキューします。

キューを使用する前に、`AQjmsDestination` で `start` コールを使用して、キューをエンキュー / デキューに対して有効にする必要があります。

処理が済むと、キューの保存期間が 0（ゼロ）の場合はそのメッセージは削除され、そうでない場合は指定された期間保存されます。メッセージが保存されている間は、次のいずれかが可能です。

- キュー表ビューに対して SQL を使用して問合せができます。
- キュー・ブラウザを使用し、処理済のメッセージのメッセージ ID を指定して、デキューできます。

キュー・センダー

クライアントは、キュー・センダーを使用して、キューにメッセージを送信します。キュー・センダーは、キューをセッションの createSender メソッドに渡すことによって作成されます。また、クライアントには、キューを指定することなくキュー・センダーを作成するオプションがあります。この場合、キューは、送信操作のたびに指定される必要があります。

クライアントは、キュー・センダーによって送信されたすべてのメッセージのデフォルト配信モード、優先順位および Time-To-Live（生存期間）を指定できます。また、クライアントは、これらのオプションをメッセージ単位で定義できます。

サンプル・コード

BooksOnLine アプリケーションでは、新規の注文が new_orders_queue に送信されます。JMS コネクションおよびセッションを作成した後、次のようにキュー・センダーを作成します。

```
public void enqueue_new_orders(QueueSession jms_session, BolOrder new_order)
{
   QueueSender   sender;
   Queue         queue;
   ObjectMessage obj_message;

   try
   {
       /* get a handle to the new_orders queue */
       queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");
       sender = jms_session.createSender(queue);
       obj_message = jms_session.createObjectMessage();
       obj_message.setJMSCorrelationID("RUSH");
       obj_message.setObject(new_order);
       sender.send(obj_message);
       jms_session.commit();
    }
    catch (JMSException ex)
    {
      System.out.println("Exception: " + ex);
    }
}
```

キュー・レシーバ

クライアントは、キュー・レシーバを使用して、キューからメッセージを受信します。キュー・レシーバは、セッションの createQueueReceiver メソッドを使用して作成されます。キュー・レシーバは、メッセージ・セレクタを使用して作成できます。これによって、クライアントは、コンシューマに配信されるメッセージをセレクタにマッチするメッセージに制限できるようになります。

ペイロード型 TextMessage、StreamMessage、BytesMessage、ObjectMessage、MapMessage を含むキューのセレクタには、次のいずれか 1 つ以上を組み合せた式を含めることができます。

- JMSMessageID = 'ID:23452345'。指定されたメッセージ ID（すべてのメッセージは接頭辞 ID: を持つ）を持つメッセージを取り出します。
- JMS メッセージ・ヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'

  JMSCorrelationID LIKE 'RE%'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

AdtMessage を含むキューの場合、セレクタはメッセージ・ペイロード内容、優先順位または相関 ID に対する SQL 式である必要があります。

- メッセージ ID に対するセレクタ - 特定のメッセージ ID を持つメッセージを取り出します。

  ```
  msgid = '23434556566767676'
  ```

  注意： この場合、メッセージ ID に、接頭辞 ID: を使用しないでください。

- 優先順位または相関 ID に対するセレクタは、次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードに対するセレクタは、次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

サンプル・シナリオおよびコード

BooksOnLine アプリケーションでは、新規の注文が new_orders_queue から取り出されます。これらの注文は、OE.OE_bookedorders_topic に対してパブリッシュされます。JMS コネクションおよびセッションを作成した後、次のようにメッセージの受信者を作成します。

```
public void get_new_orders(QueueSession jms_session)
{
 QueueReceiver   receiver;
 Queue           queue;
 ObjectMessage   obj_message;
 BolOrder        new_order;
 BolCustomer     customer;
 String          state;
 String          cust_name;

 try
 {

   /* get a handle to the new_orders queue */
   queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");

   receiver = jms_session.createReceiver(queue);

   for(;;)
   {
     /* wait for a message to show up in the queue */
     obj_message = (ObjectMessage)receiver.receive(10);

     new_order = (BolOrder)obj_message.getObject();

     customer = new_order.getCustomer();
     state    = customer.getState();

     obj_message.clearBody();

     /* determine customer region and assign a shipping region*/
  if((state.equals("CA")) || (state.equals("TX")) ||
   (state.equals("WA")) || (state.equals("NV")))
  obj_message.setStringProperty("Region", "WESTERN");
     else
  obj_message.setStringProperty("Region", "EASTERN");

     cust_name = new_order.getCustomer().getName();

     obj_message.setStringProperty("Customer", cust_name);
```

```
    if(obj_message.getJMSCorrelationID().equals("RUSH"))
    book_rush_order(obj_message);
    else
    book_new_order(obj_message);

       jms_session.commit();
     }
   }
   catch (JMSException ex)
   {
     System.out.println("Exception: " + ex);
   }
}
```

キュー・ブラウザ

キュー・ブラウザを使用すると、クライアントはメッセージを削除することなく、キュー上でメッセージを参照できます。このブラウズ用メソッドは、キューのメッセージをスキャンするために使用される java.util.Enumeration を戻します。nextElement に対する最初のコールが、キューのスナップショットを取得します。キュー・ブラウザも、メッセージをスキャンしているときに、メッセージをオプションでロックする場合があります。これは、メッセージに対する SELECT...for UPDATE コマンドの場合と似ています。これによって、他のコンシューマがスキャン中のメッセージを削除することはなくなります。

キュー・ブラウザも、メッセージ・セレクタを使用して作成できます。これによって、クライアントは、コンシューマに配信されるメッセージをセレクタにマッチするメッセージに制限できるようになります。

ペイロード型 TextMessage、StreamMessage、BytesMessage、ObjectMessage、MapMessage を含むキューのセレクタには、次のいずれか 1 つ以上を組み合せた式を含めることができます。

- JMSMessageID = 'ID:23452345'。指定されたメッセージ ID（すべてのメッセージは接頭辞 ID: を持つ）を持つメッセージを取り出します。
- JMS メッセージ・ヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  ```

  ```
  JMSCorrelationID LIKE 'RE%'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

AdtMessage を含むキューの場合、セレクタはメッセージ・ペイロード内容、優先順位または相関 ID に対する SQL 式である必要があります。

- メッセージ ID に対するセレクタ - 特定のメッセージ ID を持つメッセージを取り出します。

  ```
  msgid = '23434556566767676'
  ```

  注意： この場合、メッセージ ID に、接頭辞 ID: を使用しないでください。

- 優先順位または相関 ID に対するセレクタは、次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードに対するセレクタは、次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

サンプル・シナリオおよびコード

BooksOnLine アプリケーションでは、新規の注文が new_orders_queue に入れられます。クライアントは、選択されたメッセージをブラウズできます。

```
public void browse_rush_orders(QueueSession jms_session)
{
   QueueBrowser    browser;
   Queue           queue;
   ObjectMessage   obj_message;
   BolOrder        new_order;
   Enumeration     messages;
   String          customer_name;

   try
   {
     /* get a handle to the new_orders queue */
     queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");

     /* create a Browser to look at RUSH orders in USA */
     browser = jms_session.createBrowser(queue,
                          "JMSCorrelationID = 'RUSH' and country = 'USA' ");

     for (messages = browser.getEnumeration() ; messages.hasMoreElements() ;)
     {
       obj_message = (ObjectMessage)messages.nextElement();

       new_order = (BolOrder)obj_message.getObject();

       customer_name = new_order.getCustomer().getName();
       System.out.println("Customer " +  customer_name +
```

```
            " has placed a RUSH order");
      }

      browser.close();
    }
    catch (Exception ex)
    {
      System.out.println("Exception " + ex);
    }
}
```

JMS パブリッシュ・サブスクライブ・モデル機能

この項の内容は次のとおりです。

- トピック
- 永続サブスクライバ
- トピック・パブリッシャ
- 受信者リスト
- トピック・レシーバ
- トピック・ブラウザ

トピック

JMS には様々な機能があり、ユーザーはパブリッシュ・サブスクライブ・モデルに基づいたアプリケーションを開発できます。このアプリケーション・モデルの目的は、パブリッシャ（出版者）の機能を持つアプリケーションとサブスクライバ（購読者）の役割を果たすアプリケーションとの間の柔軟で動的な通信を可能にすることです。具体的な設計のポイントは、そのように異なる役割を果たすアプリケーションを通信上では分離し、メッセージおよびメッセージ内容に基づいた相互作用をさせるという点です。

分散メッセージでは、パブリッシャ・アプリケーションが明示的にメッセージ受信者を処理または管理する必要はありません。このため、パブリッシャ・アプリケーションのロジックを変更しなくても、受信メッセージに新しいサブスクライバ・アプリケーションを動的に追加できます。サブスクライバ・アプリケーションは、メッセージを送信しているパブリッシャ・アプリケーションに関係なく、メッセージの内容に基づいてメッセージを受信します。このため、サブスクライバ・アプリケーションのロジックを変更しなくても、サブスクライバ・アプリケーションを動的に追加できます。サブスクライバ・アプリケーションは、メッセージ・プロパティまたはトピックのメッセージ内容に対してルールベースのサブスクリプション（予約購読）を定義することで、どのようなメッセージに関心があるのかを指定できます。システムは、ルールベースのサブスクリプションを使用して、パブリッシュされたメッセージの受信者を計算し、自動的にルーティングします。

パブリッシュ・サブスクライブ・モデルでは、メッセージはトピックに対してパブリッシュされ、トピックから受信されます。トピックは、AQjmsSession の CreateTopic メソッドを使用して作成されます。クライアントは、AQjmsSession の getTopic メソッドを使用して、事前に作成されたトピックに対するハンドルを取得する場合があります。

JMS でパブリッシュ・サブスクライブ・モデルの通信を使用するには、次の各手順を実行します。

- AQjmsDestination で start コールを使用して、トピックに対するエンキュー / デキューを有効にします。
- メッセージを保持するために 1 つ以上のトピックを設定します。これらのトピックは、関心がある領域またはサブジェクトを表す必要があります。たとえば、請求済注文情報を表すようにトピックを設定できます。
- **永続サブスクライバ**の集合を作成します。各サブスクライバは、受信を希望するメッセージ仕様（選択）を表すセレクタを指定する場合があります。NULL セレクタは、そのトピックに対してパブリッシュされたすべてのメッセージの受信をサブスクライバが希望していることを示します。
- サブスクライバは、ローカルまたはリモートのいずれかです。ローカル・サブスクライバは、メッセージがパブリッシュされるトピックと同じトピックに対して定義された永続サブスクライバです。リモート・サブスクライバは、特定のキューのサブスクライバとして定義された別のトピックまたはそのトピックに対する受信者です。リモート・サブスクライバを使用するには、ローカルおよびリモート・トピックの間に伝播を設定する必要があります。伝播の詳細は、第 9 章「管理インタフェース」を参照してください。
- セッションの createPublisher メソッドを使用してトピック・パブリッシャを作成します。メッセージは publish コールを使用してパブリッシュされます。メッセージは、トピックのすべてのサブスクライバ、またはトピックに対する指定された受信者のサブセットに対してパブリッシュされます。
- サブスクライバは、receive メソッドを使用して、トピックに関するメッセージを受信します。
- サブスクライバは、**メッセージ・リスナー**を使用して、非同期にメッセージを受信する場合もあります。**リモート・サブスクライバ**および**伝播**の概念は、JMS に対する Oracle の拡張機能です。

サンプル・シナリオ

BooksOnLine アプリケーションでは、すべての入力済注文情報が OE_bookedorders_topic に対してパブリッシュされます。東部地域の顧客の注文情報は ES.ES_bookedorders_topic にルーティングされ、西部地域の顧客の注文情報は WS.WS_bookedorders_topic にルーティングされます。また、重要な顧客に対するメッセージを追跡するために OE_bookedorders_topic にサブスクライブするアプリケーションもあります。後述するサンプル・コードを参照してください。

永続サブスクライバ

永続サブスクライバは、次のいずれかの方法で作成されます。

- クライアントが、セッションの createDurableSubscriber メソッドを使用して、永続サブスクライバを作成します。
- メッセージ・セレクタを使用して永続サブスクライバを作成します。これによって、クライアントは、サブスクライバに配信されるメッセージをセレクタにマッチするメッセージに制限できます。

ペイロード型 TextMessage、StreamMessage、BytesMessage、ObjectMessage、MapMessage を含むトピックのセレクタには、次のいずれか 1 つ以上を組み合せた式を含めることができます。

- JMS メッセージ・ヘッダー・フィールドまたはプロパティ

```
JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
```

- ユーザー定義のメッセージ・プロパティ

```
color IN ('RED', BLUE', 'GREEN') AND price < 30000
```

AdtMessage を含むトピックの場合、セレクタはメッセージ・ペイロード内容、優先順位または相関 ID に対する SQL 式である必要があります。

- 優先順位または相関 ID に対するセレクタは、次のように指定します。

```
priority < 3 AND corrid = 'Fiction'
```

- メッセージ・ペイロードに対するセレクタは、次のように指定します。

```
tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
```

セレクタ構文の詳細は、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の createDurableSubscriber を参照してください。

リモート・サブスクライバは、createRemoteSubscriber コールを使用して定義されます。リモート・サブスクライバは、リモート・トピックの特定のコンシューマまたはすべてのサブスクライバです。

リモート・サブスクライバは、AQjmsAgent 構造を使用して定義されます。AQjmsAgent は、名前およびアドレスで構成されます。名前は、リモート・トピックのコンシューマ名を参照します。アドレスは、次のようにしてリモート・トピックを参照します。

```
<schema>.<topic_name>[@dblink]
```

- リモート・トピックで特定のコンシューマに対してメッセージをパブリッシュするには、リモート・トピックでの受信者のサブスクライバ名が、AQjmsAgent の name フィールドに指定される必要があります。リモート・トピックは、AQjmsAgent の address フィールドに指定される必要があります。

- リモート・トピックのすべてのサブスクライバに対してメッセージをパブリッシュするには、AQjmsAgent の name フィールドを NULL に設定する必要があります。リモート・トピックは、AQjmsAgent の address フィールドに指定される必要があります。

サンプルとして使用している BooksOnLine アプリケーションには、1 つのローカル・サブスクライバ SUBS1 および次の 2 つのリモート・サブスクライバがあります。

- リモート・トピック WS.WS_bookedorders_topic での West_Shipping
- ES.ES_booked_orders_topic での East_Shipping

サンプル・コード

```
public void create_booked_orders_subscribers(TopicSession jms_session)
{
   Topic            topic;
   TopicSubscriber  tsubs;
   AQjmsAgent       agt_east;
   AQjmsAgent       agt_west;

   try
   {

     /* get a handle to the OE_bookedorders_topic */
     topic = ((AQjmsSession)jms_session).getTopic("OE",
                  "OR_bookedorders_topic");


     /* Create local subscriber - to track messages for some customers */
     tsubs = jms_session.createDurableSubscriber(topic, "SUBS1",
             "JMSPriority < 3 AND Customer = 'MARTIN'",
                   false);

     /* Create remote subscribers in the western and eastern region */
     agt_west = new AQjmsAgent("West_Shipping", "WS.WS_bookedorders_topic");

     ((AQjmsSession)jms_session).createRemoteSubscriber(topic, agt_west,
                    "Region = 'WESTERN'");

     agt_east = new AQjmsAgent("East_Shipping", "ES.ES_bookedorders_topic");

     ((AQjmsSession)jms_session).createRemoteSubscriber(topic, agt_east,
                    "Region = 'EASTERN'");

     /* schedule propagation between bookedorders_topic and
   WS_bookedorders_topic, ES.ES_bookedorders_topic */
     ((AQjmsDestination)topic).schedulePropagation(jms_session,
                     "WS.WS_bookedorders_topic",
```

```
                        null, null, null, null);

     ((AQjmsDestination)topic).schedulePropagation(jms_session,
                  "ES.ES_bookedorders_topic",
                     null, null, null, null);
   }
   catch (Exception ex)
   {
     System.out.println("Exception " + ex);
   }

}
```

トピック・パブリッシャ

メッセージは、トピック・パブリッシャを使用してパブリッシュされます。

トピック・パブリッシャは、セッションの createPublisher メソッドにトピックを渡すことによって作成されます。クライアントには、トピックを指定せずにトピック・パブリッシャを作成するオプションもあります。この場合、トピックは、パブリッシュ操作のたびに指定される必要があります。クライアントは、トピック・パブリッシャによって送信されたすべてのメッセージのデフォルト配信モード、優先順位および Time-To-Live を指定できます。また、クライアントは、これらのオプションをメッセージ単位で指定できます。

サンプル・シナリオおよびコード

BooksOnLine アプリケーションでは、入力済注文情報が OE.OE_bookedorders_topic に対してパブリッシュされます。

```
public void book_new_order(TopicSession jms_session, ObjectMessage obj_message)
{
   TopicPublisher  publisher;
   Topic           topic;

   try
   {
     /* get a handle to the booked_orders topic */
     topic = ((AQjmsSession) jms_session).getTopic("OE",
                    "OE_bookedorders_topic");

     publisher = jms_session.createPublisher(topic);

     publisher.publish(topic, obj_message);

     jms_session.commit();
   }
   catch (JMSException ex)
```

```
   {
     System.out.println("Exception: " + ex);
   }

}
```

BooksOnLine アプリケーションでは、各出荷地域は、対応する入力済注文情報トピック（WS_bookedorder_topic または ES_bookedorder_topic）からメッセージを受信します。ローカル・サブスクライバ SUBS1 は、OE_booked_orders_topic からメッセージを受信します。

```
public void get_martins_orders(TopicSession jms_session)
{
   Topic            topic;
   TopicSubscriber  tsubs;
   ObjectMessage    obj_message;
   BolCustomer      customer;
   BolOrder         new_order;
   String           state;
   int              i = 0;

   try
   {
     /* get a handle to the OE_bookedorders_topic */
     topic = ((AQjmsSession)jms_session).getTopic("OE",
                 "OE_bookedorders_topic");

     /* Create local subscriber - to track messages for some customers */
     tsubs = jms_session.createDurableSubscriber(topic, "SUBS1",
                "JMSPriority < 3 AND Customer = 'MARTIN'",
                   false);

     /* process 10 messages */
     for(i=0; i<10; i++)
     {
       /* wait for a message to show up in the topic */
       obj_message = (ObjectMessage)tsubs.receive(10);

       new_order = (BolOrder)obj_message.getObject();

       customer = new_order.getCustomer();
       state    = customer.getState();

       System.out.println("Order: " + i + " for customer " +
           customer.getName());
       jms_session.commit();
     }
```

```
   }
   catch (Exception ex)
   {
     System.out.println("Exception " + ex);
   }
}
```

受信者リスト

JMS パブリッシュ・サブスクライブ・モデルでは、クライアントは、トピックのすべてのサブスクライバにメッセージを送信するのではなく、明示的な受信者リストを指定できます。これらの受信者は、トピックの既存のサブスクライバである場合もあれば、そうでない場合もあります。受信者リストは、このメッセージのトピックのサブスクリプション・リストをオーバーライドします。受信者リストの概念は、JMS に対する Oracle の拡張です。

サンプル・シナリオおよびコード

優先順位が高いメッセージを、OE_bookedorders_topic のすべてのサブスクライバに対してパブリッシュするのではなく、東部地域の SUBS1 および Fedex_Shipping のみに送信するとします。

```
public void book_rush_order(TopicSession jms_session,
            ObjectMessage obj_message)
{

   TopicPublisher  publisher;
   Topic           topic;
   AQjmsAgent[]    recp_list = new AQjmsAgent[2];

   try
   {
     /* get a handle to the booked_orders topic */
     topic = ((AQjmsSession) jms_session).getTopic("OE",
                    "OE_bookedorders_topic");

     publisher = jms_session.createPublisher(null);

     recp_list[0] = new AQjmsAgent("SUBS1", null);
     recp_list[1] = new AQjmsAgent("Fedex_Shipping",
              "ES.ES_bookedorders_topic");

     publisher.setPriority (1);
     ((AQjmsTopicPublisher)publisher).publish(topic, obj_message, recp_list);

     jms_session.commit();

   }
```

```
   catch (Exception ex)
   {
     System.out.println("Exception: " + ex);
   }
}
```

トピック・レシーバ

受信者名が受信者リストに明示的に指定されていても、その受信者がキューのサブスクライバではない場合は、その受信者に送信されるメッセージは、トピック・レシーバを作成することによって受信できます。トピック・レシーバは JMS に対する Oracle の拡張です。

トピック・レシーバをメッセージ・セレクタを使用して作成することもできます。これによって、クライアントは、受信者に配信されるメッセージをセレクタにマッチするメッセージに制限できます。

トピック・レシーバのセレクタに対する構文は、QueueReceiver の構文と同じです。

サンプル・シナリオおよびコード

```
public void ship_rush_orders(TopicSession jms_session)
{
   Topic            topic;
   TopicReceiver    trec;
   ObjectMessage    obj_message;
   BolCustomer      customer;
   BolOrder         new_order;
   String           state;
   int              i = 0;

   try
   {
     /* get a handle to the OE_bookedorders_topic */
     topic = ((AQjmsSession)jms_session).getTopic("ES",
                  "ES_bookedorders_topic");

     /* Create local subscriber - to track messages for some customers */
     trec = ((AQjmsSession)jms_session).createTopicReceiver(topic,
                        "Fedex_Shipping",
                        null);

     /* process 10 messages */
     for(i = 0; i < 10; i++)
     {
       /* wait for a message to show up in the topic */
       obj_message = (ObjectMessage)trec.receive(10);

       new_order = (BolOrder)obj_message.getObject();
```

```
      customer = new_order.getCustomer();
      state    = customer.getState();

      System.out.println("Rush Order for customer " +
          customer.getName());
      jms_session.commit();
    }
  }
  catch (Exception ex)
  {
    System.out.println("Exception ex: " + ex);
  }
}
```

リモート・サブスクライバの場合、リモート・トピックでのサブスクライバ名が createRemoteSubscriber コールに明示的に指定されているときは、トピック・レシーバを使用してメッセージを受信できます。

```
public void get_westernregion_booked_orders(TopicSession jms_session)
{
  Topic            topic;
  TopicReceiver    trec;
  ObjectMessage    obj_message;
  BolCustomer      customer;
  BolOrder         new_order;
  String           state;
  int              i = 0;

  try
  {
    /* get a handle to the WS_bookedorders_topic */
    topic = ((AQjmsSession)jms_session).getTopic("WS",
                  "WS_bookedorders_topic");

    /* Create local subscriber - to track messages for some customers */
    trec = ((AQjmsSession)jms_session).createTopicReceiver(topic,
                       "West_Shipping",
                       null);

    /* process 10 messages */
    for(i = 0; i < 10; i++)
    {
      /* wait for a message to show up in the topic */
      obj_message = (ObjectMessage)trec.receive(10);

      new_order = (BolOrder)obj_message.getObject();
```

```
        customer = new_order.getCustomer();
        state    = customer.getState();

        System.out.println("Received Order for customer " +
            customer.getName());
        jms_session.commit();
      }
    }
    catch (Exception ex)
    {
      System.out.println("Exception ex: " + ex);
    }

}
```

サブスクライバ名が createRemoteSubscriber コールに指定されていない場合、クライアントがメッセージを受信するには、リモート・サイトで永続サブスクライバを使用する必要があります。

トピック・ブラウザ

トピック・ブラウザを使用すると、クライアントはメッセージを削除することなく、トピック上でメッセージを参照できます。このブラウズ用メソッドは、トピック・メッセージをスキャンするために使用される java.util.Enumeration を戻します。nextElement に対する最初のコールが、トピックのスナップショットを取得します。トピック・ブラウザも、メッセージをスキャンしているときに、オプションでメッセージをロックする場合があります。これは、メッセージに対する SELECT ... for UPDATE コマンドの場合と似ています。これによって、他のコンシューマがスキャン中のメッセージを削除することはなくなります。

トピック・ブラウザも、メッセージ・セレクタを使用して作成できます。これによって、クライアントは、コンシューマに配信されるメッセージをセレクタにマッチするメッセージに制限できるようになります。

トピック・ブラウザのためのセレクタは、次のすべての形式で許可されます。

- JMSMessageID = 'ID:23452345'。指定されたメッセージ ID（すべてのメッセージは接頭辞 ID: を持つ）を持つメッセージを取り出します。
- JMS メッセージ・ヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  JMSCorrelationID LIKE 'RE%'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

AdtMessage を含むキューの場合、セレクタはメッセージ・ペイロード内容、優先順位または相関 ID に対する SQL 式である必要があります。

- メッセージ ID に対するセレクタ - 特定のメッセージ ID を持つメッセージを取り出します。

  ```
  msgid = '23434556566767676'
  ```

  注意： この場合、メッセージ ID に、接頭辞 ID: を使用しないでください。

  優先順位または相関 ID に対するセレクタは、次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードに対するセレクタは、次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

トピックのコンシューマに関して、トピック・ブラウザを作成することができるのは永続サブスクライバのみです。

トピック・ブラウザは、パージ機能もサポートしています。これによって、トピック・ブラウザを使用するクライアントは、トピックに関する現行のブラウズ操作中に参照されたすべてのメッセージを廃棄できます。パージとは、参照済のすべてのメッセージを破壊的に受信することと同じです（トピック・サブスクライバを使用して削除した場合と似ています）。

削除のために、トピック・ブラウザの java.lang.Enumeration に対する nextElement() 操作へのコールを使用してクライアントに戻されたメッセージは、参照済とみなされます。クライアントがまだ参照していないメッセージは、パージ中は廃棄されません。パージ操作は、同じトピック・ブラウザに対して何度も実行される場合があります。

それ以外のすべての JMS メッセージ操作に関しては、トピック・ブラウザの作成に使用した JMS セッションがコミットされると、パージは正常に実行されます。セッションに対する操作がロールバックされた場合、パージ操作も取り消されます。

サンプル・シナリオおよびコード

BooksOnLine アプリケーションでは、すべての入力済注文情報が OE_booked_orders_topic に対してパブリッシュされます。クライアントは、選択されたメッセージをブラウズできます。

```
import oracle.jms.TopicBrowser;
// ...
public void browse_rush_orders(TopicSession jms_session)
{
    TopicBrowser    browser;
    Topic           topic;
    ObjectMessage   obj_message;
    BolOrder        new_order;
    Enumeration     messages;
    String          customer_name;

    try
    {
        /* get a handle to the OE_booked_orders_topic topic */
        topic = ((AQjmsSession) jms_session).getTopic("OE",
            "OE_booked_orders_topic");

        /* create a Browser to look at RUSH orders */
        browser = jms_session.createBrowser(
            topic, "SUBS1", "JMSCorrelationID = 'RUSH'");

        int count = 0;
        for (messages = browser.getEnumeration() ; messages.hasMoreElements() ;)
        {
            obj_message = (ObjectMessage)messages.nextElement();
            new_order = (BolOrder)obj_message.getObject();

            customer_name = new_order.getCustomer().getName();
            System.out.println("Customer " +  customer_name +
                " has placed a RUSH order");

            ++count;
        }

        /* purge messages seen during this browse if there are too many */
        if (count > 100)
        {
            browser.purgeSeen();
        }

        browser.close();
    }
```

```
    catch (Exception ex)
    {
    System.out.println("Exception " + ex);
    }
}
```

JMS メッセージ・プロデューサ機能

- メッセージの優先順位および順序付け
- 時間指定 : 遅延
- 時間指定 : 期限切れ
- メッセージのグループ化

メッセージの優先順位および順序付け

メッセージの順序付けとは、キューまたはトピックからメッセージが受信される順序を示します。優先順位の指定は、キューまたはトピックに対してキュー表が作成されたときに指定されます（9-4 ページの「キュー表の作成」を参照）。現在、AQ は、次の 2 つのメッセージ属性に対する順序付けをサポートしています。

- 優先順位
- エンキュー時刻

これらが組み合されると、次の 4 つの順序付け方法が可能になります。

メッセージの FIFO による順序付け　エンキュー時刻が順序付け基準として選択されると、メッセージはエンキュー時刻の順序で受信されます。エンキュー時刻は、メッセージのパブリッシュ / 送信時に AQ によってメッセージに割り当てられます。これはデフォルトの順序付けです。

メッセージの優先順位による順序付け　優先順位による順序付けが選択されると、各メッセージに優先順位が割り当てられます。優先順位は、パブリッシュ / 送信時にメッセージ・プロデューサによってメッセージ・プロパティとして指定できます。メッセージは、割り当てられた優先順位の順序で受信されます。

FIFO 優先順位による順序付け　FIFO 優先順位のトピック / キューも、優先順位およびエンキュー時刻の両方でメッセージのソート順を指定することで作成できます。FIFO 優先順位のトピック / キューは、優先順位による順序付けに似ていますが、同じ優先順位のメッセージが 2 つあるときにエンキュー時刻の順序で受信されるところが異なります。

エンキュー時刻に続く優先順位による順序付け　同じエンキュー時刻のメッセージは、優先順位の順序で受信されます。2 つのメッセージの順序付け基準が同じ場合、受信される順序は予想できません。ただし、AQ によって、同じセッション内で同じ順序付け基準で送信 / パブリッシュされたメッセージは、確実に送信された順序で受信されます。

サンプル・シナリオおよびコード

BooksOnLine アプリケーションを使用すると、顧客は次のいずれかの方法を要求できます。

- FedEx による出荷（優先順位 3）
- 優先扱い航空便による出荷（優先順位 2）
- 通常地上便による出荷（優先順位 1）

優先順位は、`setPriority` コールを使用してメッセージ・プロデューサ・レベルで指定するか、`send` または `publish` コール中に指定できます。後者が優先されます。

注文入力アプリケーションは、新規の注文情報を FIFO キューに格納します。新規の注文情報は、注文入力アプリケーションによって処理され、入力済注文情報トピックに対してパブリッシュされます。注文入力アプリケーションは、エンキュー時刻の順序で新規の注文情報キューからメッセージを取り出します。注文入力アプリケーションは、入力済注文情報を FIFO 優先順位のトピックに格納します。入力済注文情報は、地域の入力済注文情報トピックに伝播されます。各地域では、その地域の入力済注文情報トピックにある注文情報が、出荷の優先順位に従って処理されます。次のコールで注文入力アプリケーションに FIFO 優先順位トピックを作成して、入力済注文情報を格納します。

```
public static void createPriorityTopic(TopicSession jms_session)
{
    AQQueueTableProperty       qt_prop;
    AQQueueTable               pr_qtable;
    AQjmsDestinationProperty   dest_prop;
    Topic                      bookedorders_topic;

    try
    {

   /* Create a priority queue table for OE */
   qt_prop = new AQQueueTableProperty("SYS.AQ$_JMS_OBJECT_MESSAGE");
   qt_prop.setComment("Order Entry Priority " +
            "MultiConsumer Orders queue table");
   qt_prop.setCompatible("8.1");
   qt_prop.setMultiConsumer(true);

   /* Set a FIFO-priority order */
   qt_prop.setSortOrder("priority, enq_time");

   pr_qtable = ((AQjmsSession)jms_session).createQueueTable("OE",
                    "OE_orders_pr_mqtab", qt_prop);
```

```
   /* Create a Queue in this queue table */
   dest_prop = new AQjmsDestinationProperty();

   bookedorders_topic =((AQjmsSession)jms_session).createTopic(pr_qtable,
              "OE_bookedorders_topic", dest_prop);

   /* Enable enqueue and dequeue on the topic */
   ((AQjmsDestination)bookedorders_topic).start(jms_session, true, true);

    }
    catch (Exception ex)
    {
   System.out.println("Exception: " + ex);
    }
}

/* When an order arrives, the order entry application can use the following
   procedure to publish the order into its booked orders topic. A shipping
   priority is specified for each order: */
public static void order_enqueue(TopicSession jms_session, String book_title,
             int book_qty, int order_num, int cust_no,
             String cust_name, int ship_priority,
             String cust_state, String cust_country,
             String cust_order_type)
{
    BolOrder          order;
    BolCustomer       cust_data;
    BolBook           book_data;
    BolOrderItem[]    item_list;
    Topic             topic;
    ObjectMessage     obj_message;
    TopicPublisher    tpub;

    try
    {
      book_data = new BolBook(book_title);
      cust_data = new BolCustomer(cust_no, cust_name);

      order = new BolOrder(order_num, cust_data);

      item_list = new BolOrderItem[1];
      item_list[0] = new BolOrderItem(book_data, book_qty);

      order.setItemList(item_list);

      /* get a handle to the OE bookedorders_topic */
```

```
      topic = ((AQjmsSession)jms_session).getTopic("OE",
                    "OE_bookedorders_topic");

      /* Create the topic publisher */
      tpub = jms_session.createPublisher(topic);

      obj_message = jms_session.createObjectMessage();
      obj_message.setObject(order);

      /* Send message - specify priority */
      tpub.publish(topic, obj_message, DeliveryMode.PERSISTENT,
         ship_priority,0);

      jms_session.commit();
    }
    catch (Exception ex)
    {
      System.out.println("Exception ex: " + ex);
    }
}
```

時間指定 : 遅延

メッセージをキュー / トピックに対して送信 / パブリッシュするときに、**遅延**を指定できます。遅延は、そのメッセージがメッセージ・コンシューマに対して使用可能になるまでの時間を表します。遅延指定されたメッセージは、遅延の期限が切れて、使用可能になるまで待機状態になります。メッセージの遅延は、メッセージ・プロパティ（JMS_OracleDelay）として指定されます。このプロパティは、JMS 標準では指定されません。これは、JMS メッセージ・プロパティに対する AQ 拡張機能です。

遅延処理には、AQ 用のバックグラウンド・プロセス、キュー・モニターを起動する必要があります。msgid による受信で遅延がオーバーライドされることにも注意してください。

サンプル・シナリオおよびコード

BooksOnLine アプリケーションでは、遅延は遅延請求処理を実装するために使用できます。請求処理アプリケーションは 1 つのキューを定義して、そのキューの中には、出荷済ですぐには請求されない注文情報が遅延とともに入力されます。たとえば、法人顧客のような顧客アカウントの中には、15 日間経過するまで請求が発行されないクラスがあります。請求アプリケーションは、受け取った出荷済注文情報メッセージを（shippedorders キューから）デキューし、それが法人顧客からの注文である場合は、遅延とともに遅延請求処理キューにエンキューします。サンプル・シナリオは提供されていませんが、遅延はパブリッシュについても同様に処理します。

```
public static void defer_billing(QueueSession jms_session,
             BolOrder deferred_order)
{
```

```
    Queue             def_bill_q;
    ObjectMessage     obj_message;
    QueueSender       qsender;

    try
    {
   /* get a handle to the deferred billing queue */
   def_bill_q = ((AQjmsSession)jms_session).getQueue("CBADM",
                  "deferbilling_que");

   /* Create the QueueSender */
   qsender = jms_session.createSender(def_bill_q);

   obj_message = jms_session.createObjectMessage();
   obj_message.setObject(deferred_order);

   /* Set Delay as 15 days
    * Delay is specified in seconds
         */
   obj_message.setIntProperty("JMS_OracleDelay", 15*60*60*24);

   qsender.send(obj_message);

   jms_session.commit();

    }
    catch (Exception ex)
    {
   System.out.println("Exception " + ex);
    }

}
```

時間指定 : 期限切れ

メッセージ・プロデューサは、期限切れ制限またはメッセージの Time-To-Live（TimeToLive としてコーディングされています）を指定できます。これによって、そのメッセージがメッセージ・コンシューマに対して使用可能な期間が定義されます。

Time-To-Live は、送信 / パブリッシュ時に指定するか、メッセージ・プロデューサの Time-To-Live 設定メソッドを使用して指定できます。前者が優先されます。Time-To-Live を実装するには、AQ 用のバックグラウンド・プロセス、キュー・モニターが実行されている必要があります。

サンプル・シナリオ

BooksOnLine アプリケーションでは、Time-To-Live は受注残処理にあてられる時間を制御するために使用できます。出荷処理アプリケーションは、処理できない書籍注文情報を受注残トピックに送ります。すべての受注残を 1 週間以内に必ず出荷するという方針であれば、1 週間という期限切れとともにメッセージを受注残トピックにパブリッシュできます。この場合、1 週間以内に処理されなかった受注残は例外トピックに移されて、EXPIRED というメッセージ状態になります。これは、受注残の出荷方針に従って未出荷になっている注文情報にフラグを付けるために使用できます。

サンプル・コード

```
/* Re-enqueue a back order into a back_order Topic and set a timeToLive of
   7 days;
   All back orders must be processed in 7 days or they are moved to the
   exception queue */
public static void requeue_back_order(TopicSession jms_session,
                  String sale_region, BolOrder back_order)
{
    Topic             back_order_topic;
    ObjectMessage     obj_message;
    TopicPublisher    tpub;
    long              timetolive;

    try
    {
   /* Look up a back order topic based on the region */
   if(sale_region.equals("WEST"))
   {
       back_order_topic = ((AQjmsSession)jms_session).getTopic("WS",
                    "WS_backorders_topic");
   }
   else if(sale_region.equals("EAST"))
   {
       back_order_topic = ((AQjmsSession)jms_session).getTopic("ES",
                    "ES_backorders_topic");
   }
```

```
    else
    {
        back_order_topic = ((AQjmsSession)jms_session).getTopic("OS",
                    "OS_backorders_topic");
    }

    obj_message = jms_session.createObjectMessage();
    obj_message.setObject(back_order);

    tpub = jms_session.createPublisher(null);


    /* Set message expiration to 7 days: */
    timetolive = 7*60*60*24*1000;          // specified in  milliseconds


    /* Publish the message */
    tpub.publish(back_order_topic, obj_message, DeliveryMode.PERSISTENT,
            1, timetolive);


    jms_session.commit();
     }
     catch (Exception ex)
     {
    System.out.println("Exception :" + ex);
     }
}
```

メッセージのグループ化

1 つのキュー / トピックに属しているメッセージをグループ化して 1 つのセットにし、一度に 1 コンシューマ以外は使用できないようにできます。そのためには、トランザクション処理でのメッセージのグループ化に対応したキュー表に、そのキュー / トピックを作成する必要があります（9-4 ページの「キュー表の作成」を参照）。1 つのグループに属するメッセージは、すべて同一のトランザクションで作成される必要があります。また、1 つのトランザクションで作成されるメッセージは、すべて同一のグループに属します。この機能を使用すると、複雑なメッセージを、複数の単純なメッセージにセグメント化できます。これは AQ 拡張機能であり、JMS 仕様の一部ではありません。

たとえば、あるキュー宛てのメッセージに請求書情報が含まれている場合、そのメッセージは、ヘッダーのメッセージ、詳細情報のメッセージ、フッターのメッセージで構成されるグループとして作成できます。小さいオブジェクトに分割できるイメージやビデオなどの複合 LOB がメッセージ・ペイロードにある場合は、メッセージのグループ化が非常に有効です。

グループに含まれるメッセージの一般的なメッセージ・プロパティ（優先順位、遅延、期限切れ）は、単にグループの最初のメッセージ（ヘッダー）のプロパティによってのみ判断され、グループの他のメッセージのプロパティは無視されます。

グループ化メッセージのプロパティは、伝播されても保持されます。ただし、メッセージが伝播される宛先トピックも、トランザクション処理でグループ化可能である必要があります。トランザクション処理でグループ化可能なキューからメッセージをデキューするときに、グループ化メッセージのプロパティを保持する場合、他にも認識しておく必要がある制限があります（詳細は、「デキューの方法」および「デキューのモード」を参照）。

サンプル・シナリオ

BooksOnLine アプリケーションでは、メッセージのグループ化は新規の注文を扱うために使用できます。各注文情報には、注文された多数の書籍が 1 つずつ連続して入っています。Web を経由して注文された項目も同様です。

次の例では、各 `send` が注文情報の中の個々の書籍に対応し、グループ / トランザクションが 1 つの完全な注文情報を表します。最初のメッセージにのみ、顧客情報が含まれています。`OE_neworders_que` は、トランザクショション処理でグループ化できるキュー表 `OE_orders_sqtab` に定義されることに注意してください。

サンプル・コード

```
public static void createMsgGroupQueueTable(QueueSession jms_session)
{
    AQQueueTableProperty       sqt_prop;
    AQQueueTable               sq_table;
    AQjmsDestinationProperty   dest_prop;
    Queue                      neworders_q;

    try
    {
   /* Create a single-consumer orders queue table
        * with message grouping = TRANSACTIONAL
        */
   sqt_prop = new AQQueueTableProperty("BOLADM.order_typ");
   sqt_prop.setComment("Order Entry Single-Consumer Orders queue table");
   sqt_prop.setCompatible("8.1");
   sqt_prop.setMessageGrouping(AQQueueTableProperty.TRANSACTIONAL);

   sq_table = ((AQjmsSession)jms_session).createQueueTable("OE",
                "OE_orders_sqtab", sqt_prop);

   /* Create new orders queue for OE */
   dest_prop = new AQjmsDestinationProperty();
   neworders_q = ((AQjmsSession)jms_session).createQueue(sq_table,
                          "OE_neworders_que",
                     dest_prop);

    }
    catch (Exception ex)
    {
   System.out.println("Exception: " + ex);
    }
}

/* This method send an order to the specified queue */
public static void enqueue_order(QueueSession jms_session, Queue queue,
                int order_num, String cust_name, int cust_id,
            int book_qty, String book_title)
{
   QueueSender       sender;
   ObjectMessage     obj_message;
   BolOrder          order;
   BolCustomer       cust_data=null;
   BolBook           book_data;
   BolOrderItem[]    item_list;

   try
```

```
    {
      book_data = new BolBook(book_title);

      if(cust_name != null)
      {
        cust_data = new BolCustomer(cust_id, cust_name);
      }

      order = new BolOrder(order_num, cust_data);

      item_list = new BolOrderItem[1];
      item_list[0] = new BolOrderItem(book_data, book_qty);

      order.setItemList(item_list);


      sender = jms_session.createSender(queue);

      obj_message = jms_session.createObjectMessage();

      obj_message.setObject(order);

      sender.send(obj_message);
    }
    catch (Exception ex)
    {
      System.out.println("Exception ex: " + ex);
    }
}

/* Enqueue groups of orders */
public static void enqueue_order_groups(QueueSession jms_session)
{
  Queue neworders_q;

  try
  {
    neworders_q = ((AQjmsSession)jms_session).getQueue("OE",
                         "OE_neworders_que");

    /* Enqueue first group */
    enqueue_order(jms_session, neworders_q, 1, "John", 1000, 2,
        "John's first book");

    enqueue_order(jms_session, neworders_q, 1, null, 0, 1,
        "John's second book");
```

```
    jms_session.commit();

    /* Enqueue second group */
    enqueue_order(jms_session, neworders_q, 2, "Mary", 1001, 1,
        "Mary's first book");

    enqueue_order(jms_session, neworders_q, 2, null, 0, 1,
        "Mary's second book");

    enqueue_order(jms_session, neworders_q, 2, null, 0, 1,
        "Mary's third book");

    jms_session.commit();

    /* Enqueue third group */
    enqueue_order(jms_session, neworders_q, 3, "Scott", 1002, 1,
        "Scott's first book");

    enqueue_order(jms_session, neworders_q, 3, null, 0, 2,
        "Scott's second book");

    enqueue_order(jms_session, neworders_q, 3, null, 0, 2,
        "Scott's third book");

    jms_session.commit();
  }
  catch (Exception ex)
  {
    System.out.println("Exception ex: " + ex);
  }

}
```

メッセージ・コンシューマ機能

- メッセージの受信
- 受信におけるメッセージのナビゲーション
- メッセージ受信モード
- 遅延間隔をおける再試行
- メッセージ・リスナーを使用したメッセージの非同期受信
- AQ の例外処理

メッセージの受信

JMS アプリケーションは、メッセージ・コンシューマを作成することによって、メッセージを受信できます。メッセージは、receive コールを使用して同期に受信できるか、メッセージ・リスナーを使用して非同期に受信できます。

受信モードには次の 3 つがあります。

- メッセージがコンシューマに届くまでブロック
- 指定時間までブロック
- 非ブロック

サンプル・コード : メッセージが届くまでブロック

```
public BolOrder get_new_order1(QueueSession jms_session)
    {
      Queue            queue;
      QueueReceiver    qrec;
      ObjectMessage    obj_message;
      BolCustomer      customer;
      BolOrder         new_order = null;
      String           state;

      try
      {
      /* get a handle to the new_orders queue */
       queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");

       qrec = jms_session.createReceiver(queue);

       /* wait for a message to show up in the queue */
       obj_message = (ObjectMessage)qrec.receive();

       new_order = (BolOrder)obj_message.getObject();
```

```
       customer = new_order.getCustomer();
       state    = customer.getState();

       System.out.println("Order:  for customer " +
                          customer.getName());

      }
      catch (JMSException ex)
      {
        System.out.println("Exception: " + ex);
      }
      return new_order;

   }
```

サンプル・コード : 最長 60 秒間のブロック

```
public BolOrder get_new_order2(QueueSession jms_session)
    {
      Queue            queue;
      QueueReceiver    qrec;
      ObjectMessage    obj_message;
      BolCustomer      customer;
      BolOrder         new_order = null;
      String           state;

      try
      {
            /* get a handle to the new_orders queue */
       queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");

       qrec = jms_session.createReceiver(queue);

       /* wait for 60 seconds for a message to show up in the queue */
       obj_message = (ObjectMessage)qrec.receive(60000);

       new_order = (BolOrder)obj_message.getObject();

       customer = new_order.getCustomer();
       state    = customer.getState();

       System.out.println("Order:  for customer " +
                          customer.getName());

      }
      catch (JMSException ex)
```

```
      {
        System.out.println("Exception: " + ex);
      }
      return new_order;

   }
```

サンプル・コード : 非ブロック

```
public BolOrder poll_new_order3(QueueSession jms_session)
    {
      Queue            queue;
      QueueReceiver    qrec;
      ObjectMessage    obj_message;
      BolCustomer      customer;
      BolOrder         new_order = null;
      String           state;

      try
      {
            /* get a handle to the new_orders queue */
       queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");

       qrec = jms_session.createReceiver(queue);

       /* check for a message to show in the queue */
       obj_message = (ObjectMessage)qrec.receiveNoWait();

       new_order = (BolOrder)obj_message.getObject();

       customer = new_order.getCustomer();
       state    = customer.getState();

       System.out.println("Order:  for customer " +
                          customer.getName());

      }
      catch (JMSException ex)
      {
        System.out.println("Exception: " + ex);
      }
      return new_order;

   }
```

受信におけるメッセージのナビゲーション

コンシューマは、そのセッションで最初に受信するとき、キューまたはトピックで最初のメッセージを取得します。次の受信で次のメッセージを取得し、以後同様に続きます。デフォルトの動作は、FIFO キューおよびトピックでは正常に動作しますが、優先順位によって順序付けられたキューでは正常に動作しません。優先順位が高いメッセージがコンシューマに届く場合、すでに届いているメッセージを削除するまで、このクライアント・プログラムはこのメッセージを受信しません。

コンシューマが、メッセージに対してより効率的にキューのナビゲーションを制御できるように、AQ ナビゲーション・モードが JMS 拡張機能として用意されています。これらのモードは、トピック・サブスクライバ、キュー・レシーバまたはトピック・レシーバで設定できます。

- FIRST_MESSAGE は、コンシューマの位置をキューの先頭にリセットします。このモードでは、コンシューマはキューの一番上にあるメッセージを削除できるため、優先順位によって順序付けられたキューに有効です。
- NEXT_MESSAGE は、確立されたコンシューマの位置の後のメッセージを取得します。たとえば、位置が 4 番目のメッセージに確立された後で発行された NEXT_MESSAGE は、そのキューの 2 番目のメッセージを取得します。これはデフォルトの動作です。

トランザクションのグループ化については次のとおりです。

- FIRST_MESSAGE は、コンシューマの位置をキューの先頭にリセットします。
- NEXT_MESSAGE は、位置を同一トランザクションの次のメッセージに設定します。
- NEXT_TRANSACTION は、位置を次のトランザクションの最初のメッセージに設定します。

次の方法でメッセージが受信される場合、グループ化トランザクションのプロパティは無効になる場合があります。

- セレクタに相関識別子を指定する受信
- セレクタにメッセージ識別子を指定する受信
- トランザクション・グループのメッセージがすべて受信される前のコミット

キューのナビゲート中に、NEXT_MESSAGE または NEXT_TRANSACTION オプションを使用したプログラムがキューの最後に到達したとします。ブロッキング受信を指定していた場合は、ナビゲート位置は自動的にそのキューの先頭に変更されます。

デフォルトでは、キュー・レシーバ、トピック・レシーバまたはトピック・サブスクライバは、最初の receive コールに FIRST_MESSAGE を、次の receive コールに NEXT_MESSAGE を使用します。

サンプル・シナリオ

get_new_orders() プロシージャは、OE_neworders_que から注文情報を取り出します。各トランザクションはそれぞれの注文情報を参照し、各メッセージはその注文情報に含まれる各書籍に対応しています。get_orders() プロシージャは、メッセージをループして書籍注文情報を取り出します。このプロシージャは、最初の受信の前に、FIRST_MESSAGE オプションを使用して、位置をキューの先頭にリセットします。その後、next message ナビゲーション・オプションで、注文情報（トランザクション）から次の書籍（メッセージ）を取り出します。現行のグループまたはトランザクションのすべてのメッセージがフェッチされたという例外が戻された後、ナビゲーション・オプションを next_transaction に変更して、次の注文情報の最初の書籍を取り出します。次に、next message オプションに戻して、同一トランザクションの次のメッセージをフェッチします。すべての注文情報（トランザクション）がフェッチされるまで、この処理を繰り返します。

サンプル・コード

```
public void get_new_orders(QueueSession jms_session)
    {
      Queue            queue;
      QueueReceiver    qrec;
      ObjectMessage    obj_message;
      BolCustomer      customer;
      BolOrder         new_order;
      String           state;
      int              new_orders = 1;

      try
      {

         /* get a handle to the new_orders queue */
         queue = ((AQjmsSession) jms_session).getQueue("OE","OE_neworders_que");
         qrec = jms_session.createReceiver(queue);

    /* set navigation to first message */
        ((AQjmsTopicSubscriber)qrec).setNavigationMode(AQjmsConstants.NAVIGATION_
FIRST_MESSAGE);

        while(new_orders != 0)
        {
          try{

            /* wait for a message to show up in the topic */
            obj_message = (ObjectMessage)qrec.receiveNoWait();

            if (obj_message != null)   /* no more orders in the queue */
            {
              System.out.println(" No more orders ");
```

```
              new_orders = 0;
            }
            new_order = (BolOrder)obj_message.getObject();
            customer = new_order.getCustomer();
            state    = customer.getState();

            System.out.println("Order: for customer " +
                               customer.getName());

           /* Now get the next message */

((AQjmsTopicSubscriber)qrec).setNavigationMode(AQjmsConstants.NAVIGATION_NEXT_
MESSAGE);

          }catch(AQjmsException ex)
          {  if (ex.getErrorNumber() == 25235)
             {
               System.out.println("End of transaction group");

((AQjmsTopicSubscriber)qrec).setNavigationMode(AQjmsConstants.NAVIGATION_NEXT_
TRANSACTION);
             }
             else
               throw ex;
          }
        }
     }catch (JMSException ex)
     {
        System.out.println("Exception: " + ex);
      }
   }
```

メッセージ受信モード

Point-to-Point モード

デキューするクライアントがキューからメッセージを削除できる通常の受信の他に、JMS では、JMS クライアントがキューで自身に対するメッセージをブラウズできるようにするインタフェースを提供しています。キュー・ブラウザは、キュー・セッションから `createBrowser` メソッドを使用して作成できます。

メッセージは、ブラウズ後でも処理可能です。メッセージは、ブラウズされた後は、同時セッションからの `receive` コールがそのメッセージを削除する場合があるため、JMS セッションに再使用できる保証はないことに注意してください。

一度参照したメッセージが同時 JMS クライアントによって削除されないようにするために、ロック・モードでメッセージを参照できます。このためには、JMS インタフェースに対する AQ 拡張機能を使用して、ロック・モードを持つキュー・ブラウザを作成する必要があります。ロック・モードを持つブラウザによるメッセージのロックは、セッションがコミットまたはロールバックを実行したときに解放されます。

キュー・ブラウザによって参照されたメッセージを削除するには、セッションがキュー・レシーバを作成し、`JMSmesssageID` をセレクタとして使用する必要があります。

サンプル・コード

Point-to-Point 機能のキュー・ブラウザの例を参照してください。

取出しを伴わないメッセージの削除

メッセージ・コンシューマは、`receiveNoData` コールを使用して、メッセージを取り出さずにキューまたはトピックから削除できます。これは、アプリケーションがキュー・ブラウザを使用してすでにメッセージを調べている場合に有効です。このモードによって、JMS クライアントは、データベースからペイロードを取り出す場合のオーバーヘッドを回避できます。このオーバーヘッドは、大量のメッセージでは相当量になる可能性があります。

サンプル・シナリオおよびコード

次の BooksOnLine のシナリオでは、海外からの注文情報（メキシコおよびカナダ向け）が、通商政策および送料割引の理由で別々に処理されます。したがって、（他の同時ユーザーが削除できないように）メッセージをキュー・ブラウザを使用してロック・モードで参照し、顧客の国（メッセージ・ペイロード）を確認します。顧客の国がメキシコまたはカナダの場合、（ペイロードはすでにわかっているため）`remove with no data` モードでメッセージをキューから削除します。これを実行しないと、そのメッセージのロックはコミット・コールによって解除されるためです。`receive` コールには、ロック・モードのブラウザから取得したメッセージ識別子を使用することに注意してください。

```
public void process_international_orders(QueueSession jms_session)
 {
  QueueBrowser    browser;
```

```
   Queue           queue;
   ObjectMessage   obj_message;
   BolOrder        new_order;
   Enumeration     messages;
   String          customer_name;
   String          customer_country;
   QueueReceiver   qrec;
   String          msg_sel;

   try
   {
     /* get a handle to the new_orders queue */
     queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");

     /* create a Browser to look at RUSH orders */
     browser = ((AQjmsSession)jms_session).createBrowser(queue, null, true);

     for (messages = browser.getEnumeration() ; messages.hasMoreElements() ;)
     {
       obj_message = (ObjectMessage)messages.nextElement();

       new_order = (BolOrder)obj_message.getObject();

       customer_name = new_order.getCustomer().getName();

       customer_country = new_order.getCustomer().getCountry();

       if (customer_country equals ("Canada") || customer_country equals (
"Mexico"))
       {
         System.out.println("Order for Canada or Mexico");
         msg_sel = "JMSMessageID = '" + obj_message. getJMSMessageID()+ "'";
         qrec = jms_session.createReceiver(queue, msg_sel);
         ((AQjmsQueueReceiver)qrec).receiveNoData();
       }
     }
   }catch (JMSException ex)
    { System.out.println("Exception " + ex);

    }
  }
```

遅延間隔をおける再試行

再試行の最大値

キュー / トピックからメッセージを受信するトランザクションが失敗した場合、そのメッセージを削除する試行に失敗したとみなされます。AQ は、メッセージ削除の試行に失敗した回数をメッセージ履歴に記録します。

さらに、AQ では、アプリケーションがメッセージに対する再試行の最大回数を、キュー / トピック・レベルで指定できます。メッセージの削除がこの数よりも多く失敗した場合、メッセージは例外キューに移され、アプリケーションで使用できなくなります。

再試行の遅延

メッセージを受信するトランザクションが異常終了した場合、たとえば、書籍が在庫不足のため注文を受けることができなかったなどの、不十分な状態が原因である場合があります。在庫更新は 12 時間ごとに行われているため、更新後に再試行することも有効です。4 回試行しても注文を受けられなかった場合は、問題がある可能性があります。

AQ では、ユーザーは max_retries とともに retry_delay も指定できます。これは、受信の試行に失敗したメッセージを、retry_delay 間隔後に引き続きキューで参照し、デキューできることを意味します。そのときまで、このメッセージは WAITING 状態になります。AQ バックグラウンド・プロセスであるタイム・マネージャは、再試行のディレイ・プロパティを施行します。

再試行の最大回数および再試行の遅延は、キュー / トピックのプロパティです。このプロパティは、キュー / トピックの作成時、またはキュー / トピックに対する変更メソッドを使用して設定できます。max_retries のデフォルト値は 5 です。

サンプル・シナリオおよびコード

在庫不足のため注文を受けることができない場合、その注文を処理するトランザクションは異常終了されます。booked_orders トピックは、max_retries = 4 時間および retry_delay = 12 時間で設定されます。したがって、注文は、2 日間履行されなければ、例外キューに移されます。

```
public BolOrder process_booked_order(TopicSession jms_session)
  {
    Topic            topic;
    TopicSubscriber  tsubs;
    ObjectMessage    obj_message;
    BolCustomer      customer;
    BolOrder         booked_order = null;
    String           country;
    int              i = 0;

    try
    {
```

```
   /* get a handle to the OE_bookedorders_topic */
   topic = ((AQjmsSession)jms_session).getTopic("WS",
                                                "WS_bookedorders_topic");

   /* Create local subscriber - to track messages for Western Region  */
   tsubs = jms_session.createDurableSubscriber(topic, "SUBS1",
                                    "Region = 'Western' ",
                                                false);

    /* wait for a message to show up in the topic */
    obj_message = (ObjectMessage)tsubs.receive(10);

    booked_order = (BolOrder)obj_message.getObject();

    customer = booked_order.getCustomer();
    country    = customer.getCountry();

    if (country == "US")
    {
       jms_session.commit();
    }
    else
    {
       jms_session.rollback();
       booked_order = null;
    }
 }catch (JMSException ex)
 { System.out.println("Exception " + ex) ;}

  return booked_order;
}
```

メッセージ・リスナーを使用したメッセージの非同期受信

メッセージ・コンシューマ用のメッセージ・リスナー

JMS クライアントは、コンシューマで使用可能な setMessageListener メソッドを使用してメッセージ・リスナーを設定することによって、メッセージを非同期に受信できます。

メッセージ・コンシューマにメッセージが届いた場合、メッセージ・リスナーの onMessage メソッドがそのメッセージで起動されます。メッセージ・リスナーは、コミットするか、メッセージの受信を異常終了できます。メッセージ・リスナーは、JMS コネクションが停止されている場合、メッセージを受信しません。一度メッセージ・リスナーがコンシューマに対して設定されると、メッセージの受信に receive コールを使用することはできません。

サンプル・コード

新規注文のキューを処理するアプリケーションは、そのキューからメッセージを非同期受信するように設定できます。

```
public class OrderListener implements MessageListener
  {
     QueueSession   the_sess;

     /* constructor */
     OrderListener(QueueSession  my_sess)
     {
       the_sess = my_sess;
     }

     /* message listener interface */
     public void onMessage(Message m)
     {
       ObjectMessage    obj_msg;
       BolCustomer      customer;
       BolOrder         new_order = null;

       try {
         /* cast to JMS Object Message */
         obj_msg = (ObjectMessage)m;

         /*  Print some useful information */
         new_order = (BolOrder)obj_msg.getObject();
         customer = new_order.getCustomer();
         System.out.println("Order:  for customer " +  customer.getName());

         /* call the process order method
         * NOTE: we are assuming it is defined elsewhere
```

```
        * /
        process_order(new_order);

        /* commit the asynchronous receipt of the message */
         the_sess.commit();
       }catch (JMSException ex)
       { System.out.println("Exception " + ex) ;}

    }
}

  public void setListener1(QueueSession jms_session)
 {
   Queue            queue;
   QueueReceiver    qrec;
   MessageListener  ourListener;

   try
   {
    /* get a handle to the new_orders queue */
    queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");

    /* create a queue receiver */
    qrec = jms_session.createReceiver(queue);

    /* create the message listener */
    ourListener = new OrderListener(jms_session);

    /* set the message listener for the receiver */
    qrec.setMessageListener(ourListener);
   }
   catch (JMSException ex)
   {
     System.out.println("Exception: " + ex);
   }
}
```

セッションのすべてのコンシューマ用のメッセージ・リスナー

JMS クライアントは、セッションでメッセージ・リスナーを設定することによって、そのセッションのすべてのコンシューマに対してメッセージを非同期に受信できます。

セッションのどのメッセージ・コンシューマにメッセージが届いても、メッセージ・リスナーの onMessage メソッドがそのメッセージで起動されます。メッセージ・リスナーは、コミットするか、メッセージの受信を異常終了できます。メッセージ・リスナーは、JMS コネクションが停止されている場合、メッセージを受信しません。一度メッセージ・リスナーが設定されると、そのセッションでは、その他のメッセージ受信モードを使用できません。

サンプル・シナリオおよびコード

BooksOnLine シナリオの顧客サービス・コンポーネントでは、異なるデータベースから顧客サービス・トピックに届いたメッセージにはその状態が示されます。顧客サービス・アプリケーションはそのトピックを監視し、顧客の注文情報に関するメッセージを検出するたびに、order_status_table の注文情報状態を更新します。アプリケーションはセッション・リスナーを使用して様々なトピックを監視します。トピックのいずれかにメッセージがあるときは、常に、セッション・メッセージ・リスナーの onMessage メソッドが起動されます。

```
/* define our message listener class */
   public class CustomerListener implements MessageListener
   {
      TopicSession   the_sess;

      /* constructor */
      CustomerListener(TopicSession  my_sess)
      {
        the_sess = my_sess;
      }

      /* message listener interface */
      public void onMessage(Message m)
      {
        ObjectMessage    obj_msg;
        BolCustomer      customer;
        BolOrder         new_order = null;

        try
   {
          /* cast to JMS Object Message */
          obj_msg = (ObjectMessage)m;

          /*  Print some useful information */
          new_order = (BolOrder)obj_msg.getObject();
          customer = new_order.getCustomer();
          System.out.println("Order:  for customer " +  customer.getName());

          /* call the update status method
           * NOTE: we are assuming it is defined elsewhere
           * /
          update_status(new_order, new_order.getStatus());

          /* commit the asynchronous receipt of the message */
          the_sess.commit();
       }catch (JMSException ex)
      {
        System.out.println("Exception: " + ex);
```

```
    }
  }

}
public void  monitor_status_topics(TopicSession jms_session)
 {
   Topic[]             topic = new Topic[4];
   TopicSubscriber[]   tsubs= new TopicSubscriber[4];

   try
   {
     /* get a handle to the OE_bookedorders_topic */
     topic[0] = ((AQjmsSession)jms_session).getTopic("CS",
                                                "CS_bookedorders_topic");
     tsubs[0] = jms_session.createDurableSubscriber(topic[0], "BOOKED_ORDER");

     topic[1] = ((AQjmsSession)jms_session).getTopic("CS",
                                                "CS_billedorders_topic");
     tsubs[1] = jms_session.createDurableSubscriber(topic[1], "BILLED_ORDER");

     topic[2] = ((AQjmsSession)jms_session).getTopic("CS",
                                                "CS_backdorders_topic");
     tsubs[2] = jms_session.createDurableSubscriber(topic[2], "BACKED_ORDER");

     topic[3] = ((AQjmsSession)jms_session).getTopic("CS",
                                                "CS_shippedorders_topic");
     tsubs[3] = jms_session.createDurableSubscriber(topic[3], "SHIPPED_ORDER");

     MessageListener  mL = new CustomerListener(jms_session);

     /* set the session's message listener */
     jms_session.setMessageListener(mL);

   }catch(JMSException ex)
    { System.out.println("Exception: " + ex); }
 }
```

AQ の例外処理

AQ は、例外キュー、期限切れ、再試行の最大数および再試行の遅延の 4 つの統合化メカニズムによって、アプリケーションの例外処理をサポートします。

例外キューは、期限切れまたは処理できないすべてのメッセージのリポジトリになります。アプリケーションから例外キューには直接エンキューできません。ただし、期限切れまたは処理できないメッセージを処理するアプリケーションは、例外キューからこれらのメッセージを受信または削除できます。

例外キューからメッセージを取り出すには、JMS クライアントは Point-to-Point インタフェースを使用する必要があります。トピック用のメッセージの例外キューは、使用可能な複数のコンシューマでキュー表に作成する必要があります。他のキューと同様に、例外キューも AQOracleQueue クラスで start メソッドを使用して、メッセージを受信できる必要があります。例外キューをエンキュー可能に設定しようとすると、例外が発生します。

例外キューは、「JMS_OracleExcpQ」と呼ばれるプロバイダ（Oracle）固有のメッセージ・プロパティです。これは、メッセージの送信 / パブリッシュ前に、そのメッセージで設定できます。例外キューが指定されていないと、デフォルトの例外キューが使用されます。キュー表（QTAB）にキュー / トピックが作成された場合、デフォルトの例外キューは AQ$_QTAB_E です。デフォルトの例外キューは、キュー表が作成されると自動的に作成されます。

メッセージは、次の条件が成立するときに AQ によって例外キューに移されます。

- そのメッセージが、指定された Time-To-Live 内にデキューされない場合。複数のサブスクライバを指定したメッセージの場合、指定されていながら指定された Time-To-Live 内にそのメッセージをデキューできない受信者が 1 つでもあると、例外キューに移されます。Time-To-Live がメッセージに（publish または send コールでも、またはパブリッシャまたは送信者としても）指定されていない場合、メッセージは期限切れになりません。
- メッセージが正常に受信されたが、メッセージ処理中のエラーのため、アプリケーションが受信を実行したトランザクションを異常終了した場合。そのメッセージはキュー / トピックに戻され、メッセージ受信のために待機中のどのアプリケーションでも使用可能になります。これは失敗したメッセージ受信の試行であるため、その再試行回数は更新されます。

  メッセージの再試行回数が、メッセージが常駐するキュー / トピックに指定された最大値を超える場合、そのメッセージは例外キューに移されます。メッセージが複数のサブスクライバを持つ場合、そのメッセージは、すべての受信者が再試行制限を超えたときにのみ、例外キューに移されます。

  アプリケーションがトランザクション全体を異常終了するか、受信前のセーブポイントまでロールバックしたときは、受信処理がロールバックまたは UNDO されたとみなされます。
- クライアント・プログラムがメッセージの受信に成功したにもかかわらず、トランザクションをコミットする前に終了した場合。

サンプル・シナリオ

遅延間隔をおいたセクションの再試行には、MAX_RETRIES の例があります。BooksOnLine アプリケーションでは、各出荷地域のビジネス・ルールによって、すぐに応じることができない注文情報は受注残キューに移されます。受注残アプリケーションは、毎日 1 回その注文情報を出荷しようとします。7 日以内に出荷できない場合、その注文情報は例外キューに移されて特別に処理されます。例外キューにメッセージの Time-To-Live プロパティを使用して、この処理を実装します。

1. 例外キュー WS_back_order_exp_que を作成します。

```
public void create_excp_que(TopicSession jms_session)
    {
      AQQueueTable     q_table;
      Queue            excpq;

      try {
         /* create the exception queue in the queue table with multiple
          * consumer flag true
          */
          q_table = ((AQjmsSession)jms_session).getQueueTable("WS", "WS_orders_
mqtab");

          AQjmsDestinationProperty dest_prop = new AQjmsDestinationProperty();

          dest_prop.setQueueType(AQjmsDestinationProperty.EXCEPTION_QUEUE);
     excpq = ((AQjmsSession)jms_session).createQueue(q_table,
                              "WS_back_orders_excp_que",
                         dest_prop);
          /* start the exception queue for receiving (dequeuing) messages only */
          ((AQjmsDestination)excpq).start(jms_session, false, true);

          }
      catch (JMSException ex)
      { System.out.println("Exception  " + ex); }
    }
```

2. 注文残キューに関するメッセージを、WS_back_orders_excp_que に設定した例外キューとともにパブリッシュします。

```
 public static void requeue_back_order(TopicSession jms_session,
                  String sale_region, BolOrder back_order)
 {
    Topic             back_order_topic;
    ObjectMessage     obj_message;
    TopicPublisher    tpub;
    long              timetolive;
```

```
    try
    {
   back_order_topic = ((AQjmsSession)jms_session).getTopic("WS",
                     "WS_backorders_topic");
   obj_message = jms_session.createObjectMessage();
   obj_message.setObject(back_order);

        /* set exception queue */
        obj_message.setStringProperty("JMS_OracleExcpQ", "WS.WS_back_orders_excp_
que");

   tpub = jms_session.createPublisher(null);

   /* Set message expiration to 7 days: */
   timetolive = 7*60*60*24*1000;          // specified in  milliseconds

   /* Publish the message */
   tpub.publish(back_order_topic, obj_message, DeliveryMode.PERSISTENT,
          1, timetolive);
   jms_session.commit();
    }
    catch (Exception ex)
    {
   System.out.println("Exception :" + ex);
    }
}
```

3. Point-to-Point インタフェースを使用して、例外キューから期限切れメッセージを受信します。

```
public BolOrder get_expired_order(QueueSession jms_session)
   {
      Queue            queue;
      QueueReceiver    qrec;
      ObjectMessage    obj_message;
      BolCustomer      customer;
      BolOrder         exp_order = null;

      try
      {
      /* get a handle to the exception queue */
       queue = ((AQjmsSession) jms_session).getQueue("WS", "WS_back_orders_excp_
que");

       qrec = jms_session.createReceiver(queue);

       /* wait for a message to show up in the queue */
       obj_message = (ObjectMessage)qrec.receive();
```

```
        exp_order = (BolOrder)obj_message.getObject();

        customer = exp_order.getCustomer();

        System.out.println("Expired Order:  for customer " +
                           customer.getName());

      }
      catch (JMSException ex)
      {
        System.out.println("Exception: " + ex);
      }
      return exp_order;
    }
```

JMS 伝播

- リモート・サブスクライバ
- 伝播スケジュール
- 拡張伝播スケジュール機能
- 伝播中の例外処理

リモート・サブスクライバ

この機能によって、同じデータベースに接続しなくても、アプリケーション同士が互いに通信できます。

AQ を使用すると、リモート・サブスクライバ（他のデータベースにあるサブスクライバ）がトピックからサブスクライブできます。トピックに対してパブリッシュされたメッセージがリモート・サブスクライバの基準を満たしている場合、AQ はリモート・サブスクライバに指定されているリモート・データベースにあるキュー / トピックにメッセージを自動的に伝播します。

スナップショット（ジョブ・キュー）のバックグラウンド・プロセスが伝播を実行します。伝播は、データベース・リンクおよび Oracle Net Services を使用して実行されます。

リモート・サブスクライバを実装するには、次の 2 つの方法があります。

- createRemoteSubscriber メソッドは、トピック上またはトピックに対してリモート・サブスクライバを作成するために使用します。このリモート・サブスクライバは、クラス AQjmsAgent のインスタンスとして指定されます。
- AQjmsAgent には、名前およびアドレスがあります。アドレスは、キュー / トピックおよびサブスクライバのデータベースへのデータベース・リンク（dblink）で構成されます。

リモート・サブスクライバには、次の 2 種類があります。

ケース 1 リモート・サブスクライバがトピックである場合。これは、AQjmsAgent オブジェクトのリモート・サブスクライバに名前が指定されず、アドレスがトピックである場合に発生します。サブスクライバのサブスクリプションを満たすメッセージが、リモート・トピックに伝播されます。伝播されたメッセージは、それが満たすリモート・トピックのすべてのサブスクリプションに対して使用可能になります。

ケース 2 メッセージに対して特定のリモート受信者を指定する場合。リモート・サブスクリプションは、リモート・データベースにある特定のコンシューマに対して指定できます。リモート受信者の名前が（AQjmsAgent オブジェクトに）指定される場合、サブスクリプションを満たすメッセージが、その受信者専用のリモート・データベースに伝播されます。リモート・データベースにある受信者は、TopicReceiver インタフェースを使用してメッセージを取り出します。リモート・サブスクリプションは、Point-to-Point キューに対して指定することもできます。

ケース 1 のサンプル・シナリオ

注文入力アプリケーションおよび西部向け出荷処理アプリケーションが、db1 および db2 という異なるデータベース上にあるとします。また、データベース db1（注文入力データベース）からデータベース db2（西部向け出荷処理データベース）へのデータベース・リンク dblink_oe_ws があるとします。db2 にある WS_bookedorders_topic は、db1 にある OE_bookedorders_topic のリモート・サブスクライバです。

ケース 2 のサンプル・シナリオ

注文入力アプリケーションおよび西部向け出荷処理アプリケーションが、db1 および db2 という異なるデータベース上にあるとします。さらに、db1（ローカル注文入力データベース）から db2（西部向け出荷処理データベース）へのデータベース・リンク dblink_oe_ws があるとします。db2 の WS_bookedorders_topic にあるエージェント「優先順位」は、db1 にある OE_bookedorders_topic のリモート・サブスクライバです。WS_bookedorders_topic に伝播されたメッセージは、「優先順位」専用です。

```
public void remote_subscriber(TopicSession jms_session)
   {
     Topic            topic;
     ObjectMessage    obj_message;
     AQjmsAgent       remote_sub;
```

```
    try
    {
      /* get a handle to the OE_bookedorders_topic */
      topic = ((AQjmsSession)jms_session).getTopic("OE",
                                                   "OE_bookedorders_topic");
      /* create the remote subscriber, name unspecified  and address
       * the topic WS_booked_orders_topic at db2
       */
      remote_sub = new AQjmsAgent(null, "WS.WS_bookedorders_topic@dblink_oe_ws");

      /* subscribe for western region orders */
      ((AQjmsSession)jms_session).createRemoteSubscriber(topic, remote_sub, "Region
= 'Western' ");
    }
    catch (JMSException ex)
    { System.out.println("Exception :" + ex); }
    catch (java.sql.SQLException  ex1)
    {System.out.println("SQL Exception :" + ex1); }
  }
```

データベース db2 - 出荷処理データベース : WS_booked_orders_topic は 2 つのサブスクライバを持ち、1 つは優先順位による出荷用で、もう 1 つは通常の出荷用です。注文入力データベースからのメッセージは、出荷処理データベースに伝播され、正しいサブスクライバに配信されます。優先順位による注文には、優先順位が 1 のメッセージがあります。

```
public void  get_priority_messages(TopicSession jms_session)
   {
     Topic            topic;
     TopicSubscriber  tsubs;
     ObjectMessage    obj_message;
     BolCustomer      customer;
     BolOrder         booked_order;

    try
    {
      /* get a handle to the OE_bookedorders_topic */
      topic = ((AQjmsSession)jms_session).getTopic("WS",
                                                   "WS_bookedorders_topic");

       /* Create local subscriber - for priority messages */
      tsubs = jms_session.createDurableSubscriber(topic, "PRIORITY",
                                     " JMSPriority = 1 ", false);

      obj_message = (ObjectMessage) tsubs.receive();

      booked_order = (BolOrder)obj_message.getObject();
```

```
      customer = booked_order.getCustomer();
      System.out.println("Priority Order:  for customer " +  customer.getName());

      jms_session.commit();
    }
    catch (JMSException ex)
    { System.out.println("Exception :" + ex); }
  }

  public void  get_normal_messages(TopicSession jms_session)
   {
     Topic            topic;
     TopicSubscriber  tsubs;
     ObjectMessage    obj_message;
     BolCustomer      customer;
     BolOrder         booked_order;

    try
    {
      /* get a handle to the OE_bookedorders_topic */
      topic = ((AQjmsSession)jms_session).getTopic("WS",
                                                   "WS_bookedorders_topic");

       /* Create local subscriber - for priority messages */
      tsubs = jms_session.createDurableSubscriber(topic, "PRIORITY",
                                     " JMSPriority > 1 ", false);

      obj_message = (ObjectMessage) tsubs.receive();

      booked_order = (BolOrder)obj_message.getObject();
      customer = booked_order.getCustomer();
      System.out.println("Normal Order:  for customer " +  customer.getName());

      jms_session.commit();
    }
    catch (JMSException ex)
    { System.out.println("Exception :" + ex); }
  }


public void remote_subscriber1(TopicSession jms_session)
   {
     Topic            topic;
     ObjectMessage    obj_message;
     AQjmsAgent       remote_sub;

    try
```

```
    {
      /* get a handle to the OE_bookedorders_topic */
      topic = ((AQjmsSession)jms_session).getTopic("OE",
                                                   "OE_bookedorders_topic");
      /* create the remote subscriber, name "Priority"  and address
       * the topic WS_booked_orders_topic at db2
       */
      remote_sub = new AQjmsAgent("Priority", "WS.WS_bookedorders_topic@dblink_oe_
ws");

      /* subscribe for western region orders */
      ((AQjmsSession)jms_session).createRemoteSubscriber(topic, remote_sub, "Region
= 'Western' ");
    }
    catch (JMSException ex)
    { System.out.println("Exception :" + ex); }
    catch (java.sql.SQLException  ex1)
    {System.out.println("SQL Exception :" + ex1); }
  }


   Remote database:
   database db2 - Western Shipping database.
/* get messages for subscriber priority */
   public void  get_priority_messages1(TopicSession jms_session)
   {
     Topic            topic;
     TopicReceiver    trecs;
     ObjectMessage    obj_message;
     BolCustomer      customer;
     BolOrder         booked_order;

    try
    {
      /* get a handle to the OE_bookedorders_topic */
      topic = ((AQjmsSession)jms_session).getTopic("WS",
                                                   "WS_bookedorders_topic");

      /* create a local receiver "Priority" for the remote subscription
       * to WS_bookedorders_topic
       */
      trecs = ((AQjmsSession)jms_session).createTopicReceiver(topic, "Priority",
null);

      obj_message = (ObjectMessage) trecs.receive();

      booked_order = (BolOrder)obj_message.getObject();
```

```
      customer = booked_order.getCustomer();
      System.out.println("Priority Order:  for customer " +  customer.getName());

      jms_session.commit();
    }
    catch (JMSException ex)
    { System.out.println("Exception :" + ex); }
  }
```

伝播スケジュール

伝播は、メッセージがターゲットの接続先データベースに伝播されるすべてのトピックについて、schedule_propagation メソッドを使用してスケジューリングされる必要があります。

スケジュールは時間の枠を示し、メッセージはその枠内でソース・トピックから伝播されます。この時間枠は、ネットワーク通信量、ソース・データベースの負荷、接続先データベースの負荷などの多くの要因に左右されます。したがって、スケジュールは特定のソースおよび宛先に合せて作成する必要があります。スケジュールが作成されると、ジョブは自動的にジョブ・キューに発行され、伝播が処理されます。

伝播スケジュールのための運用管理コールによって、スケジュール管理を柔軟に行うことができます（9-72 ページの「キューの伝播のスケジューリング」を参照）。あるスケジュールの存続時間（duration）または伝播枠パラメータによって、伝播が開始される時間枠が指定されます。存続時間が指定されない場合、時間枠は無制限の単一枠になります。枠を定期的に繰り返す必要がある場合、連続する枠の間の周期的間隔を定義する next_time パラメータを使用して有限の存続時間を指定します。

スケジュールの待機時間（latency）パラメータが関係するのは、キューに伝播されるべきメッセージが 1 つもないときのみです。このパラメータは、キューを再チェックする時間間隔を指定します。待機時間パラメータが適用された場合、job_queue_processes 用の job_queue_interval パラメータは、待機時間パラメータより小さい必要がある点に注意してください。あるキューに定義された伝播スケジュールは、そのキューの有効期間中いつでも変更または削除できます。さらに、（スケジュールを削除するかわりに）一時的に使用不可能にするコール、および使用不可能のスケジュールを使用可能にするコールがあります。メッセージがスケジュール内で伝播されているとき、そのスケジュールはアクティブです。すべての運用管理コールは、スケジュールがアクティブかどうかに関係なく実行されます。アクティブなスケジュールでは、コールが実行されるまでに数秒かかります。

伝播が開始されるには、ジョブ・キュー・プロセスを起動する必要があります。少なくとも 2 つのジョブ・キュー・プロセスを起動する必要があります。接続先データベースへのデータベース・リンクも、有効である必要があります。伝播のソースおよび宛先トピックは、同じメッセージ型である必要があります。リモート・トピックは、エンキューできる必要があります。データベース・リンクのユーザーも、リモート・トピックに対するエンキュー権限を持っている必要があります。

サンプル・コード

```
 public void  schedule_propagation(TopicSession jms_session)
  {
     Topic            topic;

    try
    {
      /* get a handle to the OE_bookedorders_topic */
      topic = ((AQjmsSession)jms_session).getTopic("WS",
                                                   "WS_bookedorders_topic");

      /* Schedule propagation immediately with duration of 5 minutes and latency 20
sec */
      ((AQjmsDestination)topic).schedulePropagation(jms_session, "dba", null,
                                     new Double(5*60), null, new Double(20));
    }catch (JMSException ex)
    {System.out.println("Exception: " + ex);}
  }

  Propagation schedule parameters can also be altered.


  /* alter duration to 10 minutes and latency to zero */
  public void  alter_propagation(TopicSession jms_session)
  {
     Topic            topic;
    try
    {
      /* get a handle to the OE_bookedorders_topic */
      topic = ((AQjmsSession)jms_session).getTopic("WS",
                                                   "WS_bookedorders_topic");

      /* Schedule propagation immediately with duration of 5 minutes and latency 20
sec */
    ((AQjmsDestination)topic).alterPropagationSchedule(jms_session, "dba",
                       new Double(10*60), null, new Double(0));
    }catch (JMSException ex)
    {System.out.println("Exception: " + ex);}
  }
```

拡張伝播スケジュール機能

スケジュールに関する詳細情報は、伝播のために定義されたカタログ・ビューから取得できます。アクティブ・スケジュールに関する情報、たとえば、そのスケジュールを処理しているバックグラウンド・プロセス名、伝播を処理しているセッションの SID（セッションのシリアル番号）、スケジュールを処理する Oracle インスタンス（Real Application Clusters が使用されている場合）などの情報は、そのカタログ・ビューから取得できます。同じカタログ・ビューによって、先行して正常に実行されたスケジュール（最後に正常に伝播されたメッセージ）および次に実行されるスケジュールに関する情報も得られます。

各スケジュールには、次の伝播の詳細情報が保持されます。

- スケジュールの中で伝播されたメッセージの合計数
- スケジュールの中で伝播されたバイトの合計数
- 伝播枠の中で伝播されたメッセージの最大数
- 伝播枠の中で伝播されたバイトの最大値
- 伝播枠の中で伝播されたメッセージの平均数
- 伝播されたメッセージの平均サイズ
- 伝播されたメッセージの平均時間

このような統計は、キュー管理者が最も効率的なスケジュール調整のために利用できるように設計されています。

伝播機能には、障害対処およびエラー・レポートが組み込まれています。たとえば、指定されたデータベース・リンクが無効な場合、リモート・データベースが使用できない場合、またはリモート・トピック / キューにエンキューできない場合、適切なエラー・メッセージがレポートされます。伝播は指数バックオフ・スキームを使用して、障害が発生したスケジュールからの伝播を再試行します。あるスケジュールが続けて障害を発生したときは、最初の再試行は 30 秒後、次の再試行は 60 秒後、3 回目の再試行は 120 秒後、というように続きます。再試行時間が現行の伝播枠の期限切れ時刻を超える場合は、次の再試行は、次の伝播枠の開始時刻に行われます。

最大 16 回の再試行が行われた後、そのスケジュールは自動的に使用不可能になります。障害のためにスケジュールが自動的に使用不可能になると、関連情報がアラート・ログに書き込まれます。どの場合でも、スケジュールに障害が発生しているか、発生している場合は連続的な障害がいくつあるかということが、障害の原因および直前の障害の発生時刻を示すエラー・メッセージによって確認できます。この情報を調べることで、管理者は障害を回復し、スケジュールを使用可能にできます。

再試行の間に伝播が成功したときは、障害の数は 0（ゼロ）にリセットされます。伝播機能には Oracle Real Application Clusters サポートが組み込まれていますが、ユーザーおよび管理者には透過的です。伝播を処理するジョブは、ソース・トピックが常駐しているキュー表の所有者と同じインスタンスに送られます。あるインスタンスに障害があってトピックを保存しているキュー表が他のインスタンスに移される場合は、伝播ジョブも必ず自動的に新しいインスタンスに移行されます。これによって、インスタンス間の ping 操作は最小限に抑

えられ、パフォーマンスが向上します。伝播は、同時スケジュールをいくつでも処理できるように設計されています。

job_queue_processes の数は、最大 1000 に制限され、その中のいくつかは伝播以外の関連ジョブを処理するために使用される場合があることに注意してください。このように、伝播にはマルチタスキングおよびロード・バランスのサポートが組み込まれています。伝播アルゴリズムは、複数スケジュールが単一スナップショット（ジョブ・キュー）のプロセスによって処理できるように設計されています。ジョブ・キュー・プロセスに対する伝播の負荷は、異なるソース・トピックからのメッセージ到着割合に基づいて偏りが発生する場合があります。あるプロセスが数個のアクティブ・スケジュールによって過負荷になっている一方で、別のプロセスは受動的なスケジュールが多いために余力があるというとき、伝播はプロセス間で負荷が均等になるようにスケジュールを自動的に再分配します。

サンプル・シナリオ

BooksOnLine の例では、OE_bookedorders_topic は、そこに含まれるメッセージが別の出荷サイトに伝播されるためビジーになります。エラー・チェックおよびスケジュール監視のための拡張伝播スケジュールをサポートしているコールを、次のサンプル・コードで示します。

サンプル・コード

```
CONNECT OE/OE;
/*  get averages
select avg_time, avg_number, avg_size from user_queue_schedules;

/*  get totals
select total_time, total_number, total_bytes from user_queue_schedules;

/*  get maximums for a window
select max_number, max_bytes from user_queue_schedules;

/*  get current status information of schedule
select process_name, session_id, instance, schedule_disabled
   from user_queue_schedules;

/*  get information about last and next execution
select last_run_date, last_run_time, next_run_date, next_run_time
   from user_queue_schedules;

/*  get last error information if any
select failures, last_error_msg, last_error_date, last_error_time
   from user_queue_schedules;
```

伝播中の例外処理

ネットワーク障害のようなシステム・エラーが発生した場合、AQ は指数関数的に実行間隔をあけてメッセージを伝播する試みを継続します。状況がアプリケーション・エラーを示しているとき、メッセージの伝播でエラーが発生した場合は、AQ はそのメッセージに UNDELIVERABLE というマークを付けます。

このようなエラーは、リモート・キュー / トピックが存在しないときやソース・キュー / トピックとリモート・キュー / トピックの型が一致しない場合に発生します。このような状況では、ユーザーは DBA_SCHEDULES ビューを問い合せて、特定の宛先への伝播中に発生した最新のエラーを調べます。$ORACLE_HOME/rdbms/log ディレクトリのトレース・ファイルには、そのエラーに関する追加情報があります。

JMS AQ のメッセージ変換

この項の内容は次のとおりです。

- メッセージ変換の定義
- 変換による宛先へのメッセージの送信
- 変換による宛先からのメッセージの受信
- トピック・サブスクライバ作成時の変換の指定
- リモート・サブスクライバ作成時の変換の指定

メッセージ変換の定義

あるフォーマットのメッセージを別のフォーマットのメッセージにマップするために変換を定義できます。変換は、同一の情報を異なるフォーマットで表示するアプリケーション同士を統合する場合に有効です。変換は SQL 式および PL/SQL ファンクションです。

変換は、DBMS_TRANSFORM.create_transformation プロシージャを使用して行います。変換は、次の操作を行う場合に指定できます。

- キューまたはトピックへのメッセージの送信
- キューまたはトピックからのメッセージの受信
- トピック・サブスクライバの作成
- リモート・サブスクライバの作成 これによって、異なるフォーマットのトピック間でメッセージを伝播できます。

メッセージ変換機能は、標準 JMS インタフェースに対する AQ 拡張機能です。

サンプル・シナリオ

BooksOnLine の例で、注文入力アプリケーションおよび出荷処理アプリケーションについて考えます。これらの例では、ユーザー定義型ペイロードを持つトピックを使用します。

サンプル・コード

注文入力トピック OE.OE_bookedorders_topic が、ペイロード型 OE.OE_ORDER を持っているとします。

```
create or replace TYPE OE_order as OBJECT (
             orderno         NUMBER,
             status          VARCHAR2(30),
             ordertype       VARCHAR2(30),
             orderregion     VARCHAR2(30),
             customer        CUSTOMER_TYP,
             paymentmethod   VARCHAR2(30),
             creditcard#     VARCHAR2(30);
             items           ORDERITEMLIST_VARTYP,
             order_date      DATE,
             total           NUMBER);
```

西部向け出荷処理トピック WS_bookedorders_topic は、ペイロード型 WS.WS_ORDER を持っています。

```
create or replace TYPE WS_Order AS OBJECT (
       customer_name   VARCHAR2(100),
       address         VARCHAR2(1000),
       city            VARCHAR2(1000),
       state           VARCHAR2(1000),
       country         VARCHAR2(1000),
       zipcode         VARCHAR2(1000),
       orderno         NUMBER,
       status          VARCHAR2(30),
       ordertype       VARCHAR2(30),
       items           ORDERITEMLIST_VARTYP,
       order_date      VARCHAR2(10));
```

これらの型に対して、JPublisher ユーティリティを使用して（CustomDatum インタフェースを実装する）Java クラスを生成できます。

OE.OE_Order から WS.WS_ORDER へのマッピングを定義する変換を次のように定義します。

```
execute dbms_transform.create_transformation(
schema => 'OE',
        name => 'OE2WS',
        from_schema => 'OE,
        from_type => 'OE_order',
```

```
to_schema => 'WS',
to_type => 'WS_order',
transformation => 'OE_order(source.user_data.customer.name, \
    source.user_data.customer.street, \
    source.user_data.customer.city, \
    source.user_data.customer.state, \
    source.user_data.customer.country, \
    source.user_data.customer.zipcode, \
    source.user_data.customer.country, \
    source.user_data.orderno, \
    source.user_data.status, \
    source.user_data.ordertype, \
    source.user_date.items, \
    TO_CHAR(source.user_date.order_date, 'MM:DD:YYYY'))');
```

変換による宛先へのメッセージの送信

キュー / トピックにメッセージを送信 / パブリッシュするときに変換を適用できます。メッセージを変換してから、キュー / トピックに送信します。

変換は、AQjmsQueueSender および AQjmsTopicPublisher の setTransformation インタフェースを使用して指定できます。

サンプル・コード

注文入力アプリケーションによって処理された注文情報を、WS_bookedorders_topic にパブリッシュする必要があると想定します。

正しいフォーマットでトピックにメッセージを挿入するには、前の項で定義した変換 OE2WS を使用します。

```
public void ship_booked_orders(TopicSession    jms_session,
                               AQjmsADTMessage adt_message)
{
       TopicPublisher  publisher;
       Topic           topic;

       try
       {
         /* get a handle to the WS_bookedorders_topic */
         topic = ((AQjmsSession)jms_session).getTopic("WS",
                                                      "WS_bookedorders_topic");
         publisher = jms_session.createPublisher(topic);

         /* set the transformation in the publisher */
        ((AQjmsTopicPublisher)publisher).setTransformation("OE2WS");

         publisher.publish(topic, adt_message);
```

```
        }
        catch (JMSException ex)
        {
           System.out.println("Exception :" ex);
        }
}
```

変換による宛先からのメッセージの受信

キューまたはトピックからメッセージを受信するときに変換を適用できます。メッセージを変換してから、JMS アプリケーションに戻します。

変換は、AqjmsQueueReceiver、AQjmsTopicSubscriber および AQjmsTopicReceiver の setTransformation() インタフェースを使用して指定できます。

サンプル・コード

西部向け出荷処理アプリケーションが OE_bookedorders_topic からメッセージを取り出すと想定します。変換 OE2WS を指定して、WS_order ユーザー定義型としてメッセージを取り出します。

Java クラス WSOrder が JPublisher によって生成され、Oracle オブジェクト型 WS.WS_order にマップされるとします。

```
public AQjmsAdtMessage retrieve_booked_orders(TopicSession jms_session)
       AQjmsTopicReceiver  receiver;
       Topic               topic;
       Message             msg = null;

       try
       {
         /* get a handle to the OE_bookedorders_topic */
         topic = ((AQjmsSession)jms_session).getTopic("OE",
                   "OE_bookedorders_topic");

         /* Create a receiver for WShip */
         receiver = ((AQjmsSession)jms_session).createTopicReceiver(topic,
                                      "WShip, null, WSOrder.getFactory());

         /* set the transformation in the publisher */
         receiver.setTransformation("OE2WS");

         msg = receiver.receive(10);
       }
       catch (JMSException ex)
       {
```

```
            System.out.println("Exception :" ex);
        }

        return (AQjmsAdtMessage)msg;
}
```

トピック・サブスクライバ作成時の変換の指定

CreateDurableSubscriber コールを使用してトピック・サブスクライバを作成する場合も、変換を指定することができます。取り出されたメッセージを変換してから、サブスクライバに戻します。指定されたサブスクライバがすでに CreateDurableSubscriber に存在する場合、その変換は指定した変換に設定されます。

サンプル・コード

西部向け出荷処理アプリケーションは、変換 OE2WS とともに OE_bookedorders_topic にサブスクライブします。この変換はメッセージに適用され、Oracle オブジェクト型 WS.WS_order のメッセージが戻されます。

Java クラス WSOrder が JPublisher によって生成され、Oracle オブジェクト型 WS.WS_order にマップされるとします。

```
public AQjmsAdtMessage retrieve_booked_orders(TopicSession jms_session)
{
        TopicSubscriber     subscriber;
        Topic               topic;
        AQjmsAdtMessage     msg = null;

        try
        {
          /* get a handle to the OE_bookedorders_topic */
          topic = ((AQjmsSession)jms_session).getTopic("OE",
                                                "OE_bookedorders_topic");

          /* create a subscriber with the transformation OE2WS */
          subs = ((AQjmsSession)jms_session).createDurableSubscriber(topic,
                  'WShip', null, false, WSOrder.getFactory(), "OE2WS");

          msg = subscriber.receive(10);
        }
        catch (JMSException ex)
        {
            System.out.println("Exception :" ex);
        }

        return (AQjmsAdtMessage)msg;
}
```

リモート・サブスクライバ作成時の変換の指定

AQ によって、リモート・サブスクライバ（他のデータベースにあるサブスクライバ）がトピックからサブスクライブできます。

createRemoteSubscriber を使用してリモート・サブスクライバを作成する場合、変換を指定できます。これによって、異なるフォーマットのトピック間でメッセージを伝播できます。トピックに対してパブリッシュされたメッセージがリモート・サブスクライバの基準を満たしている場合、AQ はリモート・サブスクライバに指定されているリモート・データベースにあるキュー / トピックにメッセージを自動的に伝播します。変換が指定される場合も、AQ はその変換をメッセージに適用してリモート・データベース上のキュー / トピックに伝播します。

サンプル・コード

メッセージが自動的に WS.WS_bookedorders_topic に伝播されるようにリモート・サブスクライバを OE.OE_bookedorders_topic に作成します。変換 OE2WS は、WS_bookedorders_topic に到達したメッセージが正しいフォーマットとなるように、リモート・サブスクライバを作成するときに指定します。

Java クラス WSOrder が JPublisher によって生成され、Oracle オブジェクト型 WS.WS_order にマップされるとします。

```
public void create_remote_sub(TopicSession jms_session)
{
     AQjmsAgent          subscriber;
     Topic               topic;

     try
     {
       /* get a handle to the OE_bookedorders_topic */
       topic = ((AQjmsSession)jms_session).getTopic("OE",
                                          "OE_bookedorders_topic");

       subscriber = new AQjmsAgent("WShip", "WS.WS_bookedorders_topic");

       ((AQjmsSession )jms_session).createRemoteSubscriber(topic,
                                 subscriber, null, WSOrder.getFactory(),"OE2WS");
     }
     catch (JMSException ex)
     {
       System.out.println("Exception :" ex);
     }
}
```

13

JMS 管理インタフェース : 基本操作

この章では、Oracle Advanced Queuing の管理インタフェースを利用方法に沿って説明します。それぞれの操作（キュー表の作成など）を、その操作名ごとに利用方法に沿って説明します。この章の先頭に、すべての利用方法をリストします（13-2 ページの「利用モデル : JMS 管理インタフェース - 基本操作」を参照）。

図 13-1 に、すべての利用方法を 1 つの図にまとめています。HTML 版のマニュアルをご使用の場合、この図の中の利用方法のタイトルをクリックして、関心のある利用方法に移動できます。

個々の利用方法は、次の形式で説明されています。

- **利用図 :** 利用方法を表す図
- **用途 :** この利用方法の用途
- **使用上の注意 :** 実装に有効なガイドライン
- **構文 :** このアクティビティの実行に使用する主な構文
- **例 :** 各プログラム環境での利用例

利用モデル : JMS 管理インタフェース - 基本操作

表 13-1 利用モデル : JMS 管理インタフェース - 基本操作

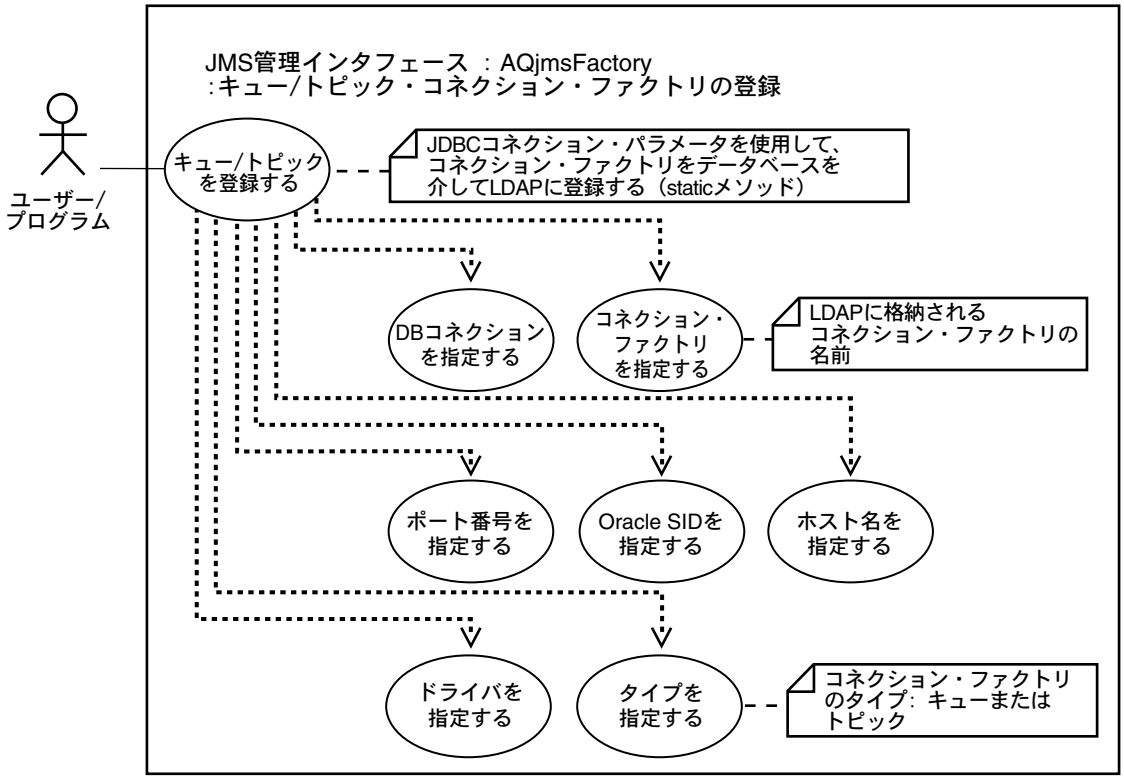
利用方法
キュー / トピック・コネクション・ファクトリのデータベースを介した登録 : JDBC コネクション・パラメータの使用（13-4 ページ）
キュー / トピック・コネクション・ファクトリのデータベースを介した登録 : JDBC URL の使用（13-6 ページ）
キュー / トピック・コネクション・ファクトリの LDAP を介した登録 : JDBC コネクション・パラメータの使用（13-8 ページ）
キュー / トピック・コネクション・ファクトリの LDAP を介した登録 : JDBC URL の使用（13-11 ページ）
LDAP 内のキュー / トピック・コネクション・ファクトリのデータベースを介した登録解除（13-14 ページ）
LDAP 内のキュー / トピック・コネクション・ファクトリの LDAP を介した登録解除（13-16 ページ）
キュー・コネクション・ファクトリの取得 : JDBC URL の使用（13-18 ページ）
キュー・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用（13-20 ページ）
トピック・コネクション・ファクトリの取得 : JDBC URL の使用（13-22 ページ）
トピック・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用（13-24 ページ）
LDAP 内のキュー / トピック・コネクション・ファクトリの取得（13-26 ページ）
LDAP 内のキュー / トピックの取得（13-28 ページ）
キュー表の作成（13-30 ページ）
キュー表の作成（キュー表のプロパティの指定）（13-32 ページ）
キュー表の取得（13-34 ページ）
宛先プロパティの指定（13-36 ページ）
キューの作成 : Point-to-Point（13-38 ページ）
トピックの作成 : パブリッシュ・サブスクライブ（13-40 ページ）
システム権限の付与（13-42 ページ）
システム権限の取消し（13-44 ページ）
トピック権限の付与 : パブリッシュ・サブスクライブ（13-46 ページ）
トピック権限の取消し : パブリッシュ・サブスクライブ（13-48 ページ）
キュー権限の付与 : Point-to-Point（13-50 ページ）

表 13-1 利用モデル : JMS 管理インタフェース - 基本操作（続き）

利用方法
キュー権限の取消し : Point-to-Point（13-52 ページ）
宛先の開始（13-54 ページ）
宛先の停止（13-56 ページ）
宛先の変更（13-58 ページ）
宛先の削除（13-60 ページ）
伝播のスケジューリング（13-62 ページ）
伝播スケジュールの使用可能化（13-64 ページ）
伝播スケジュールの変更（13-66 ページ）
伝播スケジュールの使用不可能化（13-68 ページ）
伝播スケジュールの解除（13-70 ページ）

キュー / トピック・コネクション・ファクトリのデータベースを介した登録 : JDBC コネクション・パラメータの使用

図 13-1 データベースを介した登録 : JDBC コネクション・パラメータの使用



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-6 ページの「キュー / トピック・コネクション・ファクトリのデータベースを介した登録 : JDBC URL の使用」も参照してください。

用途

JDBC コネクション・パラメータを使用して、キュー / トピック・コネクション・ファクトリをデータベースを介して LDAP に登録します。

使用上の注意

registerConnectionFactory は、static メソッドです。正常にコネクション・ファクトリを登録するには、registerConnectionFactory に渡される DB コネクションには、AQ_ADMINISTRATOR_ROLE を付与する必要があります。登録後、JNDI を使用してコネクション・ファクトリを検索します。

構文

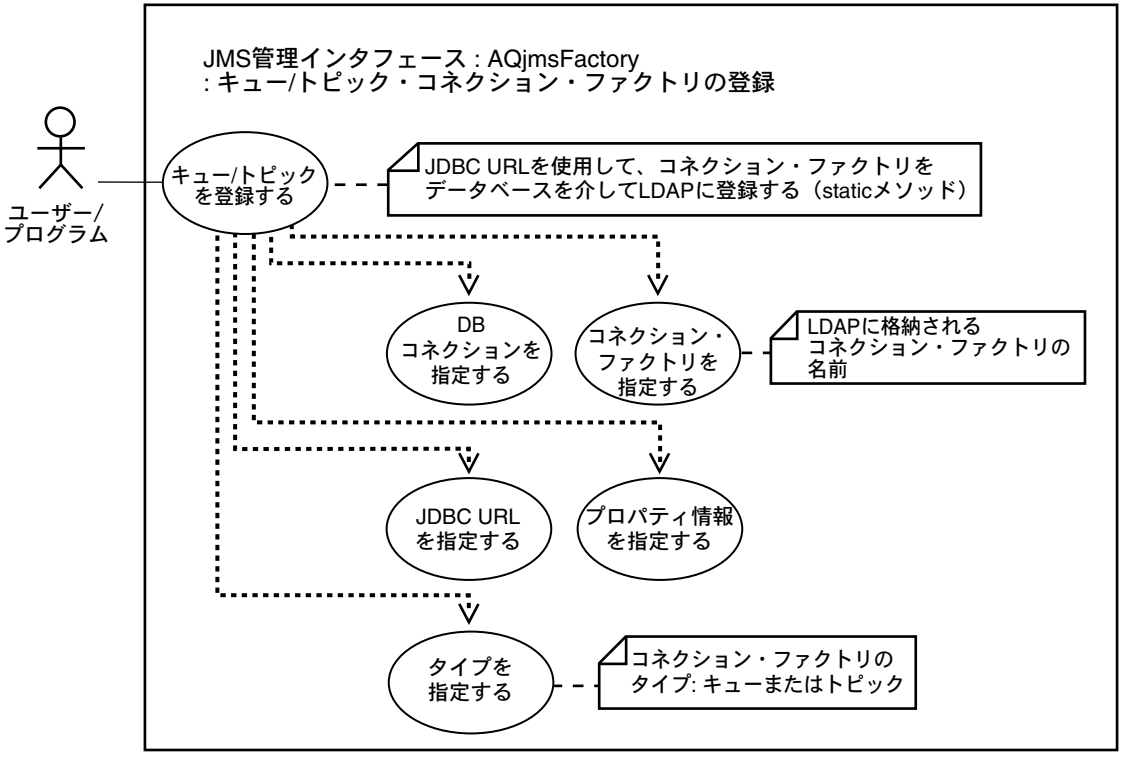
Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の registerConnectionFactory を参照してください。

例

```
String               url;
java.sql.connection  db_conn;

url = "jdbc:oracle:thin:@sun-123:1521:db1";
db_conn = DriverManager.getConnection(url, "scott", "tiger");
AQjmsFactory.registerConnectionFactory(db_conn, "queue_conn1", "sun-123",
  "db1", 1521, "thin", "queue");
```

キュー / トピック・コネクション・ファクトリのデータベースを介した登録 : JDBC URL の使用

図 13-2 データベースを介した登録 : JDBC URL の使用



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-4 ページの「キュー / トピック・コネクション・ファクトリのデータベースを介した登録 : JDBC コネクション・パラメータの使用」も参照してください。

用途

JDBC URL を使用して、キュー / トピック・コネクション・ファクトリをデータベースを介して LDAP に登録します。

使用上の注意

registerConnectionFactory は、static メソッドです。正常にコネクション・ファクトリを登録するには、registerConnectionFactory に渡される DB コネクションには、AQ_ADMINISTRATOR_ROLE を付与する必要があります。登録後、JNDI を使用してコネクション・ファクトリを検索します。

構文

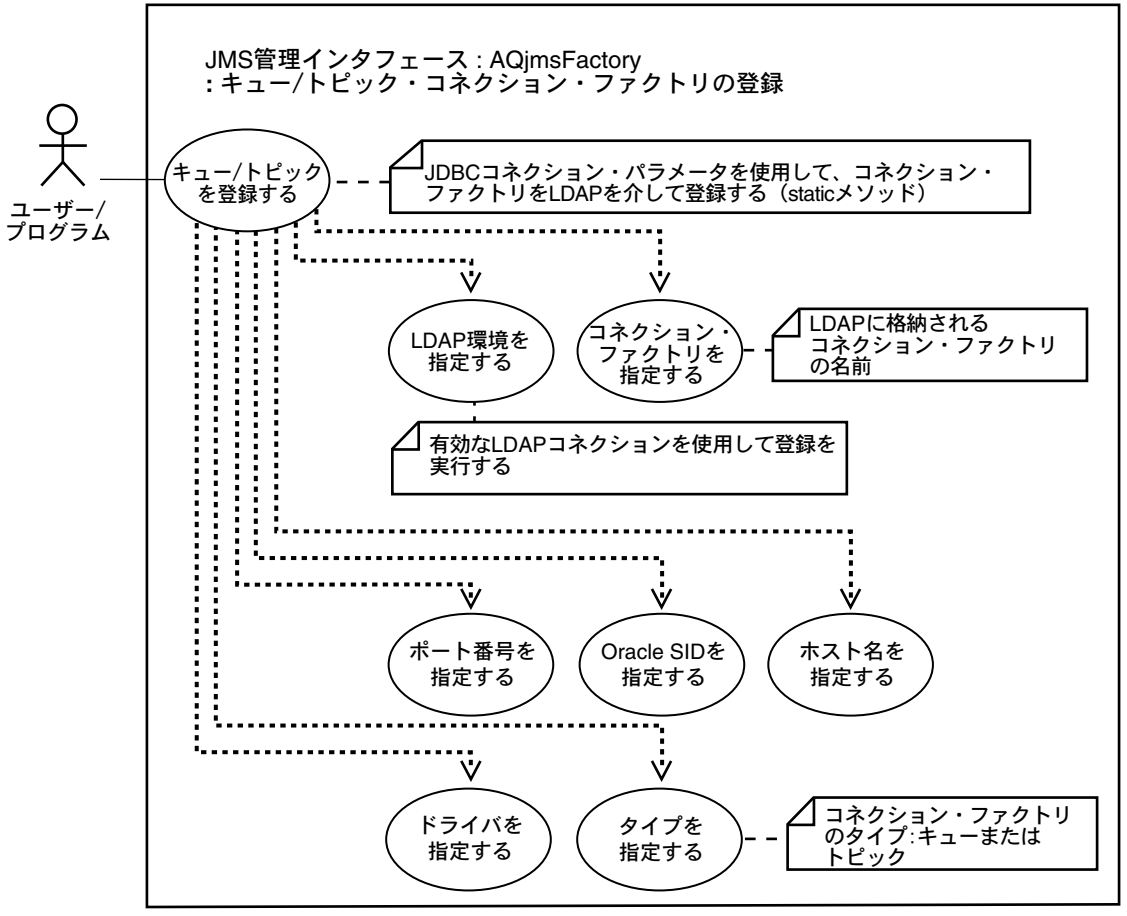
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の registerConnectionFactory を参照してください。

例

```
String                          url;
java.sql.connection             db_conn;

url = "jdbc:oracle:thin:@sun-123:1521:db1";
db_conn = DriverManager.getConnection(url, "scott", "tiger");
AQjmsFactory.registerConnectionFactory(db_conn, "topic_conn1", url,
   null, "topic");
```

キュー / トピック・コネクション・ファクトリの LDAP を介した登録 : JDBC コネクション・パラメータの使用

図 13-3 LDAP を介した登録 : JDBC コネクション・パラメータの使用



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-11 ページ「キュー / トピック・コネクション・ファクトリの LDAP を介した登録 : JDBC URL の使用」も参照してください。

用途

JDBC コネクション・パラメータを使用して、キュー / トピック・コネクション・ファクトリを LDAP に登録します。

使用上の注意

`registerConnectionFactory` は、static メソッドです。正常にコネクション・ファクトリを登録するには、`registerConnectionFactory` に渡されるハッシュ表に、LDAP サーバーと有効なコネクションを確立するための情報が含まれている必要があります。さらに、このコネクションには、LDAP サーバー内のコネクション・ファクトリのエントリに対する書込み権限が必要です（LDAP ユーザーがデータベースそのものであるか、または LDAP ユーザーに `global_aq_user_role` が付与されている必要があります）。登録後、JNDI を使用してコネクション・ファクトリを検索します。

構文

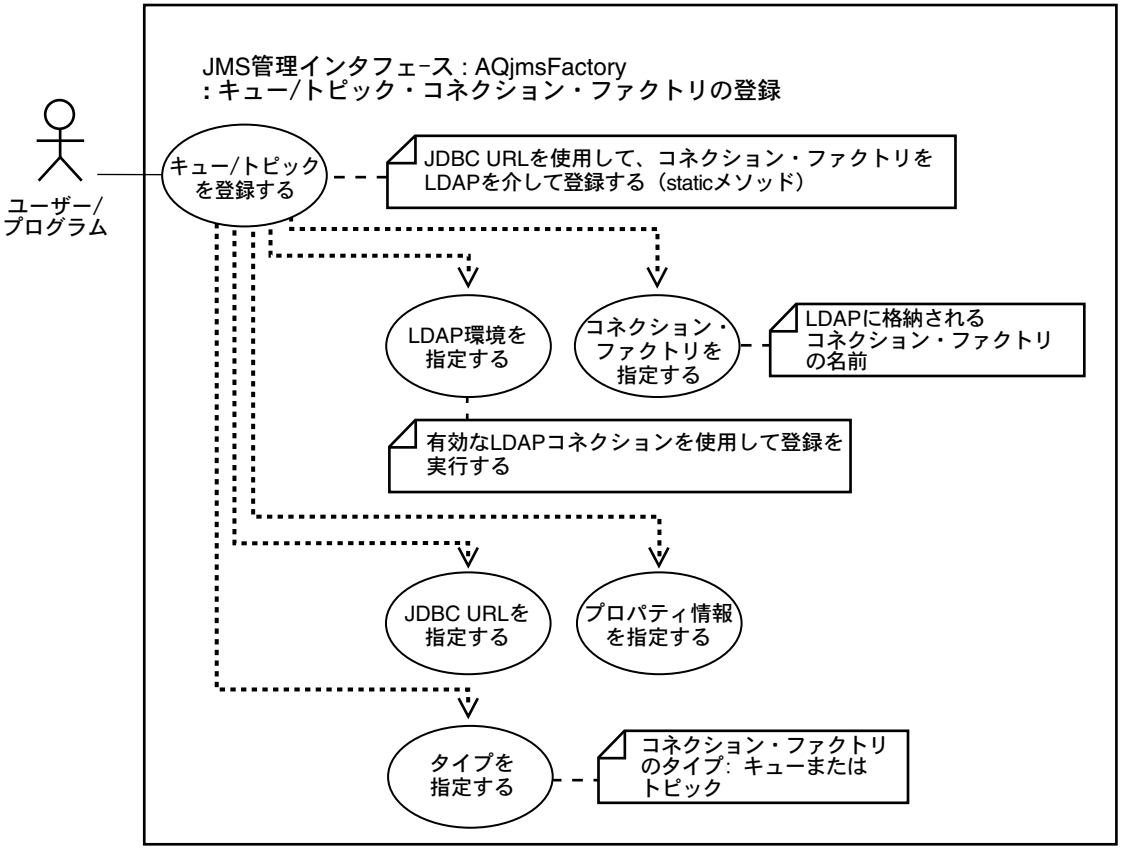
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の `registerConnectionFactory` を参照してください。

例

```
Hashtable            env = new Hashtable(5, 0.75f);
/* the following statements set in hashtable env:
   * service provider package
   * the URL of the ldap server
   * the distinguished name of the database server
   * the authentication method (simple)
   * the LDAP user name
   * the LDAP user password
*/
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://sun-456:389");
env.put("searchbase", "cn=db1,cn=Oraclecontext,cn=acme,cn=com");
env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, "cn=db1aqadmin,cn=acme,cn=com");
env.put(Context.SECURITY_CREDENTIALS, "welcome");

AQjmsFactory.registerConnectionFactory(env, "queue_conn1", "sun-123",
  "db1", 1521, "thin", "queue");
```

キュー / トピック・コネクション・ファクトリの LDAP を介した登録 : JDBC URL の使用

図 13-4 LDAP を介した登録 : JDBC URL の使用



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-8 ページ「キュー / トピック・コネクション・ファクトリの LDAP を介した登録 : JDBC コネクション・パラメータの使用」も参照してください。

用途

JDBC URL を使用して、キュー / トピック・コネクション・ファクトリを LDAP に登録します。

使用上の注意

`registerConnectionFactory` は、static メソッドです。正常にコネクション・ファクトリを登録するには、`registerConnectionFactory` に渡されるハッシュ表に、LDAP サーバーと有効なコネクションを確立するための情報が含まれている必要があります。さらに、このコネクションには、LDAP サーバー内のコネクション・ファクトリのエントリに対する書込み権限が必要です（LDAP ユーザーがデータベースそのものであるか、または LDAP ユーザーに `global_aq_user_role` が付与されている必要があります）。登録後、JNDI を使用してコネクション・ファクトリを検索します。

構文

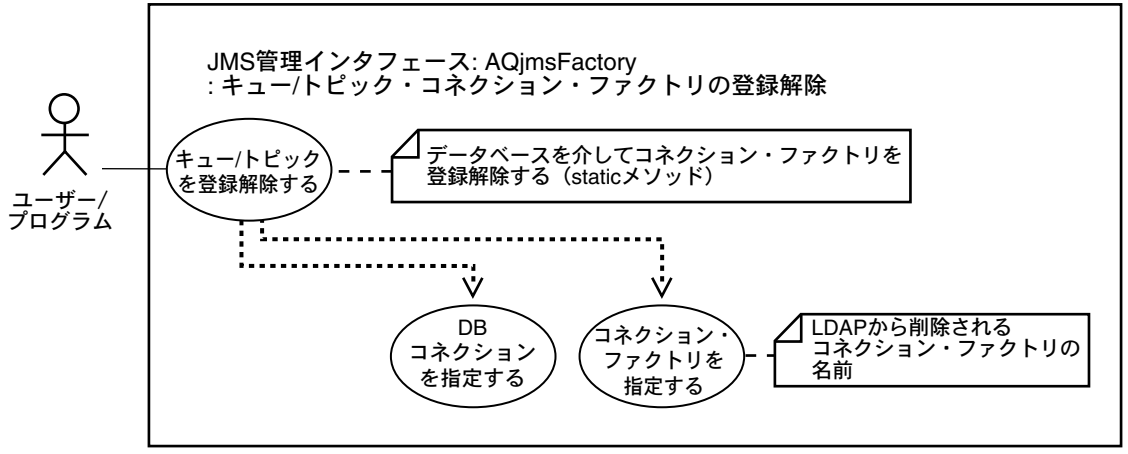
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の `registerConnectionFactory` を参照してください。

例

```
String              url;
Hashtable           env = new Hashtable(5, 0.75f);

/* the following statements set in hashtable env:
   * service provider package
   * the URL of the ldap server
   * the distinguished name of the database server
   * the authentication method (simple)
   * the LDAP user name
   * the LDAP user password
*/
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://sun-456:389");
env.put("searchbase", "cn=db1,cn=Oraclecontext,cn=acme,cn=com");
env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, "cn=db1aqadmin,cn=acme,cn=com");
env.put(Context.SECURITY_CREDENTIALS, "welcome");
url = "jdbc:oracle:thin:@sun-123:1521:db1";
AQjmsFactory.registerConnectionFactory(env, "topic_conn1", url, null, "topic");
```

LDAP 内のキュー / トピック・コネクション・ファクトリのデータベースを介した登録解除

図 13-5 LDAP 内のキュー / トピック・コネクション・ファクトリのデータベースを介した登録解除



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-16 ページの「LDAP 内のキュー / トピック・コネクション・ファクトリの LDAP を介した登録解除」も参照してください。

用途

LDAP 内のキュー / トピック・コネクション・ファクトリを登録解除します。

使用上の注意

unregisterConnectionFactory は、static メソッドです。正常にコネクション・ファクトリを登録解除するには、unregisterConnectionFactory に渡される DB コネクションに、AQ_ADMINISTRATOR_ROLE を付与する必要があります。

構文

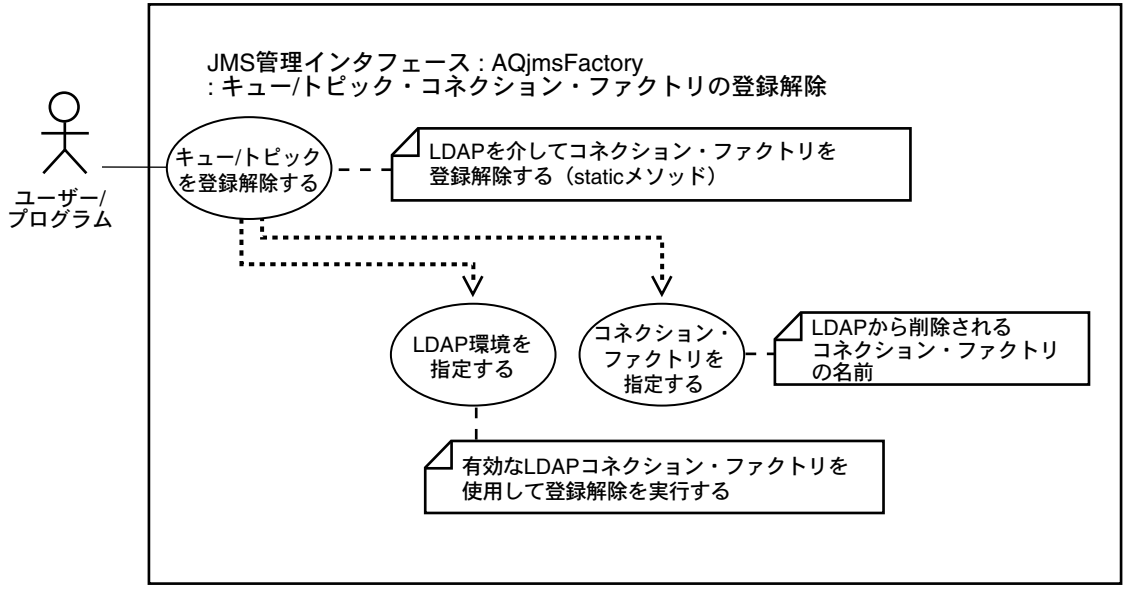
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の unregisterConnectionFactory を参照してください。

例

```
String               url;
java.sql.connection  db_conn;

url = "jdbc:oracle:thin:@sun-123:1521:db1";
db_conn = DriverManager.getConnection(url, "scott", "tiger");
AQjmsFactory.unregisterConnectionFactory(db_conn, "topic_conn1");
```

LDAP 内のキュー / トピック・コネクション・ファクトリの LDAP を介した登録解除

図 13-6 LDAP 内のキュー / トピック・コネクション・ファクトリの LDAP を介した登録解除



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-14 ページの「LDAP 内のキュー / トピック・コネクション・ファクトリのデータベースを介した登録解除」も参照してください。

用途

LDAP 内のキュー / トピック・コネクション・ファクトリを登録解除します。

使用上の注意

unregisterConnectionFactory は、static メソッドです。正常にコネクション・ファクトリを登録解除するには、unregisterConnectionFactory に渡されるハッシュ表に、LDAP サーバーと有効なコネクションを確立するための情報が含まれている必要があります。さらに、このコネクションには、LDAP サーバー内のコネクション・ファクトリのエントリに対する書込み権限が必要です（LDAP ユーザーがデータベースそのものであるか、または LDAP ユーザーに global_aq_user_role が付与されている必要があります）。

構文

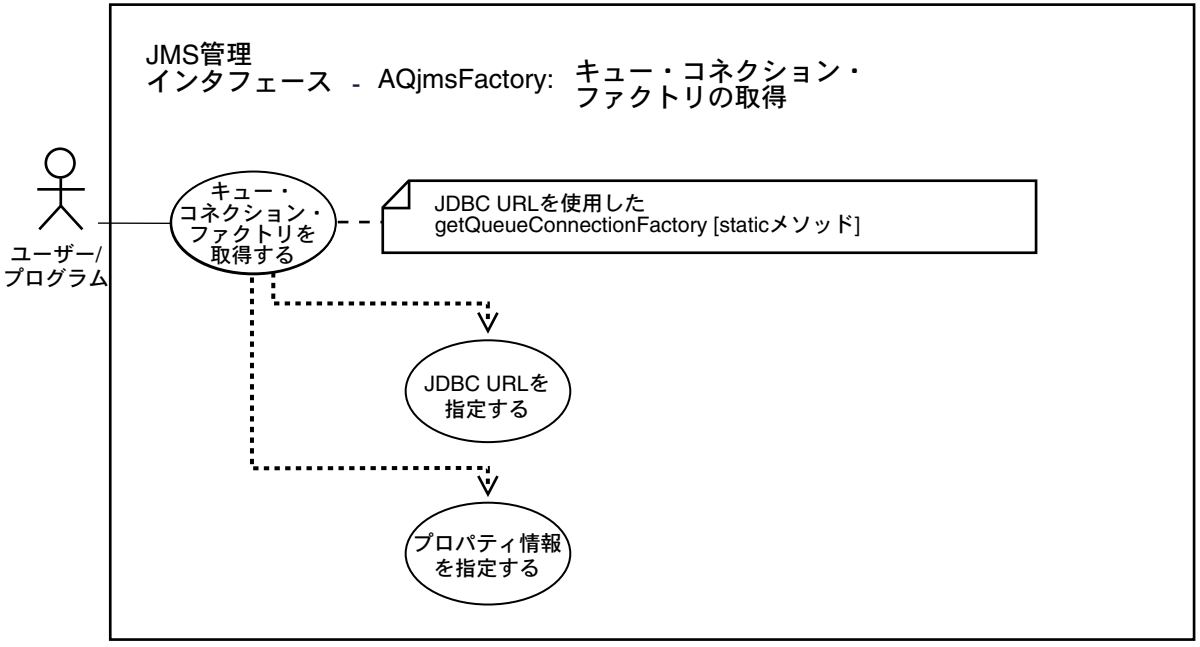
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の unregisterConnectionFactory を参照してください。

例

```
String              url;
Hashtable           env = new Hashtable(5, 0.75f);

/* the following statements set in hashtable env:
   * service provider package
   * the URL of the ldap server
   * the distinguished name of the database server
   * the authentication method (simple)
   * the LDAP user name
   * the LDAP user password
*/

env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://sun-456:389");
env.put("searchbase", "cn=db1,cn=Oraclecontext,cn=acme,cn=com");
env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, "cn=db1aqadmin,cn=acme,cn=com");
env.put(Context.SECURITY_CREDENTIALS, "welcome");
url = "jdbc:oracle:thin:@sun-123:1521:db1";
AQjmsFactory.unregisterConnectionFactory(env, "queue_conn1");
```

キュー・コネクション・ファクトリの取得 : JDBC URL の使用

図 13-7 キュー・コネクション・ファクトリの取得 : JDBC URL の使用



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-20 ページの「キュー・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用」も参照してください。

用途

JDBC URL を使用して、キュー・コネクション・ファクトリを取得します。

使用上の注意

`getQueueConnectionFactory` は static メソッドです。

構文

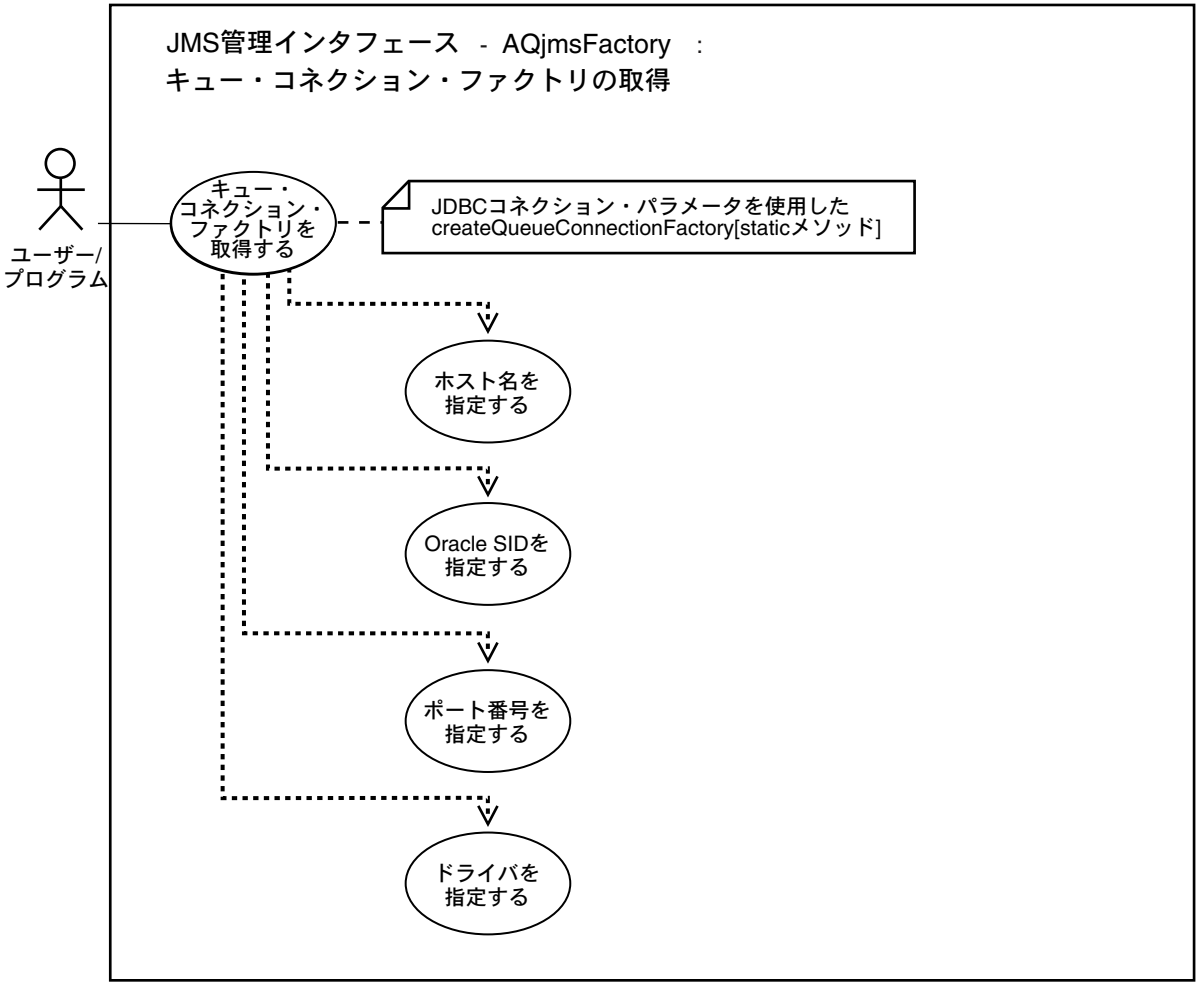
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の `getQueueConnectionFactory` を参照してください。

例

```
String      url          = "jdbc:oracle:oci8:internal/oracle"
Properties  info         = new Properties();
QueueConnectionFactory   qc_fact;

info.put("internal_logon", "sysdba");
qc_fact = AQjmsFactory.getQueueConnectionFactory(url, info);
```

キュー・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用

図 13-8 キュー・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-18 ページの「キュー・コネクション・ファクトリの取得 : JDBC URL の使用」も参照してください。

用途

JDBC コネクション・パラメータを使用して、キュー・コネクション・ファクトリを取得します。

使用上の注意

`getQueueConnectionFactory` は static メソッドです。

構文

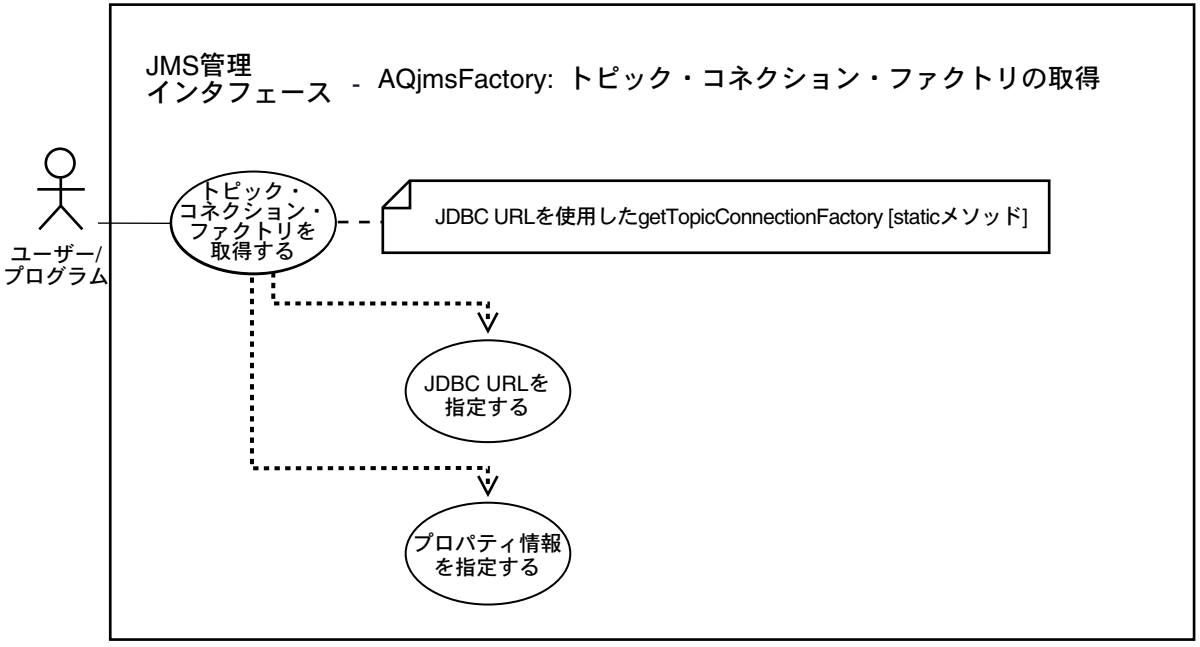
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の `getQueueConnectionFactory` を参照してください。

例

```
String      host         = "dlsun";
String      ora_sid      = "rdbms8i"
String      driver       = "thin";
int         port         = 5521;
QueueConnectionFactory   qc_fact;

qc_fact = AQjmsFactory.getQueueConnectionFactory(host, ora_sid, port, driver);
```

トピック・コネクション・ファクトリの取得 : JDBC URL の使用

図 13-9 トピック・コネクション・ファクトリの取得 : JDBC URL の使用



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-24 ページの「トピック・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用」も参照してください。

用途

JDBC URL を使用して、トピック・コネクション・ファクトリを取得します。

使用上の注意

`getTopicConnectionFactory` は static メソッドです。

構文

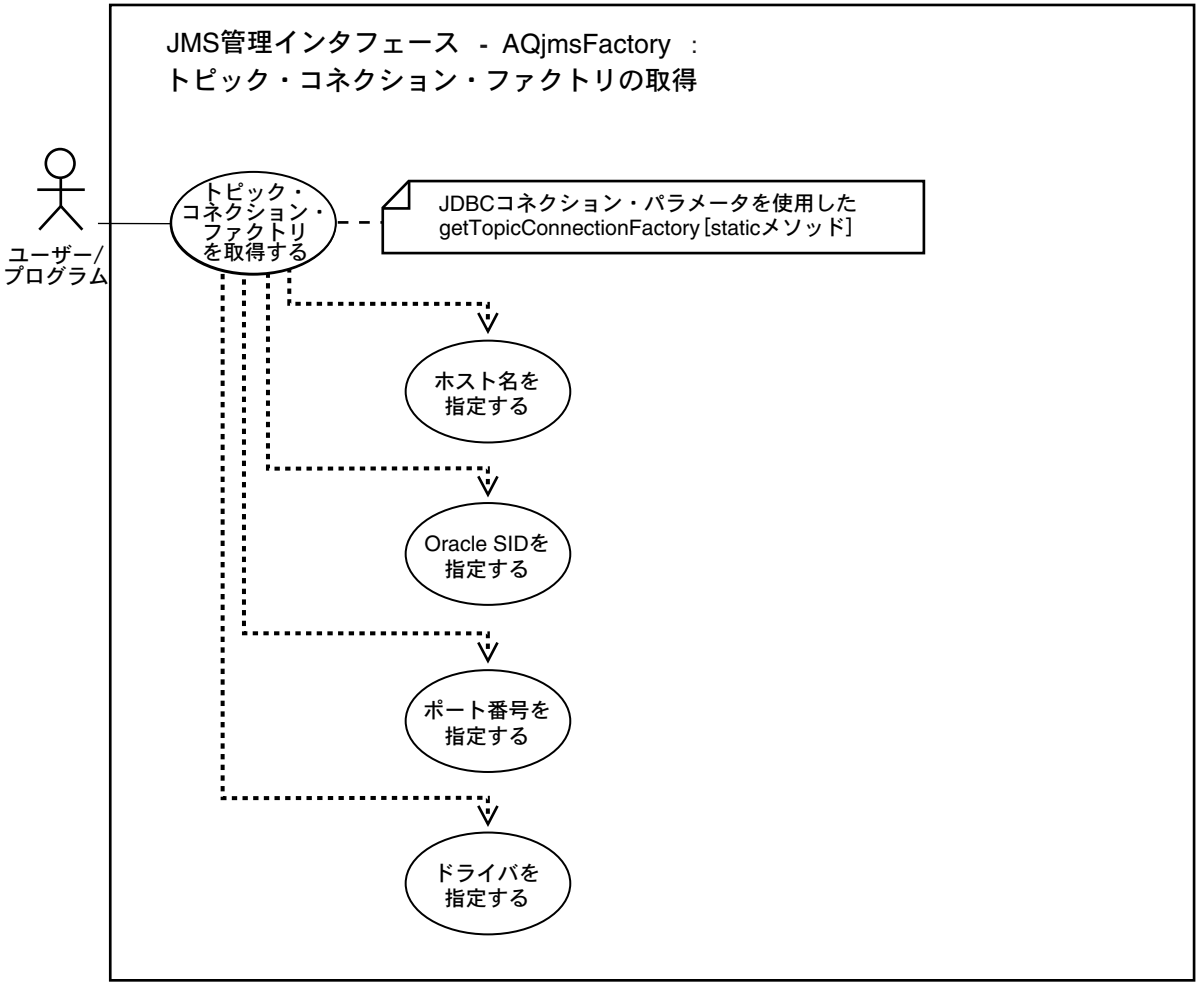
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の `getTopicConnectionFactory` を参照してください。

例

```
String      url          = "jdbc:oracle:oci8:internal/oracle"
Properties  info         = new Properties();
TopicConnectionFactory   tc_fact;

info.put("internal_logon", "sysdba");
tc_fact = AQjmsFactory.getTopicConnectionFactory(url, info);
```

トピック・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用

図 13-10 トピック・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- 13-22 ページの「トピック・コネクション・ファクトリの取得 : JDBC URL の使用」も参照してください。

使用上の注意

`getTopicConnectionFactory` は static メソッドです。

用途

JDBC コネクション・パラメータを使用して、トピック・コネクション・ファクトリを取得します。

構文

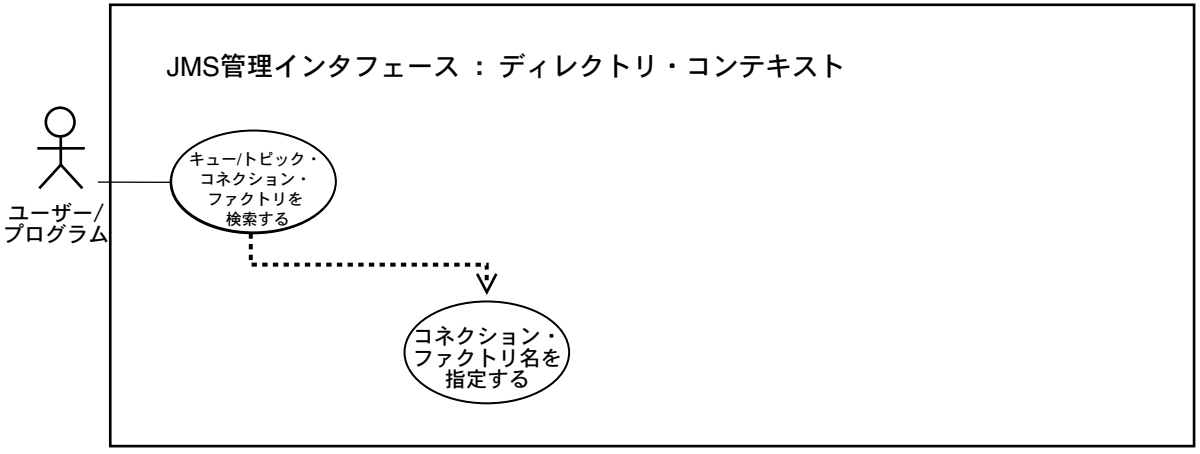
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsFactory」の `getTopicConnectionFactory` を参照してください。

例

```
String      host         = "dlsun";
String      ora_sid      = "rdbms8i"
String      driver       = "thin";
int         port         = 5521;
TopicConnectionFactory   tc_fact;

tc_fact = AQjmsFactory.getTopicConnectionFactory(host, ora_sid, port, driver);
```

LDAP 内のキュー / トピック・コネクション・ファクトリの取得

図 13-11 LDAP 内のキュー / トピック・コネクション・ファクトリの取得



参照 : JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。

用途

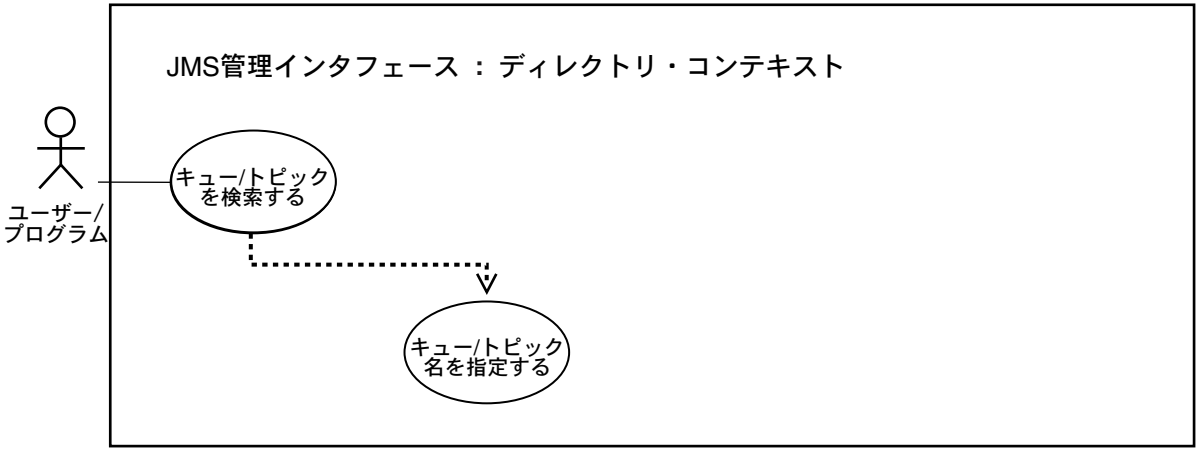
LDAP からキュー / トピック・コネクション・ファクトリを取得します。

例

```
Hashtable              env = new Hashtable(5, 0.75f);
DirContext             ctx;
queueConnectionFactory qc_fact;

/* the following statements set in hashtable env:
   * service provider package
   * the URL of the ldap server
   * the distinguished name of the database server
   * the authentication method (simple)
   * the LDAP user name
   * the LDAP user password
*/
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://sun-456:389");
env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, "cn=db1aquser1,cn=acme,cn=com");
env.put(Context.SECURITY_CREDENTIALS, "welcome");

ctx = new InitialDirContext(env);
ctx =
(DirContext)ctx.lookup("cn=OracleDBConnections,cn=db1,cn=Oraclecontext,cn=acme,cn=co
m");
qc_fact = (queueConnectionFactory)ctx.lookup("cn=queue_conn1");
```

LDAP 内のキュー / トピックの取得

図 13-12 LDAP 内のキュー / トピックの取得



参照： JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。

用途

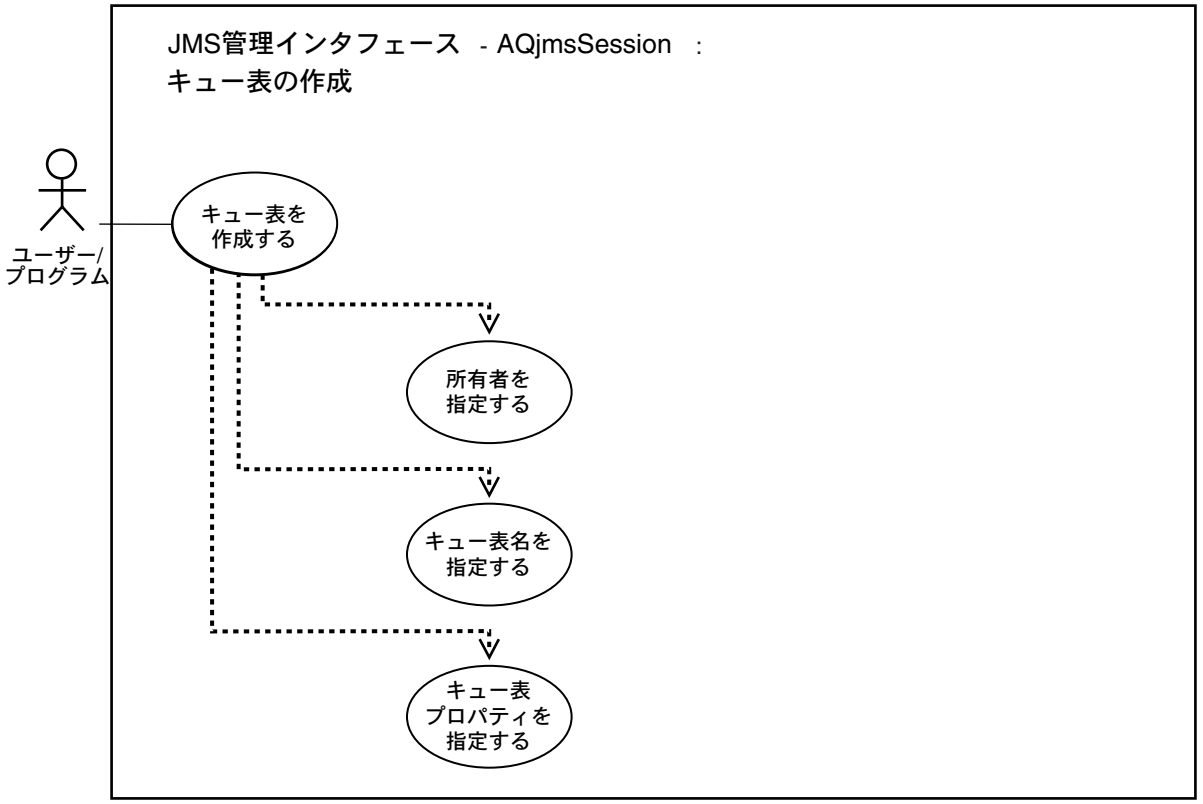
LDAP からキュー / トピックを取得します。

例

```
Hashtable              env = new Hashtable(5, 0.75f);
DirContext             ctx;
topic                  topic_1;

/* the following statements set in hashtable env:
   * service provider package
   * the URL of the ldap server
   * the distinguished name of the database server
   * the authentication method (simple)
   * the LDAP user name
   * the LDAP user password
*/
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://sun-456:389");
env.put(Context.SECURITY_AUTHENTICATION, "simple");
env.put(Context.SECURITY_PRINCIPAL, "cn=db1aquser1,cn=acme,cn=com");
env.put(Context.SECURITY_CREDENTIALS, "welcome");

ctx = new InitialDirContext(env);
ctx =
(DirContext)ctx.lookup("cn=OracleDBQueues,cn=db1,cn=Oraclecontext,cn=acme,cn=com");
topic_1 = (topic)ctx.lookup("cn=topic_1");
```

キュー表の作成

図 13-13 キュー表の作成



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 13-32 ページの「キュー表の作成（キュー表のプロパティの指定）」も参照してください。

用途

キュー表を作成します。

使用上の注意

AQ オブジェクト型では、CLOB、BLOB、BFILE オブジェクトは有効な属性です。ただし、Oracle8*i* 以上では、CLOB および BLOB のみ AQ 伝播を使用して伝播できます。

構文

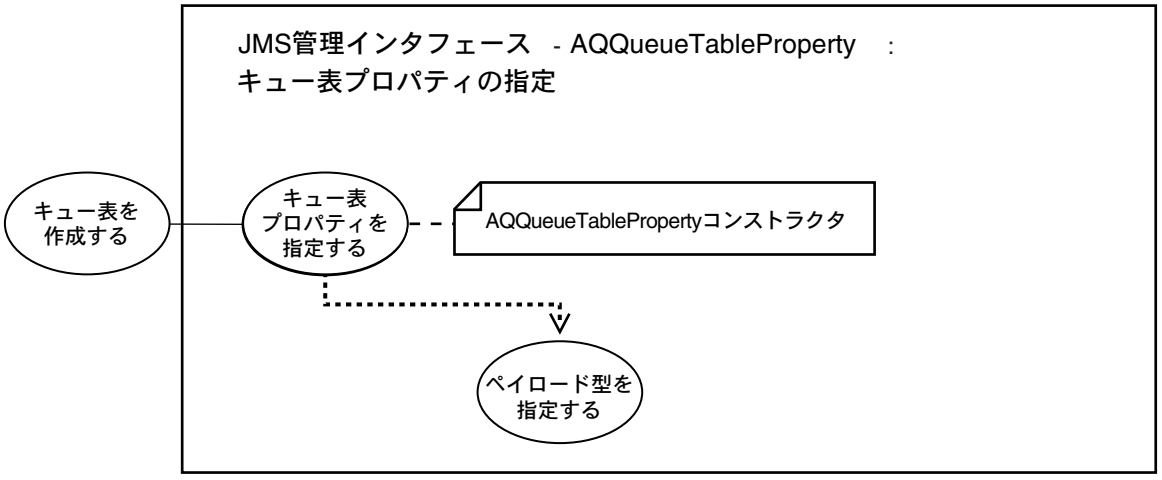
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の createQueueTable を参照してください。

例

```
QueueSession              q_sess    = null;
AQQueueTable              q_table   = null;
AQQueueTableProperty      qt_prop   = null;

qt_prop = new AQQueueTableProperty("SYS.AQ$_JMS_BYTES_MESSAGE");
  q_table = ((AQjmsSession)q_sess).createQueueTable("boluser",
  "bol_ship_queue_table", qt_prop);
```

キュー表の作成（キュー表のプロパティの指定）

図 13-14 キュー表の作成（キュー表のプロパティの指定）



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-60 ページの「クラス - oracle.AQ.AQQueueTableProperty」も参照してください。
- 13-30 ページの「キュー表の作成」も参照してください。

用途

キュー表のプロパティを指定します。

使用上の注意

ありません。

構文

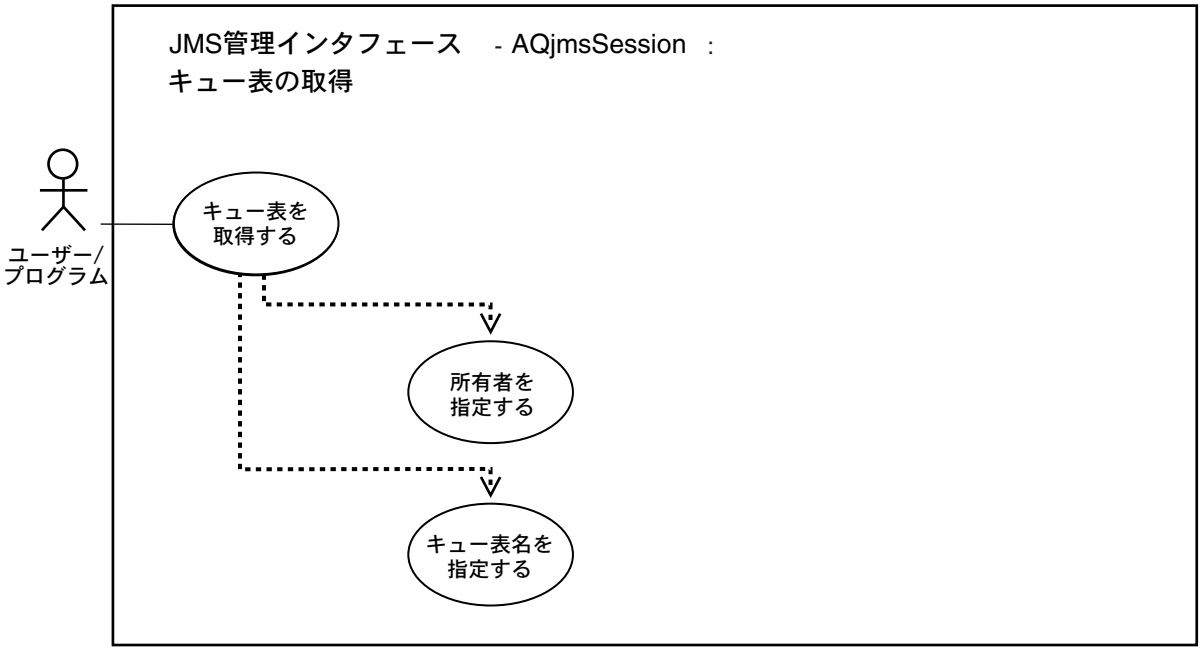
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.AQ」の「`AQQueueTableProperty`」を参照してください。

例

```
QueueSession              q_sess    = null;
AQQueueTable              q_table   = null;
AQQueueTableProperty      qt_prop   = null;

qt_prop = new AQQueueTableProperty("SYS.AQ$_JMS_BYTES_MESSAGE");
q_table = ((AQjmsSession)q_sess).createQueueTable("boluser",
   "bol_ship_queue_table", qt_prop);
```

キュー表の取得

図 13-15 キュー表の取得



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

キュー表を取得します。

使用上の注意

コネクションをオープンしたコール側がキュー表の所有者でない場合、コール側には、キュー表内のキュー / トピックに対する AQ エンキュー / デキュー権限が必要です。この権限がない場合、キュー表は取得できません。

構文

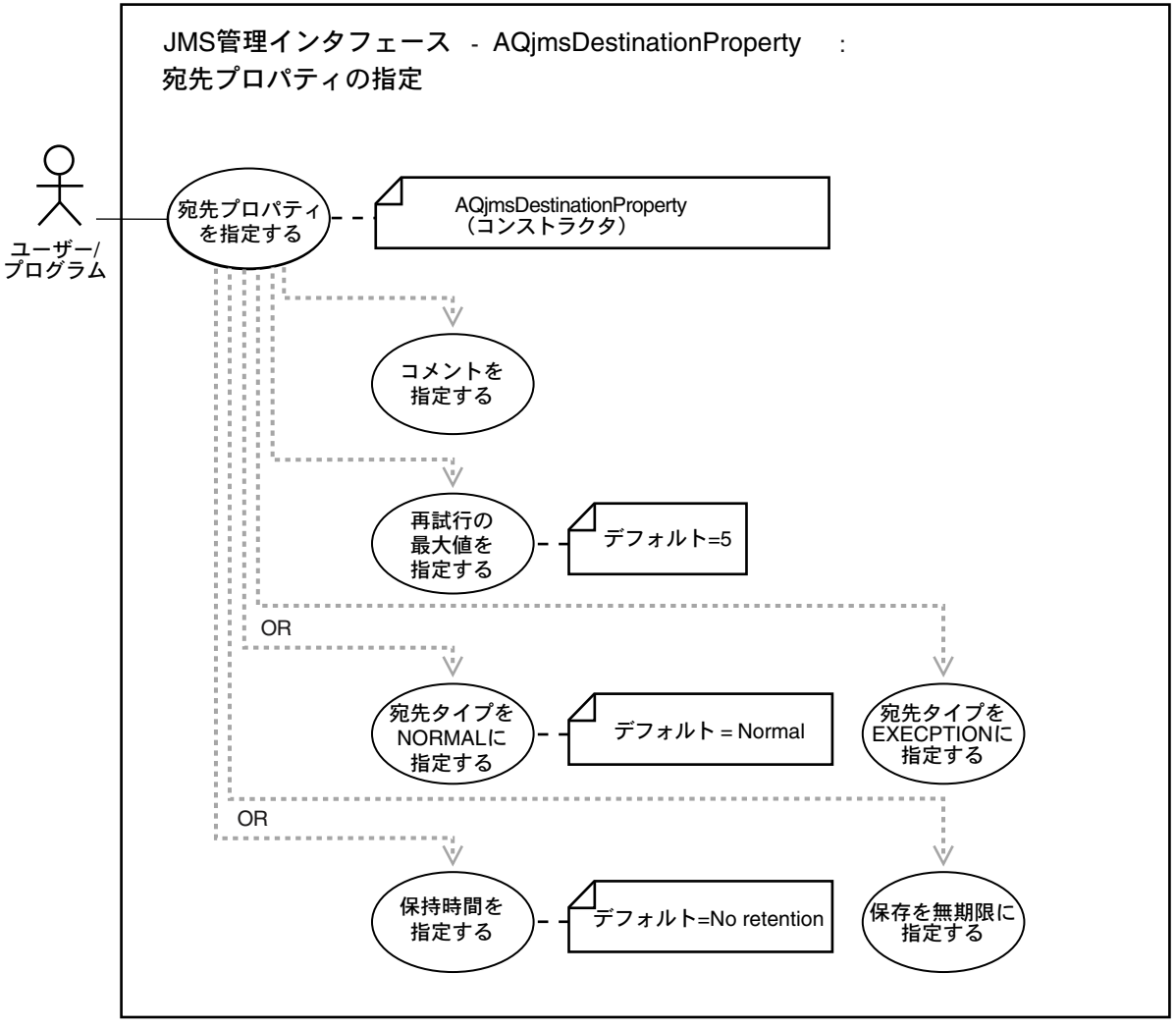
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `getQueueTable` を参照してください。

例

```
QueueSession              q_sess;
AQQueueTable              q_table;

 q_table = ((AQjmsSession)q_sess).getQueueTable("boluser","bol_ship_queue_table");
```

宛先プロパティの指定

図 13-16 宛先プロパティの指定



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-50 ページの「クラス - oracle.jms.AQjmsDestinationProperty」も参照してください。

用途

宛先プロパティを指定します。

使用上の注意

ありません。

構文

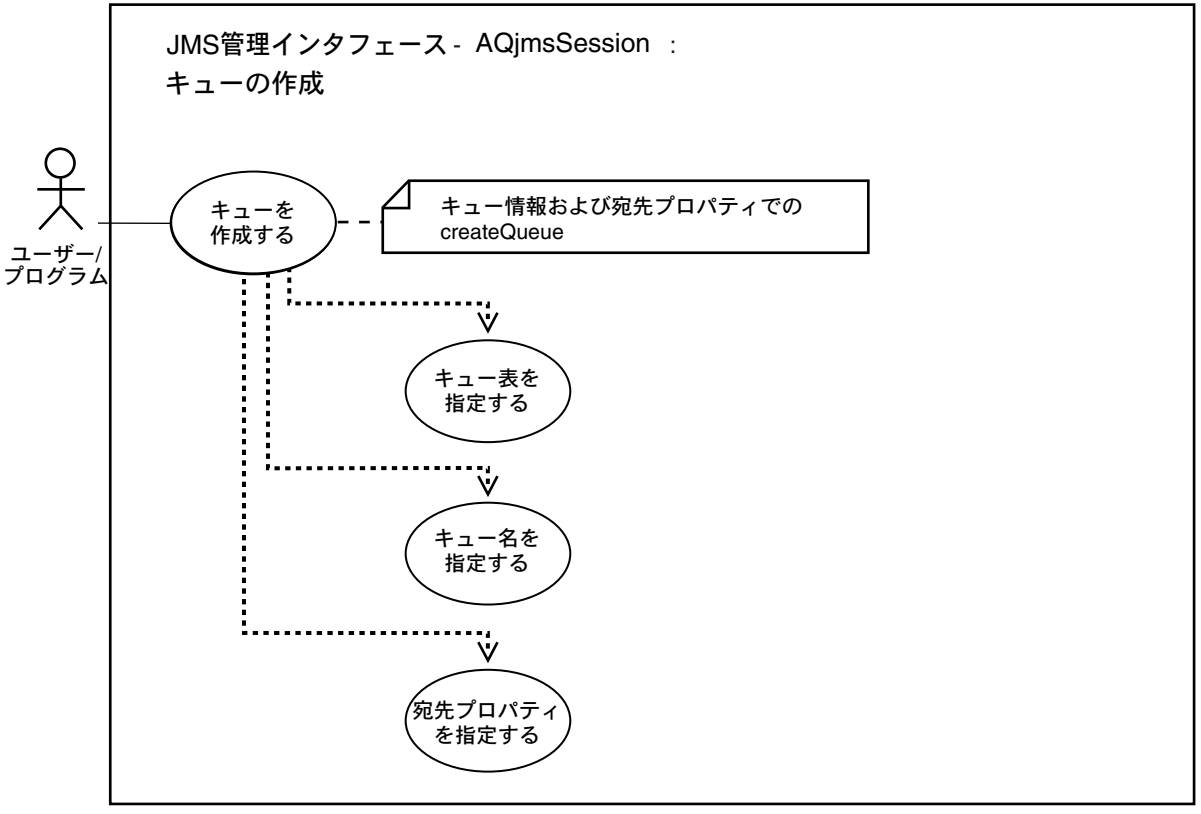
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「`AQjmsDestinationProperty`」を参照してください。

例

今回のリリースでは、例は記載していません。

キューの作成 : Point-to-Point

図 13-17 キューの作成 : Point-to-Point



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

指定されたキュー表内にキューを作成します。

使用上の注意

キューが作成されるキュー表は、シングル・コンシューマ・キュー表である必要があります。

構文

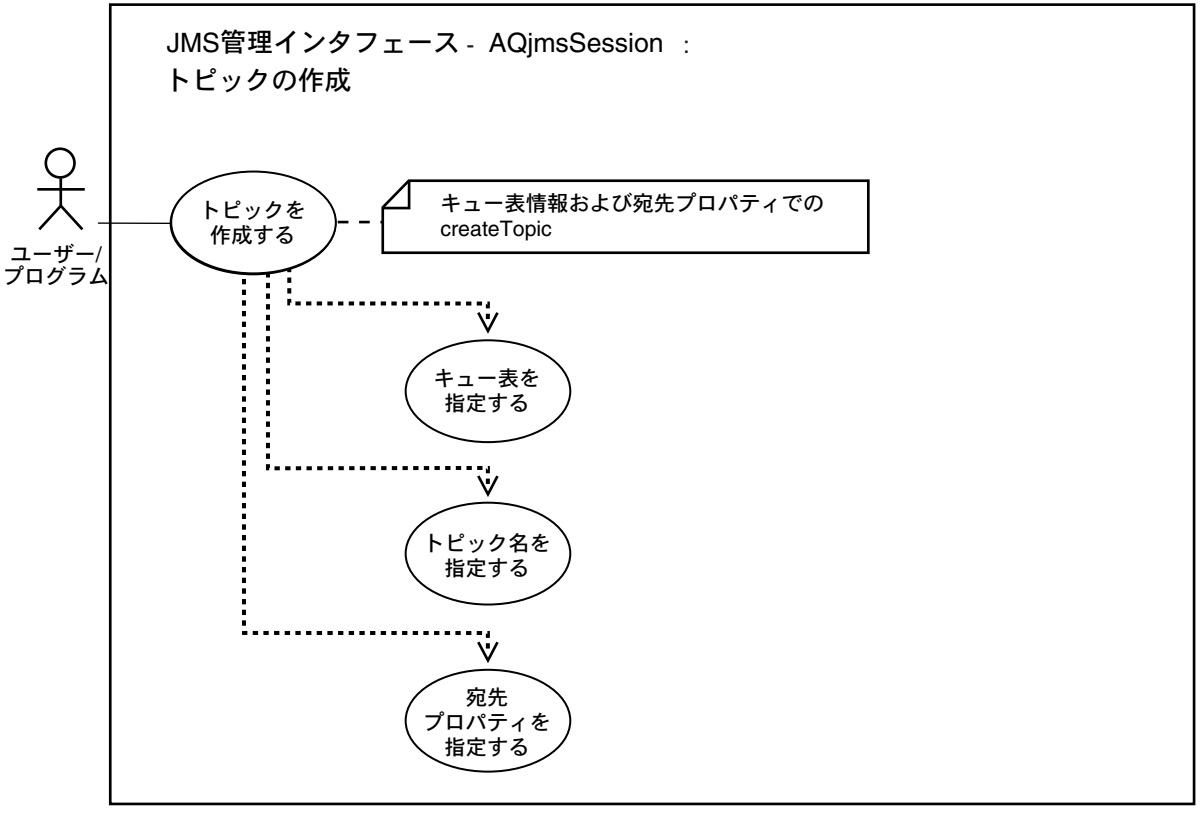
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createQueue` を参照してください。

例

```
QueueSession             q_sess;
AQQueueTable             q_table;
AqjmsDestinationProperty dest_prop;
Queue                    queue;

queue = ((AQjmsSession)q_sess).createQueue(q_table, "jms_q1", dest_prop);
```

トピックの作成 : パブリッシュ・サブスクライブ

図 13-18 トピックの取得 : パブリッシュ・サブスクライブ



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

パブリッシュ・サブスクライブ・モデルでトピックを作成します。

使用上の注意

ありません。

構文

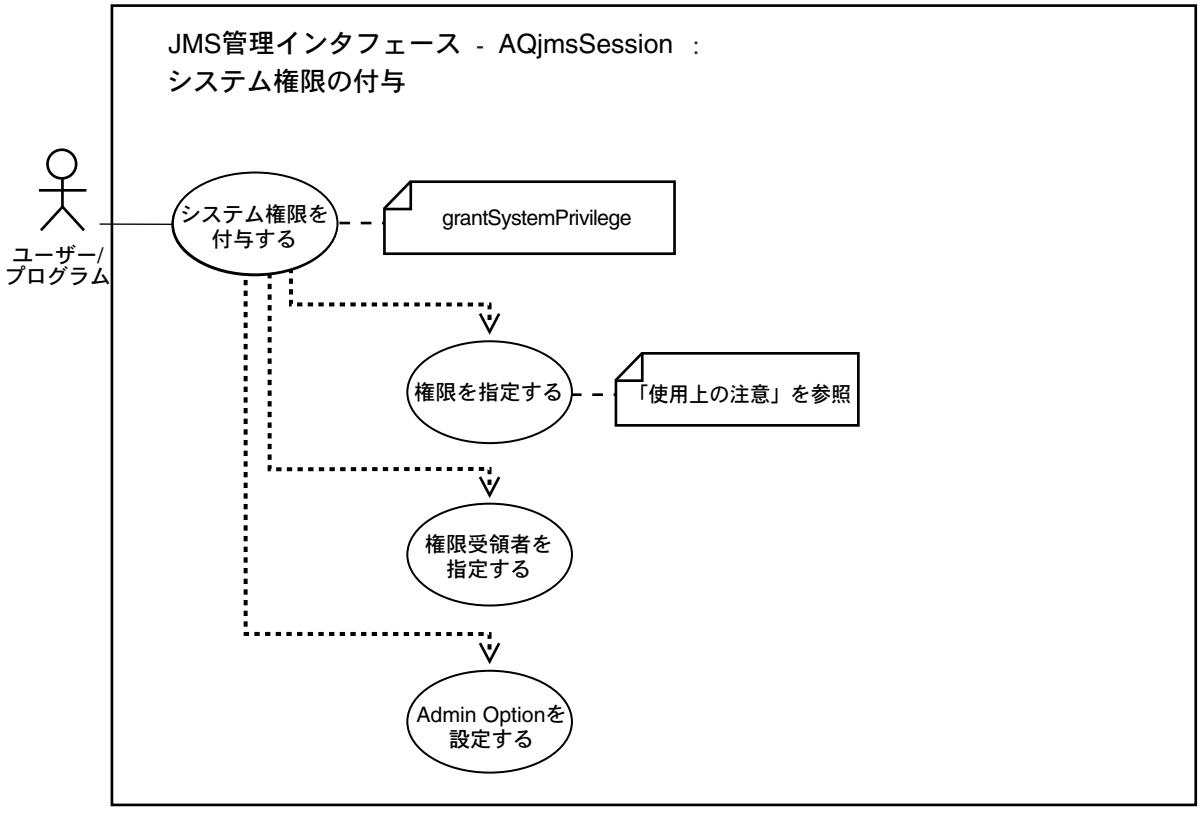
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createTopic` を参照してください。

例

```
TopicSession             t_sess;
AQQueueTable             q_table;
AqjmsDestinationProperty dest_prop;
Topic                    topic;

topic = ((AQjmsSessa)t_sess).createTopic(q_table, "jms_t1", dest_prop);
```

システム権限の付与

図 13-19 システム権限の付与



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

ユーザー / ロールに AQ システム権限を付与します。

使用上の注意

最初は、SYS および SYSTEM のみが、このプロシージャを正常に使用できます。この権限とは、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY です。

構文

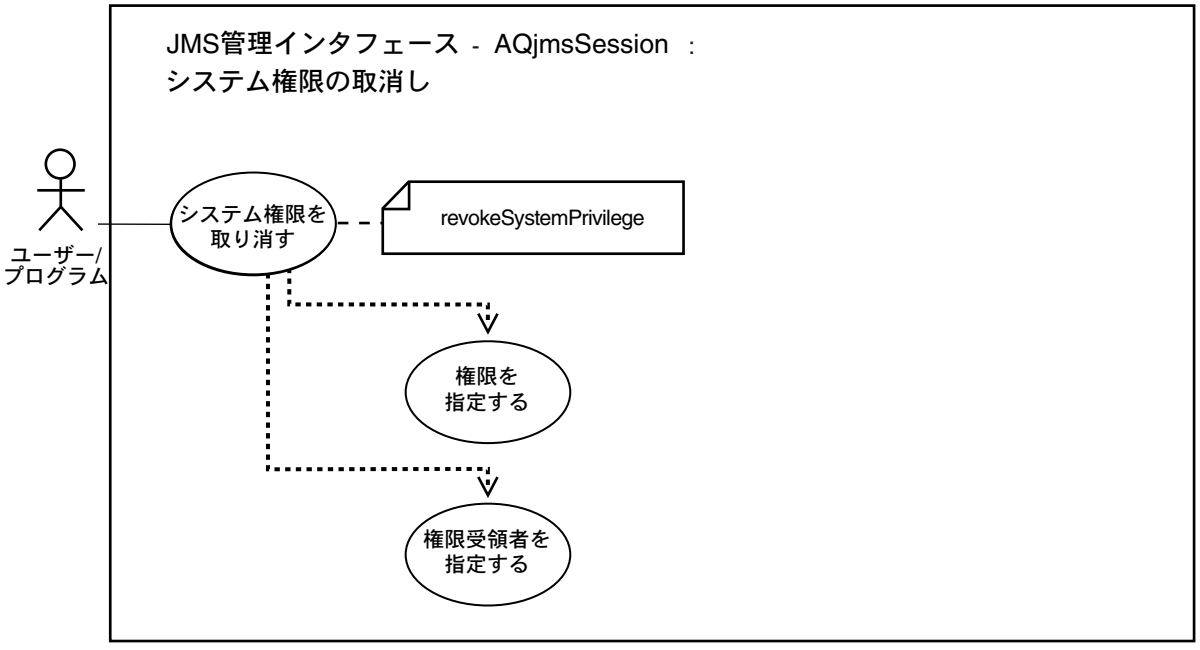
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の grantSystemPrivilege を参照してください。

例

```
TopicSession             t_sess;

((AQjmsSession)t_sess).grantSystemPrivilege("ENQUEUE_ANY", "scott", false);
```

システム権限の取消し

図 13-20 システム権限の取消し



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

ユーザー / ロールの AQ システム権限を取り消します。

使用上の注意

この権限とは、ENQUEUE_ANY、DEQUEUE_ANY および MANAGE_ANY です。

構文

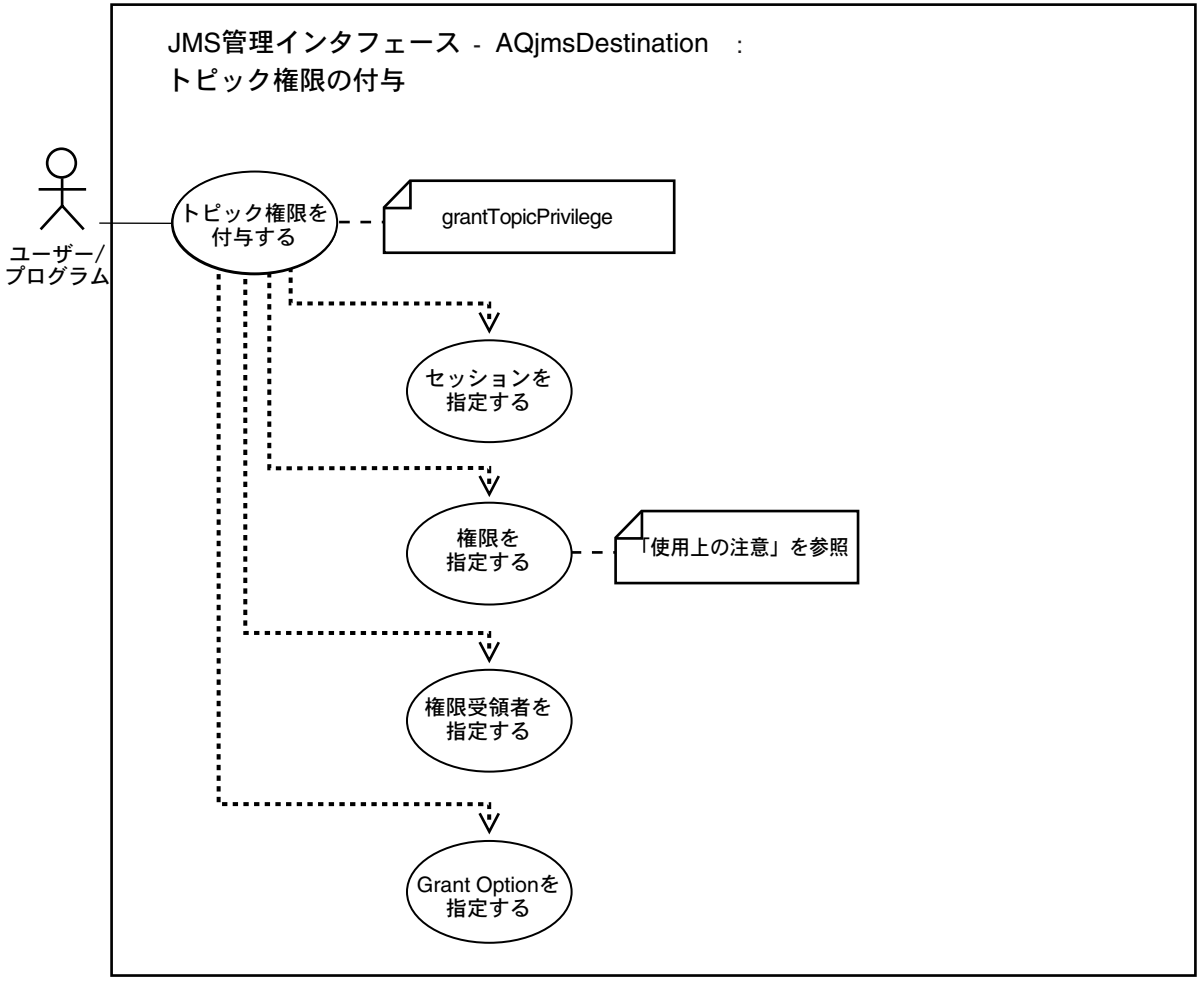
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `revokeSystemPrivilege` を参照してください。

例

```
TopicSession            t_sess;

((AQjmsSession)t_sess).revokeSystemPrivilege("ENQUEUE_ANY", "scott");
```

トピック権限の付与 : パブリッシュ・サブスクライブ

図 13-21 トピック権限の付与 : パブリッシュ・サブスクライブ



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

パブリッシュ・サブスクライブ・モデルにトピック権限を付与します。

使用上の注意

この権限とは、ENQUEUE、DEQUEUE および ALL です。ALL は両方を意味します。最初は、キュー表所有者のみが、このプロシージャを使用してトピックに対する権限を付与できます。

構文

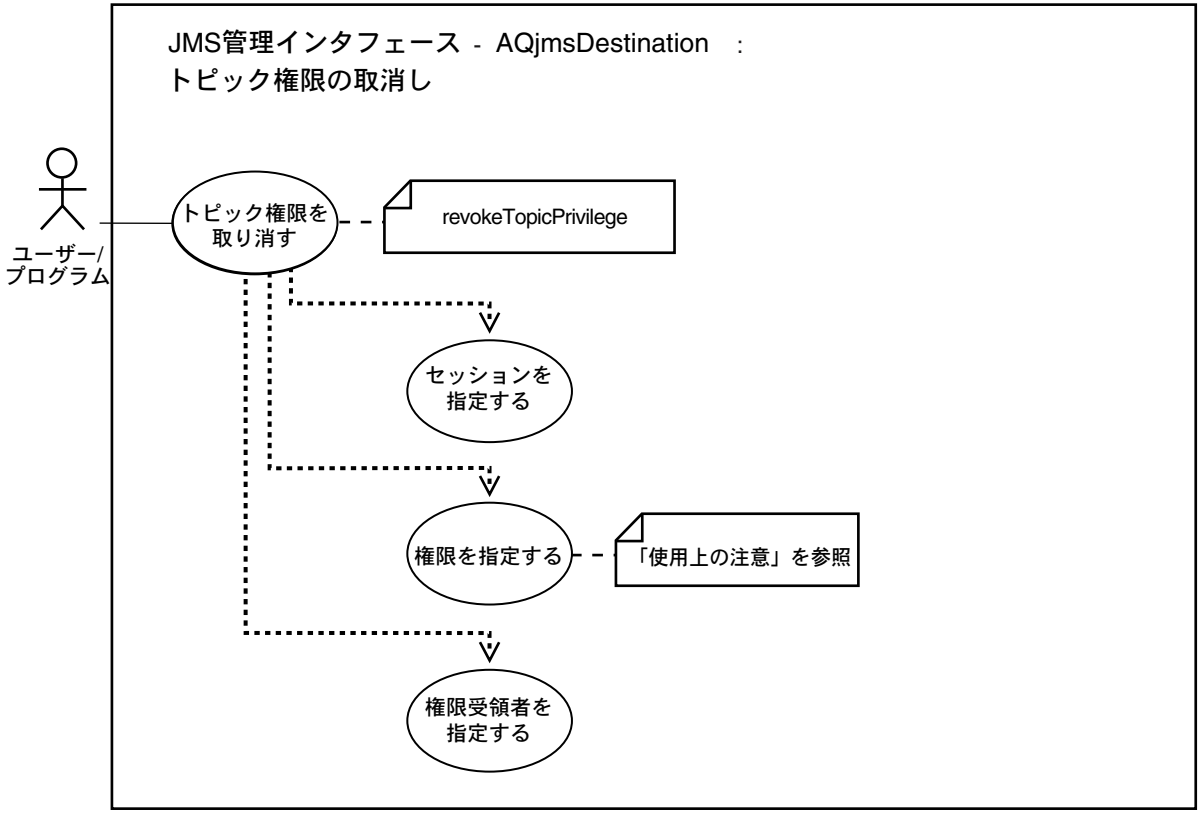
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の grantTopicPrivilege を参照してください。

例

```
TopicSession            t_sess;
Topic                   topic;

((AQjmsDestination)topic).grantTopicPrivilege(t_sess, "ENQUEUE", "scott", false);
```

トピック権限の取消し : パブリッシュ・サブスクライブ

図 13-22 トピック権限の取消し : パブリッシュ・サブスクライブ



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

パブリッシュ・サブスクライブ・モデルでトピック権限を取り消します。

使用上の注意

この権限とは、ENQUEUE、DEQUEUE および ALL です。ALL は両方を意味します。

構文

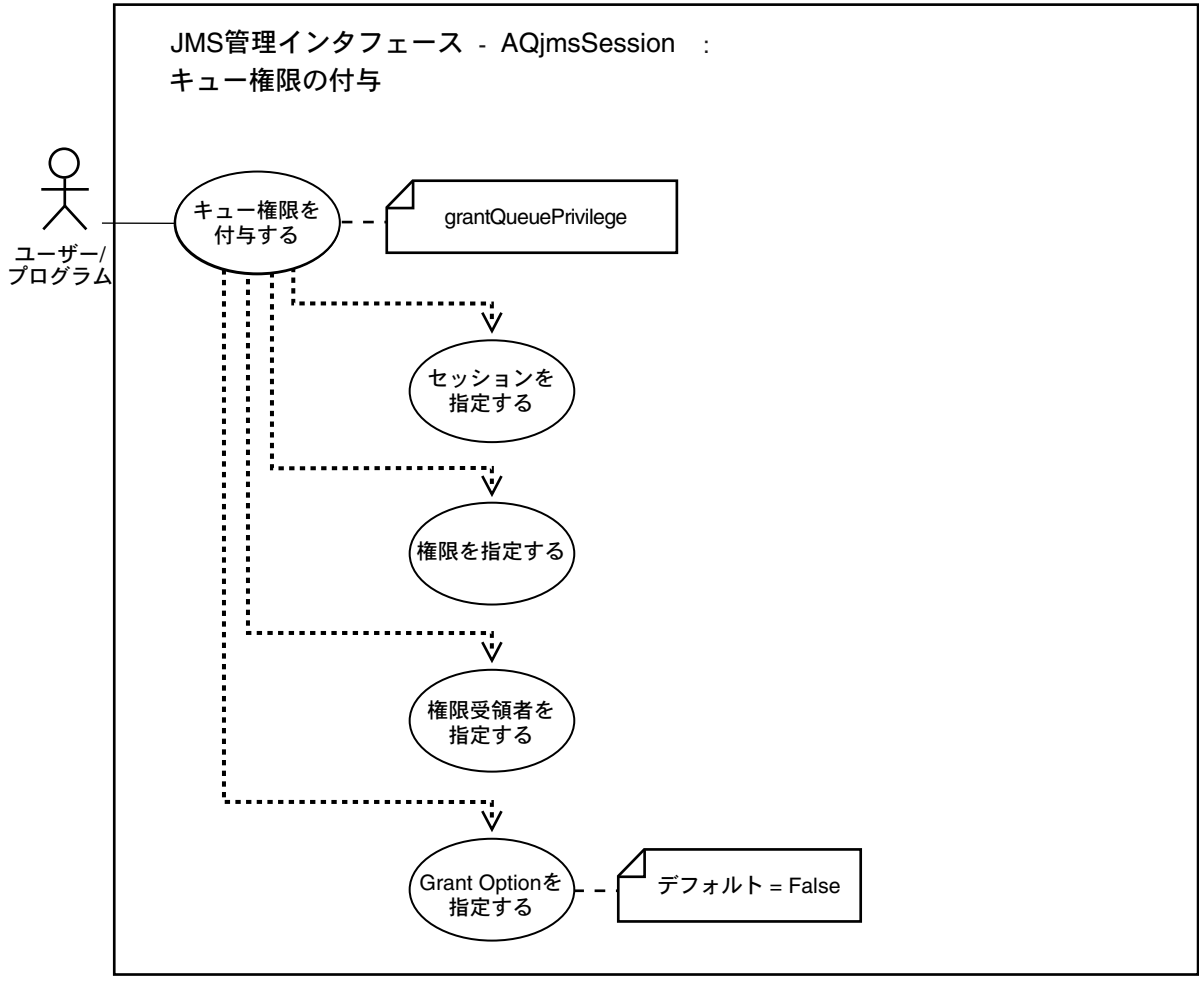
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の revokeTopicPrivilege を参照してください。

例

```
TopicSession            t_sess;
Topic                   topic;

((AQjmsDestination)topic).revokeTopicPrivilege(t_sess, "ENQUEUE", "scott");
```

キュー権限の付与 : Point-to-Point

図 13-23 キュー権限の付与 : Point-to-Point



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

Point-to-Point モデルでキュー権限を付与します。

使用上の注意

この権限とは、ENQUEUE、DEQUEUE および ALL です。ALL は両方を意味します。最初は、キュー表所有者のみが、このプロシージャを使用してキューに対する権限を付与できます。

構文

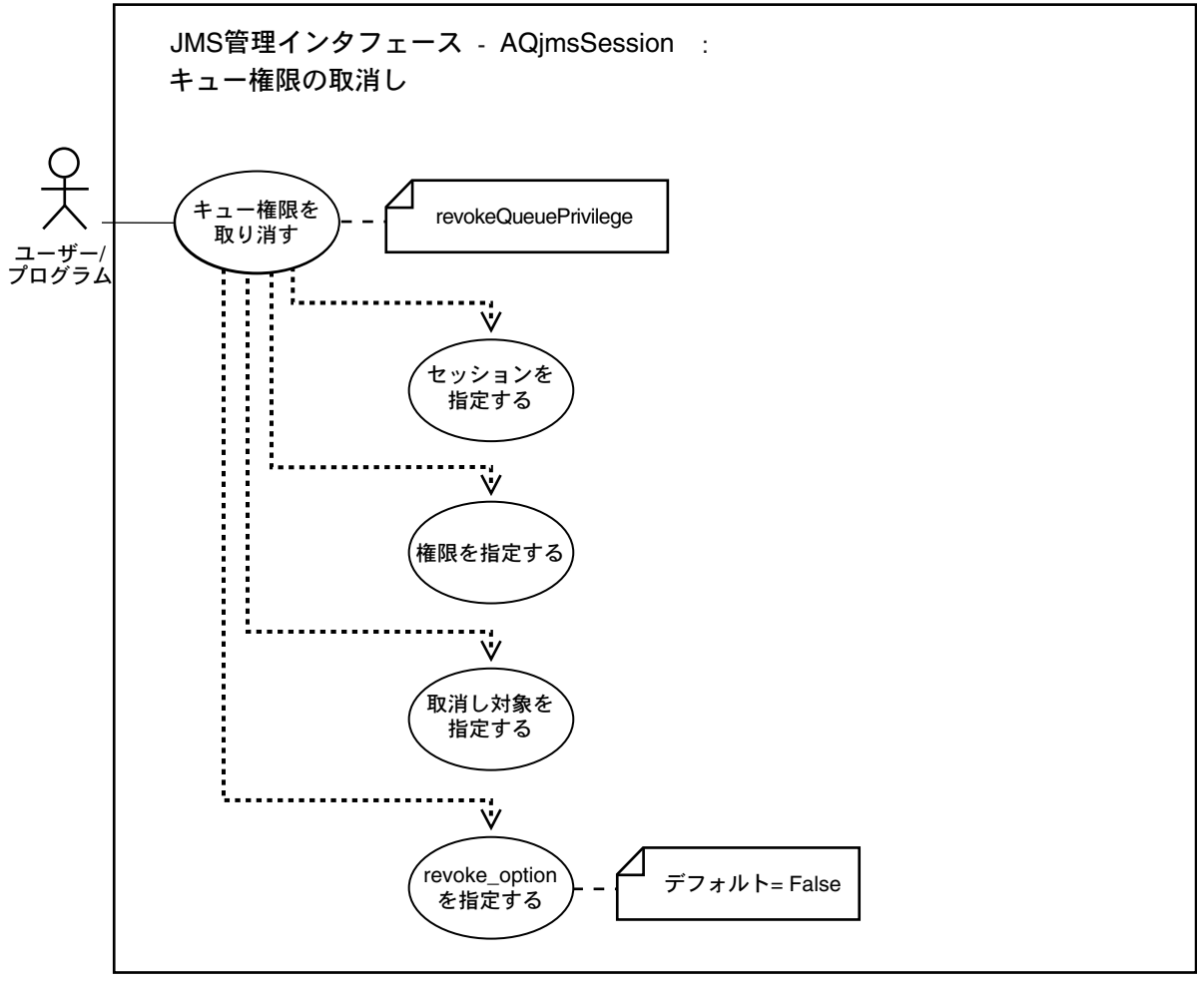
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の grantQueuePrivilege を参照してください。

例

```
QueueSession              q_sess;
Queue                     queue;

((AQjmsDestination)queue).grantQueuePrivilege(q_sess, "ENQUEUE", "scott", false);
```

キュー権限の取消し : Point-to-Point

図 13-24 キュー権限の取消し : Point-to-Point



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

Point-to-Point モデルでキュー権限を取り消します。

使用上の注意

この権限とは、ENQUEUE、DEQUEUE および ALL です。ALL は両方を意味します。権限を取り消すユーザーは、取消しの対象となる権限の付与者である必要があります。GRANT オプションによって伝播された権限は、伝播させた付与者の権限が取り消されたときに取り消されます。

構文

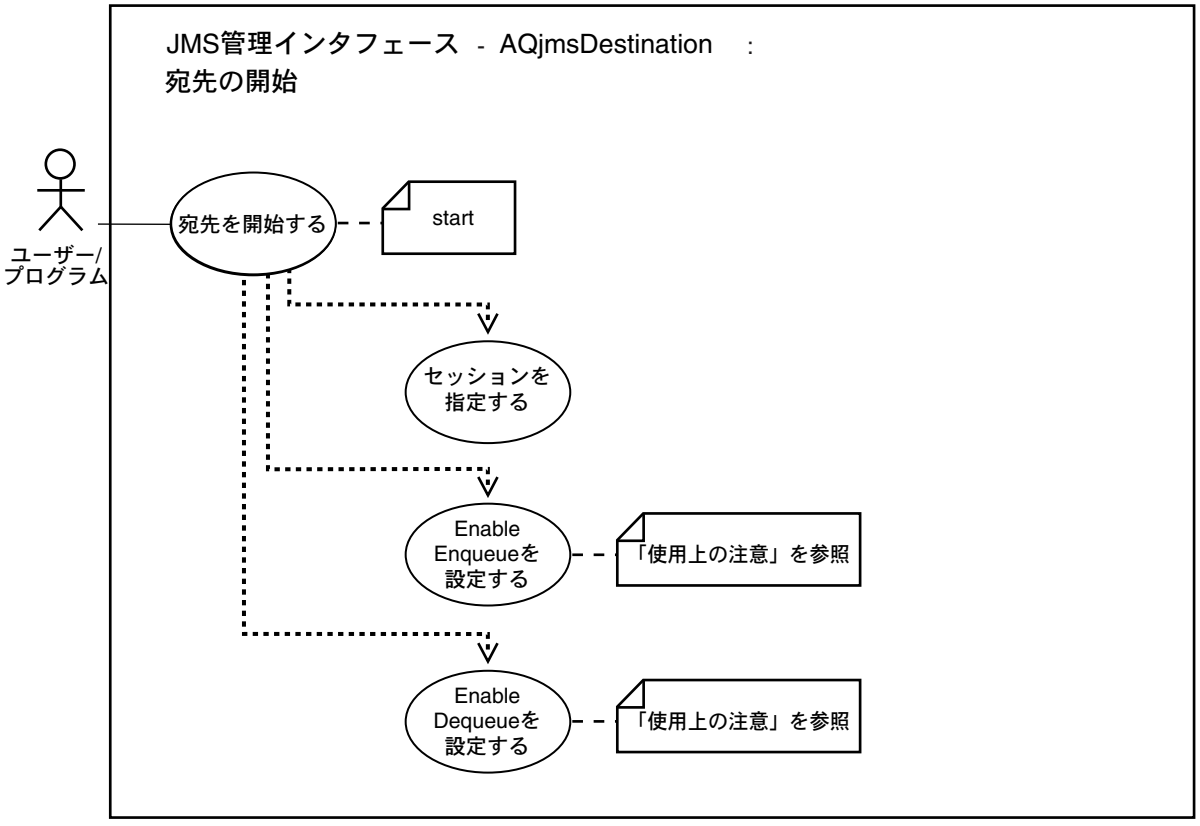
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の revokeQueuePrivilege を参照してください。

例

```
QueueSession            q_sess;
Queue                   queue;

((AQjmsDestination)queue).revokeQueuePrivilege(q_sess, "ENQUEUE", "scott");
```

宛先の開始

図 13-25 宛先の開始



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

宛先を開始します。

使用上の注意

宛先の作成後、管理者は、start メソッドを使用して宛先を使用可能にする必要があります。Enable Enqueue が TRUE に設定されている場合、宛先はエンキュー可能になります。Enable Enqueue が FALSE に設定されている場合、宛先はエンキュー禁止になります。同様に、Enable Dequeue が TRUE に設定されている場合、宛先はデキュー可能になります。Enable Dequeue が FALSE に設定されている場合、宛先はデキュー禁止になります。

構文

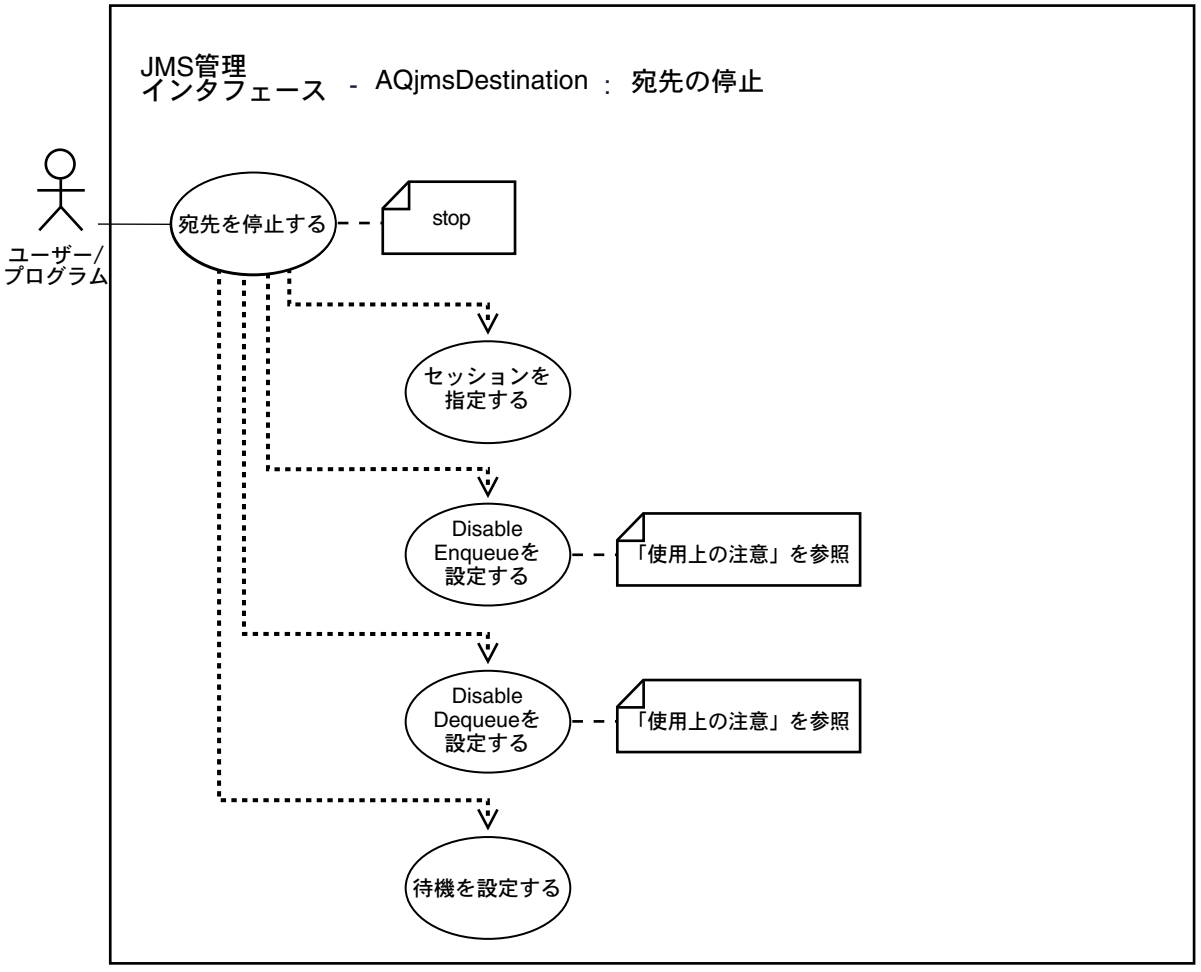
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の start を参照してください。

例

```
TopicSession t_sess;
QueueSession q_sess;
Topic        topic;
Queue        queue;

(AQjmsDestination)topic.start(t_sess, true, true);
(AQjmsDestination)queue.start(q_sess, true, true);
```

宛先の停止

図 13-26 宛先の停止



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

宛先を停止します。

使用上の注意

Disable Dequeue が TRUE に設定されている場合、宛先はデキュー禁止になります。Disable Dequeue が FALSE に設定されている場合、現在の設定は変更されません。同様に Disable Enqueue が TRUE に設定されている場合、宛先はエンキュー禁止になります。Disable Enqueue が FALSE に設定されている場合、現在の設定は変更されません。

構文

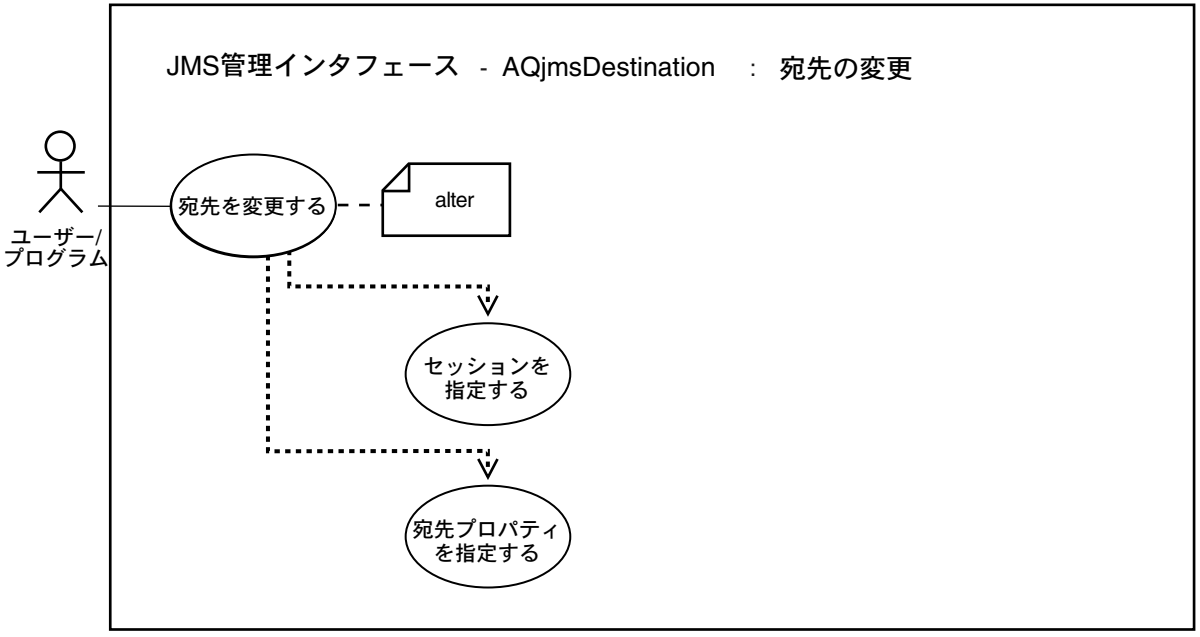
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の stop を参照してください。

例

```
TopicSession t_sess;
Topic        topic;

((AQjmsDestination)topic).stop(t_sess, true, false);
```

宛先の変更

図 13-27 宛先の変更



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

宛先を変更します。

使用上の注意

ありません。

構文

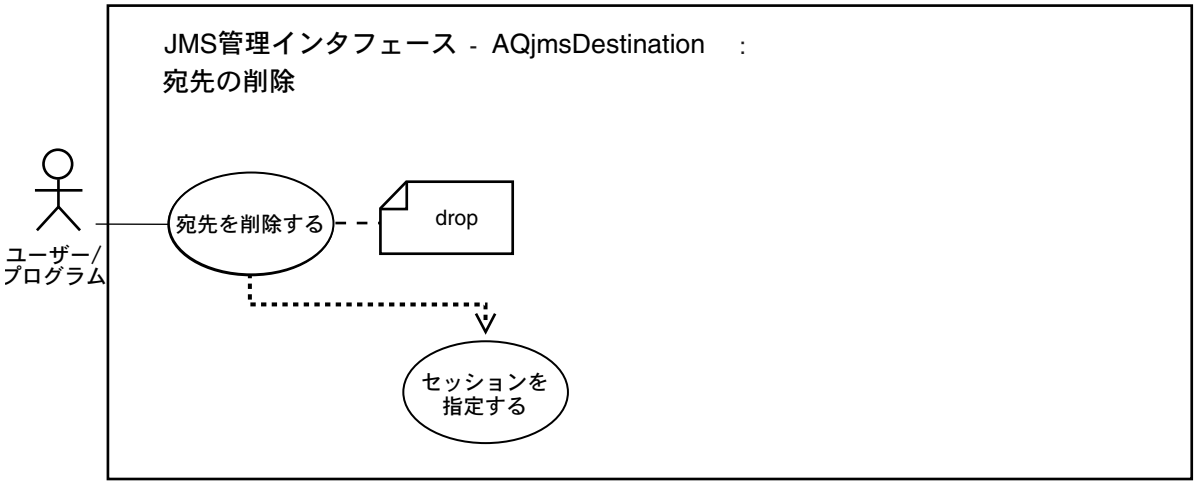
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の `alter` を参照してください。

例

```
QueueSession q_sess;
Queue        queue;
TopicSession t_sess;
Topic        topic;

AQjmsDestionationProperty dest_prop1, dest_prop2;

((AQjmsDestination)queue).alter(dest_prop1);
((AQjmsDestination)topic).alter(dest_prop2);
```

宛先の削除

図 13-28 宛先の削除



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

宛先を削除します。

使用上の注意

ありません。

構文

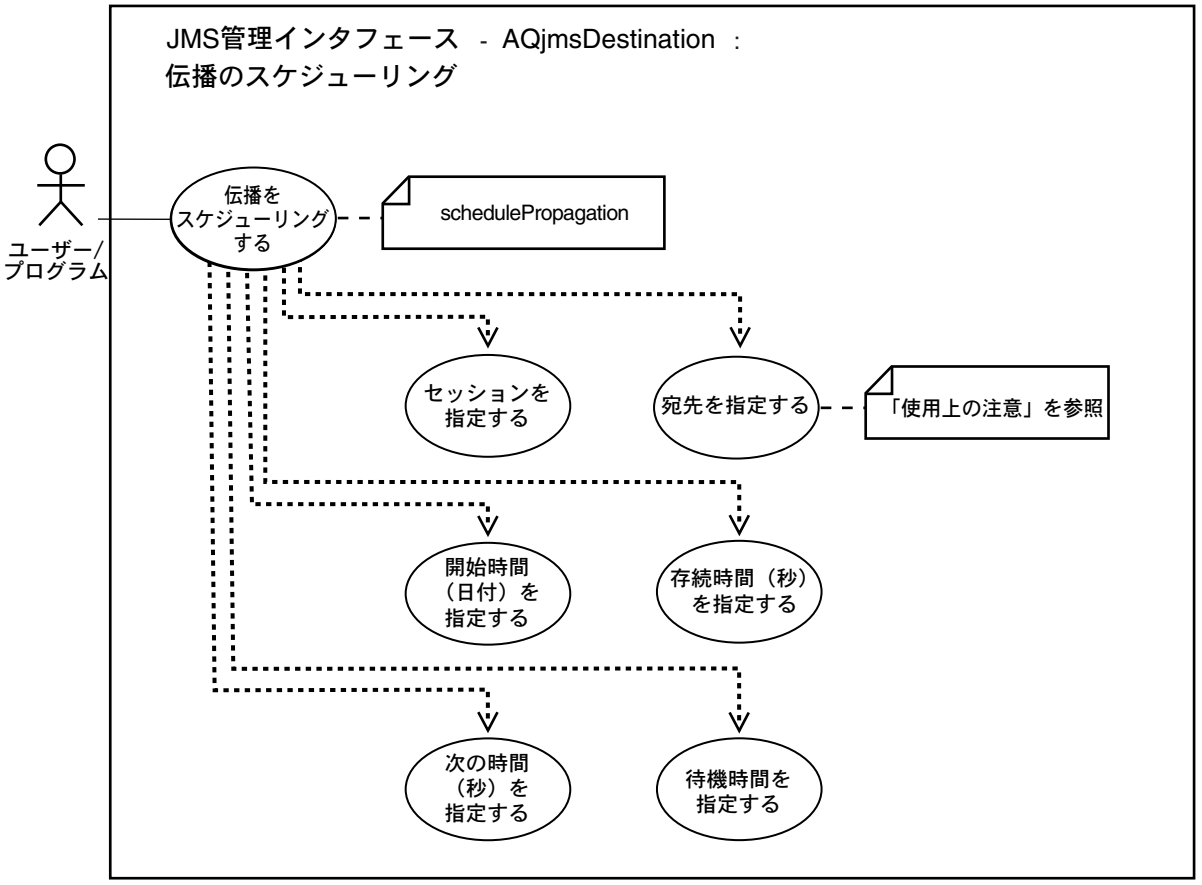
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の `drop` を参照してください。

例

```
QueueSession q_sess;
Queue        queue;
TopicSession t_sess;
Topic        topic;

((AQjmsDestionation)queue).drop(q_sess);
((AQjmsDestionation)topic).drop(t_sess);
```

伝播のスケジューリング

図 13-29 伝播のスケジューリング



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

伝播をスケジューリングします。

使用上の注意

NULL の宛先を指定すると、同じデータベース内の他のトピックにメッセージを伝播できます。メッセージ受信者が、同一または異なるキュー内の同じ宛先に複数存在する場合、メッセージはすべての受信者に同時に伝播されます。

構文

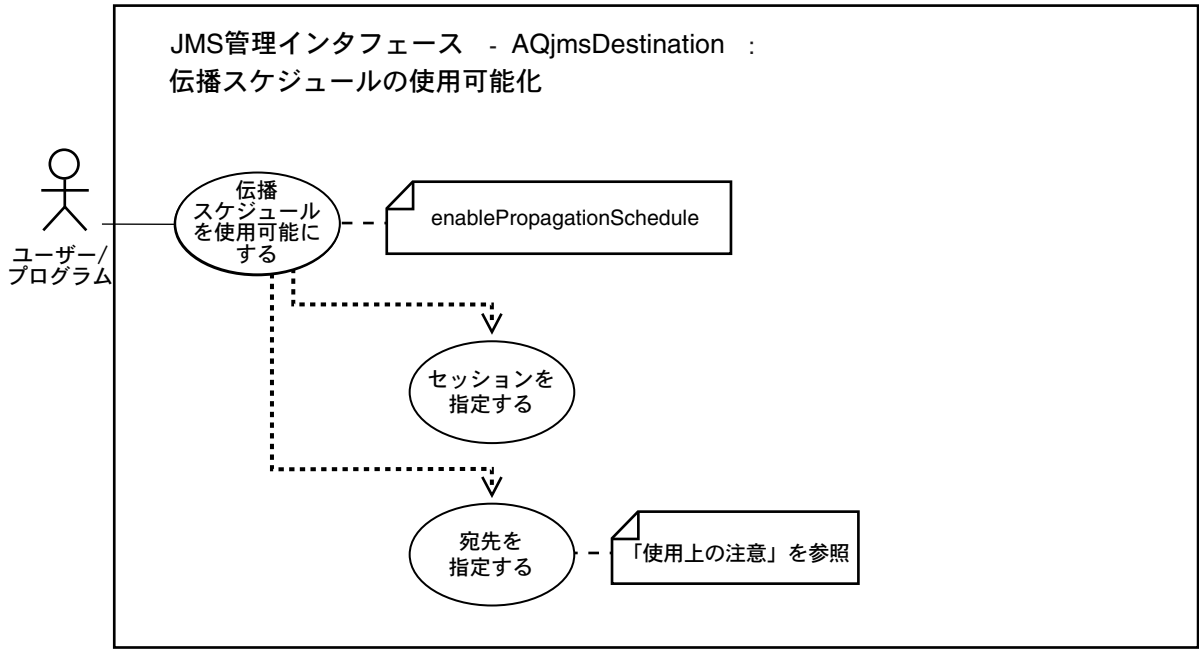
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の schedulePropagation を参照してください。

例

```
TopicSession t_sess;
Topic        topic;

((AQjmsDestination)topic).schedulePropagation(t_sess, null, null, null, null, new
Double(0));
```

伝播スケジュールの使用可能化

図 13-30 伝播スケジュールの使用可能化



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

伝播スケジュールを使用可能にします。

使用上の注意

NULL の宛先を指定すると、ローカル・データベースに伝播されます。

構文

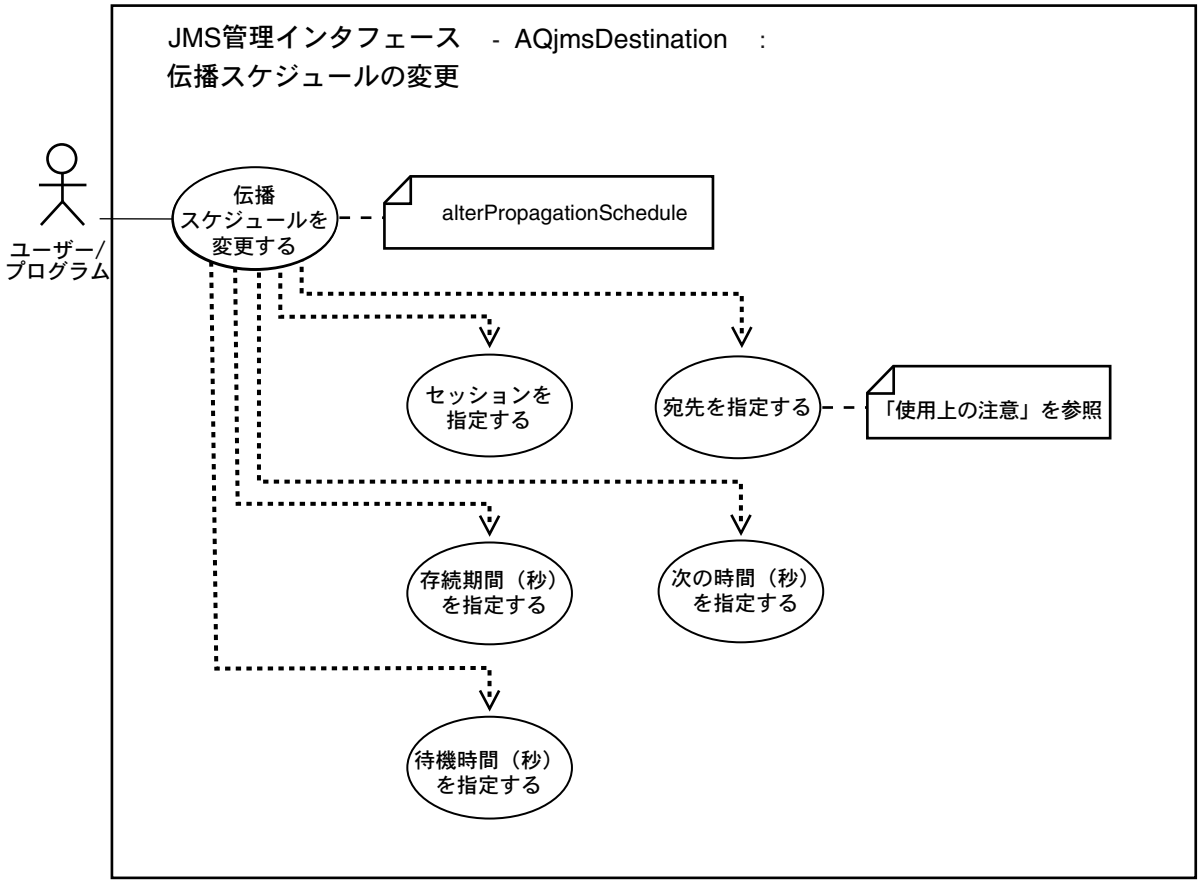
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の `enablePropagationSchedule` を参照してください。

例

```
TopicSession            t_sess;
Topic                   topic;

 ((AQjmsDestination)topic).enablePropagationSchedule(t_sess, "dbs1");
```

伝播スケジュールの変更

図 13-31 伝播スケジュールの変更



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

伝播スケジュールを変更します。

使用上の注意

NULL の宛先を指定すると、ローカル・データベースに伝播されます。

構文

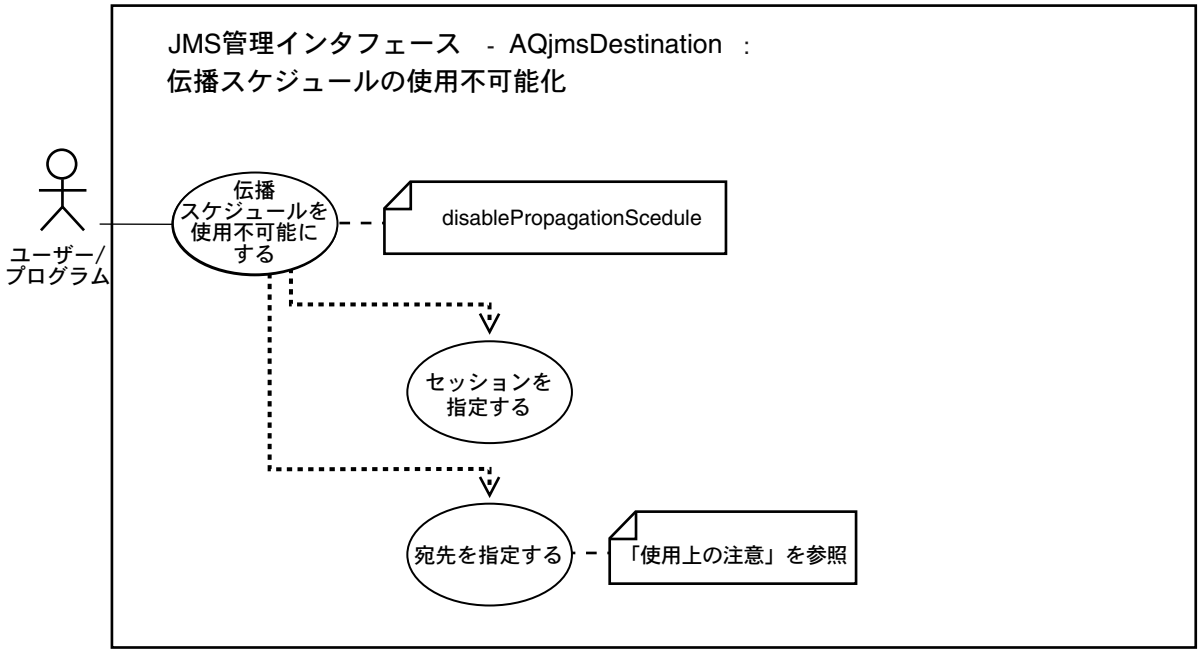
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の `alterPropagationSchedule` を参照してください。

例

```
TopicSession            t_sess;
Topic                   topic;

 ((AQjmsDestination)topic).alterPropagationSchedule(t_sess, null, 30, null, new
Double(30));
```

伝播スケジュールの使用不可能化

図 13-32 伝播スケジュールの使用不可能化



参照：

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

伝播スケジュールを使用不可能にします。

使用上の注意

NULL の宛先を指定すると、ローカル・データベースに伝播されます。

構文

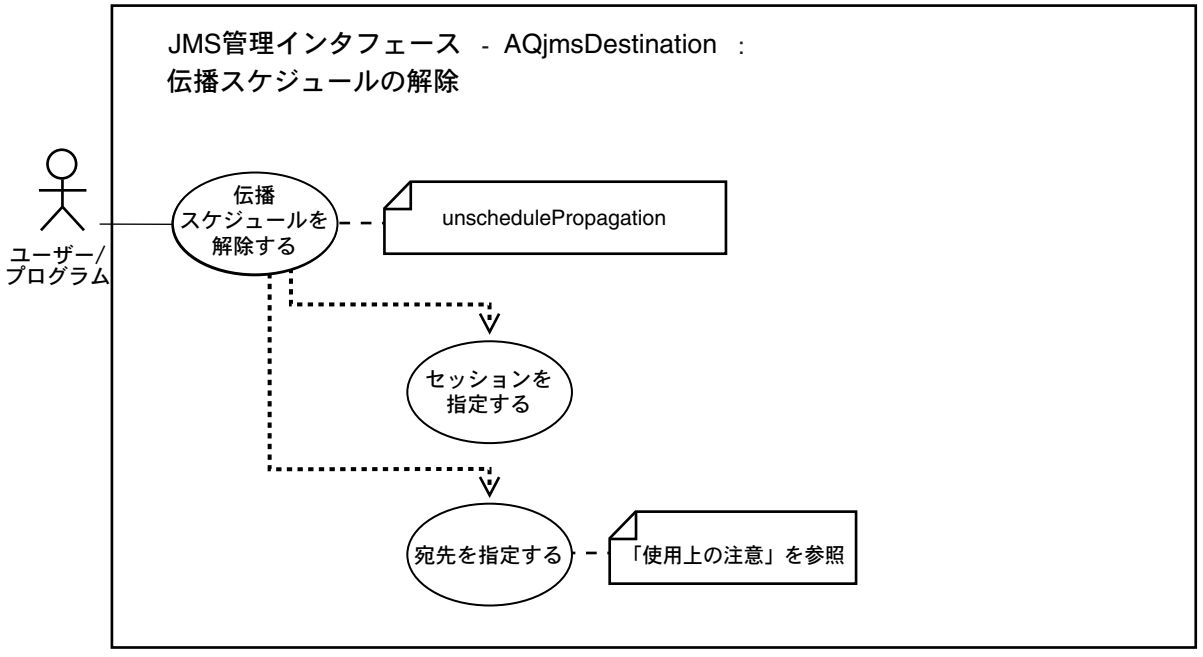
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の `disablePropagationSchedule` を参照してください。

例

```
TopicSession            t_sess;
Topic                   topic;

 ((AQjmsDestination)topic).disablePropagationSchedule(t_sess, "dbs1");
```

伝播スケジュールの解除

図 13-33 伝播スケジュールの解除



参照 :

- JMS 管理インタフェースの基本操作の詳細は、表 13-1 を参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。

用途

伝播スケジュールを解除します。

使用上の注意

以前にスケジューリングされた伝播を解除してください。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsDestination」の `unschedulePropagation` を参照してください。

例

```
TopicSession   t_sess;
Topic          topic;

((AQjmsDestination)topic).unschedulePropagation(t_sess, "dbs1");
```

14

JMS 操作インタフェース : 基本操作（Point-to-Point）

この章では、Oracle Advanced Queuing の操作インタフェースを利用方法に沿って説明します。それぞれの操作（キュー・センダーの作成など）を、その操作名ごとに利用方法に沿って説明します。この章の先頭に、すべての利用方法を示します（14-2 ページの「利用モデル : JMS 操作インタフェース - 基本操作（Point-to-Point）」を参照）。

図 14-1 に、すべての利用方法を 1 つの図にまとめています。HTML 版のマニュアルをご使用の場合、この図の中の関連する利用方法のタイトルをクリックすることで、関心のある利用方法に移動することができます。

個々の利用方法は、次の形式で説明されています。

- **利用図 :** 利用方法を表す図
- **用途 :** この利用方法の用途
- **使用上の注意 :** 実装に有効なガイドライン
- **構文 :** このアクティビティの実行に使用する主な構文
- **例 :** 各プログラム環境での利用例

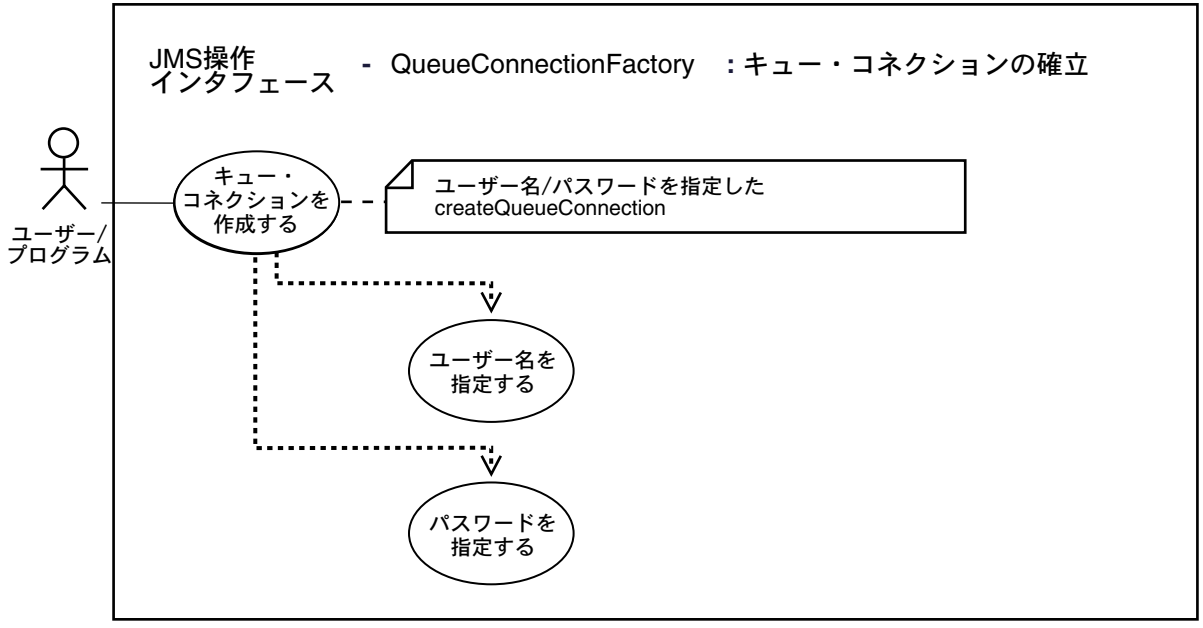
利用モデル : JMS 操作インタフェース - 基本操作（Point-to-Point）

表 14-1 利用モデル : JMS 操作インタフェース - 基本操作（Point-to-Point）

利用方法
キュー・コネクションの確立 : ユーザー名 / パスワードの使用（14-3 ページ）
キュー・コネクションの確立 : オープンしている JDBC コネクションの使用（14-5 ページ）
キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用（14-7 ページ）
キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用（14-9 ページ）
キュー・セッションの作成（14-11 ページ）
キュー・センダーの作成（14-13 ページ）
メッセージの送信 : デフォルト送信オプションのキュー・センダーの使用（14-14 ページ）
メッセージの送信 : 送信オプションを指定したキュー・センダーの使用（14-16 ページ）
Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成（14-19 ページ）
Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成 : メッセージをロック（14-21 ページ）
Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成（14-23 ページ）
Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成 : メッセージをロック（14-26 ページ）
キュー・ブラウザを使用したメッセージのブラウズ（14-28 ページ）
キュー・レシーバの作成 : 標準 JMS 型メッセージ・キュー（14-30 ページ）
キュー・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・キュー（14-32 ページ）
キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用（14-35 ページ）

キュー・コネクションの確立 : ユーザー名 / パスワードの使用

図 14-1 キュー・コネクションの確立 : ユーザー名 / パスワードの使用



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-33 ページの「インタフェース - javax.jms.QueueConnectionFactory」も参照してください。
- 14-5 ページの「キュー・コネクションの確立 : オープンしている JDBC コネクションの使用」も参照してください。
- 14-7 ページの「キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用」も参照してください。
- 14-9 ページの「キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用」も参照してください。

用途
ユーザー名 / パスワードを使用して、キュー・コネクションを確立します。

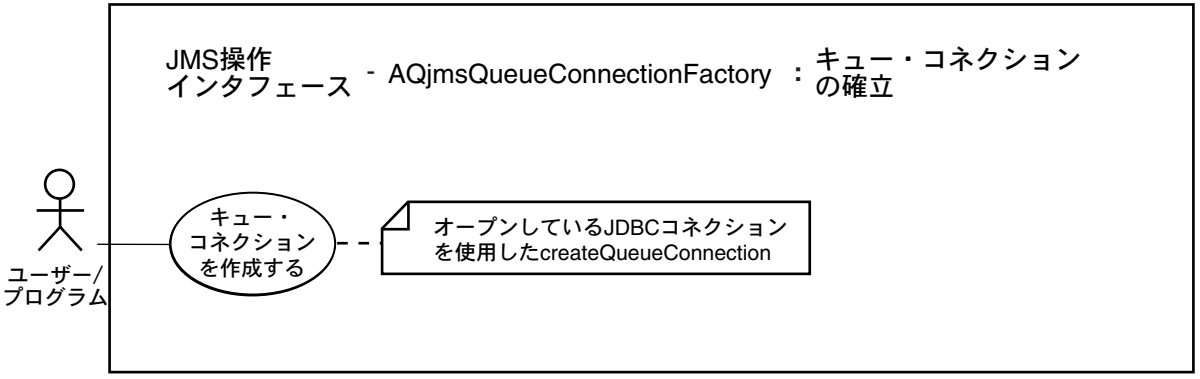
使用上の注意
ありません。

構文
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueConnectionFactory」の `createQueueConnection` を参照してください。

例
```
QueueConnectionFactory qc_fact = AQjmsFactory.getQueueConnectionFactory("sun123",
"oratest", 5521, "thin");
/* Create a queue connection using a username/password */
QueueConnection qc_conn = qc_fact.createQueueConnection("jmsuser", "jmsuser");
```

キュー・コネクションの確立 : オープンしている JDBC コネクションの使用

図 14-2 キュー・コネクションの確立 : オープンしている JDBC コネクションの使用



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsQueueConnectionFactory」も参照してください。
- 14-3 ページの「キュー・コネクションの確立 : ユーザー名 / パスワードの使用」も参照してください。
- 14-7 ページの「キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用」も参照してください。
- 14-9 ページの「キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用」も参照してください。

用途

オープンしている JDBC コネクションを使用して、キュー・コネクションを確立します。

使用上の注意

これは static メソッドです。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueConnectionFactory」の `createQueueConnection` を参照してください。

例

例 1

ユーザーが JMS 操作に対して既存の（たとえばコネクション・プールの）JDBC コネクションを使用するときに、このメソッドが使用されます。この場合、JMS は新しいコネクションをオープンするのではなく、提供された JDBC コネクションを使用して、JMS キュー・コネクション・オブジェクトを作成します。

```
Connection db_conn;     /* previously opened JDBC connection */
 QueueConnection qc_conn = AQjmsQueueConnectionFactory.createQueueConnection(db_
conn);
```

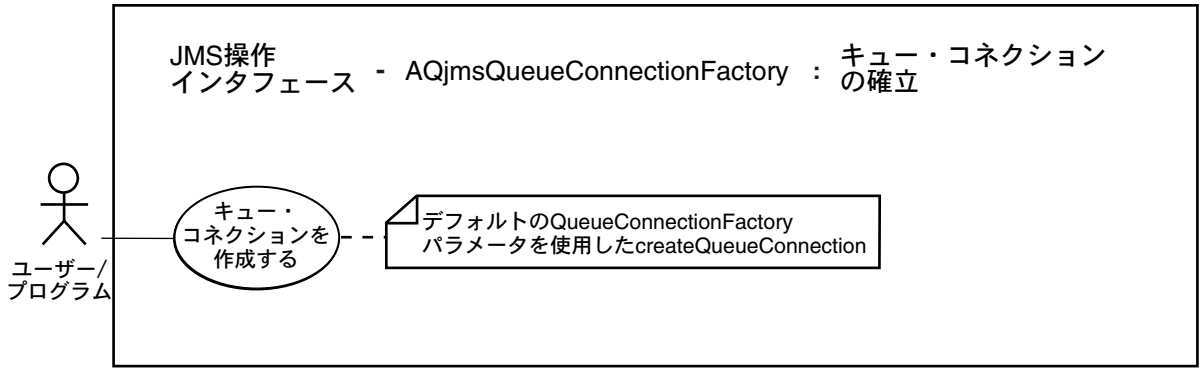
例 2

このメソッドは、データベース（JDBC サーバー・ドライバ）内の Java ストアド・プロシージャから JMS を使用する場合に、JMS キュー・コネクションを確立する唯一の方法です。

```
OracleDriver ora = new OracleDriver();
QueueConnection qc_conn =
AQjmsQueueConnectionFactory.createQueueConnection(ora.defaultConnection());
```

キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用

図 14-3 キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsQueueConnectionFactory」も参照してください。
- 14-3 ページの「キュー・コネクションの確立 : ユーザー名 / パスワードの使用」も参照してください。
- 14-5 ページの「キュー・コネクションの確立 : オープンしている JDBC コネクションの使用」も参照してください。
- 14-9 ページの「キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用」も参照してください。

用途

デフォルトのコネクション・ファクトリ・パラメータを使用して、キュー・コネクションを確立します。

使用上の注意

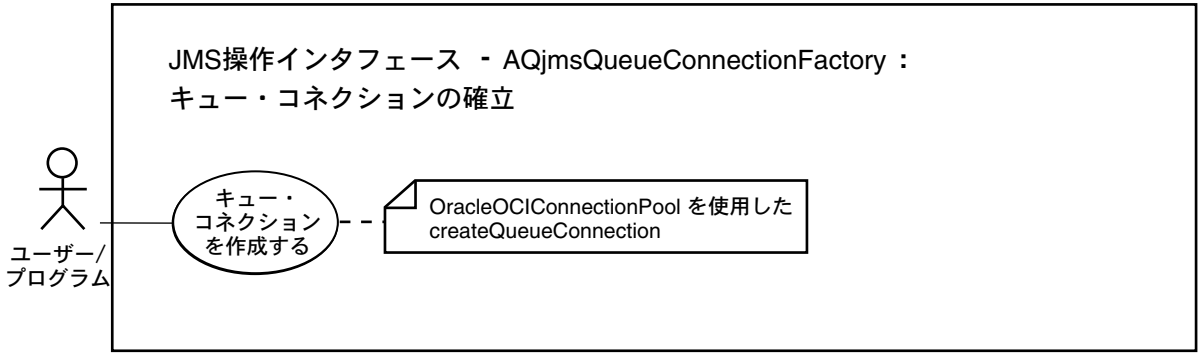
QueueConnectionFactory のプロパティには、デフォルトのユーザー名およびパスワードを含める必要があります。そうでない場合、このメソッドでは JMS 例外が発生します。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueConnectionFactory」の createQueueConnection を参照してください。

キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用

図 14-4 キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsQueueConnectionFactory」も参照してください。
- 14-3 ページの「キュー・コネクションの確立 : ユーザー名 / パスワードの使用」も参照してください。
- 14-5 ページの「キュー・コネクションの確立 : オープンしている JDBC コネクションの使用」も参照してください。
- 14-7 ページの「キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用」も参照してください。

用途
オープンしている OracleOCIConnectionPool を使用して、キュー・コネクションを確立します。

使用上の注意
これは static メソッドです。

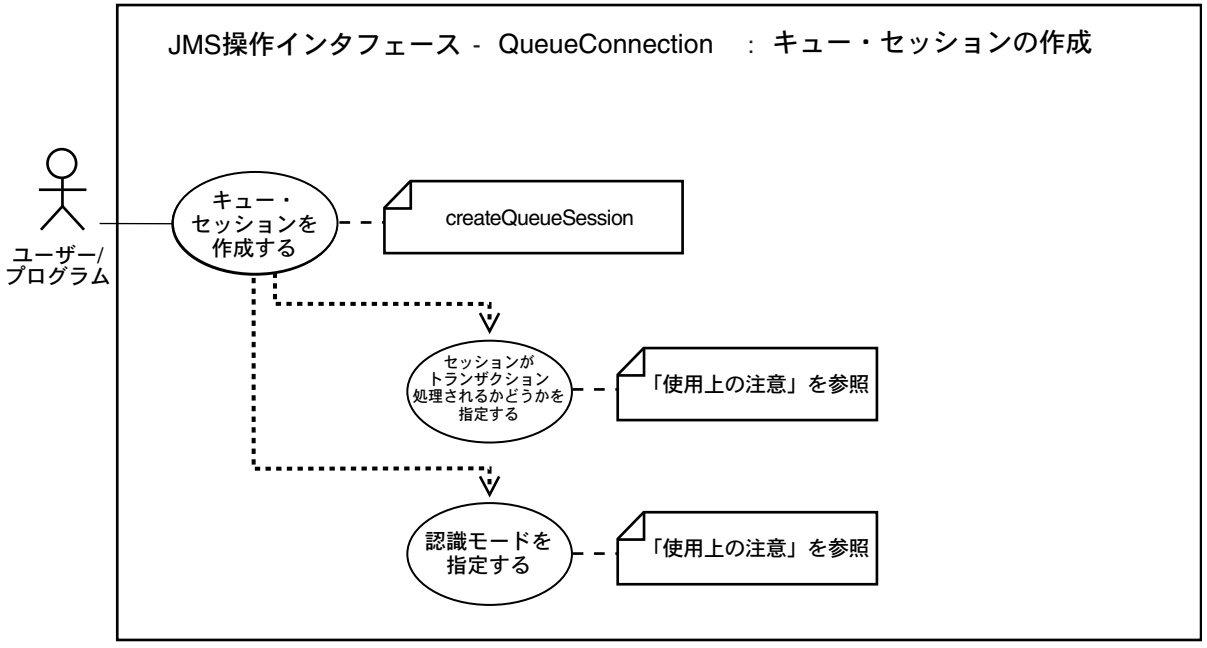
構文
Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueConnectionFactory」の createQueueConnection を参照してください。

例
ユーザーが JMS 操作に対して既存の OracleOCIConnectionPool インスタンスを使用するときに、このメソッドが使用されます。この場合、JMS は新しい OracleOCIConnectionPool インスタンスをオープンするのではなく、提供された OracleOCIConnectionPool インスタンスを使用して、JMS キュー・コネクション・オブジェクトを作成します。

```
OracleOCIConnectionPool cpool; /* previously created OracleOCIConnectionPool */
QueueConnection qc_conn = AQjmsQueueConnectionFactory.createQueueConnection(cpool);
```

キュー・セッションの作成

図 14-5 キュー・セッションの作成



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-33 ページの「インタフェース - javax.jms.QueueConnection」も参照してください。

用途

キュー・セッションを作成します。

使用上の注意

トランザクション処理されるセッションおよびトランザクション処理されないセッションがサポートされます。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `createQueueSession` を参照してください。

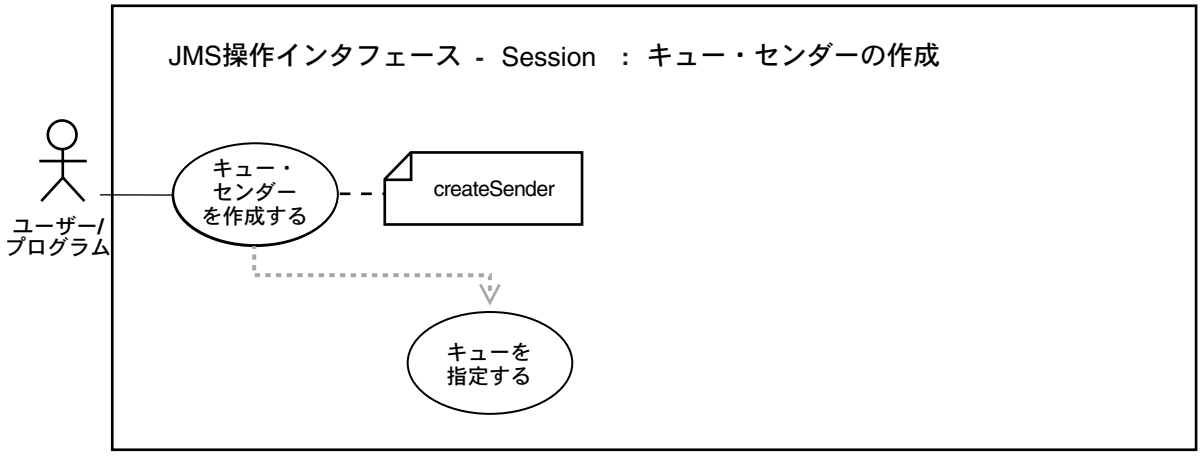
例

トランザクション処理されるセッションの場合、次のとおり作成します。

```
QueueConnection qc_conn;
QueueSession  q_sess = qc_conn.createQueueSession(true, 0);
```

キュー・センダーの作成

図 14-6 キュー・センダーの作成



参照：

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

キュー・センダーを作成します。

使用上の注意

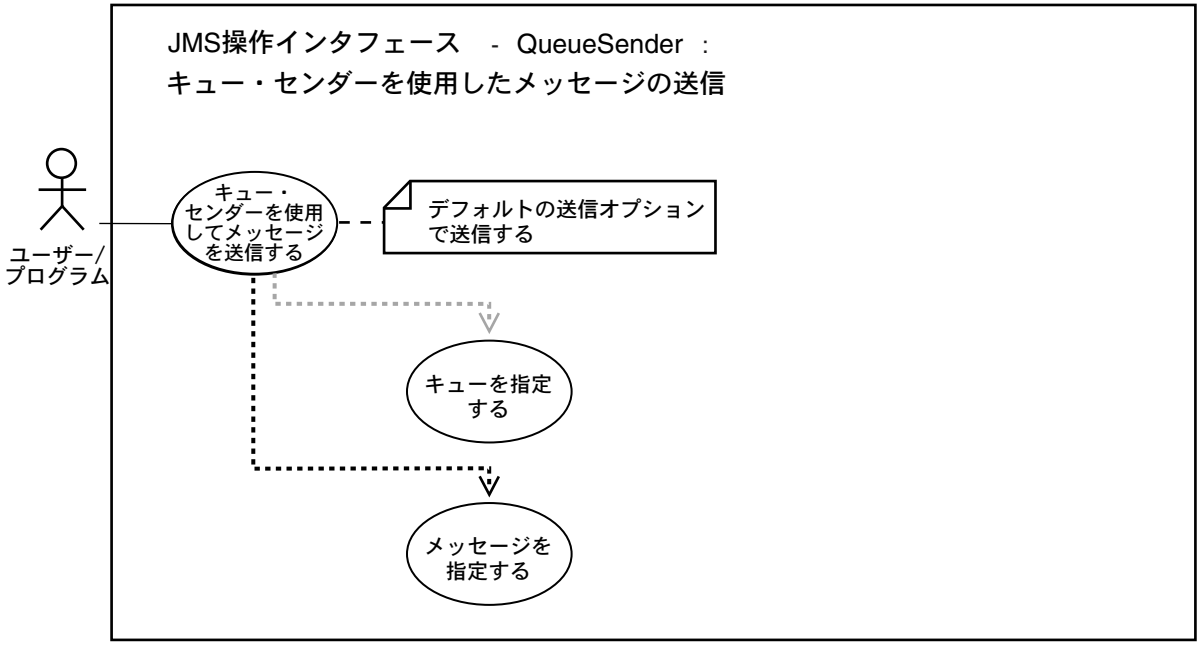
デフォルトのキューを使用しないでセンダーが作成された場合、送信操作を行うたびに宛先キューを指定する必要があります。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createSender` を参照してください。

メッセージの送信 : デフォルト送信オプションのキュー・センダーの使用

図 14-7 メッセージの送信 : デフォルト送信オプションのキュー・センダーの使用



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-34 ページの「インタフェース - javax.jms.QueueSender」も参照してください。

用途

デフォルトの送信オプションを持つキュー・センダーを使用して、メッセージを送信します。

使用上の注意

キュー・センダーがデフォルト・キューで作成されている場合、必ずしも send コールにキュー・パラメータを指定する必要はありません。キューが送信操作で指定されている場合、この値はキュー・センダーのデフォルト・キューをオーバーライドします。

キュー・センダーがデフォルト・キューを使用しないで作成されている場合、send コールごとにキュー・パラメータを指定する必要があります。

この送信操作では、メッセージの優先順位（1）および Time-To-Live（無制限）のデフォルト値が使用されます。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueSender」の send を参照してください。

例

例 1

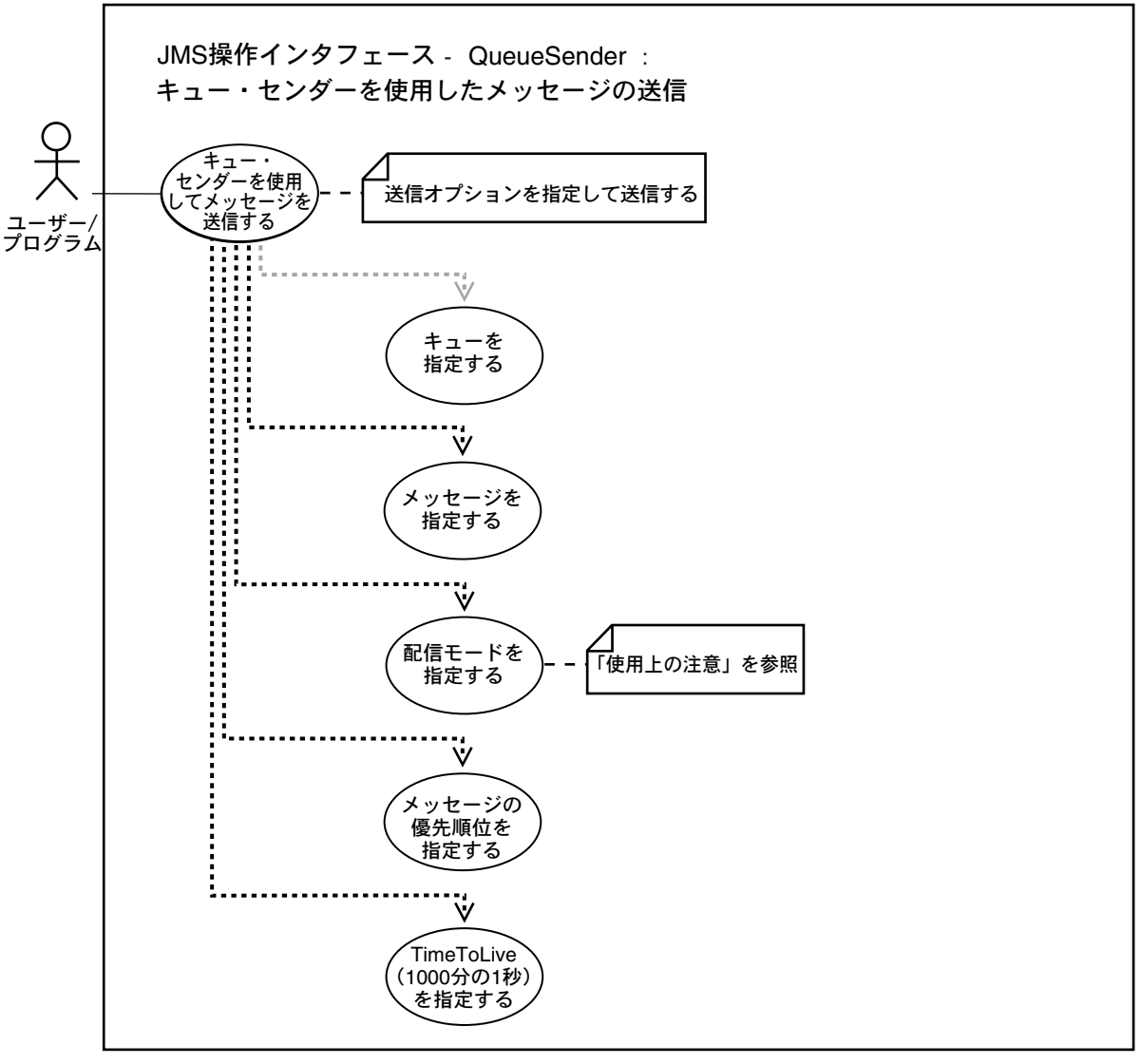
```
/* Create a sender to send messages to any queue */
QueueSession  jms_sess;
QueueSender  sender1;
TextMessage  message;
sender1 = jms_sess.createSender(null);
sender1.send(queue, message);
```

例 2

```
/* Create a sender to send messages to a specific queue */
QueueSession jms_sess;
QueueSender sender2;
Queue   billed_orders_que;
TextMessage  message;
sender2 = jms_sess.createSender(billed_orders_que);
sender2.send(queue, message);
```

メッセージの送信 : 送信オプションを指定したキュー・センダーの使用

図 14-8 メッセージの送信 : 送信オプションを指定したキュー・センダーの使用



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-34 ページの「インタフェース - javax.jms.QueueSender」も参照してください。

用途

送信オプションを指定することによって、キュー・センダーを使用してメッセージを送信します。

使用上の注意

キュー・センダーがデフォルト・キューで作成されている場合、必ずしも send コールにキュー・パラメータを指定する必要はありません。キューが送信操作で指定されている場合、この値はキュー・センダーのデフォルト・キューをオーバーライドします。

キュー・センダーがデフォルト・キューを使用しないで作成されている場合、send コールごとにキュー・パラメータを指定する必要があります。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueSender」の send を参照してください。

例

例 1

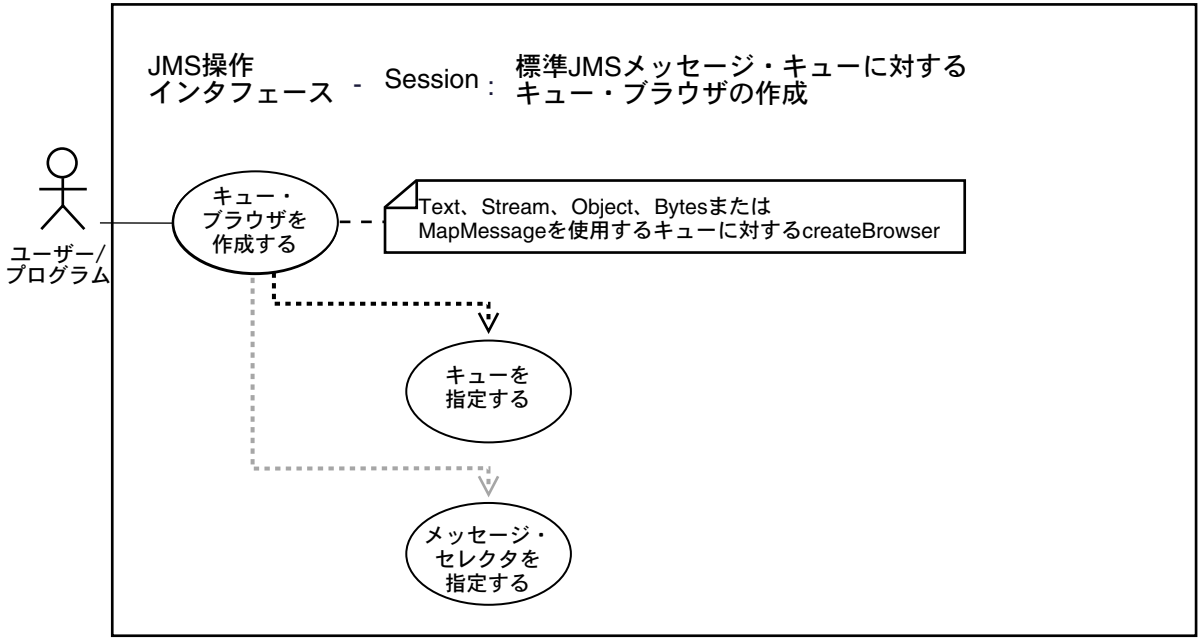
```
/* Create a sender to send messages to any queue */
/* Send a message to new_orders_que with priority 2 and timetoLive 100000
   milliseconds */
QueueSession  jms_sess;
QueueSender  sender1;
TextMessage mesg;
Queue   new_orders_que
sender1 = jms_sess.createSender(null);
sender1.send(new_orders_que, mesg, DeliveryMode.PERSISTENT, 2, 100000);
```

例 2

```
/* Create a sender to send messages to a specific  queue */
/* Send a message with priority 1 and timetoLive 400000 milliseconds */
QueueSession jms_sess;
QueueSender sender2;
Queue   billed_orders_que;
TextMessage mesg;
sender2 = jms_sess.createSender(billed_orders_que);
sender2.send(mesg, DeliveryMode.PERSISTENT, 1, 400000);
```

Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成

図 14-9 Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成



参照：

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

Text、Stream、Object、Bytes または MapMessage を使用するキューに対するキュー・ブラウザを作成します。

使用上の注意

特定の基準に一致するメッセージを取り出すには、キュー・ブラウザに対するセレクタは、次の 1 つ以上の組合せによる式で指定します。

- JMS メッセージ ID

  ```
  JMSMessageID = 'ID:23452345'
  ```

  指定されたメッセージ ID を持つメッセージを取り出します。

- JMS メッセージ・ヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

すべてのメッセージ ID には、接頭辞 ID: が必要です。

`java.util.Enumeration` 内のメソッドを使用して、メッセージのリストを参照してください。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createBrowser` を参照してください。

例

例 1

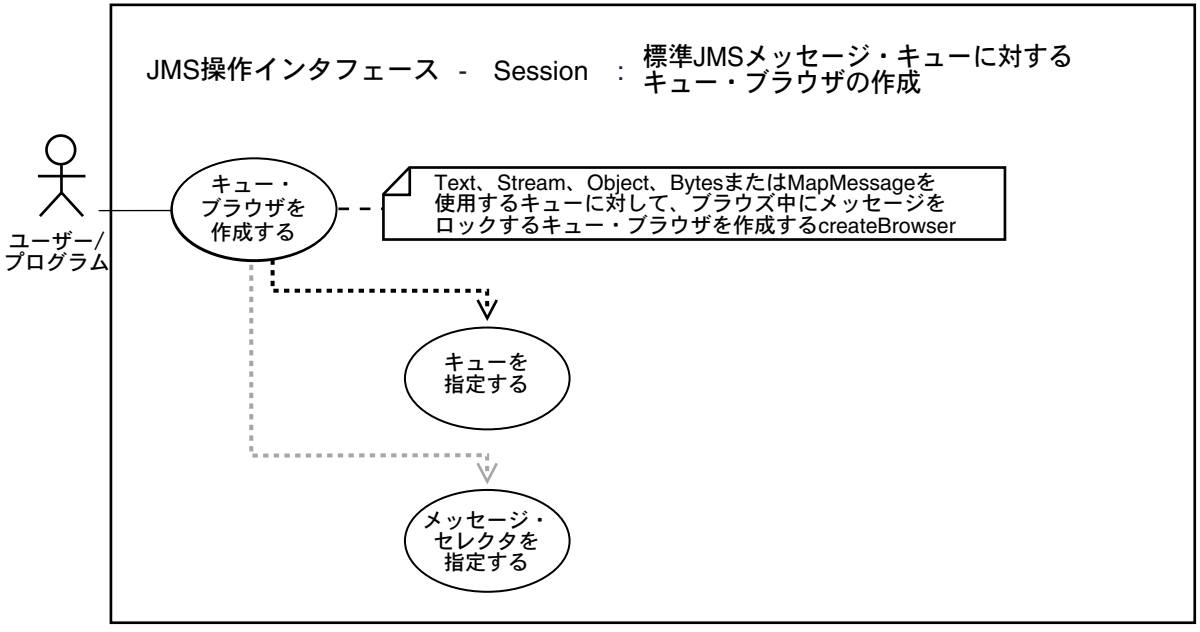
```
/* Create a browser without a selector */
QueueSession    jms_session;
QueueBrowser    browser;
Queue           queue;

browser = jms_session.createBrowser(queue);
```

例 2

```
/* Create a browser for queues with a specified selector */
QueueSession    jms_session;
QueueBrowser    browser;
Queue           queue;

/* create a Browser to look at messages with correlationID = RUSH  */
browser = jms_session.createBrowser(queue, "JMSCorrelationID = 'RUSH'");
```

Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成 : メッセージをロック

図 14-10 Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成 : メッセージをロック



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

Text、Stream、Object、Bytes または MapMessage を使用するキューに対して、ブラウズ中にメッセージをロックするキュー・ブラウザを作成します。

使用上の注意

locked パラメータが TRUE に指定されている場合、ブラウズ中のメッセージはロックされます。したがって、ブラウズ中のセッションがトランザクションを終了するまで、他のコンシューマがこれらのメッセージを削除することはできません。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の createBrowser を参照してください。

例

例 1

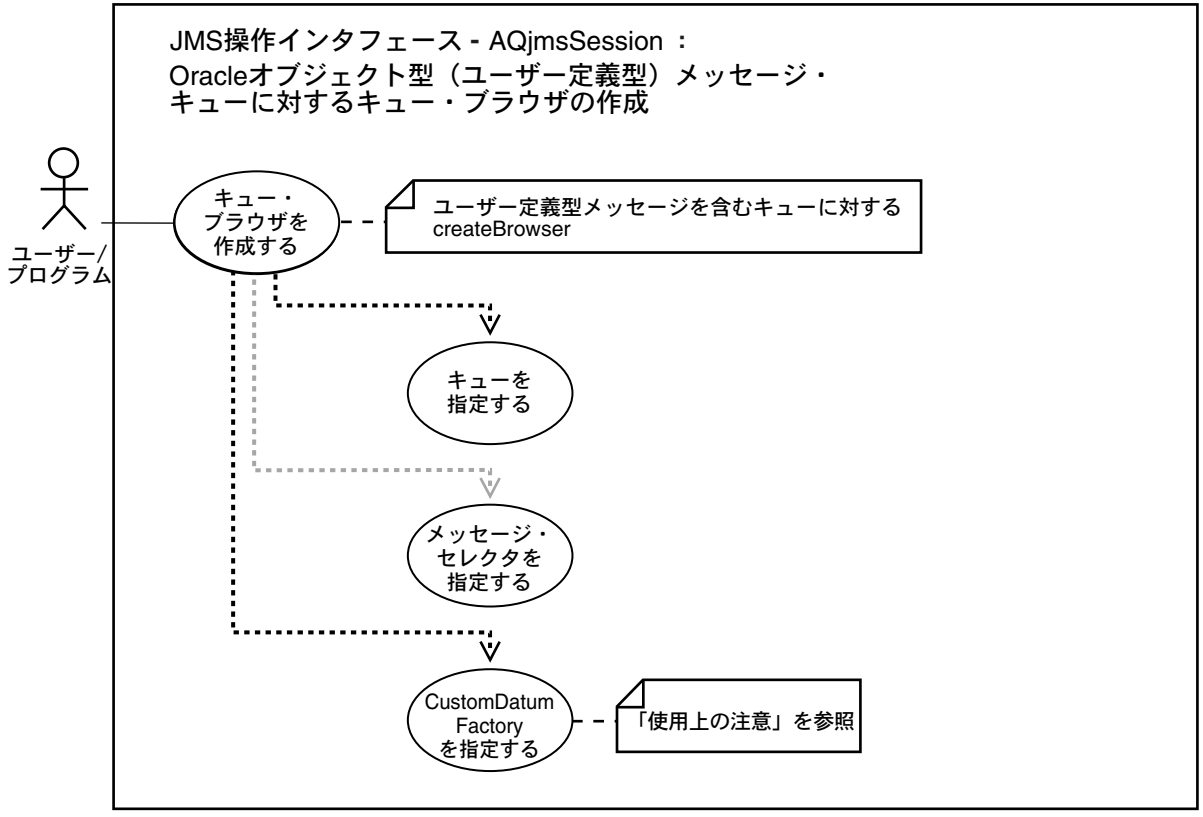
```
/* Create a browser without a selector */
QueueSession    jms_session;
QueueBrowser    browser;
Queue           queue;

browser = jms_session.createBrowser(queue, null, true);
```

例 2

```
/* Create a browser for queues with a specified selector */
QueueSession    jms_session;
QueueBrowser    browser;
Queue           queue;

/* create a Browser to look at messages with
correlationID = RUSH in lock mode */

browser = jms_session.createBrowser(queue, "JMSCorrelationID = 'RUSH'", true);
```

Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成

図 14-11 Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成



参照：

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザを作成します。

使用上の注意

AdtMessage を含むキューの場合、キュー・ブラウザに対するセレクタは、メッセージ・ペイロード内容、メッセージ ID、優先順位または相関識別子に対する SQL 式で指定します。

- メッセージ ID に対するセレクタは、次のように指定します。指定したメッセージ ID を持つメッセージが取り出されます。

  ```
  msgid = '23434556566767676'
  ```

  注意： この場合は、メッセージ ID に接頭辞 ID: を使用しないでください。

- 優先順位または相関識別子に対するセレクタは、次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードに対するセレクタは、次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の createBrowser を参照してください。

例

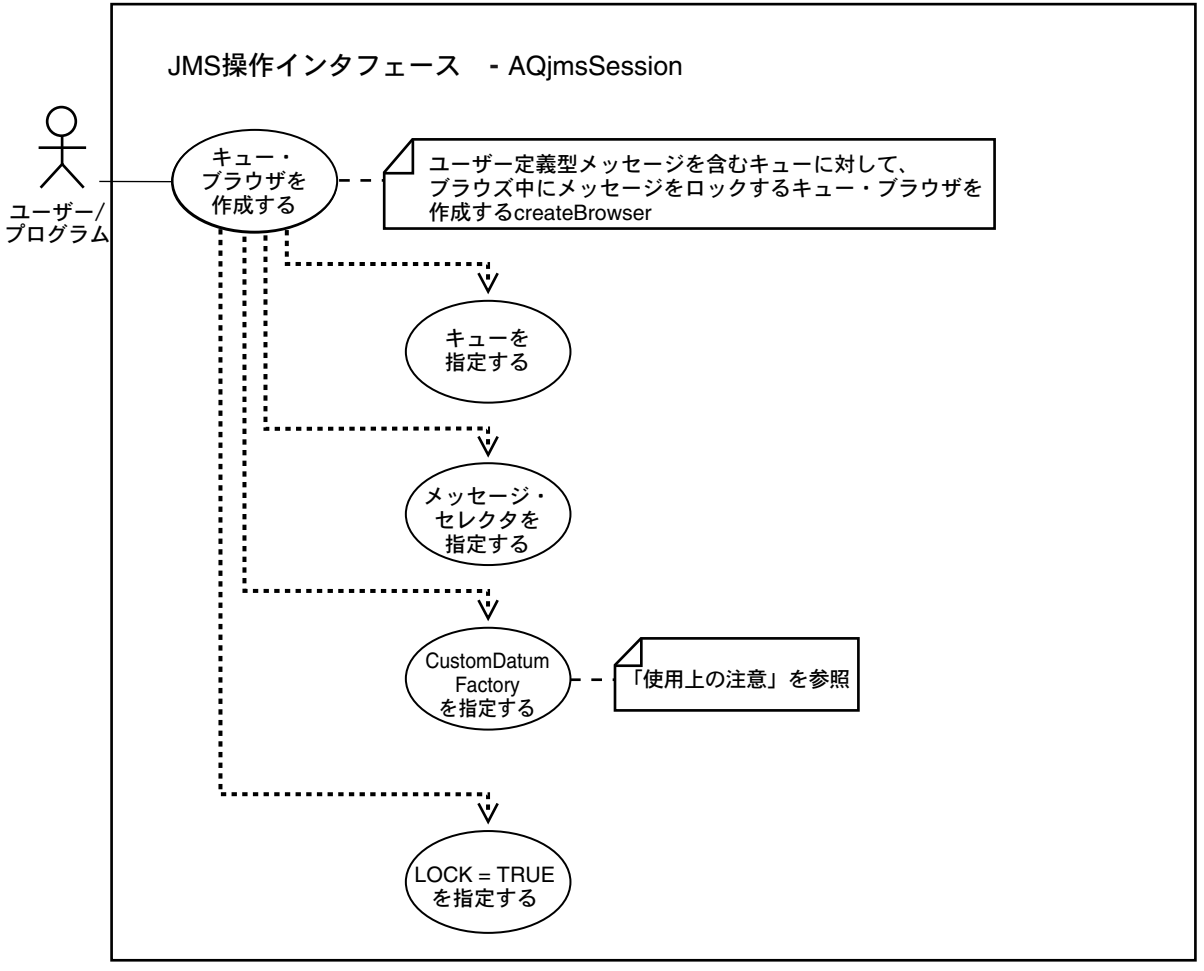
SQL ユーザー定義型ペイロードにマップする特定の Java クラスに対する CustomDatumFactory は、static メソッド getFactory を使用して取得できます。

キュー test_queue は SCOTT.EMPLOYEE 型のペイロードを持ち、このユーザー定義型に対して、Employee という Java クラスが JPublisher によって生成されたとします。Employee クラスは、CustomDatum インタフェースを実装します。このクラスに対する CustomDatumFactory は、Employee.getFactory() メソッドを使用して取得できます。

```
 /* Create a browser for a Queue with Adt messages of type EMPLOYEE*/
QueueSession jms_session
QueueBrowser browser;
Queue        test_queue;

browser = ((AQjmsSession)jms_session).createBrowser(test_queue,
                                     "corrid='EXPRESS'", Employee.getFactory());
```

Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成 : メッセージをロック

図 14-12 Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成 : メッセージをロック



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対して、ブラウズ中にメッセージをロックするキュー・ブラウザを作成します。

使用上の注意

ありません。

構文

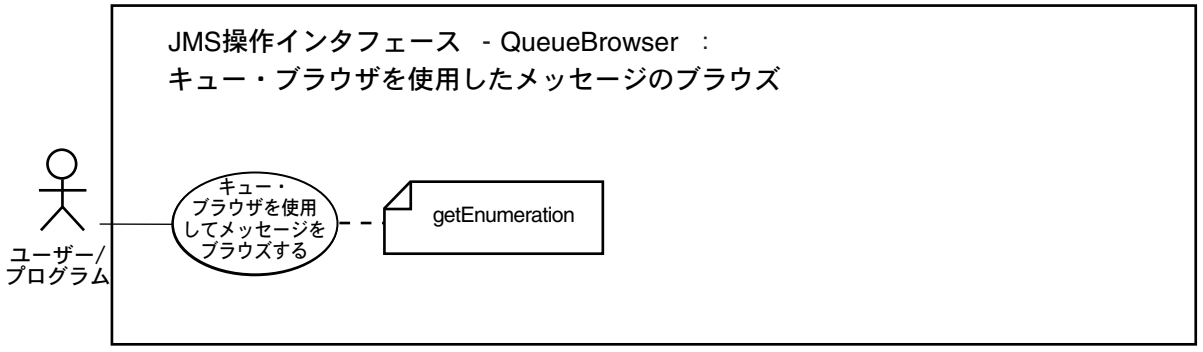
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createBrowser` を参照してください。

例

```
 /* Create a browser for a Queue with Adt messages of type EMPLOYEE* in lock mode/
QueueSession jms_session
QueueBrowser browser;
Queue        test_queue;

browser = ((AQjmsSession)jms_session).createBrowser(test_queue, null,
Employee.getFactory(), true);
```

キュー・ブラウザを使用したメッセージのブラウズ

図 14-13 キュー・ブラウザを使用したメッセージのブラウズ



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-32 ページの「インタフェース - javax.jms.QueueBrowser」も参照してください。

用途

キュー・ブラウザを使用して、メッセージをブラウズします。

使用上の注意

`java.util.Enumeration` 内のメソッドを使用して、メッセージのリストを参照してください。

構文

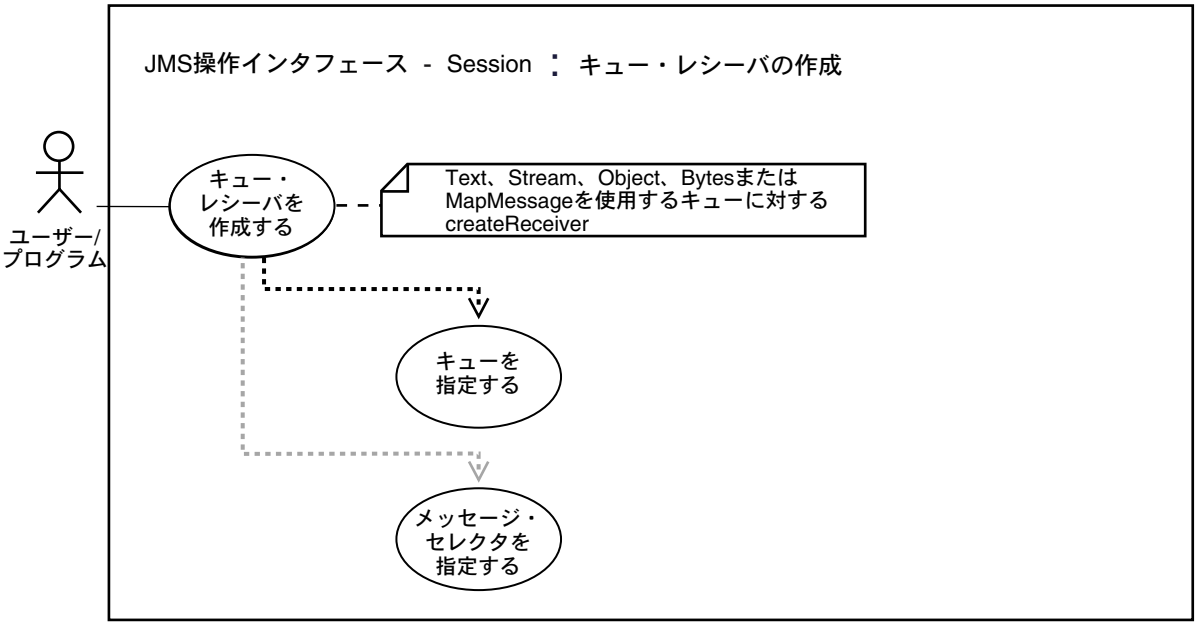
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueBrowser」を参照してください。

例

```
/* Create a browser for queues with a specified selector */
public void browse_rush_orders(QueueSession jms_session)
{
   QueueBrowser    browser;
   Queue           queue;
   ObjectMessage   obj_message
   BolOrder        new_order;
   Enumeration     messages;

   /* get a handle to the new_orders queue */
   queue = ((AQjmsSession) jms_session).getQueue("OE", "OE_neworders_que");

   /* create a Browser to look at RUSH orders */
   browser = jms_session.createBrowser(queue, "JMSCorrelationID = 'RUSH'");

   /* Browse through the messages */
   for (messages = browser.elements() ; message.hasMoreElements() ;)
   {
     obj_message = (ObjectMessage)message.nextElement();
   }
}
```

キュー・レシーバの作成 : 標準 JMS 型メッセージ・キュー

図 14-14 キュー・レシーバの作成 : 標準 JMS 型メッセージ・キュー



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

標準 JMS 型メッセージ・キューに対するキュー・レシーバを作成します。

使用上の注意

キュー・レシーバに対するセレクタは次の 1 つ以上の組合せによる式で指定します。

- JMS メッセージ ID

  ```
  JMSMessageID = 'ID:23452345'
  ```

  指定されたメッセージ ID を持つメッセージを取り出します。

- JMS メッセージ・ヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

すべてのメッセージ ID には、接頭辞 ID: が必要です。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createReceiver` を参照してください。

例

例 1

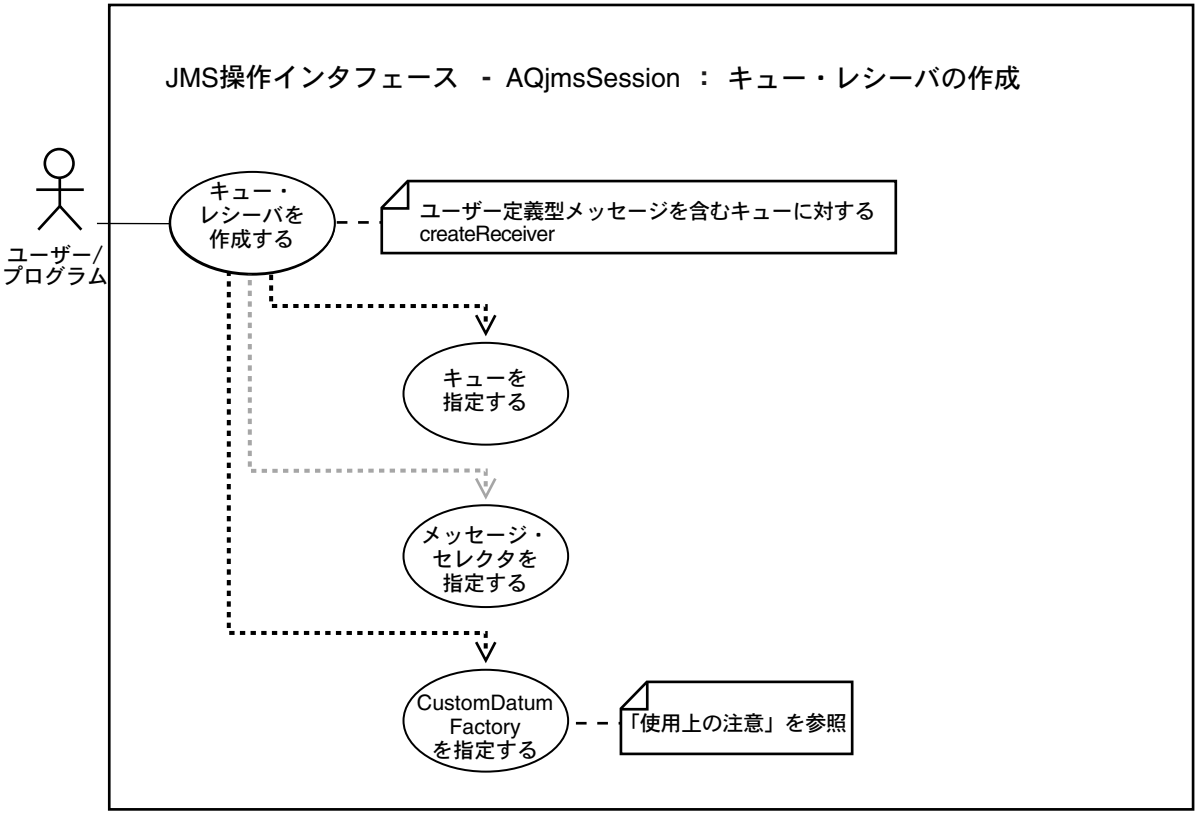
```
/* Create a receiver without a selector */
QueueSession    jms_session
QueueReceiver   receiver;
Queue           queue;

receiver = jms_session.createReceiver(queue);
```

例 2

```
/* Create a receiver for queues with a specified selector */
QueueSession    jms_session;
QueueReceiver   receiver;
Queue           queue;

/* create a Receiver to receive messages with correlationID starting with EXP  */
browser = jms_session.createReceiver(queue, "JMSCorrelationID LIKE 'EXP%'");
```

キュー・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・キュー

図 14-15 キュー・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・キュー



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・レシーバを作成します。

使用上の注意

SQL ユーザー定義型ペイロードにマップする特定の Java クラスに対する `CustomDatumFactory` は、static メソッド `getFactory` を使用して取得できます。

`AdtMessage` を含むキューの場合、キュー・レシーバに対するセレクタは、メッセージ・ペイロード内容、メッセージ ID、優先順位または相関識別子に対する SQL 式で指定します。

- メッセージ ID に対するセレクタは、次のように指定します。指定したメッセージ ID を持つメッセージが取り出されます。

  ```
  msgid = '23434556566767676'
  ```

  注意 : この場合は、メッセージ ID に接頭辞 ID: を使用しないでください。

- 優先順位または相関識別子に対するセレクタは、次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードに対するセレクタは、次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createReceiver` を参照してください。

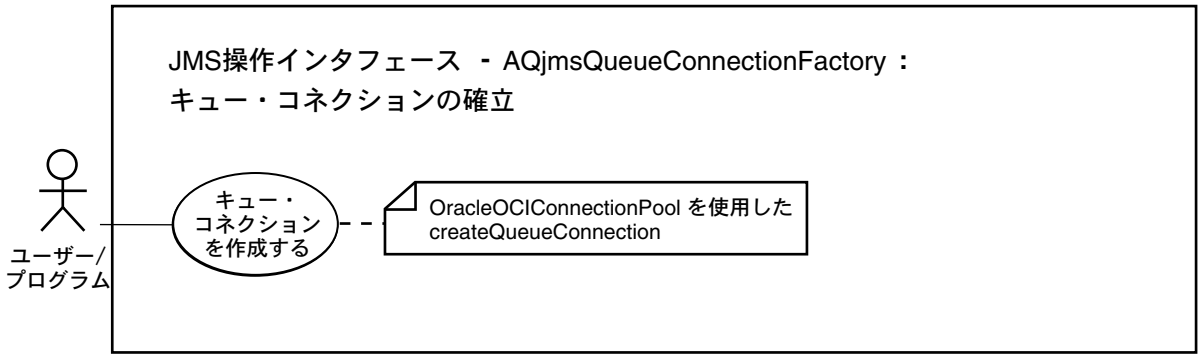
例

キュー test_queue に SCOTT.EMPLOYEE 型のペイロードが含まれ、このユーザー定義型に対して、Employee という Java クラスが JPublisher によって生成されたとします。Employee クラスは、CustomDatum インタフェースを実装します。このクラスに対する CustomDatumFactory は、Employee.getFactory() メソッドを使用して取得できます。

```
/* Create a receiver for a Queue with Adt messages of type EMPLOYEE*/
QueueSession jms_session
QueueReceiver receiver;
Queue        test_queue;

browser = ((AQjmsSession)jms_session).createReceiver(test_queue, "JMSCorrelationID =
'MANAGER', Employee.getFactory());
```

キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用

図 14-16 キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用



参照 :

- JMS 操作インタフェースの基本操作の詳細は、表 14-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsQueueConnectionFactory」も参照してください。

用途

オープンしている `OracleOCIConnectionPool` を使用してキュー・コネクションを確立します。

使用上の注意

これは static メソッドです。

構文

Java（JDBC）:『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueConnectionFactory」の `createQueueConnection` を参照してください。

例

ユーザーが JMS 操作に対して既存の OracleOCIConnectionPool インスタンスを使用するときに、このメソッドが使用されます。この場合、JMS は新しい OracleOCIConnectionPool インスタンスをオープンするのではなく、提供された OracleOCIConnectionPool インスタンスを使用して、JMS キュー・コネクション・オブジェクトを作成します。

```
OracleOCIConnectionPool cpool; /* previously created OracleOCIConnectionPool */
QueueConnection qc_conn = AQjmsQueueConnectionFactory.createQueueConnection(cpool);
Connection db_conn;     /* previously opened JDBC connection */
 QueueConnection qc_conn = AQjmsQueueConnectionFactory.createQueueConnection(db_
conn);
```

15

JMS 操作インタフェース : 基本操作（パブリッシュ・サブスクライブ）

この章では、Oracle Advanced Queuing の操作インタフェース（パブリッシュ・サブスクライブ）を利用方法に沿って説明します。それぞれの操作（メッセージのパブリッシュなど）を、その操作名ごとに利用方法に沿って説明します。この章の先頭に、すべての利用方法を示します（15-2 ページの「利用モデル : JMS 操作インタフェース - 基本操作（パブリッシュ・サブスクライブ）」を参照）。

図 15-1 に、すべての利用方法を 1 つの図にまとめています。HTML 版のマニュアルをご使用の場合、この図の中の関連する利用方法のタイトルをクリックすることで、関心のある利用方法に移動することができます。

個々の利用方法は、次の形式で説明されています。

- **利用図 :** 利用方法を表す図
- **用途 :** この利用方法の用途
- **使用上の注意 :** 実装に有効なガイドライン
- **構文 :** このアクティビティの実行に使用する主な構文
- **例 :** 各プログラム環境での利用例

利用モデル : JMS 操作インタフェース - 基本操作（パブリッシュ・サブスクライブ）

表 15-1 利用モデル : JMS 操作インタフェース - 基本操作（パブリッシュ・サブスクライブ）

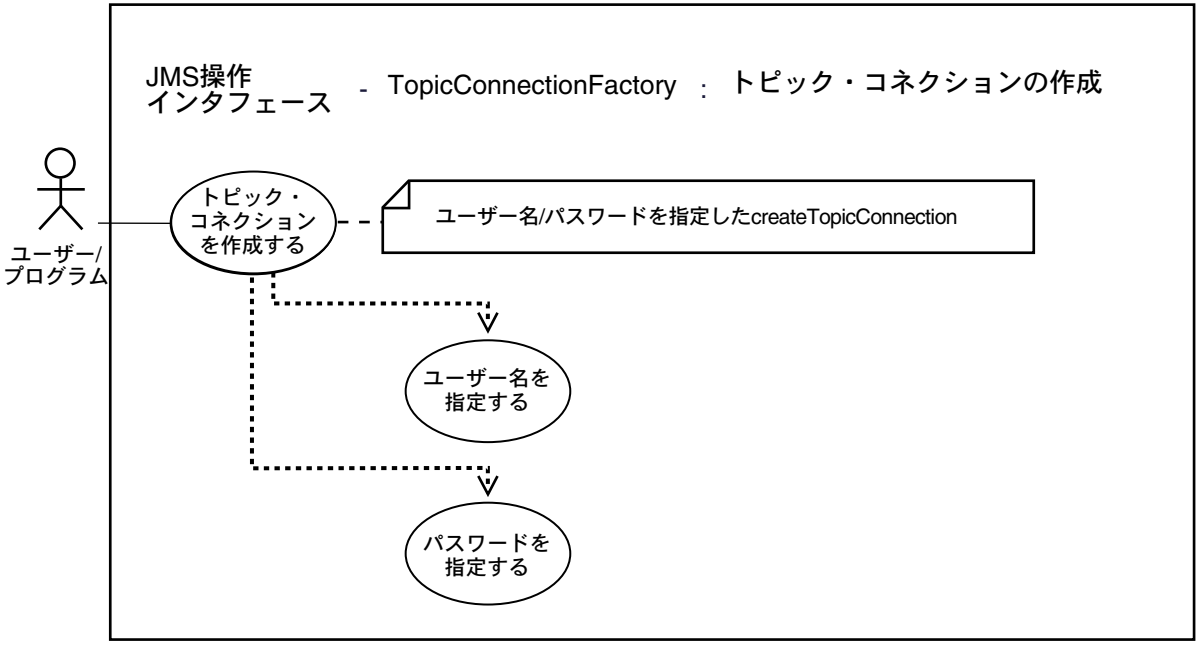
利用方法
トピック・コネクションの確立 : ユーザー名 / パスワードの使用（15-4 ページ）
トピック・コネクションの確立 : オープンしている JDBC コネクションの使用（15-6 ページ）
トピック・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用（15-8 ページ）
トピック・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用（15-10 ページ）
トピック・セッションの作成（15-12 ページ）
トピック・パブリッシャの作成（15-14 ページ）
トピック・パブリッシャを使用したメッセージのパブリッシュ : 最小限の指定（15-15 ページ）
トピック・パブリッシャを使用したメッセージのパブリッシュ : 相関および遅延を指定（15-18 ページ）
トピック・パブリッシャを使用したメッセージのパブリッシュ : 優先順位および Time-To-Live を指定（15-21 ページ）
トピック・パブリッシャを使用したメッセージのパブリッシュ : トピック・サブスクライバをオーバーライドする受信者リストを指定（15-24 ページ）
JMS トピックに対する永続サブスクライバの作成 : セレクタの指定なし（15-27 ページ）
JMS トピックに対する永続サブスクライバの作成 : セレクタの指定あり（15-29 ページ）
ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定なし（15-32 ページ）
ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定あり（15-34 ページ）
リモート・サブスクライバの作成 : JMS 型メッセージ・トピック（15-37 ページ）
リモート・サブスクライバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック（15-40 ページ）
永続サブスクリプションのサブスクライブの解除 : ローカル・サブスクライバ（15-43 ページ）
永続サブスクリプションのサブスクライブの解除 : リモート・サブスクライバ（15-45 ページ）
トピック・レシーバの作成 : 標準 JMS 型メッセージ・トピック（15-47 ページ）
トピック・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック（15-49 ページ）
Text、Stream、Object、Bytes、MapMessage を使用するトピックに対するトピック・ブラウザの作成（15-52 ページ）
Text、Stream、Object、Bytes、MapMessage を使用するトピックに対するトピック・ブラウザの作成 : メッセージをロック（15-54 ページ）

表 15-1 利用モデル : JMS 操作インタフェース - 基本操作（パブリッシュ・サブスクライブ）（続き）

利用方法
Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザの作成（15-56 ページ）
Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザの作成 : メッセージをロック（15-59 ページ）
トピック・ブラウザを使用したメッセージのブラウズ（15-61 ページ）

トピック・コネクションの確立 : ユーザー名 / パスワードの使用

図 15-1 トピック・コネクションの確立 : ユーザー名 / パスワードの使用（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-39 ページの「インタフェース - javax.jms.TopicConnectionFactory」も参照してください。
- 15-6 ページの「トピック・コネクションの確立 : オープンしている JDBC コネクションの使用」も参照してください。
- 15-8 ページの「トピック・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用」も参照してください。
- 15-10 ページの「トピック・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用」も参照してください。

用途

ユーザー名 / パスワードを使用して、トピック・コネクションを確立します。

使用上の注意

ありません。

構文

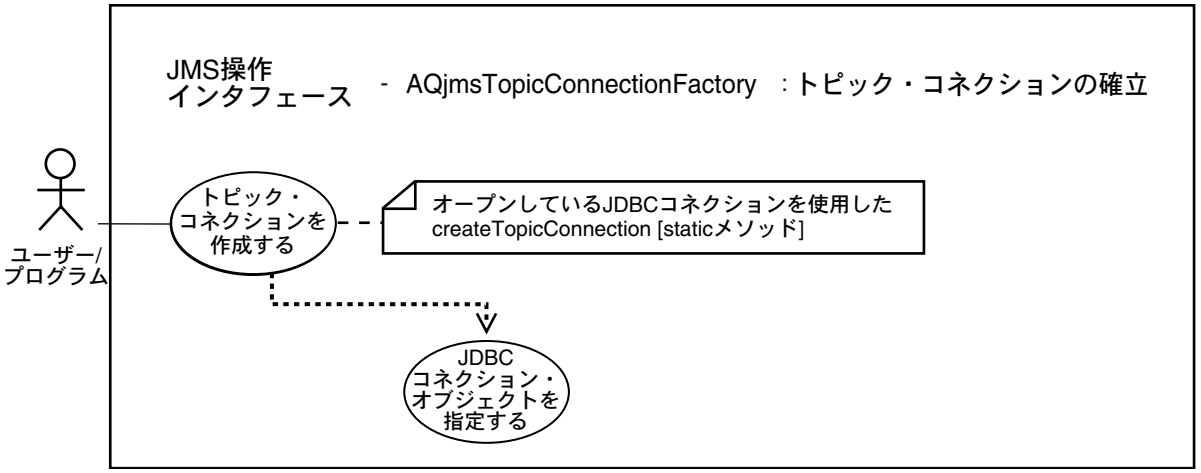
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicConnectionFactory」の `createTopicConnection` を参照してください。

例

```
 TopicConnectionFactory tc_fact = AQjmsFactory.getTopicConnectionFactory("sun123",
"oratest", 5521, "thin");
/* Create a topic connection using a username/password */
TopicConnection tc_conn = tc_fact.createTopicConnection("jmsuser", "jmsuser");
```

トピック・コネクションの確立 : オープンしている JDBC コネクションの使用

図 15-2 トピック・コネクションの確立 : オープンしている JDBC コネクションの使用（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-57 ページの「クラス - oracle.jms.AQjmsTopicConnectionFactory」も参照してください。
- 15-4 ページの「トピック・コネクションの確立 : ユーザー名 / パスワードの使用」も参照してください。
- 15-8 ページの「トピック・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用」も参照してください。
- 15-10 ページの「トピック・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用」も参照してください。

用途

オープンしている JDBC コネクションを使用して、トピック・コネクションを作成します。

使用上の注意

ありません。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicConnectionFactory」の `createTopicConnection` を参照してください。

例

例 1

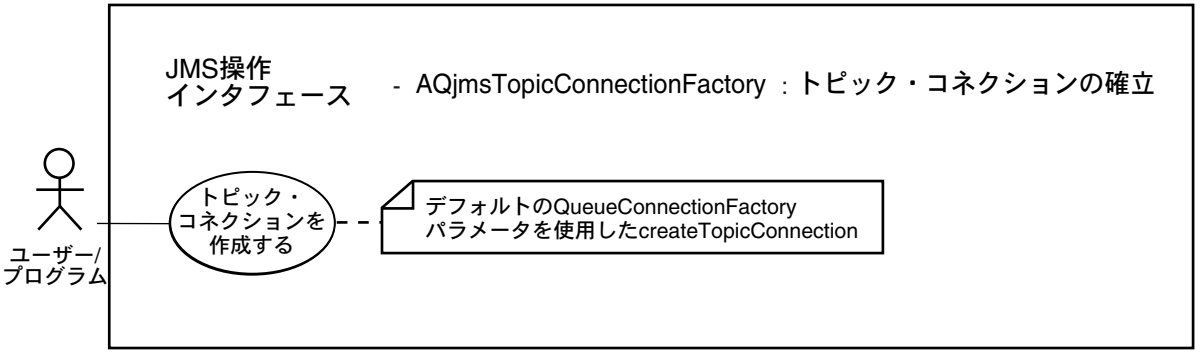
```
Connection db_conn;   /*previously opened JDBC connection */
TopicConnection tc_conn = AQjmsTopicConnectionFactory.createTopicConnection(db_
conn);
```

例 2

```
OracleDriver ora = new OracleDriver();
TopicConnection tc_conn =
AQjmsTopicConnectionFactory.createTopicConnection(ora.defaultConnection());
```

トピック・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用

図 15-3 トピック・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-57 ページの「クラス - oracle.jms.AQjmsTopicConnectionFactory」も参照してください。
- 15-4 ページの「トピック・コネクションの確立 : ユーザー名 / パスワードの使用」も参照してください。
- 15-6 ページの「トピック・コネクションの確立 : オープンしている JDBC コネクションの使用」も参照してください。
- 15-10 ページの「トピック・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用」も参照してください。

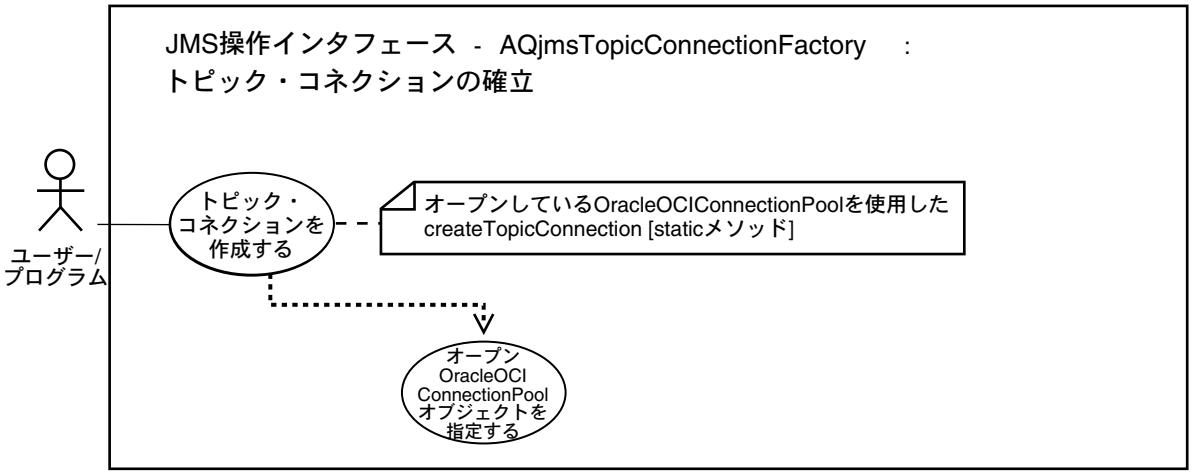
用途
デフォルトのコネクション・ファクトリ・パラメータを使用して、トピック・コネクションを確立します。

使用上の注意
ありません。

構文
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicConnectionFactory」の `createTopicConnection` を参照してください。

トピック・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用

図 15-4 トピック・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-57 ページの「クラス - oracle.jms.AQjmsTopicConnectionFactory」も参照してください。
- 15-4 ページの「トピック・コネクションの確立 : ユーザー名 / パスワードの使用」も参照してください。
- 15-6 ページの「トピック・コネクションの確立 : オープンしている JDBC コネクションの使用」も参照してください。
- 15-8 ページの「トピック・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用」も参照してください。

用途

オープンしている OracleOCIConnectionPool を使用して、トピック・コネクションを確立します。

使用上の注意

これは static メソッドです。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicConnectionFactory」の createTopicConnection を参照してください。

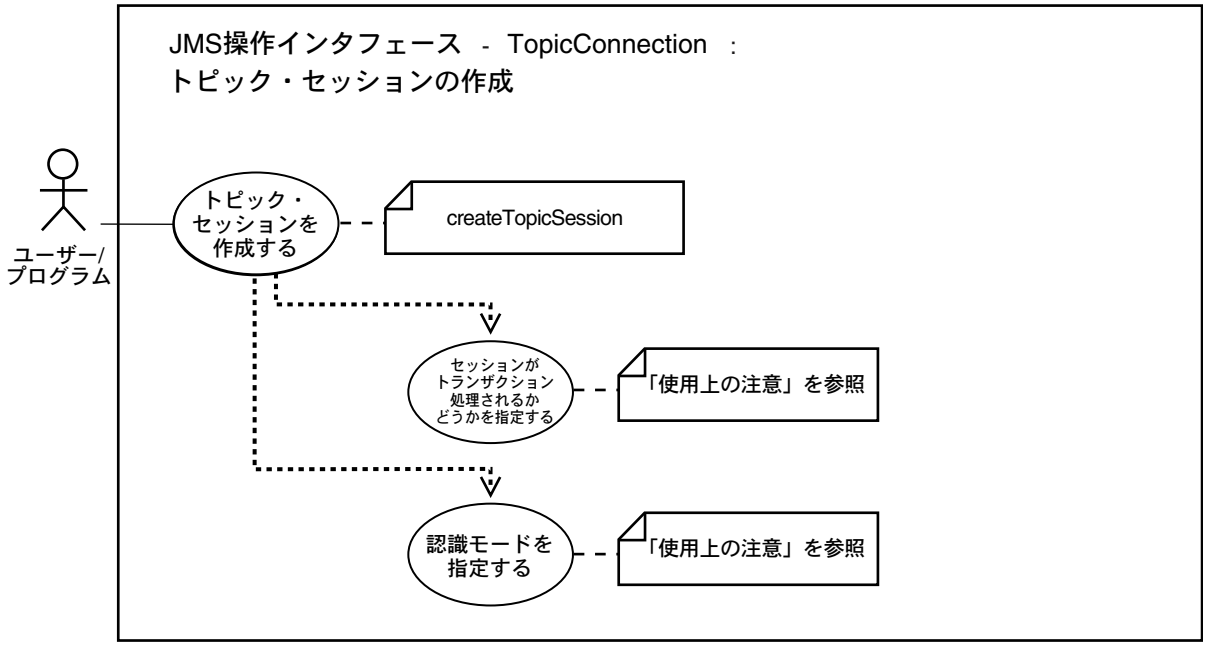
例

ユーザーが JMS 操作に対して既存の OracleOCIConnectionPool インスタンスを使用するときに、このメソッドが使用されます。この場合、JMS は新しい OracleOCIConnectionPool インスタンスをオープンするのではなく、提供された OracleOCIConnectionPool インスタンスを使用して、JMS トピック・コネクション・オブジェクトを作成します。

```
OracleOCIConnectionPool cpool; /* previously created OracleOCIConnectionPool */
TopicConnection tc_conn = AQjmsTopicConnectionFactory.createTopicConnection(cpool);
```

トピック・セッションの作成

図 15-5 トピック・セッションの作成（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-38 ページの「インタフェース - javax.jms.TopicConnection」も参照してください。

用途

トピック・セッションを作成します。

使用上の注意

ありません。

構文

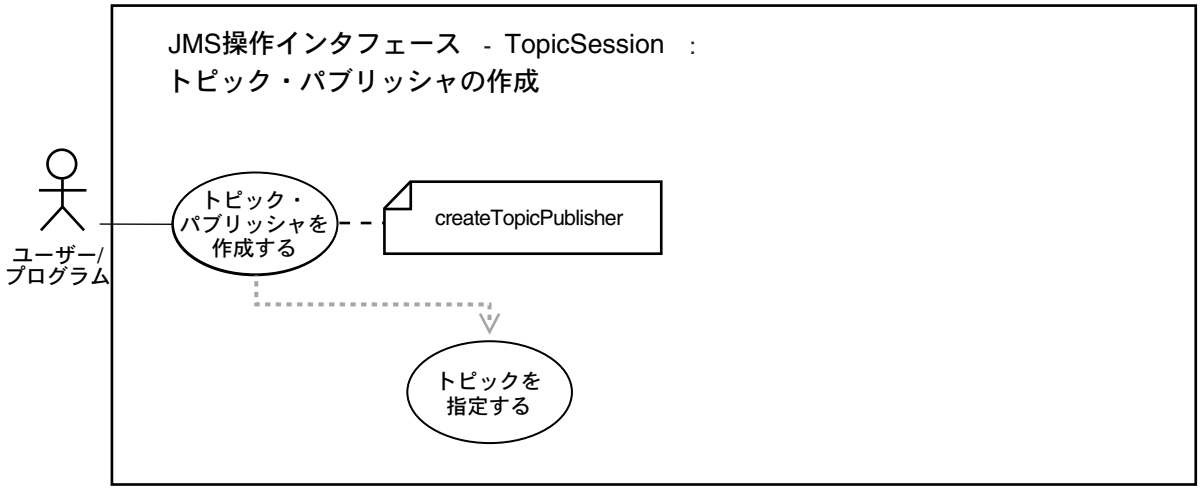
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `createTopicSession` を参照してください。

例

```
TopicConnection tc_conn;
TopicSession t_sess = tc_conn.createTopicSession(true,0);
```

トピック・パブリッシャの作成

図 15-6 トピック・パブリッシャの作成（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-40 ページの「インタフェース - javax.jms.TopicSession」も参照してください。

用途

トピック・パブリッシャを作成します。

使用上の注意

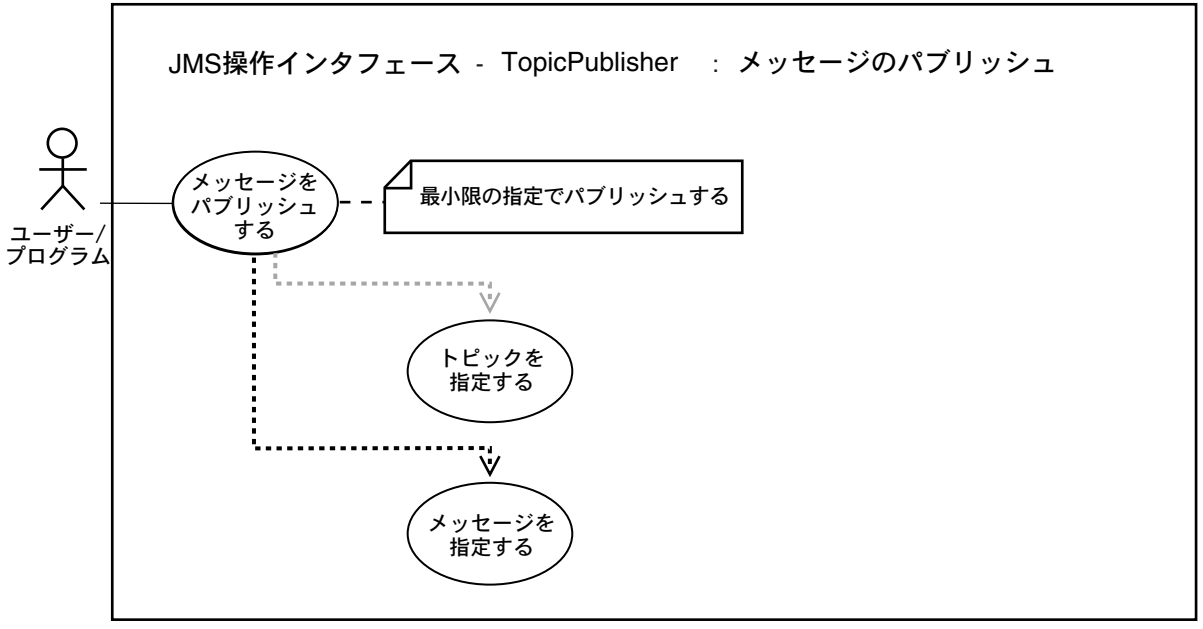
ありません。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createPublisher` を参照してください。

トピック・パブリッシャを使用したメッセージのパブリッシュ : 最小限の指定

図 15-7 最小限の指定でのメッセージのパブリッシュ（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-39 ページの「インタフェース - javax.jms.TopicPublisher」も参照してください。
- 15-18 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 相関および遅延を指定」も参照してください。
- 15-21 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 優先順位および Time-To-Live を指定」も参照してください。
- 15-24 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : トピック・サブスクライバをオーバーライドする受信者リストを指定」も参照してください。

用途
最小限の指定でメッセージをパブリッシュします。

使用上の注意
トピック・パブリッシャがデフォルト・トピックで作成されている場合、必ずしも publish コールにトピック・パラメータを指定する必要はありません。トピックが送信操作で指定されている場合、その値はトピック・パブリッシャのデフォルト・トピックをオーバーライドします。トピック・パブリッシャがデフォルト・トピックを使用しないで作成されている場合、publish コールごとにトピックを指定する必要があります。トピック・パブリッシャでは、メッセージの優先順位（1）および Time-To-Live（無制限）のデフォルト値が使用されます。

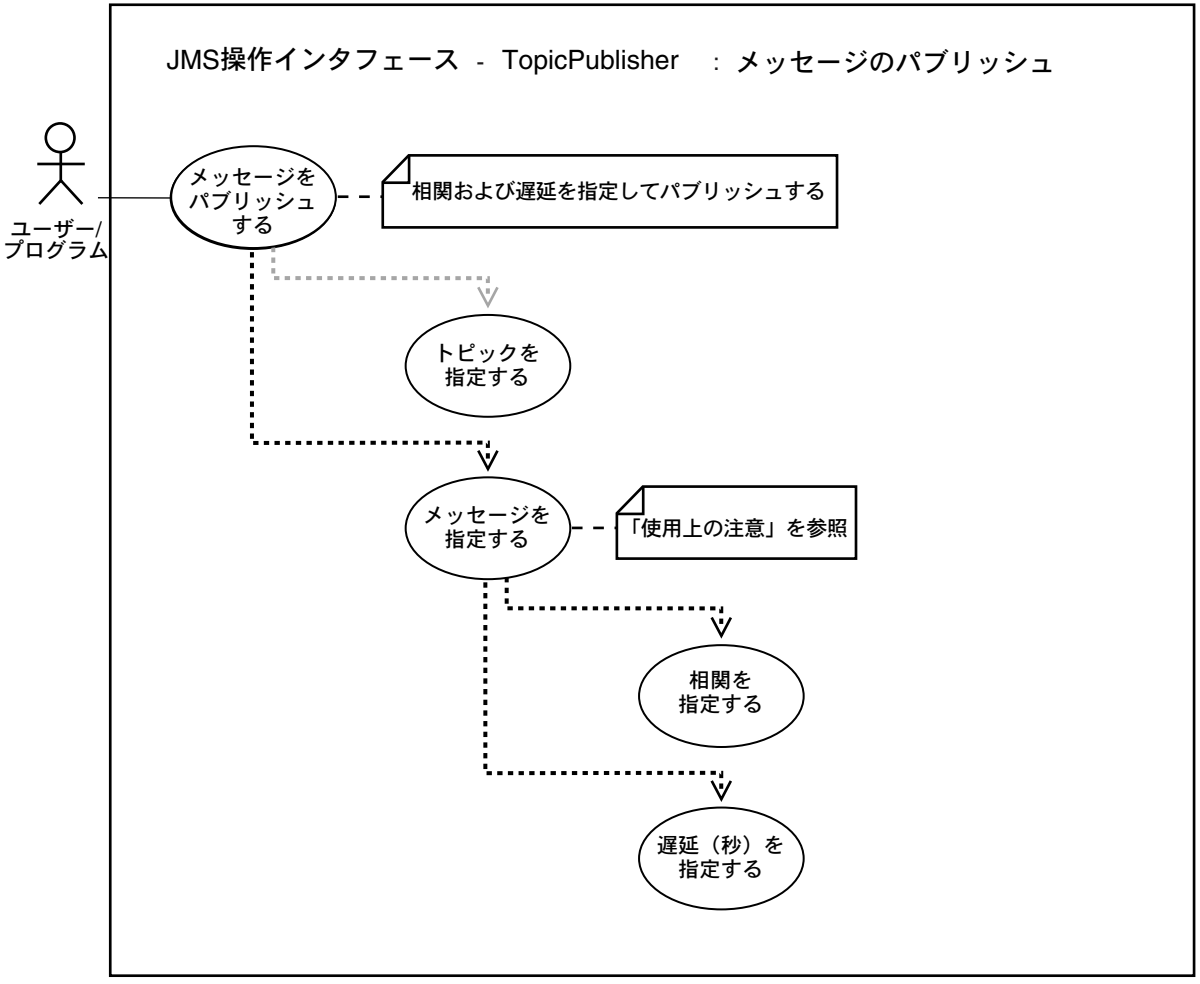
構文
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicPublisher」の publish を参照してください。

例
```
Example 1 - publish specifying topic

TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicPublisher            publisher1;
Topic                     shipped_orders;
int                       myport = 5521;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory('MYHOSTNAME',
                                                 'MYSID', myport, 'oci8');
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

/* create topic publisher */
publisher1 = jms_sess.createPublisher(null);

/* get topic object */
shipped_orders = ((AQjmsSession )jms_sess).getTopic('WS', 'Shipped_Orders_Topic');

/* create text message */
TextMessage     jms_sess.createTextMessage();

/* publish specifying the topic */
publisher1.publish(shipped_orders, text_message);
```

```
Example 2 - publish without specifying topic

TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicPublisher            publisher1;
Topic                     shipped_orders;
int                       myport = 5521;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");

/* create topic session */
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

/* get shipped orders topic */
shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");
publisher1 = jms_sess.createPublisher(shipped_orders);

/* create text message */
TextMessage     jms_sess.createTextMessage();

/* publish without specifying the topic */
publisher1.publish(text_message);
```

トピック・パブリッシャを使用したメッセージのパブリッシュ : 相関および遅延を指定

図 15-8 相関および遅延を指定したメッセージのパブリッシュ（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-39 ページの「インタフェース - javax.jms.TopicPublisher」も参照してください。
- 15-15 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 最小限の指定」も参照してください。
- 15-21 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 優先順位および Time-To-Live を指定」も参照してください。
- 15-24 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : トピック・サブスクライバをオーバーライドする受信者リストを指定」も参照してください。

用途

相関および遅延を指定して、メッセージをパブリッシュします。

使用上の注意

パブリッシャは、パブリッシュ前に遅延や相関などのメッセージ・プロパティを設定できます。

構文

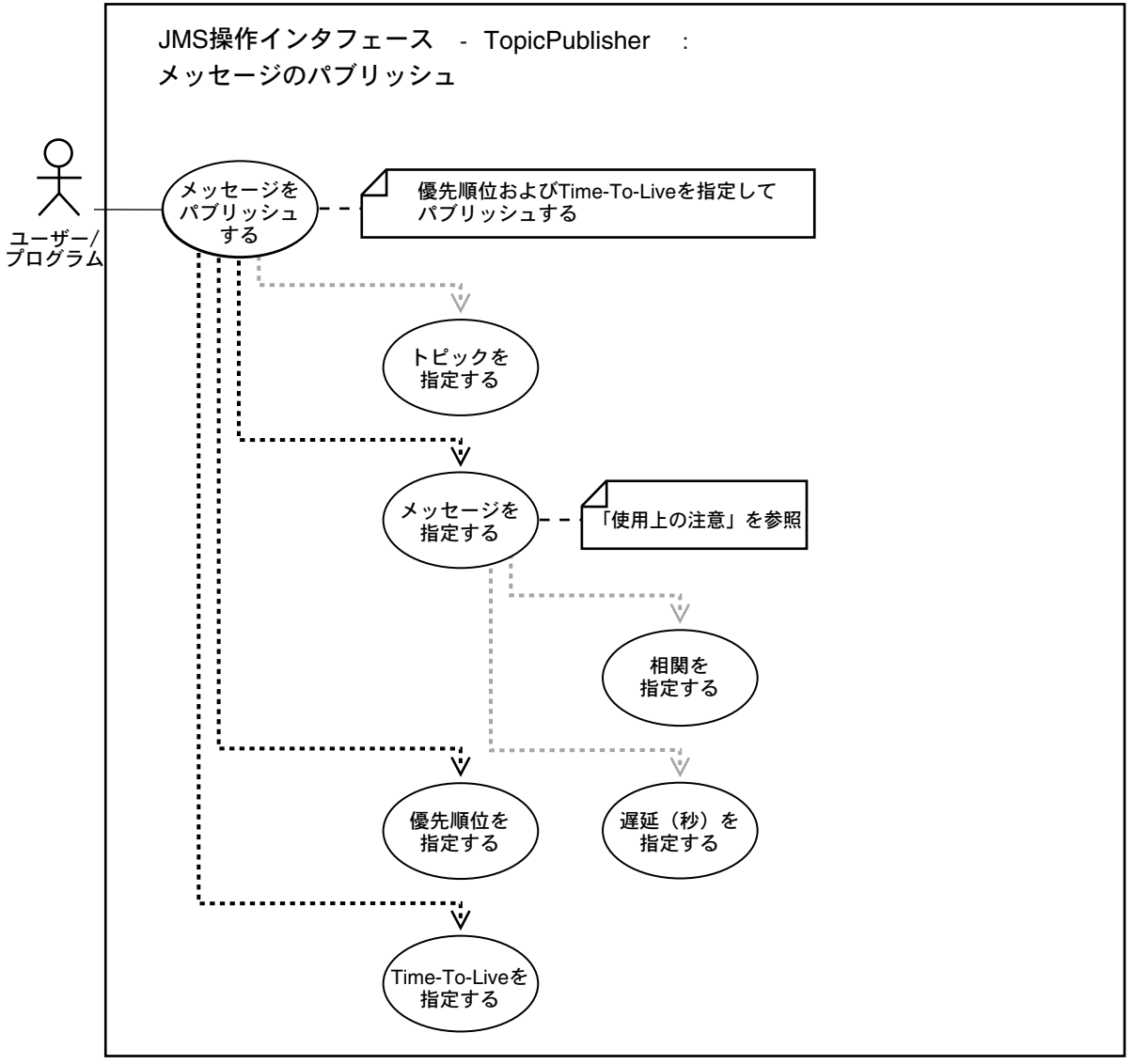
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicPublisher」の `publish()` を参照してください。

例

```
 Example 1 - publish specifying delay, correlation

TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicPublisher            publisher1;
Topic                     shipped_orders;
int                       myport = 5521;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");
publisher1 = jms_sess.createPublisher(shipped_orders);

/* create text message */
TextMessage     jms_sess.createTextMessage();

/* Set correlation and delay */

/* set correlation */
jms_sess.setJMSCorrelationID("FOO");

/* set delay of 30 seconds */
jms_sess.setLongProperty("JMS_OracleDelay", 30);

/* publish  */
publisher1.publish(text_message);
```

トピック・パブリッシャを使用したメッセージのパブリッシュ : 優先順位および Time-To-Live を指定

図 15-9 優先順位および Time-To-Live を指定したメッセージのパブリッシュ（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-39 ページの「インタフェース - javax.jms.TopicPublisher」も参照してください。
- 15-15 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 最小限の指定」も参照してください。
- 15-18 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 相関および遅延を指定」も参照してください。
- 15-24 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : トピック・サブスクライバをオーバーライドする受信者リストを指定」も参照してください。

用途

優先順位および Time-To-Live を指定して、メッセージをパブリッシュします。

使用上の注意

メッセージの優先順位および Time-To-Live は、`publish` コールで指定できます。このリリースでは、配信モード PERSISTENT のみがサポートされています。

構文

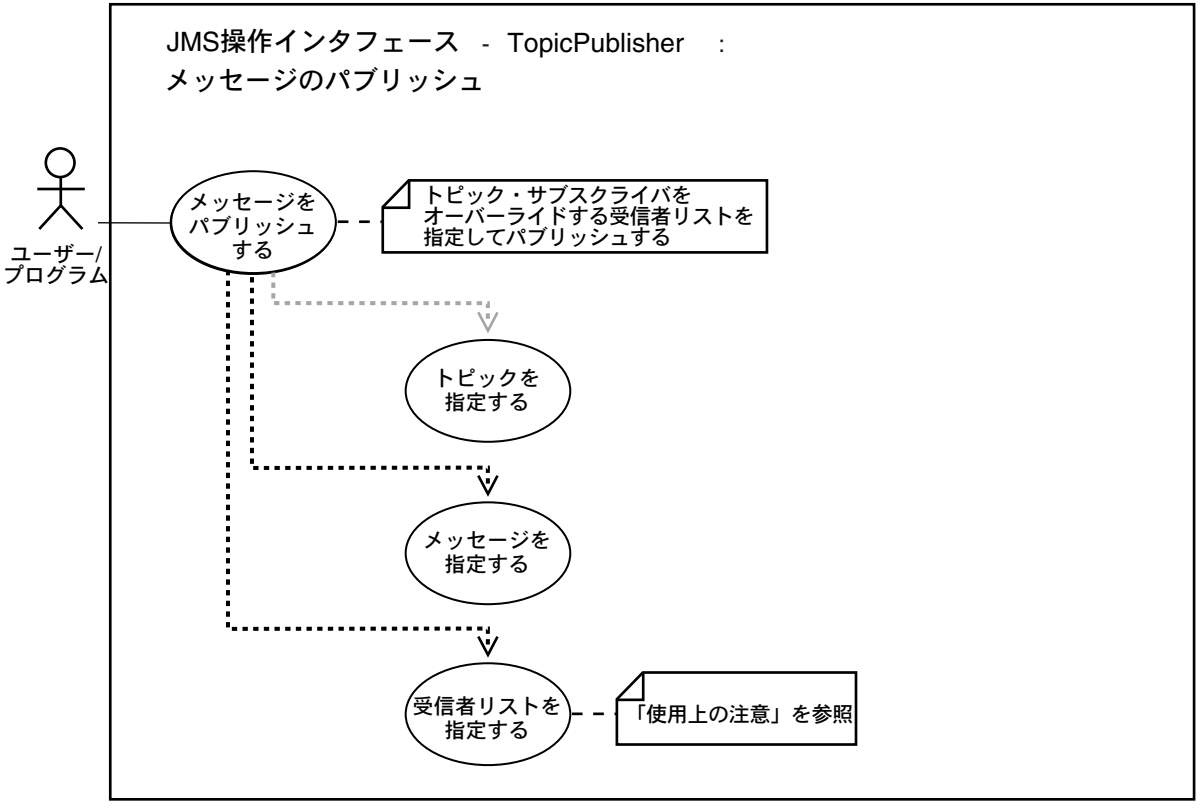
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicPublisher」の `publish` を参照してください。

例

```
 Example 1 - publish specifying priority, timeToLive

TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicPublisher            publisher1;
Topic                     shipped_orders;
int                       myport = 5521;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");
publisher1 = jms_sess.createPublisher(shipped_orders);

/* create text message */
TextMessage     jms_sess.createTextMessage();

/* publish  message with priority 1 and time to live 200 seconds */
publisher1.publish(text_message, DeliveryMode.PERSISTENT, 1, 200000);
```

トピック・パブリッシャを使用したメッセージのパブリッシュ : トピック・サブスクライバをオーバーライドする受信者リストを指定

図 15-10 トピック・サブスクライバをオーバーライドする受信者リストを指定したメッセージのパブリッシュ（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-39 ページの「インタフェース - javax.jms.TopicPublisher」も参照してください。
- 15-15 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 最小限の指定」も参照してください。
- 15-18 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 相関および遅延を指定」も参照してください。
- 15-21 ページの「トピック・パブリッシャを使用したメッセージのパブリッシュ : 優先順位および Time-To-Live を指定」も参照してください。

用途

トピック・サブスクライバをオーバーライドする受信者リストを指定して、メッセージをパブリッシュします。

使用上の注意

トピックのサブスクリプション・リストは、publish コールで受信者リストを指定することによって、オーバーライドできます。

構文

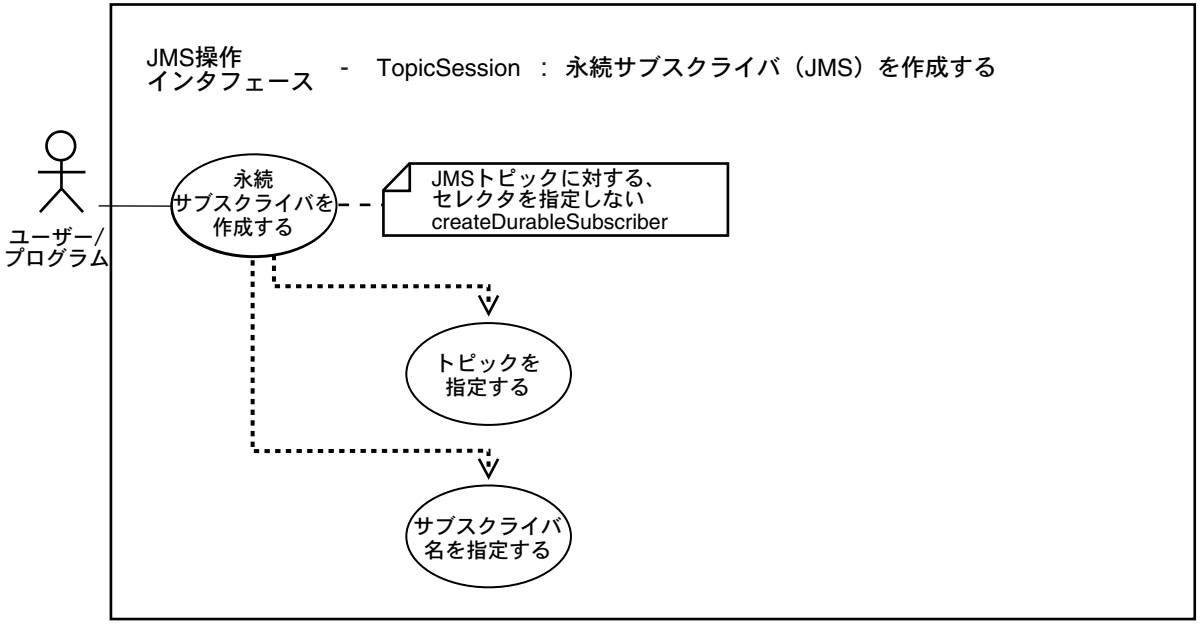
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicPublisher」の publish を参照してください。

例

```
 Example 1 - publish specifying priority, timeToLive

TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicPublisher            publisher1;
Topic                     shipped_orders;
int                       myport = 5521;
AQjmsAgent[]              recipList;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");
publisher1 = jms_sess.createPublisher(shipped_orders);

/* create text message */
TextMessage     jms_sess.createTextMessage();

/* create two receivers */
recipList = new AQjmsAgent[2];

recipList[0] = new AQjmsAgent("ES", "ES.shipped_orders_topic",
                           AQAgent.DEFAULT_AGENT_PROTOCOL);

recipList[1] = new AQjmsAgent("WS", "WS.shipped_orders_topic",
                           AQAgent.DEFAULT_AGENT_PROTOCOL);

/* publish  message specifying a recipient list */
publisher1.publish(text_message, recipList);
```

JMS トピックに対する永続サブスクライバの作成 : セレクタの指定なし

図 15-11 JMS トピックに対する永続サブスクライバの作成 : セレクタの指定なし（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-40 ページの「インタフェース - javax.jms.TopicSession」も参照してください。
- 15-29 ページの「JMS トピックに対する永続サブスクライバの作成 : セレクタの指定あり」も参照してください。

用途

セレクタを指定せずに、JMS トピックに対する永続サブスクライバを作成します。

使用上の注意

永続サブスクライバを作成するには、サブスクライバ名および JMS トピックを指定する必要があります。トピックのサブスクリプションを終了するには、unsubscribe コールが必要です。

構文

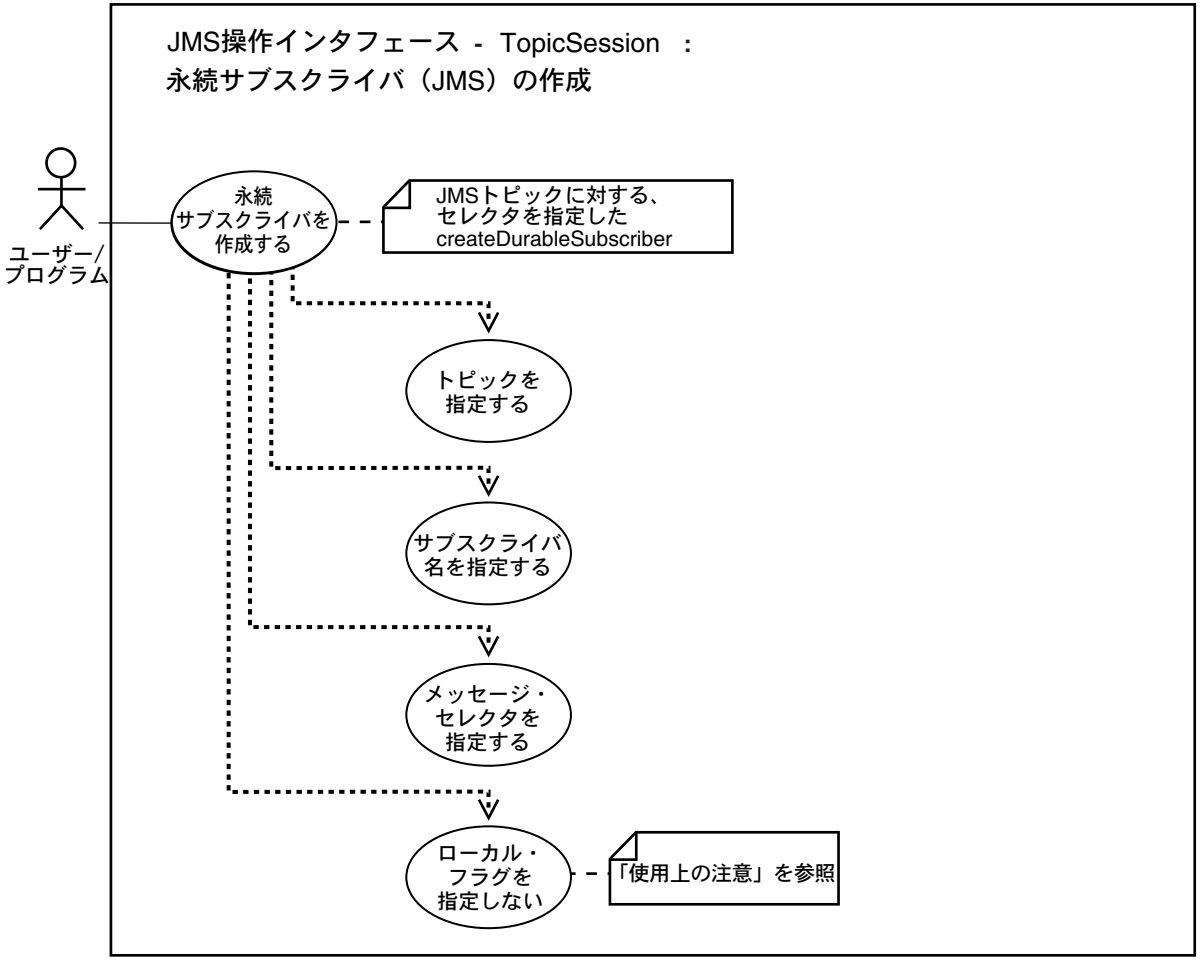
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の CreateDurableSubscriber を参照してください。

例

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicSubscriber           subscriber1;
Topic                     shipped_orders;
int                       myport = 5521;
AQjmsAgent[]              recipList;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");
/* create a durable subscriber on the shipped_orders topic*/
subscriber1 = jms_sess.createDurableSubscriber(shipped_orders, 'WesternShipping');
```

JMS トピックに対する永続サブスクライバの作成 : セレクタの指定あり

図 15-12 JMS トピックに対する永続サブスクライバの作成 : セレクタの指定あり（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-40 ページの「インタフェース - javax.jms.TopicSession」も参照してください。
- 15-27 ページの「JMS トピックに対する永続サブスクライバの作成 : セレクタの指定なし」も参照してください。

用途

セレクタを指定して、JMS トピックに対する永続サブスクライバを作成します。

使用上の注意

クライアントは、サブスクライバ名および JMS トピックを指定することによって、永続サブスクライバを作成します。オプションで、メッセージ・セレクタを指定できます。メッセージの選択指定式と一致したプロパティを持つメッセージのみ、サブスクライバに配信されます。セレクタ値は NULL である場合があります。セレクタには、次のうち 1 つ以上が組み合された SQL92 式を含めることができます。

- JMS メッセージ・ヘッダー・フィールドまたはプロパティ : JMSPriority（Int）、JMSCorrelationID（String）、JMSType（String）、JMSXUserID（String）、JMSXAppID（String）、JMSXGroupID（String）、JMSXGroupSeq（Int）

  次に、使用例を示します。

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  ```

- ユーザー定義のメッセージ・プロパティ

  次に、使用例を示します。

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

使用可能な演算子は次のとおりです。

- 優先順位 NOT、AND、OR の論理演算子
- 比較演算子 =、>、>=、<、<=、<>、!（<> および ! は、不等号として使用できます。）
- 優先順位 +、- unary、*、/、+、- の算術演算子
- 識別子 [NOT] IN (文字列リテラル 1, 文字列リテラル 2, ..)
- 算術式 1 [NOT] BETWEEN 算術式 2 and 算術式 3
- 識別子 [NOT] LIKE パターン値 [ESCAPE エスケープ文字]
- パターン値は文字列リテラルで、% は連続するすべての文字を表します。

- _ はすべての単一文字を表します。
- オプションのエスケープ文字は、パターン値内の「_」および「%」を文字列として処理する場合に使用します。
- 識別子 IS [NOT] NULL

クライアントは、同一の名前および異なるメッセージ・セレクタで永続トピック・サブスクライバを作成することによって、既存の永続サブスクリプションを変更できます。トピックのサブスクリプションを終了するには、unsubscribe コールが必要です。

構文

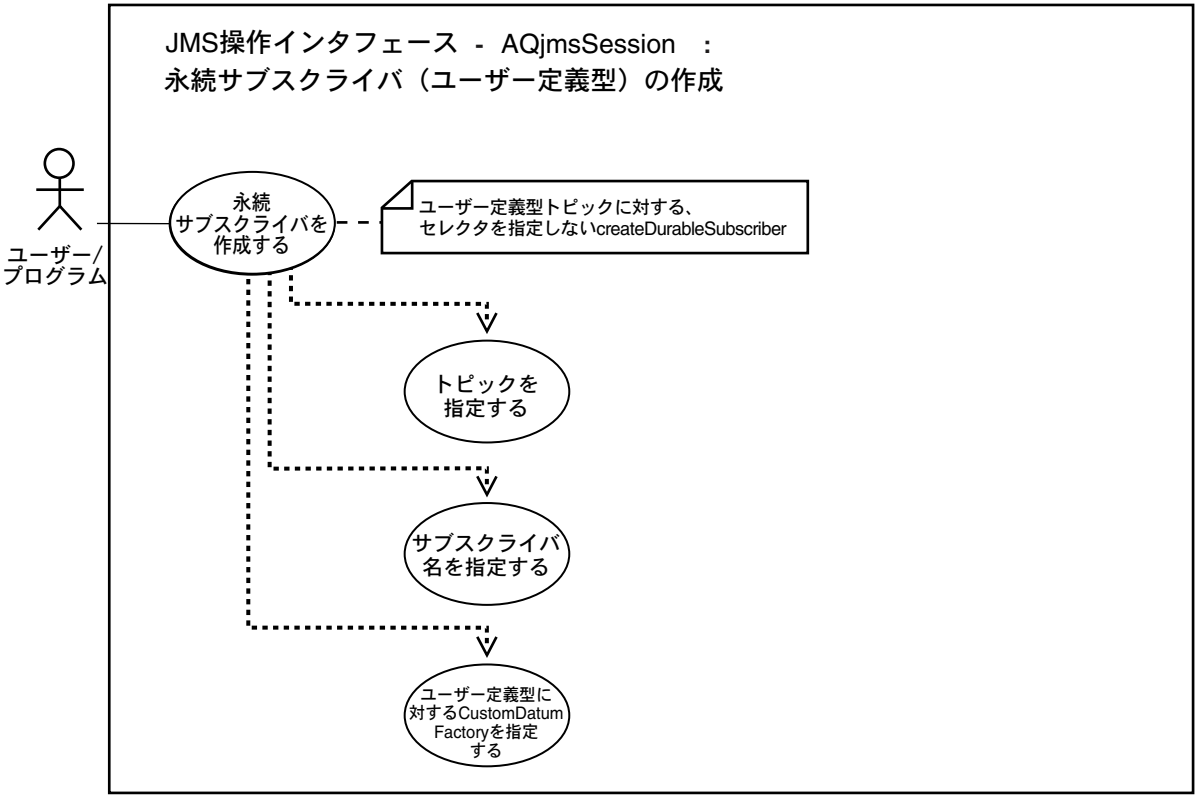
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsTopicPublisher」の publish を参照してください。

例

```
Example 1 - subscribe specifying selector

TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicSubscriber           subscriber1;
Topic                     shipped_orders;
int                       myport = 5521;
AQjmsAgent[]              recipList;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");

/* create a subscriber */
/*  with condition on JMSPriority and user property 'Region' */
subscriber1 = jms_sess.createDurableSubscriber(shipped_orders, 'WesternShipping',
                               "JMSPriority > 2 and Region like 'Western%'",
false);
```

ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定なし

図 15-13 ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定なし（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-34 ページの「ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定あり」も参照してください。

用途

セレクタを指定せずに、ユーザー定義型トピックに対する永続サブスクライバを作成します。

使用上の注意

Oracle オブジェクト型のトピックに対して永続サブスクライバを作成するには、クライアントは、トピックおよびサブスクライバ名の他に、Oracle オブジェクト型に対する `CustomDatumFactory` を指定する必要があります。

構文

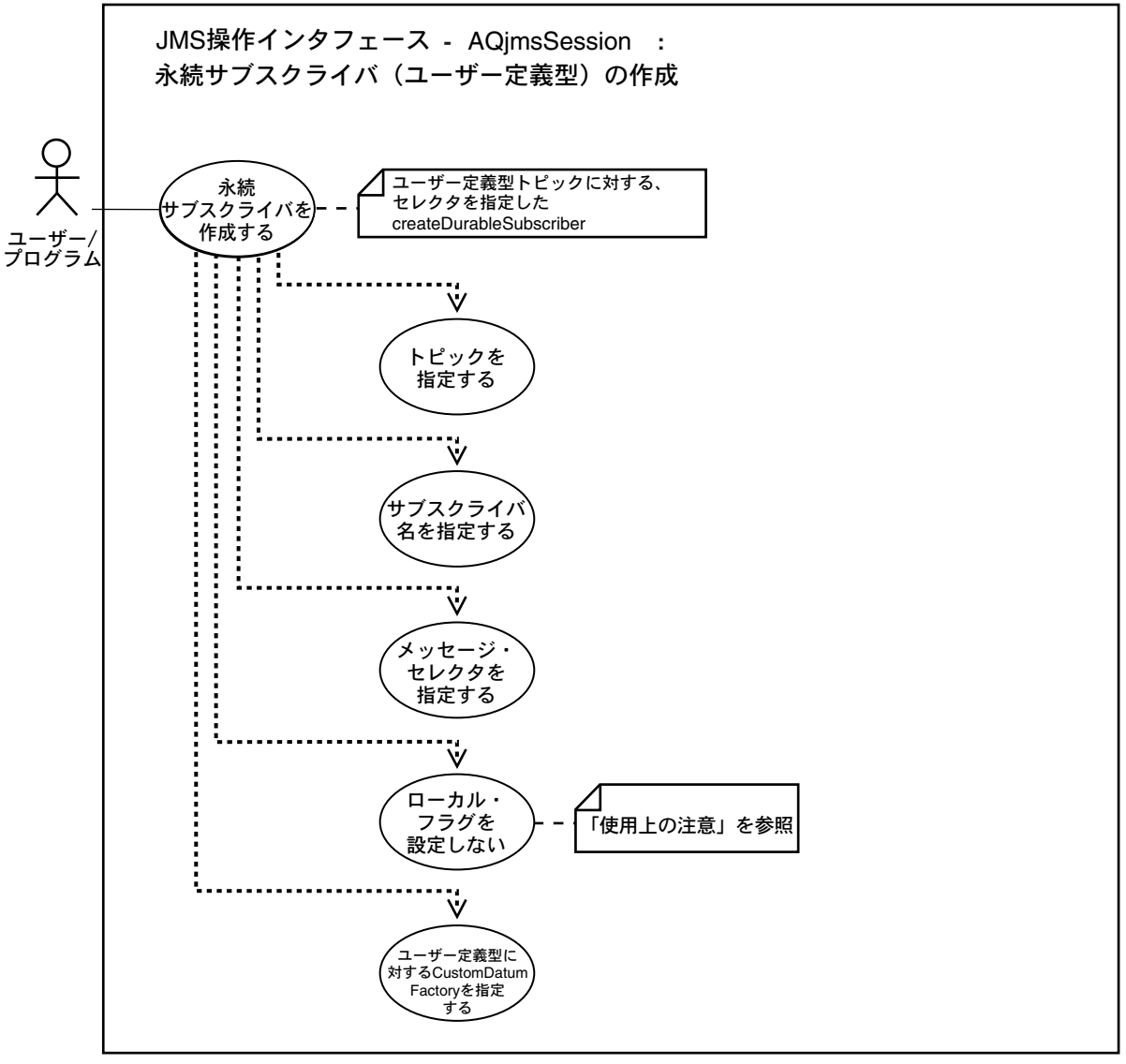
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `CreateDurableSubscriber` を参照してください。

例

```
 subscribe to an ADT queue

TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
TopicSubscriber           subscriber1;
Topic                     shipped_orders;
int                       my[port = 5521;
AQjmsAgent[]              recipList;
/* the java mapping of the oracle object type created by J Publisher */
ADTMessage                message;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");

/* create a subscriber, specifying the correct CustomDatumFactory */
subscriber1 = jms_sess.createDurableSubscriber(shipped_orders, 'WesternShipping',
AQjmsAgent.getFactory());
```

ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定あり

図 15-14 ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定あり（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-32 ページの「ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定なし」も参照してください。

用途

セレクタを指定して、ユーザー定義型トピックに対する永続サブスクライバを作成します。

使用上の注意

Oracle オブジェクト型のトピックに対して永続サブスクライバを作成するには、クライアントは、トピックおよびサブスクライバ名の他に、Oracle オブジェクト型に対する `CustomDatumFactory` を指定する必要があります。

オプションで、メッセージ・セレクタを指定できます。セレクタと一致したメッセージのみ、サブスクライバに配信されます。

ユーザー定義型メッセージには、ユーザー定義のプロパティが含まれません。ただし、セレクタを使用して、優先順位、相関識別子またはメッセージ・ペイロードの属性値に基づいて、メッセージを選択できます。

ユーザー定義型メッセージを含むキューに対するセレクタの構文は、標準 JMS ペイロード（Text、Stream、Object、Bytes、Map）を含むキューに対するセレクタの構文とは異なります。

セレクタは、AQ ルールの構文に似ています。

a. 優先順位または相関識別子のセレクタは、次のように指定されます。

```
priority > 3 AND corrid = 'Fiction'
```

b. メッセージ・ペイロードのセレクタは、次のように指定されます。属性名には、接頭辞 `tab.user_data` を付ける必要があります。

次に、使用例を示します。

```
tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
```

構文

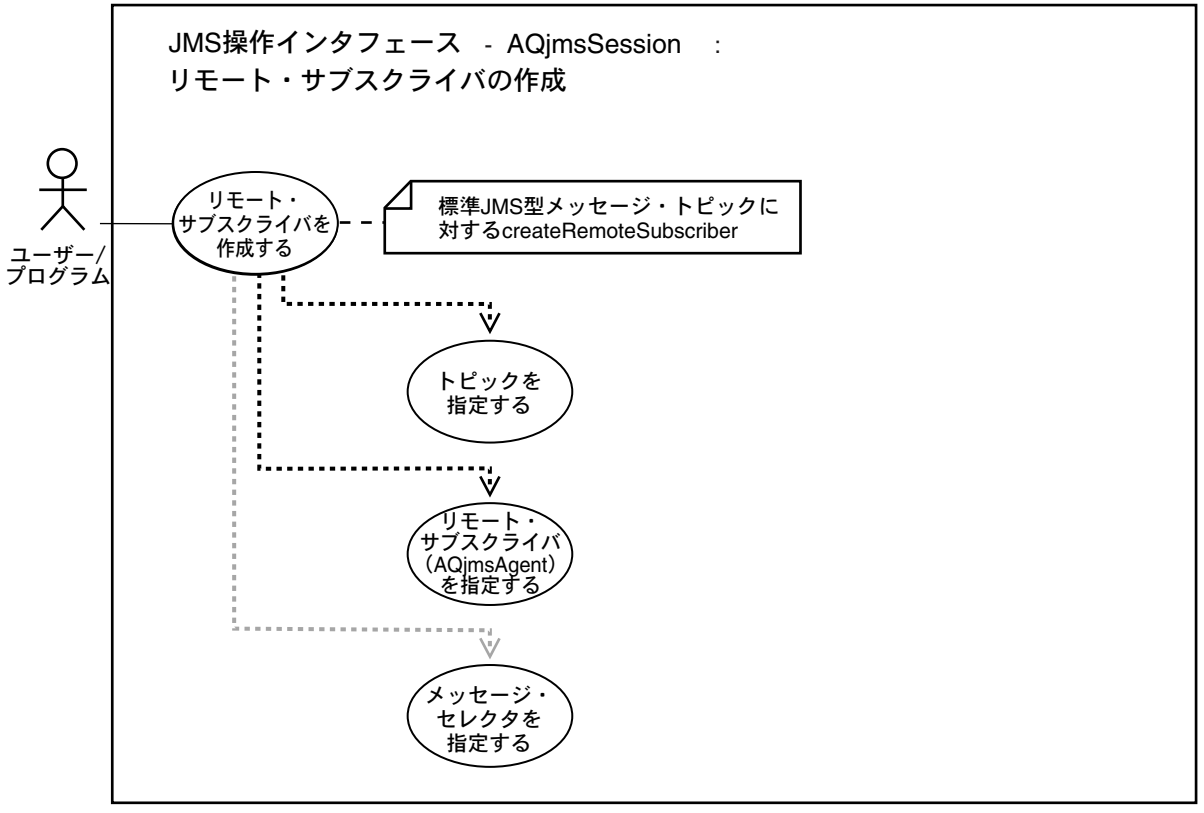
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `CreateDurableSubscriber` を参照してください。

例

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicSubscriber           subscriber1;
Topic                     shipped_orders;
int                       myport = 5521;
AQjmsAgent[]              recipList;
/* the java mapping of the oracle object type created by J Publisher */
ADTMessage                message;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");

/* create a subscriber, specifying the correct CustomDatumFactory  and selector */
subscriber1 = jms_sess.createDurableSubscriber(shipped_orders, "WesternShipping", "
priority > 1 and tab.user_data.region like 'WESTERN %'",
false, ADTMessage.getFactory());
```

リモート・サブスクライバの作成 : JMS 型メッセージ・トピック

図 15-15 リモート・サブスクライバの作成 : JMS 型メッセージ・トピック（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-40 ページの「リモート・サブスクライバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック」も参照してください。

用途

セレクタなしで、JMS 型メッセージ・トピックに対するリモート・サブスクライバを作成します。

使用上の注意

AQ では、トピックがリモート・サブスクライバ（同一または異なるデータベース内の他のトピックのサブスクライバ）を持つことができます。リモート・サブスクライバを使用するには、ローカル・トピックとリモート・トピック間の伝播を設定する必要があります。

リモート・サブスクライバは、リモート・トピックの特定のコンシューマまたはリモート・トピックのすべてのサブスクライバである場合があります。リモート・サブスクライバは、AQjmsAgent 構造を使用して定義されます。AQjmsAgent は、名前およびアドレスで構成されます。名前は、リモート・トピックのコンシューマ名を参照します。アドレスは、`(schema).(topic_name)[@dblink]` 構文のリモート・トピックを参照します。

メッセージをリモート・トピックの特定のコンシューマにパブリッシュするには、リモート・トピック・レシーバのサブスクリプション名を AQjmsAgent の name フィールドに指定する必要があります。リモート・トピックは、AQjmsAgent の address フィールドに指定する必要があります。

メッセージをリモート・トピックのすべてのサブスクライバにパブリッシュするには、AQjmsAgent の name フィールドを `NULL` に設定する必要があります。リモート・トピックは、AQjmsAgent の address フィールドに指定する必要があります。

また、メッセージ・セレクタも指定できます。セレクタを満たすメッセージのみが、リモート・サブスクライバに配信されます。メッセージ・セレクタは `NULL` になり得ます。セレクタに対する構文は、`createDurableSubscriber` の構文と同じです。

構文

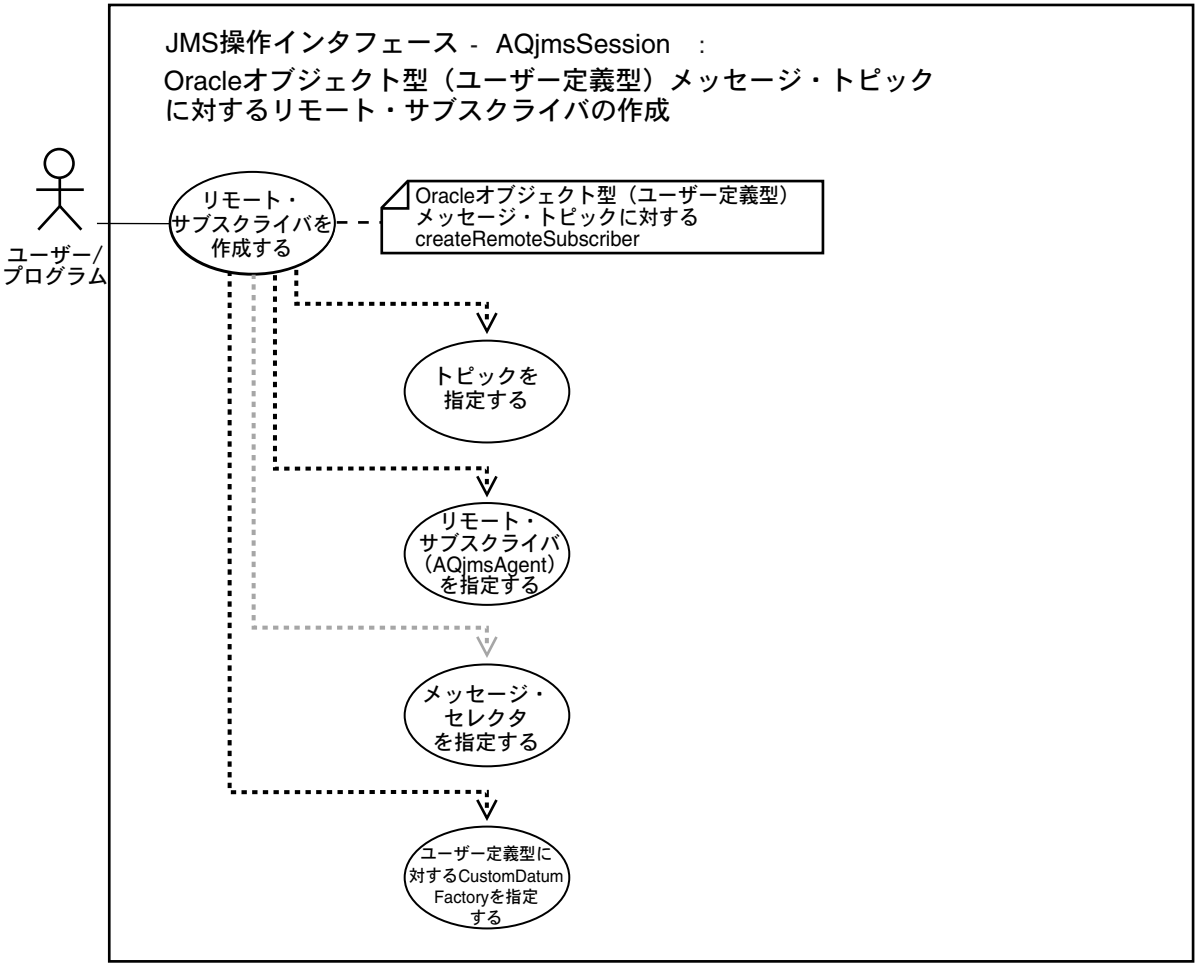
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createRemoteSubscriber` を参照してください。

例

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
TopicSubscriber           subscriber1;
Topic                     shipped_orders;
int                       my[port = 5521;
AQjmsAgent                remoteAgent;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");

remoteAgent = new AQjmsAgent("WesternRegion", "WS.shipped_orders_topic", null);

/* create a remote subscriber (selector is null )*/
subscriber1 = ((AQjmsSession)jms_sess).createRemoteSubscriber(shipped_orders,
    remoteAgent, null);
```

リモート・サブスクライバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック

図 15-16 リモート・サブスクライバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-37 ページの「リモート・サブスクライバの作成 : JMS 型メッセージ・トピック」も参照してください。

用途

Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するリモート・サブスクライバを作成します。

使用上の注意

AQ では、トピックがリモート・サブスクライバ（同一または異なるデータベース内の他のトピックのサブスクライバ）を持つことができます。リモート・サブスクライバを使用するには、ローカル・トピックとリモート・トピック間の伝播を設定する必要があります。

リモート・サブスクライバは、リモート・トピックの特定のコンシューマまたはリモート・トピックのすべてのサブスクライバである場合があります。リモート・サブスクライバは、AQjmsAgent 構造を使用して定義されます。

AQjmsAgent は、名前およびアドレスで構成されます。名前は、リモート・トピックのコンシューマ名を参照します。アドレスは、`(schema).(topic_name)[@dblink]` 構文のリモート・トピックを参照します。

メッセージをリモート・トピックの特定のコンシューマにパブリッシュするには、リモート・トピック・レシーバのサブスクリプション名を AQjmsAgent の name フィールドに指定する必要があります。リモート・トピックは、AQjmsAgent の address フィールドに指定する必要があります。

メッセージをリモート・トピックのすべてのサブスクライバにパブリッシュするには、AQjmsAgent の name フィールドを `NULL` に設定する必要があります。リモート・トピックは、AQjmsAgent の address フィールドに指定する必要があります。

トピックの Oracle オブジェクト型の `CustomDatumFactory` を指定する必要があります。また、メッセージ・セレクタも指定できます。セレクタを満たすメッセージのみが、リモート・サブスクライバに配信されます。メッセージ・セレクタは `NULL` になり得ます。メッセージ・セレクタの構文は、ユーザー定義型メッセージ・トピックの `createDurableSubscriber` の構文と同じです。

構文

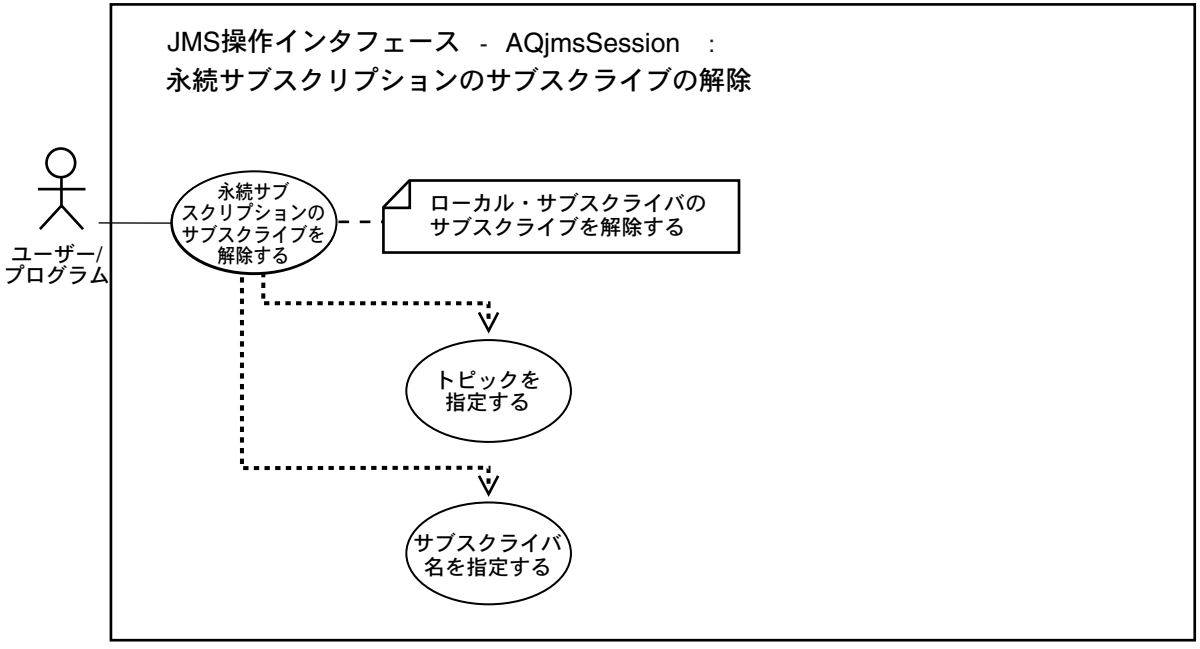
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createRemoteSubscriber` を参照してください。

例

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
TopicSubscriber           subscriber1;
Topic                     shipped_orders;
int                       my[port = 5521;
AQjmsAgent                remoteAgent;
ADTMessage                message;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
"MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");

/* create Topic session */
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

/* get the Shipped order topic */
shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");
/* create a remote agent */
remoteAgent = new AQjmsAgent("WesternRegion", "WS.shipped_orders_topic", null);

/* create a remote subscriber  with null selector*/
subscriber1 = ((AQjmsSession)jms_sess).createRemoteSubscriber
(shipped_orders, remoteAgent, null, message.getFactory);
```

永続サブスクリプションのサブスクライブの解除 : ローカル・サブスクライバ

図 15-17 永続サブスクリプションのサブスクライブの解除 : ローカル・サブスクライバ（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-45 ページの「永続サブスクリプションのサブスクライブの解除 : リモート・サブスクライバ」も参照してください。

用途

ローカル・サブスクライバに対する永続サブスクリプションのサブスクライブを解除します。

使用上の注意

特定のトピックのクライアントによって作成された永続サブスクリプションのサブスクライブを解除します。

構文

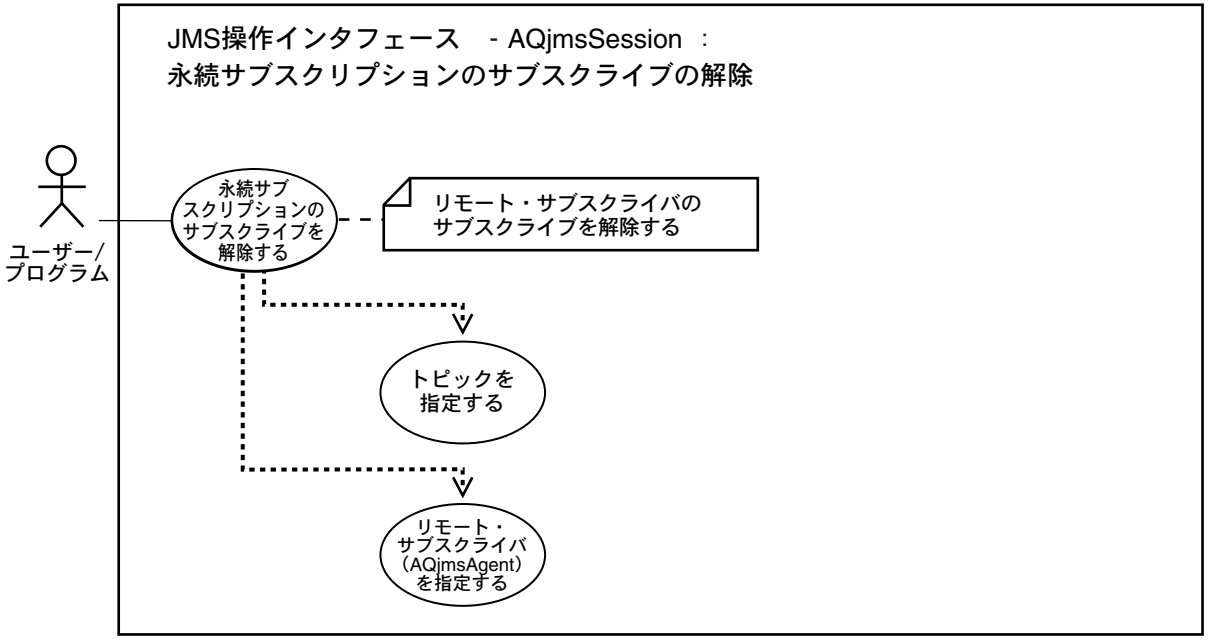
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の unsubscribe を参照してください。

例

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              jms_sess;
TopicSubscriber           subscriber1;
Topic                     shipped_orders;
int                       myport = 5521;
AQjmsAgent[]              recipList;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");
/* unsusbcribe "WesternShipping" from shipped_orders */
jms_sess.unsubscribe(shipped_orders, "WesternShipping");
```

永続サブスクリプションのサブスクライブの解除 : リモート・サブスクライバ

図 15-18 永続サブスクリプションのサブスクライブの解除 : リモート・サブスクライバ（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-43 ページの「永続サブスクリプションのサブスクライブの解除 : ローカル・サブスクライバ」も参照してください。

用途

リモート・サブスクライバに対する永続サブスクリプションのサブスクライブを解除します。

使用上の注意

ありません。

構文

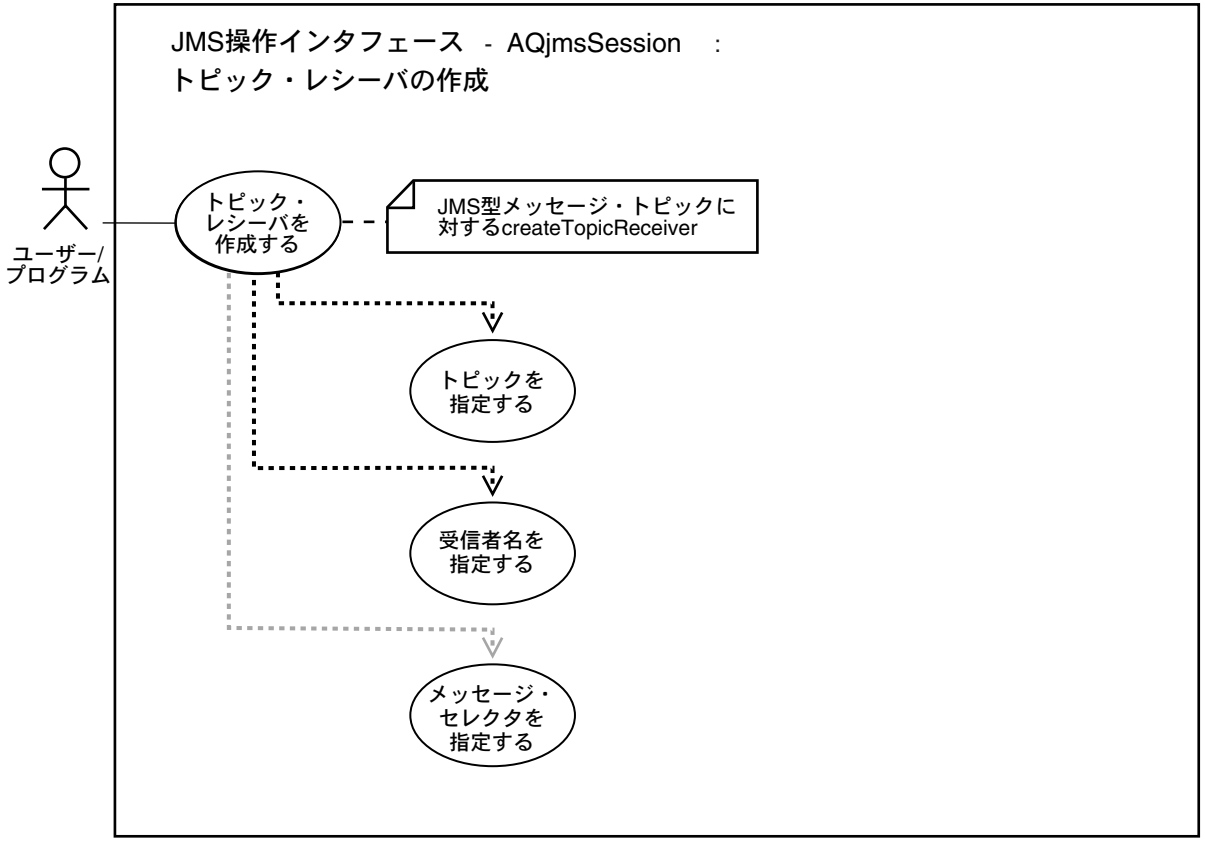
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の unsubscribe を参照してください。

例

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
Topic                     shipped_orders;
int                       myport = 5521;
AQjmsAgent                remoteAgent;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("OE", "Shipped_Orders_Topic");

remoteAgent = new AQjmsAgent("WS", "WS.Shipped_Orders_Topic", null);

/* unsubscribe the remote agent from shipped_orders */
((AQjmsSession)jms_sess).unsubscribe(shipped_orders, remoteAgent);
```

トピック・レシーバの作成 : 標準 JMS 型メッセージ・トピック

図 15-19 トピック・レシーバの作成 : 標準 JMS 型メッセージ・トピック（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-49 ページの「トピック・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック」も参照してください。

用途

標準 JMS 型メッセージ・トピックに対するトピック・レシーバを作成します。

使用上の注意

AQ では、特定の受信者にメッセージを送信できます。これらの受信者は、トピックのサブスクライバである場合とそうでない場合があります。受信者がトピックに対するサブスクライバでない場合、明示的にアドレス指定されているメッセージのみを受信します。

このメソッドは、永続サブスクライバ以外のコンシューマに対するトピック・レシーバ・オブジェクトの作成に使用する必要があります。メッセージ・セレクタが指定できます。メッセージ・セレクタの構文は、標準 JMS 型メッセージのキューに対するキュー・レシーバの構文と同じです。

構文

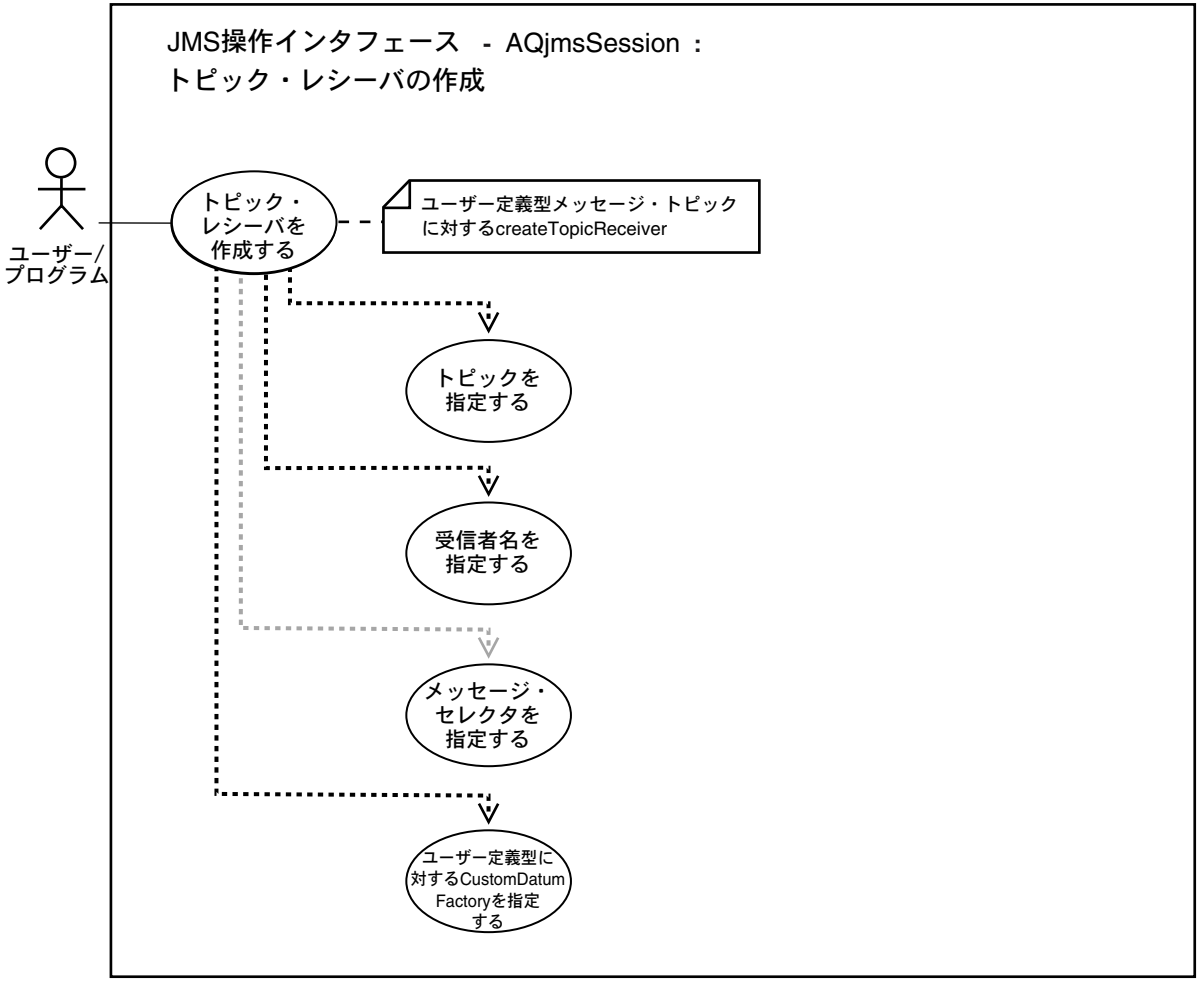
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createTopicReceiver` を参照してください。

例

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
Topic                     shipped_orders;
int                       myport = 5521;
TopicReceiver             receiver;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("WS", "Shipped_Orders_Topic");

receiver = ((AQjmsSession)jms_sess).createTopicReceiver(shipped_orders,
"WesternRegion", null);
```

トピック・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック

図 15-20 トピック・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック（パブリッシュ・サブスクライブ）



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-47 ページの「トピック・レシーバの作成 : 標準 JMS 型メッセージ・トピック」も参照してください。

用途

セレクタを指定して、ユーザー定義型メッセージ・トピックに対するトピック・レシーバを作成します。

使用上の注意

AQ では、トピックのすべてのサブスクライバ、または特定の受信者にメッセージを送信できます。これらの受信者は、トピックのサブスクライバである場合とそうでない場合があります。受信者がトピックに対するサブスクライバでない場合、明示的にアドレス指定されているメッセージのみを受信します。

このメソッドは、永続サブスクライバ以外のコンシューマに対するトピック・レシーバ・オブジェクトの作成に使用する必要があります。キューの Oracle オブジェクト型の `CustomDatumFactory` を指定する必要があります。また、メッセージ・セレクタも指定できます。これは `NULL` になり得ます。メッセージ・セレクタの構文は、ユーザー定義型メッセージのキューに対するキュー・レシーバの構文と同じです。

構文

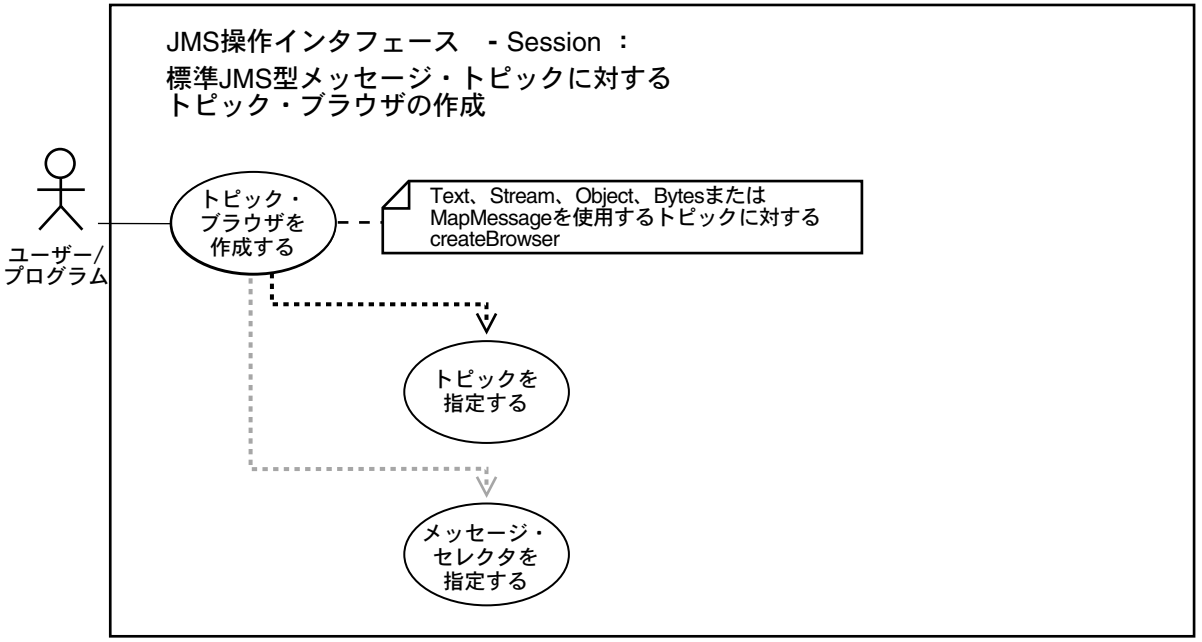
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createTopicReceiver` を参照してください。

例

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
Topic                     shipped_orders;
int                       myport = 5521;
TopicReceiver             receiver;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("WS", "Shipped_Orders_Topic");

receiver = ((AQjmsSession)jms_sess).createTopicReceiver(shipped_orders,
"WesternRegion", null);
```

Text、Stream、Object、Bytes、MapMessage を使用するトピックに対するトピック・ブラウザの作成

図 15-21 Text、Stream、Object、Bytes、MapMessage を使用するトピックに対するトピック・ブラウザの作成



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。
- 15-54 ページの「Text、Stream、Object、Bytes、MapMessage を使用するトピックに対するトピック・ブラウザの作成 : メッセージをロック」も参照してください。

用途

Text、Stream、Object、Bytes または MapMessage を使用するトピックに対するトピック・ブラウザを作成します。

使用上の注意

特定の相関識別子を持つメッセージを取り出すには、トピック・ブラウザに対するセレクタを次のいずれかに指定します。

- JMS メッセージ ID

  ```
  JMSMessageID = 'ID:23452345'
  ```

  指定されたメッセージ ID を持つメッセージを取り出します。

- JMS メッセージ・ヘッダー・フィールドまたはプロパティ

  ```
  JMSPriority < 3 AND JMSCorrelationID = 'Fiction'
  ```

- ユーザー定義のメッセージ・プロパティ

  ```
  color IN ('RED', BLUE', 'GREEN') AND price < 30000
  ```

すべてのメッセージ ID には、接頭辞 ID: が必要です。`java.util.Enumeration` 内のメソッドを使用して、メッセージのリストを参照してください。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createBrowser` を参照してください。

例

例 1

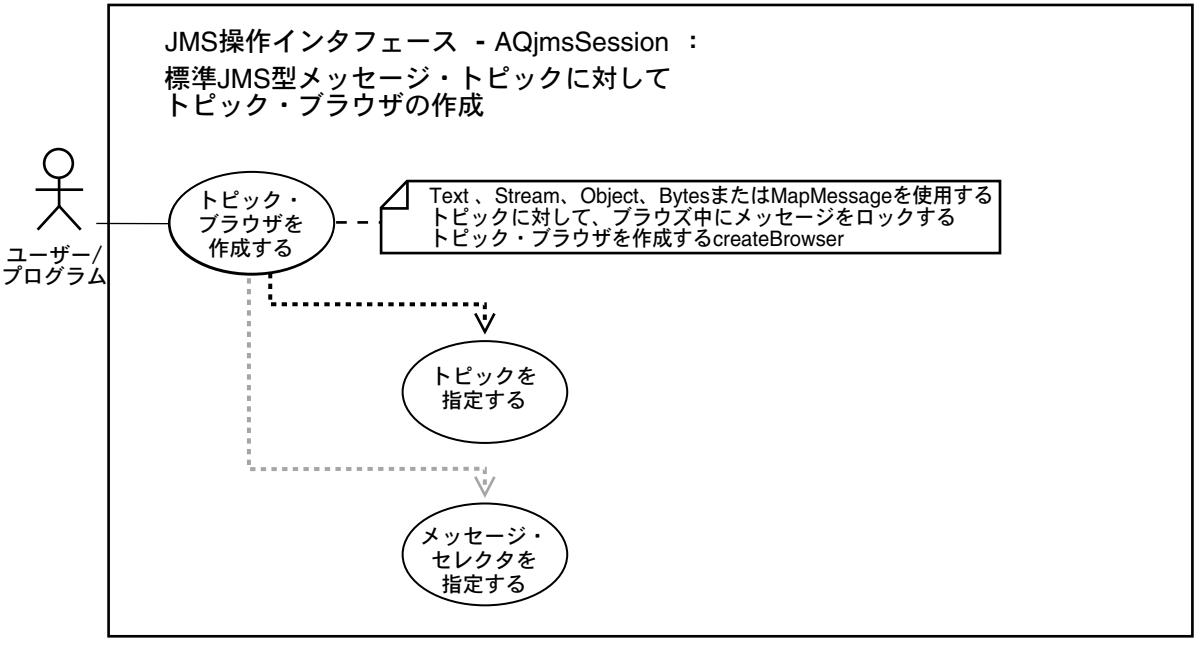
```
/* Create a browser without a selector */
TopicSession    jms_session;
TopicBrowser    browser;
Topic           topic;

browser = ((AQjmsSession) jms_session).createBrowser(topic, "SUBS1");
```

例 2

```
/* Create a browser for topics with a specified selector */
TopicSession    jms_session;
TopicBrowser    browser;
Topic           topic;

/* create a Browser to look at messages with correlationID = RUSH  */
browser = ((AQjmsSession) jms_session).createBrowser(topic, "SUBS1",
    "JMSCorrelationID = 'RUSH'");
```

Text、Stream、Object、Bytes、MapMessage を使用するトピックに対するトピック・ブラウザの作成 : メッセージをロック

図 15-22 Text、Stream、Object、Bytes、MapMessage を使用するトピックに対するトピック・ブラウザの作成 : メッセージをロック



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。
- 15-52 ページの「Text、Stream、Object、Bytes、MapMessage を使用するトピックに対するトピック・ブラウザの作成」も参照してください。

用途

Text、Stream、Object、Bytes または MapMessage を使用するトピックに対して、ブラウズ中にメッセージをロックするトピック・ブラウザを作成します。

使用上の注意

locked パラメータが TRUE に指定されている場合、ブラウズ中のメッセージはロックされます。したがって、ブラウズ中のセッションがトランザクションを終了するまで、他のコンシューマがこれらのメッセージを削除することはできません。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の createBrowser を参照してください。

例

例 1

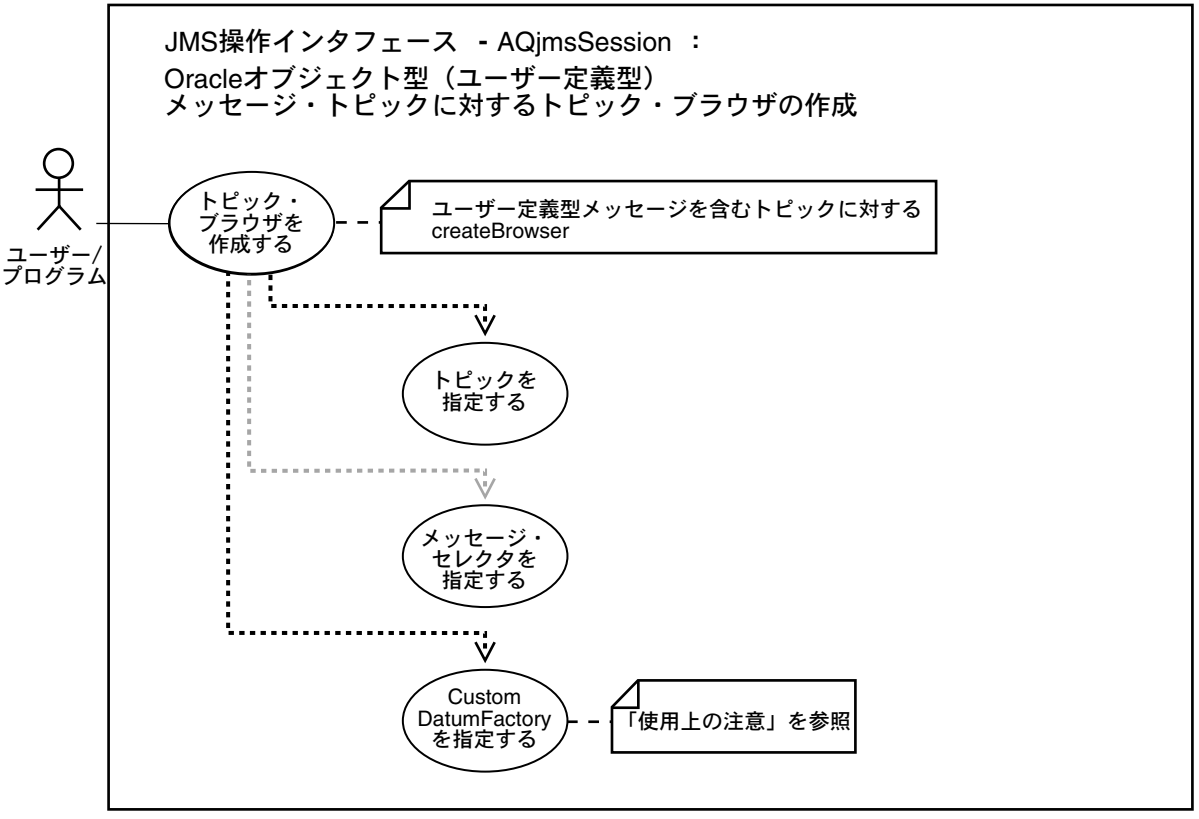
```
/* Create a browser without a selector */
TopicSession    jms_session;
TopicBrowser    browser;
Topic           topic;

browser = ((AQjmsSession) jms_session).createBrowser(topic,
    "SUBS1", true);
```

例 2

```
/* Create a browser for topics with a specified selector */
TopicSession    jms_session;
TopicBrowser    browser;
Topic           topic;

/* create a Browser to look at messages with correlationID = RUSH in
lock mode */

browser = ((AQjmsSession) jms_session).createBrowser(topic,
    "SUBS1", "JMSCorrelationID = 'RUSH'", true);
```

Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザの作成

図 15-23 Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザの作成



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-59 ページの「Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザの作成 : メッセージをロック」も参照してください。

用途

Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザを作成します。

使用上の注意

AdtMessage を含むトピックの場合、トピック・ブラウザのセレクタは、メッセージ・ペイロード内容、メッセージ ID、優先順位または相関識別子に対する SQL 式で指定します。

- メッセージ ID に対するセレクタは、次のように指定します。指定したメッセージ ID を持つメッセージが取り出されます。

  ```
  msgid = '23434556566767676'
  ```

  注意： この場合は、メッセージ ID に接頭辞 ID: を使用しないでください。

- 優先順位または相関識別子に対するセレクタは、次のように指定します。

  ```
  priority < 3 AND corrid = 'Fiction'
  ```

- メッセージ・ペイロードに対するセレクタは、次のように指定します。

  ```
  tab.user_data.color = 'GREEN' AND tab.user_data.price < 30000
  ```

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の createBrowser を参照してください。

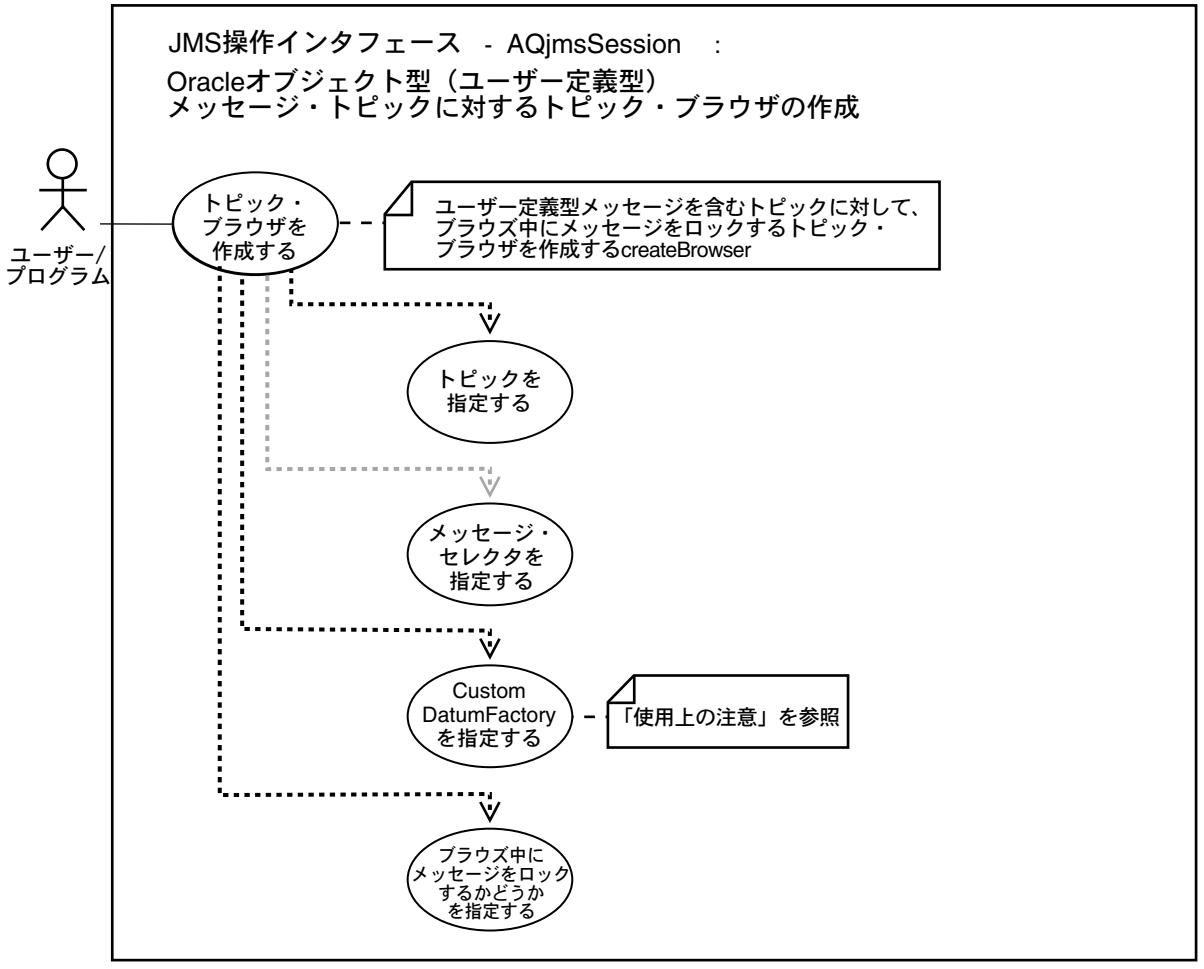
例

SQL ユーザー定義型ペイロードにマップする特定の Java クラスに対する CustomDatumFactory は、static メソッド getFactory を使用して取得できます。トピック test_topic は SCOTT.EMPLOYEE 型のペイロードを持ち、このユーザー定義型に対して、Employee という Java クラスが JPublisher によって生成されたとします。Employee クラスは、CustomDatum インタフェースを実装します。このクラスに対する CustomDatumFactory は、Employee.getFactory() メソッドを使用して取得できます。

```
/* Create a browser for a Topic with Adt messages of type EMPLOYEE*/
TopicSession jms_session
TopicBrowser browser;
Topic        test_topic;

browser = ((AQjmsSession) jms_session).createBrowser(test_topic,
    "SUBS1", Employee.getFactory());
```

Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザの作成 : メッセージをロック

図 15-24 Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザの作成 : メッセージをロック



参照 :

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。
- 15-56 ページの「Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対するトピック・ブラウザの作成」も参照してください。

用途

Oracle オブジェクト型（ユーザー定義型）メッセージ・トピックに対して、ブラウズ中にメッセージをロックするトピック・ブラウザを作成します。

使用上の注意

ありません。

構文

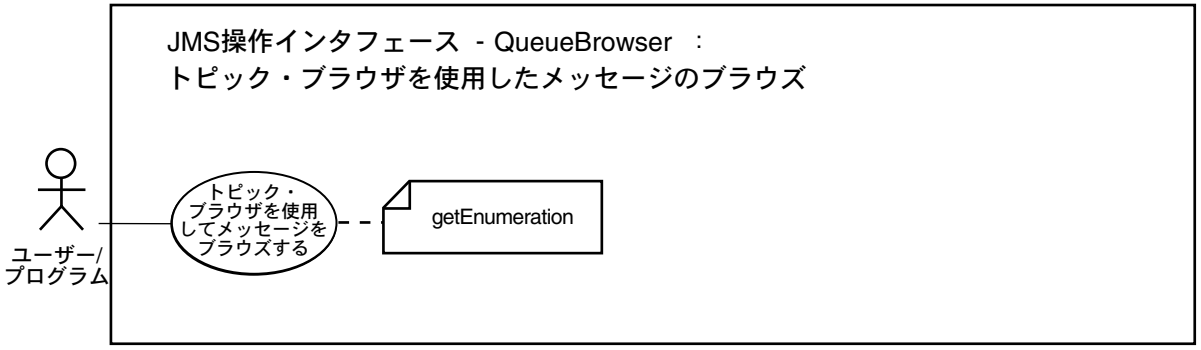
Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createBrowser` を参照してください。

例

```
/* Create a browser for a Topic with ADT messages of type EMPLOYEE* in
lock mode/
TopicSession jms_session
TopicBrowser browser;
Topic        test_topic;

browser = ((AQjmsSession) jms_session).createBrowser(test_topic,
    "SUBS1", Employee.getFactory(), true);
```

トピック・ブラウザを使用したメッセージのブラウズ

図 15-25 トピック・ブラウザを使用したメッセージのブラウズ



参照：

- JMS 操作インタフェースのパブリッシュ・サブスクライブの基本操作の詳細は、表 15-1 を参照してください。
- B-62 ページの「クラス - oracle.jms.AQjmsTopicBrowser」も参照してください。

用途

トピック・ブラウザを使用して、メッセージをブラウズします。

使用上の注意

`java.util.Enumeration` 内のメソッドを使用して、メッセージのリストを参照してください。現行のブラウズ中に参照したメッセージを削除するには、トピック・ブラウザ内の `purgeSeen` メソッドを使用してください。

構文

Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「TopicBrowser」の `AQjmsTopicBrowser` を参照してください。

例

```
/* Create a browser for topics with a specified selector */
public void browse_rush_orders(TopicSession jms_session)
{
    TopicBrowser    browser;
    Topic           topic;
    ObjectMessage   obj_message
    BolOrder        new_order;
    Enumeration     messages;

    /* get a handle to the new_orders topic */
    topic = ((AQjmsSession) jms_session).getTopic("OE", "OE_bookedorders_topic");

    /* create a Browser to look at RUSH orders */
    browser = ((AQjmsSession) jms_session).createBrowser(topic,
        "SUBS1", "JMSCorrelationID = 'RUSH'");

    /* Browse through the messages */
    for (messages = browser.elements() ; message.hasMoreElements() ;)
    {
        obj_message = (ObjectMessage)message.nextElement();
    }

    /* Purge messages seen during this browse */
    browser.purgeSeen();
}
```

16

JMS 操作インタフェース : 基本操作（共有インタフェース）

この章では、Oracle Advanced Queuing の操作インタフェース（共有インタフェース）を利用方法に沿って説明します。それぞれの操作（メッセージのエンキューなど）を、その操作名ごとに利用方法に沿って説明します。この章の先頭に、すべての利用方法を示します（16-2 ページの「利用モデル : 操作インタフェース - 基本操作（共有インタフェース）」を参照）。

図 16-1 に、すべての利用方法を 1 つの図にまとめています。HTML 版のマニュアルをご使用の場合、この図の中の関連する利用方法のタイトルをクリックすることで、関心のある利用方法に移動することができます。

個々の利用方法は、次の形式で説明されています。

- **利用図 :** 利用方法を表す図
- **用途 :** この利用方法の用途
- **使用上の注意 :** 実装に有効なガイドライン
- **構文 :** このアクティビティの実行に使用する主な構文
- **例 :** 各プログラム環境での利用例

利用モデル : JMS 操作インタフェース - 基本操作（共有インタフェース）

表 16-1 利用モデル : 操作インタフェース - 基本操作（共有インタフェース）

利用方法
JMS コネクションの開始（16-5 ページ）
セッションからの JMS コネクションの取得（16-7 ページ）
セッションにおけるすべての操作のコミット（16-9 ページ）
セッションにおけるすべての操作のロールバック（16-11 ページ）
JMS セッションからの JDBC コネクションの取得（16-13 ページ）
JMS コネクションからの OracleOCIConnectionPool の取得（16-15 ページ）
BytesMessage の作成（16-17 ページ）
MapMessage の作成（16-19 ページ）
StreamMessage の作成（16-21 ページ）
ObjectMessage の作成（16-23 ページ）
TextMessage の作成（16-25 ページ）
JMS メッセージの作成（16-27 ページ）
JMS メッセージの作成（ヘッダーのみ）（16-29 ページ）
ユーザー定義型メッセージの作成（16-30 ページ）
メッセージの相関識別子の指定（16-32 ページ）
JMS メッセージ・プロパティの指定（16-34 ページ）
JMS メッセージ・プロパティを Boolean として指定（16-37 ページ）
JMS メッセージ・プロパティを String として指定（16-40 ページ）
JMS メッセージ・プロパティを Int として指定（16-43 ページ）
JMS メッセージ・プロパティを Double として指定（16-46 ページ）
JMS メッセージ・プロパティを Float として指定（16-49 ページ）
JMS メッセージ・プロパティを Byte として指定（16-52 ページ）
JMS メッセージ・プロパティを Long として指定（16-55 ページ）
JMS メッセージ・プロパティを Short として指定（16-58 ページ）

表 16-1 利用モデル : 操作インタフェース - 基本操作（共有インタフェース）（続き）

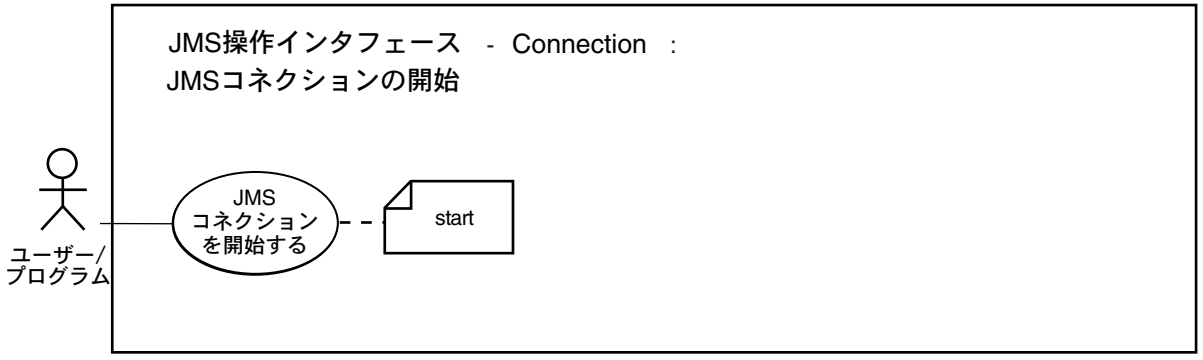
利用方法
JMS メッセージ・プロパティを Object として指定（16-61 ページ）
メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの Time-To-Live の設定（16-64 ページ）
メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの優先順位の設定（16-66 ページ）
AQjms エージェントの作成（16-68 ページ）
メッセージ・コンシューマを使用したメッセージの同期受信 : タイムアウトを指定（16-70 ページ）
メッセージ・コンシューマを使用したメッセージの同期受信 : 待機なし（16-72 ページ）
メッセージの受信に対するナビゲーション・モードの指定（16-74 ページ）
メッセージを非同期受信するメッセージ・リスナーの指定 : メッセージ・コンシューマ（16-77 ページ）
メッセージを非同期受信するメッセージ・リスナーの指定 : セッション（16-80 ページ）
メッセージの相関識別子の取得（16-82 ページ）
メッセージのメッセージ ID を Byte として取得（16-83 ページ）
メッセージのメッセージ ID を String として取得（16-85 ページ）
JMS メッセージ・プロパティの取得（16-87 ページ）
JMS メッセージ・プロパティを Boolean として取得（16-89 ページ）
JMS メッセージ・プロパティを String として取得（16-92 ページ）
JMS メッセージ・プロパティを Int として取得（16-95 ページ）
JMS メッセージ・プロパティを Double として取得（16-98 ページ）
JMS メッセージ・プロパティを Float として取得（16-101 ページ）
JMS メッセージ・プロパティを Byte として取得（16-104 ページ）
JMS メッセージ・プロパティを Long として取得（16-107 ページ）
JMS メッセージ・プロパティを Short として取得（16-110 ページ）
JMS メッセージ・プロパティを Object として取得（16-110 ページ）
メッセージ・プロデューサのクローズ（16-116 ページ）
メッセージ・コンシューマのクローズ（16-117 ページ）
JMS コネクションの停止（16-118 ページ）
JMS セッションを閉じる（16-120 ページ）

表 16-1 利用モデル : 操作インタフェース - 基本操作（共有インタフェース）（続き）

利用方法
JMS コネクションのクローズ（16-122 ページ）
JMS 例外のエラー・コードの取得（16-124 ページ）
JMS 例外のエラー番号の取得（16-125 ページ）
JMS 例外のエラー・メッセージの取得（16-126 ページ）
JMS 例外にリンクされた例外の取得（16-127 ページ）
JMS 例外のスタック・トレースの出力（16-129 ページ）
例外リスナーの設定（16-130 ページ）
例外リスナーの取得（16-132 ページ）
例外リスナーの ping 周期の設定（16-133 ページ）
例外リスナーの ping 周期の取得（16-135 ページ）

JMS コネクションの開始

図 16-1 JMS コネクションの開始



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-25 ページの「インタフェース - javax.jms.Connection」も参照してください。

用途

メッセージを受信するために、JMS コネクションを開始します。

使用上の注意

`start` メソッドを使用して、受信メッセージのコネクション配信を開始（または再開）します。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `start`

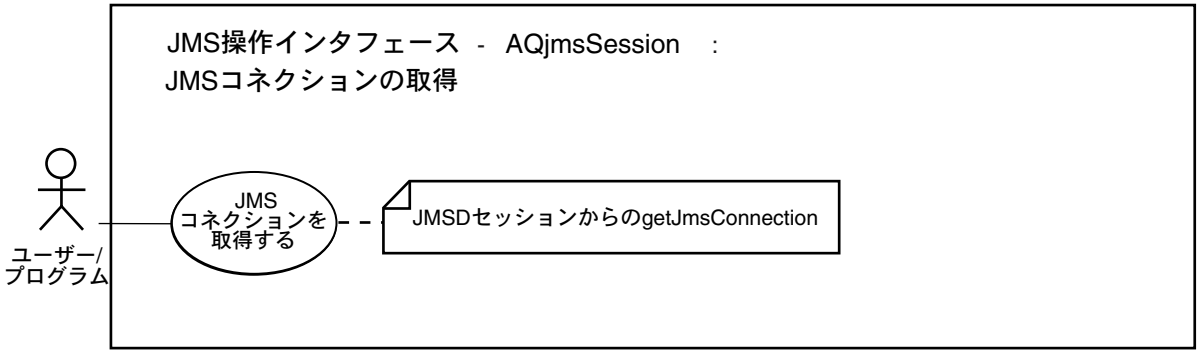
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

セッションからの JMS コネクションの取得

図 16-2 セッションからの JMS コネクションの取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

セッションから JMS コネクションを取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `getJmsConnection`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

セッションにおけるすべての操作のコミット

図 16-3 セッションにおけるすべての操作のコミット

参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

セッションにおけるすべての操作をコミットします。

使用上の注意

このメソッドは、このセッションで実行されているすべての JMS および SQL 操作をコミットします。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `commit`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

セッションにおけるすべての操作のロールバック

図 16-4 セッションにおけるすべての操作のロールバック

参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

セッションにおけるすべての操作をロールバックします。

使用上の注意

このメソッドは、このセッションで実行されているすべての JMS および SQL 操作を強制終了させます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `rollback`

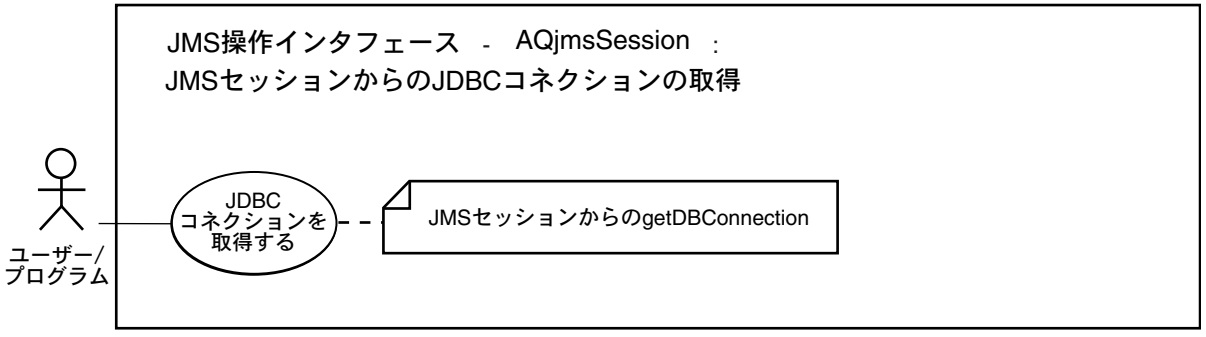
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS セッションからの JDBC コネクションの取得

図 16-5 JMS セッションからの JDBC コネクションの取得



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

JMS セッションから、JDBC コネクションを取得します。

使用上の注意

このメソッドは、JMS セッションから、JDBC コネクションを取得するために使用されます。JDBC コネクションは、JMS 操作が実行される同一のトランザクションの一部として、SQL 操作を実行するために使用される場合があります。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `getDBConnection`

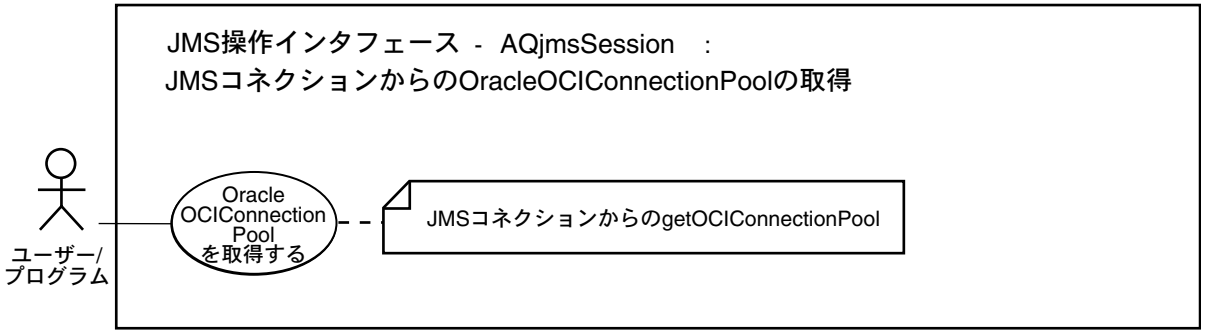
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

```
java.sql.Connection db_conn;
QueueSession     jms_sess;
db_conn = ((AQjmsSession)jms_sess).getDBConnection();
```

JMS コネクションからの OracleOCIConnectionPool の取得

図 16-6 JMS コネクションからの OracleOCIConnectionPool の取得



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

JMS コネクションから `OracleOCIConnectionPool` を取得します。

使用上の注意

このメソッドは、JMS コネクションから、`OracleOCIConnectionPool` インスタンスを取得するために使用されます。`OracleOCIConnectionPool` インスタンスの設定は、コネクションの使用状況（ユーザーが既存のコネクションを使用して確立するセッション数など）に応じてユーザーがチューニングできます。ただし、JMS コネクションが使用している `OracleOCIConnectionPool` インスタンスはクローズしないでください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

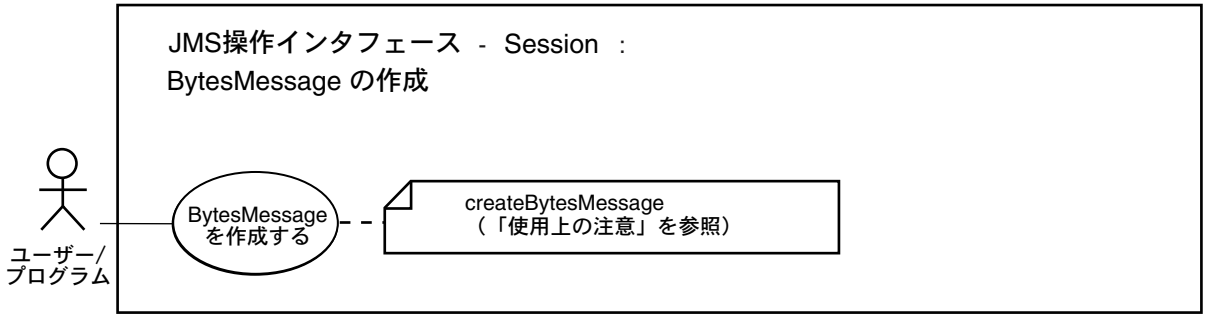
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `getOCIConnectionPool`

例

```
oracle.jdbc.pool.OracleOCIConnectionPool cpool;
QueueConnection jms_conn;
cpool = ((AQjmsConnection)jms_conn).getOCIConnectionPool();
```

BytesMessage の作成

図 16-7 BytesMessage の作成



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

BytesMessage を作成します。

使用上の注意

このメソッドは、宛先キュー / トピックを含むキュー表が、ペイロード型 `SYS.AQ$_JMS_BYTES_MESSAGE` または `AQ$_JMS_MESSAGE` で作成された場合にのみ、使用できます。

BytesMessage の移入に使用されるメソッドについては、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createBytesMessage`

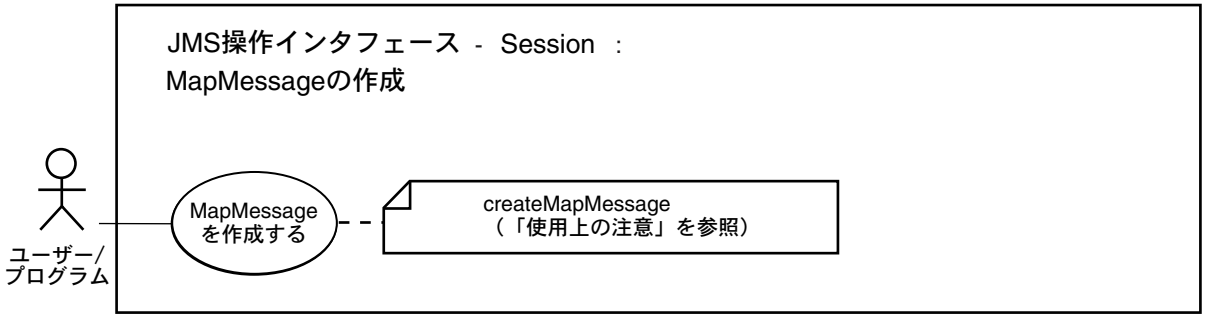
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

MapMessage の作成

図 16-8 MapMessage の作成



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

MapMessage を作成します。

使用上の注意

このメソッドは、宛先キュー / トピックを含むキュー表が、ペイロード型 `SYS.AQ$_JMS_MAP_MESSAGE` または `AQ$_JMS_MESSAGE` で作成された場合にのみ、使用できます。

MapMessage の移入に使用されるメソッドについては、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createMapMessage`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

StreamMessage の作成

図 16-9 StreamMessage の作成

参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

StreamMessage を作成します。

使用上の注意

このメソッドは、宛先キュー / トピックを含むキュー表が、ペイロード型 `SYS.AQ$_JMS_STREAM_MESSAGE` または `AQ$_JMS_MESSAGE` で作成された場合にのみ、使用できます。

StreamMessage の移入に使用されるメソッドについては、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createStreamMessage`

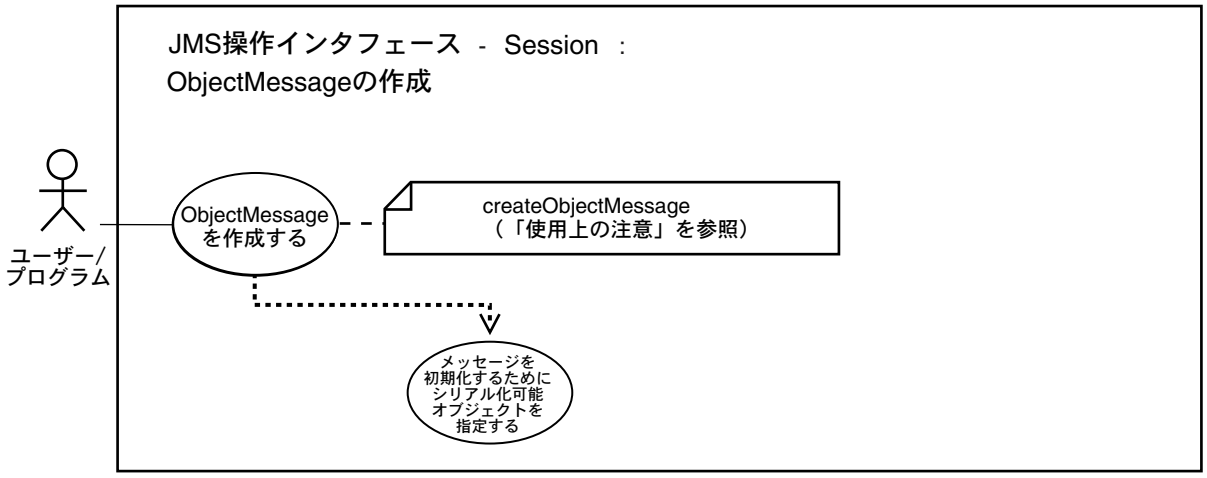
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

ObjectMessage の作成

図 16-10 ObjectMessage の作成



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

ObjectMessage を作成します。

使用上の注意

このメソッドは、宛先キュー / トピックを含むキュー表が、ペイロード型 `SYS.AQ$_JMS_OBJECT_MESSAGE` または `AQ$_JMS_MESSAGE` で作成された場合にのみ、使用できます。

ObjectMessage の移入に使用されるメソッドについては、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createObjectMessage`

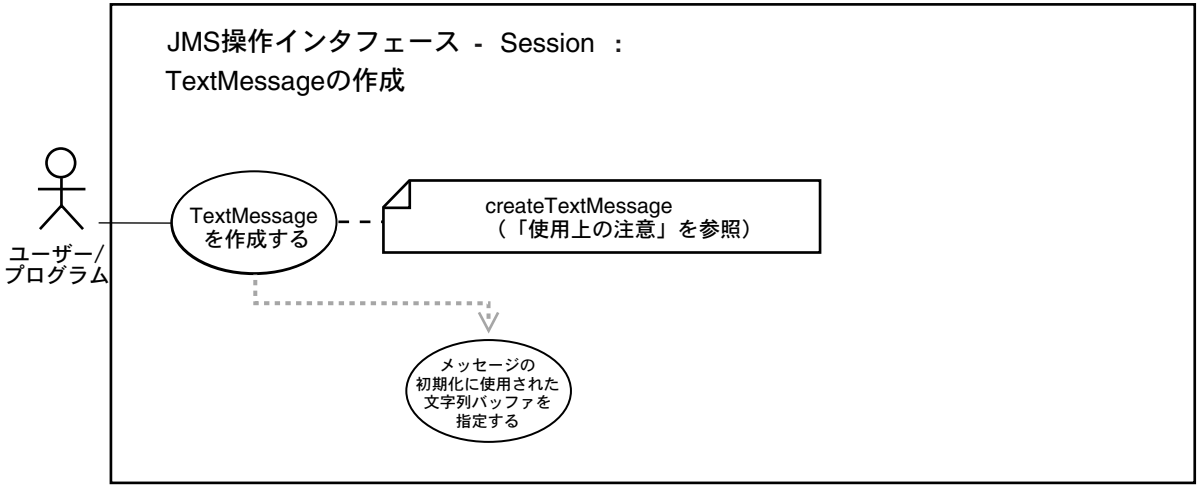
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

TextMessage の作成

図 16-11 TextMessage の作成



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

TextMessage を作成します。

使用上の注意

このメソッドは、宛先キュー / トピックを含むキュー表が、ペイロード型 `SYS.AQ$_JMS_TEXT_MESSAGE` または `AQ$_JMS_MESSAGE` で作成された場合にのみ、使用できます。

TextMessage の移入に使用されるメソッドについては、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

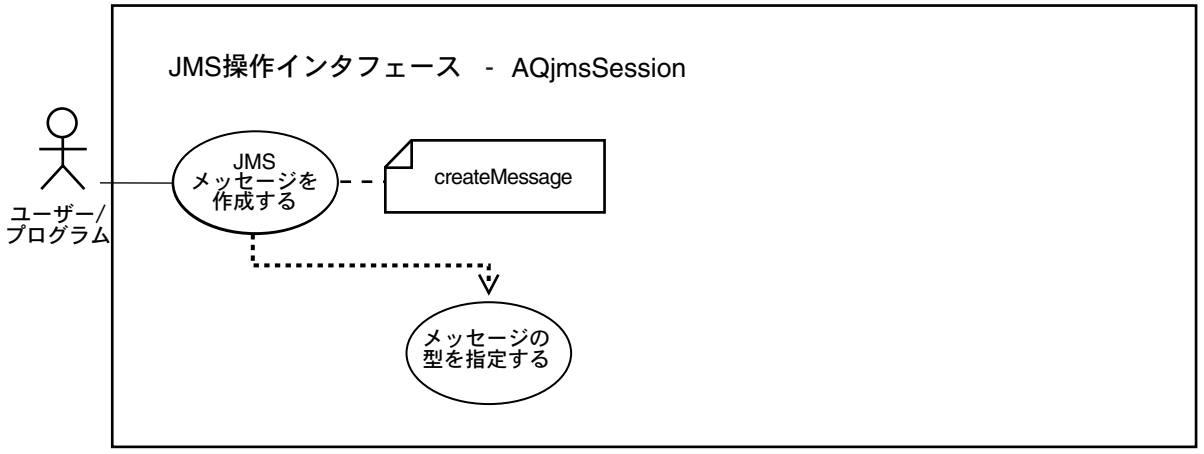
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createTextMessage`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

JMS メッセージの作成

図 16-12 JMS メッセージの作成



用途

JMS メッセージを作成します。

使用上の注意

このユーザー定義型は、BytesMessage（JMSBytes）、MapMessage（JMSMap）、StreamMessage（JMSStream）、ObjectMessage（JMSObject）、TextMessage（JMSText）などの JMS メッセージ型の一部またはすべてを格納するために使用します。

AQ$_JMS_MESSAGE メッセージを使用して、様々な型のメッセージを構成できます。メッセージ型は、次のいずれかである必要があります。

- DBMS_AQ.JMS_TEXT_MESSAGE
- DBMS_AQ.JMS_OBJECT_MESSAGE
- DBMS_AQ.JMS_MAP_MESSAGE
- DBMS_AQ.JMS_BYTES_MESSAGE
- DBMS_AQ.JMS_STREAM_MESSAGE

> **参照：** JMS 型の詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createMessage`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

JMS メッセージの作成（ヘッダーのみ）

図 16-13 JMS メッセージの作成（ヘッダーのみ）

用途

ヘッダーのみの JMS メッセージを作成します。

使用上の注意

このユーザー定義型は、BytesMessage（`JMSBytes`）、MapMessage（`JMSMap`）、StreamMessage（`JMSStream`）、ObjectMessage（`JMSObject`）、TextMessage（`JMSText`）などの JMS メッセージ型の一部またはすべてを格納するために使用します。

> **参照：** JMS 型の詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

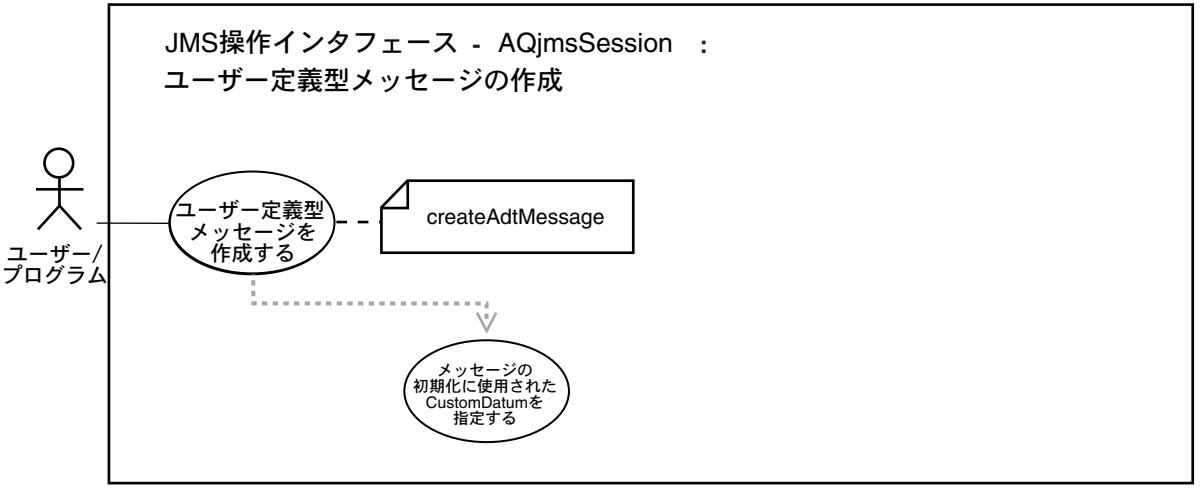
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `createMessage`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

ユーザー定義型メッセージの作成

図 16-14 ユーザー定義型メッセージの作成



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

用途

ユーザー定義型メッセージを作成します。

使用上の注意

このメソッドは、キュー / トピックを含むキュー表が、Oracle のユーザー定義型（`SYS.AQ$_JMS*` 型を除く）で作成された場合にのみ、使用できます。

ユーザー定義型メッセージは、`CustomDatum` インタフェースを実装するオブジェクトで移入される必要があります。このオブジェクトは、キュー / トピックのペイロードとして定義されている、SQL のユーザー定義型 Java マッピングである必要があります。SQL のユーザー定義型に対応する Java クラスは、JPublisher ツールを使用して生成される場合があります。`CustomDatum` インタフェースおよび JPublisher の詳細は、JDBC ドキュメントを参照してください。

AdtMessage の移入に使用されるメソッドについては、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の createAdtMessage

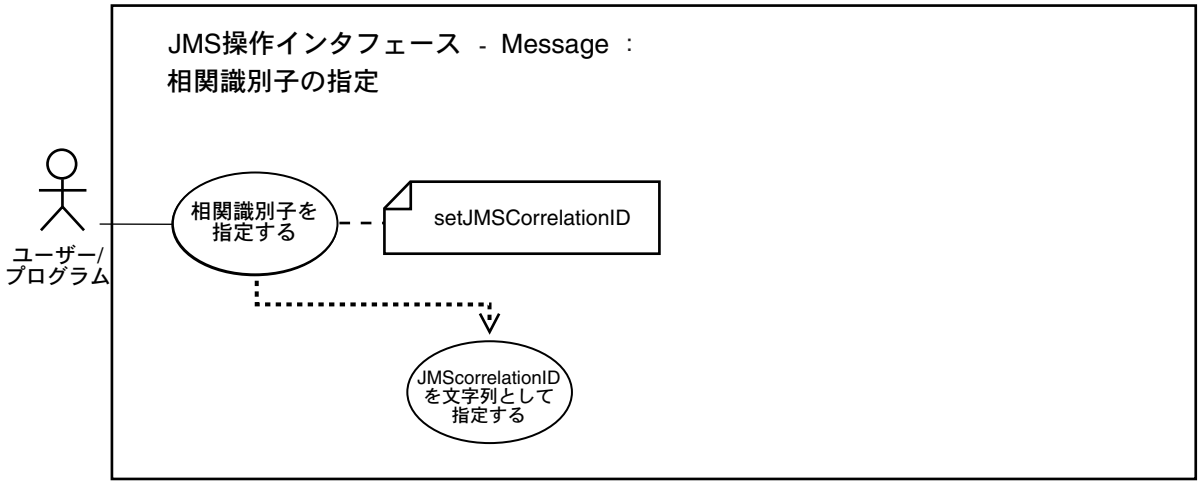
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

メッセージの相関識別子の指定

図 16-15 メッセージの相関識別子の指定



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。

用途

メッセージの相関識別子を指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setJMSCorrelationID`

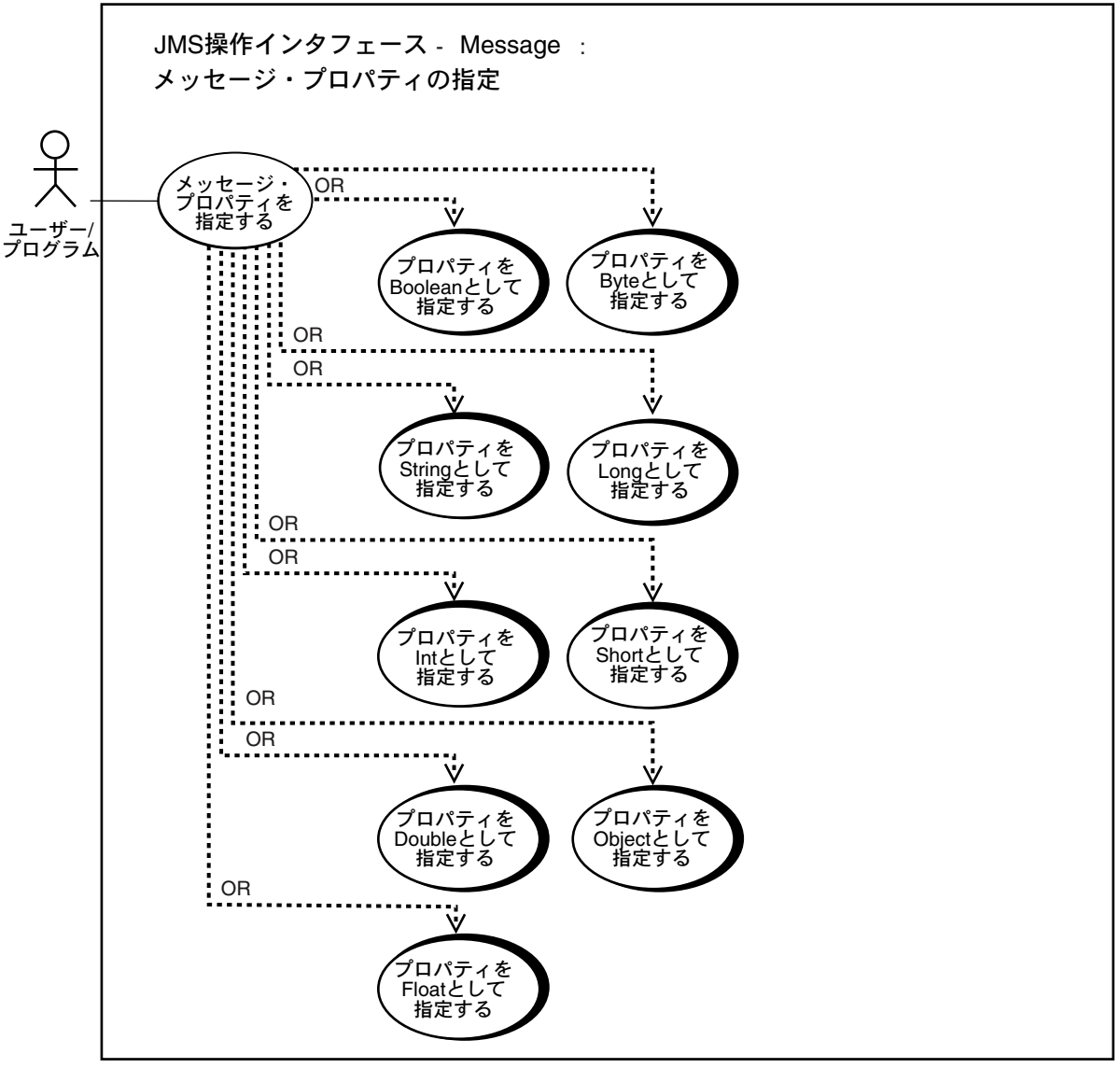
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティの指定

図 16-16 JMS メッセージ・プロパティの指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

使用上の注意

JMS で始まるプロパティ名は、プロバイダ固有です。ユーザー定義のプロパティは、JMS で始めることができません。

次のプロバイダのプロパティは、クライアントが TextMessage、StreamMessage、ObjectMessage、BytesMessage または MapMessage を使用して、設定する場合があります。

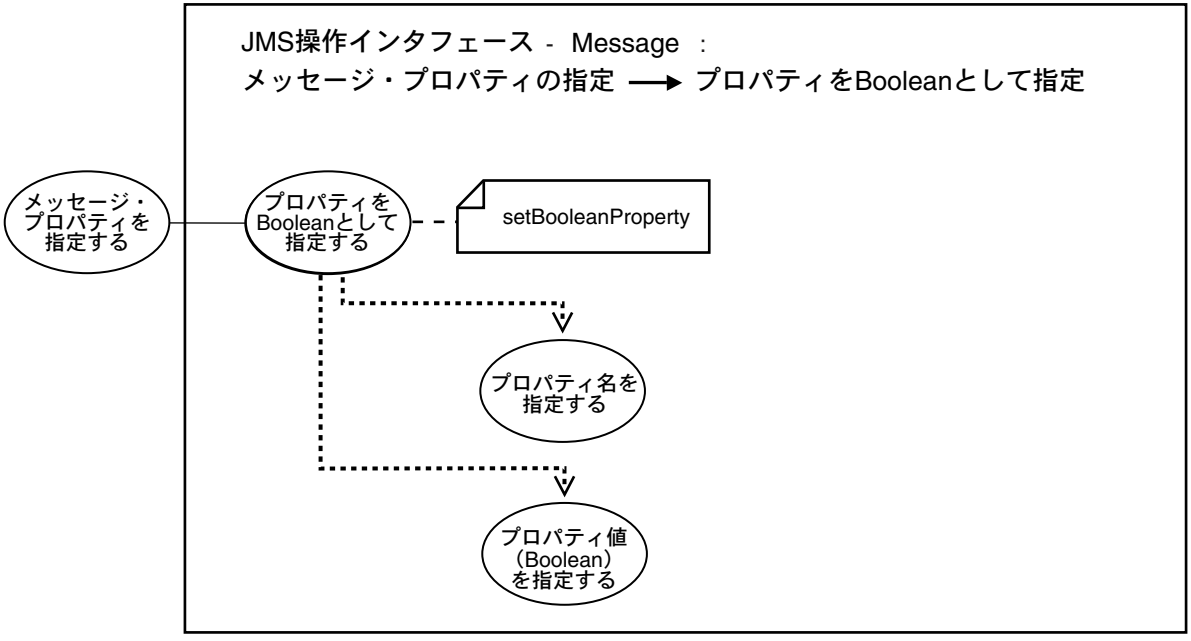
- JMSXAppID（`String`）
- JMSXGroupID（`String`）
- JMSXGroupSeq（`Int`）
- JMS_OracleExcpQ（`String`）- 例外キュー
- JMS_OracleDelay（`Int`）- メッセージ遅延（秒）

次のプロパティは、AdtMessage に対して設定されます。

- JMS_OracleExcpQ（`String`）- 例外キュー - <schema>.queue_name として指定
- JMS_OracleDelay（`Int`）- メッセージ遅延（秒）

JMS メッセージ・プロパティを Boolean として指定

図 16-17 メッセージ・プロパティを Boolean として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

用途

メッセージ・プロパティを `Boolean` として指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setBooleanProperty`

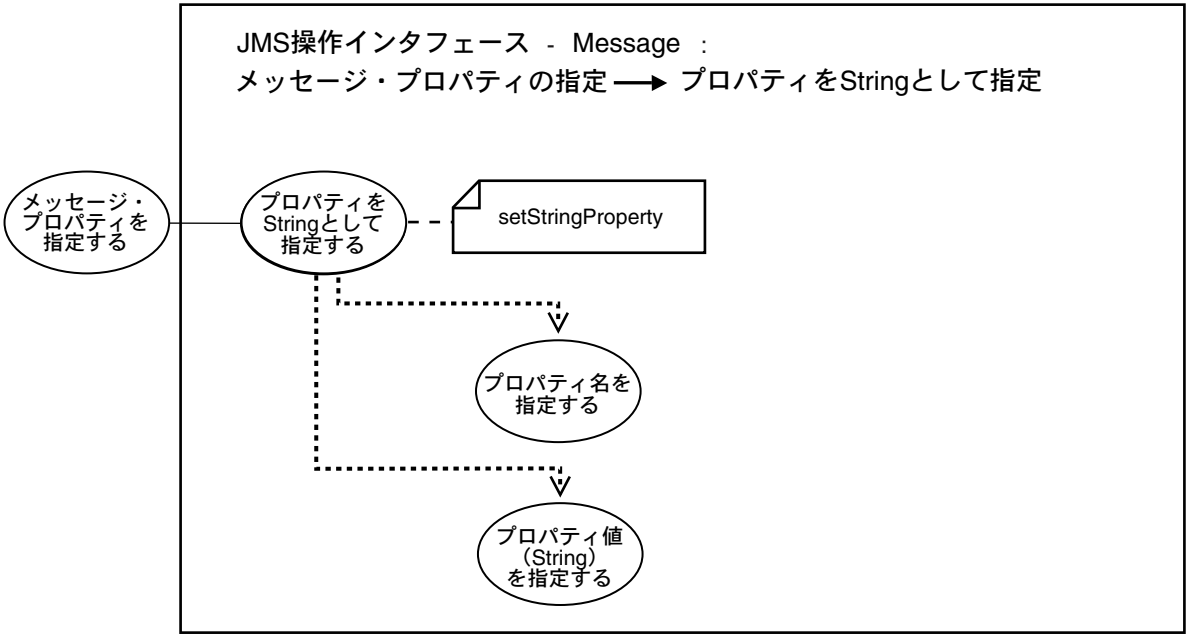
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを String として指定

図 16-18 メッセージ・プロパティを String として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

用途

メッセージ・プロパティを `String` として指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setStringProperty`

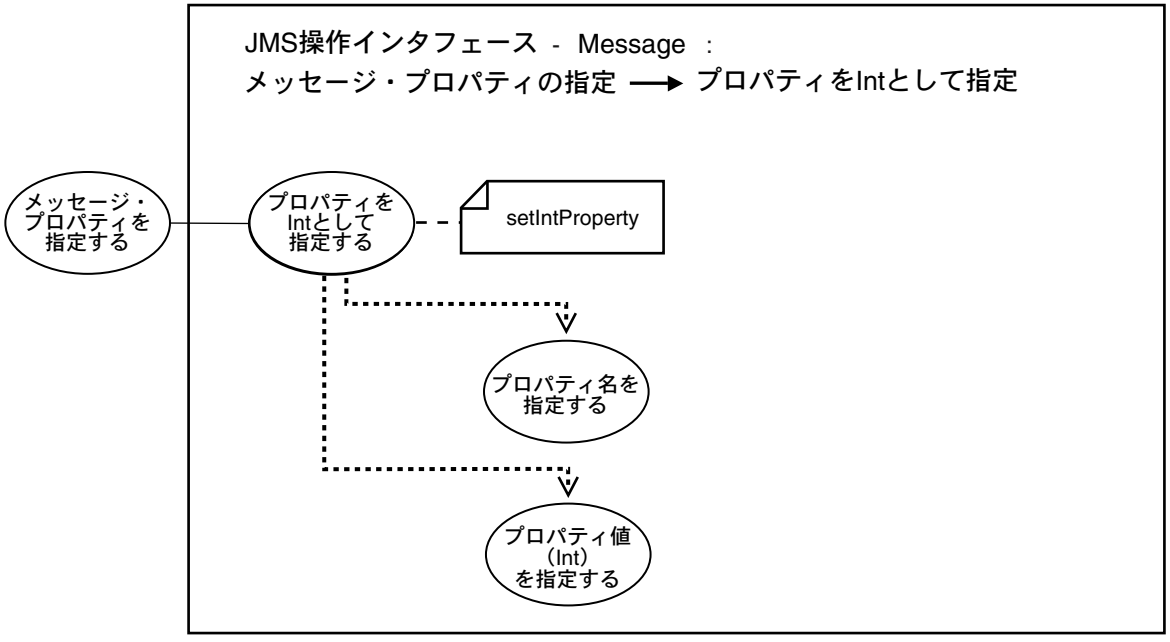
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Int として指定

図 16-19 メッセージ・プロパティを Int として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

用途

メッセージ・プロパティを `Int` として指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setIntProperty`

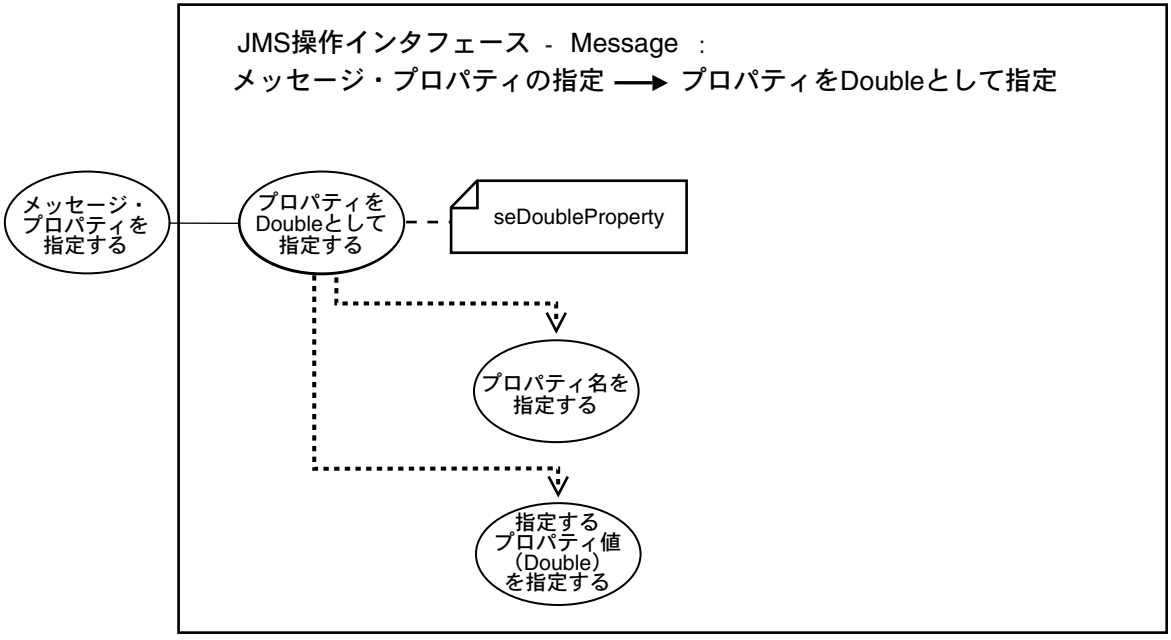
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Double として指定

図 16-20 メッセージ・プロパティを Double として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

用途

メッセージ・プロパティを `Double` として指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setDoubleProperty`

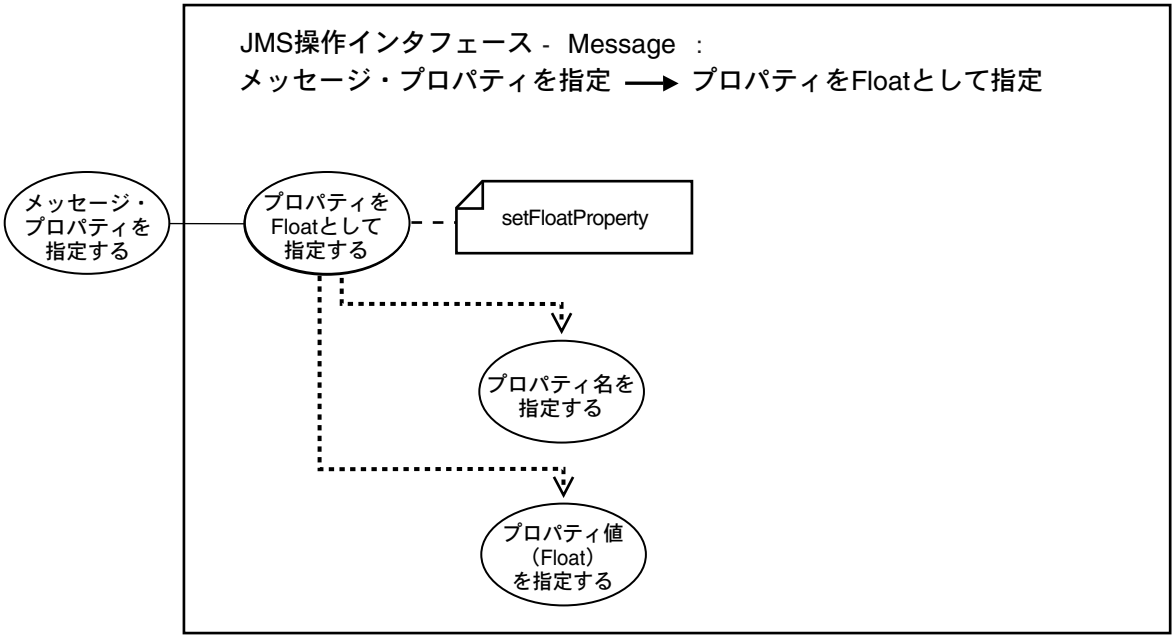
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Float として指定

図 16-21 メッセージ・プロパティを Float として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

用途

メッセージ・プロパティを `Float` として指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setFloatProperty`

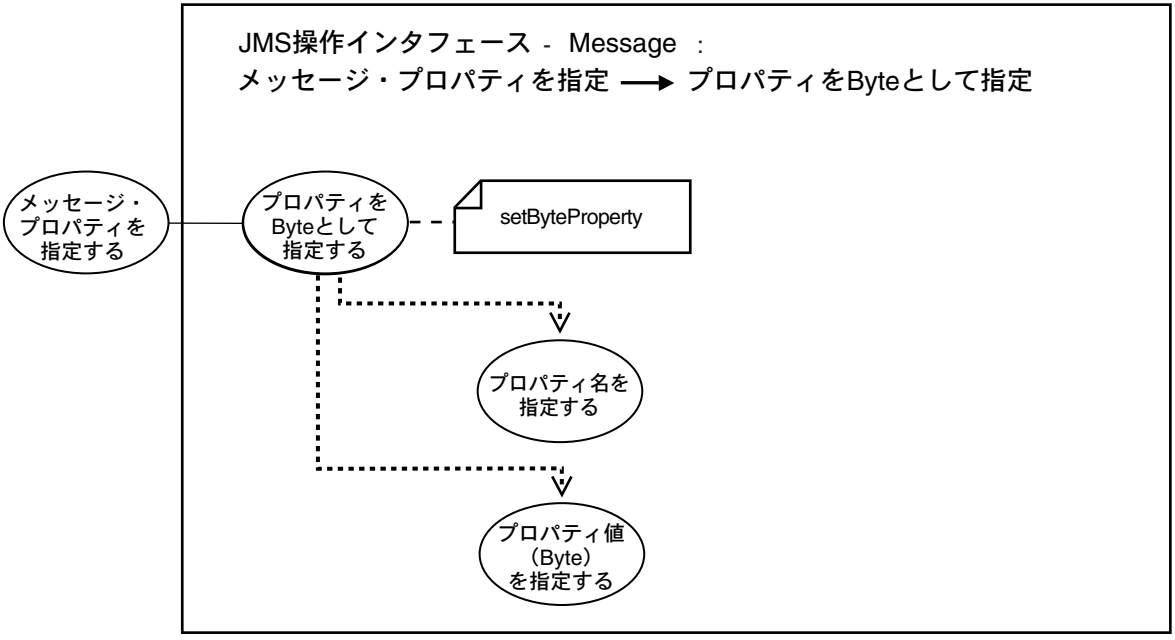
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Byte として指定

図 16-22 メッセージ・プロパティを Byte として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setByteProperty`

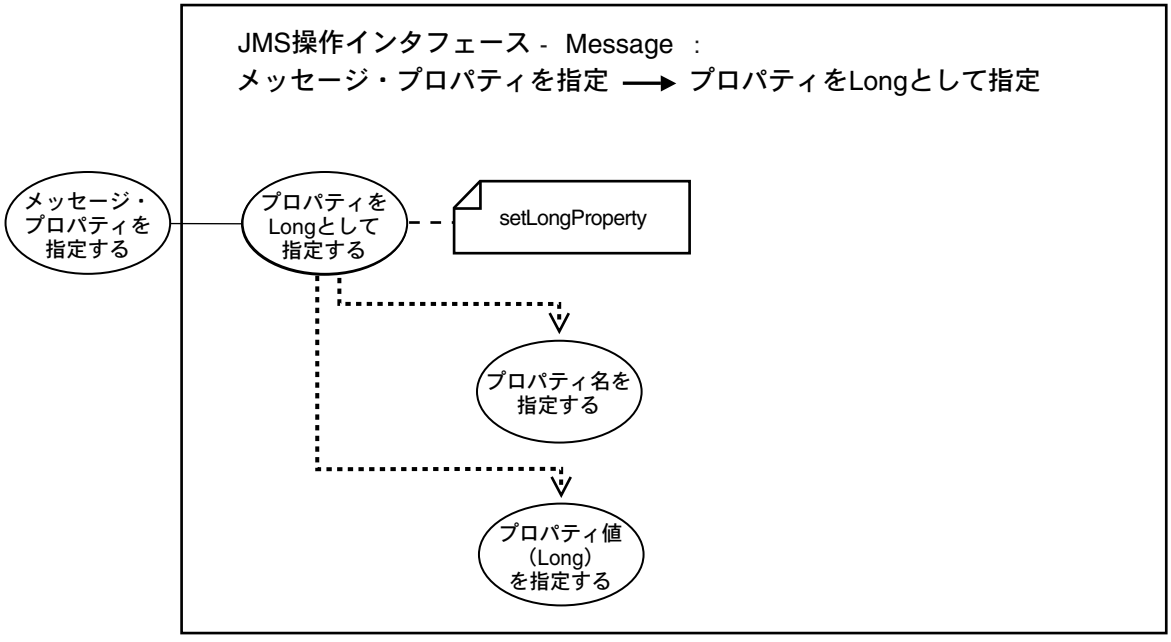
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Long として指定

図 16-23 利用図 : メッセージ・プロパティを Long として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

用途

メッセージ・プロパティを Long として指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setLongProperty`

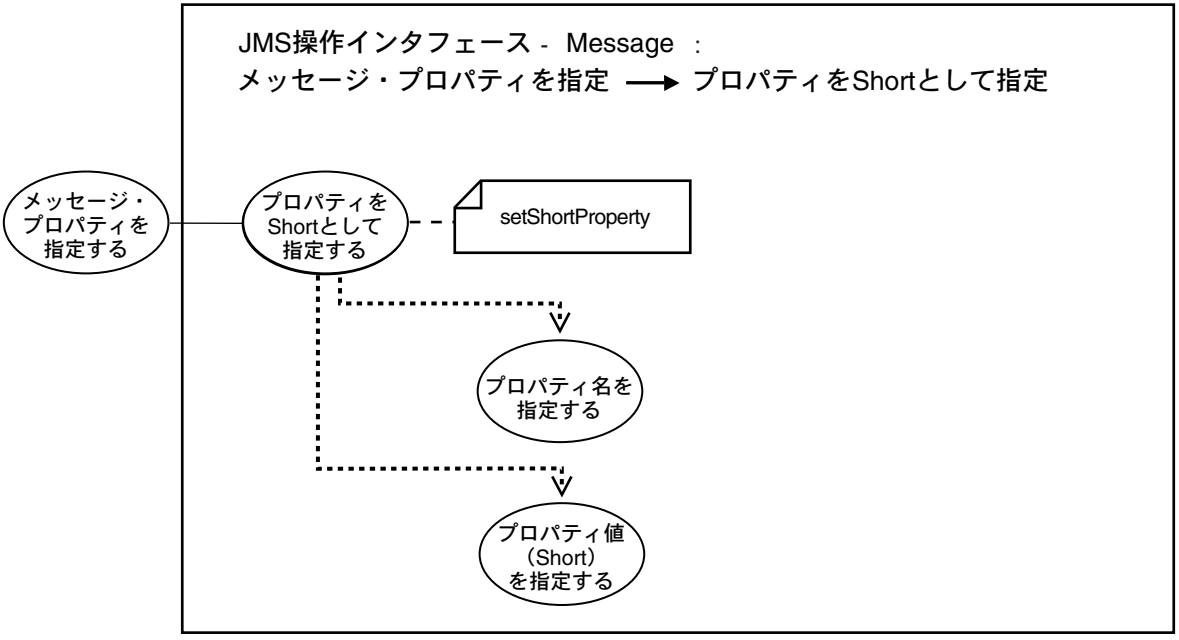
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Short として指定

図 16-24 メッセージ・プロパティを Short として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-61 ページの「JMS メッセージ・プロパティを Object として指定」も参照してください。

用途

メッセージ・プロパティを `Short` として指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setShortProperty`

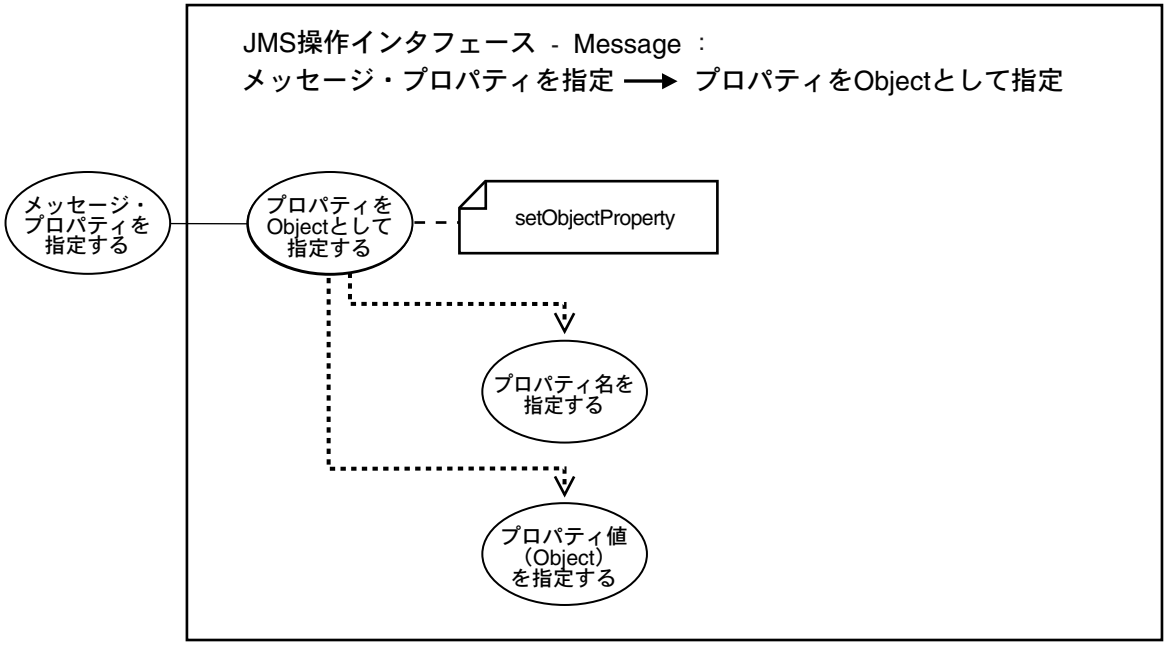
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Object として指定

図 16-25 メッセージ・プロパティを Object として指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-34 ページの「JMS メッセージ・プロパティの指定」も参照してください。
- 16-37 ページの「JMS メッセージ・プロパティを Boolean として指定」も参照してください。
- 16-40 ページの「JMS メッセージ・プロパティを String として指定」も参照してください。
- 16-43 ページの「JMS メッセージ・プロパティを Int として指定」も参照してください。
- 16-46 ページの「JMS メッセージ・プロパティを Double として指定」も参照してください。
- 16-49 ページの「JMS メッセージ・プロパティを Float として指定」も参照してください。
- 16-52 ページの「JMS メッセージ・プロパティを Byte として指定」も参照してください。
- 16-55 ページの「JMS メッセージ・プロパティを Long として指定」も参照してください。
- 16-58 ページの「JMS メッセージ・プロパティを Short として指定」も参照してください。

用途

メッセージ・プロパティを `Object` として指定します。

使用上の注意

Java 基本型の値（`Boolean`、`Byte`、`Short`、`Int`、`Long`、`Float`、`Double` および `String`）のみを含むオブジェクトがサポートされています。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `setObjectProperty`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの Time-To-Live の設定

図 16-26 メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの Time-To-Live の設定

参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-31 ページの「インタフェース - javax.jms.MessageProducer」も参照してください。

用途

メッセージ・プロデューサが送信するすべてのメッセージに対して、デフォルトの Time-To-Live（生存期間）を設定します。

使用上の注意

Time-To-Live は、1000 分の 1 秒単位で指定されます。これは、メッセージが `ready` 状態になった後（メッセージ遅延が有効になった後）に計算されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsProducer」の `setTimeToLive`

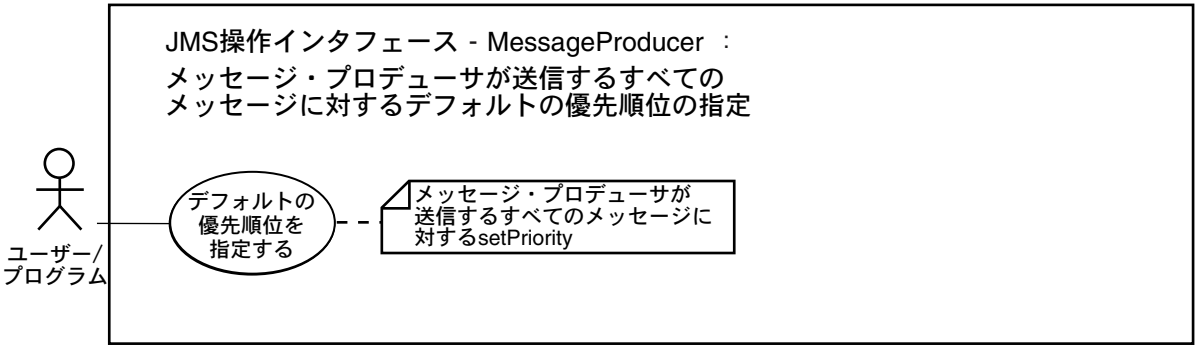
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

```
/* Set default timeToLive value to 100000 milliseconds for all messages sent by the
QueueSender*/
QueueSender sender;
sender.setTimeToLive(100000);
```

メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの優先順位の設定

図 16-27 メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの優先順位の設定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-31 ページの「インタフェース - javax.jms.MessageProducer」も参照してください。

用途

メッセージ・プロデューサが送信するすべてのメッセージに対して、デフォルトの優先順位の設定をします。

使用上の注意

優先順位の値には、どの整数値でも指定できます。数値が小さいほど、優先順位が高いことを表します。

優先順位の値が、送信操作中に明示的に指定されている場合は、このメソッドで設定されたプロデューサのデフォルト値をオーバーライドします。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsProducer」の `setPriority`

例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

例 1

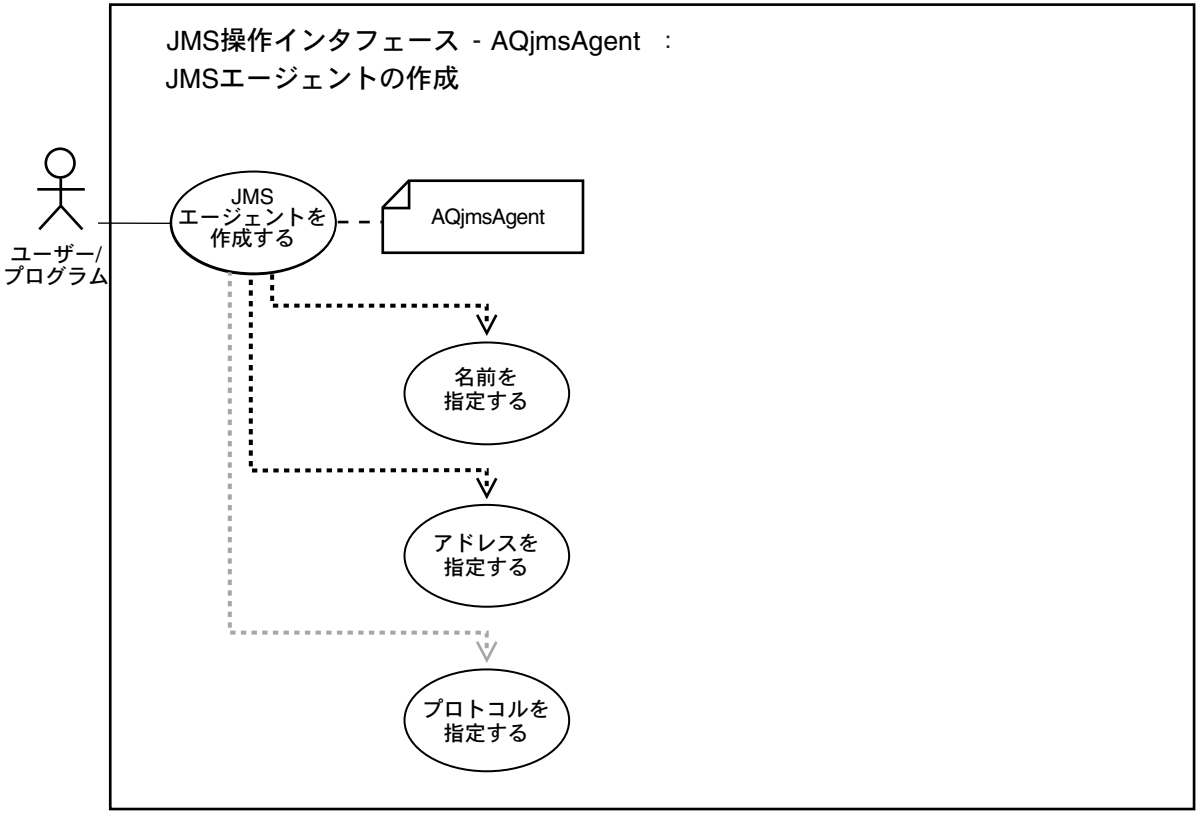
```
/* Set default priority value to 2 for all messages sent by the QueueSender*/
QueueSender sender;
sender.setPriority(2);
```

例 2

```
/* Set default priority value to 2 for all messages sent by the TopicPublisher*/
TopicPublisher publisher;
publisher.setPriority(1);
```

AQjms エージェントの作成

図 16-28 AQjms エージェントの作成



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-46 ページの「クラス - oracle.jms.AQjmsAgent」も参照してください。

用途

AQjms エージェントを作成します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsAgent」

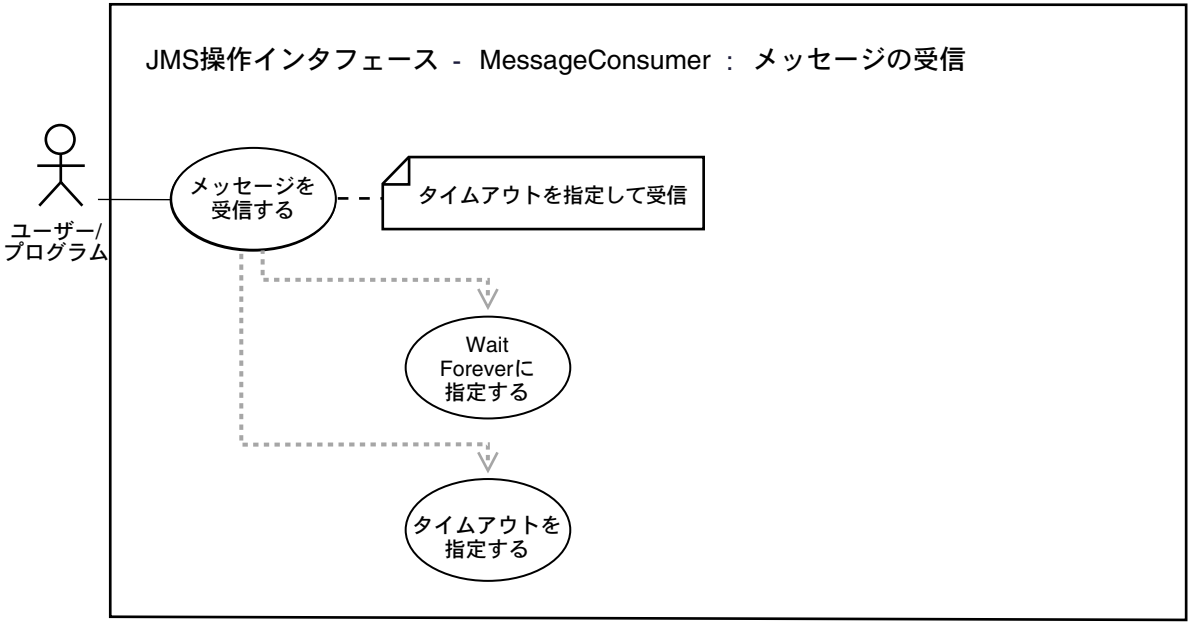
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

メッセージ・コンシューマを使用したメッセージの同期受信 : タイムアウトを指定

図 16-29 メッセージ・コンシューマを使用したメッセージの受信 : タイムアウトを指定



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-30 ページの「インタフェース - javax.jms.MessageConsumer」も参照してください。
- 16-72 ページの「メッセージ・コンシューマを使用したメッセージの同期受信 : 待機なし」も参照してください。

用途

タイムアウトを指定し、メッセージ・コンシューマを使用してメッセージを受信します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 2 章「パッケージ oracle.jms」の「AQjmsConsumer」の `receive`

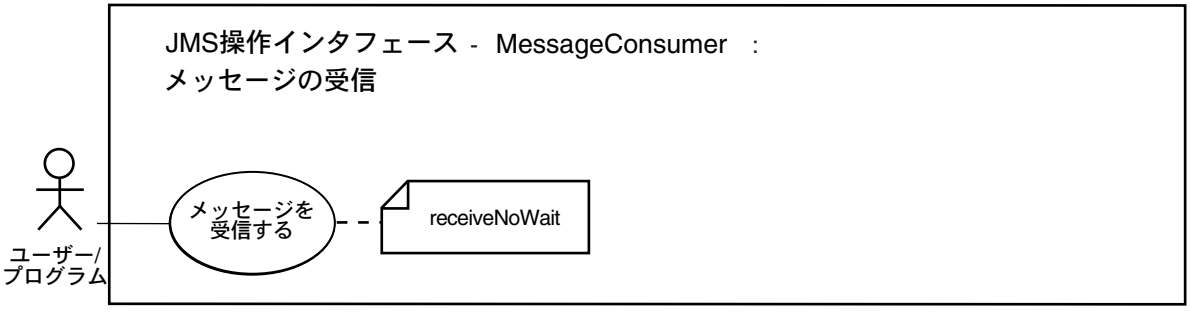
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
Topic                     shipped_orders;
int                       myport = 5521;

/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");

t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("WS",
"Shipped_Orders_Topic");

/* create a subscriber, specifying the correct CustomDatumFactory  and
selector */
subscriber1 = jms_sess.createDurableSubscriber(shipped_orders,
'WesternShipping',
             " priority > 1 and tab.user_data.region like 'WESTERN %'",
              false,AQjmsAgent.getFactory());

/* receive, blocking for 30 seconds if there were no messages */
Message = subscriber.receive(30000);
```

メッセージ・コンシューマを使用したメッセージの同期受信 : 待機なし

図 16-30 メッセージ・コンシューマを使用したメッセージの受信 : 待機なし



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-30 ページの「インタフェース - javax.jms.MessageConsumer」も参照してください。
- 16-70 ページの「メッセージ・コンシューマを使用したメッセージの同期受信 : タイムアウトを指定」も参照してください。

用途

待機なしで、メッセージ・コンシューマを使用してメッセージを受信します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConsumer」の `receiveNoWait`

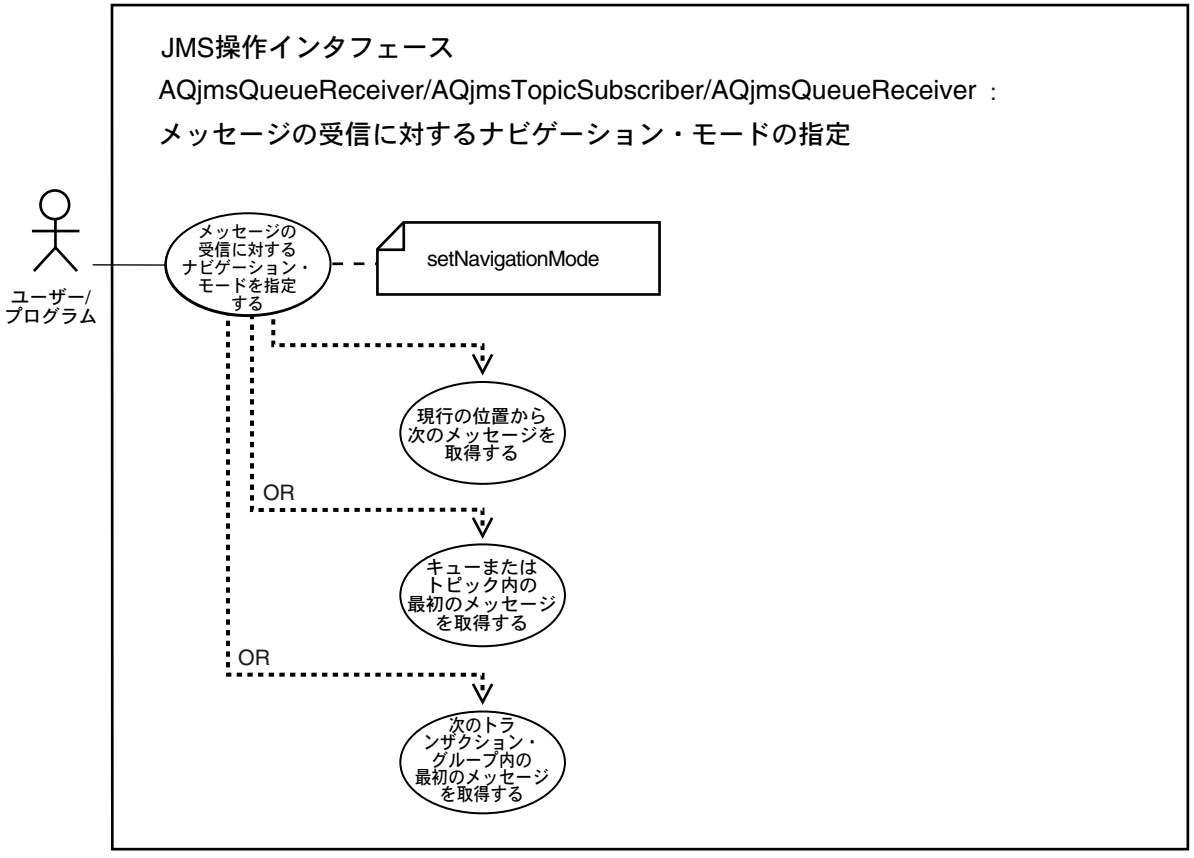
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

メッセージの受信に対するナビゲーション・モードの指定

図 16-31 メッセージの受信に対するナビゲーション・モードの指定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-44 ページの「インタフェース - oracle.jms.AQjmsQueueReceiver」も参照してください。
- B-45 ページの「インタフェース - oracle.jms.AQjmsTopicSubscriber」も参照してください。
- B-46 ページの「インタフェース - oracle.jms.AQjmsTopicReceiver」も参照してください。

用途

メッセージの受信に対して、ナビゲーション・モードを指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsQueueReceiver」の `setNavigationMode`、「AQjmsTopicReceiver」の `setNavigationMode`、「AQjmsTopicSubscriber」の `setNavigationMode`

例

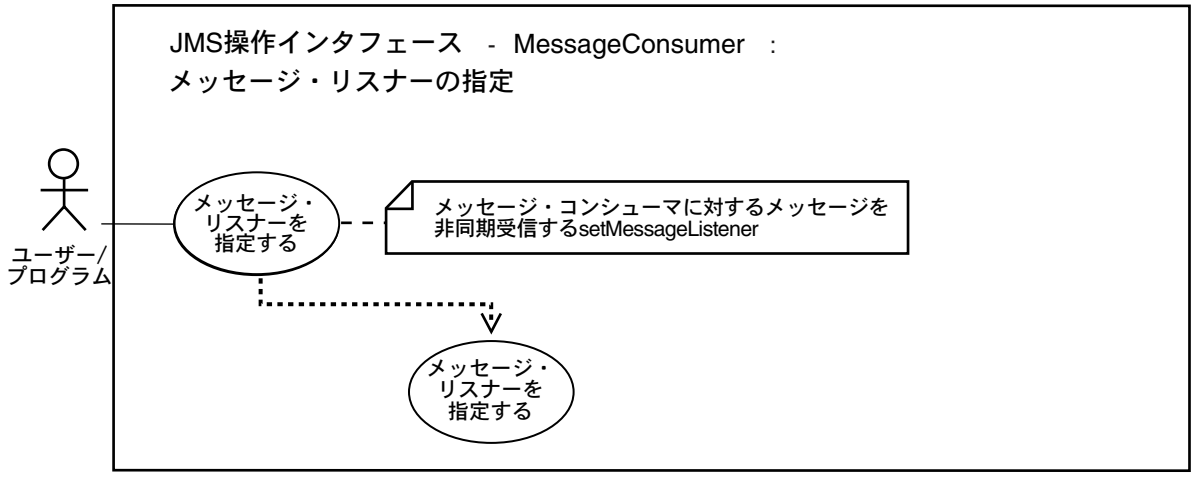
各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
Topic                     shipped_orders;
int                       myport = 5521;

/* create connection and session */

tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
```

```
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("WS",
"Shipped_Orders_Topic");
/* create a subscriber, specifying the correct CustomDatumFactory  and
selector */

subscriber1 = jms_sess.createDurableSubscriber(shipped_orders,
'WesternShipping',
                " priority > 1 and tab.user_data.region like 'WESTERN %'",
                  false,AQjmsAgent.getFactory());

subscriber1.setNavigationMode(AQjmsConstants.NAVIGATION_FIRST_MESSAGE);

/* get message for the subscriber, returning immediately if there was no
message */
Message = subscriber.receive();
```

メッセージを非同期受信するメッセージ・リスナーの指定 : メッセージ・コンシューマ

図 16-32 メッセージ・コンシューマに対するメッセージ・リスナーの指定



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-30 ページの「インタフェース - javax.jms.MessageConsumer」も参照してください。
- 16-80 ページの「メッセージを非同期受信するメッセージ・リスナーの指定 : セッション」も参照してください。

用途

メッセージ・コンシューマに対するメッセージ・リスナーを指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConsumer」の `setMessageListener`

例

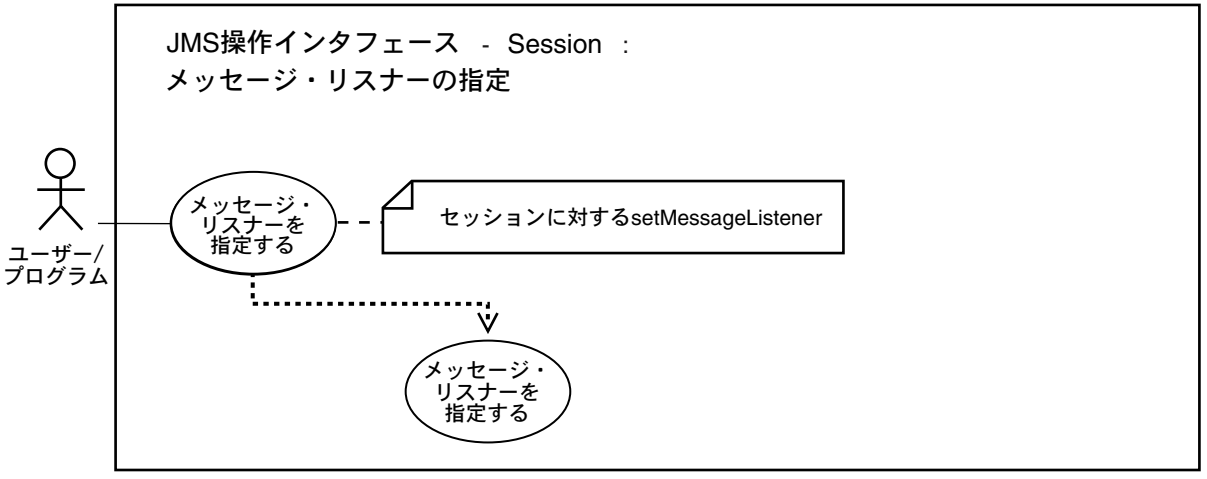
各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

```
TopicConnectionFactory    tc_fact   = null;
TopicConnection           t_conn    = null;
TopicSession              t_sess    = null;
TopicSession              jms_sess;
Topic                     shipped_orders;
int                       myport = 5521;
MessageListener           mLis = null;
/* create connection and session */
tc_fact = AQjmsFactory.getTopicConnectionFactory("MYHOSTNAME",
                                                 "MYSID", myport, "oci8");
t_conn = tc_fact.createTopicConnection("jmstopic", "jmstopic");
jms_sess = t_conn.createTopicSession(true, Session.CLIENT_ACKNOWLEDGE);

shipped_orders = ((AQjmsSession )jms_sess).getTopic("WS",
"Shipped_Orders_Topic");
/* create a subscriber, specifying the correct CustomDatumFactory  and
selector */
subscriber1 = jms_sess.createDurableSubscriber(shipped_orders,
'WesternShipping',
        " priority > 1 and tab.user_data.region like 'WESTERN %'",
         false,AQjmsAgent.getFactory());

mLis = new myListener(jms_sess, "foo");
/* get message for the subscriber, returning immediately if there was no
message */
subscriber.setMessageListener(mLis);

The definition of the myListener class
import oracle.AQ.*;
import oracle.jms.*;
import javax.jms.*;
import java.lang.*;
import java.util.*;
public class myListener implements MessageListener
```

```
{
   TopicSession   mySess;
   String         myName;

   /* constructor */
   myListener(TopicSession t_sess, String t_name)
   {
      mySess = t_sess;
      myName = t_name;
   }

   public onMessage(Message m)
   {
      System.out.println("Retrieved message with correlation: " ||
m.getJMSCorrelationID());

      try{
        /* commit the dequeue */
        mySession.commit();
      } catch (java.sql.SQLException e)
      {System.out.println("SQL Exception on commit"); }

   }

}
```

メッセージを非同期受信するメッセージ・リスナーの指定 : セッション

図 16-33 セッションに対するメッセージ・リスナーの指定



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。
- 16-77 ページの「メッセージを非同期受信するメッセージ・リスナーの指定 : メッセージ・コンシューマ」も参照してください。

用途

セッションに対するメッセージ・リスナーを指定します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `setMessageListener`

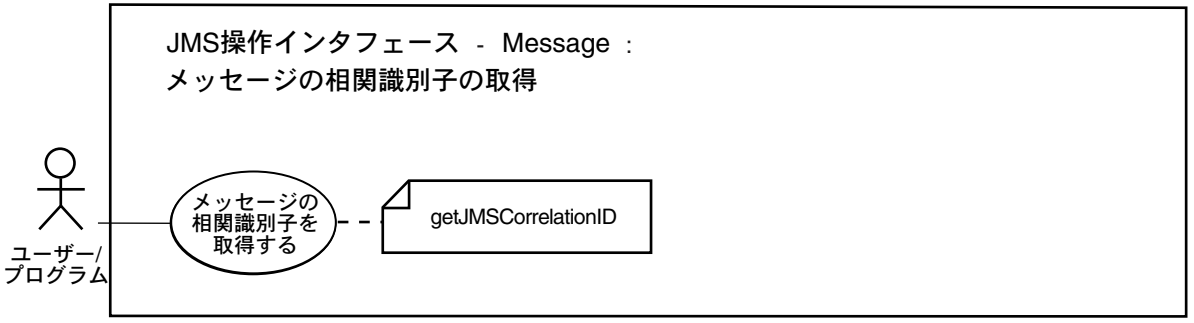
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

メッセージの相関識別子の取得

図 16-34 メッセージの相関識別子の取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。

用途

メッセージの相関識別子を取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getJMSCorrelationID`

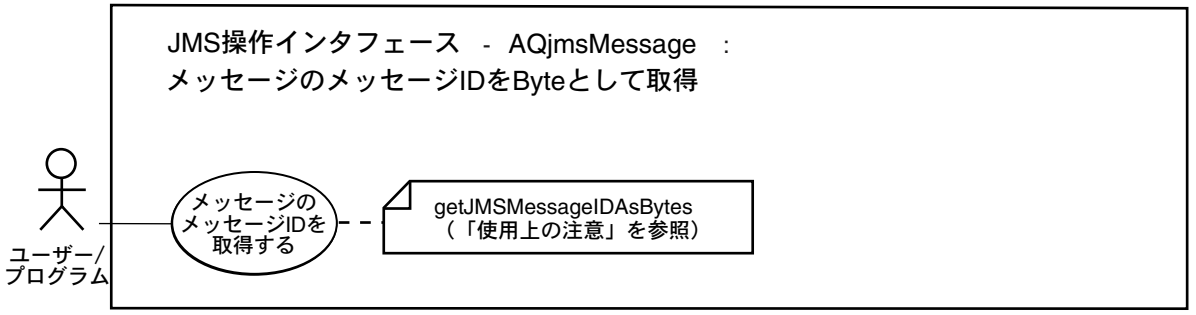
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

メッセージのメッセージ ID を Byte として取得

図 16-35 メッセージのメッセージ ID を Byte として取得



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-52 ページの「クラス - oracle.jms.AQjmsMessage」も参照してください。
- 16-85 ページの「メッセージのメッセージ ID を String として取得」も参照してください。

用途

メッセージのメッセージ ID を `Byte` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getJMSMessageID`

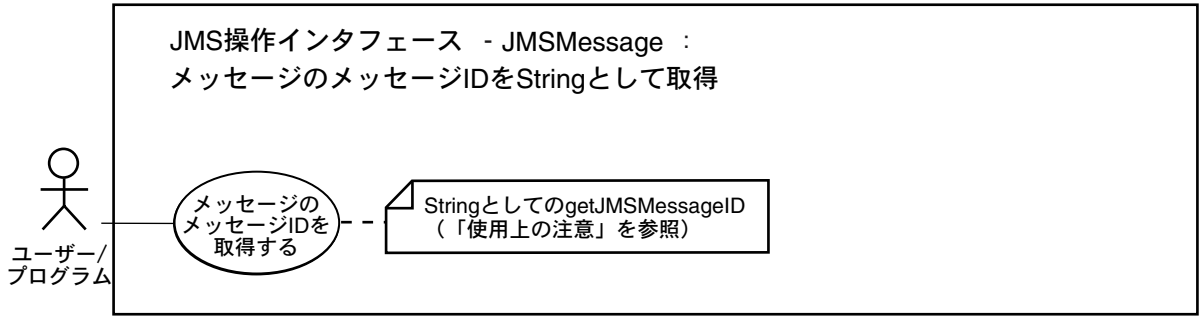
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

メッセージのメッセージ ID を String として取得

図 16-36 メッセージのメッセージ ID を String として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-83 ページの「メッセージのメッセージ ID を Byte として取得」も参照してください。

用途

メッセージのメッセージ ID を `String` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getJMSMessageID`

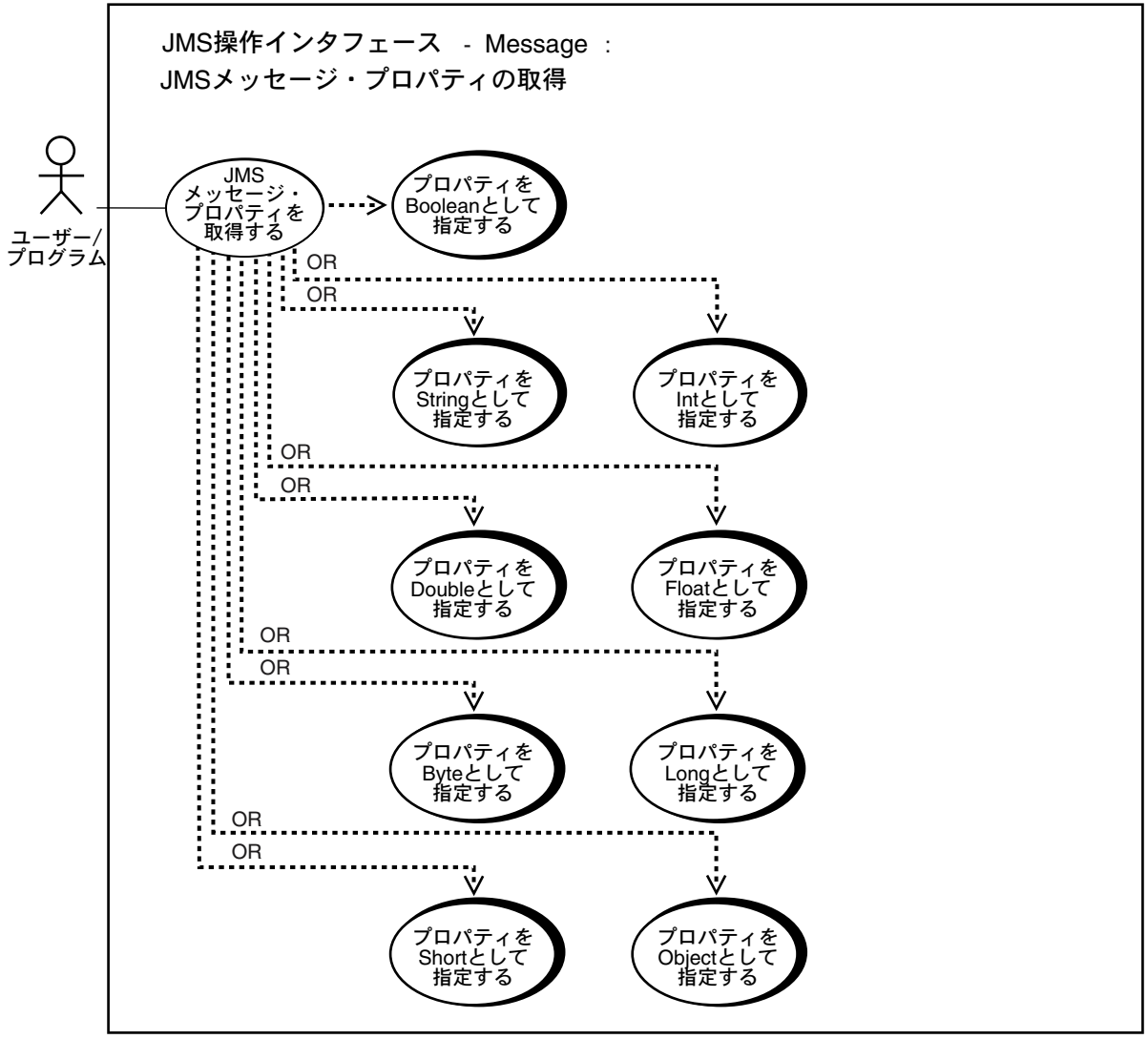
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティの取得

図 16-37 JMS メッセージ・プロパティの取得

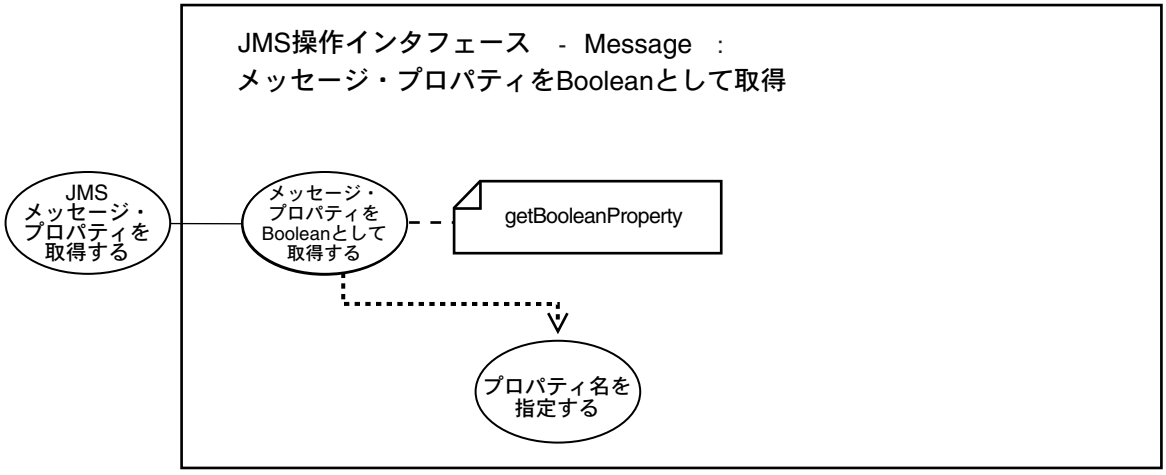


参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

JMS メッセージ・プロパティを Boolean として取得

図 16-38 メッセージ・プロパティを Boolean として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

用途

メッセージ・プロパティを `Boolean` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getBooleanProperty`

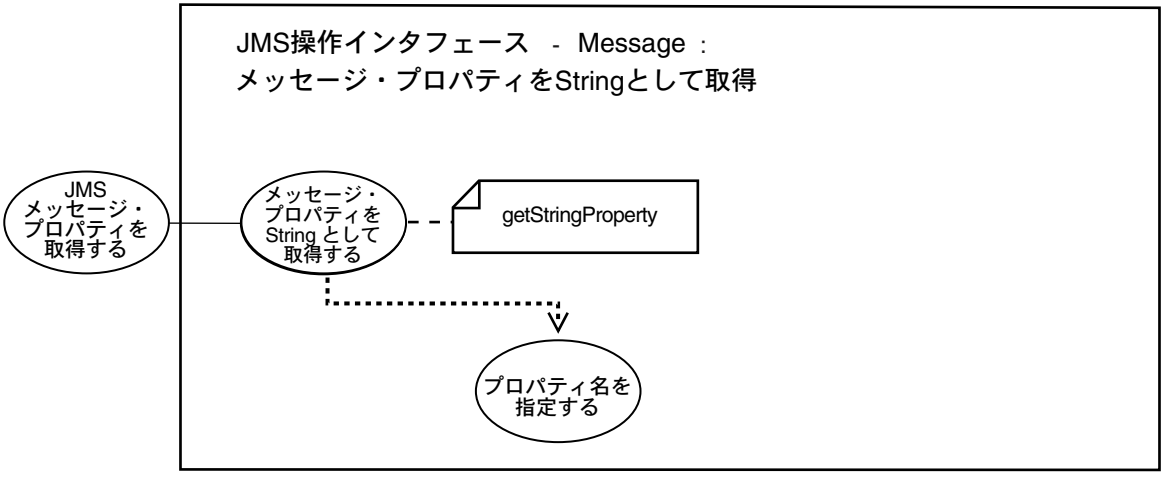
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを String として取得

図 16-39 メッセージ・プロパティを String として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

用途

メッセージ・プロパティを `String` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getStringProperty`

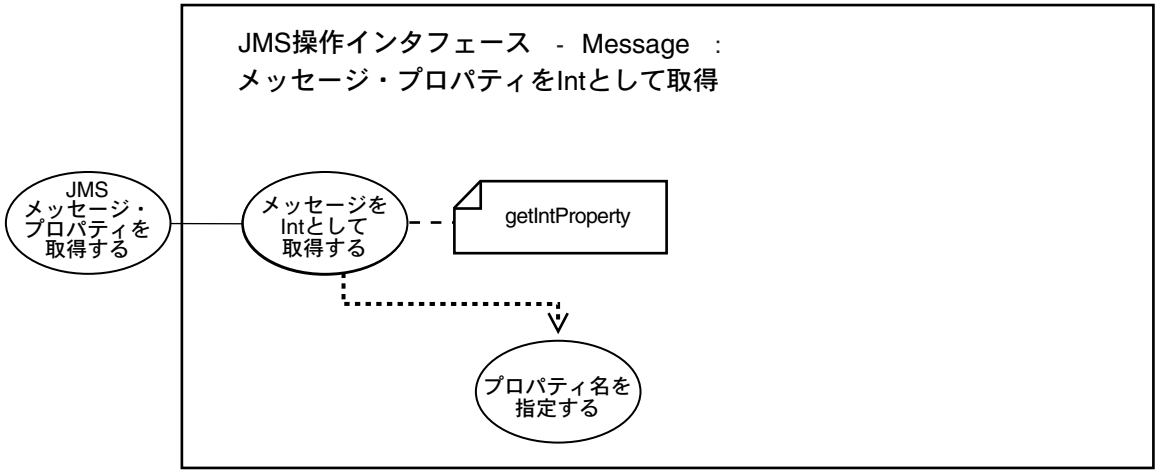
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

```
TextMessage message;

message.setStringProperty("JMS_OracleExcpQ", "scott.text_ecxcp_queue"); /*set
exception queue for message*/

message.setStringProperty("color", "red"); /*set user-defined property - color */
```

JMS メッセージ・プロパティを Int として取得

図 16-40 メッセージ・プロパティを Int として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

用途

メッセージ・プロパティを `Int` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getIntProperty`

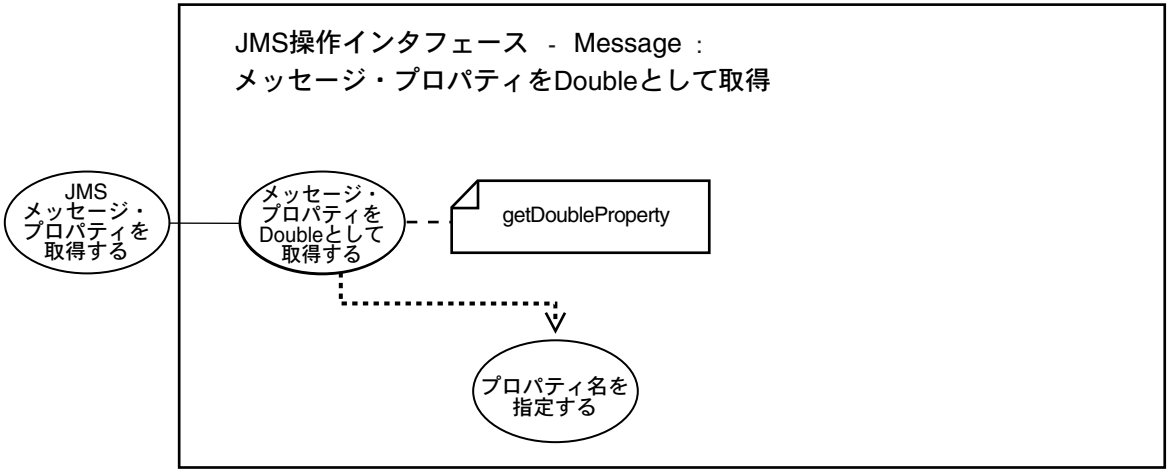
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

```
StreamMessage message;
message.setIntProperty("MMS_OracleDelay", 10); /*set message delay to 10 seconds*/

message.setIntProperty("empid", 1000); /*set user-defined property - empId*/
```

JMS メッセージ・プロパティを Double として取得

図 16-41 メッセージ・プロパティを Double として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

用途

メッセージ・プロパティを `Double` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getDoubleProperty`

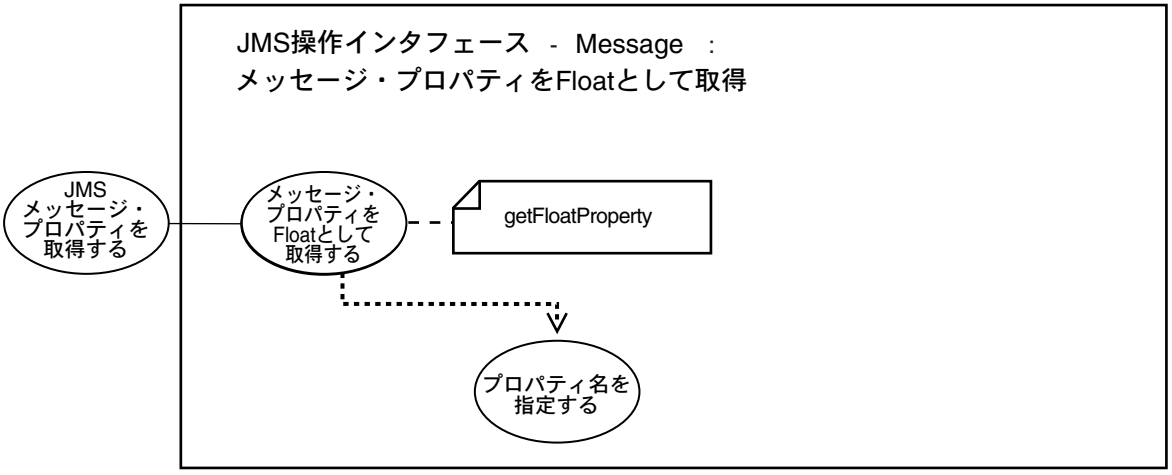
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Float として取得

図 16-42 メッセージ・プロパティを Float として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

用途

メッセージ・プロパティを `Float` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getFloatProperty`

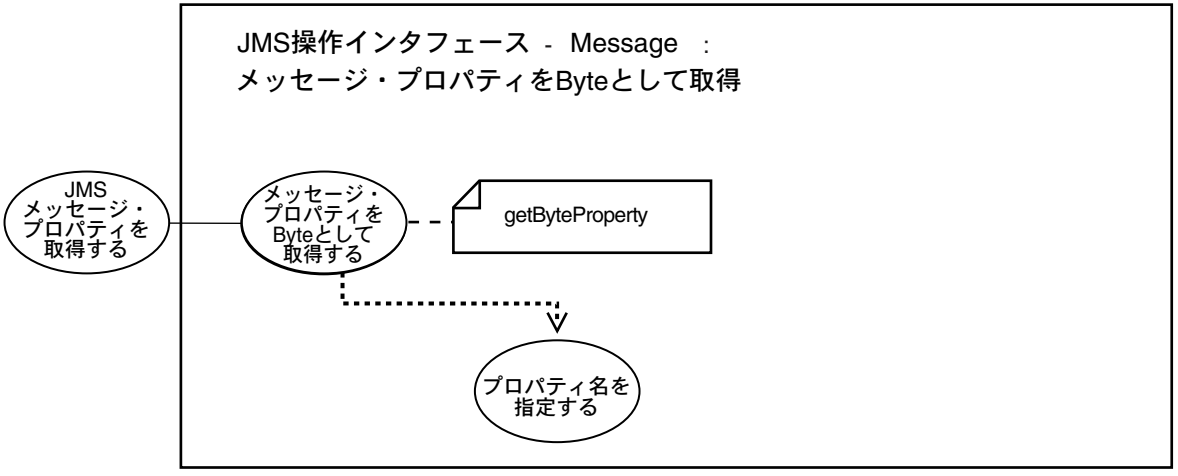
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Byte として取得

図 16-43 メッセージ・プロパティを Byte として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

用途

メッセージ・プロパティを `Byte` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getByteProperty`

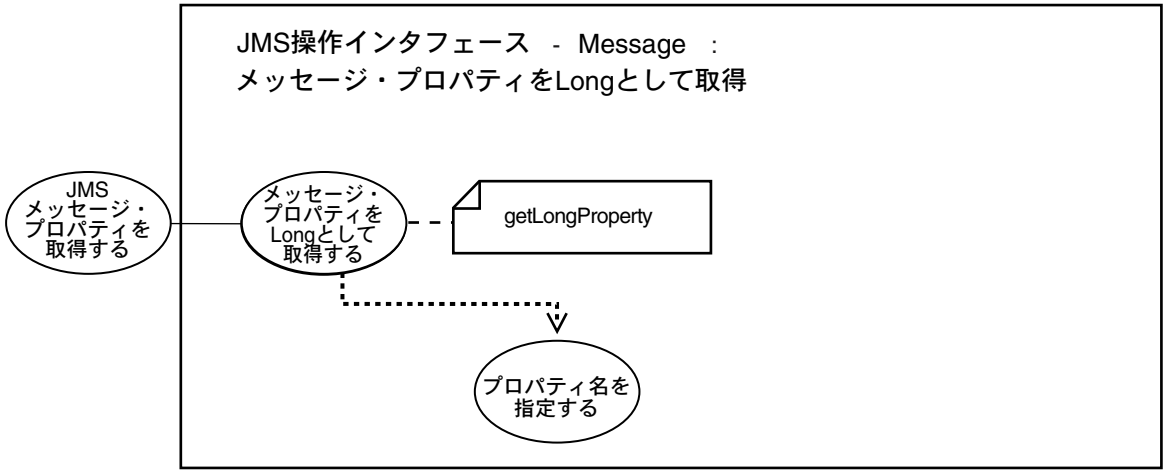
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Long として取得

図 16-44 メッセージ・プロパティを Long として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

用途

メッセージ・プロパティを Long として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getLongProperty`

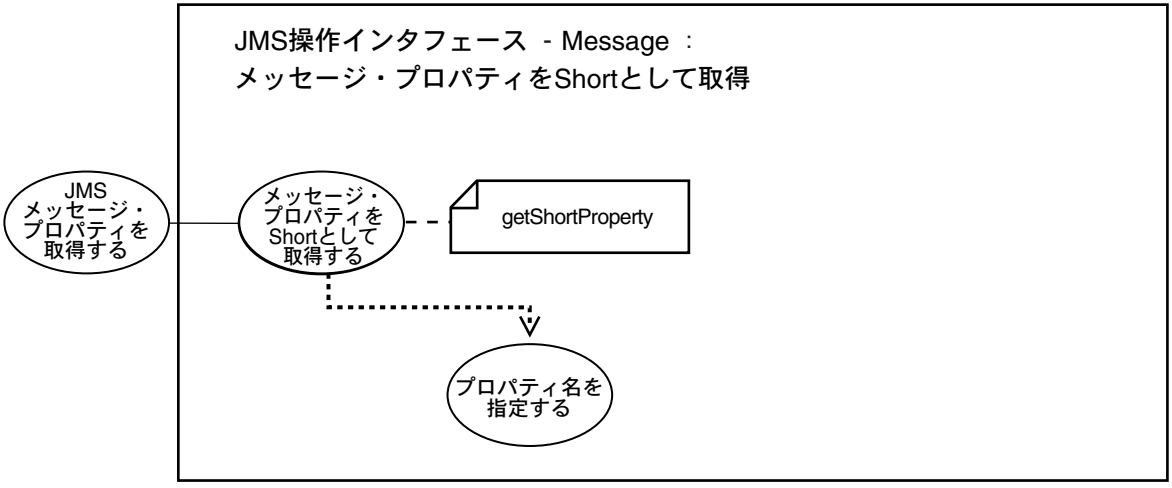
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Short として取得

図 16-45 メッセージ・プロパティを Short として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-113 ページの「JMS メッセージ・プロパティを Object として取得」も参照してください。

用途

メッセージ・プロパティを `Short` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getShortProperty`

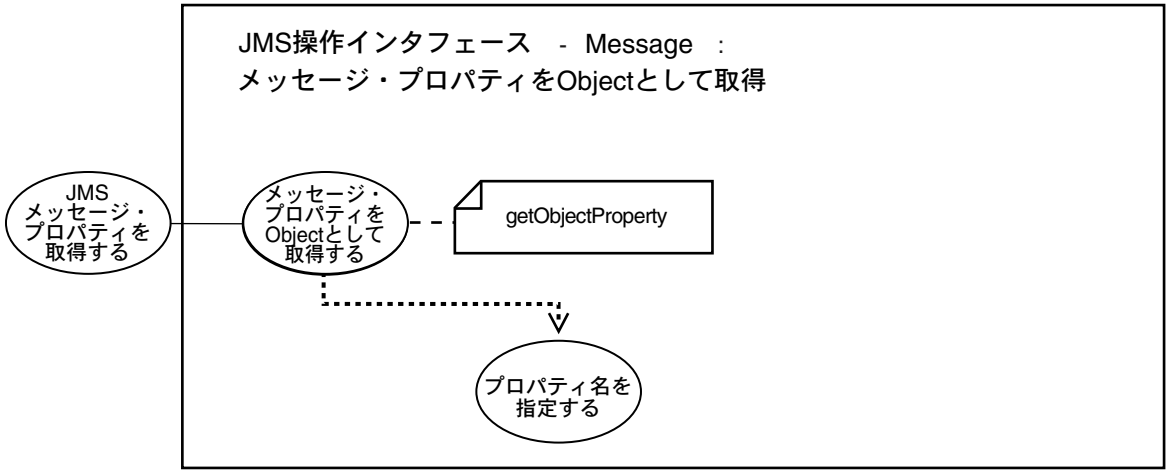
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS メッセージ・プロパティを Object として取得

図 16-46 メッセージ・プロパティを Object として取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-28 ページの「インタフェース - javax.jms.Message」も参照してください。
- 16-87 ページの「JMS メッセージ・プロパティの取得」も参照してください。
- 16-89 ページの「JMS メッセージ・プロパティを Boolean として取得」も参照してください。
- 16-92 ページの「JMS メッセージ・プロパティを String として取得」も参照してください。
- 16-95 ページの「JMS メッセージ・プロパティを Int として取得」も参照してください。
- 16-98 ページの「JMS メッセージ・プロパティを Double として取得」も参照してください。
- 16-101 ページの「JMS メッセージ・プロパティを Float として取得」も参照してください。
- 16-104 ページの「JMS メッセージ・プロパティを Byte として取得」も参照してください。
- 16-107 ページの「JMS メッセージ・プロパティを Long として取得」も参照してください。
- 16-110 ページの「JMS メッセージ・プロパティを Short として取得」も参照してください。

用途

メッセージ・プロパティを `Object` として取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsMessage」の `getObjectProperty`

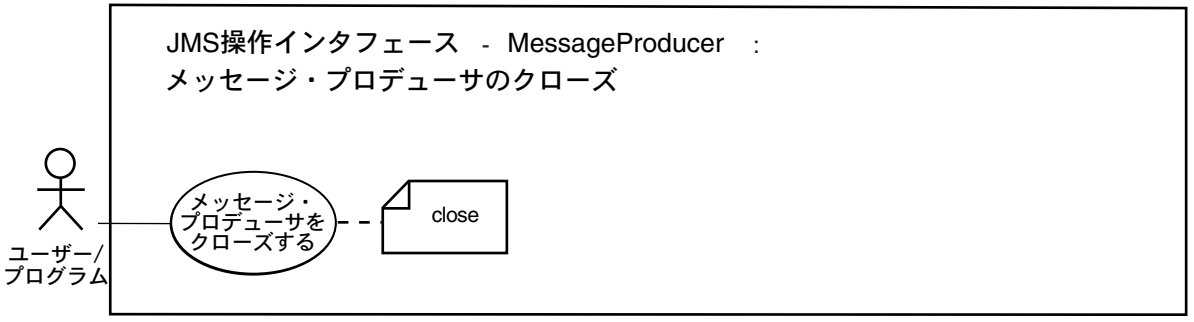
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

```
TextMessage message;
message.setObjectProperty("empid", new Integer(1000);
```

メッセージ・プロデューサのクローズ

図 16-47 メッセージ・プロデューサのクローズ



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-31 ページの「インタフェース - javax.jms.MessageProducer」も参照してください。

用途

メッセージ・プロデューサをクローズします。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsProducer」の `close`

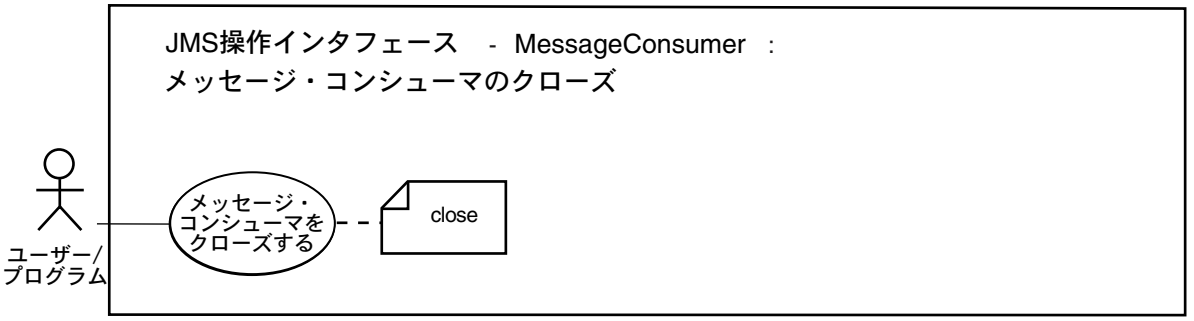
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

メッセージ・コンシューマのクローズ

図 16-48 メッセージ・コンシューマのクローズ



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-30 ページの「インタフェース - javax.jms.MessageConsumer」も参照してください。

用途

メッセージ・コンシューマをクローズします。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConsumer」の `close`

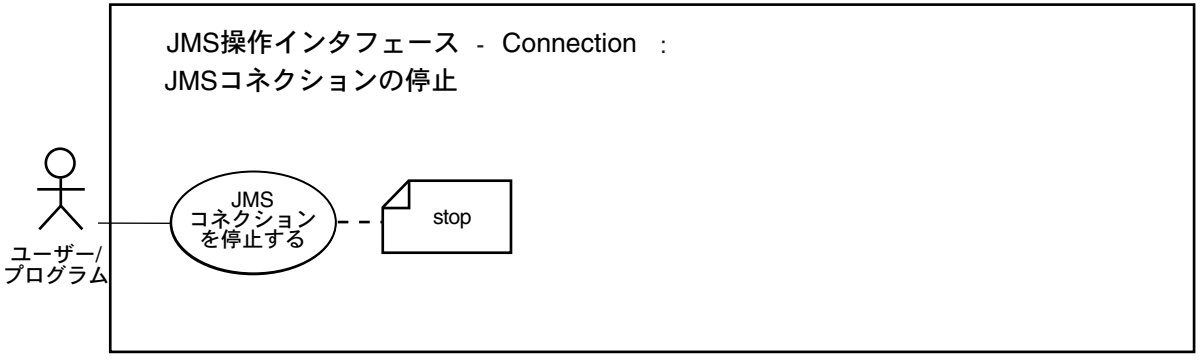
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS コネクションの停止

図 16-49 JMS コネクションの停止



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-25 ページの「インタフェース - javax.jms.Connection」も参照してください。

用途

JMS コネクションを停止します。

使用上の注意

このメソッドは、受信メッセージのコネクション配信を一時的に停止するために使用します。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `stop`

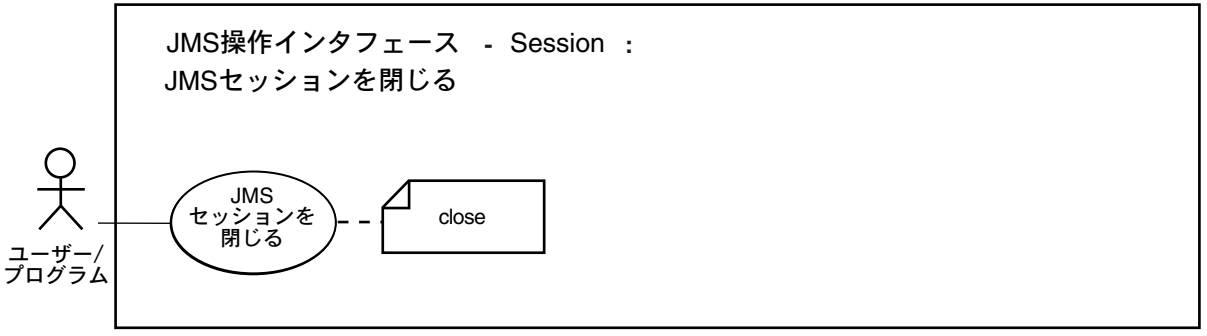
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS セッションを閉じる

図 16-50 JMS セッションを閉じる



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-35 ページの「インタフェース - javax.jms.Session」も参照してください。

用途

JMS セッションを閉じます。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsSession」の `close`

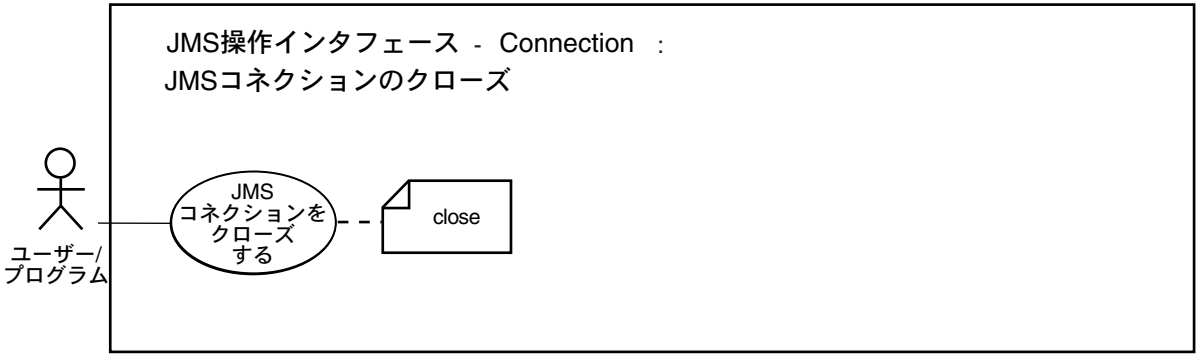
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS コネクションのクローズ

図 16-51 JMS コネクションのクローズ



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-25 ページの「インタフェース - javax.jms.Connection」も参照してください。

用途

JMS コネクションをクローズします。

使用上の注意

このメソッドはコネクションをクローズし、コネクションのために割り当てられているすべてのリソースを解放します。通常、JMS のプロバイダは重要なリソースをコネクションのために Java VM の外部に割り当てるため、クライアントは、これらのリソースが必要ない場合はクローズする必要があります。ガベージ・コレクションによる最終的なリソースの解放を待つ必要はありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `close`

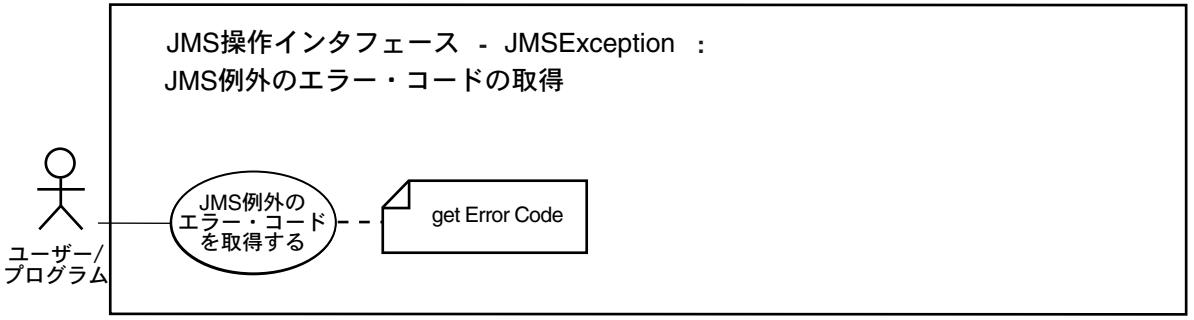
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS 例外のエラー・コードの取得

図 16-52 JMS 例外のエラー・コードの取得



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-42 ページの「例外 - javax.jms.JMSException」も参照してください。

用途

JMS 例外のエラー・コードを取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsException」の `getErrorCode`

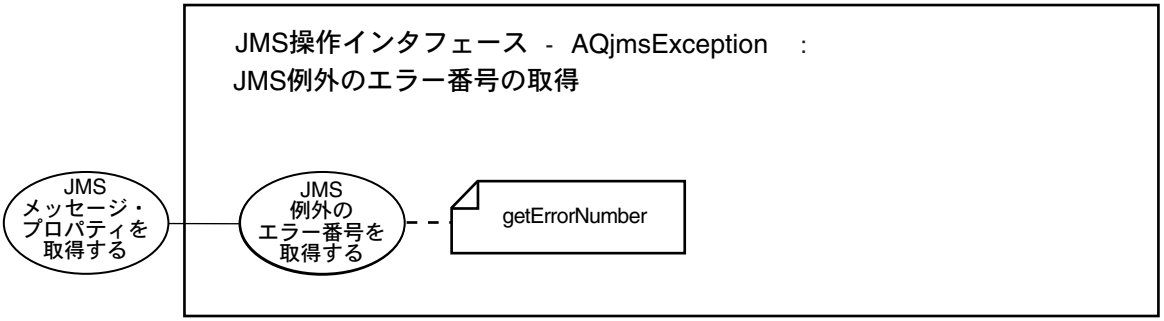
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS 例外のエラー番号の取得

図 16-53 JMS 例外のエラー番号の取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-58 ページの「例外 - oracle.jms.AQjmsException」も参照してください。

用途

JMS 例外のエラー番号を取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsException」の `getErrorNumber`

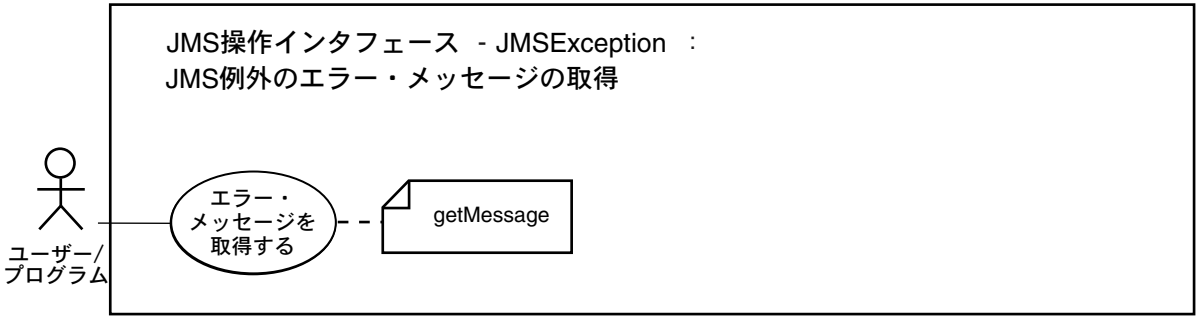
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS 例外のエラー・メッセージの取得

図 16-54 JMS 例外のエラー・メッセージの取得



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-42 ページの「例外 - javax.jms.JMSException」も参照してください。

用途

JMS 例外のエラー・メッセージを取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsException」の `getMessage`

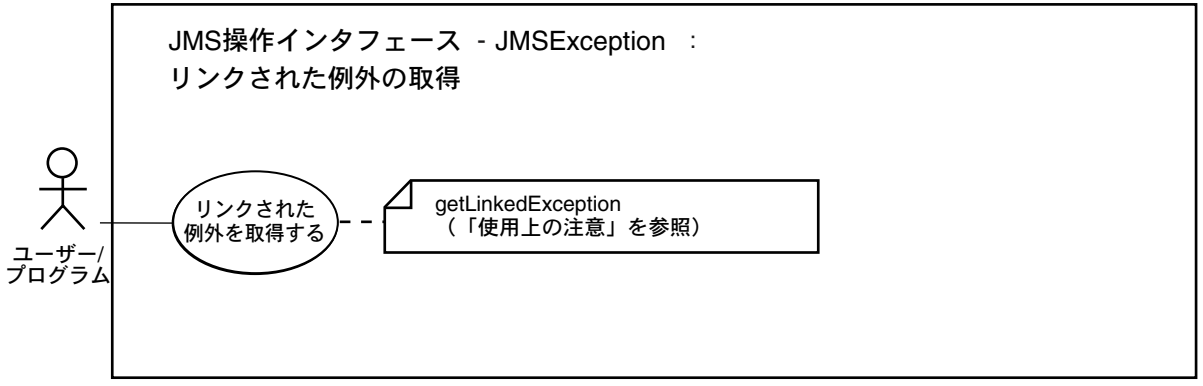
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS 例外にリンクされた例外の取得

図 16-55 JMS 例外にリンクされた例外の取得



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-42 ページの「例外 - javax.jms.JMSException」も参照してください。

用途

JMS 例外にリンクされた例外を取得します。

使用上の注意

このメソッドは、この JMS 例外にリンクされた例外を取得するために使用します。一般に、これにはデータベースが呼び出す SQL 例外が含まれます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsException」の `getLinkedException`

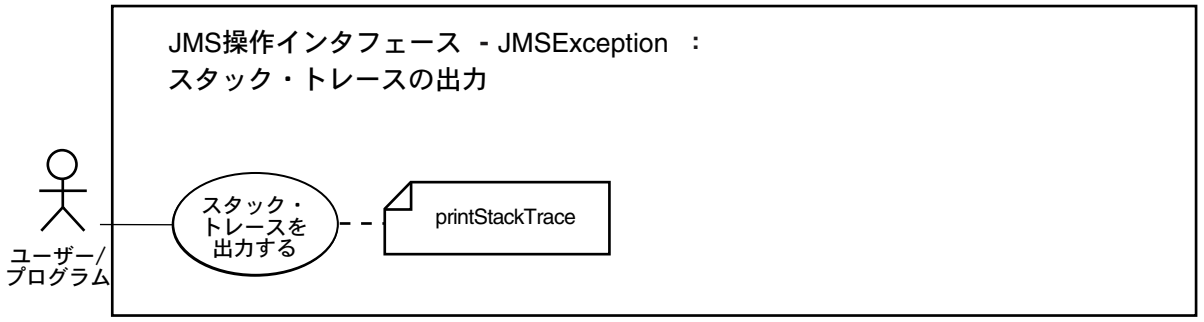
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

JMS 例外のスタック・トレースの出力

図 16-56 JMS 例外のスタック・トレースの出力



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-42 ページの「例外 - javax.jms.JMSException」も参照してください。

用途

JMS 例外のスタック・トレースを出力します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsException」の `printStackTrace`

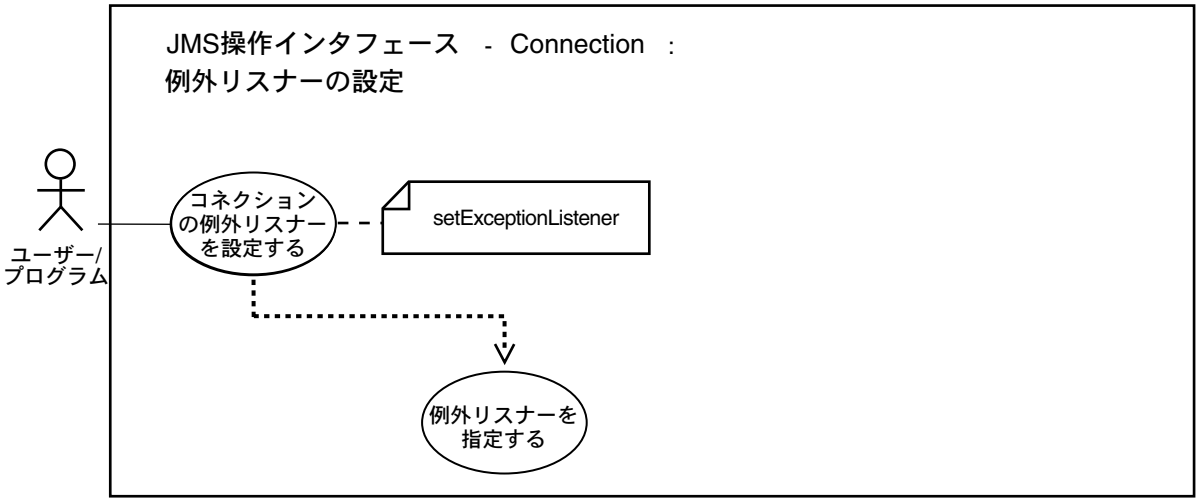
例

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。

今回のリリースでは、例は記載していません。

例外リスナーの設定

図 16-57 例外リスナーの設定



参照：

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-25 ページの「インタフェース - javax.jms.Connection」も参照してください。

用途

コネクション用の例外リスナーを指定します。

使用上の注意

コネクションに重大な問題が検出された場合、そのコネクションの例外リスナー（登録されている場合）に通知されます。これは、リスナーの `onException()` メソッドをコールして、問題を説明する `JMSException` を渡すことで実行されます。これによって、問題が JMS クライアントに非同期に通知されます。メッセージを処理するのみのコネクションもあるため、コネクションが失敗したことを知るための他の方法がありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

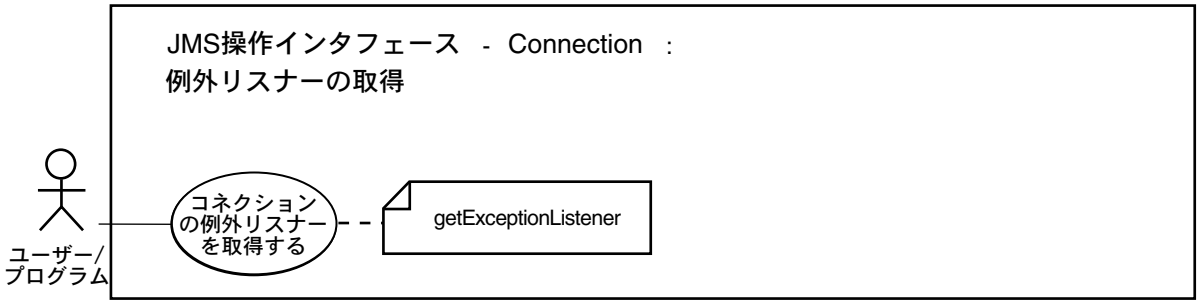
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `setExceptionListener`

例

```
//register an exception listener
Connection jms_connection;
jms_connection.setExceptionListener(
    new ExceptionListener() {
        public void onException (JMSException jmsException) {
            System.out.println("JMS-EXCEPTION: " + jmsException.toString());
        }
    };
  );
```

例外リスナーの取得

図 16-58 例外リスナーの取得



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-25 ページの「インタフェース - javax.jms.Connection」も参照してください。

用途

コネクション用の例外リスナーを取得します。

使用上の注意

ありません。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

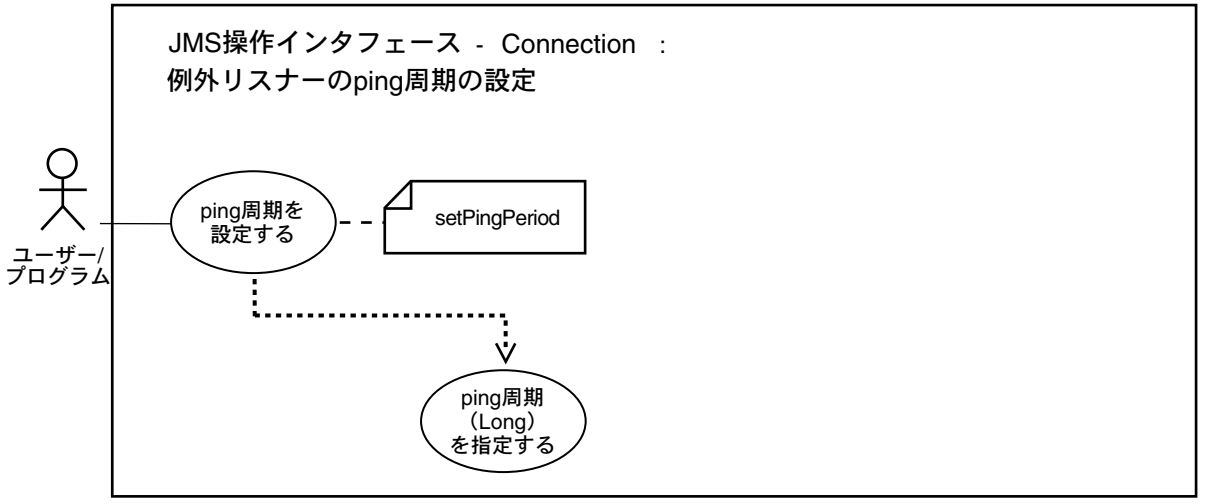
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `getExceptionListener`

例

```
//Get the exception listener
Connection jms_connection;
ExceptionListener el = jms_connection.getExceptionListener();
```

例外リスナーの ping 周期の設定

図 16-59 例外リスナーの ping 周期の設定



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-25 ページの「インタフェース - javax.jms.Connection」も参照してください。

用途

例外リスナーの ping 周期を指定します。

使用上の注意

コネクションに例外リスナーが設定されている場合、そのコネクションは、データベースがアクセス可能であることを確認するために、定期的にデータベースを ping します。ping 周期は、1000 分の 1 秒単位で指定されます。デフォルト値は 2 分です。コネクションに例外リスナーが設定されていない場合、データベースは ping されません。このメソッドは、例外リスナーの設定前または設定後にコールできます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

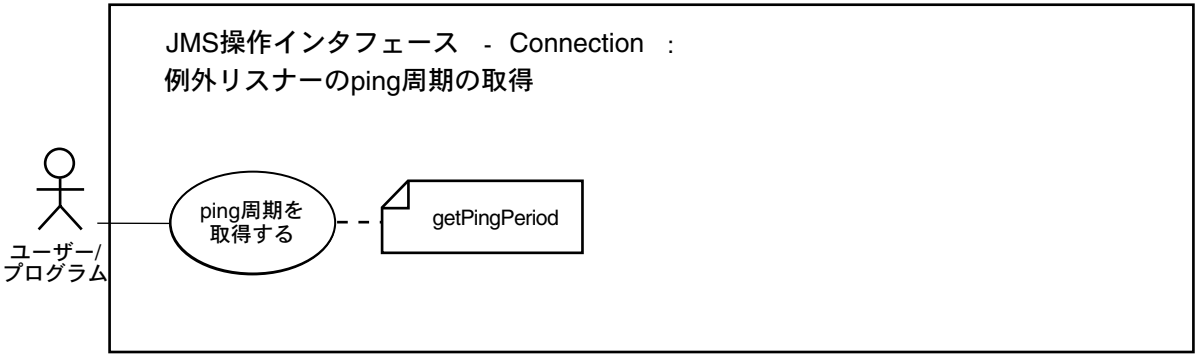
- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `setPingPeriod`

例

```
//set the ping period to 4 minutes
Connection jms_connection;
jms_connection.setPingPeriod(4*60*1000);
```

例外リスナーの ping 周期の取得

図 16-60 例外リスナーの ping 周期の取得



参照 :

- JMS 共有操作インタフェースの基本操作の詳細は、表 16-1 を参照してください。
- B-25 ページの「インタフェース - javax.jms.Connection」も参照してください。

用途

例外リスナーの ping 周期を取得します。

使用上の注意

コネクションに例外リスナーが設定されている場合、そのコネクションは、データベースがアクセス可能であることを確認するために、定期的にデータベースを ping します。ping 周期は、1000 分の 1 秒単位で指定されます。デフォルト値は 2 分です。コネクションに例外リスナーが設定されていない場合、データベースは ping されません。このメソッドは、`setPingPeriod` に対する最新のコールによって設定された周期の値を戻します。`setPingPeriod` が一度もコールされていない場合、デフォルト値が戻されます。

構文

各プログラム環境で使用可能な機能のリストは、第 3 章「AQ プログラム環境」を参照してください。各プログラム環境における構文については、次のマニュアルを参照してください。

- Java（JDBC）：『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』の第 4 章「パッケージ oracle.jms」の「AQjmsConnection」の `getPingPeriod`

例

```
//get the ping period
Connection jms_connection;
long pp = jms_connection.getPingPeriod();
```

17

AQ へのインターネット・アクセス

Simple Object Access Protocol（SOAP）を使用することによって、インターネット経由で AQ にアクセスできます。Internet Data Access Presentation（iDAP）は、AQ 操作の SOAP 仕様です。iDAP は、SOAP 要求の本体に対して XML メッセージ構造を定義します。iDAP によって構造化されたメッセージは、HTTP などの転送プロトコルを使用してインターネット経由で転送されます。

内容は次のとおりです。

- インターネット経由のアドバンスト・キューイング操作の概要
- Internet Data Access Presentation（iDAP）
- SOAP スキーマおよび AQ XML スキーマ
- AQ XML サーブレットの配置
- HTTP を使用した AQ XML サーブレットへのアクセス
- HTTP および HTTPS を使用したアドバンスト・キューイング伝播
- AQ サーブレットのカスタマイズ

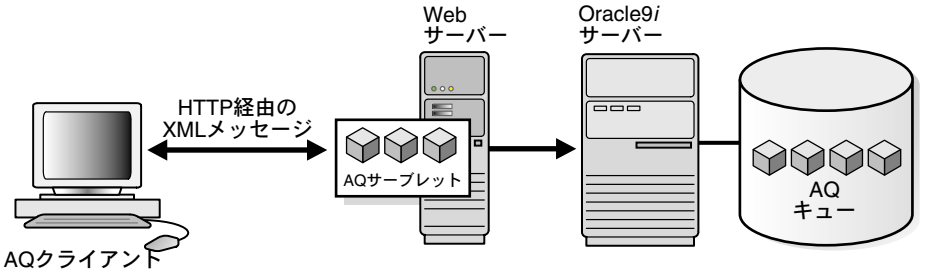
インターネット経由のアドバンスト・キューイング操作の概要

図 17-1 に、HTTP 上で AQ 操作を実行するためのアーキテクチャを示します。主要コンポーネントは次のとおりです。

- AQ クライアント・プログラム
- AQ サーブレットのホスト Web サーバー / サーブレット・コンテナ
- Oracle データベース・サーバー

AQ クライアント・プログラムは、XML メッセージ（iDAP 準拠）を AQ サーブレットに送信します。Web ブラウザなど、任意の HTTP クライアントを使用できます。AQ サーブレットのホスト Web サーバー / サーブレット・コンテナは、着信 XML メッセージを解析します。これには、Apache JServ や Tomcat などがあります。AQ サーブレットは、Oracle データベース・サーバーに接続し、ユーザーのキューに対して操作を実行します。

図 17-1 HTTP を使用して AQ 操作を実行するためのアーキテクチャ



詳細は、17-57 ページの「HTTP を使用した AQ XML サーブレットへのアクセス」および 17-61 ページの「HTTP および HTTPS を使用したアドバンスト・キューイング伝播」を参照してください。

Internet Data Access Presentation（iDAP）

Internet Data Access Presentation（iDAP）は、Content-Type ヘッダーの text/xml を使用して、SOAP リクエストの本体を指定します。XML では、iDAP リクエストおよびレスポンス・メッセージは、次のように表現されます。

- すべてのリクエストおよびレスポンス・タグは、SOAP 名前空間で有効です。
- AQ 操作は、iDAP 名前空間で有効です。
- 送信者は、SOAP 本体の iDAP 要素および属性に名前空間を含めます。
- 受信者は、適切な名前空間を含む SOAP メッセージを処理します。不適切な名前空間を含むリクエストの場合、受信者はリクエストが無効であるというエラーを戻します。
- SOAP 名前空間の値は、http://schemas.xmlsoap.org/soap/envelope/ です。
- iDAP 名前空間の値は、http://ns.oracle.com/AQ/schemas/access です。

SOAP メッセージの構造

SOAP は、次のようにメッセージ・リクエストまたはレスポンスを構造化します。

- SOAP エンベロープ（XML ツリーのルート要素または最上位要素）
- SOAP ヘッダー（ルート下の最初の要素）
- SOAP 本体（AQ XML 文書）

SOAP エンベロープ

SOAP エンベロープはルート要素で、タグは SOAP:Envelope です。SOAP では、グローバル属性 SOAP:encodingStyle が定義されます。この属性は、SOAP 仕様で記述されたルールのかわりに使用されるシリアライズ・ルールを示します。この属性は、すべての要素に適用でき、要素のみでなく、この属性を含まないすべての子要素にも有効です。SOAP:encodingStyle が省略されている場合は、（親要素によってオーバーライドされていないかぎり）型指定に従っていることを意味します。

名前空間で修飾されている場合、SOAP エンベロープには名前空間宣言およびその他の属性も含まれます。本体には、名前空間で修飾された他のサブ要素を続けることができます。

SOAP ヘッダー

SOAP ヘッダーはルート下の最初の要素で、タグは SOAP:Header です。SOAP ヘッダーは、トランザクション ID などの必要な情報をリクエストとともに渡します。ヘッダーは、SOAP:Envelope という XML 要素の子としてエンコードされます。ヘッダーは名前要素で識別され、名前空間で修飾されます。ヘッダー・エントリは、埋込み要素としてエンコードされます。

SOAP 本体

SOAP 本体は、SOAP:Body でタグ付けされ、最初のサブ要素を含みます。このサブ要素の名前はメソッド名です。メソッドをリクエストするこの要素には、各入力パラメータおよび出力パラメータの要素が含まれます。要素名はパラメータ名になります。本体には、エラーに関する情報を示す SOAP:Fault も含まれます。

AQ 操作を実行するには、SOAP 本体に AQ XML 文書を含める必要があります。AQ XML 文書の名前空間は、http://ns.oracle.com/AQ/schemas/access となります。

SOAP メソッドの起動

メソッドの起動は、リクエストのヘッダーおよび本体を作成し、戻されたレスポンスのヘッダーおよび本体を処理することによって実行されます。リクエストおよびレスポンスのヘッダーは、標準の転送プロトコル固有のヘッダーおよび拡張ヘッダーで構成されます。

HTTP ヘッダー

HTTP リクエストのヘッダー内の POST メソッドによって、SOAP メソッドの起動が実行されます。リクエストには、ヘッダー SOAPMethodName を含める必要があります。このヘッダーの値は、ターゲットで起動するメソッドを示します。値は、次に示すように、URI の後に「#」が続き、その後にメソッド名が続きます（メソッド名に「#」を含めることはできません）。

```
SOAPMethodName: http://ns.oracle.com/AQ/schemas/access#AQXmlSend
```

インタフェースに使用する URI は、ペイロードの SOAP:Body 部に存在する、メソッド名要素の暗示または指定された名前空間修飾に一致する必要があります。

メソッド起動本体

SOAP メソッドの起動は、メソッド・リクエストおよびメソッド・レスポンス（オプション）で構成されます。SOAP のメソッド・リクエストおよびメソッド・レスポンスは、それぞれ HTTP のリクエストおよびレスポンスで、その内容は、ルート要素および必須の本体要素で構成される XML 文書です。この章では、この XML 文書を SOAP ペイロードと呼びます。

SOAP ペイロードの定義は次のとおりです。

- SOAP のルート要素は、XML ツリーの最上位要素です。
- SOAP ペイロードのヘッダーには、リクエストとともに送信する必要がある追加情報が含まれます。
- メソッド・リクエストは、XML 要素およびパラメータの追加要素として表されます。メソッド・リクエストは、SOAP:Body 要素の最初の子になります。このリクエストは、次の項に示す AQ XML クライアント・リクエストのいずれかです。
- レスポンスは、戻り値、またはクライアントに戻されるエラーまたは例外です。

受信サイトでは、リクエストの結果は次のいずれかになります。

a. 受信サイトの HTTP インフラストラクチャが、リクエストを受信および処理できます。

b. 受信サイトの HTTP インフラストラクチャが、リクエストを受信および処理できません。

c. 受信サイトの SOAP インフラストラクチャが、入力パラメータをデコードし、サーバー・アドレスで示される適切なサーバーにディスパッチし、メソッド・リクエストに示されているメソッドに意味的に対応するアプリケーション・レベルのファンクションを起動できます。

d. 受信サイトの SOAP インフラストラクチャが、入力パラメータをデコードし、サーバー・アドレスで示される適切なサーバーにディスパッチし、メソッド・リクエストに示されているインタフェースまたはメソッドに意味的に対応するアプリケーション・レベルのファンクションを起動できません。

a の場合、HTTP インフラストラクチャは、SOAP インフラストラクチャにヘッダーおよび本体を渡します。b の場合、HTTP レスポンスには、状態フィールドに HTTP エラーが含まれ、XML の本体が含まれません。c の場合、メソッド・リクエストの結果は、レスポンスまたはエラーで構成されます。d の場合、メソッドの結果はエラーになり、受信側のインフラストラクチャのディスパッチが正常終了できません。c および d の場合、拡張性の目的で、リクエストの結果に追加のメッセージ・ヘッダーが表示される場合があります。

メソッド・リクエストの結果

リクエストの結果は、リクエスト / レスポンスの形式で提供されます。HTTP のレスポンスでは、Content-Type ヘッダーが `text/xml` である必要があります。SOAP の結果は成功を示し、エラーは失敗を示します。メソッド・レスポンスには、結果とエラーの両方が含まれることはありません。

iDAP ドキュメント

SOAP メッセージの本体は、iDAP メッセージです。この XML 文書の名前空間は、http://ns.oracle.com/AQ/schemas/access となります。この本体は次の内容を表しています。

- クライアントによるエンキュー、デキューおよび登録のリクエスト
- クライアントによるエンキュー、デキューおよび登録のリクエストに対するサーバーのレスポンス
- サーバーからクライアントへの通知

> **注意：** AQ インターネット・アクセスは、8.1 形式のキューのみでサポートされています。iDAP を使用して 8.0 形式のキューにアクセスすることはできません。

クライアントによるエンキューのリクエスト

クライアントによるエンキューのリクエスト（SEND リクエストおよび PUBLISH リクエスト）には、次のメソッドを使用します。

- AQXmlSend - シングル・コンシューマ・キューへのエンキュー
- AQXmlPublish - マルチ・コンシューマ・キュー / トピックへのエンキュー

表 17-1 に、AQXmlSend および AQXmlPublish が取る引数および引数属性を示します。必須の引数は太字で示します。

表 17-1 クライアントによるエンキューのリクエスト - AQXmlSend および AQXmlPublish の引数および属性

引数	属性
producer_options	**destination** - メッセージを送信するキュー / トピックを指定します。destination 要素には、destination 要素の値の解析方法を決める lookup_type 属性が含まれます。 ■ DATABASE（デフォルト）- 宛先は schema.queue_name と解析されます。 ■ LDAP - LDAP サーバーを使用して、宛先を解決します。 visibility ■ ON_COMMIT - エンキューは現行のトランザクションの一部になります。トランザクションをコミットすると、操作が完了します。これはデフォルトです。 ■ IMMEDIATE - リクエストの完了直後に、エンキューが反映されます。エンキューは、現行のトランザクションの一部になりません。その操作のみで 1 つのトランザクションを構成します。 transformation - メッセージをエンキューする前に、PL/SQL 変換を開始します。

表 17-1 クライアントによるエンキューのリクエスト - AQXmlSend および AQXmlPublish の引数および属性（続き）

引数	属性
message_set - 1 つ以上のメッセージが含まれます。	各メッセージは、message_header および message_payload で構成されます。
■ message_header	message_id - デキュー中に提供される、メッセージの一意の識別子。 correlation - メッセージの相関識別子。
	expiration - メッセージのデキューが可能な時間（秒）。このパラメータは、遅延に対するオフセットです。デフォルトでは、メッセージに期限はありません。 期限切れになる前にデキューされない場合、メッセージは EXPIRED 状態で例外キューに移されます。 delay - メッセージの処理が可能な時間（秒）。 priority - メッセージの優先順位。数値が小さいほど、優先順位が高いことを表します。優先順位には、負の数を含むすべての数値を指定できます。
	sender_id - アプリケーション固有の識別子。 ■ agent_name、address、protocol ■ agent_alias - 指定すると、LDAP を使用して、名前、アドレス、プロトコルが変換されます。 recipient_list - 受信者のリスト。デフォルトのサブスクライバ・リストをオーバーライドします。各受信者は、次の属性で構成されます。 ■ agent_name、address、protocol ■ agent_alias - 指定すると、LDAP を使用して、名前、アドレス、プロトコルが変換されます。
	message_state - デキュー中にメッセージの状態が自動的に書き込まれます。 0: メッセージはすぐに処理できます。 1: メッセージ遅延に到達していません。 2: メッセージは処理され、保存されています。 3: メッセージは例外キューに移されています。

表 17-1 クライアントによるエンキューのリクエスト - AQXmlSend および AQXmlPublish の引数および属性（続き）

引数	属性
	exception_queue - 例外が発生した場合に、正常に処理されなかったメッセージが移されるキューの名前。メッセージのデキューが正常に行われなかった回数が max_retries を超えたり、メッセージが期限切れになった場合、または例外キューのすべてのメッセージが EXPIRED 状態にある場合に、メッセージは移されます。 デフォルトは、キュー表に対応付けられた例外キューです。メッセージの移動時に指定された例外キューが存在しない場合、メッセージはキュー表に対応付けられたデフォルトの例外キューに移され、アラート・ファイルに警告が記録されます。デフォルトの例外キューが使用されると、デキュー時にパラメータが NULL 値を戻します。
■ **message_payload**	宛先キュー / トピックのペイロード型に基づいて、異なるサブ要素を持つことができます。様々なペイロード型については、次の項を参照してください。
AQXmlCommit	これは空要素です。指定すると、リクエストの終了時にユーザー・トランザクションがコミットされます。

メッセージ・ペイロード

AQ では、次のメッセージ型がサポートされています。

- RAW 型
- Oracle オブジェクト（ユーザー定義型）
- JMS 型
  - TextMessage
  - MapMessage
  - BytesMessage
  - ObjectMessage

これらすべての型のキューに SOAP を使用してアクセスできます。キューに RAW、Oracle オブジェクトまたは JMS フォーマットのメッセージが含まれる場合、XML のペイロードはエンキュー中に適切な内部フォーマットに変換され、キューに格納されます。デキュー中、前述のいずれかのフォーマットのメッセージを含むキューからメッセージが取得された場合、そのメッセージはクライアントに送信される前に XML に変換されます。

メッセージ・ペイロード型は、操作が実行されているキューの型によって異なります。次の項では、キューの型について説明します。

RAW キュー　RAW キューの内容は、バイト列です。XML メッセージでは、メッセージ・ペイロードを 16 進表示（たとえば、<raw>023f4523</raw>）にする必要があります。

Oracle オブジェクト型（ユーザー定義型）のキュー　JMS キューではない（AQ$_JMS_* 型ではない）ユーザー定義型キューの場合、ペイロードの型は、キューを保持するキュー表の作成中に指定した型によって異なります。ここで指定された XML は、キュー表に対するペイロードの SQL 型にマップする必要があります。

> **参照：** XML に対する SQL 型のマッピングの詳細は、『Oracle9*i* XML データベース開発者ガイド -Oracle XML DB』を参照してください。

例　キューの型を EMP_TYP と定義し、次のような構造であるとします。

```
create or replace type emp_typ as object (
     empno NUMBER(4),
     ename VARCHAR2(10),
     job VARCHAR2(9),
     mgr NUMBER(4),
     hiredate DATE,
     sal   NUMBER(7,2),
     comm  NUMBER(7,2)
     deptno NUMBER(2));
```

対応する XML 表現は、次のとおりです。

```
<EMP_TYP>
  <EMPNO>1111</EMPNO>
  <ENAME>Mary</ENAME>
  <MGR>5000</MGR>
  <HIREDATE>1996-01-01 0:0:0</HIREDATE>
  <SAL>10000</SAL>
  <COMM>100.12</COMM>
  <DEPTNO>60</DEPTNO>
</EMP_TYP>
```

JMS 型のキュー / トピック　JMS 型のキュー（ペイロード型が AQ$_JMS_* のキュー）の場合、JMS 型に応じて 4 つの XML 要素があります。iDAP では、JMS 型が TextMessage、MapMessage、BytesMessage および ObjectMessage のキュー / トピックがサポートされています。iDAP では、ペイロード型が StreamMessage の JMS キューはサポートされていません。

表 17-2 に、JMS 型および XML コンポーネントを示します。各列には、各 JMS 型に固有の XML 要素を示します。

表 17-2 JMS 型および XML コンポーネント - キュー / トピックに使用されるペイロード型

AQ$_JMS_TEXT_MESSAGE	AQ$_JMS_MAP_MESSAGE	AQ$_JMS_BYTES_MESSAGE	AQ$_JMS_OBJECT_MESSAGE
jms_text_message	**jms_map_message**	**jms_bytes_message**	**jms_object_message**
oracle_jms_properties	oracle_jms_properties	oracle_jms_properties	oracle_jms_properties
user_properties	user_properties	user_properties	user_properties
text_data - Text ペイロードを表す文字列	**map_data** - 次の名前と値の組（項目）の集合 ■ name ■ int_value string_value long_value double_value boolean_value float_value short_value byte_value	**bytes_data** - ペイロード・バイトの 16 進表示	**ser_object_data** - シリアル化オブジェクトの 16 進表示

表 17-2 では、必須の要素は太字で示します。

すべての JMS メッセージは、次の共通要素で構成されます。

- 次の要素で構成される oracle_jms_properties。
  - type - メッセージの型。
  - reply_to - agent_name、address および protocol で構成されます。
  - userid - AQ によって指定されます。クライアントは指定できません。
  - appid - アプリケーションの識別子。
  - groupid - グループの識別子。
  - group_sequence - group_id で識別されるグループ内の順序。
  - timestamp - メッセージが送信された時間。エンキュー中は指定できません。デキューされるメッセージに自動的に移入されます。
  - recv_timestamp - メッセージが受信された時間。
- user_properties - 前述の事前定義のプロパティの他に、独自のメッセージ・プロパティを名前と値の組で指定できます。user_properties は、プロパティ要素のリストで構成されます。各プロパティは、次の名前と値の組で構成されます。
  - name - プロパティ名
  - int_value - プロパティ値（Interger）

    string_value - プロパティ値（String）

    long_value - プロパティ値（Long）

    double_value - プロパティ値（Double）

    boolean_value - プロパティ値（Boolean）

    float_value - プロパティ値（Float）

    short_value - プロパティ値（Short）

    byte_value - プロパティ値（Byte）

次の項では、様々なメッセージ型およびキュー型を使用したエンキュー・リクエストの例を示します。

エンキュー・リクエストの例 - シングル・コンシューマ・キューへのユーザー定義型メッセージの送信

キュー QS.NEW_ORDER_QUE のペイロード型は、ORDER_TYP です。

```
<?xml version="1.0"?>
  <Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
     <Body>
       <AQXmlSend xmlns = "http://ns.oracle.com/AQ/schemas/access">
         <producer_options>
           <destination>QS.NEW_ORDERS_QUE</destination>
         </producer_options>

         <message_set>
           <message_count>1</message_count>

           <message>
             <message_number>1</message_number>

             <message_header>
               <correlation>ORDER1</correlation>
               <sender_id>
                 <agent_name>scott</agent_name>
               </sender_id>
             </message_header>

             <message_payload>

              <ORDER_TYP>
                    <ORDERNO>100</ORDERNO>
                    <STATUS>NEW</STATUS>
                    <ORDERTYPE>URGENT</ORDERTYPE>
                    <ORDERREGION>EAST</ORDERREGION>
                    <CUSTOMER>
                       <CUSTNO>1001233</CUSTNO>
                       <CUSTID>MA1234555623212</CUSTID>
                       <NAME>AMERICAN EXPRESS</NAME>
                       <STREET>EXPRESS STREET</STREET>
                       <CITY>REDWOOD CITY</CITY>
                       <STATE>CA</STATE>
                       <ZIP>94065</ZIP>
                       <COUNTRY>USA</COUNTRY>
                    </CUSTOMER>
                    <PAYMENTMETHOD>CREDIT</PAYMENTMETHOD>
                    <ITEMS>
                       <ITEMS_ITEM>
                          <QUANTITY>10</QUANTITY>
                          <ITEM>
```

```
                        <TITLE>Perl</TITLE>
                        <AUTHORS>Randal</AUTHORS>
                        <ISBN>ISBN20200</ISBN>
                        <PRICE>19</PRICE>
                     </ITEM>
                     <SUBTOTAL>190</SUBTOTAL>
                  </ITEMS_ITEM>
                  <ITEMS_ITEM>
                     <QUANTITY>20</QUANTITY>
                     <ITEM>
                        <TITLE>XML</TITLE>
                        <AUTHORS>Micheal</AUTHORS>
                        <ISBN>ISBN20212</ISBN>
                        <PRICE>59</PRICE>
                     </ITEM>
                     <SUBTOTAL>590</SUBTOTAL>
                  </ITEMS_ITEM>
               </ITEMS>
               <CCNUMBER>NUMBER01</CCNUMBER>
               <ORDER_DATE>2000-08-23 0:0:0</ORDER_DATE>
            </ORDER_TYP>
         </message_payload>
      </message>
   </message_set>
</AQXmlSend>
</Body>
</Envelope>
```

エンキュー・リクエストの例 - マルチ・コンシューマ・キューへのユーザー定義型メッセージのパブリッシュ

マルチ・コンシューマ・キュー AQUSER.EMP_TOPIC のペイロード型は、EMP_TYP です。EMP_TYP の構造は、次のとおりです。

```
create or replace type emp_typ as object (
    empno NUMBER(4),
    ename VARCHAR2(10),
    job VARCHAR2(9),
    mgr NUMBER(4),
    hiredate DATE,
    sal   NUMBER(7,2),
    comm  NUMBER(7,2)
    deptno NUMBER(2));
```

PUBLISH リクエストのフォーマットは、次のようになります。

```
<?xml version="1.0"?>
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">

      <Body>
        <AQXmlPublish xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <producer_options>
            <destination>AQUSER.EMP_TOPIC</destination>
          </producer_options>

              <message_set>
                <message_count>1</message_count>

                <message>
                  <message_number>1</message_number>

                  <message_header>
                     <correlation>NEWEMP</correlation>
                     <sender_id>
                        <agent_name>scott</agent_name>
                     </sender_id>
                  </message_header>

                  <message_payload>
                  <EMP_TYP>
                    <EMPNO>1111</EMPNO>
                    <ENAME>Mary</ENAME>
                    <MGR>5000</MGR>
                    <HIREDATE>1996-01-01 0:0:0</HIREDATE>
                    <SAL>10000</SAL>
                    <COMM>100.12</COMM>
                    <DEPTNO>60</DEPTNO>
                  </EMP_TYP>
                 </message_payload>
                </message>
              </message_set>
            </AQXmlPublish>
          </Body>
    </Envelope>
```

エンキュー・リクエストの例 - JMS キューへのメッセージの送信

JMS キュー AQUSER.JMS_TEXTQ のペイロード型は、JMS の TextMessage（SYS.AQ$_JMS_TEXT_MESSAGE）です。SEND リクエストのフォーマットは、次のようになります。

```
<?xml version="1.0"?>
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>

        <AQXmlSend xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <producer_options>
            <destination>AQUSER.JMS_TEXTQ</destination>
          </producer_options>

          <message_set>
            <message_count>1</message_count>

            <message>
              <message_number>1</message_number>

              <message_header>
                <correlation>text_msg</correlation>
                  <sender_id>
                     <agent_name>john</agent_name>
                  </sender_id>
              </message_header>

              <message_payload>

              <jms_text_message>
                <oracle_jms_properties>
                 <appid>AQProduct</appid>
                 <groupid>AQ</groupid>
                </oracle_jms_properties>

                <user_properties>
                  <property>
                    <name>Country</name>
                    <string_value>USA</string_value>
                  </property>
                  <property>
                     <name>State</name>
                    <string_value>California</string_value>
                  </property>
                 </user_properties>

                 <text_data>All things bright and beautiful</text_data>
```

```
            </jms_text_message>
          </message_payload>
        </message>
      </message_set>
    </AQXmlSend>
  </Body>
</Envelope>
```

エンキュー・リクエストの例 - JMS トピックへのメッセージのパブリッシュ

JMS トピック AQUSER.JMS_MAP_TOPIC のペイロード型は、JMS の MapMessage（SYS.AQ$_JMS_MAP_MESSAGE）です。PUBLISH リクエストのフォーマットは、次のようになります。

```
<?xml version="1.0"?>

<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>

        <AQXmlPublish xmlns = "http://ns.oracle.com/AQ/schemas/access">

          <producer_options>
            <destination>AQUSER.JMS_MAP_TOPIC</destination>
          </producer_options>

          <message_set>
            <message_count>1</message_count>

            <message>
              <message_number>1</message_number>

              <message_header>
                  <correlation>toyota</correlation>
                  <sender_id >
                     <agent_name>john</agent_name>
                  </sender_id>
                  <recipient_list>
                     <recipient>
                        <agent_name>scott</agent_name>
                     </recipient>
                     <recipient>
                        <agent_name>aquser</agent_name>
                     </recipient>
                     <recipient>
                        <agent_name>jmsuser</agent_name>
                     </recipient>
```

```
                    </recipient_list>
                  </message_header>

                  <message_payload>

                    <jms_map_message>
                      <oracle_jms_properties>
                        <reply_to>
                           <agent_name>oracle</agent_name>
                        </reply_to>
                       <groupid>AQ</groupid>
                      </oracle_jms_properties>

                      <user_properties>
                        <property>
                          <name>Country</name>
                          <string_value>USA</string_value>
                        </property>
                        <property>
                          <name>State</name>
                          <string_value>California</string_value>
                        </property>
                       </user_properties>

                      <map_data>
                       <item>
                        <name>Car</name>
                        <string_value>Toyota</string_value>
                       </item>
                       <item>
                         <name>Color</name>
                         <string_value>Blue</string_value>
                       </item>
                       <item>
                         <name>Price</name>
                         <int_value>20000</int_value>
                       </item>
                      </map_data>
                     </jms_map_message>
                   </message_payload>
                 </message>
               </message_set>
             </AQXmlPublish>
          </Body>
</Envelope>
```

エンキュー・リクエストの例 - RAW 型のペイロードを持つキューへのメッセージの送信

キュー AQUSER.RAW_MSGQ のペイロード型は、RAW です。SEND リクエストのフォーマットは、次のようになります。

```
<?xml version="1.0"?>
   <Envelope  xmlns =  "http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <AQXmlSend xmlns = "http://ns.oracle.com/AQ/schemas/access">
        <producer_options>
          <destination>AQUSER.RAW_MSGQ</destination>
        </producer_options>
        <message_set>
          <message_count>1</message_count>

          <message>
             <message_number>1</message_number>

                <message_header>
                  <correlation>TKAXAS11</correlation>
                  <sender_id>
                     <agent_name>scott</agent_name>
                  </sender_id>
                </message_header>
                <message_payload>

<RAW>426C6F622064617461202D20626C6F622064617461202D20626C6F62206461746120426C6F622064617461202D20626C6F622064617461202D20626C6F62206461746120426</RAW>
                </message_payload>
              </message>
            </message_set>
          </AQXmlSend>
      </Body>
</Envelope>
```

エンキュー・リクエストの例 - トランザクションの送信 / パブリッシュおよびコミット

```
<?xml version="1.0"?>
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">

      <Body>
        <AQXmlPublish xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <producer_options>
            <destination>AQUSER.EMP_TOPIC</destination>
          </producer_options>

          <message_set>
            <message_count>1</message_count>

            <message>
              <message_number>1</message_number>

               <message_header>
          <correlation>NEWEMP</correlation>
          <sender_id>
             <agent_name>scott</agent_name>
          </sender_id>
               </message_header>

               <message_payload>
                 <EMP_TYP>
                   <EMPNO>1111</EMPNO>
                   <ENAME>Mary</ENAME>
                   <MGR>5000</MGR>
                   <HIREDATE>1996-01-01 0:0:0</HIREDATE>
                   <SAL>10000</SAL>
                   <COMM>100.12</COMM>
                   <DEPTNO>60</DEPTNO>
                </EMP_TYP>
              </message_payload>
            </message>
          </message_set>

          <AQXmlCommit/>

          </AQXmlPublish>
        </Body>
</Envelope>
```

クライアントによるデキューのリクエスト

クライアントによるデキューのリクエストには、AQXmlReceive メソッドを使用します。表 17-3 に、このメソッドが取る引数および引数属性を示します。必須の引数は太字で示します。

表 17-3 クライアントによるデキューのリクエスト - AQXmlReceive の引数および引数属性

引数	属性
consumer_options	**destination** - メッセージを受信するキュー / トピックを指定します。destination 要素には、destination 要素の値の解析方法を決める lookup_type 属性が含まれます。 ■ DATABASE（デフォルト）- 宛先は schema.queue_name と解析されます。 ■ LDAP - LDAP サーバーを使用して、宛先を解決します。 consumer_name - コンシューマの名前。コンシューマ名に一致するメッセージのみがアクセスされます。キューがマルチ・コンシューマ用に設定されていない場合は、このフィールドを指定しないでください。 wait_time - 検索基準に一致するメッセージが現在存在しない場合の待機時間（秒）。 selector - メッセージを選択するために使用する基準。次のいずれかに指定します。 ■ correlation - デキューするメッセージの相関識別子 ■ message_id - デキューするメッセージのメッセージ識別子 ■ condition - この条件を満たすデキュー・メッセージ 条件は、SQL 問合せの WHERE 句に類似した構文を使用するブール式で指定します。このブール式には、メッセージ・プロパティ、ユーザー・データ・プロパティ（オブジェクト・ペイロードのみ）および（SQL 問合せの WHERE 句に指定する）PL/SQL ファンクションまたは SQL ファンクションを含めることができます。メッセージ・プロパティには、priority、corrid およびキュー表内の他の列を含めることができます。 メッセージ・ペイロード（オブジェクト・ペイロード）にデキュー条件を指定するには、句にオブジェクト型の属性を使用します。各属性には、ペイロードを格納するキュー表の特定の列を示す識別子として、接頭辞 tab.user_data が必要です。deq_condition パラメータは、4,000 文字に制限されています。

表 17-3 クライアントによるデキューのリクエスト - AQXmlReceive の引数および引数属性（続き）

引数	属性
	visibility ■ ON_COMMIT（デフォルト）- デキューは現行のトランザクションの一部になります。トランザクションをコミットすると、操作が完了します。 ■ IMMEDIATE - リクエストの完了直後に、デキューが反映されます。デキューは、現行のトランザクションの一部になりません。その操作のみで 1 つのトランザクションを構成します。 dequeue_mode - デキューに対応付けられたロック動作を指定します。dequeue_mode は、次のいずれかに指定できます。 ■ REMOVE（デフォルト）- メッセージを読み込み、そのメッセージを更新または削除します。これはデフォルトです。メッセージは、retention プロパティに基づいて、キュー表に保存できます。 ■ BROWSE - メッセージのロックを取得せずに、メッセージを読み込みます。これは、SELECT 文と同じです。 ■ LOCKED - メッセージの書込みロックを読み込み、取得します。ロックは、トランザクションが継続している間、有効です。これは UPDATE 文の選択と同じです。 navigation_mode - 取り出すメッセージの位置を指定します。まず、位置を決定します。次に、検索基準を適用します。最後に、メッセージを取り出します。navigation_mode は、次のいずれかに指定できます。 ■ FIRST_MESSAGE - 使用可能で検索基準に一致する最初のメッセージを取り出します。これによって、位置がキューの先頭にリセットされます。 ■ NEXT_MESSAGE（デフォルト）- 使用可能で検索基準に一致する次のメッセージを取り出します。前のメッセージがメッセージ・グループに属する場合、AQ は、そのメッセージ・グループに属するメッセージの中から、検索基準に一致する次の使用可能なメッセージを取り出します。これはデフォルトです。 ■ NEXT_TRANSACTION - 現行のトランザクション・グループ（存在する場合）の残りをスキップし、次のトランザクション・グループの最初のメッセージを取り出します。このオプションは、現行のキューでメッセージのグループ化が可能な場合のみに使用できます。 transformation - メッセージをデキューした後に、PL/SQL 変換を開始します。
AQXmlCommit	これは空要素です。指定すると、リクエストの終了時にユーザー・トランザクションがコミットされます。

次の項では、様々な AQXmlReceive 属性を使用したデキュー・リクエストの例を示します。

デキュー・リクエストの例 - シングル・コンシューマ・キューからのメッセージの受信

シングル・コンシューマ・キュー QS.NEW_ORDERS_QUE を使用する場合、RECEIVE リクエストのフォーマットは次のようになります。

```
<?xml version="1.0"?>

<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
        <AQXmlReceive xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <consumer_options>
            <destination>QS.NEW_ORDERS_QUE</destination>
            <wait_time>0</wait_time>
          </consumer_options>
        </AQXmlReceive>
      </Body>
</Envelope>
```

デキュー・リクエストの例 - マルチ・コンシューマ・キューからのメッセージの受信

マルチ・コンシューマ・キュー AQUSER.EMP_TOPIC を APP1 サブスクライバで使用する場合、RECEIVE リクエストのフォーマットは次のようになります。

```
<?xml version="1.0"?>
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
        <AQXmlReceive xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <consumer_options>
            <destination>AQUSER.EMP_TOPIC</destination>
            <consumer_name>APP1</consumer_name>
            <wait_time>0</wait_time>
            <navigation_mode>FIRST_MESSAGE</navigation_mode>
          </consumer_options>
        </AQXmlReceive>
      </Body>
</Envelope>
```

デキュー・リクエストの例 - 特定の相関識別子からのメッセージの受信

シングル・コンシューマ・キュー QS.NEW_ORDERS_QUE を使用して相関識別子が NEW のメッセージを受信する場合、RECEIVE リクエストのフォーマットは次のようになります。

```
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
        <AQXmlReceive xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <consumer_options>
            <destination>QS.NEW_ORDERS_QUE</destination>
            <wait_time>0</wait_time>
            <selector>
                 <correlation>NEW</correlation>
            </selector>
          </consumer_options>
        </AQXmlReceive>
      </Body>
</Envelope>
```

デキュー・リクエストの例 - 特定の条件を満たすメッセージの受信

マルチ・コンシューマ・キュー AQUSER.EMP_TOPIC をサブスクライバ APP1 および条件 deptno=60 で使用する場合、RECEIVE リクエストのフォーマットは次のようになります。

```
<?xml version="1.0"?>
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
        <AQXmlReceive xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <consumer_options>
            <destination>AQUSER.EMP_TOPIC</destination>
            <consumer_name>APP1</consumer_name>
            <wait_time>0</wait_time>
            <selector>
                 <condition>tab.user_data.deptno=60</condition>
            </selector>
          </consumer_options>
        </AQXmlReceive>
      </Body>
</Envelope>
```

デキュー・リクエストの例 - メッセージの受信およびコミット

デキュー・リクエストの例で、RECEIVE リクエストの最後に AQXmlCommit を含めると、操作の完了後にトランザクションがコミットされます。17-22 ページの「デキュー・リクエストの例 - マルチ・コンシューマ・キューからのメッセージの受信」で、受信リクエストに次に示すコミット・フラグを含めることができます。

```
 <?xml version="1.0"?>

<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
        <AQXmlReceive xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <consumer_options>
            <destination>QS.NEW_ORDERS_QUE</destination>
            <wait_time>0</wait_time>
          </consumer_options>

          <AQXmlCommit/>

        </AQXmlReceive>
      </Body>
</Envelope>
```

デキュー・リクエストの例 - メッセージのブラウズ

デフォルトでは、メッセージは REMOVE モードでデキューされます。BROWSE モードで QS.NEW_ORDERS_QUE からメッセージを受信するには、RECEIVE リクエストを次のように変更します。

```
<?xml version="1.0"?>

<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>
        <AQXmlReceive xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <consumer_options>
            <destination>QS.NEW_ORDERS_QUE</destination>
            <wait_time>0</wait_time>
            <dequeue_mode>BROWSE</dequeue_mode>
          </consumer_options>
         </AQXmlReceive>
      </Body>
</Envelope>
```

クライアントによる登録のリクエスト

クライアントによる登録のリクエストには、AQXmlRegister メソッドを使用します。表 17-4 に、このメソッドが取る引数および引数属性を示します。必須の引数は太字で示します。

表 17-4 クライアント登録 - AQXmlRegister の引数および属性

引数	属性
register_options	**destination** - 通知を登録するキューまたはトピックを指定します。destination 要素には、destination 要素の値の解析方法を決める lookup_type 属性が含まれます。 ■ DATABASE（デフォルト）- 宛先は schema.queue_name と解析されます。 ■ LDAP - LDAP サーバーを使用して、宛先を解決します。 consumer_name - マルチ・コンシューマ・キューまたはトピックのコンシューマ名。シングル・コンシューマ・キューの場合は、このパラメータを指定しないでください。 **notify_url** - メッセージのエンキュー時に通知を送信する場所。フォームは、http://<url>、mailto://<email address> または plsql://<pl/sql procedure> です。

登録リクエストの例 - 電子メール・アドレスでの通知の登録

キュー AQUSER.EMP_TOPIC の APP1 コンシューマにエンキューされているメッセージの電子メール・アドレスを通知する場合、REGISTER リクエストのフォーマットは次のようになります。

```
<?xml version="1.0"?>
<Envelope xmlns= "http://schemas.xmlsoap.org/soap/envelope/">
      <Body>

        <AQXmlRegister xmlns = "http://ns.oracle.com/AQ/schemas/access">

          <register_options>
            <destination>AQUSER.EMP_TOPIC</destination>
            <consumer_name>APP1</consumer_name>
            <notify_url>mailto:app1@hotmail.com</notify_url>
          </register_options>

          <AQXmlCommit/>

        </AQXmlRegister>
      </Body>
</Envelope>
```

クライアントによるトランザクションのコミット・リクエスト

セッション中にユーザーが実行するすべてのアクションのコミット・リクエストには、AQXmlCommit メソッドを使用します。

コミット・リクエストの例

COMMIT リクエストのフォーマットは、次のようになります。

```
<?xml version="1.0"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <AQXmlCommit xmlns="http://ns.oracle.com/AQ/schemas/access"/>
   </Body>
</Envelope>
```

クライアントによるトランザクションのロールバック・リクエスト

セッション中にユーザーが実行するすべてのアクションのロールバック・リクエストには、AQXmlRollback メソッドを使用します。可視性を IMMEDIATE に設定して実行するアクションは、ロールバックされません。

ロールバック・リクエストの例

ROLLBACK リクエストのフォーマットは、次のようになります。

```
<?xml version="1.0"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <AQXmlRollback xmlns="http://ns.oracle.com/AQ/schemas/access"/>
   </Body>
</Envelope>
```

サーバーのエンキュー・レスポンス

シングル・コンシューマ・キューへのエンキュー・リクエストのレスポンスには、AQXmlSendResponse メソッドを使用します。表 17-5 に、レスポンスのコンポーネントを示します。

表 17-5 シングル・コンシューマ・キューへのサーバーのエンキュー・レスポンス（AQXmlSendResponse）

レスポンス	属性
status_response	status_code - 成功（0）または失敗（-1）
	error_code - エラーの Oracle コード
	error_message - エラーの説明
send_result	destination - メッセージが送信された場所
	message_id - 送信された各メッセージの識別子

サーバー・リクエストの例 - シングル・コンシューマ・キューへの単一メッセージのエンキュー

シングル・コンシューマ・キュー QS.NEW_ORDERS_QUE への SEND リクエストの結果のフォーマットは、次のようになります。

```
<?xml version = '1.0'?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <AQXmlSendResponse xmlns="http://ns.oracle.com/AQ/schemas/access">
         <status_response>
            <status_code>0</status_code>
         </status_response>
         <send_result>
            <destination>QS.NEW_ORDERS_QUE</destination>
            <message_id>12341234123412341234</message_id>
         </send_result>
      </AQXmlSendResponse>
   </Body>
</Envelope>
```

マルチ・コンシューマ・キューまたはトピックへのエンキュー・リクエストのレスポンスには、AQXmlPublishResponse メソッドを使用します。表 17-6 に、レスポンスのコンポーネントを示します。

表 17-6 マルチ・コンシューマ・キューまたはトピックへのエンキューのサーバー・レスポンス（AQXmlPublishResponse）

レスポンス	属性
status_response	status_code - 成功（0）または失敗（-1）
	error_code - エラーの Oracle コード
	error_message - エラーの説明
publish_result	destination - メッセージが送信された場所
	message_id - 送信された各メッセージの識別子

サーバー・リクエストの例 - マルチ・コンシューマ・キューへのエンキュー

マルチ・コンシューマ・キュー AQUSER.EMP_TOPIC への SEND リクエストの結果のフォーマットは、次のようになります。

```
<?xml version = '1.0'?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <AQXmlPublishResponse xmlns="http://ns.oracle.com/AQ/schemas/access">
         <status_response>
            <status_code>0</status_code>
         </status_response>
         <publish_result>
            <destination>AQUSER.EMP_TOPIC</destination>
            <message_id>2343443543545646546546546</message_id>
         </publish_result>
      </AQXmlPublishResponse>
   </Body>
</Envelope>
```

デキュー・リクエストへのサーバー・レスポンス

デキュー・リクエストへのレスポンスには、AQXmlReceiveResponse メソッドを使用します。表 17-7 に、レスポンスのコンポーネントを示します。

表 17-7 キューまたはトピックからのデキューへのサーバー・レスポンス（AQXmlReceiveResponse）

レスポンス	属性
status_response	status_code - 成功（0）または失敗（-1） error_code - エラーの Oracle コード error_message - エラーの説明
receive_result	destination - メッセージが送信された場所 message_set - デキューされたメッセージの集合

デキュー・レスポンスの例 - ユーザー定義型キューからのメッセージの受信（AQXmlReceiveResponse）

ペイロード型が EMP_TYP のキュー AQUSER.EMP_TOPIC に対する RECEIVE リクエストの結果のフォーマットは、次のようになります。

```
<?xml version = '1.0'?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <AQXmlReceiveResponse xmlns="http://ns.oracle.com/AQ/schemas/access">
         <status_response>
            <status_code>0</status_code>
         </status_response>
         <receive_result>
            <destination>AQUSER.EMP_TOPIC</destination>
            <message_set>
               <message_count>1</message_count>
               <message>
                  <message_number>1</message_number>
                  <message_header>
                     <message_id>1234344545565667</message_id>
                     <correlation>TKAXAP10</correlation>
                     <priority>1</priority>
                     <delivery_count>0</delivery_count>
                     <sender_id>
                        <agent_name>scott</agent_name>
                     </sender_id>
                     <message_state>0</message_state>
                  </message_header>
                  <message_payload>
                    <EMP_TYP>
```

```
                        <EMPNO>1111</EMPNO>
                        <ENAME>Mary</ENAME>
                        <MGR>5000</MGR>
                        <HIREDATE>1996-01-01 0:0:0</HIREDATE>
                        <SAL>10000</SAL>
                        <COMM>100.12</COMM>
                        <DEPTNO>60</DEPTNO>
                     </EMP_TYP>
                  </message_payload>
               </message>
            </message_set>
         </receive_result>
      </AQXmlReceiveResponse>
   </Body>
</Envelope>
```

デキュー・レスポンスの例 - JMS キューからのメッセージの受信

ペイロード型が JMS の TextMessage のキューに対する RECEIVE リクエストの結果のフォーマットは、次のようになります。

```
<?xml version = '1.0'?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
<Body>
      <AQXmlReceiveResponse xmlns="http://ns.oracle.com/AQ/schemas/access">
         <status_response>
            <status_code>0</status_code>
         </status_response>
         <receive_result>
            <destination>AQUSER.JMS_TEXTQ</destination>
            <message_set>
               <message_count>1</message_count>
               <message>
                  <message_number>1</message_number>
                  <message_header>
                     <message_id>12233435454656567</message_id>
                     <correlation>TKAXAP01</correlation>
                     <delay>0</delay>
                     <priority>1</priority>
                     <message_state>0</message_state>
                     <sender_id>
                        <agent_name>scott</agent_name>
                     </sender_id>
                  </message_header>
                  <message_payload>
                     <jms_text_message>
                        <oracle_jms_properties>
                           <reply_to>
```

```
                        <agent_name>oracle</agent_name>
                        <address>redwoodshores</address>
                        <protocol>100</protocol>
                     </reply_to>
                     <userid>AQUSER</userid>
                     <appid>AQProduct</appid>
                     <groupid>AQ</groupid>
                     <timestamp>01-12-2000</timestamp>
                     <recv_timestamp>12-12-2000</recv_timestamp>
                  </oracle_jms_properties>
                  <user_properties>
                     <property>
                        <name>Country</name>
                        <string_value>USA</string_value>
                     </property>
                     <property>
                        <name>State</name>
                        <string_value>California</string_value>
                     </property>
                  </user_properties>
                  <text_data>All things bright and beautiful</text_data>
               </jms_text_message>
            </message_payload>
         </message>
      </message_set>
   </receive_result>
</AQXmlReceiveResponse>
</Body>
</Envelope>
```

登録リクエストへのサーバー・レスポンス

REGISTER リクエストへのレスポンスには、AQXmlRegisterResponse メソッドを使用します。このメソッドは、status_response で構成されます（status_response の詳細は、表 17-7 を参照）。

コミット・レスポンス

コミット・リクエストへのレスポンスには、AQXmlCommitResponse メソッドを使用します。このメソッドは、status_response で構成されます（status_response の詳細は、表 17-7 を参照）。

例

COMMIT リクエストへのレスポンスのフォーマットは、次のようになります。

```
<?xml version = '1.0'?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
```

```
   <Body>
      <AQXmlCommitResponse xmlns="http://ns.oracle.com/AQ/schemas/access">
         <status_response>
            <status_code>0</status_code>
         </status_response>
      </AQXmlCommitResponse>
   </Body>
</Envelope>
```

ロールバック・レスポンス

ロールバック・リクエストへのレスポンスには、AQXmlRollbackResponse メソッドを使用します。このメソッドは、status_response で構成されます。（status_response の詳細は、表 17-7 を参照）。

通知

クライアントが登録したイベントが発生した場合、REGISTER リクエストで指定した URL にクライアントへの通知が送信されます。AQXmlNotification は、次の要素で構成されます。

- 次の要素で構成される notification_options
  - destination - イベントが発生した宛先キュー / トピック
  - consumer_name - イベントが発生したコンシューマ名（マルチ・コンシューマ・キュー / トピックの場合）
- message_set - メッセージ・プロパティの集合

エラー発生時のレスポンス

前述のいずれかのリクエストでエラーが発生した場合、FAULT 要素が生成されます。FAULT 要素は、次の要素で構成されます。

- faultcode - フォールトのエラー・コード。
- faultstring - クライアント・エラーまたはサーバー・エラー。クライアント・エラーは、リクエストが無効であることを示します。サーバー・エラーは、AQ サーブレットが正しく設定されていないことを示します。
- 次の要素で構成される detail
  - status_response

例

FAULT メッセージのフォーマットは、次のようになります。

```
<?xml version = '1.0'?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
   <Body>
      <Fault xmlns="http://schemas.xmlsoap.org/soap/envelope/">
         <faultcode>100</faultcode>
         <faultstring>Server Fault</faultstring>
         <detail>
            <status_response>
               <status_code>-1</status_code>
               <error_code>410</error_code>
               <error_message>JMS-410: XML SQL Excetpion
ORA-24031: invalid value, OWNER_NAME should be non-NULL
ORA-06512: at "SYS.DBMS_AQJMS", line 177
ORA-06512: at line 1
</error_message>
            </status_response>
         </detail>
      </Fault>
   </Body>
</Envelope>
```

SOAP スキーマおよび AQ XML スキーマ

iDAP は、SOAP スキーマおよび AQ XML スキーマをクライアントに公開します。送信されるすべてのドキュメントは、これらのスキーマに対して妥当性をチェックされます。

- SOAP スキーマ - `http://schemas.xmlsoap.org/soap/envelope`
- AQ XML スキーマ - `http://ns.oracle.com/AQ/schemas/access`

SOAP スキーマ

SOAP スキーマは、エンベロープ、ヘッダーおよび本体で構成されるドキュメントの構造を記述します。

```
<?xml version='1.0'?>
<!-- XML Schema for SOAP v 1.1 Envelope -->
<schema xmlns='http://www.w3.org/2001/XMLSchema'
        xmlns:tns='http://schemas.xmlsoap.org/soap/envelope/'
        targetNamespace='http://schemas.xmlsoap.org/soap/envelope/'>

  <!-- SOAP envelope, header and body -->

  <element name="Envelope" type="tns:Envelope"/>
```

```
<complexType name='Envelope'>
 <sequence>
  <element ref='tns:Header' minOccurs='0'/>
  <element ref='tns:Body' minOccurs='1'/>
  <any minOccurs='0' maxOccurs='*'/>
 </sequence>
  <anyAttribute/>
</complexType>

<element name="Header" type="tns:Header"/>
<complexType name='Header'>
 <sequence>
  <any minOccurs='0' maxOccurs='*'/>
 </sequence>
  <anyAttribute/>
</complexType>

<element name="Body" type="tns:Body"/>
<complexType name='Body'>
 <sequence>
  <any minOccurs='0' maxOccurs='*'/>
 </sequence>
  <anyAttribute/>
</complexType>

<!-- Global Attributes.  The following attributes are intended
     to be usable via qualified attribute names on any complex type
     referencing them.  -->

<attribute name="mustUnderstand" type="tns:mutype" use="optional" value="0"/>
</attribute>

<simpleType  name="mutype">
   <restriction base="string">
<enumeration value="0"/>
      <enumeration value="1"/>
   </restriction>
</simpleType>

<attribute name='actor' type='anyURI'/>

<!-- 'encodingStyle' indicates any canonicalization conventions followed
     in the contents of the containing element.  For example, the value
     'http://schemas.xmlsoap.org/soap/encoding/' indicates
     the pattern described in SOAP specification. -->

<simpleType name='encodingStyle'>
```

```
    <list itemType='anyURI'/>
  </simpleType>
  <attributeGroup name='encodingStyle'>
    <attribute name='encodingStyle' type='tns:encodingStyle'/>
  </attributeGroup>

  <!-- SOAP fault reporting structure -->
  <complexType name='Fault' final='extension'>
   <sequence>
    <element name='faultcode' type='QName'/>
    <element name='faultstring' type='string'/>
    <element name='faultactor' type='anyURI' minOccurs='0'/>
    <element name='detail' type='tns:detail' minOccurs='0'/>
   </sequence>
  </complexType>

  <complexType name='detail'>
   <sequence>
    <any minOccurs='0' maxOccurs='*'/>
   </sequence>
    <anyAttribute/>
  </complexType>

</schema>
```

iDAP スキーマ

iDAP スキーマは、AQ 機能にインターネット・アクセスするための iDAP 本体の内容を記述します。

```
<?xml version="1.0"?>

<!-- ****************** AQ xml schema  ****************** -->

<schema xmlns = "http://www.w3.org/2001/XMLSchema"
        targetNamespace = "http://ns.oracle.com/AQ/schemas/access"
        xmlns:aq = "http://ns.oracle.com/AQ/schemas/access"
        xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

<import namespace = "http://schemas.xmlsoap.org/soap/envelope/"
        schemaLocation = "soap_env.xsd" />



<!-- ****************** AQ xml client operations  ****************** -->

   <element name="AQXmlSend">
      <complexType mixed="true">
        <sequence>
           <element ref="aq:producer_options" minOccurs="1" maxOccurs="1" />
           <element ref="aq:message_set" minOccurs="1" maxOccurs="1"/>
           <element ref="aq:AQXmlCommit" minOccurs="0" maxOccurs="1"/>
        </sequence>
      </complexType>
    </element>

   <element name="AQXmlPublish">
      <complexType mixed="true">
        <sequence>
           <element ref="aq:producer_options" minOccurs="1" maxOccurs="1" />
           <element ref="aq:message_set" minOccurs="1" maxOccurs="1"/>
           <element ref="aq:AQXmlCommit" minOccurs="0" maxOccurs="1"/>
        </sequence>
      </complexType>
    </element>


   <element name="AQXmlReceive">
      <complexType mixed="true">
        <sequence>
           <element ref="aq:consumer_options" minOccurs="1" maxOccurs="1" />
           <element ref="aq:AQXmlCommit" minOccurs="0" maxOccurs="1"/>
        </sequence>
```

```
   </complexType>
</element>


<element name="AQXmlRegister">
   <complexType mixed="true">
     <sequence>
        <element ref="aq:register_options" minOccurs="1" maxOccurs="1" />
        <element ref="aq:AQXmlCommit" minOccurs="0" maxOccurs="1"/>
     </sequence>
   </complexType>
</element>

<element name="AQXmlCommit">
   <complexType>
   </complexType>
</element>

<element name="AQXmlRollback">
   <complexType>
   </complexType>
</element>



<!-- ****************** AQ xml server responses  ****************** -->

<element name="AQXmlSendResponse">
   <complexType mixed="true">
     <sequence>
        <element ref="aq:status_response" minOccurs="1" maxOccurs="1"/>
        <element ref="aq:send_result" minOccurs="0" maxOccurs="1"/>
     </sequence>
   </complexType>
 </element>

<element name="AQXmlPublishResponse">
   <complexType mixed="true">
     <sequence>
        <element ref="aq:status_response" minOccurs="1" maxOccurs="1"/>
        <element ref="aq:publish_result" minOccurs="0" maxOccurs="1"/>
     </sequence>
   </complexType>
 </element>


<element name="AQXmlReceiveResponse">
```

```
   <complexType mixed="true">
     <sequence>
        <element ref="aq:status_response" minOccurs="1" maxOccurs="1"/>
        <element ref="aq:receive_result" minOccurs="0" maxOccurs="1"/>
     </sequence>
   </complexType>
 </element>


<element name="AQXmlRegisterResponse">
   <complexType mixed="true">
     <sequence>
        <element ref="aq:status_response" minOccurs="1" maxOccurs="1"/>
     </sequence>
   </complexType>
 </element>

<element name="AQXmlCommitResponse">
   <complexType mixed="true">
     <sequence>
        <element ref="aq:status_response" minOccurs="1" maxOccurs="1"/>
     </sequence>
   </complexType>
 </element>

<element name="AQXmlRollbackResponse">
   <complexType mixed="true">
     <sequence>
        <element ref="aq:status_response" minOccurs="1" maxOccurs="1"/>
     </sequence>
   </complexType>
 </element>


<element name="destination">
  <complexType>
   <simpleContent>
     <extension base='string'>
       <attribute name="lookup_type" type="aq:dest_lookup_type"
        default="DATABASE"/>
     </extension>
  </simpleContent>
</complexType>
</element>

<!-- **** destination lookup type ******* -->
<!-- lookup_type can be specified to either lookup LDAP or use  -->
```

```
<simpleType name="dest_lookup_type">
 <restriction base="string">
   <enumeration value="DATABASE"/>
   <enumeration value="LDAP"/>
  </restriction>
 </simpleType>

<!-- ***************** Producer Options ***************** -->
<element name="producer_options">
   <complexType mixed="true">
     <sequence>
     <element ref="aq:destination" minOccurs="1" maxOccurs="1"/>
     <element ref="aq:visibility" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:transformation" minOccurs="0" maxOccurs="1"/>
     </sequence>
   </complexType>
 </element>

<!-- ***************** Consumer Options ***************** -->
<element name="consumer_options">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:destination" minOccurs="1" maxOccurs="1"/>
     <element ref="aq:consumer_name" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:wait_time" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:selector" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:batch_size" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:visibility" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:dequeue_mode" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:navigation_mode" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:transformation" minOccurs="0" maxOccurs="1"/>
     </sequence>
    </complexType>
 </element>


<!-- ***************** Register Options ***************** -->
<element name="register_options">
   <complexType mixed="true">
     <sequence>
     <element ref="aq:destination" minOccurs="1" maxOccurs="1"/>
     <element ref="aq:consumer_name" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:notify_url" minOccurs="1" maxOccurs="1"/>
     </sequence>
   </complexType>
 </element>
```

```
   <element name="recipient_list">
     <complexType mixed="true">
      <sequence>
<element ref="aq:recipient" minOccurs="1" maxOccurs="*"/>
      </sequence>
     </complexType>
   </element>


   <!-- ***************** Message Set ************************ -->
   <element name="message_set">
      <complexType mixed="true">
       <sequence>
        <element ref="aq:message_count" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:message" minOccurs="0" maxOccurs="*"/>
       </sequence>
      </complexType>
   </element>


   <!-- ***************** Message ************************ -->
   <element name="message">
      <complexType mixed="true">
       <sequence>
        <element ref="aq:message_number" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:message_header" minOccurs="1" maxOccurs="1"/>
        <element ref="aq:message_payload" minOccurs="0" maxOccurs="1"/>
       </sequence>
      </complexType>
   </element>


   <!-- ***************** Message header ****************** -->
   <element name="message_header">
      <complexType mixed="true">
       <sequence>
        <element ref="aq:message_id" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:correlation" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:delay" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:expiration" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:priority" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:delivery_count" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:sender_id" minOccurs="1" maxOccurs="1"/>
        <element ref="aq:recipient_list" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:message_state" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:exception_queue" minOccurs="0" maxOccurs="1"/>
      </sequence>
```

```
  </complexType>
 </element>


<!-- ***************** Oracle JMS properties ***************** -->
<element name="oracle_jms_properties">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:type" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:reply_to" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:userid" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:appid" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:groupid" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:group_sequence" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:timestamp" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:recv_timestamp" minOccurs="0" maxOccurs="1"/>
    </sequence>
   </complexType>
 </element>


<!-- ***************** Message payload ***************** -->
<element name="message_payload">
  <complexType>
     <choice>
       <element ref="aq:raw" minOccurs="0" maxOccurs="1"/>
       <element ref="aq:jms_text_message" minOccurs="0" maxOccurs="1"/>
       <element ref="aq:jms_map_message" minOccurs="0" maxOccurs="1"/>
       <element ref="aq:jms_bytes_message" minOccurs="0" maxOccurs="1"/>
       <element ref="aq:jms_object_message" minOccurs="0" maxOccurs="1"/>
  <any minOccurs="0" maxOccurs="*" processContents="skip"/>
     </choice>
  </complexType>
</element>


<!-- ***************** User-defined properties ***************** -->
<element name="user_properties">
   <complexType mixed="true">
     <sequence>
     <element ref="aq:property" minOccurs="0" maxOccurs="*"/>
     </sequence>
   </complexType>
</element>

<!-- ***************** Property  ***************** -->
<element name="property">
```

```
   <complexType mixed="true">
     <sequence>
 <element ref="aq:name" minOccurs="1" maxOccurs="1"/>
 <choice>
   <element ref="aq:int_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:string_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:long_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:double_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:boolean_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:float_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:short_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:byte_value" minOccurs="1" maxOccurs="1"/>
 </choice>
     </sequence>
   </complexType>
 </element>


<!-- ***************** Status response ***************** -->
<element name="status_response">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:acknowledge" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:status_code" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:error_code" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:error_message" minOccurs="0" maxOccurs="1"/>
    </sequence>
   </complexType>
</element>


<!-- ***************** Send result ***************** -->
<element name="send_result">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:destination" minOccurs="1" maxOccurs="1"/>
     <element ref="aq:message_id" minOccurs="0" maxOccurs="*"/>
   </sequence>
   </complexType>
</element>

<!-- ***************** Publish result ***************** -->
<element name="publish_result">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:destination" minOccurs="1" maxOccurs="1"/>
     <element ref="aq:message_id" minOccurs="0" maxOccurs="*"/>
```

```
    </sequence>
   </complexType>
</element>


<!-- ***************** Receive result ***************** -->
<element name="receive_result">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:destination" minOccurs="1" maxOccurs="1"/>
     <element ref="aq:message_set" minOccurs="0" maxOccurs="*"/>
    </sequence>
   </complexType>
</element>


<!-- ***************** Notification ************************** -->
<element name="notification_options">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:destination" minOccurs="1" maxOccurs="1"/>
     <element ref="aq:consumer_name" minOccurs="1" maxOccurs="1"/>
    </sequence>
   </complexType>
</element>


<element name="priority" type="integer"/>
<element name="expiration" type="integer"/>
<element name="consumer_name" type="string"/>
<element name="wait_time" type="integer"/>
<element name="batch_size" type="integer"/>

<element name="notify_url" type="string"/>
<element name="message_id" type="string"/>
<element name="message_state" type="string"/>


<element name="message_number" type="integer"/>
<element name="message_count" type="integer"/>

<element name="correlation" type="string"/>
<element name="delay" type="integer"/>
<element name="delivery_count" type="integer"/>
<element name="exception_queue" type="string"/>
<element name="agent_alias" type="string"/>
```

```
<element name="type" type="string"/>
<element name="userid" type="string"/>
<element name="appid" type="string"/>
<element name="groupid" type="string"/>
<element name="group_sequence" type="integer"/>
<element name="timestamp" type="date"/>
<element name="recv_timestamp" type="date"/>

<element name="recipient">
  <complexType>
    <choice>
      <sequence>
        <element ref="aq:agent_name" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:address" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:protocol" minOccurs="0" maxOccurs="1"/>
      </sequence>
      <element ref="aq:agent_alias" minOccurs="1" maxOccurs="1"/>
    </choice>
  </complexType>
</element>


<element name="sender_id">
  <complexType>
    <choice>
      <sequence>
        <element ref="aq:agent_name" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:address" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:protocol" minOccurs="0" maxOccurs="1"/>
      </sequence>
      <element ref="aq:agent_alias" minOccurs="1" maxOccurs="1"/>
    </choice>
  </complexType>
</element>


<element name="reply_to">
  <complexType>
    <choice>
      <sequence>
        <element ref="aq:agent_name" minOccurs="1" maxOccurs="1"/>
     <element ref="aq:address" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:protocol" minOccurs="0" maxOccurs="1"/>
      </sequence>
      <element ref="aq:agent_alias" minOccurs="1" maxOccurs="1"/>
    </choice>
  </complexType>
```

```
   </element>


   <element name="selector">
     <complexType>
<choice>
          <element ref="aq:correlation" minOccurs="0" maxOccurs="1"/>
          <element ref="aq:message_id" minOccurs="0" maxOccurs="1"/>
          <element ref="aq:condition" minOccurs="0" maxOccurs="1"/>
        </choice>
     </complexType>
   </element>


   <element name="condition" type="string"/>

   <element name="visibility">
    <simpleType>
     <restriction base="string">
      <enumeration value="ON_COMMIT"/>
      <enumeration value="IMMEDIATE"/>
     </restriction>
    </simpleType>
    </element>

   <simpleType name="del_mode_type">
     <restriction base="string">
      <enumeration value="PERSISTENT"/>
      <enumeration value="NONPERSISTENT"/>
     </restriction>
    </simpleType>

   <element name="dequeue_mode">
   <simpleType>
    <restriction base="string">
      <enumeration value="BROWSE"/>
      <enumeration value="LOCKED"/>
      <enumeration value="REMOVE"/>
      <enumeration value="REMOVE_NODATA"/>
    </restriction>
    </simpleType>
   </element>

   <element name="navigation_mode">
   <simpleType>
    <restriction base="string">
      <enumeration value="FIRST_MESSAGE"/>
```

```
   <enumeration value="NEXT_MESSAGE"/>
   <enumeration value="NEXT_TRANSACTION"/>
 </restriction>
 </simpleType>
</element>

<element name="transformation" type="string"/>

<element name="acknowledge">
  <complexType>
  </complexType>
</element>
<element name="status_code" type="string"/>
<element name="error_code" type="string"/>
<element name="error_message" type="string"/>

<element name="name" type="string"/>
<element name="int_value" type="integer"/>
<element name="string_value" type="string"/>
<element name="long_value" type="long"/>
<element name="double_value" type="double"/>
<element name="boolean_value" type="boolean"/>
<element name="float_value" type="float"/>
<element name="short_value" type="short"/>
<element name="byte_value" type="byte"/>

<element name="agent_name" type="string"/>
<element name="address" type="string"/>
<element name="protocol" type="integer"/>


<!-- ***************** RAW message ********************** -->
<element name="raw" type="string"/>


<!-- ***************** JMS text message ***************** -->
<element name="jms_text_message">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:oracle_jms_properties" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:user_properties" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:text_data" minOccurs="1" maxOccurs="1"/>
   </sequence>
   </complexType>
</element>

<element name="text_data" type="string"/>
```

```
<!-- ***************** JMS map message ***************** -->
<element name="jms_map_message">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:oracle_jms_properties" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:user_properties" minOccurs="0" maxOccurs="1"/>
     <element ref="aq:map_data" minOccurs="1" maxOccurs="1"/>
    </sequence>
   </complexType>
</element>

<!-- ***************** Map data ***************** -->
<element name="map_data">
   <complexType mixed="true">
     <sequence>
     <element ref="aq:item" minOccurs="0" maxOccurs="*"/>
     </sequence>
   </complexType>
</element>

<!-- ***************** Map Item  ***************** -->
<element name="item">
   <complexType mixed="true">
     <sequence>
 <element ref="aq:name" minOccurs="1" maxOccurs="1"/>
 <choice>
   <element ref="aq:int_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:string_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:long_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:double_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:boolean_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:float_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:short_value" minOccurs="1" maxOccurs="1"/>
   <element ref="aq:byte_value" minOccurs="1" maxOccurs="1"/>
 </choice>
     </sequence>
   </complexType>
 </element>


<!-- ***************** JMS bytes message ***************** -->
<element name="jms_bytes_message">
   <complexType mixed="true">
    <sequence>
     <element ref="aq:oracle_jms_properties" minOccurs="0" maxOccurs="1"/>
```

```
        <element ref="aq:user_properties" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:bytes_data" minOccurs="1" maxOccurs="1"/>
       </sequence>
      </complexType>
   </element>

   <element name="bytes_data" type="string"/>


   <!-- ***************** JMS object message ***************** -->
   <element name="jms_object_message">
      <complexType mixed="true">
        <sequence>
        <element ref="aq:oracle_jms_properties" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:user_properties" minOccurs="0" maxOccurs="1"/>
        <element ref="aq:ser_object_data" minOccurs="1" maxOccurs="1"/>
        </sequence>
      </complexType>
   </element>

   <element name="ser_object_data" type="string"/>

</schema>
```

AQ XML サーブレットの配置

AQ XML サーブレットは、oracle.AQ.xml.AQxmlServlet クラスを拡張する Java クラスです。AQxmlServlet クラスは、javax.servlet.http.HttpServlet クラスを拡張します。

注意： AQ XML サーブレットのデモは、$ORACLE_HOME/rdbms/demo/ にあります。詳細は、aqxmlREADME.txt を参照してください。

AQ XML サーブレットでは、Content-Type ヘッダーが text/xml または application/x-www-form-urlencoded であるリクエストが受け入れられます。リクエストの Content-Type ヘッダーが application/x-www-form-urlencoded に設定されている場合、パラメータ名を aqxmldoc に設定する必要があります。また、その値は URL でエンコードされた AQ XML 文書である必要があります。

AQ XML サーブレット・クラスの作成

AQ サーブレットは、Oracle9*i* サーバーに接続するための JDBC OCI コネクション・プールを作成します。サーブレットの init() メソッドに、データベース接続パラメータ、ユーザー名およびパスワードをカプセル化する AQxmlDataSource オブジェクトを指定する必要があります。AQxmlDataSource クラスの詳細は、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

AQxmlDataSource に指定したユーザーは、AQ サーブレットのスーパー・ユーザーになります。このユーザーには、DBMS_AQIN パッケージの実行権限と CREATE SESSION 権限が必要です。

例：

次のように、AQ サーブレットのスーパー・ユーザーとしてユーザー AQADM を作成します。

```
connect sys/change_on_install as sysdba;
grant connect, resource to aqadm identified by aqadm;
grant create session to aqadm;
grant execute on dbms_aqjms to aqadm;
```

このスーパー・ユーザーを使用して、次のようにサンプル・サーブレットを作成できます。

```
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.AQ.xml.*;

/**
 * This is a sample AQ Servlet.
 */
public class AQTestServlet extends oracle.AQ.xml.AQxmlServlet
```

```
{

  /* The init method must be overloaded to specify the AQxmlDataSource */
  public void init()
  {
      AQxmlDataSource  db_drv = null;

      try
      {
        /* Create data source with username, password, sid, host, port */
        db_drv = new AQxmlDataSource("AQADM", "AQADM", "test_db", "sun-248",
"5521");

        this.setAQDataSource(db_drv);
      }
      catch (Exception ex)
      {
          System.out.println("Exception in init: " + ex);
      }
}
```

スーパークラス oracle.AQ.xml.AQxmlServlet は、javax.servlet.http.HttpServlet に doPost() メソッドおよび doGet() メソッドを実装します。doPost() メソッドは、受信した SOAP リクエストを処理し、要求された AQ 操作を実行します。

> **注意：** 例では、AQ サーブレットが、Javasoft 社製 Servlet2.2 仕様（Tomcat 3.1 など）を実装する Web サーバーにインストールされていることを想定しています。Servlet 2.0 仕様（Apache Jserv など）を実装する Web サーバーの場合、AQxmlServlet クラスではなく oracle.AQ.xml.AQxmlServlet20 クラスを拡張し、適切な write() メソッドをオーバーライドしてください。

AQ XML サーブレットのコンパイル

AQ サーブレットは、Javasoft 社製 Servlet2.0 または Servlet2.2 インタフェース（Apache Jserv、Tomcat など）を実装するすべての Web サーバーまたはサーブレット・コンテナを使用して配置できます。次のことを考慮してください。

- サーブレットは JDBC OCI ドライバを使用して Oracle9*i* サーバーに接続するため、サーブレットのホスト・マシンに Oracle9*i* クライアント・ライブラリをインストールし、LD_LIBRARY_PATH に $ORACLE_HOME/lib を含める必要があります。
- サーブレットは、JDK 1.1.x または JDK 1.2.x のライブラリを使用してコンパイルできます。

- JDK 1.1.x の場合、CLASSPATH に次のパスを含める必要があります。

```
$ORACLE_HOME/jdbc/lib/classes111.zip
$ORACLE_HOME/jdbc/lib/jta.zip
$ORACLE_HOME/jdbc/lib/nls_charset11.zip
$ORACLE_HOME/jdbc/lib/jndi.zip
$ORACLE_HOME/lib/lclasses11.zip
$ORACLE_HOME/lib/xmlparserv2.jar
$ORACLE_HOME/lib/xschema.jar
$ORACLE_HOME/rdbms/jlib/aqapi11.jar
$ORACLE_HOME/rdbms/jlib/jmscommon.jar
$ORACLE_HOME/rdbms/jlib/aqxml.jar
$ORACLE_HOME/rdbms/jlib/xsu111.jar
$ORACLE_HOME/jis/lib/servlet.jar
```

- JDK 1.2.x の場合、CLASSPATH に次のパスを含める必要があります。

```
$ORACLE_HOME/jdbc/lib/classes12.zip
$ORACLE_HOME/jdbc/lib/jta.zip
$ORACLE_HOME/jdbc/lib/nls_charset12.zip
$ORACLE_HOME/jdbc/lib/jndi.zip
$ORACLE_HOME/lib/lclasses12.zip
$ORACLE_HOME/lib/xmlparserv2.jar
$ORACLE_HOME/lib/xschema.jar
$ORACLE_HOME/rdbms/jlib/aqapi.jar
$ORACLE_HOME/rdbms/jlib/jmscommon.jar
$ORACLE_HOME/rdbms/jlib/aqxml.jar
$ORACLE_HOME/rdbms/jlib/xsu12.jar
$ORACLE_HOME/jis/lib/servlet.jar
```

- CLASSPATH の設定後、javac またはその他の Java コンパイラを使用して、サーブレットをコンパイルします。

注意： JDK1.2 のリリース 1.2.2_05a 以上で AQ XML サーブレットまたは AQ JMS API を使用している場合は、JIT コンパイラをオフにする必要があります。マルチスレッド・アプリケーションで問題を回避するには、JAVA_COMPILER = none を設定します。

ユーザー認証

サーブレットのインストール後、AQ サーブレットに POST リクエストを送信するすべてのユーザーを認証するように Web サーバーを構成する必要があります。AQ サーブレットは、認証されたユーザーのみにサーブレットへのアクセスを許可します。ユーザーが認証されていない場合、サーブレットからエラーが戻ります。

Web サーバーを構成してアクセスを制限するには、複数の方法があります。一般的な方法は、Secure Sockets Layer（SSL）上で Basic 認証（ユーザー名およびパスワード）を行う方法およびクライアント証明書を使用する方法です。サーブレットへのアクセスを制限する方法については、Web サーバーのドキュメントを参照してください。

HTTP の使用

AQ サーブレットのコンテキストでは、Web サーバーに接続するために使用されるユーザー名は、AQ の HTTP エージェントまたは AQ のインターネット・ユーザーとして認識されます。

例：

Apache では、次の構文を使用して、aqserv/servlet にインストールされたサーブレットへのアクセスを制限できます（Basic 認証を使用）。この例では、サーブレットに POST リクエストを送信するすべてのユーザーが、/apache/htdocs/userdb 内の users ファイルを使用して認証されます。

```
<Location /aqserv/servlet>
  <Limit POST>
    AuthName "AQ restricted stuff"
    AuthType Basic
    AuthUserFile /apache/htdocs/userdb/users
    require valid-user
  </Limit>
</Location>
```

ユーザー認可

AQ サーブレットに接続するユーザーを認証した後、次の手順を実行して、ユーザーに実行権限を付与します。

1. インターネット・アクセスのための AQ エージェントを登録します。
2. 1 つ以上のデータベース・ユーザーに AQ エージェントをマップします。

AQ エージェントの登録

インターネット・アクセスのための AQ エージェントを登録するには、DBMS_AQADM.CREATE_AQ_AGENT を使用します。CREATE_AQ_AGENT プロシージャは、agent_name を取ります。サーブレットにアクセスするためにユーザーが使用するプロトコルを、HTTP に指定します。

例 :

HTTP を使用して、AQ サーブレットにアクセスする AQ エージェント JOHN を作成します。

```
DBMS_AQADM.CREATE_AQ_AGENT(agent_name => 'JOHN', enable_http => true);
```

AQ エージェントを変更および削除するためのプロシージャ ALTER_AQ_AGENT および DROP_AQ_AGENT は、CREATE_AQ_AGENT と同様に機能します。これらのプロシージャの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

データベース・ユーザーへの AQ エージェントのマップ

AQ エージェントを 1 つ以上のデータベース・ユーザーにマップするには、DBMS_AQADM.ENABLE_DB_ACCESS を使用します。ENABLE_DB_ACCESS プロシージャを使用して、AQ エージェントに特定のデータベース・ユーザーの権限を付与します。これによって、エージェントは、そのエージェントがマップされたデータベース・ユーザーが参照できるすべてのキューにアクセスできるようになります。

例 :

AQ インターネット・エージェント JOHN をデータベース・ユーザー OE（海外向け出荷）および CBADM（顧客請求情報の管理者）にマップします。

```
DBMS_AQADM.ENABLE_DB_ACCESS(agent_name =>'JOHN', db_username => 'OE');
DBMS_AQADM.ENABLE_DB_ACCESS(agent_name =>'JOHN', db_username => 'CBADM');
```

データベース・セッション

ユーザーがサーブレットに POST リクエストを送信すると、サーブレットがリクエストを解析し、ユーザーがアクセスしようとしているキュー / トピックを判断します。それによって、AQ サーブレットは、AQ エージェントにマップするデータベース・ユーザー（db_user）の 1 人として、データベース・セッションを作成します。選択された db_user は、リクエストに指定されたキューへのアクセス権限を取得します。

例：
AQ エージェント JOHN が、OE.OE_NEW_ORDERS_QUE にエンキュー・リクエストを送信します。サーブレットは、JOHN が db_user OE および CBADM にマップできることを確認します。OE.OE_NEW_ORDERS_QUE は、OE スキーマ内に存在するため、OE として CREATE SESSION を行い、要求された操作を実行します。

AQ サーブレットは、AQ サーブレットのスーパー・ユーザーを使用して、Oracle サーバーへのコネクション・プールを作成します。このスーパー・ユーザーは、AQ インターネット・エージェントがマップする db_user のかわりにセッションを作成します。したがって、スーパー・ユーザーには、ENABLE_DB_ACCESS コールに指定されたすべてのユーザー用のプロキシ・セッションを作成する権限が必要です。AQ サーブレットのスーパー・ユーザーの作成方法は、17-49 ページの「AQ XML サーブレット・クラスの作成」を参照してください。

AQ サーブレットのスーパー・ユーザーには、次のようにプロキシ・セッションを作成する権限を付与できます。

```
connect sys/change_on_install as sysdba
rem grant super-user AQADM privileges to create proxy sessions as OE
alter user OE grant CONNECT THROUGH AQADM;

rem grant super-user AQADM privileges to create proxy sessions as CBADM
alter user CBADM grant CONNECT THROUGH AQADM;
```

AQ インターネット・エージェントが 2 つ以上の db_user にマップされている場合、すべての db_user に FORCE ANY TRANSACTION 権限が必要です。

```
grant FORCE ANY TRANSACTION to OE;
grant FORCE ANY TRANSACTION to CBADM;
```

エージェントとデータベース・ユーザー間のマッピングを使用禁止にするには、DBMS_AQADM.DISABLE_DB_ACCESS を使用します。

SYSTEM.AQ$INTERNET_USERS ビューには、AQ エージェント、エージェントが使用できるプロトコル、および AQ エージェントとデータベース・ユーザー間のマッピングがリスト表示されます。表 17-8 に、このビューに表示されるエントリの例を示します。

表 17-8 SYSTEM.AQ$INTERNET_USERS ビュー

agent_name	db_username	http_enabled
scott	cbadmin	YES
scott	buyer	YES
aqadmin	OE	YES
aqadmin	seller	YES
bookstore	-	NO

AQ XML サーブレットとの LDAP の使用

次の場合、LDAP サーバーが必要です。

- lookup_type 宛先属性が LDAP と指定されている場合。この場合、宛先名は LDAP サーバーを使用して schema.queue_name に変換されます。
- (agent_name, address, protocol) ではなく agent_alias を使用している場合。agent_alias は、クライアント・リクエストで指定されると、LDAP サーバーを使用して agent_name、address、protocol に変換されます。

LDAP コンテキストは、次のとおり setLDAPContext(DirContext) コールによって指定する必要があります。

```
public void init()
{
      Hashtable env = new Hashtable(5, 0.75f);
      AQxmlDataSource  db_drv = null;

      try
      {
           /* Create data source with username, password, sid, host, port */
           db_drv = new AQxmlDataSource("AQADM", "AQADM", "test_db",
                                        "sun-248", "5521");
           this.setAQDataSource(db_drv);

           env.put(Context.INITIAL_CONTEXT_FACTORY,
                   "com.sun.jndi.ldap.LdapCtxFactory");
           env.put(Context.PROVIDER_URL, "ldap://yow:389");
           env.put(SEARCHBASE, "cn=server1,cn=dbservers,cn=wei");
           env.put(Context.SECURITY_AUTHENTICATION, "simple");
           env.put(Context.SECURITY_PRINCIPAL, "cn=orcladmin");
           env.put(Context.SECURITY_CREDENTIALS, "welcome");

           DirContext inictx = new InitialDirContext(env);
           String searchbase = (String)env.get("server_dn");
```

```
            lctx = (DirContext)inictx.lookup(searchbase);

            // Set up LDAP context
            setLdapContext(lctx);

            // Set the EMAIL server address (if any)
            setEmailServerAddr("144.25.186.236");
        }
        catch (Exception ex)
        {
           System.err.println("Servlet init exception: " +ex) ;
        }
    }
```

HTTP を使用した AQ XML サーブレットへのアクセス

この項では、AQ クライアントが HTTP を使用して AQ サーブレットにリクエストを行う手順および AQ サーブレットがリクエストを処理する手順を示します。

HTTP を使用した AQ サーブレットへの AQ クライアント・リクエスト

1. クライアントがサーバーへの HTTP(S) コネクションをオープンします。

   次に例を示します。
   `https://aq.us.oracle.com:8000/aqserv/servlet/AQTestServlet`

   これによって、`aq.us.oracle.com` のポート 8000 へのコネクションがオープンされます。

2. クライアントが、次のいずれかの方法でサーバーにログインします。

   - HTTP の Basic 認証（SSL を使用する場合も、使用しない場合も含む）
   - SSL 証明書ベースのクライアント認証

3. クライアントが、`SEND`、`PUBLISH`、`RECEIVE` または `REGISTER` リクエストを表す XML メッセージを構成します。

   次に例を示します。

```
<?xml version="1.0"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
      <Body>

        <AQXmlSend  xmlns = "http://ns.oracle.com/AQ/schemas/access">
          <producer_options>
            <destination>OE.OE_NEW_ORDERS_QUE</destination>
          </producer_options>

          <message_set>
            <message_count>1</message_count>
            <message>
              <message_number>1</message_number>
              <message_header>
                 <correlation>XML_ADT_SINGLE_ENQ</correlation>
                 <sender_id>
                     <agent_name>john</agent_name>
                 </sender_id>
              </message_header>
              <message_payload>
              <ORDER_TYP>
                     <ORDERNO>100</ORDERNO>
                     <STATUS>NEW</STATUS>
                     <ORDERTYPE>NORMAL</ORDERTYPE>
```

```
                        <ORDERREGION>EAST</ORDERREGION>
                        <CUSTOMER>
                           <CUSTNO>1001233</CUSTNO>
                           <CUSTID>JOHN</CUSTID>
                           <NAME>AMERICAN EXPRESS</NAME>
                           <STREET>EXPRESS STREET</STREET>
                           <CITY>REDWOOD CITY</CITY>
                           <STATE>CA</STATE>
                           <ZIP>94065</ZIP>
                           <COUNTRY>USA</COUNTRY>
                        </CUSTOMER>
                        <PAYMENTMETHOD>CREDIT</PAYMENTMETHOD>
                        <ITEMS>
                           <ITEMS_ITEM>
                              <QUANTITY>10</QUANTITY>
                              <ITEM>
                                 <TITLE>Perl</TITLE>
                                 <AUTHORS>Randal</AUTHORS>
                                 <ISBN>ISBN20200</ISBN>
                                 <PRICE>19</PRICE>
                              </ITEM>
                              <SUBTOTAL>190</SUBTOTAL>
                           </ITEMS_ITEM>
                         </ITEMS>
                        <CCNUMBER>NUMBER01</CCNUMBER>
                        <ORDER_DATE>2000-08-23 0:0:0</ORDER_DATE>
                     </ORDER_TYP>
                   </message_payload>
                 </message>
               </message_set>
               </AQXmlSend>
          </Body>
</Envelope>
```

4. リモート・サーバーで、クライアントがサーブレットに HTTP の POST リクエストを送信します。

   HTTP を使用した POST リクエストのサンプル・コードは、$ORACLE_HOME/demo ディレクトリを参照してください。

HTTP を使用した AQ サーブレットによるリクエストの処理

1. サーバーが、クライアントの HTTP(S) コネクションを受け入れます。
2. サーバーが、クライアントによって指定されたユーザー（AQ エージェント）を認証します。
3. サーバーが POST リクエストを受信します。
4. AQ サーブレットが起動されます。

   このリクエストがこのサーブレットが処理する最初のリクエストである場合、サーブレットは初期化されます（サーブレットの init() メソッドが起動されます）。init() メソッドは、クライアントによって提供された AQxmlDataSource パラメータ（SID、ホスト、ポート、AQ サーブレットのスーパー・ユーザー名、パスワード）を使用して、Oracle サーバーへのコネクション・プールを作成します。
5. サーブレットが、次のようにメッセージを処理します。

   - このリクエストがこのクライアントからの最初のリクエストである場合、新しい HTTP セッションが作成されます。XML メッセージが解析され、メッセージの内容が検証されます。HTTP ヘッダーにクライアントによってセッション ID が渡される場合、この操作はそのセッションのコンテキストで実行されます。詳細は、次の項を参照してください。
   - サーブレットが、エージェントが操作を実行しようとしているオブジェクト（キューおよびトピック）を判断します。

     たとえば、クライアント・リクエスト（17-57 ページの「HTTP を使用した AQ サーブレットへの AQ クライアント・リクエスト」の手順 3）では、エージェント JOHN が
     OE.OE_NEW_ORDERS_QUE にアクセスしようとしています。
   - サーブレットが、（AQ$INTERNET_USERS ビューを使用して）この AQ エージェントにマップするデータベース・ユーザーのリスト全体を検索します。いずれかの db_user がリクエストに指定されたキュー / トピックへのアクセス権限を取得している場合、AQ サーブレットのスーパー・ユーザーは、この db_user のかわりにセッションを作成します。

     たとえば、エージェント JOHN が DBMS_AQADM.ENABLE_DB_ACCESS コールを使用してデータベース・ユーザー OE にマップされている場合、サーブレットはデータベース・ユーザー OE の権限でエージェント JOHN 用のセッションを作成します（ENABLE_DB_ACCESS の詳細は、17-53 ページの「データベース・ユーザーへの AQ エージェントのマップ」を参照）。
   - HTTP セッションにアクティブなトランザクションが存在しない場合は、新しいデータベース・トランザクションが開始されます。明示的な COMMIT または ROLLBACK リクエストが行われるまで、セッションの後続のリクエストは同じトランザクションの一部になります。

- 要求された操作（SEND、PUBLISH、RECEIVE、REGISTER、COMMIT、ROLLBACK）が実行されます。
- レスポンスが XML メッセージとしてフォーマットされ、クライアントに戻されます。

  たとえば、前述のリクエストに対するレスポンスは、次のとおりです。

```
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <AQXmlSendResponse xmlns="http://ns.oracle.com/AQ/schemas/access">
      <status_response>
        <status_code>0</status_code>
      </status_response>
      <send_result>
        <destination>OE.OE_NEW_ORDERS_QUE</destination>
        <message_id>12341234123412341234123412341234</message_id>
      </send_result>
    </AQXmlSendResponse>
  </Body>
</Envelope>
```

- レスポンスには、セッション ID が Cookie として HTTP ヘッダーに含まれます。たとえば、Tomcat はセッション ID を JSESSIONID=239454ds2343 として戻します。操作によってトランザクションがコミットされない場合、明示的なコミットまたはロールバックのコールを受信するまで、トランザクションはアクティブな状態のままになります。トランザクションは、そのトランザクションがコミットされるまで反映されません。トランザクションがアクティブでない状態が 120 秒間続くと、そのトランザクションは自動的に終了します。

ユーザー・セッションおよびトランザクション

クライアントが認証され、AQ サーブレットに接続すると、ユーザーのかわりに HTTP セッションが作成されます。そのセッションで初めてリクエストを行うと、新しいデータベース・トランザクションが暗黙的に開始します。このトランザクションは、明示的にコミットまたは終了するまで、オープン状態のままになります。サーブレットからのレスポンスには、HTTP ヘッダーにセッション ID が Cookie として含まれます。

クライアントが同一のトランザクションで作業を継続する場合、後続のリクエストにはセッション ID の Cookie を含むこの HTTP ヘッダーを含める必要があります。これは、ほとんどの Web ブラウザで自動的に行われます。ただし、Java または C のクライアントを使用してリクエストを転送する場合、プログラムで行う必要があります。$ORACLE_HOME/demo ディレクトリに、同じセッションの一部としてリクエストを転送するために使用する Java プログラムのサンプルがあります。

トランザクションを終了するには、明示的なコミットまたはロールバックを発行する必要があります。コミットまたはロールバックのリクエストは、他の AQ 操作（Send、Publish、Receive、Register）に組み込むこともできます。

各 HTTP セッションのデフォルトのタイムアウトは 120 秒です。ユーザーがそのセッションのリクエスト後、120 秒以内にトランザクションをコミットまたはロールバックしない場合、トランザクションは自動的に終了します。このタイムアウトは、`setSessionMaxInactiveTime()` を使用して、サーブレットの `init()` メソッドで変更できます。詳細は、17-64 ページの「AQ サーブレットのカスタマイズ」を参照してください。

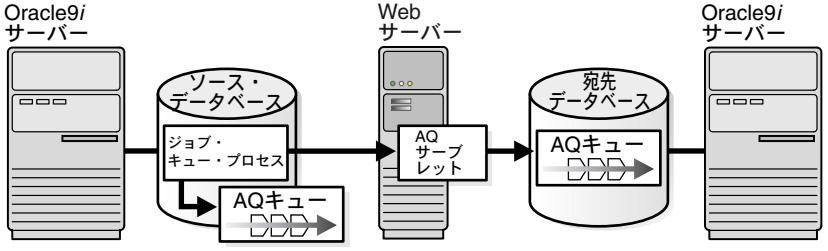
HTTP および HTTPS を使用したアドバンスト・キューイング伝播

Oracle9*i* のアドバンスト・キューイング伝播を使用すると、Oracle Net Services（以前の Net8）ではなく HTTP および HTTPS（SSL による HTTP）上で伝播できます。HTTP は、Oracle Net Services とは異なり、ファイアウォールを簡単に構成できます。

高水準アーキテクチャ

HTTP での AQ 伝播では、基礎として AQ へのインターネット・アクセスのインフラストラクチャを使用します。図 17-2 に示すように、伝播を行うバックグラウンド・プロセスが AQ サーブレットにメッセージを転送し、その AQ サーブレットが接続先データベースにメッセージをエンキューします。

図 17-2 HTTP でのアドバンスト・キューイング伝播



HTTP での伝播は、転送のみが Oracle Net Services と異なり、ほとんどの設定は Oracle Net Services での伝播と同じです。次の項では、追加の手順および相違点の概要を示します。

HTTP での伝播の設定（および Oracle Net での伝播との相違点）

1. ソース・データベースの dblink を別に作成する必要があります。接続文字列では、プロトコルに HTTP を指定し、AQ サーブレットを実行している Web サーバーのホストおよびポートを指定します。dblink のユーザー名およびパスワードは、Web サーバー / サーブレット・コンテナでの認証に使用されます。
2. 接続先データベースに接続する AQ サーブレットを配置する必要があります。
3. Java および XML を実行するためにソース・データベースが使用可能である必要があります。

伝播のその他の手順は同じです。管理者は、dbms_aqadm.schedule_propagation を使用して伝播を開始する必要があります。伝播は、dbms_aqadm.disable_propagation_schedule で使用不可能にし、ms_aqadm.enable_propagation_schedule で再度使用可能にできます。バックグラウンド・プロセスであるジョブ・キュー・プロセスによって、接続先データベースにメッセージが伝播されます。伝播を行うには、2 つ以上の job_queue_processes パラメータが必要です。

次の項の手順 1 ～ 3 を実行すると、既存のコードを変更せずに、アプリケーションが HTTP での AQ 伝播を使用するように簡単に設定できます。同様に、HTTP での AQ 伝播を使用するアプリケーションは、Oracle Net Services 用の dblink を再作成するのみで、その他を変更せずに、再び Oracle Net Services での伝播に戻すことができます。

HTTP での AQ 伝播の設定

1. Java および XML を実行するためのソース・データベースを作成する必要があります。
2. dblink を作成します。プロトコルに HTTP を指定し、AQ サーブレットを実行している Web サーバーのホストおよびポート、および Web サーバー / サーブレット・コンテナでの認証用のユーザー名とパスワードを指定します。

   たとえば、Web サーバーが webdest.oracle.com のマシンで実行しており、ポート 8081 に対するリクエストをリスニングしている場合、データベースの接続文字列は次のようになります。

   ```
   (DESCRIPTION=(ADDRESS=(PROTOCOL=http)(HOST=webdest.oracle.com)(PORT=8081)))
   ```

   SSL を使用する場合、接続文字列に HTTPS をプロトコルとして指定します。

   データベース・リンクは次のように作成します。

   ```
   create public database link dba connect to john identified by welcome using
   '(DESCRIPTION=(ADDRESS=(PROTOCOL=http)(HOST=webdest.oracle.com)(PORT=8081))';
   ```

   この場合、パスワード Welcome を持つユーザー John が Web サーバーの認証に使用されます。このユーザーは AQ HTTP エージェントでもあります。

3. オプションで、データベースからのすべての HTTP リクエストにプロキシを使用できるように設定できます。『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』で説明されているとおり、`UTL_HTTP.SET_PROXY` プロシージャを使用します。

4. SSL による HTTP を使用する場合、ソース・データベースにデータベース Wallet を作成する必要があります。伝播の継続中は、Wallet をオープンしておく必要があります。伝播に HTTPS を使用する場合、ソース・データベースと AQ サーブレット間の通信は暗号化され、HTTPS サーバーはソース・データベースで認証されます。データベースは、データベース・リンクのユーザー名とパスワードを使用して、データベース自体を HTTPS サーバーで認証します。

5. AQ サーブレットを配置します。

   17-49 ページの「AQ XML サーブレット・クラスの作成」を参照して、`AQxmlServlet` を拡張する `AQPropServlet` クラスを作成します。このサーブレットは、接続先データベースに接続する必要があります。サーブレットは、Web サーバー上のパス `aqserv/servlet` に配置する必要があります。

   Oracle9*i* では、伝播サーブレット名は `AQPropServlet`、配置パスは aqserv/servlet に固定されています。

6. AQ HTTP エージェント（John）に AQ 操作の実行権限があることを確認します。これは接続先データベースで行います。

   a. 次のようにして AQ エージェントを登録します。

      ```
      dbms_aqadm.create_aq_agent(agent_name => 'John', enable_http => true);
      ```

   b. 次のようにして AQ エージェントをデータベース・ユーザーにマップします。

      ```
      dbms_aqadm.enable_db_access(agent_name =>'John', db_username =>'CBADM')'
      ```

7. 次のものをコールすることによってソース・サイトで伝播を開始します。

   ```
   dbms_aqdm.schedule_propagation.
   dbms_aqadm.schedule_propagation('src_queue', 'dba');
   ```

AQ サーブレットのカスタマイズ

oracle.AQ.xml.AqxmlServlet は、コネクション・プール・サイズ、セッション・タイムアウト、スタイル・シート、および AQ 操作前後のコールバックを設定するための API を提供します。

コネクション・プール・サイズの設定

AQ データ・ソースを使用して、AQ 操作を実行するためにサーブレットが接続するバックエンド・データベースを指定します。AQ データ・ソースには、データベースの SID、ホスト名、リスナー・ポート、および AQ サーブレットのスーパー・ユーザーのユーザー名およびパスワードが含まれます。

データ・ソースは AQxmlDataSource クラスで表現され、サーブレットの setAQDataSource メソッドを使用して設定できます。詳細は、『Oracle9*i* Java パッケージ・プロシージャ・リファレンス』を参照してください。

AQ データ・ソースでは、データベース・サーバーに対するコネクションのプールが作成されます。デフォルトでは、プールの最大サイズは 50、最小サイズは 1 に設定されます。プール内のコネクション数は、受信したリクエストの数に基づいて動的に増加および減少します。コネクション数の最大値を変更する場合は、AQxmlDataSource.setCacheSize(size) メソッドを使用してキャッシュ・サイズを指定する必要があります。

セッション・タイムアウトの設定

クライアントが認証され、AQ サーブレットに接続すると、ユーザーのかわりに HTTP セッションが作成されます。そのセッションで初めてリクエストを行うと、新しいデータベース・トランザクションが暗黙的に開始します。このトランザクションは、明示的にコミットまたは終了するまで、オープン状態のままになります。

各 HTTP セッションのデフォルトのタイムアウトは 120 秒です。ユーザーがそのセッションのリクエスト後、120 秒以内にトランザクションをコミットまたはロールバックしない場合、トランザクションは自動的に終了します。このタイムアウトは、setSessionMaxInactiveTime() メソッドを使用して、サーブレットの init() メソッドで変更できます。

サーブレットを次のようにして初期化します。

```
public class AQTestServlet extends oracle.AQ.xml.AQxmlServlet
{
  /* The init method must be overloaded to specify the AQxmlDataSource */
  public void init()
  {
      AQxmlDataSource  db_drv = null;

      try
```

```
      {
        /* Create data source with username, password, sid, host, port */
        db_drv = new AQxmlDataSource("AQADM", "AQADM",
                                     "test_db", "sun-248", "5521");

        /* Set the minimum cache size to 10 connections */
       db_drv.getCacheSize(10);

        this.setAQDataSource(db_drv);

        /* Set the transaction timeout to 180 seconds */
        this.setSessionMaxInactiveTime(180);
      }
      catch (Exception ex)
      {
          System.out.println("Exception in init: " + ex);
      }
}
```

サーブレットからのすべてのレスポンスに対するスタイル・シートの設定

AQ サーブレットは XML でレスポンスを戻します。サーブレットの管理者は、このサーブレットから戻されるすべてのレスポンスに設定するスタイル・シートを指定できます。これは、サーブレットの `init()` メソッドで `setStyleSheet(type,href)` または `setStyleSheetProcessingInstr(proc_instr)` を起動することによって行うことができます。

たとえば、すべてのレスポンスに次のスタイル・シートを含めるには、次の構文を入力します。

```
<?xml-stylesheet type="text/xsl" href="http://sun-248/stylesheets/bookOrder.xsl"?>
```

サーブレットを次のようにして初期化します。

```
public class AQTestServlet extends oracle.AQ.xml.AQxmlServlet
{
  /* The init method must be overloaded to specify the AQxmlDataSource */
  public void init()
  {
      AQxmlDataSource  db_drv = null;

      try
      {
        /* Create data source with username, password, sid, host, port */
        db_drv = new AQxmlDataSource("AQADM", "AQADM",
                                     "test_db", "sun-248", "5521");
```

```
            this.setAQDataSource(db_drv);

            /* Set the bookOrder.xsl style sheet for all responses */
            setStyleSheet("text/xsl", "http://sun-248:8000/stylesheets/bookOrder.xsl");
          }
          catch (Exception ex)
          {
              System.out.println("Exception in init: " + ex);
          }
      }
```

AQ 操作前後のコールバック

AQ サーブレットを使用すると、AQ 操作の実行前後に起動するコールバックを登録できます。これによって、ユーザーは、AQ 操作と AQ 操作以外の操作を同じトランザクションで実行できます。

コールバックを受信するには、ユーザーは、oracle.AQ.xml.AQxmlCallback インタフェースを実装するオブジェクトを登録します。AQxmlCallback インタフェースのメソッドは、次のとおりです。

```
public interface AQxmlCallback
{

  /** Callback invoked before any AQ operations are performed by the servlet */
  public void beforeAQOperation(HttpServletRequest request,HttpServletResponse
response,
                                AQxmlCallbackContext ctx);

  /** Callback invoked after any AQ operations are performed by the servlet */
  public void afterAQOperation(HttpServletRequest request, HttpServletResponse
response,
                                AQxmlCallbackContext ctx);
}
```

HTTP のリクエストとレスポンスのストリームおよび AQxmlCallbackContext オブジェクトにコールバックが渡されます。オブジェクトのメソッドは、次のとおりです。

- java.sql.Connection getDBConnection() メソッドは、サーブレットが AQ 操作を実行するために使用するデータベース接続に対するハンドルを提供します。ユーザーは、このコネクション・オブジェクトを使用して、コールバック・ファンクションで他の SQL 操作を実行できます。
- このコネクション・オブジェクトでは、close()、commit() または rollback() メソッドをコールできないことに注意してください。
- org.w3c.org.Document parseRequestStream() は、解析されたリクエストのストリームを表す DOM ドキュメントを提供します。
- void setStyleSheet(String type,String href) メソッドを使用すると、ユーザーは特定のコールに対してスタイル・シートを設定できます。したがって、ユーザーは、このサーブレットからのすべてのレスポンスに単一のスタイル・シートを指定するのではなく、特定のレスポンスに対してスタイル・シートを指定できます。

  コールバックで指定したスタイル・シートは、init() メソッドでサーバーに対して指定したスタイル・シート（存在する場合）をオーバーライドします。

例：

サーブレットで AQ 操作を実行する前に、EMP 表に行を挿入する必要があるとします。これを行うには、次のようにしてコールバック・クラスを作成し、特定のサーブレットに対応付けます。

```
import javax.servlet.*;
import javax.servlet.http.*;
import oracle.AQ.xml.*;
import java.sql.*;
import javax.jms.*;

/**
 * This is a sample AQ Servlet callback
 */
public class TestCallback implements oracle.AQ.xml.AQxmlCallback
{

   /** Callback invoked before any AQ operations are performed by the servlet */
  public void beforeAQOperation(HttpServletRequest request,HttpServletResponse
response,
       AQxmlCallbackContext ctx)
  {
      Connection conn = null;
      System.out.println("Entering BeforeAQ Callback ...");

      try
      {
            // Get the connection object from the callback context
            conn = ctx.getDBConnection();

            // Insert value in the EMP table
            PreparedStatement pstmt =
            conn.prepareStatement ("insert into EMP (EMPNO, ENAME) values (100,
'HARRY')");
            pstmt.execute ();
            pstmt.close();
      }
      catch (Exception ex)
      {
          System.out.println("Exception ex: " + ex);
      }
  }

  /** Callback invoked after any AQ operations are performed by the servlet */
  public void afterAQOperation(HttpServletRequest request,HttpServletResponse
response,
      AQxmlCallbackContext ctx)
```

```
  {
      System.out.println("Entering afterAQ Callback ...");

      try
      {
         // Set style sheet for response
         ctx.setStyleSheetProcessingInstr(
                 "type='text/xsl href='http://sun-248/AQ/xslt23.html'");

      }
      catch (Exception aq_ex)
      {
          System.out.println("Exception: " + ex);

      }
  }
}

/* Sample AQ servlet - using user-defined callbacks */
public class AQTestServlet extends oracle.AQ.xml.AQxmlServlet
{

  /* The init method must be overloaded to specify the AQxmlDataSource */
  public void init()
  {
      AQxmlDataSource  db_drv = null;
      AQxmlCallback    serv_cbk  = new TestCallback();

      try
      {
        /* Create data source with username, password, sid, host, port */
        db_drv = new AQxmlDataSource("AQADM", "AQADM", "test_db", "sun-248",
"5521");

        this.setAQDataSource(db_drv);

        /* Set Callback */
        setUserCallback(serv_cbk);

      }
      catch (Exception ex)
      {
         System.out.println("Exception in init: " + ex);
      }
}
```

18

メッセージ・ゲートウェイ

Oracle9*i* のアドバンスド・キューイング機能であるメッセージ・ゲートウェイを使用すると、Oracle 以外のメッセージ・システムに基づくアプリケーションと Oracle のアドバンスト・キューイング（AQ）機能に基づくアプリケーション間の通信が可能になります。アドバンスト・キューイングでは、2 つの AQ キュー間で伝播が行われるため、E-Business が可能になります（iDAP を介した HTTP）。メッセージ・ゲートウェイでは、Oracle 以外のメッセージ・システムに基づくレガシー・アプリケーションに対してもこの伝播が行われます。

メッセージ・ゲートウェイはアドバンスト・キューイングおよび Oracle9*i* に統合されているため、完全にトランザクション処理された、安全性の高いメッセージ配信が可能です。メッセージ・ゲートウェイを使用すると、AQ と永続性がサポートされている Oracle 以外のメッセージ・システム間で、メッセージの配信が 1 回のみ行われることが保証されます。AQ に類似した PL/SQL インタフェースでは、AQ の使用方法にすでに精通している開発者を対象に、簡単に使用できる管理 API が提供されます。

今回のリリースのメッセージ・ゲートウェイでは、Oracle9*i* アドバンスト・キューイングと IBM MQSeries 5.1 ベースおよび MQSeries 5.2 ベースのアプリケーションの統合がサポートされています。

内容は次のとおりです。

- メッセージ・ゲートウェイの機能
- メッセージ・ゲートウェイのアーキテクチャ
- 伝播処理の概要
- メッセージ・ゲートウェイの設定
- メッセージ・ゲートウェイの操作
- メッセージの変換
- mgw.ora 初期化ファイル

メッセージ・ゲートウェイの機能

メッセージ・ゲートウェイの機能は次のとおりです。

- AQ メッセージ伝播の拡張

  メッセージ・ゲートウェイでは、アドバンスト・キューイングと Oracle 以外のメッセージ・システム間でメッセージが伝播されます。アドバンスト・キューイング・アプリケーションによって送信されたメッセージは、Oracle 以外のメッセージ・システム・アプリケーションで受信できます。逆に、Oracle 以外のメッセージ・システム・アプリケーションによってパブリッシュされたメッセージは、アドバンスト・キューイング・アプリケーションで使用できます。

- システム固有のメッセージ・フォーマットのサポート

  メッセージ・ゲートウェイでは、メッセージ・システム固有のメッセージ・フォーマットがサポートされています。AQ メッセージには、`RAW` ペイロードまたはすべてのユーザー定義型ペイロードを含めることができます。MQSeries のメッセージは、`TEXT` またはすべての型のバイト・メッセージにすることができます。これによって、メッセージ・システムの既存のアプリケーションを統合できます。

- メッセージの変換

  メッセージ・ゲートウェイを使用すると、AQ メッセージと Oracle 以外のメッセージ・システムのメッセージ間でメッセージを簡単に変換できます。メッセージは、メッセージ・ゲートウェイによって提供される自動メッセージ変換ルーチンまたはユーザーによってカスタマイズされたメッセージ変換ファンクションのいずれかを介して変換されます。

- Oracle データベースとの統合

  メッセージ・ゲートウェイは、AQ に類似した PL/SQL インタフェースを介して管理されます。構成情報は、Oracle データベース表に格納されます。メッセージの伝播は、Oracle データベース・サーバーの外部プロセスによって実行されます。

- メッセージ配信の保証

  メッセージ・ゲートウェイでは、伝播元のメッセージ・システムと伝播先のメッセージ・システムの両方でトランザクションがサポートされている場合、永続メッセージが 1 回のみ伝播されることが保証されます。

  メッセージが永続メッセージではないか、または伝播元および伝播先のメッセージ・システムでトランザクションがサポートされていない場合は、伝播が行われることが保証されるのみです。

- セキュリティ・サポート

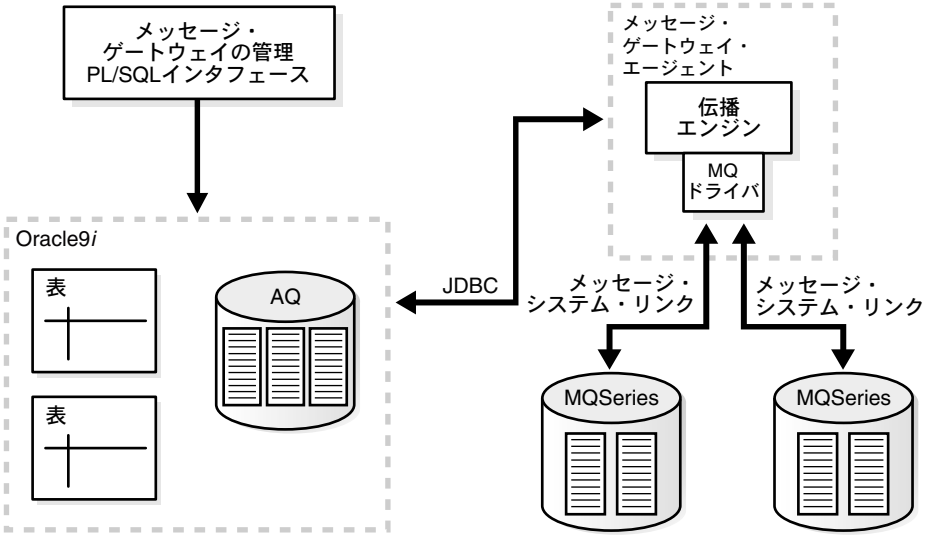
  メッセージ・ゲートウェイでは、Oracle データベースおよび Oracle 以外のメッセージ・システムのクライアント認証がサポートされています。

ゲートウェイによって作成された表、ビューおよびプロシージャへのアクセスを制御するOracle データベース管理者に対して、メッセージ・ゲートウェイでは、ゲートウェイの管理および伝播の処理のために MGW_ADMINISTRATOR_ROLE および MGW_AGENT_ROLE の 2 つのロールが定義されます。18-7 ページの「データベースへのデータベース・オブジェクトのロード」、18-11 ページの「メッセージ・ゲートウェイの管理ユーザーの作成」および 18-11 ページの「メッセージ・ゲートウェイ・エージェントのユーザーの作成」を参照してください。

メッセージ・ゲートウェイのアーキテクチャ

メッセージ・ゲートウェイの主要な構成要素は、ゲートウェイの構成管理用の DBMS_MGWADM という名前の管理パッケージと、伝播を処理するゲートウェイ・エージェントです（図 18-1 を参照）。ゲートウェイ・エージェントは、伝播エンジンおよび Oracle 以外のメッセージ・システムと通信する一連のドライバで構成されています。

図 18-1 メッセージ・ゲートウェイのアーキテクチャ



管理パッケージ

メッセージ・ゲートウェイの管理パッケージである DBMS_MGWADM では、ゲートウェイ管理者がゲートウェイ・エージェントの管理、伝播の設定および伝播の処理の監視を行うためのインタフェースが提供されます。

管理パッケージを介して、データベースへの接続を確立するための適切なユーザー名、パスワードおよび Oracle データベースのデータベース接続文字列をゲートウェイ・エージェントに構成します。パッケージ内のプロシージャをコールして、データベース接続の最大数およびエージェントに対するメモリー・ヒープ・サイズを割り当てることもできます。

Oracle 以外のメッセージ・システムとの間でメッセージの伝播を行うゲートウェイ・エージェントの場合、管理パッケージを使用して、エージェントと Oracle 以外のメッセージ・システムとの間の通信チャネルを表すメッセージ・システム・リンクを作成する必要があります。このエージェントには、複数のメッセージ・システム・リンクを構成できます。

管理パッケージを使用して、伝播に関連するすべての Oracle 以外のキューを登録する必要があります。ゲートウェイ構成に Oracle 以外のキューを登録しても、Oracle 以外のメッセージ・システムに物理的なキューは作成されず、キューにアクセスするためのメッセージ・システム・リンク、システム固有の名前、ドメイン（キューまたはトピック）など、キューに関する情報のみが記録されます。物理的なキューは、Oracle 以外のメッセージ・システムの管理インタフェースを介して作成する必要があります。

メッセージ・システム・リンクおよび Oracle 以外のキューの構成後、伝播ジョブを作成してメッセージの伝播を設定できます。メッセージ・ゲートウェイの伝播ジョブは、伝播サブスクライバおよび伝播スケジュールで構成されます。伝播サブスクライバを作成することによって、伝播ジョブのソース・キューおよび宛先キューが定義されます。伝播ジョブに関連付けられた伝播スケジュールを操作して、伝播ジョブの処理時期を制御します。

メッセージ・ゲートウェイでは、ゲートウェイ管理者が現在の構成情報、ゲートウェイ・エージェントの実行状態、および伝播ジョブの状態と統計についての問合せおよび確認を行うためのデータベース・ビューが提供されます。

ゲートウェイの構成は、ゲートウェイ・エージェントが実行中であるか停止中であるかに関係なく変更できます。エージェントが実行中である場合は、管理プロシージャからエージェントに構成の変更についての通知が送信されます。エージェントでは、ほぼすべての構成変更に対してエージェントの構成が動的に変更されます。ただし、変更を有効にするためにエージェントを停止し、再起動する必要がある場合もあります。管理パッケージ内のすべてのプロシージャはシリアル化され、これによって、ゲートウェイ・エージェントが構成変更の通知を変更順に受信することが保証されます。

> **参照：** DBMS_MGWADM の詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

ゲートウェイ・エージェント

ゲートウェイ・エージェントでは、伝播ジョブのスケジュールおよび処理が行われます。エージェントは、Oracle データベース・サーバーの外部プロセスで実行されます。エージェントを起動および終了するには、管理パッケージ内の STARTUP および SHUTDOWN プロシージャをコールします。このエージェントは、Oracle データベース・サーバーの他の外部プロセスと同様に、エージェントが存在するデータベース・サーバーが起動され、実行中である場合にのみ実行されます。

エージェントには、伝播エンジンおよび Oracle 以外のメッセージ・システム用のドライバ・セットが含まれます。マルチスレッド伝播エンジンによって、伝播ジョブが適切にスケジューリングされ、伝播処理がジョブ間およびジョブ内でパラレルに実行されます。エージェントのポーリング・スレッドによって、有効な伝播ジョブのソース・キューが定期的にポーリングされ、メッセージが使用可能な場合は、ワーカー・スレッドが起動して伝播ジョブが処理されます。メッセージ・リンクが作成されると、ゲートウェイ・エージェントのドライバがインスタンス化されます。ドライバは、すべてのメッセージ操作に対し、メッセージ・システムのクライアントとして実行されます。

エージェントによって、ログ・ファイルにログ・メッセージが書き込まれます。このメッセージには、エージェントの構成、エージェントの状態、動的な通知の受信時にエージェントが実行したアクション、伝播ジョブの状態およびすべてのエラー・メッセージが含まれます。

伝播処理の概要

伝播ジョブを作成して、メッセージの伝播を設定します。概念上、伝播ジョブは、伝播サブスクライバおよび伝播スケジュールで構成されます。

伝播サブスクライバの作成後、ソースがトピックである場合（パブリッシュ・サブスクライブ）は、ゲートウェイによって伝播ソース上にサブスクリプションが作成されます。サブスクライバの作成後、ゲートウェイによってパブリッシュされたすべてのメッセージがトピックに移されます。伝播ソースが Point-to-Point キューの場合は、ゲートウェイによってキュー内のすべてのメッセージが宛先に移されます。

関連付けられたスケジュールが作成されないかぎり、伝播ジョブは処理されません。ゲートウェイ・エージェントでは、有効な伝播ジョブが処理されます。伝播ジョブを無効にすると、ソース・キューから宛先キューへのメッセージの転送が停止します。ただし、サブスクリプションは停止しません。

伝播ジョブが処理されると、メッセージは優先順位に従ってソース・キューからデキューされ、宛先キューにエンキューされます。ソース・フォーマットから宛先フォーマットへのメッセージの変換に失敗した場合、メッセージは例外キューに移されます。伝播ソース・キューで期限切れになったメッセージは、宛先キューに伝播されません。

伝播ジョブのソース・メッセージ・システムでメッセージ・セレクタがサポートされている場合、メッセージ・ゲートウェイを使用して、伝播ジョブに伝播メッセージ・セレクタを指定できます。メッセージ・セレクタの条件を満たすメッセージのみが伝播されます。

伝播ジョブの処理中に障害が発生した場合、エージェントは、ジョブを無効にする前に指数バックオフ・スキームで最大 16 回ジョブを再試行します。

メッセージは、メッセージの伝播時に、ソース・メッセージ・システム固有のフォーマットから宛先メッセージ・システム固有のフォーマットに変換されます。ゲートウェイでは、一般的に使用されている単純なメッセージ・フォーマット間のメッセージ変換は自動で行われます。カスタマイズされたメッセージの変換用に、独自のメッセージ変換ファンクションを使用できます。

> **参照：** DBMS_TRANSFORM の詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』を参照してください。

メッセージ・ゲートウェイの設定

この項では、メッセージ・ゲートウェイをロードおよび設定する手順について説明します。

Oracle9*i* データベースの前提条件

init<sid>.ora ファイル（<sid> は、メッセージ・ゲートウェイに使用されるデータベース・インスタンスの Oracle システム ID）に、次のパラメータを指定する必要があります。

- 1 つ以上のジョブ・キュー・プロセス

  ```
  JOB_QUEUE_PROCESSES = <num_of_processes>
  ```

- 1 つ以上の AQ 時間監視プロセス

  ```
  AQ_TM_PROCESSES = <num_of_processes>
  ```

Oracle 以外のメッセージ・システムの前提条件

メッセージ・ゲートウェイをロードおよび設定する前に、Oracle 以外のメッセージ・システムをインストールします。メッセージ・ゲートウェイでは、Oracle 以外のシステムの共有ライブラリおよび Java クラス・ファイルが使用されます。

タスクのロードおよび設定

メッセージ・ゲートウェイを実行する前に、次の手順を実行する必要があります。これらのタスクは、「Windows NT のみ」または「UNIX のみ」と示されている場合を除き、UNIX と Windows NT の両方に適用されます。

1. データベースへのデータベース・オブジェクトのロード
2. 外部プロシージャに対する listener.ora の変更
3. 外部プロシージャに対する tnsnames.ora の変更
4. mgw.ora 初期化ファイルの変更
5. メッセージ・ゲートウェイの管理ユーザーの作成
6. メッセージ・ゲートウェイ・エージェントのユーザーの作成
7. メッセージ・ゲートウェイの接続情報の構成

データベースへのデータベース・オブジェクトのロード

SQL*Plus を使用して、`$ORACLE_HOME/mgw/admin` ディレクトリに存在する `catmgw.sql` を実行します。ユーザー `SYS` または `SYSDBA` として実行します。

SQL スクリプト `catmgw.sql` を実行すると、ロール、表、ビュー、オブジェクト型および PL/SQL パッケージを含む、メッセージ・ゲートウェイに必要なデータベース・オブジェクトがロードされます。メッセージ・ゲートウェイの PL/SQL パッケージおよび型に対するパブリック・シノニムが作成されます。`MGW_ADMINISTRATOR_ROLE` および `MGW_AGENT_ROLE` の 2 つのロールも作成され、特定の権限が付与されます。また、エージェントの外部プロシージャにライブラリの別名も作成されます。すべてのオブジェクトは `SYS` が所有します。

外部プロシージャに対する listener.ora の変更

Windows NT のみ : この手順は省略できます。Windows NT では、リスナーについての静的サービス情報は必要ありません。

メッセージ・ゲートウェイの PL/SQL パッケージで外部プロシージャがコールされるように、`listener.ora` を変更する必要があります。

1. `listener.ora` に、外部プロシージャのデフォルトの IPC プロトコル・アドレスが設定されていることを確認します。

外部プロシージャのプロトコル・アドレスの例

```
LISTENER = (ADDRESS_LIST=
(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC))
      .
      .
      .
```

2. listener.ora で、手順 1 のリスナーに静的サービス情報を追加します。これによって、リスナーの SID_DESC が設定されます。SID_DESC に含まれるパラメータのうち、次のパラメータはメッセージ・ゲートウェイにとって重要です。これらのパラメータは、ユーザーの状況に応じて指定する必要があります。

   a. SID_NAME: tnsnames.ora でネット・サービス名に指定された SID（「mgwextproc」など）を指定します。

   b. ORACLE_HOME: ORACLE_HOME ディレクトリを指定します。

   c. PROGRAM: 外部プロシージャのエージェントの名前（「extproc」）を指定します。

   d. ENVS: 外部プロシージャを実行するために必要な環境変数 LD_LIBRARY_PATH を設定します。

      LD_LIBRARY_PATH には次のパスを含める必要があります。

      ```
      [ORACLE_HOME]/jdk/jre/lib/[PLATFORM_TYPE]

      [ORACLE_HOME]/lib
      ```

      大カッコで囲まれた項目には、完全なスペルの適切な値を代入します（たとえば、$ORACLE_HOME を使用しても機能しません）。PLATFORM_TYPE は、プラットフォームの種類（sparc など）です。

リスナーの静的サービス情報の追加の例

```
# Add a SID_DESC
SID_LIST_LISTENER= (SID_LIST=
(SID_DESC =
   (SID_NAME= mgwextproc)
   (ENVS="LD_LIBRARY_PATH=/private/oracle/orcl9i/jdk/jre/lib/
      sparc:/private/oracle/orcl9i/lib")
   (ORACLE_HOME=/private/oracle/orcl9i)
   (PROGRAM = extproc))
      .
      .
      .
```

外部プロシージャに対する tnsnames.ora の変更

Windows NT のみ : この手順は省略できます。

外部プロシージャに対して、tnsnames.ora で、接続記述子が listener.ora で構成された情報に一致するネット・サービス名 MGW_AGENT を構成します。ネット・サービス名は、MGW_AGENT である必要があります（この値は固定）。KEY の値は、listener.ora で IPC プロトコルに対して指定された KEY の値に一致する必要があります。SID の値は、listener.ora で SID_DESC エントリの SID_NAME に対して指定された値に一致する必要があります。

tnsnames.ora の変更の例

```
MGW_AGENT =
(DESCRIPTION=
   (ADDRESS_LIST= (ADDRESS= (PROTOCOL=IPC)(KEY=EXTPROC)))
   (CONNECT_DATA= (SID=mgwextproc) (PRESENTATION=RO)))
```

mgw.ora 初期化ファイルの変更

メッセージ・ゲートウェイの初期化ファイル $ORACLE_HOME/mgw/admin/mgw.ora は、初期化パラメータを取得してエージェントを起動するために、ゲートウェイの外部プロシージャによって使用される TEXT ファイルです。
$ORACLE_HOME/mgw/admin/sample_mgw.ora を mgw.ora にコピーし、状況に応じて変更します。

次の手順を実行して、環境変数および他のパラメータを設定します。

1. 外部プロシージャに対して、ゲートウェイ・エージェントを起動するための環境変数を設定します。

   a. 次の環境変数を設定します。

      UNIX のみ : LD_LIBRARY_PATH を設定します。大カッコで囲まれた項目には、該当する具体的な値を代入します（たとえば、$ORACLE_HOME を使用しても機能しません）。PLATFORM_TYPE には、プラットフォームの種類（sparc など）を代入します。

      LD_LIBRARY_PATH には次のパスを含める必要があります。

      * [ORACLE_HOME]/jdk/jre/lib/[PLATFORM_TYPE]
      * [ORACLE_HOME]/rdbms/lib
      * [ORACLE_HOME]/oracle/lib
      * [ORACLE_HOME]/mgw/lib
      * メッセージ・ゲートウェイ・エージェントが Oracle 以外のメッセージ・システム（MQSeries ライブラリなど）にアクセスするために必要な追加ライブラリは、すべて LD_LIBRARY_PATH に含める必要があります。

      Windows NT のみ : 環境変数 MGW_PRE_PATH を設定します。この値は、jvm.dll ライブラリへのパスです。JDK リソースの場合、%ORACLE_HOME% に存在する JDK パッケージを使用します。たとえば、%ORACLE_HOME% が D:¥oracle である場合は、次のような行を追加します。

      ```
      set MGW_PRE_PATH = D:¥oracle¥jdk¥jre¥bin¥classic
      ```

      この変数は、メッセージ・ゲートウェイ・エージェントのプロセスによって継承されたパスの前に付けられます。

b. CLASSPATH を設定します（Windows NT ユーザーは、Windows NT のパス構文を使用して CLASSPATH を設定する必要があります）。

CLASSPATH には次のクラスを含める必要があります。大カッコで囲まれた項目には、該当する具体的な値を代入します（たとえば、$ORACLE_HOME を使用しても機能しません）。

* メッセージ・ゲートウェイ・クラス：

  [ORACLE_HOME]/mgw/classes/mgw.jar

* JDK 国際化クラス：[ORACLE_HOME]/jdk/jre/lib/i18n.jar
* JDK ランタイム・クラス：[ORACLE_HOME]/jdk/jre/lib/rt.jar
* Oracle JDBC クラス：[ORACLE_HOME]/jdbc/lib/classes12.zip
* Oracle 国際化クラス：[ORACLE_HOME]/jdbc/lib/nls_charset12.zip
* [ORACLE_HOME]/sqlj/lib/translator.zip
* [ORACLE_HOME]/sqlj/lib/runtime12.zip
* メッセージ・ゲートウェイが Oracle 以外のメッセージ・システム（MQSeries など）にアクセスするために必要な追加のクラス

2. log_directory および log_level パラメータを設定します。

これらのパラメータの設定は、必須ではありません。これらのパラメータは、メッセージ・ゲートウェイのロギングに影響します。これらのパラメータを設定しない場合は、デフォルト値が使用されます。log_directory のデフォルト値は、$ORACLE_HOME/mgw/log です。log_level のデフォルト値は、基本的なロギングを示す 0（ゼロ）です。

3. oracle_sid パラメータを設定します。

メッセージ・ゲートウェイの接続情報の構成時にデータベース接続文字列が提供されないように、mgw.ora に oracle_sid パラメータを設定します。18-11 ページの「メッセージ・ゲートウェイの接続情報の構成」を参照してください。

mgw.ora ファイルの例

```
#an example of mgw.ora file
log_directory=/private/mgwlog
log_level=2
set CLASSPATH=<proper value>
set LD_LIBRARY_PATH=<proper value>
```

メッセージ・ゲートウェイの管理ユーザーの作成

ゲートウェイの管理作業を実行するには、MGW_ADMINISTRATOR_ROLE 権限を所有するデータベース・ユーザーを作成する必要があります。

管理ユーザーの作成例

```
CREATE USER <admin_user> IDENTIFED BY <admin_password>;
GRANT CONNECT, RESOURCE to <admin_user>;
GRANT MGW_ADMINISTRATOR_ROLE to <admin_user>;
```

メッセージ・ゲートウェイ・エージェントのユーザーの作成

ゲートウェイ・エージェントからデータベースへの接続を確立するには、MGW_AGENT_ROLE 権限を所有するデータベース・ユーザーを作成する必要があります。

エージェント・ユーザーの作成例

```
CREATE USER <agent_user> IDENTIFED BY <agent_password>;
GRANT CONNECT, RESOURCE to <agent_user>;
GRANT MGW_AGENT_ROLE to <agent_user>;
```

メッセージ・ゲートウェイの接続情報の構成

エージェント・ユーザーの作成後、管理ユーザーとして DBMS_MGWADM.DB_CONNECT_INFO を使用し、メッセージ・ゲートウェイに、ゲートウェイ・エージェントによるデータベースへの接続に使用されるユーザー名、パスワードおよびデータベース接続文字列を構成します。18-11 ページの「エージェント・ユーザーの作成例」で作成したエージェントのユーザー名およびパスワードを使用します。データベース接続文字列パラメータは、tnsnames.ora の新しいサービス名（パフォーマンスを向上させるには IPC プロトコルを使用）または NULL のいずれかに設定できます。NULL に設定する場合、mgw.ora に oracle_sid パラメータを設定する必要があります。

Oracle9*i* リリース 2（9.2）では、DBMS_MGW_DB_CONNECT_INFO( ) をコールする場合、データベース接続文字列パラメータに対して常に NULL 以外の値を指定します。

DBMS_MGWADM.DB_CONNECT_INFO の使用例

```
connect <admin_user>/<admin_password>
exec dbms_mgwadm.db_connect_info('<agent_user>','<agent_password>', '<agent_
database>');
```

セットアップの確認

次の手順を実行して、インストールを確認します。この手順には、メッセージ・ゲートウェイ・エージェントの単純な起動および停止が含まれます。

1. データベース・リスナーを起動します。

   外部プロシージャに対するリスナーおよび通常のデータベース接続に対する他のリスナーを起動します。

2. ゲートウェイ・エージェント・ユーザーに対してデータベース接続文字列をテストします。

   sqlplus <agent_user>/<agent_password>@<agent_database> を実行します。

   正常に終了した場合、ゲートウェイ・エージェントをデータベースに接続できます。

3. ゲートウェイ・エージェントを起動します。

   a. <admin_user> として接続し、DBMS_MGWADM.STARTUP をコールしてゲートウェイ・エージェントを起動します。

   b. MGW_GATEWAY ビューを使用して、AGENT_STATUS が RUNNING に、AGENT_PING が REACHABLE に変わるまで待ちます。

4. ゲートウェイ・エージェントを停止します。

   a. <admin_user> として接続し、DBMS_MGWADM.SHUTDOWN をコールします。

   b. MGW_GATEWAY ビューを使用して、AGENT_STATUS が NOT_STARTED に変わるまで待ちます。

メッセージ・ゲートウェイのアンロード

メッセージ・ゲートウェイをアンロードするには、次の手順を実行します。

1. メッセージ・ゲートウェイを停止します。

2. ペイロードがメッセージ・ゲートウェイの標準型（MGW_BASIC_MSG_T など）であるユーザー作成のキューを削除します。

3. SQL*Plus を使用して、ユーザー SYS または SYSDBA として、$ORACLE_HOME/mgw/admin ディレクトリに存在する catnomgw.sql を実行します。

   これによって、ロール、表、ビュー、パッケージ、オブジェクト型およびシノニムを含む、メッセージ・ゲートウェイによって使用されるデータベース・オブジェクトが削除されます。

4. listener.ora および tnsnames.ora に作成されたメッセージ・ゲートウェイのエントリを削除します。

メッセージ・ゲートウェイの操作

メッセージ・ゲートウェイは、ロードおよび設定の終了後、構成および実行が可能になります。この項では、メッセージ・ゲートウェイを構成、起動および停止する方法を説明します。メッセージ・ゲートウェイ・エージェントを監視する方法も説明します。構成の例では、ペイロード型が RAW である AQ キューから MQSeries キューへのメッセージの伝播を示します。この例のすべてのコマンドは、変換を作成するコマンドを除き、MGW_ADMINISTRATOR_ROLE ロールが付与されたユーザーとして実行する必要があります。

メッセージ・ゲートウェイ・エージェントの管理

メッセージ・ゲートウェイ・エージェントは、データベースの外部プロセスとして実行されます。アドバンスド・キューイングおよびメッセージ・ゲートウェイの管理パッケージにアクセスするには、メッセージ・ゲートウェイ・エージェントからデータベースへの接続を確立する必要があります。

起動前に、データベースへの接続およびリソース制限の設定に使用される情報を含む、構成情報を登録する必要があります。

構成

DBMS_MGWADM.DB_CONNECT_INFO プロシージャを使用して、メッセージ・ゲートウェイ・エージェントによってデータベース接続に使用されるユーザーのユーザー名およびパスワード、および接続を確立するために使用されるデータベース接続文字列を、メッセージ・ゲートウェイに構成します。メッセージ・ゲートウェイ・エージェントを起動するには、ユーザーに MGW_AGENT_ROLE が付与されている必要があります。データベース接続文字列が指定されていない場合、メッセージ・ゲートウェイ・エージェントではローカル接続が使用されます。

メッセージ・ゲートウェイ・エージェントの実行中は、DBMS_MGWADM.DB_CONNECT_INFO をコールして新しい接続情報を設定することもできます。

新しい接続情報の設定例

```
SQL> exec dbms_mgwadm.db_connect_info('mgwagent', 'mgwagent_password', 'mydatabase')
```

DBMS_MGWADM.ALTER_AGENT を使用して、メッセージ・ゲートウェイ・エージェントがデータベースに接続するために使用可能なコネクション・プール内の接続の最大数、およびメッセージ・ゲートウェイ・エージェントの処理のヒープ・サイズ（MB）を設定できます。コネクション・プール内の接続の数は、パフォーマンスに影響する可能性があります。接続の数のデフォルト値は 1、メモリーのデフォルト値は 64MB です。

次の手順を実行すると、データベース接続の数が 2、ヒープ・サイズが 64MB に設定されます。

```
SQL> exec dbms_mgwadm.alter_agent(2, 64)
```

メッセージ・ゲートウェイ・エージェントの実行中は、接続の最大数を変更できます。ただし、値は増やすことしかできません。メッセージ・ゲートウェイが実行中である場合、最大メモリーは変更できません。NULL 値を入力しても、最大メモリー属性は変更されません。

次の例では、メッセージ・ゲートウェイ・エージェントの実行中、接続の最大数が 3 に更新されます。最大メモリーは変更されません。

```
SQL> exec dbms_mgwadm.alter_agent(3, NULL)
```

起動および停止

メッセージ・ゲートウェイのインストールおよび構成後、次のコマンドを実行してメッセージ・ゲートウェイを起動します。

```
SQL> exec dbms_mgwadm.startup
```

MGW_GATEWAY ビューを使用し、ログ・ファイルを監視することによって、メッセージ・ゲートウェイ・エージェントの状態を判断できます。18-26 ページの「メッセージ・ゲートウェイのログ・ファイルの監視」を参照してください。

次のとおり MGW_GATEWAY を使用して、メッセージ・ゲートウェイ・エージェントを監視します。

```
SQL> select * from mgw_gateway;

AGENT_STATUS  AGENT_PING  AGENT_JOB  AGENT_USER  AGENT_DATABASE  LAST_ERRO
------------------------------------------------------------------------
RUNNING       REACHABLE   213        MGWAGENT


(Continued) LAST_ERR  LAST_ERROR_MSG  MAX_CONNECTIONS  MAX_MEMORY
            ------------------------------------------------------
                                    3               64
```

メッセージ・ゲートウェイのインスタンス化が完了すると、AGENT_STATUS 列に RUNNING、AGENT_PING 列に REACHABLE という値が表示されます。

最初の列の AGENT_STATUS には、ゲートウェイ・エージェントの状態が表示されます。この列には、NOT_STARTED、START_SCHEDULED、INITIALIZING、STARTING、RUNNING および SHUTTING_DOWN のいずれかが表示されます。2 列目の AGENT_PING には、メッセージ・ゲートウェイ・エージェントの ping の応答が表示されます。この値は、REACHABLE または UNREACHABLE のいずれかになります。LAST_ERROR_MSG、LAST_ERROR_DATE および LAST_ERROR_TIME の列には、メッセージ・ゲートウェイ・エージェントの起動時または実行時にエラーが発生した場合に有効な情報が表示されます。

> **参照：** データベース・ビューの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の「DBMS_MGWADM」を参照してください。

次のコマンドを実行すると、メッセージ・ゲートウェイ・エージェントが停止します。

```
SQL> exec dbms_mgwadm.shutdown
```

メッセージ・ゲートウェイによる停止プロシージャが完了すると、AGENT_STATUS 列に NOT_STARTED と表示されます。

MGW_GATEWAY ビューおよびログ・ファイルを監視することによって、停止プロシージャが正常に終了したかどうかを判断できます。停止中に問題が発生したか、または予測しないイベントが発生してメッセージ・ゲートウェイの管理の一貫性が失われた場合、次のコマンドを実行してステータス情報をリセットできます。

```
SQL> exec dbms_mgwadm.cleanup_gateway(dbms_gmwadm.CLEAN_STARTUP_STATE)
```

このコマンドの実行時は、メッセージ・ゲートウェイ・エージェントの処理を実行しなでください。

メッセージ・ゲートウェイ・リンクの構成

次のスクリプトの例に示すとおり、SQL スクリプトを使用してメッセージ・ゲートウェイを構成できます。完全な例については、メッセージ・ゲートウェイのインストール時にインストールされる samples ディレクトリのサンプルを参照してください。

メッセージ・ゲートウェイ・リンクの作成

メッセージ・ゲートウェイ・リンクは、Oracle 以外のメッセージ・システムに対する一連の接続情報です。メッセージ作業または管理作業のいずれかに接続が必要な場合は、常に、メッセージ・ゲートウェイ・リンクが使用されます。

MQSeries キュー・マネージャへのリンクに対して、MQSeries クライアント・コネクションのためのキュー・マネージャの名前、チャネル、ホスト、ポート、ユーザー名、パスワードなどの情報を設定できます。また、メッセージ・ゲートウェイ・エージェントによって使用される着信伝播または発信伝播のためのログ・キューを設定し、配信が 1 回のみ行われることを保証する必要があります。2 つのキューが同じ物理キューを参照できますが、異なる物理キューを参照するとパフォーマンスが向上します。

options 引数である { 名前 , 値 } の組（いずれも文字列）の集合は、Oracle 以外のメッセージ・システムのインタフェースに固有の引数を表します。MQSeries によって認識されるプロパティ名には、次のものが含まれます。

- 対応する MQEnvironment.CCSID プロパティに対する 'MQ_ccsid'
- MQEnvironment.SEND_EXIT に対する 'MQ_SendExit'
- MQEnvironment.RECEIVE_EXIT に対する 'MQ_ReceiveExit'
- MQEnvironment.SECURITY_EXIT に対する 'MQ_SecurityExit'

次の例では、MQSeries キュー・マネージャへのメッセージ・ゲートウェイ・リンクが構成されます。このリンクは、'mqlink' という名前で、MQSeries チャネル 'mychannel' を使用して、ホスト 'myhost.mydomain' およびポート 1414 上に存在する MQSeries キュー・マネージャ 'my.queue.manager' を使用するように構成されます。この例では、options パラメータを使用して MQSeries SendExit クラスも登録します。クラス 'mySendExit' は、(mgw.ora ファイルに設定された）メッセージ・ゲートウェイ・エージェントの CLASSPATH である必要があります。メッセージ・ゲートウェイ・エージェントの CLASSPATH の設定の詳細は、18-9 ページの「mgw.ora 初期化ファイルの変更」を参照してください。

> **参照：** MQSeries システムのプロパティおよびサポートされているオプションの詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の「DBMS_MGWADM」を参照してください。

```
declare
  v_options sys.mgw_properties;
  v_prop    sys.mgw_mqseries_properties;
begin
  -- Set options.
  -- Specify an MQSeries send exit class 'mySendExit' to be associated with the
queue.
  v_options := sys.mgw_properties(sys.mgw_property('MQ_SendExit', 'mySendExit'') );

  -- set certain MQSeries properties used for MQSeries
  v_prop := sys.mgw_mqseries_properties.construct();

  v_prop.max_connections := 1;
  v_prop.username := 'mqm';             -- username given to queue manager
  v_prop.password := 'mqm';             -- password given to queue manager
  v_prop.hostname := 'myhost.mydomain'  -- hostname for queue manager host
  v_prop.port     := 1414;              -- port (1414 is MQSeries default)
  v_prop.channel  := 'mychannel';       -- MQSeries channel name
  v_prop.outbound_log_queue := 'mylogq';    -- name of MQSeries queue to be
                                            -- used for MGW logging on
                                            -- outbound jobs.
  v_prop.queue_manager := 'my.queue.manager';  -- queue manager name
```

```
  dbms_mgwadm.create_msgsystem_link(
      linkname => 'mqlink',      -- link name
      properties => v_prop,      -- MQSeries driver properties
      options => v_options );    -- options

end;
```

メッセージ・ゲートウェイでは、構成可能なリンクの数は制限されません。

メッセージ・ゲートウェイ・リンクの変更

一部のリンク情報は変更できます。MQSeries リンクの場合、作成後に変更可能なプロパティは、max_connections、username、password、inbound_log_queue および outbound_log_queue です。次の例では、18-15 ページの「メッセージ・ゲートウェイ・リンクの作成」で作成された 'mqlink' リンクが変更され、max_connections および password プロパティが変更されています。

プロパティの型が VARCHAR2 の場合は、DBMS_MGWADM.NO_CHANGE という値を使用するとプロパティが変更されません。他の型のプロパティの場合は、NULL 値を使用するとプロパティが変更されません。MQSeries リンクを変更する場合は、mgw_mqseries_properties.alter_construct ファンクションを使用します。これによって、適切な値が自動的に設定されます。次に、変更が必要な値を設定します。

```
declare
  v_options sys.mgw_properties;
  v_prop    sys.mgw_mqseries_properties;
begin

  -- Alter certain MQSeries properties used for MQSeries.
  v_prop := sys.mgw_mqseries_properties.alter_construct();

  v_prop.max_connections := 2;          -- max_connections increased
  v_prop.password := 'newpasswd';       -- change password given to queue manager

 dbms_mgwadm.alter_msgsystem_link(
      linkname => 'mqlink',    -- link name
      properties => v_prop,    -- MQSeries driver properties
                               -- options will not be changed
      comment => 'link to queue manager, my.queue.manager. on my.host ');
                               -- add comment
end;
```

リンク情報は、メッセージ・ゲートウェイ・エージェントが実行されているかどうかに関係なく変更できます。

参照： メッセージ・ゲートウェイ・エージェント実行中に行う変更に対する制限は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の「DBMS_MGWADM」を参照してください。

メッセージ・ゲートウェイ・リンクの削除

メッセージ・ゲートウェイ・リンクに関連付けられたすべての登録済キューがすでに削除されている場合のみ、Oracle 以外のメッセージ・システムへのメッセージ・ゲートウェイ・リンクを削除できます。

```
begin
    dbms_mgwadm.remove_msgsystem_link('mqlink');
end;
```

リンクは、メッセージ・ゲートウェイ・エージェントが実行中であるかどうかに関係なく削除できます。

メッセージ・ゲートウェイ・リンクの状態の監視

MGW_LINKS ビューを使用して、構成済のリンクを確認できます。このビューには、名前およびリンクの型（適用される Oracle 以外のメッセージ・システム）が表示されます。構成されたリンクの情報を確認するには、Oracle 以外のメッセージ・システムに固有のビューを使用します。MQSeries の場合は、MGW_MQSERIES_LINKS ビューに、対応するほぼすべての情報を含む列が含まれます。

リンク情報の確認例

```
SQL> select * from MGW_LINKS;

LINK_NAME   LINK_TYPE   LINK_COMMENT
-----------------------------------
MQLINK      MQSERIES


SQL> select link_name, queue_manager, channel, hostname from MGW_MQSERIES_LINKS;

LINK_NAME   QUEUE_MANAGER      CHANNEL     HOSTNAME
-------------------------------------------------------------
MQLINK      my.queue.manager   mychannel   myhost.mydomain
```

Oracle 以外のメッセージ・システムのキューの登録

伝播に関連するすべての Oracle 以外のメッセージ・システムのキューは、メッセージ・ゲートウェイの管理インタフェースを介して登録する必要があります。メッセージ・ゲートウェイでは、Oracle 以外のキューは作成されません。構成情報を使用してこれらのキューへのアクセスが実行されるのみです。

Oracle 以外のキューの登録

次の情報を使用して、Oracle 以外のキューを登録します。

- Oracle 以外のメッセージ・システムに接続するために使用されるメッセージ・ゲートウェイ・リンクの名前。
- Oracle 以外のキューのシステム固有の名前（Oracle 以外のメッセージ・システムでの名前）。
- キュー（Point-to-Point）またはトピック（パブリッシュ・サブスクライブ）のいずれか。
- Oracle 以外のメッセージ・システムに固有の一連のオプション。これらのオプションは、{ 名前 , 値 } の組（いずれも文字列）の集合です。

MQSeries の場合、オプションは 'MQ_openOptions' のみです。このプロパティは、MQSeries Base Java の MQQueueManager.accessQueue メソッドの openOptions 引数に対応しています。openOptions の値を指定しない場合、エンキュー時は MQC.MQOO_OUTPUT、デキュー時は MQC.MQOO_INPUT_SHARED のデフォルト値が設定されます。

```
-- Registering non-Oracle queue
--
declare
  v_options sys.mgw_properties;
begin
  -- No options set for this foreign queue. Below is a sample of how one would be
set.
  -- v_options := sys.mgw_properties(sys.mgw_property('MQ_openOptions', '2066'));

  -- Register the queue
  dbms_mgwadm.register_foreign_queue(
      name           => 'destq',                  -- MGW non-Oracle queue name
      linkname       => 'mqlink',                 -- name of link to use
      provider_queue => 'my_mq_queue',            -- name of MQSeries queue
      domain         => dbms_mgwadm.DOMAIN_QUEUE, -- single consumer queue
      options        => v_options);
end;
```

ドメイン・パラメータは、Point-to-Point キューに対しては DBMS_MGWADM.DOMAIN_QUEUE、パブリッシュ・サブスクライブ・キューに対しては DBMS_MGWADM.DOMAIN_TOPIC に設定されます。MQSeries では、Point-to-Point キューのみがサポートされています。

登録されたキューの変更

Oracle 以外のキューは、構成および登録後には変更できません。登録情報を削除して再作成する必要があります。

Oracle 以外のキューの登録解除

Oracle 以外のキューは、それを参照するサブスクライバまたはスケジュールが存在しない場合のみ、登録を解除できます。

キューの登録解除例

```
begin
   dbms_mwgadm.unregister_foreign_queue('destq', 'mqlink');
end;
```

登録された Oracle 以外のキューの状態の監視

MGW_FOREIGN_QUEUES ビューを使用して、登録された Oracle 以外のキューを確認します。

登録されたキューの確認例

```
SQL> select name, link_name, provider_queue from MGW_FOREIGN_QUEUES;

NAME    LINK_NAME    PROVIDER_QUEUE
-----------------------------------
DESTQ   MQLINK       my_mq_queue
```

AQ キュー

AQ キューを登録する必要はありません。AQ キューが参照された場合、メッセージ・ゲートウェイはこれらのキューに直接アクセスします。

伝播ジョブの構成

あるキューから別のキューにメッセージを伝播するには、伝播ジョブが必要です。伝播ジョブは、伝播サブスクライバおよび伝播スケジュール（以降、サブスクライバおよびスケジュール）で構成されます。サブスクライバでは、ソース・キューおよび宛先キューが指定されます。スケジュールでは、伝播ジョブが処理される時期が指定されます。関連付けられたスケジュールが存在しないサブスクライバは処理されません。サブスクライバに関連付けられるスケジュールに対して、同じ伝播元および伝播先を設定する必要があります。

メッセージ・ゲートウェイ・サブスクライバは、Oracle 以外のメッセージ・システムにメッセージ・ゲートウェイ・サブスクライバとの対応という概念が存在しないかぎり、そのシステムのサブスクライバに対応しない場合があります。AQ キューに対するメッセージ・ゲートウェイ・サブスクライバは、そのキューの AQ サブスクライバと同じではないことに注意してください。ただし、メッセージ・ゲートウェイ・サブスクライバを作成すると、対応する AQ サブスクライバが作成されます。

> **参照：** サブスクライバの追加の詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の「DBMS_MGWADM」を参照してください。

メッセージ・ゲートウェイ・サブスクライバの作成

メッセージ・ゲートウェイサブスクライバは、次の情報で構成されます。

- 伝播の形式（受信または発信）
- ソース・キュー
- 宛先キュー
- 選択ルール（オプション）
- 変換名（オプション）
- 例外キュー（オプション）

メッセージ・ゲートウェイ・サブスクライバの作成例

```
begin
  dbms_mgwadm.add_subscriber(
     subscriber_id    => 'sub_aq2mq',          -- MGW subscriber name
     propagation_type => dbms_mgwadm.outbound_propagation, -- outbound propaga
     queue_name       => 'mgwuser.srcq',       -- AQ queue name (source queue)
     destination      => 'destq@mqlink');      -- MGW foreign queue with link
                                               --(destination queue)
end;
```

この例では、AQ キューからデキューする場合にメッセージを選択するためのサブスクライバ・ルールは指定しません。変換を指定する例は、18-23 ページの「変換の使用」を参照してください。

メッセージ・ゲートウェイ・スケジュールの作成

伝播ジョブを処理するには、メッセージ・ゲートウェイ・スケジュールを構成する必要があります。スケジュールによって、メッセージが伝播される時期が決定されます。Oracle9*i* リリース 2（9.2）では、伝播ジョブを有効および無効にするためにのみスケジュールが使用されます。Oracle9*i* リリース 2（9.2）では、スケジューリング・パラメータを使用しません。

伝播スケジュールの作成例

```
begin
  dbms_mgwadm.schedule_propagation(
    schedule_id      =>  'sch_aq2mq',                    -- schedule name
    propagation_type => dbms_mgwadm.outbound_propagation, -- outbound propaga
    source           => 'mgwuser.srcq',                  -- AQ queue name
    destination      => 'destq@mqlink');          -- MGW foreign queue with link
end;
```

伝播ジョブの有効化および無効化

スケジュールの作成時、スケジュールは使用可能な状態です。関連付けられたサブスクライバが存在する場合、対応する伝播ジョブがアクティブになります。この場合、ソース・キューのメッセージに対してポーリングが行われます。伝播ジョブを無効（または有効）にするには、関連付けられたスケジュールを使用不可能（または使用可能）にする必要があります。

次の例では、スケジュール 'sch_aq2mq' を使用不可能および使用可能にします。

```
begin
    dbms_mgwadm.disable_propagation_schedule('sch_aq2mq');
end;

begin
    dbms_mgwadm.enable_propagation_schedule('sch_aq2mq');
end;
```

伝播ジョブのリセット

伝播中に問題が発生した場合、メッセージ・ゲートウェイ・エージェントは、伝播ジョブを停止する前に、失敗した操作を最大 16 回再試行します。エラー・カウントを 0（ゼロ）にリセットして伝播ジョブを再開するには、reset_subscriber( ) プロシージャを使用します。

伝播ジョブの再開例

```
begin
    dbms_mgwadm.reset_subscriber('sub_aq2mq');
end;
```

サブスクライバおよびスケジュールの変更

選択ルール、変換および例外キューは、サブスクライバの作成後に変更できます。DBMS_MGWADM.NO_CHANGE という値は、パラメータの値が変更されていないことを示します。

サブスクライバおよびスケジュールの変更例

```
begin
  dbms_mgwadm.alter_subscriber(
    subscriber_id   => 'sub_aq2mq',             -- MGW subscriber name
    rule            => dbms_mgwadm.NO_CHANGE,   -- selection rule not changed
                                                -- not used with MQSeries
    transformation  => dbms_mgwadm.NO_CHANGE,   -- transformation invoked on
                                                -- dequeue not changed
    exception_queue => 'mgwuser.my_ex_queue'); -- register exception
                                                -- queue: same type as source
end;
```

サブスクライバおよびスケジュールは、メッセージ・ゲートウェイ・エージェントが実行中であるかどうかに関係なく変更できます。

サブスクライバおよびスケジュールの削除

通常、サブスクライバの削除は、メッセージ・ゲートウェイ・エージェントがログ・キューのクリーンアップ、および Oracle 以外のメッセージ・システムのサブスクライバの削除などのクリーンアップ・アクティビティを実行できるように、エージェントの実行中に行う必要があります。

スケジュールおよびサブスクライバの削除例

```
begin
    dbms_mgwadm.unschedule_propagation('sch_aq2mq');
end;

begin
    dbms_mgwadm.remove_subscriber('sub_aq2mq', dbms_mgwadm.NO_FORCE);
end;
```

2 番目の引数では、ゲートウェイがこのサブスクライバに関連するすべてのクリーンアップ操作を実行できない場合でも、このプロシージャを正常に終了させるかどうかを指定します。有効な値は、DBMS_MGWADM.NO_FORCE および DBMS_MGWADM.FORCE です。DBMS_MGWADM.NO_FORCE が指定され、メッセージ・ゲートウェイ・エージェントが実行中でない場合、サブスクライバの状態は DELETE_PENDING になります。メッセージ・ゲートウェイ・エージェントが起動されると、クリーンアップ操作が行われます。DBMS_MGWADM.FORCE が指定された場合は、すべてのクリーンアップ操作が実行されなかった場合でもプロシージャが正常に終了します。

選択ルール

選択ルールでは、メッセージ・システムからデキューするメッセージを選択するためのオプションのサブスクライバ・ルールを指定します。アドバンスト・キューイングのルールは AQ のサブスクライバ・ルールに対応します。MQSeries では選択ルールは使用されません。

変換の使用

メッセージ・ゲートウェイの多くのアプリケーションでは、変換を提供する必要があります。任意のユーザー定義型ペイロードを持つ AQ キューからメッセージを伝播するメッセージ・ゲートウェイには、メッセージ・ゲートウェイの標準のユーザー定義型に対するマッピングが必要です。同様に、任意のユーザー定義型ペイロードを持つ AQ キューにメッセージを伝播するメッセージ・ゲートウェイには、メッセージ・ゲートウェイの標準のユーザー定義型からのマッピングが必要です。これが、変換ジョブです。発信サブスクライバに登録された変換は、伝播中にメッセージ・ゲートウェイで AQ ソース・キューからのデキューが行われた場合にアドバンスト・キューイングによって実行されます。着信サブスクライバに登

録された変換は、伝播中にメッセージ・ゲートウェイで AQ 宛先キューへのエンキューが行われた場合にアドバンスト・キューイングによって実行されます。

たとえば、trans_sampleadt_to_mgw_basic は、次の署名を持つ変換ファンクションを表すストアド・プロシージャです。

変換ファンクションの署名例

```
FUNCTION trans_sampleadt_to_mgw_basic(in_msg IN mgwuser.sampleADT)
RETURN sys.mgw_basic_msg_t;
```

次のとおり、DBMS_TRANSFORM.CREATE を使用して変換を作成します。

```
begin
  dbms_transform.create_transformation(
        schema         => 'mgwuser',
        name           => 'sample_adt_to_mgw_basic',
        from_schema    => 'mgwuser',
        from_type      => 'sampleadt',
        to_schema      => 'sys',
        to_type        => 'mgw_basic_msg_t',
        transformation => 'mgwuser.trans_sampleadt_to_mgw_basic(user_data)');
end;
```

変換の作成後、サブスクライバの作成時にこの変換をメッセージ・ゲートウェイに登録できます。

```
begin
  dbms_mgwadm.add_subscriber(
    subscriber_id    => 'sub_aq2mq',                       -- MGW subscriber name
    propagation_type => dbms_mgwadm.outbound_propagation, -- outbound propaga
    queue_name       => 'mgwuser.srcq',          -- AQ queue name (source queue)
    destination      =>'destq@mqlink',           -- MGW foreign queue with link
                                                 -- (destination queue)
    transformation => 'mgwuser.sample_adt_to_mgw_basic');  -- transformation
                                                            -- invoked on dequeue
end;
```

例外キュー

例外キューには、変換が失敗したメッセージが格納されます。このキューは、伝播ソース・キューと同じメッセージ・システムに存在する必要があります。例外キューが指定されている場合、変換が失敗したメッセージは宛先キューではなく例外キューに移されます。サブスクライバで例外キューが指定されていない場合、メッセージ変換に失敗すると、伝播ジョブが停止します。

発信伝播では、例外キューが既存の AQ キューを参照できる必要があります。また、ソース・キューおよび例外キューのペイロード型が一致する必要があります。例外キューは、

キュー・タイプを EXCEPTION_QUEUE ではなく NORMAL_QUEUE として作成する必要があります。

着信伝播では、例外キューは Oracle 以外のメッセージ・システムに登録済のキューである必要があります。また、ソース・キューおよび例外キューが、同じメッセージ・システム・リンクを使用する必要があります。

伝播ジョブの監視

MGW_SUBSCRIBERS ビューを使用して、サブスクライバの既存の構成を確認し、伝播ジョブの状態を監視できます。構成された情報の他に、ビューの列には、（メッセージ・ゲートウェイ・エージェントの起動後に）ジョブに対して伝播されたメッセージの合計数、伝播が失敗した回数、伝播ジョブの状態およびエラー情報が表示されます。

サブスクライバの状態値が ENABLED の場合は、サブスクライバが使用可能にされていることを示します。（伝播ジョブが有効にされているという意味ではないことに注意してください。伝播ジョブを有効にするには、サブスクライバおよび関連付けられたスケジュールの両方が使用可能にされている必要があります。）DELETE_PENDING は、サブスクライバの削除が保留されていることを示します。この状況は、DBMS_MGWADM.REMOVE_SUBSCRIBER がコールされ、サブスクライバに関連する特定のクリーンアップ・タスクが未完了の場合に発生する可能性があります。Oracle9*i* リリース 2（9.2）では、サブスクライバの状態は、DELETE_PENDING でないかぎり、常に ENABLED です。

エラー情報には、配信が失敗した回数、最新のエラー・メッセージ、最新のエラー日付および最新のエラー時刻が含まれます。失敗の回数が 16 回に達すると、伝播が停止します。18-22 ページの「伝播ジョブのリセット」を参照してください。

伝播されたメッセージの確認例

```
SQL> select subscriber_id, queue_name, propagated_msgs, exceptionq_msgs from mgw_
subscribers;

SUBSCRIBER_ID     QUEUE_NAME     PROPAGATED_MSGS     EXCEPTIONQ_MSGS
------------------------------------------------------------------------
SUB_AQ2MQ         MGWUSER.SRCQ   1014                10
```

エラーの確認例

```
SQL> select queue_name, failures, last_error_msg from mgw_subscribers where
subscriber_id = 'SUB_AQ2MQ';

QUEUE_NAME     FAILURES    LAST_ERROR_MSG
----------------------------------------
MGWUSER.SRCQ   0
```

MGW_SCHEDULES ビューを使用して、構成されたスケジュールおよび使用可能にされたスケジュールを確認できます。

構成および使用可能にされたスケジュールの確認例

```
SQL> select schedule_id, schedule_disabled from MGW_SCHEDULES;

SCHEDULE_ID    SCH
------------------
SCH_AQ2MQ      N
(N = not disabled; that is, enabled)
```

メッセージ・ゲートウェイのログ・ファイルの監視

メッセージ・ゲートウェイ・エージェントの状態、履歴およびエラーは、メッセージ・ゲートウェイのログ・ファイルに記録されます。デフォルトでは、このログ・ファイルは $ORACLE_HOME/mgw/log ディレクトリに存在します。ログ・ファイルには更新とエラーの両方がレポートされるため、ログ・ファイルを監視する必要があります。メッセージ・ゲートウェイ・エージェントが起動されるたびに、異なるログ・ファイルが作成されます。

サンプル・ログ・ファイル

次のサンプル・ログ・ファイルは、メッセージ・ゲートウェイ・エージェントの起動を示しています。このファイルには、トレース情報およびエラーが書き込まれています。

```
Mon Sep 10 10:27:35 2001
  MGW  C-Bootstrap  0  process-id=4313
Bootstrap program starting
Mon Sep 10 10:27:36 2001
  MGW  C-Bootstrap  0  process-id=4313
JVM created -- heapsize = 64
>>2001-09-10 10:27:38  MGW  AdminMgr  0 LOG
Connecting to database using connect string = jdbc:oracle:oci8:@
>>2001-09-10 10:27:55  MGW  Engine  0 1
Agent is initializing...
>>2001-09-10 10:27:56  MGW  MQD  0 LOG
Creating MQSeries messaging link:
  link          : MQLINK
  queue manager : mars.queue.manager
  channel       : kbchannel
  host          : pdsun-dev10.us.oracle.com
  port          : 1414
  user          :
  connections   : 1
  inbound logQ  :
  outbound logQ : kblogqueue
>>2001-09-10 10:27:56  MGW  AQD  0 LOG
Creating AQ messaging link:
  link          : oracleMgwAq
  database      :
  user          : MGWAGENT
  connections   : 10
```

```
  inbound logQ  : sys.mgw_recv_log
  outbound logQ : sys.mgw_send_log
>>2001-09-10 10:27:56  MGW  Engine  0 7
Queue DESTQ@MQLINK has been registered.
>>2001-09-10 10:27:56  MGW  Engine  0 9
Propagation Schedule SCH_AQ2MQ has been added.
>>2001-09-10 10:27:56  MGW  Engine  0 13
MGW subscriber SUB_AQ2MQ has been created.
>>2001-09-10 10:27:56  MGW  Engine  0 18
MGW subscriber SUB_AQ2MQ has been activated.
>>2001-09-10 10:27:56  MGW  Engine  0 13
MGW subscriber SUB_AQ2MQ(MGWUSER.SRCQ --> DESTQ@MQLINK) has been created.
>>2001-09-10 10:27:56  MGW  Engine  0 2
Agent is up and running.
```

起動時に構成情報が読み込まれるか、または動的構成が発生すると、その情報はログに書き込まれます。サンプル・ログ・ファイルには、リンク、登録済の外部キュー、サブスクライバおよびスケジュールが作成されていることが示されています。また、サブスクライバがアクティブにされていることも示されています。さらに、ログにはすべてのエラーも示されています。最後の行は、メッセージ・ゲートウェイ・エージェントが起動されて、実行中であることを示しています。

メッセージの変換

メッセージ・ゲートウェイでは、伝播中に、ソース側メッセージ・システム固有のメッセージ・フォーマットが宛先側メッセージ・システム固有のメッセージ・フォーマットに変換されます。メッセージ・ゲートウェイでは、標準型および AQ ベースのモデルを使用して変換が行われます。

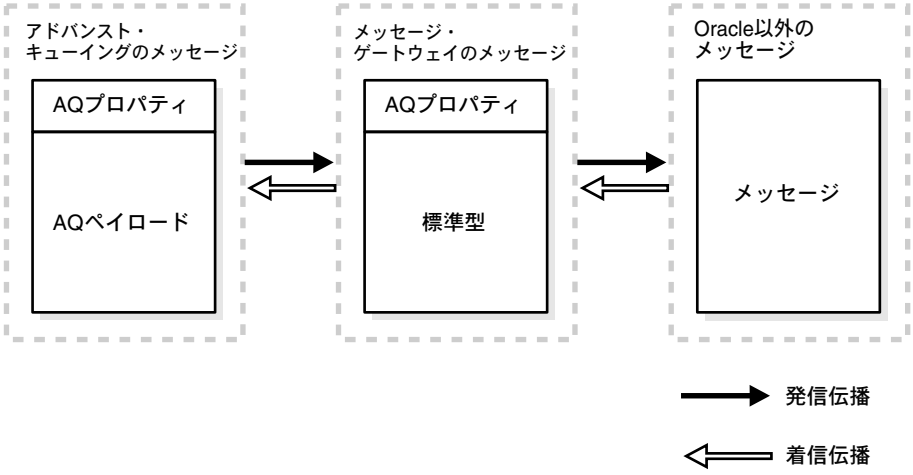
メッセージの変換処理

ゲートウェイによってメッセージが伝播される場合、メッセージは、ソース・キューのシステム固有のフォーマットから宛先キューのシステム固有のフォーマットに変換されます。

システム固有のメッセージには、メッセージ・ヘッダーおよびメッセージ本体が含まれます。ヘッダーには、アドバンスト・キューイングのメッセージ・プロパティ、MQSeries の固定ヘッダーなどの固定ヘッダー・フィールドが含まれます。このフィールドは、メッセージ・システム内のすべてのメッセージに存在します。本体には、AQ のペイロード、MQSeries のメッセージ本体などのメッセージの内容が含まれます。メッセージ・ゲートウェイでは、メッセージ・ヘッダーおよびメッセージ本体の両方の構成要素が変換されます。

メッセージ変換は、図 18-2 に示すとおり、2 段階で行われます。メッセージは、ソース・キューのシステム固有のフォーマットからゲートウェイの内部メッセージ・フォーマットに変換され、次に、ゲートウェイの内部メッセージ・フォーマットから宛先キューのシステム固有のフォーマットに変換されます。

図 18-2 メッセージの変換



ゲートウェイ・エージェントでは、AQ メッセージ・プロパティと同じヘッダーおよびゲートウェイの標準型のオブジェクトである本体で構成された、内部メッセージ・フォーマットが使用されます。

メッセージ・ゲートウェイの標準型

メッセージ・ゲートウェイでは標準型が定義され、アドバンスト・キューイングと Oracle 以外のメッセージ・システム間でのメッセージ変換がサポートされます。標準型は、メッセージ型を、Oracle9*i* データベースでの PL/SQL の抽象データ型（ユーザー定義型）の形式で表したものです。Oracle9*i* リリース 2（9.2）では、MGW_BASIC_MSG_T 標準型によって、アドバンスト・キューイングと MQSeries 間の変換がサポートされています。

MGW_BASIC_MSG_T は、メッセージ・ヘッダーおよび TEXT または RAW（バイト）メッセージ本体で構成されるメッセージを表すために使用されます。メッセージ・ヘッダーは、MGW_NAME_VALUE_T 型のオブジェクトである { 名前 , 値 } の組の集合として表されます。

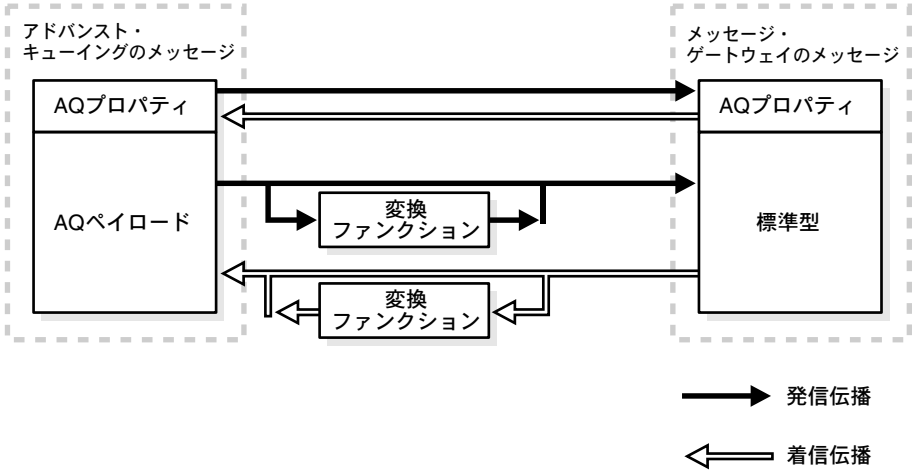
> **参照：** 次の項目の詳細は、『Oracle9*i* PL/SQL パッケージ・プロシージャおよびタイプ・リファレンス』の「DBMS_MGWMSG」を参照してください。
>
> - MGW_BASIC_MSG_T の構文および属性の情報
> - MGW_NAME_VALUE_T の構文および属性の情報
> - MGW_NAME_VALUE_T 値の型に対する定数のリスト
> - MGW_NAME_VALUE_ARRAY_T に対するヘルパー・ルーチン

アドバンスト・キューイングに対するメッセージ変換

システム固有の AQ メッセージは、AQ メッセージ・プロパティ、および RAW 型またはユーザー定義型のいずれかのメッセージ・ペイロードで構成されます。

メッセージ・ゲートウェイ・エージェントは、システム固有の AQ メッセージ・フォーマットと内部メッセージ・フォーマット間でメッセージを変換します。図 18-3 に、AQ ドライバによって実行されるメッセージ変換を示します。

図 18-3 AQ メッセージの変換



発信伝播の場合、AQ キューからメッセージがデキューされると、ゲートウェイ・エージェントによって AQ メッセージの AQ メッセージ・プロパティが内部メッセージの AQ メッセージ・プロパティにマップされ、AQ ペイロードが標準型のオブジェクトに変換されて、内部メッセージが構成されます。

着信伝播の場合、Oracle 以外のドライバから内部メッセージが受信されると、ゲートウェイ・エージェントによって標準型メッセージが AQ ペイロードに変換され、次に、そのペイロードおよび内部メッセージの AQ メッセージ・プロパティを持つメッセージがエンキューされます。

エージェントは、ペイロード型が RAW または SYS.MGW_BASIC_MSG_T のメッセージを AQ キューとの間で直接エンキューおよびデキューできます。エージェントでは、2 つのペイロード型と標準型間でのマッピングが自動的に実行されます。ペイロード型が RAW または SYS.MGW_BASIC_MSG_T 以外の場合、AQ ペイロード型と標準型間で変換を行うには、ユーザーが提供する変換を実行する必要があります。

通常、発信伝播の場合は、AQ ペイロード型またはユーザーが提供する変換の出力型が、RAW または SYS.MGW_BASIC_MSG_T のいずれかである必要があります。着信伝播の場合は、AQ ペイロードまたはユーザーが提供する変換の入力型が、RAW または SYS.MGW_BASIC_MSG_T のいずれかである必要があります。

RAW 型の AQ ペイロードの変換

発信伝播の場合は、次のルールが適用されます。

- RAW 型の AQ ペイロードは、常に、RAW 本体を持つ MGW_BASIC_MSG_T 標準型メッセージにマップされます。MGW_BASIC_MSG_T.header は、NULL に設定されます。この場合、メッセージ変換が失敗することはありません。

着信伝播の場合は、次のルールが適用されます。

- MGW_BASIC_MSG_T 標準型メッセージは、次のとおりマップされます。
  - サイズが 32KB 以下の RAW 本体の場合、RAW 本体は RAW ペイロードに直接マップされます。この場合、メッセージ変換が失敗することはありません。
  - サイズが 32KB を超える RAW 本体の場合、メッセージ変換は失敗します。
  - TEXT 本体の場合、メッセージ変換は失敗します。
  - TEXT および RAW の両方の本体を含む標準型メッセージの場合、メッセージ変換は失敗します。

MGW_BASIC_MSG_T 型の AQ ペイロードの変換

発信伝播の場合は、次のルールが適用されます。

- SYS.MGW_BASIC_MSG_T 型の AQ ペイロードは、常に MGW_BASIC_MSG_T 標準型メッセージにマップされます。
- RAW 本体に小さい値および大きい値の両方が設定されている場合、メッセージ変換は失敗します。
- TEXT 本体に小さい値および大きい値の両方が設定されている場合、メッセージ変換は失敗します。

着信伝播の場合は、次のルールが適用されます。

- MGW_BASIC_MSG_T 標準型メッセージは、直接マップされます。この場合、メッセージ変換は失敗することはありません。

変換の使用

メッセージ・ゲートウェイでは、AQ メッセージ変換を使用して、AQ キューのペイロードとゲートウェイの標準型間で変換を行うことができます。メッセージ・ゲートウェイの管理者は、DBMS_TRANSFORM パッケージを使用して変換を作成した後で、DBMS_MGWADM.ADD_SUBSCRIBER および DBMS_MGWADM.ALTER_SUBSCRIBER を使用して、変換に使用するゲートウェイ・サブスクライバを構成できます。

発信伝播の場合、ゲートウェイ・エージェントによって AQ キューからメッセージがデキューされると、変換が実行されます。着信伝播の場合、ゲートウェイ・エージェントによって AQ キューにメッセージがエンキューされると、変換が実行されます。

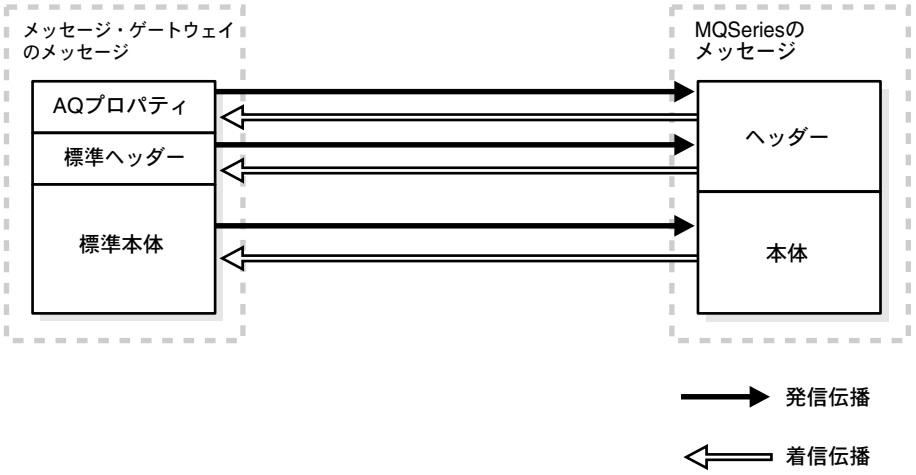
変換は、常に、ゲートウェイ・エージェントによって行われます。そのため、ゲートウェイ・エージェント・ユーザーは変換ファンクションおよび AQ ペイロード型に対する実行権限を所有している必要があります。実行権限がない場合は、PUBLIC に実行権限を付与するか、またはゲートウェイ・エージェントのユーザーに直接実行権限を付与します。

MQSeries に対するメッセージ変換

メッセージ・ゲートウェイの MQSeries ドライバでは、内部メッセージ・フォーマットと MQSeries システム固有のメッセージ・フォーマット間で変換が行われます。MQSeries システム固有のメッセージは、固定メッセージ・ヘッダーおよびメッセージ本体で構成されます。メッセージ本体は、TEXT 値または RAW（バイト）値のいずれかとして処理されます。

図 18-4 に、MQSeries ドライバによって実行されるメッセージ変換を示します。発信伝播の場合、ドライバによって、AQ メッセージ・プロパティおよび標準型メッセージが固定ヘッダーおよびメッセージ本体を含むシステム固有のメッセージにマップされます。着信伝播の場合、ドライバによって、システム固有のメッセージが一連の AQ メッセージ・プロパティおよび標準型メッセージにマップされます。

図 18-4 MQSeries メッセージの変換



発信伝播の場合、MGW_BASIC_MSG_T 標準型メッセージは、次のとおり、MQSeries システム固有のメッセージにマップされます。

- MQSeries の固定ヘッダー・フィールドは、標準型メッセージの内部 AQ メッセージ・プロパティおよび MGW_BASIC_MSG_T.header 属性に基づきます。

  値が指定されていない場合、AQ メッセージ・プロパティに基づく特定の MQSeries ヘッダー・フィールドのデフォルトのマッピングについては、表 18-1 を参照してください。

  ドライバによって MGW_BASIC_MSG_T.header 内の { 名前 , 値 } の組（表 18-4 を参照）が検索され、検索された各値が MQSeries ヘッダー・フィールドに使用されます。認識されない名前または不適切な値を含む { 名前 , 値 } の組は無視されます。

- 標準型メッセージに TEXT 本体が含まれている場合、MQSeries の *format* ヘッダー・フィールドは MQFMT_STRING に、メッセージ本体は TEXT 値に設定されます。
- 標準型メッセージに RAW 本体が含まれている場合、MQSeries の *format* ヘッダー・フィールドは "MGW_Byte" に、メッセージ本体は RAW 値に設定されます。
- TEXT および RAW の両方の本体を含む標準型メッセージの場合、メッセージ変換は失敗します。
- 標準型メッセージに TEXT 本体または RAW 本体のいずれも含まれていない場合、メッセージ本体は設定されず、MQSeries の *format* ヘッダー・フィールドは MQFMT_NONE に設定されます。

着信伝播の場合、MQSeries システム固有のメッセージは、次のとおり、MGW_BASIC_MSG_T 標準型メッセージにマップされます。

- 特定の MQSeries ヘッダー・フィールドは、表 18-1 に示すとおり、AQ メッセージ・プロパティにマップされます。
- 標準型メッセージの MGW_BASIC_MSG_T.header 属性は、表 18-4 に示すとおり、MQSeries ヘッダー・フィールドに基づいて { 名前 , 値 } の組に設定されます。
- MQSeries の *format* ヘッダー・フィールドが MQFMT_STRING の場合、MQSeries のメッセージ本体は TEXT として処理され、その値は MGW_BASIC_MSG_T.text_body にマップされます。他のフォーマット値の場合、メッセージ本体は RAW として処理され、その値は MGW_BASIC_MSG_T.raw_body にマップされます。

メッセージ・ヘッダーの変換

メッセージ・ゲートウェイでは、AQ メッセージ・プロパティと、同じセマンティクスの AQ メッセージ・プロパティに対応する値を含む Oracle 以外のメッセージ・ヘッダー・フィールドとの間で、デフォルトのマッピングが行われます。メッセージ・ゲートウェイによってマッピングが行われない場合、メッセージ・ヘッダー・フィールドはデフォルト値（通常、メッセージ・システムによって定義されたデフォルト値）に設定されます。

メッセージ・ゲートウェイでは、AQ メッセージ・プロパティに対して { 名前 , 値 } の組を定義し、Oracle 以外のメッセージ・システムに対してヘッダー・フィールドを定義して、システム固有のメッセージ・ヘッダーを変換し、デフォルトの値をオーバーライドできます。{ 名前 , 値 } の組は、ヘッダー・プロパティと呼ばれます。任意の伝播ジョブのヘッダー・プロパティにアクセスできるかどうかは、関連するメッセージ・システムによって異なり、AQ ペイロード型か変換かによっても異なります。

メッセージ・ヘッダーのデフォルト・マッピング

表 18-1 に、AQ メッセージ・プロパティと MQSeries ヘッダー・フィールド間でのデフォルト・マッピングを示します（カッコ内の数字については、表内の注意を参照）。

表 18-1 AQ メッセージ・プロパティと MQSeries ヘッダー・フィールド間のデフォルト・マッピング

AQ メッセージ・プロパティ	MQSeries ヘッダー・フィールド	発信マッピング（AQ 値から MQSeries 値へ）	着信マッピング（MQSeries 値から AQ 値へ）
priority	priority	AQ 値の 0、1、2、3、4、5、6、7、8、9 は、それぞれ MQSeries 値の 9、8、7、6、5、4、3、2、1、0 にマップされます。 0（ゼロ）未満の AQ 値は、MQSeries 値の 9 にマップされます。 10 以上の AQ 値は、MQSeries 値の 0（ゼロ）にマップされます。	MQSeries 値の 0、1、2、3、4、5、6、7、8、9 は、それぞれ AQ 値の 9、8、7、6、5、4、3、2、1、0 にマップされます。
expiration	expiry	時間単位は、0.1 秒単位でマップされます。(1) AQ 値の NEVER は、MQEI_UNLIMITED にマップされます。	時間の単位は、1 秒単位でマップされます。(1) MQEI_UNLIMITED は、NEVER にマップされます。

注意：

1. 発信伝播の場合、AQ の期限切れ値はメッセージのエンキュー時に指定された期限切れ時刻を表すため、その値を使用して残りの Time-To-Live が計算されます。着信伝播の場合、MQSeries の期限切れ値はすでに残りの Time-To-Live を表しているため、直接マッピングが行われます。

アドバンスト・キューイング・ヘッダー・プロパティ

表 18-2 に、AQ メッセージ・プロパティの記述に使用されるメッセージ・ゲートウェイの { 名前、値 } の組の定義を示します。AQ プロパティのヘッダー・プロパティ名には、「MGW_AQ_」という接頭辞が付きます。

AQ キューからメッセージがデキューされると、AQ ドライバによって、デキューされたメッセージのヘッダーに基づいて { 名前 , 値 } の組が生成されます。メッセージがエンキューされると、AQ ドライバによって、これらのプロパティに対する { 名前 , 値 } の組から AQ メッセージ・プロパティが設定されます。

表 18-2 AQ メッセージ・プロパティに対するメッセージ・ゲートウェイの名前

MGW の名前 MGW_NAME_VALUE_T.name	MGW の型 MGW_NAME_VALUE_T.type	AQ メッセージ・プロパティ	用途
"MGW_AQ_priority"	INTEGER_VALUE	priority	エンキュー デキュー
"MGW_AQ_expiration"	INTEGER_VALUE	expiration	エンキュー デキュー
"MGW_AQ_delay"	INTEGER_VALUE	delay	エンキュー デキュー
"MGW_AQ_correlation"	TEXT_VALUE (size 128)	correlation	エンキュー デキュー
"MGW_AQ_exception_queue"	TEXT_VALUE (size 61)	exception_queue	エンキュー デキュー
"MGW_AQ_enqueue_time"	DATE_VALUE	enqueue_time	デキュー
"MGW_AQ_original_msgid"	RAW_VALUE (size 16)	original_msgid	デキュー

AQ キューにメッセージがエンキューされると、AQ ドライバによって、デフォルトのマッピングを持たない AQ メッセージ・プロパティに対するデフォルト値が設定されます（表 18-1 を参照）。対応するヘッダー・プロパティは、表 18-3 に示すとおり設定されます。

表 18-3 AQ メッセージ・プロパティのデフォルト値

AQ メッセージ・プロパティの名前	デフォルト値
priority	1
expiration	NEVER
delay	NO_DELAY
correlation	NULL
exception_queue	NULL

MQSeries ヘッダー・プロパティ

この項では、MQSeries メッセージ・システムに対してサポートされているメッセージ・プロパティについて説明します。表 18-4 に、MQSeries ヘッダー・プロパティの記述に使用されるメッセージ・ゲートウェイの {名前 , 値 } の組の定義を示します（カッコ内の数字については、表内の注意を参照）。MQSeries プロパティのメッセージ・ゲートウェイの名前には、「MGW_AQ_」という接頭辞が付きます。

MQSeries メッセージ・システムからメッセージがデキューされると、MQSeries ドライバによって、デキューされたメッセージ・ヘッダーに基づいて {名前 , 値 } の組が生成され、MGW_BASIC_MSG_T 型の標準型メッセージのヘッダー部分に格納されます。MQSeries にメッセージがエンキューされると、MQSeries ドライバによって、MGW_BASIC_MSG_T 標準型メッセージのヘッダー部分に格納されたこれらのプロパティに対する {名前、値 } の組から、メッセージ・ヘッダーおよびエンキュー・オプションが設定されます。

表 18-4 MQSeries ヘッダー値に対するメッセージ・ゲートウェイの名前

メッセージ・ゲートウェイの名前 mgw_name_value_t.NAME	メッセージ・ゲートウェイの型 MGW_NAME_VALUE_T.type	MQSeries プロパティ名	用途
"MGW_MQ_priority"	INTEGER_VALUE	priority	エンキュー、デキュー
"MGW_MQ_expiry"	INTEGER_VALUE	expiry	エンキュー、デキュー
"MGW_MQ_correlationId"	RAW_VALUE (size 24)	correlationId	エンキュー (1)、デキュー
"MGW_MQ_persistence"	INTEGER_VALUE	persistence	デキュー
"MGW_MQ_report"	INTEGER_VALUE	report	エンキュー (1)、デキュー
"MGW_MQ_messageType"	INTEGER_VALUE	messageType	エンキュー、デキュー
"MGW_MQ_feedback"	INTEGER_VALUE	feedback	エンキュー、デキュー
"MGW_MQ_encoding"	INTEGER_VALUE	encoding	エンキュー、デキュー
"MGW_MQ_characterSet"	INTEGER_VALUE	characterSet	エンキュー、デキュー
"MGW_MQ_format"	TEXT_VALUE (size 8)	format	エンキュー (1)、デキュー
"MGW_MQ_backoutCount"	INTEGER_VALUE	backoutCount	デキュー

表 18-4 MQSeries ヘッダー値に対するメッセージ・ゲートウェイの名前（続き）

メッセージ・ゲートウェイの名前 mgw_name_value_t.NAME	メッセージ・ゲートウェイの型 MGW_NAME_VALUE_T.type	MQSeries プロパティ名	用途
"MGW_MQ_replyToQueueName"	TEXT_VALUE (size 48)	replyToQueueName	エンキュー、デキュー
"MGW_MQ_replyToQueueManagerName"	TEXT_VALUE (size 48)	replyToQueueManagerName	エンキュー、デキュー
"MGW_MQ_userId"	TEXT_VALUE (size 12)	userId	エンキュー、デキュー
"MGW_MQ_accountingToken"	RAW_VALUE (size 32)	accountingToken	エンキュー (1)、デキュー
"MGW_MQ_applicationIdData"	TEXT_VALUE (size 32)	applicationIdData	エンキュー (1)、デキュー
"MGW_MQ_putApplicationType"	INTEGER_VALUE	putApplicationType	エンキュー (1)、デキュー
"MGW_MQ_putApplicationName"	TEXT_VALUE (size 28)	putApplicationName	エンキュー (1)、デキュー
"MGW_MQ_putDateTime"	DATE_VALUE	putDateTime	デキュー
"MGW_MQ_applicationOriginData"	TEXT_VALUE (size 4)	applicationOriginData	エンキュー (1)、デキュー
"MGW_MQ_groupId"	RAW_VALUE (size 24)	groupId	エンキュー (1)、デキュー
"MGW_MQ_messageSequenceNumber"	INTEGER_VALUE	messageSequenceNumber	エンキュー、デキュー
"MGW_MQ_offset"	INTEGER_VALUE	offset	エンキュー、デキュー
"MGW_MQ_messageFlags"	INTEGER_VALUE	messageFlags	エンキュー、デキュー

表 18-4 MQSeries ヘッダー値に対するメッセージ・ゲートウェイの名前（続き）

メッセージ・ゲートウェイの名前 mgw_name_value_t.NAME	メッセージ・ゲートウェイの型 MGW_NAME_VALUE_T.type	MQSeries プロパティ名	用途
"MGW_MQ_originalLength"	INTEGER_VALUE	originalLength	エンキュー、デキュー
"MGW_MQ_putMessageOptions"	INTEGER_VALUE	putMessageOptions（2）	エンキュー（1）

注意：

1. この表での使用方法には、MQSeries の制約が適用されます。たとえば、送信メッセージに対して MGW_MQ_accountingToken が設定されている場合、MQSeries の値は、MGW_MQ_putMessageOptions が MQSeries 定数の MQPMD_SET_ALL_CONTEXT に設定されていないかぎり、オーバーライドされます。
2. MGW_MQ_putMessageOptions は、MQSeries Base Java の Queue.put() メソッドに対する putMessageOptions 引数として使用されます。これは MQSeries のヘッダー情報に含まれないため、実際のメッセージ・プロパティではありません。

   MQSeries Base Java の MQQueueManager.accessQueue メソッドに対する openOptions 引数の値は、DBMS_MGWADM.REGISTER_FOREIGN_QUEUE をコールして MQSeries キューが登録されるときに指定されます。これらの値の間には依存関係が存在する場合があります。たとえば、MGW_MQ_putMessageOptions に MQPMD_SET_ALL_CONTEXT を含めるには、MQ_openMessageOptions キュー・オプションに MQOO_SET_CONTEXT を含める必要があります。

ゲートウェイ・エージェントによって、指定可能なすべての値に MQPMO_SYNCPOINT という値が追加されます。

表 18-5 に、メッセージが MQSeries キューにエンキューされると、ゲートウェイ・エージェントによって MQSeries メッセージ・ヘッダーに設定されるデフォルト値を示します。他のすべてのヘッダー・フィールドには、ゲートウェイ・エージェントによってデフォルト値は設定されません。

表 18-5 MQSeries ヘッダーのデフォルト値

MQSeries プロパティ名	デフォルト値
messageType	MQMT_DATAGRAM
putMessageOption	MQPMO_SYNCPOINT は常に追加されます。表内の注意の（2）を参照してください。

ヘッダー・プロパティの使用例

次の伝播の例では、ヘッダー・プロパティの使用例を示します。

発信伝播に対する MGW_BASIC_MSG_T の使用例

AQ キューから MQSeries キューへの発信伝播ジョブについて考えてみます。MQSeries ドライバでは MGW_BASIC_MSG_T 型のみがサポートされているため、AQ ドライバによって AQ ペイロード型を MGW_BASIC_MSG_T 標準型メッセージに変換できるように、伝播ジョブを構成する必要があります。このためには、ソース・キューのペイロード型を SYS.MGW_BASIC_MSG_T にするか、または出力型が SYS.MGW_BASIC_MSG_T の変換を実行する必要があります。

発信伝播の場合は、MGW_BASIC_MSG_T.header 属性を使用して、宛先キューへのメッセージのエンキュー時に使用されるシステム固有のメッセージ・ヘッダー・プロパティを指定します。この例では、表 18-4 に示したとおり、MQSeries ヘッダー・プロパティに対する { 名前 , 値 } の組が含まれます。

AQ ドライバによって AQ メッセージ・プロパティに対する { 名前 , 値 } の組が生成されますが（表 18-2 を参照）、MQSeries メッセージ・フォーマットではユーザー定義のメッセージ・プロパティの情報を指定できないため、この情報は失われます。

着信伝播に対する MGW_BASIC_MSG_T の使用例

MQSeries キューから AQ キューへの着信伝播ジョブの場合は、MQSeries ドライバによって、常に、システム固有のメッセージが MGW_BASIC_MSG_T 標準型メッセージに変換されます。そのため、AQ ドライバによって標準型メッセージを SYS.MGW_BASIC_MSG_T ペイロード型に変換できるように伝播ジョブを構成する必要があります。このためには、宛先キューのペイロード型を SYS.MGW_BASIC_MSG_T にするか、または入力型が SYS.MGW_BASIC_MSG_T の変換を実行する必要があります。

着信伝播に使用される MGW_BASIC_MSG_T.header 属性には、ソース・キューからデキューされたメッセージのシステム固有のメッセージ・ヘッダー・プロパティに対する { 名前 , 値 } の組が含まれます。この例では、表 18-4 に示したとおり、MQSeries ヘッダー・プロパティに対する { 名前 , 値 } の組が含まれます。

MQSeries のシステム固有のメッセージ・フォーマットではユーザー定義のメッセージ・プロパティの情報を指定できないため、ゲートウェイの MQSeries ドライバで AQ メッセージ・プロパティに使用する値として解釈される値は、指定できません。その結果、宛先キューにエンキューされたメッセージの AQ メッセージ・プロパティは、表 18-1 に示したデフォルト・マッピングおよび残りの（マップされていない）AQ プロパティに対するデフォルト値に基づいて設定されます。

XML メッセージ伝播の使用例

この項では、XML メッセージを使用して、ユーザー定義型ペイロードを含む AQ キューと外部キュー間での伝播の設定方法の例を示します。

この例では、書籍注文をメッセージとして伝播します。AQ キュー AQ_book_orders のペイロード型は、book_order_typ です。外部キュー FQ_book_orders は、XML 文書を格納できます。

次の PL/SQL スクリプトを実行すると、Oracle データベースに、スクリプトの後に表示される 2 つの着信伝播および発信伝播の例に対するエントリが作成されます。データベース・ユーザー mgwuser がスクリプトを実行すると想定します。

```
-- create the type book_order_typ
CREATE OR REPLACE TYPE book_order_typ AS OBJECT
(
  order_no            number,
  book_name        varchar2(100),
  book_isbn          varchar2(15),
  book_amount     number,
  payment             varchar2(30),
  ship_addr           varchar2(160),
  order_date         date
);
/
 -- grant privilege to PUBLIC
GRANT EXECUTE ON book_order_typ to PUBLIC;

BEGIN
    -- create queue table
    dbms_aqadm.create_queue_table(
                   queue_table               => 'book_order_qtab',
                   queue_payload_type => 'book_order_typ',
                   multiple_consumers  => TRUE,
                   compatible                => '8.1');

    -- create the queue
    dbms_aqadm.create_queue(
                   queue_name => 'AQ_book_orders',
                   queue_table => 'book_order_qtab');

    -- start the queue
    dbms_aqadm.start_queue('AQ_book_orders');
END;
/
```

dbms_mgwadm.create_msgsystem_link( ) をコールして、サード・パーティのメッセージ・システムに接続するための fqlink と呼ばれるメッセージ・システム・リンクを作成します。dbms_mgwadm.register_foreign_queue( ) をコールして、サード・パーティのメッセージ・システムの外部キュー FQ_book_orders を登録します。

発信 XML メッセージの伝播例

この例では、XML 文書の形式で、AQ キュー AQ_book_orders から外部キュー FQ_book_orders に書籍注文メッセージを移す伝播を設定します。ユーザーは、DBMS_XMLSCHEMA パッケージを使用して、ユーザー定義型 book_order_typ から XML Schema を生成し、サード・パーティのメッセージ・システム側で XML メッセージを解析および処理できます。

次のスクリプトを実行すると、AQ の書籍注文メッセージを標準型 sys.mgw_basic_msg_t のオブジェクトに格納されている XML 文書に変換するためのファンクションおよび変換が定義されます。ユーザー mgwuser としてスクリプトを実行します。

```
-- create a transformation function
CREATE OR REPLACE FUNCTION fnc_order2basic (book_order IN book_order_typ)
   RETURN sys.mgw_basic_msg_t
   IS
       v_xml   XMLType;
       v_text   varchar(2000);    -- assume book orders in XML document
                                  -- are less than 2000 char long.
       v_basic sys.mgw_basic_msg_t;

   BEGIN
       -- create a XMLType object from the book_order
       v_xml  := XMLType.createXML(book_order, null, null);

       -- convert the XMLType object to XML document (text)
       v_text := v_xml.getStringVal();

       -- store the XML document in a mgw_basic_msg_t obejct
       v_basic := sys.mgw_basic_msg_t.construct;
       v_basic.text_body := sys.mgw_text_value_t(v_text, null);

       return v_basic;
   END fnc_order2basic;
            /

-- grant execute privilege to PUBLIC in order for the agent to be able to call it
GRANT EXECUTE on fnc_order2basic to PUBLIC;

-- create a transformation with the function
BEGIN
```

```
  dbms_transform.create_transformation(
    schema           => 'mgwuser',
    name              => 'order2basic',
    from_schema => 'mgwuser',
    from_type      => 'book_order_typ',
    to_schema     => 'sys',
    to_type          =>'mgw_basic_msg_t',
    transformation => 'mgwuser.fnc_order2basic(source.user_data)');
END;
/
```

MGW_ADMINSTRATOR_ROLE 権限を所有するユーザーとして次のスクリプトを実行し、発信伝播ジョブを作成します。

```
-- create an outbound propagation with the transformation
BEGIN
    dbms_mgwadm.add_subscriber(
      subscriber_id        => 'sub_aq2fq',
      propagation_type => dbms_mgwadm.outbound_propagation,
      queue_name         => 'mgwuser.AQ_book_orders',
      destination          => 'FQ_book_orders@fqlink',
      transformation     => 'mgwuser.order2basic');

    dbms_mgwadm.schedule_propagation(
      schedule_id          => 'sch_aq2fq',
      propagation_type  => dbms_mgwadm.outbound_propagation,
      source                  => 'mgwuser.AQ_book_orders',
      destination           => 'mgwuser.order2basic');
END;
/
```

前述のスクリプトが正常に終了すると、AQ キューに送信されたすべての書籍注文メッセージが、PL/SQL 型 book_order_typ に関連付けられた XML Schema に準拠する XML 文書として、サード・パーティ・キューに伝播されます。

着信 XML メッセージの伝播例

この例では、PL/SQL 型 book_order_typ に関連付けられた XML Schema に準拠する XML 文書として、外部キュー FQ_book_orders から AQ キュー AQ_book_orders に書籍注文を移す伝播を設定します。ユーザーは、DBMS_XMLSCHEMA パッケージを使用して、ユーザー定義型 book_order_typ から XML Schema を生成し、有効な XML 書籍注文メッセージを生成する必要があります。

次のスクリプトを実行すると、標準型 sys.mgw_basic_msg_t のオブジェクトに格納された XML 文書の形式の書籍注文を、ユーザー定義型 book_order_typ のオブジェクトに変換するためのファンクションおよび変換が定義されます。ユーザー mgwuser としてスクリプトを実行します。

```
-- create a transformation function
CREATE OR REPLACE  FUNCTION fnc_basic2order(basic IN sys.mgw_basic_msg_t)
   RETURN book_order_typ
   IS
       v_xml     XMLType;
       v_text     varchar(2000);    -- assume book orders in XML document
                                                  -- are less than 2000 char long
       v_order   book_order_typ;
   BEGIN
       v_text := basic.text_body.small_value;

       v_xml := XMLType.createXML(v_text);

       v_xml.toObject(v_order);

       return v_order;
   END fnc_basic2order;
/

-- grant execute privilege to PUBLIC in order for the agent to be able to call it
GRANT EXECUTE on fnc_basic2order to PUBLIC;

               -- create a transformation with the function
BEGIN
  dbms_transform.create_transformation(
    schema           => 'mgwuser',
    name              => 'basic2order',
    from_schema => 'sys',
    from_type      => 'mgw_basic_msg_t',
    to_schema     => 'mgwuser',
    to_type          => 'book_order_typ',
    transformation => 'mgwuser.fnc_basic2order(source.user_data)');
END;
/
```

MGW_ADMINSTRATOR_ROLE 権限を所有するユーザーとして次のスクリプトを実行し、着信伝播ジョブを作成します。

```
-- create an inbound propagation with the transformation
BEGIN
     dbms_mgwadm.add_subscriber(
      subscriber_id       => 'sub_fq2aq',
      propagation_type => dbms_mgwadm.inbound_propagation,
      queue_name         => 'FQ_book_orders@fqlink'',
      destination           => 'mgwuser.AQ_book_orders',
      transformation     => 'mgwuser.basic2order');

-- create a schedule for the inbound propagation
     dbms_mgwadm.schedule_propagation(
       schedule_id           => 'sch_fq2aq',
       propagation_type  => dbms_mgwadm.inbound_propagation,
       source                    => 'FQ_book_orders@fqlink',
       destination            => 'mgwuser.AQ_book_orders');
END;
/
```

前述のスクリプトが正常に終了すると、PL/SQL 型 book_order_typ に関連付けられた XML Schema に準拠する XML 文書としてサード・パーティ・キューに送信されたすべての書籍注文メッセージが、AQ キューに伝播されます。

mgw.ora 初期化ファイル

メッセージ・ゲートウェイは、メッセージ・ゲートウェイ・エージェントの起動時に読み込まれるテキスト・ファイルから、追加の初期化情報を取得できます。この初期化ファイルはオプションですが、メッセージ・ゲートウェイ・エージェントに必要な環境の設定に使用することをお薦めします。たとえば、初期化ファイルを使用すると、ライブラリ・パスおよび CLASSPATH を簡単に設定できる場合があります。これは、通常、これらのパスには、Oracle 以外のメッセージ・システムに加えて、Oracle データベースへのアクセスに必要な共有ライブラリおよび Java クラスへのパスを含める必要があるためです。

名前 : mgw.ora

位置 : <ORACLE_HOME>/mgw/admin

ファイルの内容

メッセージ・ゲートウェイの初期化ファイルには、初期化パラメータ、環境変数および Java プロパティを設定するための行が含まれています。各エンティティは 1 行で指定する必要があります。たとえば、複数の行にまたがって初期化パラメータを指定することはできません。先頭の空白は、すべての場合に切り捨てられます。

注意 : 次のすべての例は、このマニュアルでは 1 行で表示されていない場合もありますが、初期化ファイルでは 1 行のみで構成する必要があります。

- 初期化パラメータ : 通常、初期化パラメータは、「<name>=<value><NL>」という書式の行で指定されます。ここで、<name> はパラメータ名、<value> はパラメータの値、<NL> は改行を表します。

  例 : `log_level = 0`

- 環境変数 : CLASSPATH、LD_LIBRARY_PATH などの環境変数を設定すると、メッセージ・ゲートウェイ・エージェントで必要なライブラリ、共有オブジェクト、Java クラスなどを検索できます。環境変数は、「set <env var>= <value><NL>」または「setenv <env var>=<value><NL>」という書式の行で指定されます。ここで、<env var> は設定する環境変数の名前、<value> は環境変数の値、<NL> は改行を表します。

  例 : `set classpath = /myOracleHome/mgw/lib/mgw.jar:<plus_other_required_files>`

- Java プロパティ : Java プロパティは、メッセージ・ゲートウェイ・エージェントの Java VM の作成時に設定できます。Java プロパティは、「setJavaProp <prop name>=<value><NL>」という書式の行で指定されます。ここで、<prop name> は設定する Java プロパティの名前、<value> は Java プロパティの値、<NL> は改行の文字を表します。

  例 : `setJavaProp java.compiler = none`

- コメント行 : コメント行は、行の先頭に # という文字を使用して指定されます。

初期化パラメータ

log_directory

用途 : メッセージ・ゲートウェイのログ・ファイルまたはトレース・ファイルが作成されるディレクトリを指定します。

書式 : `log_directory = <value>`

デフォルト : `<ORACLE_HOME>/mgw/log`

例 : `log_directory = /private/mgwlog`

log_level

用途： メッセージ・ゲートウェイ・エージェントによって記録されるロギングの詳細レベルを指定します。ロギング・レベルは、エージェントの実行中に dbms_mgwadm.set_log_level の API によって動的に変更できます。ロギング・レベルには、常に 0（ゼロ）を指定することをお薦めします。

書式： log_level = <value>

値：

基本的なロギングの場合は、0 を使用します。
これは、dbms_mgwadm.BASIC_LOGGING と同じです。

低レベルのトレースの場合は、1 を使用します。
これは、dbms_mgwadm.TRACE_LITE_LOGGING と同じです。

高レベルのトレースの場合は、2 を使用します。
これは、dbms_mgwadm.TRACE_HIGH_LOGGING と同じです。

デバッグ・トレースの場合は、3 を使用します。
これは、dbms_mgwadm.TRACE_DEBUG_LOGGING と同じです。

デフォルト： basic logging (0)

例： log_level = 0

環境変数

ユーザーは、メッセージ・ゲートウェイの処理環境を直接制御できないため、初期化ファイルを使用して特定の環境変数を設定する必要があります。これらの環境変数は、set パラメータを使用して設定します（18-9 ページの「mgw.ora 初期化ファイルの変更」を参照）。現在メッセージ・ゲートウェイ・エージェントによって使用されている環境変数は、CLASSPATH、LD_LIBRARY_PATH、MGW_PRE_PATH および ORACLE_SID です。

次のすべての例は、このマニュアルでは 1 行で表示されていない場合もありますが、初期化ファイルでは 1 行のみで構成する必要があります。

CLASSPATH

用途： Java Virtual Machine によって、MGW エージェントに必要な Java クラスの検索に使用されます。

書式： `set CLASSPATH=<value>`

例： 次の例では、Oracle AQ と MQSeries 間でメッセージ・ゲートウェイ伝播を行うために含める必要があるクラスを指定します。

```
set CLASSPATH =
/myOracleHome/jdbc/lib/classes12.zip:/myOracleHome/jdk/jre/lib/i18n.jar:/myOracleHom
e/jdk/jre/lib/rt.jar:/myOracleHome/sqlj/lib/runtime12.zip/myOracleHome/sqlj/lib/tran
slator.zip:/myOracleHome/jdbc/lib/nls_
charset12.zip:/myOracleHome/mgw/classes/mgw.jar:/opt/mqm/java/lib/com.ibm.mq.jar:/op
t/mqm/java/lib
```

LD_LIBRARY_PATH

用途： MGW プロセスによって、外部ライブラリの検索に使用されます。Windows NT の場合は必要ありません。

書式： set LD_LIBRARY_PATH=<value>

例： 次の例では、Oracle AQ と MQSeries 間で伝播を行うためにメッセージ・ゲートウェイに必要なライブラリへのパスを指定します。

```
set LD_LIBRARY_PATH =
/myOracleHome/jdk/jre/lib/sparc:/myOracleHome/rdbms/ib:/myOracleHome/lib:/opt/mqm/ja
va/lib
```

MGW_PRE_PATH

用途： メッセージ・ゲートウェイ・プロセスによって継承されたパスの先頭に追加されます。Windows NT の場合、この変数を設定してライブラリ `jvm.dll` が存在する位置を指定する必要があります。他のオペレーティング・システムの場合、現在、この変数を設定する必要はありません。

書式： `set MGW_PRE_PATH=<value>`

例： 次の例では、ライブラリが存在する位置を指定します。

```
set MGW_PRE_PATH=¥myOracleHome¥jdk¥jre¥bin¥classic
```

ORACLE_SID

用途： メッセージ・ゲートウェイの構成時にサービス名が指定されていない場合に使用されます。

書式： `set ORACLE_SID=<value>`

例： `set ORACLE_SID=my_sid`

Java プロパティ

現在使用されていません。

A

Oracle Advanced Queuing の使用例

この付録では、様々なプログラミング環境での使用例を掲載します。

- キュー表およびキューの作成
  - オブジェクト型のキュー表およびキューの作成
  - RAW 型のキュー表およびキューの作成
  - 優先順位指定メッセージのキュー表およびキューの作成
  - マルチ・コンシューマ・キュー表およびキューの作成
  - 伝播のデモ用のキューの作成
  - Java AQ の例の設定
  - Java AQ セッションの作成
  - キュー表およびキューの作成 : Java
  - キューの作成およびエンキュー / デキューの開始 : Java
  - マルチ・コンシューマ・キューの作成およびサブスクライバの追加 : Java
- メッセージのエンキューおよびデキュー
  - オブジェクト型メッセージのエンキューおよびデキュー : PL/SQL
  - オブジェクト型メッセージのエンキューおよびデキュー : Pro*C/C++
  - オブジェクト型メッセージのエンキューおよびデキュー : OCI
  - オブジェクト型メッセージ（CustomDatum インタフェース）のエンキューおよびデキュー : Java
  - オブジェクト型メッセージ（SQLData インタフェース使用）のエンキューおよびデキュー : Java

- RAW 型メッセージのエンキューおよびデキュー : PL/SQL
- RAW 型メッセージのエンキューおよびデキュー : Pro*C/C++
- RAW 型メッセージのエンキューおよびデキュー : OCI
- RAW 型メッセージのエンキュー : Java
- メッセージのデキュー : Java
- ブラウズ・モードでのメッセージのデキュー : Java
- 優先順位によるメッセージのエンキューおよびデキュー : PL/SQL
- 優先順位によるメッセージのエンキュー : Java
- プレビュー後のメッセージのデキュー : PL/SQL
- 遅延および期限切れによるメッセージのエンキューおよびデキュー : PL/SQL
- 相関識別子およびメッセージ ID によるメッセージのエンキューおよびデキュー : Pro*C/C++
- 相関識別子およびメッセージ ID によるメッセージのエンキューおよびデキュー : OCI
- マルチ・コンシューマ・キューでのメッセージのエンキューおよびデキュー : PL/SQL
- マルチ・コンシューマ・キューでのメッセージのエンキューおよびデキュー : OCI
- メッセージのグループ化によるメッセージのエンキューおよびデキュー : PL/SQL
- LOB 属性を含むオブジェクト型メッセージのエンキューおよびデキュー : PL/SQL
- LOB 属性を含むオブジェクト型メッセージのエンキューおよびデキュー : Java

■ 伝播

- リモートのサブスクライバ / 受信者用のメッセージのマルチ・コンシューマ・キューへのエンキューおよび伝播のスケジュール : PL/SQL
- 同一データベース内の 1 つのキューから他のキューへの伝播管理 : PL/SQL
- 1 つのキューから他のデータベース内の他のキューへの伝播管理 : PL/SQL
- 伝播スケジュールの解除 : PL/SQL

■ AQ オブジェクトの削除

■ ロールおよび権限の取消し

■ AQ による XA の使用

■ AQ およびメモリーの使用

- メッセージのエンキュー（各コール後のメモリー解放）: OCI

- メッセージのエンキュー（メモリーの再利用）: OCI
- メッセージのデキュー（各コール後のメモリー解放）: OCI
- メッセージのデキュー（メモリーの再利用）: OCI

キュー表およびキューの作成

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
CONNECT system/manager;
DROP USER aqadm CASCADE;
GRANT CONNECT, RESOURCE TO aqadm;
CREATE USER aqadm IDENTIFIED BY aqadm;
GRANT EXECUTE ON DBMS_AQADM TO aqadm;
GRANT Aq_administrator_role TO aqadm;
DROP USER aq CASCADE;
CREATE USER aq IDENTIFIED BY aq;
GRANT CONNECT, RESOURCE TO aq;
GRANT EXECUTE ON dbms_aq TO aq;
```

オブジェクト型のキュー表およびキューの作成

```
/* Creating a message type: */
CREATE type aq.Message_typ as object (
subject     VARCHAR2(30),
text        VARCHAR2(80));

/* Creating a object type queue table and queue: */
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
queue_table        => 'aq.objmsgs80_qtab',
queue_payload_type => 'aq.Message_typ');

EXECUTE DBMS_AQADM.CREATE_QUEUE (
queue_name         => 'msg_queue',
queue_table        => 'aq.objmsgs80_qtab');

EXECUTE DBMS_AQADM.START_QUEUE (
queue_name         => 'msg_queue');
```

RAW 型のキュー表およびキューの作成

```
/* Creating a RAW type queue table and queue: */
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
queue_table          => 'aq.RawMsgs_qtab',
queue_payload_type   => 'RAW');

EXECUTE DBMS_AQADM.CREATE_QUEUE (
queue_name          => 'raw_msg_queue',
queue_table         => 'aq.RawMsgs_qtab');
```

```
EXECUTE DBMS_AQADM.START_QUEUE (
queue_name          => 'raw_msg_queue');
```

優先順位指定メッセージのキュー表およびキューの作成

```
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
queue_table        => 'aq.priority_msg',
sort_list          => 'PRIORITY,ENQ_TIME',
queue_payload_type => 'aq.Message_typ');

EXECUTE DBMS_AQADM.CREATE_QUEUE (
queue_name         => 'priority_msg_queue',
queue_table        => 'aq.priority_msg');

EXECUTE DBMS_AQADM.START_QUEUE (
queue_name         => 'priority_msg_queue');
```

マルチ・コンシューマ・キュー表およびキューの作成

```
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
queue_table        => 'aq.MultiConsumerMsgs_qtab',
multiple_consumers => TRUE,
queue_payload_type => 'aq.Message_typ');

EXECUTE DBMS_AQADM.CREATE_QUEUE (
queue_name         => 'msg_queue_multiple',
queue_table        => 'aq.MultiConsumerMsgs_qtab');

EXECUTE DBMS_AQADM.START_QUEUE (
queue_name         => 'msg_queue_multiple');
```

伝播のデモ用のキューの作成

```
EXECUTE DBMS_AQADM.CREATE_QUEUE (
queue_name        => 'another_msg_queue',
queue_table       => 'aq.MultiConsumerMsgs_qtab');

EXECUTE DBMS_AQADM.START_QUEUE (
queue_name         => 'another_msg_queue');
```

Java AQ の例の設定

```
CONNECT system/manager

DROP USER aqjava CASCADE;
GRANT CONNECT, RESOURCE, AQ_ADMINISTRATOR_ROLE TO aqjava IDENTIFIED BY aqjava;
GRANT EXECUTE ON DBMS_AQADM TO aqjava;
GRANT EXECUTE ON DBMS_AQ TO aqjava;
CONNECT aqjava/aqjava

/* Set up main class from which we will call subsequent examples and handle
   exceptions: */
import java.sql.*;
import oracle.AQ.*;

public class test_aqjava
{
   public static void main(String args[])
   {
      AQSession  aq_sess = null;
      try
      {
         aq_sess = createSession(args);

        /* now run the test: */
        runTest(aq_sess);
      }
      catch (Exception ex)
      {
         System.out.println("Exception-1: " + ex);
         ex.printStackTrace();
      }
   }
}
```

Java AQ セッションの作成

```
/* Creating an Java AQ Session for the 'aqjava' user as shown in the
   AQDriverManager section above: */
 public static AQSession createSession(String args[])
   {
      Connection db_conn;
      AQSession  aq_sess = null;

      try
      {

         Class.forName("oracle.jdbc.driver.OracleDriver");
         /* your actual hostname, port number, and SID will
         vary from what follows. Here we use 'dlsun736,' '5521,'
         and 'test,' respectively: */

         db_conn =
                  DriverManager.getConnection(
                  "jdbc:oracle:thin:@dlsun736:5521:test",
                  "aqjava", "aqjava");

         System.out.println("JDBC Connection opened ");
         db_conn.setAutoCommit(false);

         /* Load the Oracle8i AQ driver: */
         Class.forName("oracle.AQ.AQOracleDriver");

         /* Creating an AQ Session: */
         aq_sess = AQDriverManager.createAQSession(db_conn);
         System.out.println("Successfully created AQSession ");
      }
      catch (Exception ex)
      {
         System.out.println("Exception: " + ex);
         ex.printStackTrace();
      }
      return aq_sess;
   }
```

キュー表およびキューの作成 : Java

```
public static void runTest(AQSession aq_sess) throws AQException
{
    AQQueueTableProperty     qtable_prop;
    AQQueueProperty          queue_prop;
    AQQueueTable             q_table;
    AQQueue                  queue;

    /* Creating a AQQueueTableProperty object (payload type - RAW): */
    qtable_prop = new AQQueueTableProperty("RAW");

    /* Creating a queue table called aq_table1 in aqjava schema: */
    q_table = aq_sess.createQueueTable ("aqjava", "aq_table1", qtable_prop);
    System.out.println("Successfully created aq_table1 in aqjava schema");

    /* Creating a new AQQueueProperty object */
    queue_prop = new AQQueueProperty();

    /* Creating a queue called aq_queue1 in aq_table1: */
    queue = aq_sess.createQueue (q_table, "aq_queue1", queue_prop);
    System.out.println("Successfully created aq_queue1 in aq_table1");
}

/* Get a handle to an existing queue table and queue: */
public static void runTest(AQSession aq_sess) throws AQException
{
    AQQueueTable            q_table;
    AQQueue                 queue;

    /* Get a handle to queue table - aq_table1 in aqjava schema: */
    q_table = aq_sess.getQueueTable ("aqjava", "aq_table1");
    System.out.println("Successful getQueueTable");

    /* Get a handle to a queue - aq_queue1 in aqjava schema: */
    queue = aq_sess.getQueue ("aqjava", "aq_queue1");
    System.out.println("Successful getQueue");
}
```

キューの作成およびエンキュー / デキューの開始 : Java

```
{
     AQQueueTableProperty    qtable_prop;
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;

     /* Creating a AQQueueTable property object (payload type - RAW): */
     qtable_prop = new AQQueueTableProperty("RAW");
    qtable_prop.setCompatible("8.1");

     /* Creating a queue table called aq_table3 in aqjava schema: */
     q_table = aq_sess.createQueueTable ("aqjava", "aq_table3", qtable_prop);
     System.out.println("Successful createQueueTable");

     /* Creating a new AQQueueProperty object: */
     queue_prop = new AQQueueProperty();

     /* Creating a queue called aq_queue3 in aq_table3: */
     queue = aq_sess.createQueue (q_table, "aq_queue3", queue_prop);
     System.out.println("Successful createQueue");

     /* Enable enqueue/dequeue on this queue: */
     queue.start();
     System.out.println("Successful start queue");

     /* Grant enqueue_any privilege on this queue to user scott: */
     queue.grantQueuePrivilege("ENQUEUE", "scott");
     System.out.println("Successful grantQueuePrivilege");
}
```

マルチ・コンシューマ・キューの作成およびサブスクライバの追加 : Java

```
public static void runTest(AQSession aq_sess) throws AQException
{
     AQQueueTableProperty    qtable_prop;
     AQQueueProperty         queue_prop;
     AQQueueTable            q_table;
     AQQueue                 queue;
     AQAgent                  subs1, subs2;

     /* Creating a AQQueueTable property object (payload type - RAW): */
     qtable_prop = new AQQueueTableProperty("RAW");
     System.out.println("Successful setCompatible");

     /* Set multiconsumer flag to true: */
     qtable_prop.setMultiConsumer(true);

     /* Creating a queue table called aq_table4 in aqjava schema: */
     q_table = aq_sess.createQueueTable ("aqjava", "aq_table4", qtable_prop);
     System.out.println("Successful createQueueTable");

     /* Creating a new AQQueueProperty object: */
     queue_prop = new AQQueueProperty();
     /* Creating a queue called aq_queue4 in aq_table4 */
     queue = aq_sess.createQueue (q_table, "aq_queue4", queue_prop);
     System.out.println("Successful createQueue");

     /* Enable enqueue/dequeue on this queue: */
     queue.start();
     System.out.println("Successful start queue");

     /* Add subscribers to this queue: */
     subs1 = new AQAgent("GREEN", null, 0);
     subs2 = new AQAgent("BLUE", null, 0);

     queue.addSubscriber(subs1, null); /* no rule   */
     System.out.println("Successful addSubscriber 1");

     queue.addSubscriber(subs2, "priority < 2"); /* with rule */
     System.out.println("Successful addSubscriber 2");
}
```

メッセージのエンキューおよびデキュー

オブジェクト型メッセージのエンキューおよびデキュー : PL/SQL

他のパラメータを使用せずに 1 つのメッセージをエンキューするには、キュー名およびペイロードを指定します。

```
/* Enqueue to msg_queue: */
DECLARE
   enqueue_options     dbms_aq.enqueue_options_t;
   message_properties  dbms_aq.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;

BEGIN
   message := message_typ('NORMAL MESSAGE',
   'enqueued to msg_queue first.');

   dbms_aq.enqueue(queue_name => 'msg_queue',
         enqueue_options      => enqueue_options,
         message_properties   => message_properties,
         payload              => message,
         msgid                => message_handle);

   COMMIT;

/* Dequeue from msg_queue: */
DECLARE
   dequeue_options     dbms_aq.dequeue_options_t;
   message_properties  dbms_aq.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;

BEGIN
   DBMS_AQ.DEQUEUE(queue_name => 'msg_queue',
           dequeue_options    => dequeue_options,
           message_properties => message_properties,
           payload            => message,
           msgid              => message_handle);

   DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
                                     ' ... ' || message.text );
   COMMIT;
END;
```

オブジェクト型メッセージのエンキューおよびデキュー：Pro*C/C++

注意： 次のような設定を実行しないと機能しない例もあります。

```
$ cat >> message.typ
case=lower
type aq.message_typ
$
$ ott userid=aq/aq intyp=message.typ outtyp=message_o.typ \  code=c
hfile=demo.h
$
$ proc intyp=message_o.typ iname=<program name> \
config=<config file> SQLCHECK=SEMANTICS userid=aq/aq
```

```
#include  <stdio.h>
#include  <string.h>
#include  <sqlca.h>
#include  <sql2oci.h>
/* The header file generated by processing
object type 'aq.Message_typ': */
#include  "pceg.h"

void sql_error(msg)
char *msg;
{
EXEC SQL WHENEVER SQLERROR CONTINUE;
printf("%s\n", msg);
printf("\n% .800s \n", sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}

main()
{
Message_typ     *message = (Message_typ*)0;  /* payload */
message_type_ind *imsg;              /*payload indicator*/
char             user[60]="aq/AQ";  /* user logon password */
char             subject[30];  /* components of the */
char             txt[80];      /* payload type */

/* ENQUEUE and DEQUEUE to an OBJECT QUEUE */

/*  Connect to database: */
EXEC SQL CONNECT :user;

/* On an oracle error print the error number :*/
```

```
EXEC SQL WHENEVER SQLERROR DO sql_error("Oracle Error :");

/* Allocate memory for the host variable from the object cache : */
EXEC SQL ALLOCATE :message;

/* ENQUEUE */

strcpy(subject, "NORMAL ENQUEUE");
strcpy(txt, "The Enqueue was done through PLSQL embedded in PROC");

/* Initialize the components of message : */
EXEC SQL OBJECT SET subject, text OF  :message TO :subject, :txt;

/* Embedded PLSQL call to the AQ enqueue procedure : */
EXEC SQL EXECUTE
DECLARE
message_properties   dbms_aq.message_properties_t;
enqueue_options      dbms_aq.enqueue_options_t;
msgid                RAW(16);
BEGIN
/* Bind the host variable 'message' to the payload: */
dbms_aq.enqueue(queue_name => 'msg_queue',
message_properties => message_properties,
enqueue_options => enqueue_options,
payload => :message:imsg,    /* indicator has to be specified */
msgid => msgid);
END;
END-EXEC;
/* Commit work */
EXEC SQL COMMIT;

printf("Enqueued Message \n");
printf("Subject  :%s\n",subject);
printf("Text     :%s\n",txt);

/* Dequeue */

/* Embedded PLSQL call to the AQ dequeue procedure : */
EXEC SQL EXECUTE
DECLARE
message_properties  dbms_aq.message_properties_t;
dequeue_options     dbms_aq.dequeue_options_t;
msgid               RAW(16);
BEGIN
/* Return the payload into the host variable 'message':  */
dbms_aq.dequeue(queue_name => 'msg_queue',
message_properties => message_properties,
```

```
dequeue_options => dequeue_options,
payload => :message,
msgid => msgid);
END;
END-EXEC;
 /* Commit work :*/
EXEC SQL COMMIT;

/* Extract the components of message: */
EXEC SQL OBJECT GET SUBJECT,TEXT FROM :message INTO :subject,:txt;

printf("Dequeued Message \n");
printf("Subject  :%s\n",subject);
printf("Text     :%s\n",txt);
}
```

オブジェクト型メッセージのエンキューおよびデキュー : OCI

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>

struct message
{
  OCIString   *subject;
  OCIString   *data;
};
typedef struct message message;

struct null_message
{
  OCIInd    null_adt;
  OCIInd    null_subject;
  OCIInd    null_data;
};
typedef struct null_message null_message;

int main()
{
  OCIEnv       *envhp;
  OCIServer    *srvhp;
  OCIError     *errhp;
  OCISvcCtx    *svchp;
  dvoid        *tmp;
  OCIType      *mesg_tdo = (OCIType *) 0;
```

```
message      msg;
null_message nmsg;
message      *mesg     = &msg;
null_message *nmesg    = &nmsg;
message      *deqmesg  = (message *)0;
null_message *ndeqmesg = (null_message *)0;

OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
              (dvoid * (*)()) 0,  (void (*)()) 0 );

OCIHandleAlloc((dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
               52, (dvoid **) &tmp);

OCIEnvInit(&envhp, (ub4) OCI_DEFAULT, 21, (dvoid **) &tmp  );

OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
               52, (dvoid **) &tmp);
OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
               52, (dvoid **) &tmp);

OCIServerAttach(srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
               52, (dvoid **) &tmp);

OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
    (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

OCILogon(envhp, errhp, &svchp, "AQ", strlen("AQ"), "AQ", strlen("AQ"), 0, 0);

/* Obtain TDO of message_typ */
OCITypeByName(envhp, errhp, svchp, (CONST text *)"AQ", strlen("AQ"),
              (CONST text *)"MESSAGE_TYP", strlen("MESSAGE_TYP"),
              (text *)0, 0, OCI_DURATION_SESSION, OCI_TYPEGET_ALL, &mesg_tdo);

/* Prepare the message payload */
mesg->subject = (OCIString *)0;
mesg->data = (OCIString *)0;
OCIStringAssignText(envhp, errhp,
                    (CONST text *)"NORMAL MESSAGE", strlen("NORMAL MESSAGE"),
                     &mesg->subject);


OCIStringAssignText(envhp, errhp,
                    (CONST text *)"OCI ENQUEUE", strlen("OCI ENQUEUE"),
                    &mesg->data);
nmesg->null_adt = nmesg->null_subject = nmesg->null_data = OCI_IND_NOTNULL;
```

```
  /* Enqueue into the msg_queue */
  OCIAQEnq(svchp, errhp, (CONST text *)"msg_queue", 0, 0,
           mesg_tdo, (dvoid **)&mesg, (dvoid **)&nmesg, 0, 0);
  OCITransCommit(svchp, errhp, (ub4) 0);

  /* Dequeue from the msg_queue */
  OCIAQDeq(svchp, errhp, (CONST text *)"msg_queue", 0, 0,
           mesg_tdo, (dvoid **)&deqmesg, (dvoid **)&ndeqmesg, 0, 0);
  printf("Subject: %s\n", OCIStringPtr(envhp, deqmesg->subject));
  printf("Text: %s\n", OCIStringPtr(envhp, deqmesg->data));
  OCITransCommit(svchp, errhp, (ub4) 0);
}
```

オブジェクト型メッセージ（CustomDatum インタフェース）のエンキューおよびデキュー : Java

オブジェクト型のメッセージをエンキューおよびデキューするには、次の手順に従います。

a. キュー・ペイロードに対する SQL 型を作成します。

```
connect aquser/aquser
create type ADDRESS as object (street VARCHAR (30), city VARCHAR(30));
create type PERSON as object (name VARCHAR (30), home ADDRESS);
```

b. PERSON ユーザー定義型に対応した、CustomDatum インタフェースを実装する Java クラスを（JPublisher ツールを使用して）生成します。

```
jpub -user=aquser/aquser -sql=ADDRESS,PERSON -case=mixed -usertypes=oracle
-methods=false  -compatible=CustomDatum
```

これによって、PERSON ユーザー定義型および ADDRESS ユーザー定義型に対応した PERSON.java および ADDRESS.java という 2 つのクラスが作成されます。

c. ユーザー定義型ペイロードを使用して、キュー表およびキューを作成します。

d. オブジェクト型ペイロードを含むメッセージをエンキューおよびデキューします。

```
public static void AQObjectPayloadTest(AQSession aq_sess)
       throws AQException, SQLException, ClassNotFoundException
  {
    Connection          db_conn   = null;
    AQQueue             queue     = null;
    AQMessage           message   = null;
    AQObjectPayload     payload   = null;
    AQEnqueueOption     eq_option = null;
    AQDequeueOption     dq_option = null;
```

```
  PERSON           pers = null;
  PERSON           pers2= null;
  ADDRESS          addr = null;

  db_conn = ((AQOracleSession)aq_sess).getDBConnection();


  queue = aq_sess.getQueue("aquser", "test_queue2");


  /* Enable enqueue/dequeue on this queue */
  queue.start();

  /* Enqueue a message in test_queue2 */
  message = queue.createMessage();

  pers = new PERSON();
  pers.setName("John");
  addr = new ADDRESS();
  addr.setStreet("500 Easy Street");
  addr.setCity("San Francisco");
  pers.setHome(addr);

  payload = message.getObjectPayload();
  payload.setPayloadData(pers);
  eq_option = new AQEnqueueOption();

  /* Enqueue a message into test_queue2 */
  queue.enqueue(eq_option, message);

  db_conn.commit();

  /* Dequeue a message from test_queue2 */
  dq_option = new AQDequeueOption();
  message = ((AQOracleQueue)queue).dequeue(dq_option, PERSON.getFactory());

  payload = message.getObjectPayload();
  pers2 = (PERSON) payload.getPayloadData();

  System.out.println("Object data retrieved:  [PERSON]");
  System.out.println("Name:   " + pers2.getName());
  System.out.println("Address ");
  System.out.println("Street: " + pers2.getHome().getStreet());
  System.out.println("City:   " + pers2.getHome().getCity());

  db_conn.commit();
}
```

オブジェクト型メッセージ（SQLData インタフェース使用）のエンキューおよびデキュー : Java

オブジェクト型のメッセージをエンキューおよびデキューするには、次の手順に従います。

a. キュー・ペイロードに対する SQL 型を作成します。

```
connect aquser/aquser
create type EMPLOYEE as object (empname VARCHAR (50), empno INTEGER);
```

b. EMPLOYEE ユーザー定義型に対応した、SQLData インタフェースを実装する Java クラスを生成します。このクラスは、次の構文を使用して、JPublisher で生成することもできます。

```
jpub -user=aquser/aquser -sql=EMPLOYEE -case=mixed -usertypes=jdbc
-methods=false

import java.sql.*;
import oracle.jdbc2.*;

public class Employee implements SQLData
{
  private String sql_type;
  public String empName;
  public int empNo;
  public Employee()
  {}
  public Employee (String sql_type, String empName, int empNo)
  {
    this.sql_type = sql_type;
    this.empName = empName;
    this.empNo = empNo;
  }

  ////// implements SQLData //////
  public String getSQLTypeName() throws SQLException
  { return sql_type;
  }
  public void readSQL(SQLInput stream, String typeName)
    throws SQLException
  {
    sql_type = typeName;
    empName = stream.readString();
    empNo = stream.readInt();
  }

  public void writeSQL(SQLOutput stream)
    throws SQLException
```

```
  {
    stream.writeString(empName);
    stream.writeInt(empNo);
  }

  public String toString()
  {
String ret_str = "";
    ret_str += "[Employee]\n";
    ret_str += "Name: " + empName + "\n";
    ret_str += "Number: " + empNo + "\n";

    return ret_str;
  }
}
```

c. ユーザー定義型ペイロードを使用して、キュー表およびキューを作成します。

```
public static void createEmployeeObjQueue(AQSession aq_sess)
       throws AQException
  {
    AQQueueTableProperty qt_prop   = null;
    AQQueueProperty      q_prop    = null;
    AQQueueTable         q_table   = null;
    AQQueue              queue     = null;

    /* Message payload type is aquser.EMPLOYEE */
    qt_prop = new AQQueueTableProperty("AQUSER.EMPLOYEE");
    qt_prop.setComment("queue-table1");

    /* Creating aQTable1 */
    System.out.println("\nCreate QueueTable: [aqtable1]");
    q_table = aq_sess.createQueueTable("aquser", "aqtable1", qt_prop);

    /* Create test_queue1 */
    q_prop = new AQQueueProperty();
    queue = q_table.createQueue("test_queue1", q_prop);

    /* Enable enqueue/dequeue on this queue */
    queue.start();
  }
```

d. オブジェクト型ペイロードを含むメッセージをエンキューおよびデキューします。

```
public static void AQObjectPayloadTest2(AQSession aq_sess)
       throws AQException, SQLException, ClassNotFoundException
  {
    Connection           db_conn   = null;
```

```
AQQueue              queue     = null;
AQMessage            message   = null;
AQObjectPayload      payload   = null;
AQEnqueueOption      eq_option = null;
AQDequeueOption      dq_option = null;
Employee             emp  = null;
Employee             emp2 = null;
Hashtable            map;

db_conn = ((AQOracleSession)aq_sess).getDBConnection();

/* Get the Queue object */
queue = aq_sess.getQueue("aquser", "test_queue1");

/* Register Employee class (corresponding to EMPLOYEE Adt)
 * in the connection type map
 */
try
{
  map  = (java.util.Hashtable)(((OracleConnection)db_conn).getTypeMap());
  map.put("AQUSER.EMPLOYEE", Class.forName("Employee"));
}
catch(Exception ex)
{
  System.out.println("Error registering type: " + ex);
}

/* Enqueue a message in test_queue1 */
message = queue.createMessage();
emp = new Employee("AQUSER.EMPLOYEE", "Mark", 1007);

/* Set the object payload */
payload = message.getObjectPayload();
payload.setPayloadData(emp);

/* Enqueue a message into test_queue1*/
eq_option = new AQEnqueueOption();
queue.enqueue(eq_option, message);
db_conn.commit();


/* Dequeue a message from test_queue1 */
dq_option = new AQDequeueOption();

message = queue.dequeue(dq_option, Class.forName("Employee"));
payload = message.getObjectPayload();
emp2 = (Employee) payload.getPayloadData();
```

```
        System.out.println("\nObject data retrieved:  [EMPLOYEE]");
        System.out.println("Name  : " + emp2.empName);
        System.out.println("EmpId : " + emp2.empNo);

        db_conn.commit();
      }
```

RAW 型メッセージのエンキューおよびデキュー : PL/SQL

```
DECLARE
   enqueue_options     dbms_aq.enqueue_options_t;
   message_properties  dbms_aq.message_properties_t;
   message_handle      RAW(16);
   message             RAW(4096);

BEGIN
   message :=  HEXTORAW(RPAD('FF',4095,'FF'));
   DBMS_AQ.ENQUEUE(queue_name => 'raw_msg_queue',
           enqueue_options    => enqueue_options,
           message_properties => message_properties,
                     payload  => message,
                      msgid   => message_handle);

   COMMIT;
END;

/* Dequeue from raw_msg_queue: */
/* Dequeue from raw_msg_queue: */
DECLARE
   dequeue_options     DBMS_AQ.dequeue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   message_handle      RAW(16);
   message             RAW(4096);

BEGIN
   DBMS_AQ.DEQUEUE(queue_name => 'raw_msg_queue',
           dequeue_options    => dequeue_options,
           message_properties => message_properties,
           payload            => message,
           msgid              => message_handle);

   COMMIT;
END;
```

RAW 型メッセージのエンキューおよびデキュー : Pro*C/C++

注意 : 次のような設定を実行しないと機能しない例もあります。

```
$ cat >> message.typ
case=lower
type aq.message_typ
$
$ ott userid=aq/aq intyp=message.typ outtyp=message_o.typ \  code=c
hfile=demo.h
$
$ proc intyp=message_o.typ iname=<program name> \
config=<config file> SQLCHECK=SEMANTICS userid=aq/aq
```

```
#include  <stdio.h>
#include  <string.h>
#include  <sqlca.h>
#include  <sql2oci.h>

void sql_error(msg)
char *msg;
{
EXEC SQL WHENEVER SQLERROR CONTINUE;
printf("%s\n", msg);
printf("\n% .800s \n", sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}

main()
{
LNOCIEnv         *oeh;   /* OCI Env handle */
LNOCIError       *err;   /* OCI Err handle */
LNOCIRaw         *message= (OCIRaw*)0;   /* payload */
ub1            message_txt[100];   /* data for payload */
char           user[60]="aq/AQ";   /* user logon password */
int            status;   /* returns status of the OCI call */

/* Enqueue and dequeue to a RAW queue */

/* Connect to database: */
EXEC SQL CONNECT :user;

/* On an oracle error print the error number: */
EXEC SQL WHENEVER SQLERROR DO sql_error("Oracle Error :");
```

```
/* Get the OCI Env handle: */
if (SQLEnvGet(SQL_SINGLE_RCTX, &oeh) != OCI_SUCCESS)
{
printf(" error in SQLEnvGet \n");
exit(1);
}
/* Get the OCI Error handle: */
if (status = OCIHandleAlloc((dvoid *)oeh, (dvoid **)&err,
(ub4)OCI_HTYPE_ERROR, (ub4)0, (dvoid **)0))
{
printf(" error in OCIHandleAlloc %d \n", status);
exit(1);
}

/* Enqueue */
/* The bytes to be put into the raw payload:*/
strcpy(message_txt, "Enqueue to a Raw payload queue ");

/* Assign bytes to the OCIRaw pointer :
Memory needs to be allocated explicitly to OCIRaw*: */
if (status=OCIRawAssignBytes(oeh, err, message_txt, 100,
 &message))
{
printf(" error in  OCIRawAssignBytes  %d \n", status);
exit(1);
}

/*  Embedded PLSQL call to the AQ enqueue procedure : */
EXEC SQL EXECUTE
DECLARE
message_properties   dbms_aq.message_properties_t;
enqueue_options      dbms_aq.enqueue_options_t;
msgid                RAW(16);
BEGIN
/* Bind the host variable message to the raw payload: */
dbms_aq.enqueue(queue_name => 'raw_msg_queue',
message_properties => message_properties,
enqueue_options => enqueue_options,
payload => :message,
msgid => msgid);
END;
END-EXEC;
/* Commit work: */
EXEC SQL COMMIT;

/* Dequeue */
/*  Embedded PLSQL call to the AQ dequeue procedure :*/
```

```
EXEC SQL EXECUTE
DECLARE
message_properties  dbms_aq.message_properties_t;
dequeue_options     dbms_aq.dequeue_options_t;
msgid               RAW(16);
BEGIN
/* Return the raw payload into the host variable 'message':*/
dbms_aq.dequeue(queue_name => 'raw_msg_queue',
message_properties => message_properties,
dequeue_options => dequeue_options,
payload => :message,
msgid => msgid);
END;
END-EXEC;
/* Commit work: */
EXEC SQL COMMIT;
}
```

RAW 型メッセージのエンキューおよびデキュー : OCI

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>

int main()
{
  OCIEnv       *envhp;
  OCIServer    *srvhp;
  OCIError     *errhp;
  OCISvcCtx    *svchp;
  dvoid        *tmp;
  OCIType      *mesg_tdo = (OCIType *) 0;
  char         msg_text[100];
  OCIRaw       *mesg = (OCIRaw *)0;
  OCIRaw       *deqmesg = (OCIRaw *)0;
  OCIInd       ind = 0;
  dvoid        *indptr = (dvoid *)&ind;
  int          i;

  OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
                (dvoid * (*)()) 0,  (void (*)()) 0 );

  OCIHandleAlloc((dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
                 52, (dvoid **) &tmp);
```

```
  OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 21, (dvoid **) &tmp  );

  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
                 52, (dvoid **) &tmp);
  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
                 52, (dvoid **) &tmp);

  OCIServerAttach(srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
                 52, (dvoid **) &tmp);

  OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
             (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

  OCILogon(envhp, errhp, &svchp, "AQ", strlen("AQ"), "AQ", strlen("AQ"), 0, 0);

  /* Obtain the TDO of the RAW data type */
  OCITypeByName(envhp, errhp, svchp, (CONST text *)"AQADM", strlen("AQADM"),
                (CONST text *)"RAW", strlen("RAW"),
                (text *)0, 0, OCI_DURATION_SESSION, OCI_TYPEGET_ALL, &mesg_tdo);

  /* Prepare the message payload */
  strcpy(msg_text, "Enqueue to a RAW queue");
  OCIRawAssignBytes(envhp, errhp, msg_text, strlen(msg_text), &mesg);

  /* Enqueue the message into raw_msg_queue */
  OCIAQEnq(svchp, errhp, (CONST text *)"raw_msg_queue", 0, 0,
          mesg_tdo, (dvoid **)&mesg, (dvoid **)&indptr, 0, 0);
  OCITransCommit(svchp, errhp, (ub4) 0);

  /* Dequeue the same message into C variable deqmesg */
  OCIAQDeq(svchp, errhp, (CONST text *)"raw_msg_queue", 0, 0,
           mesg_tdo, (dvoid **)&deqmesg, (dvoid **)&indptr, 0, 0);
  for (i = 0; i < OCIRawSize(envhp, deqmesg); i++)
    printf("%c", *(OCIRawPtr(envhp, deqmesg) + i));
  OCITransCommit(svchp, errhp, (ub4) 0);
}
```

RAW 型メッセージのエンキュー : Java

```
public static void runTest(AQSession aq_sess) throws AQException
{
    AQQueueTable            q_table;
    AQQueue                 queue;
    AQMessage               message;
```

```
        AQRawPayload            raw_payload;
        AQEnqueueOption         enq_option;
        String                  test_data = "new message";
        byte[]                  b_array;
        Connection              db_conn;

        db_conn = ((AQOracleSession)aq_sess).getDBConnection();

        /* Get a handle to queue table - aq_table4 in aqjava schema: */
        q_table = aq_sess.getQueueTable ("aqjava", "aq_table4");
        System.out.println("Successful getQueueTable");

        /* Get a handle to a queue - aq_queue4 in aquser schema: */
        queue = aq_sess.getQueue ("aqjava", "aq_queue4");
        System.out.println("Successful getQueue");

        /* Creating a message to contain raw payload: */
        message = queue.createMessage();

        /* Get handle to the AQRawPayload object and populate it with raw data: */
        b_array = test_data.getBytes();

        raw_payload = message.getRawPayload();

        raw_payload.setStream(b_array, b_array.length);

        /* Creating a AQEnqueueOption object with default options: */
        enq_option = new AQEnqueueOption();
        /* Enqueue the message: */
        queue.enqueue(enq_option, message);

        db_conn.commit();
    }
```

メッセージのデキュー : Java

```
   public static void runTest(AQSession aq_sess) throws AQException
{
    AQQueueTable             q_table;
    AQQueue                  queue;
    AQMessage                message;
    AQRawPayload             raw_payload;
    AQEnqueueOption          enq_option;
    String                   test_data = "new message";
    AQDequeueOption          deq_option;
    byte[]                   b_array;
    Connection               db_conn;

    db_conn = ((AQOracleSession)aq_sess).getDBConnection();

    /* Get a handle to queue table - aq_table4 in aqjava schema: */
    q_table = aq_sess.getQueueTable ("aqjava", "aq_table4");
    System.out.println("Successful getQueueTable");

    /* Get a handle to a queue - aq_queue4 in aquser schema: */
    queue = aq_sess.getQueue ("aqjava", "aq_queue4");
    System.out.println("Successful getQueue");

    /* Creating a message to contain raw payload: */
    message = queue.createMessage();

    /* Get handle to the AQRawPayload object and populate it with raw data: */
    b_array = test_data.getBytes();

    raw_payload = message.getRawPayload();

    raw_payload.setStream(b_array, b_array.length);

    /* Creating a AQEnqueueOption object with default options: */
    enq_option = new AQEnqueueOption();

    /* Enqueue the message: */
    queue.enqueue(enq_option, message);
    System.out.println("Successful enqueue");

    db_conn.commit();

    /* Creating a AQDequeueOption object with default options: */
    deq_option = new AQDequeueOption();

    /* Dequeue a message: */
```

```
      message = queue.dequeue(deq_option);
      System.out.println("Successful dequeue");

      /* Retrieve raw data from the message: */
      raw_payload = message.getRawPayload();

      b_array = raw_payload.getBytes();

      db_conn.commit();
}
```

ブラウズ・モードでのメッセージのデキュー : Java

```
   public static void runTest(AQSession aq_sess) throws AQException
{
      AQQueueTable            q_table;
      AQQueueTable            q_table;
      AQQueue                 queue;
      AQMessage               message;
      AQRawPayload            raw_payload;
      AQEnqueueOption         enq_option;
      String                  test_data = "new message";
      AQDequeueOption         deq_option;
      byte[]                   b_array;
      Connection              db_conn;

      db_conn = ((AQOracleSession)aq_sess).getDBConnection();

      /* Get a handle to queue table - aq_table4 in aqjava schema: */
      q_table = aq_sess.getQueueTable ("aqjava", "aq_table4");
      System.out.println("Successful getQueueTable");

      /* Get a handle to a queue - aq_queue4 in aquser schema: */
      queue = aq_sess.getQueue ("aqjava", "aq_queue4");
      System.out.println("Successful getQueue");

      /* Creating a message to contain raw payload: */
      message = queue.createMessage();

      /* Get handle to the AQRawPayload object and populate it with raw data: */
      b_array = test_data.getBytes();

      raw_payload = message.getRawPayload();

      raw_payload.setStream(b_array, b_array.length);
```

```
        /* Creating a AQEnqueueOption object with default options: */
        enq_option = new AQEnqueueOption();

        /* Enqueue the message: */
        queue.enqueue(enq_option, message);
        System.out.println("Successful enqueue");

        db_conn.commit();

        /* Creating a AQDequeueOption object with default options: */
        deq_option = new AQDequeueOption();

        /* Set dequeue mode to BROWSE: */
        deq_option.setDequeueMode(AQDequeueOption.DEQUEUE_BROWSE);

        /* Set wait time to 10 seconds: */
        deq_option.setWaitTime(10);

        /* Dequeue a message: */
        message = queue.dequeue(deq_option);

        /* Retrieve raw data from the message: */
        raw_payload = message.getRawPayload();
        b_array = raw_payload.getBytes();

        String ret_value = new String(b_array);
        System.out.println("Dequeued message: " + ret_value);

        db_conn.commit();
}
```

優先順位によるメッセージのエンキューおよびデキュー : PL/SQL

2 つのメッセージが同じ優先順位でエンキューされると、先にエンキューされた方が先にデキューされます。ただし、2 つのメッセージの優先順位が異なる場合は、値の小さい（優先順位の高い）メッセージが先にデキューされます。

```
/* Enqueue two messages with priority 30 and 5: */
DECLARE
   enqueue_options     dbms_aq.enqueue_options_t;
   message_properties  dbms_aq.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;

BEGIN
```

```
   message := message_typ('PRIORITY MESSAGE',
   'enqued at priority 30.');

   message_properties.priority := 30;

   DBMS_AQ.ENQUEUE(queue_name => 'priority_msg_queue',
           enqueue_options    => enqueue_options,
           message_properties => message_properties,
           payload            => message,
           msgid              => message_handle);

   message := message_typ('PRIORITY MESSAGE',
   'Enqueued at priority 5.');

   message_properties.priority := 5;

   DBMS_AQ.ENQUEUE(queue_name => 'priority_msg_queue',
           enqueue_options    => enqueue_options,
           message_properties => message_properties,
           payload            => message,
           msgid              => message_handle);
END;

/* Dequeue from priority queue: */
DECLARE
   dequeue_options     DBMS_AQ.dequeue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;

BEGIN
   DBMS_AQ.DEQUEUE(queue_name  => 'priority_msg_queue',
         dequeue_options       => dequeue_options,
         message_properties    => message_properties,
         payload               => message,
         msgid                 => message_handle);

   DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
   ' ... ' || message.text );

   COMMIT;

   DBMS_AQ.DEQUEUE(queue_name => 'priority_msg_queue',
         dequeue_options      => dequeue_options,
         message_properties   => message_properties,
         payload              => message,
         msgid                => message_handle);
```

```
   DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
   ' ... ' || message.text );
   COMMIT;
END;

/* On return, the second message with priority set to 5 will be retrieved before the
message with priority set to 30 since priority takes precedence over enqueue time.
*/
```

優先順位によるメッセージのエンキュー : Java

```
public static void runTest(AQSession aq_sess) throws AQException
{
     AQQueueTable             q_table;
     AQQueue                  queue;
     AQMessage                message;
     AQMessageProperty        m_property;
     AQRawPayload             raw_payload;
     AQEnqueueOption          enq_option;
     String                   test_data;
     byte[]                   b_array;
     Connection               db_conn;

     db_conn = ((AQOracleSession)aq_sess).getDBConnection();

     /* Get a handle to queue table - aq_table4 in aqjava schema: */
     qtable = aq_sess.getQueueTable ("aqjava", "aq_table4");
     System.out.println("Successful getQueueTable");

     /* Get a handle to a queue - aq_queue4 in aqjava schema: */
     queue = aq_sess.getQueue ("aqjava", "aq_queue4");
     System.out.println("Successful getQueue");

     /* Enqueue 5 messages with priorities with different priorities: */
     for (int i = 0; i < 5; i++ )
       {
          /* Creating a message to contain raw payload: */
          message = queue.createMessage();

          test_data = "Small_message_" + (i+1);       /* some test data */

          /* Get a handle to the AQRawPayload object and
             populate it with raw data: */
          b_array = test_data.getBytes();
```

```
            raw_payload = message.getRawPayload();

            raw_payload.setStream(b_array, b_array.length);

            /* Set message priority: */
           m_property = message.getMessageProperty();

           if( i < 2)
              m_property.setPriority(2);
            else
               m_property.setPriority(3);

            /* Creating a AQEnqueueOption object with default options: */
            enq_option = new AQEnqueueOption();

            /* Enqueue the message: */
           queue.enqueue(enq_option, message);
           System.out.println("Successful enqueue");
      }

      db_conn.commit();
}
```

プレビュー後のメッセージのデキュー : PL/SQL

アプリケーションでは、ブラウズ・モードまたはロック・モードで、メッセージを削除せずにプレビューできます。その後で、対象メッセージをキューから削除できます。

```
/* Enqueue 6 messages to msg_queue
— GREEN, GREEN, YELLOW, VIOLET, BLUE, RED */

DECLARE
   enqueue_options     DBMS_AQ.enqueue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;

BEGIN
   message := message_typ('GREEN',
   'GREEN enqueued to msg_queue first.');

   DBMS_AQ.ENQUEUE(queue_name => 'msg_queue',
         enqueue_options      => enqueue_options,
         message_properties   => message_properties,
         payload              => message,
         msgid                => message_handle);
```

```
message := message_typ('GREEN',
'GREEN also enqueued to msg_queue second.');

DBMS_AQ.ENQUEUE(queue_name => 'msg_queue',
      enqueue_options      => enqueue_options,
      message_properties   => message_properties,
      payload              => message,
      msgid                => message_handle);

message := message_typ('YELLOW',
'YELLOW enqueued to msg_queue third.');

DBMS_AQ.ENQUEUE(queue_name => 'msg_queue',
      enqueue_options      => enqueue_options,
      message_properties   => message_properties,
      payload              => message,
      msgid                => message_handle);

DBMS_OUTPUT.PUT_LINE ('Message handle: ' || message_handle);

message := message_typ('VIOLET',
'VIOLET enqueued to msg_queue fourth.');

DBMS_AQ.ENQUEUE(queue_name => 'msg_queue',
      enqueue_options      => enqueue_options,
      message_properties   => message_properties,
      payload              => message,
      msgid                => message_handle);

message := message_typ('BLUE',
'BLUE enqueued to msg_queue fifth.');

DBMS_AQ.ENQUEUE(queue_name => 'msg_queue',
      enqueue_options      => enqueue_options,
      message_properties   => message_properties,
      payload              => message,
      msgid                => message_handle);

message := message_typ('RED',
'RED enqueued to msg_queue sixth.');

DBMS_AQ.ENQUEUE(queue_name => 'msg_queue',
      enqueue_options      => enqueue_options,
      message_properties   => message_properties,
      payload              => message,
      msgid                => message_handle);
```

```
   COMMIT;
END;

/* Dequeue in BROWSE mode until RED is found,
and remove RED from queue: */
DECLARE
   dequeue_options     DBMS_AQ.dequeue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;

BEGIN
   dequeue_options.dequeue_mode := DBMS_AQ.BROWSE;

   LOOP
      DBMS_AQ.DEQUEUE(queue_name          => 'msg_queue',
                      dequeue_options     => dequeue_options,
                      message_properties => message_properties,
                      payload             => message,
                      msgid               => message_handle);

      DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
                                          ' ... ' || message.text );

      EXIT WHEN message.subject = 'RED';

   END LOOP;

   dequeue_options.dequeue_mode := DBMS_AQ.REMOVE;
   dequeue_options.msgid        := message_handle;

   DBMS_AQ.DEQUEUE(queue_name => 'msg_queue',
           dequeue_options    => dequeue_options,
           message_properties => message_properties,
           payload            => message,
           msgid              => message_handle);

   DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
   ' ... ' || message.text );

   COMMIT;
END;

/* Dequeue in LOCKED mode until BLUE is found,
and remove BLUE from queue: */
DECLARE
```

```
dequeue_options     dbms_aq.dequeue_options_t;
message_properties  dbms_aq.message_properties_t;
message_handle      RAW(16);
message             aq.message_typ;

BEGIN
dequeue_options.dequeue_mode := dbms_aq.LOCKED;

      LOOP

dbms_aq.dequeue(queue_name => 'msg_queue',
                dequeue_options    => dequeue_options,
                message_properties => message_properties,
                payload            => message,
                msgid              => message_handle);

dbms_output.put_line ('Message: ' || message.subject ||
          ' ... ' || message.text );

EXIT WHEN message.subject = 'BLUE';
      END LOOP;

dequeue_options.dequeue_mode := dbms_aq.REMOVE;
dequeue_options.msgid        := message_handle;

dbms_aq.dequeue(queue_name => 'msg_queue',
dequeue_options      => dequeue_options,
message_properties   => message_properties,
payload              => message,
msgid => message_handle);

DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
' ... ' || message.text );

      COMMIT;
END;
```

遅延および期限切れによるメッセージのエンキューおよびデキュー : PL/SQL

注意 : 期限切れは最も早いデキュー時刻から計算されています。たとえば、アプリケーション側でメッセージを今から 1 週間後以降 3 週間以内にデキューする場合には、期限切れを 2 週間に設定する必要があります。このシナリオを、次のサンプル・コードで説明します。

```
/* Enqueue message for delayed availability: */
DECLARE
enqueue_options     dbms_aq.enqueue_options_t;
message_properties  dbms_aq.message_properties_t;
message_handle      RAW(16);
message             aq.Message_typ;

BEGIN
message := Message_typ('DELAYED',
'This message is delayed one week.');
message_properties.delay := 7*24*60*60;
message_properties.expiration := 2*7*24*60*60;

dbms_aq.enqueue(queue_name => 'msg_queue',
enqueue_options      => enqueue_options,
message_properties   => message_properties,
payload            => message,
msgid              => message_handle);

      COMMIT;
END;
```

相関識別子およびメッセージ ID によるメッセージのエンキューおよびデキュー : Pro*C/C++

注意 : 次のような設定を実行しないと機能しない例もあります。

```
$ cat >> message.typ
case=lower
type aq.message_typ
$
$ ott userid=aq/aq intyp=message.typ outtyp=message_o.typ \  code=c
hfile=demo.h
$
$ proc intyp=message_o.typ iname=<program name> \
config=<config file> SQLCHECK=SEMANTICS userid=aq/aq
```

```
#include  <stdio.h>
#include  <string.h>
#include  <sqlca.h>
#include  <sql2oci.h>
/* The header file generated by processing
object type 'aq.Message_typ': */
#include  "pceg.h"

void sql_error(msg)
char *msg;
{
EXEC SQL WHENEVER SQLERROR CONTINUE;
printf("%s\n", msg);
printf("\n% .800s \n", sqlca.sqlerrm.sqlerrmc);
EXEC SQL ROLLBACK WORK RELEASE;
exit(1);
}

main()
{
LNOCIEnv            *oeh;  /* OCI Env Handle */
LNOCIError          *err;  /* OCI Error Handle */
Message_typ      *message = (Message_typ*)0; /* queue  payload */
message_type_ind *imsg;              /*payload indicator*/
LNOCIRaw            *msgid = (OCIRaw*)0;  /* message id */
ub1               msgmem[16]="";  /* memory for msgid */
char              user[60]="aq/AQ";  /* user login password */
char              subject[30];  /* components of */
char              txt[80];  /* Message_typ */
char              correlation1[30];  /* message correlation  */
```

```
char              correlation2[30];
int               status;  /* code returned by the OCI calls */

/* Dequeue by correlation and msgid */

/* Connect to the database: */
EXEC SQL CONNECT :user;
EXEC SQL WHENEVER SQLERROR DO sql_error("Oracle Error :");

/* Allocate space in the object cache for the host variable: */
EXEC SQL ALLOCATE :message;

/* Get the OCI Env handle: */
if (SQLEnvGet(SQL_SINGLE_RCTX, &oeh) != OCI_SUCCESS)
{
 printf(" error in SQLEnvGet \n");
 exit(1);
}
/* Get the OCI Error handle: */
if (status = OCIHandleAlloc((dvoid *)oeh, (dvoid **)&err,
(ub4)OCI_HTYPE_ERROR, (ub4)0, (dvoid **)0))
{
printf(" error in OCIHandleAlloc %d \n", status);
exit(1);
}


/* Assign memory for msgid:
Memory needs to be allocated explicitly to OCIRaw*: */
if (status=OCIRawAssignBytes(oeh, err, msgmem, 16, &msgid))
{
printf(" error in  OCIRawAssignBytes  %d \n", status);
exit(1);
}

/* First enqueue */

strcpy(correlation1, "1st message");
strcpy(subject, "NORMAL ENQUEUE1");
strcpy(txt, "The Enqueue was done through PLSQL embedded in PROC");

/* Initialize the components of message: */
EXEC SQL OJECT SET subject, text OF :message TO :subject, :txt;

/* Embedded PLSQL call to the AQ enqueue procedure: */
EXEC SQL EXECUTE
DECLARE
```

```
message_properties   dbms_aq.message_properties_t;
enqueue_options      dbms_aq.enqueue_options_t;
BEGIN
/* Bind the host variable 'correlation1': to message correlation*/
message_properties.correlation := :correlation1;

/* Bind the host variable 'message' to payload and
 return message id into host variable 'msgid': */
dbms_aq.enqueue(queue_name => 'msg_queue',
message_properties => message_properties,
enqueue_options => enqueue_options,
payload => :message:imsg,    /* indicator has to be specified */
msgid => :msgid);
END;
END-EXEC;
/* Commit work: */
EXEC SQL COMMIT;

printf("Enqueued Message \n");
printf("Subject  :%s\n",subject);
printf("Text     :%s\n",txt);

/* Second enqueue */

strcpy(correlation2, "2nd message");
strcpy(subject, "NORMAL ENQUEUE2");
strcpy(txt, "The Enqueue was done through PLSQL embedded in PROC");

/* Initialize the components of message: */
EXEC SQL OBJECT SET subject, text OF :messsage TO :subject,:txt;

/* Embedded PLSQL call to the AQ enqueue procedure: */
EXEC SQL EXECUTE
DECLARE
message_properties   dbms_aq.message_properties_t;
enqueue_options      dbms_aq.enqueue_options_t;
msgid                RAW(16);
BEGIN
/* Bind the host variable 'correlation2':  to message correlaiton */
message_properties.correlation := :correlation2;

/* Bind the host variable 'message': to payload */
dbms_aq.enqueue(queue_name => 'msg_queue',
message_properties => message_properties,
enqueue_options => enqueue_options,
payload => :message,
msgid => msgid);
```

```
END;
END-EXEC;
/* Commit work: */
EXEC SQL COMMIT;
printf("Enqueued Message \n");
printf("Subject  :%s\n",subject);
printf("Text     :%s\n",txt);

/* First dequeue - by  correlation */

EXEC SQL EXECUTE
DECLARE
message_properties  dbms_aq.message_properties_t;
dequeue_options     dbms_aq.dequeue_options_t;
msgid               RAW(16);
BEGIN
/* Dequeue by  correlation in host variable 'correlation2': */
dequeue_options.correlation := :correlation2;

/* Return the payload into host variable 'message': */
dbms_aq.dequeue(queue_name => 'msg_queue',
message_properties => message_properties,
dequeue_options => dequeue_options,
payload => :message,
msgid => msgid);
END;
END-EXEC;
/* Commit work : */
EXEC SQL COMMIT;

/* Extract the values of the components of message: */
EXEC SQL OBJECT GET subject, text FROM :message INTO :subject,:txt;

printf("Dequeued Message \n");
printf("Subject  :%s\n",subject);
printf("Text     :%s\n",txt);

/* SECOND DEQUEUE - by MSGID  */

EXEC SQL EXECUTE
DECLARE
message_properties  dbms_aq.message_properties_t;
dequeue_options     dbms_aq.dequeue_options_t;
msgid               RAW(16);
BEGIN
/* Dequeue by msgid in host variable 'msgid': */
dequeue_options.msgid := :msgid;
```

```
/* Return the payload into host variable 'message':  */
dbms_aq.dequeue(queue_name => 'msg_queue',
message_properties => message_properties,
dequeue_options => dequeue_options,
payload => :message,
msgid => msgid);
END;
END-EXEC;
/* Commit work: */
EXEC SQL COMMIT;
}
```

相関識別子およびメッセージ ID によるメッセージのエンキューおよびデキュー : OCI

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>

struct message
{
  OCIString   *subject;
  OCIString   *data;
};
typedef struct message message;

struct null_message
{
  OCIInd    null_adt;
  OCIInd    null_subject;
  OCIInd    null_data;
};
typedef struct null_message null_message;

int main()
{
  OCIEnv       *envhp;
  OCIServer    *srvhp;
  OCIError     *errhp;
  OCISvcCtx    *svchp;
  dvoid        *tmp;
  OCIType      *mesg_tdo = (OCIType *) 0;
  message      msg;
```

```
null_message nmsg;
message      *mesg     = &msg;
null_message *nmesg    = &nmsg;
message      *deqmesg  = (message *)0;
null_message *ndeqmesg = (null_message *)0;

OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
              (dvoid * (*)()) 0,  (void (*)()) 0 );

OCIHandleAlloc((dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
               52, (dvoid **) &tmp);

OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 21, (dvoid **) &tmp  );

OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
               52, (dvoid **) &tmp);
OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
               52, (dvoid **) &tmp);

OCIServerAttach(srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
               52, (dvoid **) &tmp);

OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
           (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

OCILogon(envhp, errhp, &svchp, "AQ", strlen("AQ"), "AQ", strlen("AQ"), 0, 0);

/* Obtain TDO of message_typ */
OCITypeByName(envhp, errhp, svchp, (CONST text *)"AQ", strlen("AQ"),
              (CONST text *)"MESSAGE_TYP", strlen("MESSAGE_TYP"),
              (text *)0, 0, OCI_DURATION_SESSION, OCI_TYPEGET_ALL, &mesg_tdo);

/* Prepare the message payload */
mesg->subject = (OCIString *)0;
mesg->data = (OCIString *)0;
OCIStringAssignText(envhp, errhp,
                    (CONST text *)"NORMAL MESSAGE", strlen("NORMAL MESSAGE"),
                    &mesg->subject);
OCIStringAssignText(envhp, errhp,
                    (CONST text *)"OCI ENQUEUE", strlen("OCI ENQUEUE"),
                    &mesg->data);
nmesg->null_adt = nmesg->null_subject = nmesg->null_data = OCI_IND_NOTNULL;

/* Enqueue into the msg_queue */
OCIAQEnq(svchp, errhp, (CONST text *)"msg_queue", 0, 0,
```

```
            mesg_tdo, (dvoid **)&mesg, (dvoid **)&nmesg, 0, 0);
  OCITransCommit(svchp, errhp, (ub4) 0);

  /* Dequeue from the msg_queue */
  OCIAQDeq(svchp, errhp, (CONST text *)"msg_queue", 0, 0,
            mesg_tdo, (dvoid **)&deqmesg, (dvoid **)&ndeqmesg, 0, 0);
  printf("Subject: %s\n", OCIStringPtr(envhp, deqmesg->subject));
  printf("Text: %s\n", OCIStringPtr(envhp, deqmesg->data));
  OCITransCommit(svchp, errhp, (ub4) 0);
}
```

マルチ・コンシューマ・キューでのメッセージのエンキューおよびデキュー: PL/SQL

```
/* Create subscriber list: */
DECLARE
   subscriber aq$_agent;

   /* Add subscribers RED and GREEN to the suscriber list: */
BEGIN
   subscriber := aq$_agent('RED', NULL, NULL);
   DBMS_AQADM.ADD_SUBSCRIBER(queue_name => 'msg_queue_multiple',
   subscriber => subscriber);

   subscriber := aq$_agent('GREEN', NULL, NULL);
   DBMS_AQADM.ADD_SUBSCRIBER(queue_name => 'msg_queue_multiple',
   subscriber => subscriber);
END;

DECLARE
   enqueue_options     DBMS_AQ.enqueue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   recipients          DBMS_AQ.aq$_recipient_list_t;
   message_handle      RAW(16);
   message             aq.message_typ;

   /* Enqueue MESSAGE 1 for subscribers to the queue
   i.e. for RED and GREEN: */
BEGIN
   message := message_typ('MESSAGE 1',
   'This message is queued for queue subscribers.');

   DBMS_AQ.ENQUEUE(queue_name => 'msg_queue_multiple',
   enqueue_options    => enqueue_options,
   message_properties => message_properties,
```

```
   payload            => message,
   msgid              => message_handle);

   /* Enqueue MESSAGE 2 for specified recipients i.e. for RED and BLUE.*/
   message := message_typ('MESSAGE 2',
   'This message is queued for two recipients.');
   recipients(1) := aq$_agent('RED', NULL, NULL);
   recipients(2) := aq$_agent('BLUE', NULL, NULL);
   message_properties.recipient_list := recipients;

   DBMS_AQ.ENQUEUE(queue_name => 'msg_queue_multiple',
           enqueue_options    => enqueue_options,
           message_properties => message_properties,
           payload            => message,
           msgid              => message_handle);

   COMMIT;
END;
```

RED はキューのサブスクライバであると同時に、MESSAGE 2 の指定された受信者であることに注意してください。それとは対照的に GREEN は、受信者が指定されていないキュー内のメッセージ（この場合 MESSAGE）に対するサブスクライバでしかありません。BLUE は、キューのサブスクライバではありませんが、MESSAGE 2 の宛先に指定されています。

```
/* Dequeue messages from msg_queue_multiple: */
DECLARE
   dequeue_options     DBMS_AQ.dequeue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;
   no_messages         exception;
   pragma exception_init (no_messages, -25228);

BEGIN

   dequeue_options.wait := DBMS_AQ.NO_WAIT;
   BEGIN
   /* Consumer BLUE will get MESSAGE 2: */
   dequeue_options.consumer_name := 'BLUE';
   dequeue_options.navigation := FIRST_MESSAGE;

   LOOP

   DBMS_AQ.DEQUEUE(queue_name   => 'msg_queue_multiple',
             dequeue_options    => dequeue_options,
             message_properties => message_properties,
             payload            => message,
```

```
            msgid              => message_handle);

      DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
            ' ... ' || message.text );
      dequeue_options.navigation := NEXT_MESSAGE;

   END LOOP;
   EXCEPTION
   WHEN no_messages THEN
   DBMS_OUTPUT.PUT_LINE ('No more messages for BLUE');
   COMMIT;
END;

BEGIN
/* Consumer RED will get MESSAGE 1 and MESSAGE 2: */
   dequeue_options.consumer_name := 'RED';
dequeue_options.navigation := DBMS_AQ.FIRST_MESSAGE
   LOOP
      DBMS_AQ.DEQUEUE(queue_name   => 'msg_queue_multiple',
                dequeue_options    => dequeue_options,
                message_properties => message_properties,
                payload            => message,
                msgid              => message_handle);

      DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
                                          ' ... ' || message.text );
      dequeue_options.navigation := NEXT_MESSAGE;
   END LOOP;
   EXCEPTION
   WHEN no_messages THEN
      DBMS_OUTPUT.PUT_LINE ('No more messages for RED');
   COMMIT;
END;

BEGIN
   /* Consumer GREEN will get MESSAGE 1: */
   dequeue_options.consumer_name := 'GREEN';
   dequeue_options.navigation := FIRST_MESSAGE;
   LOOP
      DBMS_AQ.DEQUEUE(queue_name   => 'msg_queue_multiple',
                dequeue_options    => dequeue_options,
                message_properties => message_properties,
                payload            => message,
                msgid              => message_handle);

      DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
          ' ... ' || message.text );
```

```
      dequeue_options.navigation := NEXT_MESSAGE;
   END LOOP;
   EXCEPTION
   WHEN no_messages THEN
      DBMS_OUTPUT.PUT_LINE ('No more messages for GREEN');
   COMMIT;
END;
```

マルチ・コンシューマ・キューでのメッセージのエンキューおよびデキュー: OCI

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
CONNECT aqadm/aqadm
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE(
   queue_table        => 'aq.qtable_multi',
   multiple_consumers => true,
   queue_payload_type => 'aq.message_typ');
EXECUTE DBMS_AQADM.START_QUEUE('aq.msg_queue_multiple');
CONNECT aq/aq
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>

struct message
{
  OCIString   *subject;
  OCIString   *data;
};
typedef struct message message;

struct null_message
{
  OCIInd    null_adt;
  OCIInd    null_subject;
  OCIInd    null_data;
};
typedef struct null_message null_message;

int main()
{
```

```
OCIEnv                *envhp;
OCIServer             *srvhp;
OCIError              *errhp;
OCISvcCtx             *svchp;
dvoid                 *tmp;
OCIType               *mesg_tdo = (OCIType *) 0;
message               msg;
null_message          nmsg;
message               *mesg = &msg;
null_message          *nmesg = &nmsg;
message               *deqmesg = (message *)0;
null_message          *ndeqmesg = (null_message *)0;
OCIAQMsgProperties    *msgprop = (OCIAQMsgProperties *)0;
OCIAQAgent            *agents[2];
OCIAQDeqOptions       *deqopt = (OCIAQDeqOptions *)0;
ub4                   wait = OCI_DEQ_NO_WAIT;
ub4                   navigation = OCI_DEQ_FIRST_MSG;

OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
              (dvoid * (*)()) 0,  (void (*)()) 0 );



OCIHandleAlloc((dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
               52, (dvoid **) &tmp);

OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 21, (dvoid **) &tmp  );

OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
               52, (dvoid **) &tmp);
OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
               52, (dvoid **) &tmp);

OCIServerAttach(srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
               52, (dvoid **) &tmp);

OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
           (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

OCILogon(envhp, errhp, &svchp, "AQ", strlen("AQ"), "AQ", strlen("AQ"), 0, 0);

/* Obtain TDO of message_typ */
OCITypeByName(envhp, errhp, svchp, (CONST text *)"AQ", strlen("AQ"),
             (CONST text *)"MESSAGE_TYP", strlen("MESSAGE_TYP"),
             (text *)0, 0, OCI_DURATION_SESSION, OCI_TYPEGET_ALL, &mesg_tdo);
```

```
/* Prepare the message payload */
mesg->subject = (OCIString *)0;
mesg->data = (OCIString *)0;
OCIStringAssignText(envhp, errhp,
                    (CONST text *)"MESSAGE 1", strlen("MESSAGE 1"),
                    &mesg->subject);
OCIStringAssignText(envhp, errhp,
                   (CONST text *)"mesg for queue subscribers",
                   strlen("mesg for queue subscribers"), &mesg->data);
nmesg->null_adt = nmesg->null_subject = nmesg->null_data = OCI_IND_NOTNULL;

/* Enqueue MESSAGE 1 for subscribers to the queue i.e. for RED and GREEN */
OCIAQEnq(svchp, errhp, (CONST text *)"msg_queue_multiple", 0, 0,
         mesg_tdo, (dvoid **)&mesg, (dvoid **)&nmesg, 0, 0);

/* Enqueue MESSAGE 2 for specified recipients i.e. for RED and BLUE */
/* prepare message payload */
OCIStringAssignText(envhp, errhp,
                    (CONST text *)"MESSAGE 2", strlen("MESSAGE 2"),
                    &mesg->subject);
OCIStringAssignText(envhp, errhp,
     (CONST text *)"mesg for two recipients",
     strlen("mesg for two recipients"), &mesg->data);

/* Allocate AQ message properties and agent descriptors */
OCIDescriptorAlloc(envhp, (dvoid **)&msgprop,
                   OCI_DTYPE_AQMSG_PROPERTIES, 0, (dvoid **)0);
OCIDescriptorAlloc(envhp, (dvoid **)&agents[0],
                   OCI_DTYPE_AQAGENT, 0, (dvoid **)0);
OCIDescriptorAlloc(envhp, (dvoid **)&agents[1],
                   OCI_DTYPE_AQAGENT, 0, (dvoid **)0);

/* Prepare the recipient list, RED and BLUE */
OCIAttrSet(agents[0], OCI_DTYPE_AQAGENT, "RED", strlen("RED"),
           OCI_ATTR_AGENT_NAME, errhp);
OCIAttrSet(agents[1], OCI_DTYPE_AQAGENT, "BLUE", strlen("BLUE"),
           OCI_ATTR_AGENT_NAME, errhp);
OCIAttrSet(msgprop, OCI_DTYPE_AQMSG_PROPERTIES, (dvoid *)agents, 2,
           OCI_ATTR_RECIPIENT_LIST, errhp);

OCIAQEnq(svchp, errhp, (CONST text *)"msg_queue_multiple", 0, msgprop,
         mesg_tdo, (dvoid **)&mesg, (dvoid **)&nmesg, 0, 0);

OCITransCommit(svchp, errhp, (ub4) 0);

/* Now dequeue the messages using different consumer names */
```

```
/* Allocate dequeue options descriptor to set the dequeue options */
OCIDescriptorAlloc(envhp, (dvoid **)&deqopt, OCI_DTYPE_AQDEQ_OPTIONS, 0,
                   (dvoid **)0);

/* Set wait parameter to NO_WAIT so that the dequeue returns immediately */
OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS, (dvoid *)&wait, 0,
           OCI_ATTR_WAIT, errhp);

/* Set navigation to FIRST_MESSAGE so that the dequeue resets the position */
/* after a new consumer_name is set in the dequeue options           */
OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS, (dvoid *)&navigation, 0,
           OCI_ATTR_NAVIGATION, errhp);

/* Dequeue from the msg_queue_multiple as consumer BLUE */
OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS, (dvoid *)"BLUE", strlen("BLUE"),
           OCI_ATTR_CONSUMER_NAME, errhp);

while (OCIAQDeq(svchp, errhp, (CONST text *)"msg_queue_multiple", deqopt, 0,
                mesg_tdo, (dvoid **)&deqmesg, (dvoid **)&ndeqmesg, 0, 0)
                == OCI_SUCCESS)
{
  printf("Subject: %s\n", OCIStringPtr(envhp, deqmesg->subject));
  printf("Text: %s\n", OCIStringPtr(envhp, deqmesg->data));
}
OCITransCommit(svchp, errhp, (ub4) 0);

/* Dequeue from the msg_queue_multiple as consumer RED */
OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS, (dvoid *)"RED", strlen("RED"),
      OCI_ATTR_CONSUMER_NAME, errhp);
while (OCIAQDeq(svchp, errhp, (CONST text *)"msg_queue_multiple", deqopt, 0,
  mesg_tdo, (dvoid **)&deqmesg, (dvoid **)&ndeqmesg, 0, 0)
  == OCI_SUCCESS)
{
  printf("Subject: %s\n", OCIStringPtr(envhp, deqmesg->subject));
  printf("Text: %s\n", OCIStringPtr(envhp, deqmesg->data));
}
OCITransCommit(svchp, errhp, (ub4) 0);

/* Dequeue from the msg_queue_multiple as consumer GREEN */
OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS,(dvoid *)"GREEN",strlen("GREEN"),
      OCI_ATTR_CONSUMER_NAME, errhp);
while (OCIAQDeq(svchp, errhp, (CONST text *)"msg_queue_multiple", deqopt, 0,
  mesg_tdo, (dvoid **)&deqmesg, (dvoid **)&ndeqmesg, 0, 0)
  == OCI_SUCCESS)
{
  printf("Subject: %s\n", OCIStringPtr(envhp, deqmesg->subject));
  printf("Text: %s\n", OCIStringPtr(envhp, deqmesg->data));
```

```
  }
  OCITransCommit(svchp, errhp, (ub4) 0);
}
```

メッセージのグループ化によるメッセージのエンキューおよびデキュー : PL/SQL

```
CONNECT aq/aq

EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE (
   queue_table          => 'aq.msggroup',
   queue_payload_type   => 'aq.message_typ',
   message_grouping     => DBMS_AQADM.TRANSACTIONAL);

EXECUTE DBMS_AQADM.CREATE_QUEUE(
   queue_name      => 'msggroup_queue',
   queue_table     => 'aq.msggroup');

EXECUTE DBMS_AQADM.START_QUEUE(
   queue_name => 'msggroup_queue');

/* Enqueue three messages in each transaction */
DECLARE
   enqueue_options     DBMS_AQ.enqueue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;

BEGIN

  /* Loop through three times, committing after every iteration */
  FOR txnno in 1..3 LOOP

    /* Loop through three times, enqueuing each iteration */
    FOR mesgno in 1..3 LOOP
      message := message_typ('GROUP#' || txnno,
                'Message#' || mesgno ||  ' in group' || txnno);

      DBMS_AQ.ENQUEUE(queue_name         => 'msggroup_queue',
                  enqueue_options        => enqueue_options,
                  message_properties     => message_properties,
                  payload                => message,
                  msgid                  => message_handle);
    END LOOP;
    /* Commit the transaction */
```

```
    COMMIT;
  END LOOP;
END;

/* Now dequeue the messages as groups */
DECLARE
   dequeue_options     DBMS_AQ.dequeue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   message_handle      RAW(16);
   message             aq.message_typ;

   no_messages   exception;
   end_of_group  exception;

   PRAGMA EXCEPTION_INIT (no_messages, -25228);
   PRAGMA EXCEPTION_INIT (end_of_group, -25235);

BEGIN
   dequeue_options.wait       := DBMS_AQ.NO_WAIT;
   dequeue_options.navigation := DBMS_AQ.FIRST_MESSAGE;

   LOOP
      BEGIN
      DBMS_AQ.DEQUEUE(queue_name   => 'msggroup_queue',
                dequeue_options    => dequeue_options,
                message_properties => message_properties,
                payload            => message,
                msgid              => message_handle);

     DBMS_OUTPUT.PUT_LINE ('Message: ' || message.subject ||
          ' ... ' || message.text );

     dequeue_options.navigation := DBMS_AQ.NEXT_MESSAGE;

     EXCEPTION
       WHEN end_of_group THEN
         DBMS_OUTPUT.PUT_LINE ('Finished processing a group of messages');
         COMMIT;
         dequeue_options.navigation := DBMS_AQ.NEXT_TRANSACTION;
     END;
   END LOOP;
   EXCEPTION
     WHEN no_messages THEN
       DBMS_OUTPUT.PUT_LINE ('No more messages');
END;
```

LOB 属性を含むオブジェクト型メッセージのエンキューおよびデキュー : PL/SQL

```
/* Create the message payload object type with one or more LOB attributes. On
   enqueue, set the LOB attribute to EMPTY_BLOB. After the enqueue completes,
   before you commit your transaction. Select the LOB attribute from the
   user_data column of the queue table or queue table view. You can now
   use the LOB interfaces (which are available through both OCI and PL/SQL) to
   write the LOB data to the queue.  On dequeue, the message payload
   will contain the LOB locator. You can use this LOB locator after
   the dequeue, but before you commit your transaction, to read the LOB data.
 */
/* Setup the accounts: */

connect system/manager

CREATE USER aqadm IDENTIFIED BY aqadm;
GRANT CONNECT, RESOURCE TO aqadm;
GRANT aq_administrator_role TO aqadm;

CREATE USER aq IDENTIFIED BY aq;
GRANT CONNECT, RESOURCE TO aq;
GRANT EXECUTE ON DBMS_AQ TO aq;
CREATE TYPE aq.message AS OBJECT(id      NUMBER,
                                 subject VARCHAR2(100),
                                 data    BLOB,
                                 trailer NUMBER);
CREATE TABLESPACE aq_tbs DATAFILE 'aq.dbs' SIZE 2M REUSE;


/* create the queue table, queues and start the queue: */

CONNECT aqadm/aqadm
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE(
   queue_table        => 'aq.qt1',
   queue_payload_type => 'aq.message');
EXECUTE DBMS_AQADM.CREATE_QUEUE(
   queue_name  => 'aq.queue1',
   queue_table =>  'aq.qt1');
EXECUTE DBMS_AQADM.START_QUEUE(queue_name => 'aq.queue1');

/* End set up: */

/* Enqueue of Large data types: */

CONNECT aq/aq
CREATE OR REPLACE PROCEDURE blobenqueue(msgno IN NUMBER) AS
```

```
enq_userdata aq.message;
enq_msgid    RAW(16);
enqopt       DBMS_AQ.enqueue_options_t;
msgprop      DBMS_AQ.message_properties_t;
lob_loc      BLOB;
buffer       RAW(4096);

BEGIN

   buffer := HEXTORAW(RPAD('FF', 4096, 'FF'));
   enq_userdata := aq.message(msgno, 'Large Lob data', EMPTY_BLOB(), msgno);
   DBMS_AQ.ENQUEUE('aq.queue1', enqopt, msgprop, enq_userdata, enq_msgid);

   --select the lob locator for the queue table
   SELECT t.user_data.data INTO lob_loc
      FROM qt1 t
      WHERE t.msgid = enq_msgid;

   DBMS_LOB.WRITE(lob_loc, 2000, 1, buffer );
   COMMIT;
END;

/* Dequeue lob data: */

CREATE OR REPLACE PROCEDURE blobdequeue AS
   dequeue_options    DBMS_AQ.dequeue_options_t;
   message_properties DBMS_AQ.message_properties_t;
   mid                RAW(16);
   pload              aq.message;
   lob_loc            BLOB;
   amount             BINARY_INTEGER;
   buffer             RAW(4096);

BEGIN
   DBMS_AQ.DEQUEUE('aq.queue1', dequeue_options, message_properties,
                   pload, mid);
   lob_loc := pload.data;

   -- read the lob data info buffer
   amount := 2000;
   DBMS_LOB.READ(lob_loc, amount, 1, buffer);
   DBMS_OUTPUT.PUT_LINE('Amount of data read: '||amount);
   COMMIT;
END;

/* Do the enqueues and dequeues: */
 SET SERVEROUTPUT ON
```

```
BEGIN
   FOR i IN 1..5 LOOP
      blobenqueue(i);
   END LOOP;
END;

BEGIN
   FOR i IN 1..5 LOOP
     blobdequeue();
   END LOOP;
END;
```

LOB 属性を含むオブジェクト型メッセージのエンキューおよびデキュー : Java

1. メッセージ型（CLOB および BLOB を含むユーザー定義型）を作成します。

```
connect aquser/aquser

create type LobMessage as object(id        NUMBER,
                                 subject   varchar2(100),
                                 data      blob,
                                 cdata     clob,
                                 trailer   number);
```

2. キュー表およびキューを作成します。

```
connect aquser/aquser
execute dbms_aqadm.create_queue_table(
          queue_table => 'qt_adt',
          queue_payload_type => 'LOBMESSAGE',
          comment => 'single-consumer, default sort ordering, ADT Message',
          compatible => '8.1.0'
        );

execute dbms_aqadm.create_queue(
          queue_name  => 'q1_adt',
          queue_table => 'qt_adt'
        );

execute dbms_aqadm.start_queue(queue_name => 'q1_adt');
```

3. JPublisher を実行し、LobMesage に対応する Java クラスを生成します。

```
Oracle object type

jpub -user=aquser/aquser -sql=LobMessage -case=mixed -methods=false
-usertypes=oracle -compatible=CustomDatum
```

4. メッセージをエンキューおよびデキューします。

```
public static void runTest(AQSession aq_sess)
{
  Connection                db_conn   = null;
  AQEnqueueOption           eq_option = null;
  AQDequeueOption           dq_option = null;
  AQQueue                   queue1    = null;
  AQMessage                 adt_msg   = null;
  AQMessage                 adt_msg2  = null;
  AQObjectPayload           sPayload  = null;
  AQObjectPayload           sPayload2 = null;
  LobMessage                sPayl     = null;
  LobMessage                sPayl2    = null;
  AQObjectPayload           rPayload  = null;
  LobMessage                rPayl     = null;
  byte[]                    smsgid;
  AQMessage                 rMessage  = null;
  int                       i         = 0;
  int                       j         = 0;
  int                       id        = 0;
  boolean                   more      = false;
  byte[]                    b_array;
  char[]                    c_array;
  String                    mStr      = null;
  BLOB                      b1        = null;
  CLOB                      c1        = null;
  BLOB                      b2        = null;
  CLOB                      c2        = null;
  BLOB                      b3        = null;
  CLOB                      c3        = null;
  int                       b_len     = 0;
  int                       c_len     = 0;
  OracleCallableStatement   blob_stmt0= null;
  OracleCallableStatement   clob_stmt0= null;
  OracleResultSet           rset0     = null;
  OracleResultSet           rset1     = null;
  OracleCallableStatement   blob_stmt = null;
  OracleResultSet           rset2     = null;
  OracleCallableStatement   clob_stmt = null;
```

```
OracleResultSet           rset3     = null;

try
{

 db_conn = ((AQOracleSession)aq_sess).getDBConnection();

 queue1  = aq_sess.getQueue("aquser", "q1_adt");


  b_array = new byte[5000];
  c_array = new char[5000];
  for (i = 0; i < 5000; i++)
  {
     b_array[i] = 67;
     c_array[i] = 'c';
  }
  sPayl = new LobMessage();


  System.out.println("Enqueue Long messages");

  eq_option = new AQEnqueueOption();

  /* Enqueue messages with LOB attributes */
  for ( i = 0; i < 10; i++)
  {
    adt_msg = queue1.createMessage();

    sPayload = adt_msg.getObjectPayload();

    /*  Get Empty BLOB handle */
    blob_stmt0 = (OracleCallableStatement)db_conn.prepareCall(
                 "select empty_blob() from dual");
    rset0 = (OracleResultSet) blob_stmt0.executeQuery ();
    try
    {
      if (rset0.next())
      {
        b1 = (oracle.sql.BLOB)rset0.getBlob(1);
      }
      if (b1 == null)
      {
        System.out.println("select empty_blob() from dual failed");
      }
    }
    catch (Exception ex)
```

```
  {
    System.out.println("Exception during select from dual " + ex);
    ex.printStackTrace();
  }

/*  Get Empty CLOB handle */
  clob_stmt0 = (OracleCallableStatement)db_conn.prepareCall(
               "select empty_clob() from dual");
  rset1 = (OracleResultSet) clob_stmt0.executeQuery ();
  try
  {
    if (rset1.next())
    {
      c1 = (oracle.sql.CLOB)rset1.getClob(1);
    }
    if (c1 == null)
    {
      System.out.println("select empty_clob() from dual failed");
    }
  }
  catch (Exception ex)
  {
    System.out.println("Exception2 during select from dual " + ex);
    ex.printStackTrace();
  }
  id = i+1;
  mStr = "Message #" + id;
  sPayl.setId(new BigDecimal(id));
  sPayl.setTrailer(new BigDecimal(id));
  sPayl.setSubject(mStr);
  sPayl.setData(b1);
  sPayl.setCdata(c1);

  /* Set Object Payload data */
  sPayload.setPayloadData(sPayl);

  /* Enqueue the message */
  queue1.enqueue(eq_option, adt_msg);
  System.out.println("Enqueued Message: " + id );
  smsgid = adt_msg.getMessageId();

  /*
   * Note: The message is initially enqueued with an EMPTY BLOB and CLOB
   * After enqueuing the message, we need to get the lob locators and
   * then populate the LOBs
   */
  blob_stmt = (OracleCallableStatement)db_conn.prepareCall(
```

```
      "SELECT user_data FROM qt_adt where msgid = ?");
blob_stmt.setBytes(1,smsgid);
rset2 = (OracleResultSet) blob_stmt.executeQuery ();
try
{
  if (rset2.next())
  {
    /* Get message contents */
    sPayl2 = (LobMessage)rset2.getCustomDatum(1,
                  ((CustomDatumFactory)LobMessage.getFactory()));

    /* Get BLOB locator */
    b2 = sPayl2.getData();

    /* Popuate the BLOB */
    if (b2 == null)
    {
       System.out.println("Blob select null");
    }
    if ((i % 3) == 0)
    {
      b_len = b2.putBytes(1000,b_array);
    }
    else
    {
      b_len = b2.putBytes(1,b_array);
    }

    /* Get CLOB locator */
    c2 = sPayl2.getCdata();

    /* Populate the CLOB */
    if (c2 == null)
    {
       System.out.println("Clob select null");
    }
    if ((i % 4) == 0)
    {
      c_len = c2.putChars(2500,c_array);
    }
    else
    {
      c_len = c2.putChars(1,c_array);
    }
  }
}
catch (Exception ex)
```

```
    {
      System.out.println("Blob or Clob exception: " + ex);
    }

  }

  Thread.sleep(30000);


 // dequeue messages
 dq_option = new AQDequeueOption();
 dq_option.setWaitTime(AQDequeueOption.WAIT_NONE);


 for (i = 0 ; i < 10 ; i++)
 {
   /* Dequeue the message */
    adt_msg2 = ((AQOracleQueue)queue1).dequeue(dq_option,
                                              LobMessage.getFactory());

    /* Get payload containing LOB data */
    rPayload = adt_msg2.getObjectPayload();
    rPayl = (LobMessage) rPayload.getPayloadData();

    System.out.println("\n  Message: #" + (i+1));
    System.out.println("    Id: " + rPayl.getId());
    System.out.println("    Subject: " + rPayl.getSubject());

    /* Get BLOB data */
    b3 = rPayl.getData();
    System.out.println("    " + b3.length() + " bytes of data");

    /* Get CLOB data */
    c3 = rPayl.getCdata();
    System.out.println("    " + c3.length() + " chars of data");
    System.out.println("    Trailer: " + rPayl.getTrailer());
    db_conn.commit();
 }


}
catch (java.sql.SQLException sql_ex)
{
  System.out.println("SQL Exception: " + sql_ex);
  sql_ex.printStackTrace();
}
catch (Exception ex)
```

```
    {
      System.out.println("Exception-2: " + ex);
      ex.printStackTrace();
    }

  }
```

伝播

注意： キューやキュー表を作成したり、キューを開始または使用可能にしないと、機能しない例もあります。

リモートのサブスクライバ / 受信者用のメッセージのマルチ・コンシューマ・キューへのエンキューおよび伝播のスケジュール : PL/SQL

```
/* Create subscriber list: */
DECLARE
   subscriber aq$_agent;

   /* Add subscribers RED and GREEN with different addresses  to the suscriber
   list: */
BEGIN
   BEGIN
      /* Add subscriber RED that will dequeue messages from another_msg_queue
      queue in the same datatbase */
      subscriber := aq$_agent('RED', 'another_msg_queue', NULL);
      DBMS_AQADM.ADD_SUBSCRIBER(queue_name => 'msg_queue_multiple',
      subscriber => subscriber);

      /* Schedule propagation from msg_queue_multiple to other queues in the
      same
      database: */
      DBMS_AQADM.SCHEDULE_PROPAGATION(queue_name => 'msg_queue_multiple');

      /* Add subscriber GREEN that will dequeue messages from the msg_queue
      queue
      in another database reached by the database link another_db.world */
      subscriber := aq$_agent('GREEN', 'msg_queue@another_db.world', NULL);
      DBMS_AQADM.ADD_SUBSCRIBER(queue_name => 'msg_queue_multiple',
      subscriber => subscriber);

      /* Schedule propagation from msg_queue_multiple to other queues in the
      database "another_database": */
   END;
```

```
   BEGIN
      DBMS_AQADM.SCHEDULE_PROPAGATION(queue_name => 'msg_queue_multiple',
      destination  => 'another_db.world');
   END;
END;

DECLARE
   enqueue_options     DBMS_AQ.enqueue_options_t;
   message_properties  DBMS_AQ.message_properties_t;
   recipients          DBMS_AQ.aq$_recipient_list_t;
   message_handle      RAW(16);
   message             aq.message_typ;

/* Enqueue MESSAGE 1 for subscribers to the queue
i.e. for RED at address another_msg_queue and GREEN at address msg_queue@another_
db.world: */
BEGIN
   message := message_typ('MESSAGE 1',
   'This message is queued for queue subscribers.');

   DBMS_AQ.ENQUEUE(queue_name => 'msg_queue_multiple',
           enqueue_options    => enqueue_options,
           message_properties => message_properties,
           payload            => message,
           msgid              => message_handle);

   /* Enqueue MESSAGE 2 for specified recipients i.e. for RED at address
   another_msg_queue and BLUE.*/
   message := message_typ('MESSAGE 2',
   'This message is queued for two recipients.');
   recipients(1) := aq$_agent('RED', 'another_msg_queue', NULL);
   recipients(2) := aq$_agent('BLUE', NULL, NULL);
   message_properties.recipient_list := recipients;

   DBMS_AQ.ENQUEUE(queue_name => 'msg_queue_multiple',
           enqueue_options    => enqueue_options,
           message_properties => message_properties,
           payload            => message,
           msgid              => message_handle);

   COMMIT;
END;
```

注意： アドレス another_msg_queue の RED は、キューのサブスクライバであると同時に、MESSAGE 2 の指定された受信者です。それとは対照的に、アドレス msg_queue@another_db.world の GREEN は、受信者が指定されていないキュー内のメッセージ（この場合 MESSAGE 1）に対するサブスクライバでしかありません。BLUE はキューのサブスクライバではありませんが、MESSAGE 2 の宛先に指定されています。

同一データベース内の 1 つのキューから他のキューへの伝播管理 : PL/SQL

```
/* Schedule propagation from queue q1def to other queues in the same database */
EXECUTE DBMS_AQADM.SCHEDULE_PROPAGATION(queue_name => 'q1def');

/* Disable propagation from queue q1def to other queues in the same
database */
EXECUTE DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE(
   queue_name => 'q1def');

/* Alter schedule from queue q1def to other queues in the same database */
EXECUTE DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE(
   queue_name => 'q1def',
   duration   => '2000',
   next_time  => 'SYSDATE + 3600/86400',
   latency    => '32');

/* Enable propagation from queue q1def to other queues in the same database */
EXECUTE DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE(
   queue_name => 'q1def');

/* Unschedule propagation from queue q1def to other queues in the same database */
EXECUTE DBMS_AQADM.UNSCHEDULE_PROPAGATION(
   queue_name => 'q1def');
```

1 つのキューから他のデータベース内の他のキューへの伝播管理 : PL/SQL

```
/* Schedule propagation from queue q1def to other queues in another  database
reached by the database link another_db.world  */
EXECUTE DBMS_AQADM.SCHEDULE_PROPAGATION(
   queue_name   => 'q1def',
   destination  => 'another_db.world');

/* Disable propagation from queue q1def to other queues in another database reached
by the database link another_db.world */
EXECUTE DBMS_AQADM.DISABLE_PROPAGATION_SCHEDULE(
   queue_name  => 'q1def',
```

```
   destination => 'another_db.world');

/* Alter schedule from queue q1def to other queues in another database  reached by
the database link another_db.world  */
EXECUTE DBMS_AQADM.ALTER_PROPAGATION_SCHEDULE(
   queue_name  => 'q1def',
   destination => 'another_db.world',
   duration    => '2000',
   next_time   => 'SYSDATE +  3600/86400',
   latency     => '32');

/* Enable propagation from queue q1def to other queues in another  database reached
by the database link another_db.world */
EXECUTE DBMS_AQADM.ENABLE_PROPAGATION_SCHEDULE(
   queue_name  => 'q1def',
   destination => 'another_db.world');

/* Unschedule propagation from queue q1def to other queues in another  database
reached by the database link another_db.world */
EXECUTE DBMS_AQADM.UNSCHEDULE_PROPAGATION(
   queue_name  => 'q1def',
   destination => 'another_db.world');
```

伝播スケジュールの解除 : PL/SQL

```
/* Unschedule propagation from msg_queue_multiple to the destination another_
db.world */
EXECUTE DBMS_AQADM.UNSCHEDULE_PROPAGATION(
   queue_name => 'msg_queue_multiple',
   destination => 'another_db.world');
```

参照：

- サンプル・コード : 9-83 ページの「PL/SQL（DBMS_AQADM）: 伝播スケジュールの変更」を参照してください。
- サンプル・コード : 9-86 ページの「PL/SQL（DBMS_AQADM）: 伝播の使用可能化」を参照してください。
- サンプル・コード : 9-89 ページの「PL/SQL（DBMS_AQADM）: 伝播スケジュールの使用不可能化」を参照してくだい。

AQ オブジェクトの削除

注意： キューやキュー表を作成したり、キューを開始、停止または使用可能にしないと、機能しない例もあります。

```
/* Cleans up all objects related to the object type: */
CONNECT aq/aq

EXECUTE DBMS_AQADM.STOP_QUEUE (
   queue_name => 'msg_queue');

EXECUTE DBMS_AQADM.DROP_QUEUE (
   queue_name => 'msg_queue');

EXECUTE DBMS_AQADM.DROP_QUEUE_TABLE (
   queue_table => 'aq.objmsgs80_qtab');

/* Cleans up all objects related to the RAW type: */
EXECUTE DBMS_AQADM.STOP_QUEUE (
   queue_name      => 'raw_msg_queue');

EXECUTE DBMS_AQADM.DROP_QUEUE (
   queue_name      => 'raw_msg_queue');

EXECUTE DBMS_AQADM.DROP_QUEUE_TABLE (
   queue_table => 'aq.RawMsgs_qtab');

/* Cleans up all objects related to the priority queue: */
EXECUTE DBMS_AQADM.STOP_QUEUE (
   queue_name     => 'priority_msg_queue');

EXECUTE DBMS_AQADM.DROP_QUEUE (
   queue_name     => 'priority_msg_queue');
```

```
EXECUTE DBMS_AQADM.DROP_QUEUE_TABLE (
   queue_table   => 'aq.priority_msg');

/* Cleans up all objects related to the multiple-consumer queue: */
EXECUTE DBMS_AQADM.STOP_QUEUE (
   queue_name  => 'msg_queue_multiple');

EXECUTE DBMS_AQADM.DROP_QUEUE (
   queue_name  => 'msg_queue_multiple');

EXECUTE DBMS_AQADM.DROP_QUEUE_TABLE (
   queue_table => 'aq.MultiConsumerMsgs_qtab');

DROP TYPE aq.message_typ;
```

ロールおよび権限の取消し

```
CONNECT sys/change_on_install
DROP USER aq;
```

AQ による XA の使用

注意： 次のようなデータ構造を設定しないと機能しない例もあります。

```
CONNECT system/manager;
DROP USER aqadm CASCADE;
GRANT CONNECT, RESOURCE TO aqadm;
CREATE USER aqadm IDENTIFIED BY aqadm;
GRANT EXECUTE ON DBMS_AQADM TO aqadm;
GRANT Aq_administrator_role TO aqadm;
DROP USER aq CASCADE;
CREATE USER aq IDENTIFIED BY aq;
GRANT CONNECT, RESOURCE TO aq;
GRANT EXECUTE ON dbms_aq TO aq;
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE(
   queue_table => 'aq.qtable',
   queue_payload_type => 'RAW');

EXECUTE DBMS_AQADM.CREATE_QUEUE(
   queue_name => 'aq.aqsqueue',
   queue_table => 'aq.qtable');

EXECUTE DBMS_AQADM.START_QUEUE(queue_name => 'aq.aqsqueue');
```

```
/*
 * The program uses the XA interface to enqueue 100 messages and then
 * dequeue them.
 * Login: aq/aq
 * Requires: AQ_USER_ROLE to be granted to aq
 *        a RAW queue called "aqsqueue" to be created in aqs schema
 *        (above steps can be performed by running aqaq.sql)
 * Message Format: Msgno: [0-1000] HELLO, WORLD!
 * Author: schandra@us.oracle.com
 */

#ifndef OCI_ORACLE
#include <oci.h>
#endif

#include <xa.h>

/* XA open string */
char xaoinfo[] =  "oracle_xa+ACC=P/AQ/AQ+SESTM=30+Objects=T";

/* template for generating XA XIDs */
```

```
XID  xidtempl = { 0x1e0a0a1e, 12, 8, "GTRID001BQual001" };

/* Pointer to Oracle XA function table */
extern struct xa_switch_t xaosw;                          /* Oracle XA switch */
static struct xa_switch_t *xafunc = &xaosw;

/* dummy stubs for ax_reg and ax_unreg */
int ax_reg(rmid, xid, flags)
int  rmid;
XID *xid;
long flags;
{
  xid->formatID = -1;
  return 0;
}

int ax_unreg(rmid, flags)
int     rmid;
long    flags;
{
  return 0;
}

/* generate an XID */
void xidgen(xid, serialno)
XID *xid;
int  serialno;
{
  char seq [11];

  sprintf(seq, "%d", serialno);
  memcpy((void *)xid, (void *)&xidtempl, sizeof(XID));
  strncpy((&xid->data[5]), seq, 3);
}

/* check if XA operation succeeded */
#define checkXAerr(action, funcname)    \
    if ((action) != XA_OK)        \
    {                  \
      printf("%s failed!\n", funcname);   \
      exit(-1);            \
    } else

/* check if OCI operation succeeded */
static void checkOCIerr(errhp, status)
LNOCIError *errhp;
sword      status;
```

```
{
  text errbuf[512];
  ub4 buflen;
  sb4 errcode;

  if (status == OCI_SUCCESS) return;

  if (status == OCI_ERROR)
  {
    OCIErrorGet((dvoid *) errhp, 1, (text *)0, &errcode, errbuf,
      (ub4)sizeof(errbuf), OCI_HTYPE_ERROR);
    printf("Error - %s\n", errbuf);
  }
  else
    printf("Error - %d\n", status);
  exit (-1);
}


void main(argc, argv)
int    argc;
char **argv;
{
  int         msgno = 0;             /* message being enqueued */
  OCIEnv       *envhp;                /* OCI environment handle */
  OCIError     *errhp;                 /* OCI Error handle */
  OCISvcCtx    *svchp;                    /* OCI Service handle */
  char      message[128];               /* message buffer */
  ub4      mesglen;             /* length of message */
  OCIRaw       *rawmesg = (OCIRaw *)0;       /* message in OCI RAW format */
  OCIInd      ind = 0;                /* OCI null indicator */
  dvoid          *indptr = (dvoid *)&ind;       /* null indicator pointer */
  OCIType      *mesg_tdo = (OCIType *) 0;         /* TDO for RAW datatype */
  XID      xid;                 /* XA's global transaction id */
  ub4      i;                      /* array index */

  checkXAerr(xafunc->xa_open_entry(xaoinfo, 1, TMNOFLAGS), "xaoopen");

  svchp = xaoSvcCtx((text *)0);           /* get service handle from XA */
  envhp = xaoEnv((text *)0);          /* get enviornment handle from XA */

  if (!svchp || !envhp)
  {
    printf("Unable to obtain OCI Handles from XA!\n");
    exit (-1);
  }
```

```
OCIHandleAlloc((dvoid *)envhp, (dvoid **)&errhp,
     OCI_HTYPE_ERROR, 0, (dvoid **)0);  /* allocate error handle */

/* enqueue 1000 messages, 1 message per XA transaction */
for (msgno = 0; msgno < 1000; msgno++)
{
  sprintf((const char *)message, "Msgno: %d, Hello, World!", msgno);
  mesglen = (ub4)strlen((const char *)message);
  xidgen(&xid, msgno);                /* generate an XA xid */

  checkXAerr(xafunc->xa_start_entry(&xid, 1, TMNOFLAGS), "xaostart");

  checkOCIerr(errhp, OCIRawAssignBytes(envhp, errhp, (ub1 *)message, mesglen,
              &rawmesg));

  if (!mesg_tdo)          /* get Type descriptor (TDO) for RAW type */
    checkOCIerr(errhp, OCITypeByName(envhp, errhp, svchp,
                    (CONST text *)"AQADM", strlen("AQADM"),
                     (CONST text *)"RAW", strlen("RAW"),
                (text *)0, 0, OCI_DURATION_SESSION,
                OCI_TYPEGET_ALL, &mesg_tdo));

  checkOCIerr(errhp, OCIAQEnq(svchp, errhp, (CONST text *)"aqsqueue",
            0, 0, mesg_tdo, (dvoid **)&rawmesg, &indptr,
          0, 0));

  checkXAerr(xafunc->xa_end_entry(&xid, 1, TMSUCCESS), "xaoend");
  checkXAerr(xafunc->xa_commit_entry(&xid, 1, TMONEPHASE), "xaocommit");
  printf("%s Enqueued\n", message);
}

/* dequeue 1000 messages within one XA transaction */
xidgen(&xid, msgno);                          /* generate an XA xid */
checkXAerr(xafunc->xa_start_entry(&xid, 1, TMNOFLAGS), "xaostart");
for (msgno = 0; msgno < 1000; msgno++)
{
  checkOCIerr(errhp, OCIAQDeq(svchp, errhp, (CONST text *)"aqsqueue",
           0, 0, mesg_tdo, (dvoid **)&rawmesg, &indptr,
          0, 0));
  if (ind)
    printf("Null Raw Message");
  else
    for (i = 0; i < OCIRawSize(envhp, rawmesg); i++)
 printf("%c", *(OCIRawPtr(envhp, rawmesg) + i));
  printf("\n");

}
```

```
  checkXAerr(xafunc->xa_end_entry(&xid, 1, TMSUCCESS), "xaoend");
  checkXAerr(xafunc->xa_commit_entry(&xid, 1, TMONEPHASE), "xaocommit");
}
```

AQ およびメモリーの使用

Creat_types.sql: Scott のスキーマへのペイロード型およびキューの作成

注意： 次のような SQL 文を実行しないと機能しない例もあります。

```
/* Create_types.sql */
CONNECT system/manager
GRANT AQ_ADMINISTRATOR_ROLE, AQ_USER_ROLE TO scott;
CONNECT scott/tiger
CREATE TYPE MESSAGE AS OBJECT (id NUMBER, data VARCHAR2(80));
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE(
   queue_table        => 'qt',
   queue_payload_type => 'message');
EXECUTE DBMS_AQADM.CREATE_QUEUE('msgqueue', 'qt');
EXECUTE DBMS_AQADM.START_QUEUE('msgqueue');
```

メッセージのエンキュー（各コール後のメモリー解放）: OCI

このプログラム enqnoreuse.c は、前述の create_types.sql を介して scott のスキーマに作成されたキュー msgqueue から、テキストの各行をデキューします。メッセージは、enqnoreuse.c または enqreuse.c を使用してエンキューされます（次の例を参照）。メッセージが存在しない場合は、60 秒待機してからタイムアウトします。このプログラムでは、デキュー・サブルーチンはクライアント側のオブジェクトのメモリーを再利用しません。必要なメモリーをデキューの前に割り当て、デキューの完了後にそのメモリーを解放します。

```
#ifndef OCI_ORACLE
#include <oci.h>
#endif

#include <stdio.h>

static void checkerr(OCIError *errhp, sword status);
static void deqmesg(text *buf, ub4 *buflen);

LNOCIEnv       *envhp;
LNOCIError     *errhp;
LNOCISvcCtx    *svchp;
```

```
struct message
{
  OCINumber    id;
  OCIString   *data;
};
typedef struct message message;

struct null_message
{
  OCIInd    null_adt;
  OCIInd    null_id;
  OCIInd    null_data;
};
typedef struct null_message null_message;

static void deqmesg(buf, buflen)
text  *buf;
ub4   *buflen;
{
  OCIType         *mesgtdo = (OCIType *)0;   /* type descr of SCOTT.MESSAGE */
  message         *mesg    = (dvoid *)0;     /* instance of SCOTT.MESSAGE */
  null_message    *mesgind = (dvoid *)0;     /* null indicator */
  OCIAQDeqOptions *deqopt  = (OCIAQDeqOptions *)0;
  ub4              wait    = 60;             /* timeout after 60 seconds */
  ub4              navigation = OCI_DEQ_FIRST_MSG;/* always get head of q */


  /* Get the type descriptor object for the type SCOTT.MESSAGE: */
  checkerr(errhp, OCITypeByName(envhp, errhp, svchp,
           (CONST text *)"SCOTT", strlen("SCOTT"),
           (CONST text *)"MESSAGE", strlen("MESSAGE"),
           (text *)0, 0, OCI_DURATION_SESSION,
           OCI_TYPEGET_ALL, &mesgtdo));

  /* Allocate an instance of SCOTT.MESSAGE, and get its null indicator: */
  checkerr(errhp, OCIObjectNew(envhp, errhp, svchp, OCI_TYPECODE_OBJECT,
           mesgtdo, (dvoid *)0, OCI_DURATION_SESSION,
           TRUE, (dvoid **)&mesg));
  checkerr(errhp, OCIObjectGetInd(envhp, errhp, (dvoid *)mesg,
           (dvoid **)&mesgind));

  /* Allocate a descriptor for dequeue options and set wait time, navigation: */
  checkerr(errhp, OCIDescriptorAlloc(envhp, (dvoid **)&deqopt,
           OCI_DTYPE_AQDEQ_OPTIONS, 0, (dvoid **)0));
  checkerr(errhp, OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS,
           (dvoid *)&wait, 0, OCI_ATTR_WAIT, errhp));
  checkerr(errhp, OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS,
```

```
            (dvoid *)&navigation, 0,
            OCI_ATTR_NAVIGATION, errhp));

  /* Dequeue the message and commit: */
  checkerr(errhp, OCIAQDeq(svchp, errhp, (CONST text *)"msgqueue",
           deqopt, 0, mesgtdo, (dvoid **)&mesg,
           (dvoid **)&mesgind, 0, 0));

  checkerr(errhp, OCITransCommit(svchp, errhp, (ub4) 0));

  /* Copy the message payload text into the user buffer: */
  if (mesgind->null_data)
    *buflen = 0;
  else
    memcpy((dvoid *)buf, (dvoid *)OCIStringPtr(envhp, mesg->data),
           (size_t)(*buflen = OCIStringSize(envhp, mesg->data)));

  /* Free the dequeue options descriptor: */
  checkerr(errhp, OCIDescriptorFree((dvoid *)deqopt, OCI_DTYPE_AQDEQ_OPTIONS));

  /* Free the memory for the objects: */
  Checkerr(errhp, OCIObjectFree(envhp, errhp, (dvoid *)mesg,
           OCI_OBJECTFREE_FORCE));
}                          /* end deqmesg */

void main()
{
  OCIServer     *srvhp;
  OCISession    *usrhp;
  dvoid         *tmp;
  text           buf[80];                /* payload text */
  ub4            buflen;

  OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
                (dvoid * (*)()) 0,  (void (*)()) 0 );

  OCIHandleAlloc((dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
                 52, (dvoid **) &tmp);

  OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 21, (dvoid **) &tmp  );

  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
                 52, (dvoid **) &tmp);
  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
                 52, (dvoid **) &tmp);

  OCIServerAttach(srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);
```

```
  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
                 52, (dvoid **) &tmp);

  /* Set attribute server context in the service context: */
  OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
             (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

  /* Allocate a user context handle: */
  OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
                 (size_t) 0, (dvoid **) 0);

  OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
             (dvoid *)"scott", (ub4)strlen("scott"), OCI_ATTR_USERNAME, errhp);

  OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
             (dvoid *)"tiger", (ub4)strlen("tiger"), OCI_ATTR_PASSWORD, errhp);

  checkerr(errhp, OCISessionBegin (svchp, errhp, usrhp, OCI_CRED_RDBMS,
           OCI_DEFAULT));

  OCIAttrSet((dvoid *)svchp, (ub4)OCI_HTYPE_SVCCTX,
             (dvoid *)usrhp, (ub4)0, OCI_ATTR_SESSION, errhp);

  do {
    deqmesg(buf, &buflen);
    printf("%.*s\n", buflen, buf);
  } while(1);
}                          /* end main */

static void checkerr(errhp, status)
LNOCIError  *errhp;
sword      status;
{
  text errbuf[512];
  ub4  buflen;
  sb4 errcode;

  if (status == OCI_SUCCESS) return;

  switch (status)
  {
  case OCI_ERROR:
    OCIErrorGet ((dvoid *) errhp, (ub4) 1, (text *) NULL, &errcode,
                 errbuf, (ub4) sizeof(errbuf), (ub4) OCI_HTYPE_ERROR);
    printf("Error - %s\n", errbuf);
    break;
```

```
  case OCI_INVALID_HANDLE:
    printf("Error - OCI_INVALID_HANDLE\n");
    break;
  default:
    printf("Error - %d\n", status);
    break;
  }
  exit(-1);
}                          /* end checkerr */
```

メッセージのエンキュー（メモリーの再利用）: OCI

このプログラム enqreuse.c は、create_types.sql を実行して scott のスキーマに作成されたキュー msgqueue に、テキストの各行をエンキューします。ユーザーが入力するテキストの各行は、ユーザーが EOF を入力するまでキューの中に格納されます。このプログラムでは、エンキュー・サブルーチンは AQ メッセージ・プロパティ記述子のみでなくメッセージ・ペイロード用のメモリーも再利用します。

```
#ifndef OCI_ORACLE
#include <oci.h>
#endif

#include <stdio.h>

static void checkerr(OCIError *errhp, sword status);
static void enqmesg(ub4 msgno, text *buf);

struct message
{
  OCINumber     id;
  OCIString    *data;
};
typedef struct message message;

struct null_message
{
  OCIInd    null_adt;
  OCIInd    null_id;
  OCIInd    null_data;
};
typedef struct null_message null_message;

/* Global data reused on calls to enqueue: */
LNOCIEnv             *envhp;
LNOCIError           *errhp;
LNOCISvcCtx          *svchp;
message             msg;
```

```
null_message        nmsg;
LNOCIAQMsgProperties *msgprop;

static void enqmesg(msgno, buf)
ub4     msgno;
text   *buf;
{
  OCIType          *mesgtdo = (OCIType *)0; /* type descr of SCOTT.MESSAGE */
  message          *mesg = &msg;            /* instance of SCOTT.MESSAGE */
  null_message     *mesgind = &nmsg;        /* null indicator */
  text              corrid[128];            /* correlation identifier */

  /* Get the type descriptor object for the type SCOTT.MESSAGE: */
  checkerr(errhp, OCITypeByName(envhp, errhp, svchp,
           (CONST text *)"SCOTT", strlen("SCOTT"),
           (CONST text *)"MESSAGE", strlen("MESSAGE"),
           (text *)0, 0, OCI_DURATION_SESSION,
           OCI_TYPEGET_ALL, &mesgtdo));

  /* Fill in the attributes of SCOTT.MESSAGE: */
  checkerr(errhp, OCINumberFromInt(errhp, &msgno, sizeof(ub4), 0, &mesg->id));
  checkerr(errhp, OCIStringAssignText(envhp, errhp, buf, strlen(buf),
           &mesg->data));
  mesgind->null_adt = mesgind->null_id = mesgind->null_data = 0;

  /* Set the correlation id in the message properties descriptor: */
  sprintf((char *)corrid, "Msg#: %d", msgno);
  checkerr(errhp, OCIAttrSet(msgprop, OCI_DTYPE_AQMSG_PROPERTIES,
           (dvoid *)&corrid, strlen(corrid),
           OCI_ATTR_CORRELATION, errhp));

  /* Enqueue the message and commit: */
  checkerr(errhp, OCIAQEnq(svchp, errhp, (CONST text *)"msgqueue",
           0, msgprop, mesgtdo, (dvoid **)&mesg,
           (dvoid **)&mesgind, 0, 0));

  checkerr(errhp, OCITransCommit(svchp, errhp, (ub4) 0));
}                           /* end enqmesg */

void main()
{
  OCIServer    *srvhp;
  OCISession   *usrhp;
  dvoid        *tmp;
  text          buf[80];               /* user supplied text */
  int           msgno = 0;
```

```
OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
              (dvoid * (*)()) 0,  (void (*)()) 0 );

OCIHandleAlloc((dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
               52, (dvoid **) &tmp);

OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 21, (dvoid **) &tmp  );

OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
               52, (dvoid **) &tmp);
OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
               52, (dvoid **) &tmp);

OCIServerAttach(srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
               52, (dvoid **) &tmp);

/* Set attribute server context in the service context: */
OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
           (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

/* Allocate a user context handle: */
OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
               (size_t) 0, (dvoid **) 0);

OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
           (dvoid *)"scott", (ub4)strlen("scott"), OCI_ATTR_USERNAME, errhp);

OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
           (dvoid *)"tiger", (ub4)strlen("tiger"), OCI_ATTR_PASSWORD, errhp);

checkerr(errhp, OCISessionBegin (svchp, errhp, usrhp, OCI_CRED_RDBMS,
         OCI_DEFAULT));

OCIAttrSet((dvoid *)svchp, (ub4)OCI_HTYPE_SVCCTX,
           (dvoid *)usrhp, (ub4)0, OCI_ATTR_SESSION, errhp);

/* Allocate a message properties descriptor to fill in correlation id :*/
checkerr(errhp, OCIDescriptorAlloc(envhp, (dvoid **)&msgprop,
         OCI_DTYPE_AQMSG_PROPERTIES,
         0, (dvoid **)0));
do {
  printf("Enter a line of text (max 80 chars):");
  if (!gets((char *)buf))
    break;
  enqmesg((ub4)msgno++, buf);
```

```
  } while(1);

  /* Free the message properties descriptor: */
  checkerr(errhp, OCIDescriptorFree((dvoid *)msgprop,
           OCI_DTYPE_AQMSG_PROPERTIES));

}                          /* end main */

static void checkerr(errhp, status)
LNOCIError   *errhp;
sword      status;
{
  text errbuf[512];
  ub4  buflen;
  sb4  errcode;

  if (status == OCI_SUCCESS) return;

  switch (status)
  {
  case OCI_ERROR:
    OCIErrorGet ((dvoid *) errhp, (ub4) 1, (text *) NULL, &errcode,
                 errbuf, (ub4) sizeof(errbuf), (ub4) OCI_HTYPE_ERROR);
    printf("Error - %s\n", errbuf);
    break;
  case OCI_INVALID_HANDLE:
    printf("Error - OCI_INVALID_HANDLE\n");
    break;
  default:
    printf("Error - %d\n", status);
    break;
  }
  exit(-1);
}                           /* end checkerr */
```

メッセージのデキュー（各コール後のメモリー解放）: OCI

このプログラム deqnoreuse.c は、create_types.sql を実行して scott のスキーマに作成されたキュー msgqueue から、テキストの各行をデキューします。メッセージは、enqnoreuse または enqreuse を使用してエンキューされます。メッセージが存在しない場合は、60 秒待機してからタイムアウトします。このプログラムでは、デキュー・サブルーチンはクライアント側のオブジェクトのメモリーを再利用しません。必要なメモリーをデキューの前に割り当て、デキューの完了後にそのメモリーを解放します。

```
#ifndef OCI_ORACLE
#include <oci.h>
#endif
```

```
#include <stdio.h>

static void checkerr(OCIError *errhp, sword status);
static void deqmesg(text *buf, ub4 *buflen);

LNOCIEnv       *envhp;
LNOCIError     *errhp;
LNOCISvcCtx    *svchp;

struct message
{
  OCINumber     id;
  OCIString    *data;
};
typedef struct message message;

struct null_message
{
  OCIInd    null_adt;
  OCIInd    null_id;
  OCIInd    null_data;
};
typedef struct null_message null_message;

static void deqmesg(buf, buflen)
text      *buf;
ub4       *buflen;
{
  OCIType          *mesgtdo = (OCIType *)0;  /* type descr of SCOTT.MESSAGE */
  message          *mesg = (dvoid *)0;       /* instance of SCOTT.MESSAGE */
  null_message     *mesgind   = (dvoid *)0;       /* null indicator */
  OCIAQDeqOptions  *deqopt    = (OCIAQDeqOptions *)0;
  ub4               wait      = 60;               /* timeout after 60 seconds */
  ub4              navigation = OCI_DEQ_FIRST_MSG;/* always get head of q */

   /* Get the type descriptor object for the type SCOTT.MESSAGE: */
  checkerr(errhp, OCITypeByName(envhp, errhp, svchp,
           (CONST text *)"SCOTT", strlen("SCOTT"),
           (CONST text *)"MESSAGE", strlen("MESSAGE"),
           (text *)0, 0, OCI_DURATION_SESSION,
           OCI_TYPEGET_ALL, &mesgtdo));

  /* Allocate an instance of SCOTT.MESSAGE, and get its null indicator: */
  checkerr(errhp, OCIObjectNew(envhp, errhp, svchp, OCI_TYPECODE_OBJECT,
           mesgtdo, (dvoid *)0, OCI_DURATION_SESSION,
           TRUE, (dvoid **)&mesg));
```

```
  checkerr(errhp, OCIObjectGetInd(envhp, errhp, (dvoid *)mesg,
           (dvoid **)&mesgind));

  /* Allocate a descriptor for dequeue options and set wait time, navigation: */
  checkerr(errhp, OCIDescriptorAlloc(envhp, (dvoid **)&deqopt,
           OCI_DTYPE_AQDEQ_OPTIONS, 0, (dvoid **)0));
  checkerr(errhp, OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS,
           (dvoid *)&wait, 0, OCI_ATTR_WAIT, errhp));
  checkerr(errhp, OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS,
           (dvoid *)&navigation, 0,
           OCI_ATTR_NAVIGATION, errhp));

  /* Dequeue the message and commit: */
  checkerr(errhp, OCIAQDeq(svchp, errhp, (CONST text *)"msgqueue",
           deqopt, 0, mesgtdo, (dvoid **)&mesg,
           (dvoid **)&mesgind, 0, 0));

  checkerr(errhp, OCITransCommit(svchp, errhp, (ub4) 0));

  /* Copy the message payload text into the user buffer: */
  if (mesgind->null_data)
    *buflen = 0;
  else
    memcpy((dvoid *)buf, (dvoid *)OCIStringPtr(envhp, mesg->data),
           (size_t)(*buflen = OCIStringSize(envhp, mesg->data)));

  /* Free the dequeue options descriptor: */
  checkerr(errhp, OCIDescriptorFree((dvoid *)deqopt, OCI_DTYPE_AQDEQ_OPTIONS));

  /* Free the memory for the objects: */
  checkerr(errhp, OCIObjectFree(envhp, errhp, (dvoid *)mesg,
           OCI_OBJECTFREE_FORCE));
}                                       /* end deqmesg */

void main()
{
  OCIServer    *srvhp;
  OCISession   *usrhp;
  dvoid        *tmp;
  text          buf[80];                /* payload text */
  ub4           buflen;

  OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
                (dvoid * (*)()) 0,  (void (*)()) 0 );

  OCIHandleAlloc((dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
                 52, (dvoid **) &tmp);
```

```
  OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 21, (dvoid **) &tmp  );

  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
                 52, (dvoid **) &tmp);
  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
                 52, (dvoid **) &tmp);

  OCIServerAttach(srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
                 52, (dvoid **) &tmp);

  /* Set attribute server context in the service context: */
  OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
             (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

  /* Allocate a user context handle: */
  OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
                 (size_t) 0, (dvoid **) 0);

  OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
             (dvoid *)"scott", (ub4)strlen("scott"), OCI_ATTR_USERNAME, errhp);

  OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
             (dvoid *)"tiger", (ub4)strlen("tiger"), OCI_ATTR_PASSWORD, errhp);

  checkerr(errhp, OCISessionBegin (svchp, errhp, usrhp, OCI_CRED_RDBMS,
           OCI_DEFAULT));

  OCIAttrSet((dvoid *)svchp, (ub4)OCI_HTYPE_SVCCTX,
             (dvoid *)usrhp, (ub4)0, OCI_ATTR_SESSION, errhp);

  do {
    deqmesg(buf, &buflen);
    printf("%.*s\n", buflen, buf);
  } while(1);
}                          /* end main */

static void checkerr(errhp, status)
LNOCIError   *errhp;
sword      status;
{
  text errbuf[512];
  ub4  buflen;
  sb4  errcode;
```

```
  if (status == OCI_SUCCESS) return;

  switch (status)
  {
  case OCI_ERROR:
    OCIErrorGet ((dvoid *) errhp, (ub4) 1, (text *) NULL, &errcode,
                 errbuf, (ub4) sizeof(errbuf), (ub4) OCI_HTYPE_ERROR);
    printf("Error - %s\n", errbuf);
    break;
  case OCI_INVALID_HANDLE:
    printf("Error - OCI_INVALID_HANDLE\n");
    break;
  default:
    printf("Error - %d\n", status);
    break;
  }
  exit(-1);
}                          /* end checkerr */
```

メッセージのデキュー（メモリーの再利用）: OCI

このプログラム deqreuse.c は、create_types.sql を実行して scott のスキーマに作成されたキュー msgqueue から、テキストの各行をデキューします。メッセージは、enqnoreuse.c または enqreuse.c を使用してエンキューされます。メッセージが存在しない場合は、60 秒待機してからタイムアウトします。このプログラムでは、デキュー・サブルーチンは、LNOCIAQDeq の起動停止から再起動までの間、クライアント側のオブジェクトのメモリーを再利用します。LNOCIAQDeq に対する最初のコール中に、OCI はメッセージ・ペイロード用のメモリーを自動的に割り当てます。LNOCIAQDeq に対する後続のコール中に、同一のペイロード・ポインタが渡され、OCI は必要に応じてペイロード・メモリーのサイズを自動的に変更します。

```
#ifndef OCI_ORACLE
#include <oci.h>
#endif

#include <stdio.h>

static void checkerr(OCIError *errhp, sword status);
static void deqmesg(text *buf, ub4 *buflen);

struct message
{
  OCINumber    id;
  OCIString   *data;
};
typedef struct message message;
```

```
struct null_message
{
  OCIInd    null_adt;
  OCIInd    null_id;
  OCIInd    null_data;
};
typedef struct null_message null_message;

/* Global data reused on calls to enqueue: */
LNOCIEnv          *envhp;
LNOCIError        *errhp;
LNOCISvcCtx       *svchp;
LNOCIAQDeqOptions *deqopt;
message         *mesg = (message *)0;
null_message    *mesgind = (null_message *)0;

static void deqmesg(buf, buflen)
text       *buf;
ub4        *buflen;
{

  OCIType        *mesgtdo    = (OCIType *)0; /* type descr of SCOTT.MESSAGE */
  ub4             wait       = 60;           /* timeout after 60 seconds */
  ub4             navigation = OCI_DEQ_FIRST_MSG;/* always get head of q */

  /* Get the type descriptor object for the type SCOTT.MESSAGE: */
  checkerr(errhp, OCITypeByName(envhp, errhp, svchp,
           (CONST text *)"SCOTT", strlen("SCOTT"),
           (CONST text *)"MESSAGE", strlen("MESSAGE"),
           (text *)0, 0, OCI_DURATION_SESSION,
           OCI_TYPEGET_ALL, &mesgtdo));

  /* Set wait time, navigation in dequeue options: */
  checkerr(errhp, OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS,
           (dvoid *)&wait, 0, OCI_ATTR_WAIT, errhp));
  checkerr(errhp, OCIAttrSet(deqopt, OCI_DTYPE_AQDEQ_OPTIONS,
           (dvoid *)&navigation, 0,
           OCI_ATTR_NAVIGATION, errhp));

  /*
   * Dequeue the message and commit. The memory for the payload will be
   * automatically allocated/resized by OCI:
   */
  checkerr(errhp, OCIAQDeq(svchp, errhp, (CONST text *)"msgqueue",
           deqopt, 0, mesgtdo, (dvoid **)&mesg,
           (dvoid **)&mesgind, 0, 0));
```

```
  checkerr(errhp, OCITransCommit(svchp, errhp, (ub4) 0));

  /* Copy the message payload text into the user buffer: */
  if (mesgind->null_data)
    *buflen = 0;
  else
    memcpy((dvoid *)buf, (dvoid *)OCIStringPtr(envhp, mesg->data),
           (size_t)(*buflen = OCIStringSize(envhp, mesg->data)));
}                           /* end deqmesg */

void main()
{
  OCIServer    *srvhp;
  OCISession   *usrhp;
  dvoid        *tmp;
  text          buf[80];                /* payload text */
  ub4           buflen;

  OCIInitialize((ub4) OCI_OBJECT, (dvoid *)0,  (dvoid * (*)()) 0,
                (dvoid * (*)()) 0,  (void (*)()) 0 );

  OCIHandleAlloc((dvoid *) NULL, (dvoid **) &envhp, (ub4) OCI_HTYPE_ENV,
                 52, (dvoid **) &tmp);

  OCIEnvInit( &envhp, (ub4) OCI_DEFAULT, 21, (dvoid **) &tmp  );

  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, (ub4) OCI_HTYPE_ERROR,
                 52, (dvoid **) &tmp);
  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, (ub4) OCI_HTYPE_SERVER,
                 52, (dvoid **) &tmp);

  OCIServerAttach(srvhp, errhp, (text *) 0, (sb4) 0, (ub4) OCI_DEFAULT);

  OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, (ub4) OCI_HTYPE_SVCCTX,
                 52, (dvoid **) &tmp);

  /* set attribute server context in the service context */
  OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *)srvhp, (ub4) 0,
             (ub4) OCI_ATTR_SERVER, (OCIError *) errhp);

  /* allocate a user context handle */
  OCIHandleAlloc((dvoid *)envhp, (dvoid **)&usrhp, (ub4) OCI_HTYPE_SESSION,
                 (size_t) 0, (dvoid **) 0);

  OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
             (dvoid *)"scott", (ub4)strlen("scott"), OCI_ATTR_USERNAME, errhp);
```

```
  OCIAttrSet((dvoid *)usrhp, (ub4)OCI_HTYPE_SESSION,
             (dvoid *)"tiger", (ub4)strlen("tiger"), OCI_ATTR_PASSWORD, errhp);

  checkerr(errhp, OCISessionBegin (svchp, errhp, usrhp, OCI_CRED_RDBMS,
           OCI_DEFAULT));

  OCIAttrSet((dvoid *)svchp, (ub4)OCI_HTYPE_SVCCTX,
             (dvoid *)usrhp, (ub4)0, OCI_ATTR_SESSION, errhp);

  /* allocate the dequeue options descriptor */
  checkerr(errhp, OCIDescriptorAlloc(envhp, (dvoid **)&deqopt,
           OCI_DTYPE_AQDEQ_OPTIONS, 0, (dvoid **)0));

  do {
    deqmesg(buf, &buflen);
    printf("%.*s\n", buflen, buf);
  } while(1);

  /*
   * This program never reaches this point as the dequeue timesout & exits.
   * If it does reach here, it will be a good place to free the dequeue
   * options descriptor using OCIDescriptorFree and free the memory allocated
   * by OCI for the payload using OCIObjectFree
   */
}                          /* end main */

static void checkerr(errhp, status)
LNOCIError *errhp;
sword      status;
{
  text errbuf[512];
  ub4  buflen;
  sb4  errcode;

  if (status == OCI_SUCCESS) return;

  switch (status)
  {
  case OCI_ERROR:
    OCIErrorGet ((dvoid *) errhp, (ub4) 1, (text *) NULL, &errcode,
                 errbuf, (ub4) sizeof(errbuf), (ub4) OCI_HTYPE_ERROR);
    printf("Error - %s\n", errbuf);
    break;
  case OCI_INVALID_HANDLE:
    printf("Error - OCI_INVALID_HANDLE\n");
    break;
  default:
```

```
    printf("Error - %d\n", status);
    break;
  }
  exit(-1);
}                          /* end checkerr */
```

B

Oracle JMS インタフェース、クラスおよび例外

この付録では、表 B-1 に示す JMS インタフェース、クラスおよび例外について説明します。

表 B-1 インタフェース、クラスおよび例外

インタフェース / クラス / 例外
Oracle JMS クラス（パート 1）（B-6 ページ）
Oracle JMS クラス（パート 2）（B-8 ページ）
Oracle JMS クラス（パート 3）（B-9 ページ）
Oracle JMS クラス（パート 4）（B-10 ページ）
Oracle JMS クラス（パート 5）（B-11 ページ）
Oracle JMS クラス（パート 6）（B-12 ページ）
Oracle JMS クラス（パート 7）（B-15 ページ）
Oracle JMS クラス（パート 8）（B-17 ページ）
Oracle JMS クラス（パート 9）（B-19 ページ）
Oracle JMS クラス（パート 10）（B-21 ページ）
インタフェース - javax.jms.BytesMessage（B-24 ページ）
インタフェース - javax.jms.Connection（B-25 ページ）
インタフェース - javax.jms.ConnectionFactory（B-26 ページ）
インタフェース - javax.jms.ConnectionMetaData（B-26 ページ）
インタフェース - javax.jms.DeliveryMode（B-27 ページ）
インタフェース - javax.jms.Destination（B-27 ページ）
インタフェース - javax.jms.MapMessage（B-27 ページ）
インタフェース - javax.jms.Message（B-28 ページ）
インタフェース - javax.jms.MessageConsumer（B-30 ページ）
インタフェース - javax.jms.MessageListener（B-31 ページ）
インタフェース - javax.jms.MessageProducer（B-31 ページ）
インタフェース - javax.jms.ObjectMessage（B-32 ページ）
インタフェース - javax.jms.Queue（B-32 ページ）
インタフェース - javax.jms.QueueBrowser（B-32 ページ）
インタフェース - javax.jms.QueueConnection（B-33 ページ）
インタフェース - javax.jms.QueueConnectionFactory（B-33 ページ）

表 B-1 インタフェース、クラスおよび例外（続き）

インタフェース / クラス / 例外
インタフェース - javax.jms.QueueReceiver（B-34 ページ）
インタフェース - javax.jms.QueueSender（B-34 ページ）
インタフェース - javax.jms.QueueSession（B-35 ページ）
インタフェース - javax.jms.Session（B-35 ページ）
インタフェース - javax.jms.StreamMessage（B-37 ページ）
インタフェース - javax.jms.TextMessage（B-38 ページ）
インタフェース - javax.jms.Topic（B-38 ページ）
インタフェース - javax.jms.TopicConnection（B-38 ページ）
インタフェース - javax.jms.TopicConnectionFactory（B-39 ページ）
インタフェース - javax.jms.TopicPublisher（B-39 ページ）
インタフェース - javax.jms.TopicSession（B-40 ページ）
インタフェース - javax.jms.TopicSubscriber（B-40 ページ）
例外 - javax.jms.InvalidDestinationException（B-41 ページ）
例外 - javax.jms.InvalidSelectorException（B-41 ページ）
例外 - javax.jms.JMSException（B-42 ページ）
例外 - javax.jms.MessageEOFException（B-42 ページ）
例外 - javax.jms.MessageFormatException（B-43 ページ）
例外 - javax.jms.MessageNotReadableException（B-43 ページ）
例外 - javax.jms.MesageNotWriteableException（B-43 ページ）
インタフェース - oracle.jms.AdtMessage（B-44 ページ）
インタフェース - oracle.jms.AQjmsQueueReceiver（B-44 ページ）
インタフェース - oracle.jms.AQjmsQueueSender（B-44 ページ）
インタフェース - oracle.jms.AQjmsTopicPublisher（B-45 ページ）
インタフェース - oracle.jms.TopicReceiver（B-45 ページ）
インタフェース - oracle.jms.AQjmsTopicSubscriber（B-45 ページ）
インタフェース - oracle.jms.AQjmsTopicReceiver（B-46 ページ）

表 B-1 インタフェース、クラスおよび例外（続き）

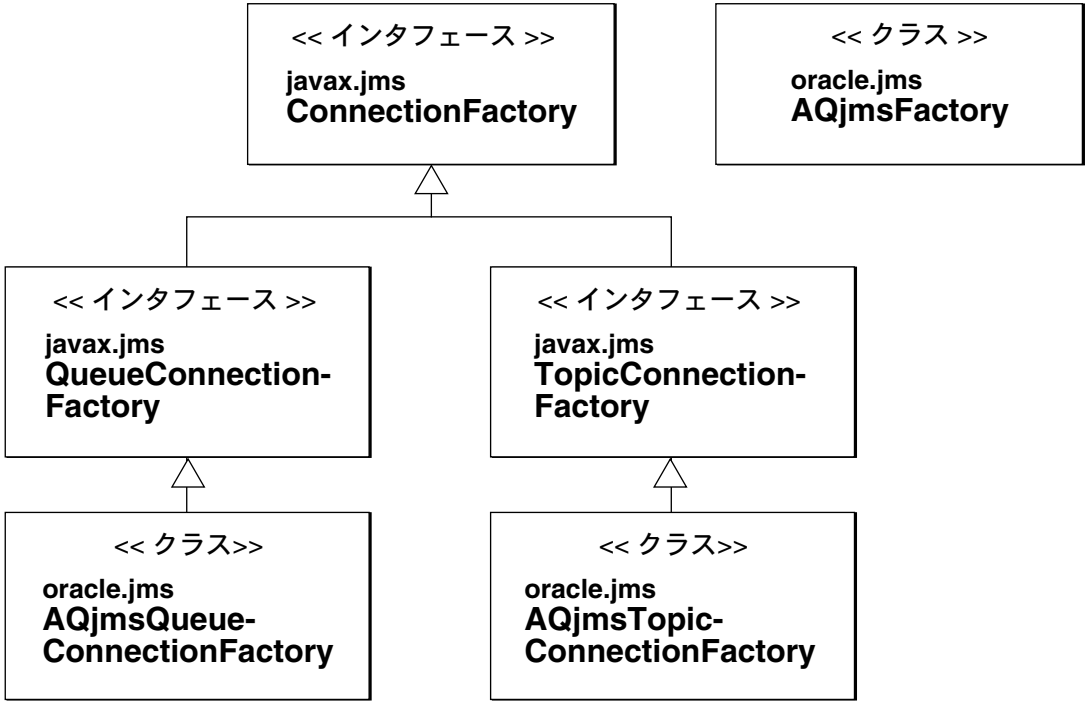
インタフェース / クラス / 例外
クラス - oracle.jms.AQjmsAdtMessage（B-46 ページ）
クラス - oracle.jms.AQjmsAgent（B-46 ページ）
クラス - oracle.jms.AQjmsBytesMessage（B-47 ページ）
クラス - oracle.jms.AQjmsConnection（B-47 ページ）
インタフェース - oracle.jms.AQjmsConnectionMetadata（B-47 ページ）
クラス - oracle.jms.AQjmsConstants（B-48 ページ）
インタフェース - oracle.jms.AQjmsConsumer（B-48 ページ）
クラス - oracle.jms.AQjmsDestination（B-49 ページ）
クラス - oracle.jms.AQjmsDestinationProperty（B-50 ページ）
クラス - oracle.jms.AQjmsFactory（B-51 ページ）
クラス - oracle.jms.AQjmsMapMessage（B-52 ページ）
クラス - oracle.jms.AQjmsMessage（B-52 ページ）
クラス - oracle.jms.AQjmsObjectMessage（B-53 ページ）
クラス - oracle.jms.AQjmsOracleDebug（B-53 ページ）
クラス - oracle.jms.AQjmsProducer（B-53 ページ）
クラス - oracle.jms.AQjmsQueueBrowser（B-54 ページ）
クラス - oracle.jms.AQjmsQueueConnectionFactory（B-54 ページ）
クラス - oracle.jms.AQjmsSession（B-54 ページ）
クラス - oracle.jms.AQjmsStreamMessage（B-57 ページ）
クラス - oracle.jms.AQjmsTextMessage（B-57 ページ）
クラス - oracle.jms.AQjmsTopicConnectionFactory（B-57 ページ）
例外 - oracle.jms.AQjmsInvalidDestinationException（B-58 ページ）
例外 - oracle.jms.AQjmsInvalidSelectorException（B-59 ページ）
例外 - oracle.jms.AQjmsMessageEOFException（B-59 ページ）
例外 - oracle.jms.AQjmsMessageFormatException（B-59 ページ）
例外 - oracle.jms.AQjmsMessageNotReadableException（B-59 ページ）

表 B-1 インタフェース、クラスおよび例外（続き）

インタフェース / クラス / 例外
例外 - oracle.jms.AQjmsMesssageNotWriteableException（B-59 ページ）
インタフェース - oracle.AQ.AQQueueTable（B-60 ページ）
クラス - oracle.AQ.AQQueueTableProperty（B-60 ページ）
インタフェース - oracle.jms.TopicBrowser（B-61 ページ）
クラス - oracle.jms.AQjmsTopicBrowser（B-62 ページ）

Oracle JMS クラス（パート 1）

図 B-1 クラス図 : Oracle クラスのクラス（パート 1）

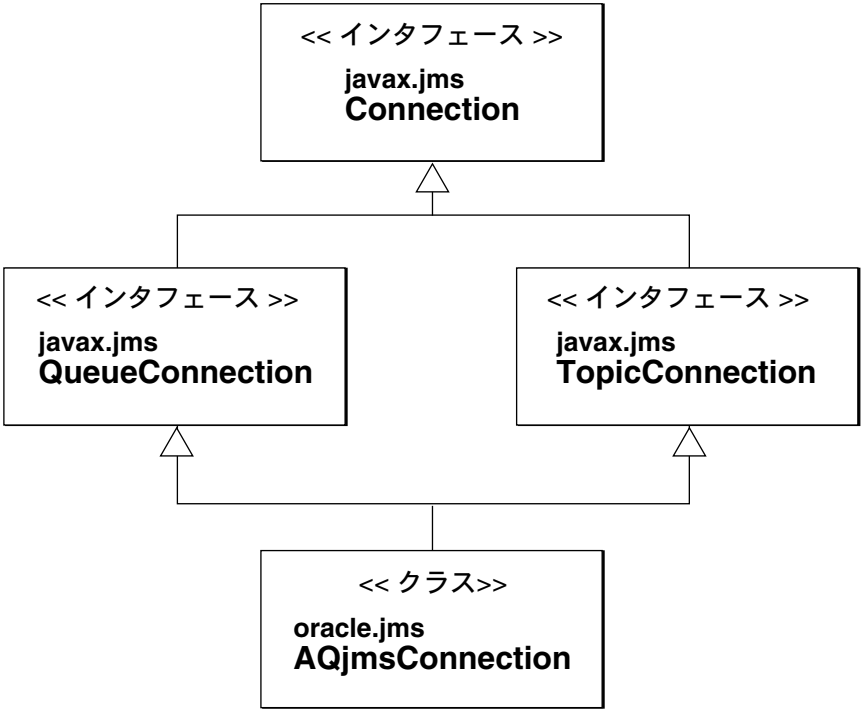


参照：

- B-26 ページの「インタフェース - javax.jms.ConnectionFactory」を参照してください。
- B-51 ページの「クラス - oracle.jms.AQjmsFactory」も参照してください。
- B-33 ページの「インタフェース - javax.jms.QueueConnectionFactory」も参照してください。
- B-39 ページの「インタフェース - javax.jms.TopicConnectionFactory」も参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsQueueConnectionFactory」も参照してください。
- B-57 ページの「クラス - oracle.jms.AQjmsTopicConnectionFactory」も参照してください。

Oracle JMS クラス（パート 2）

図 B-2 クラス図 : Oracle クラスのクラス（パート 2）

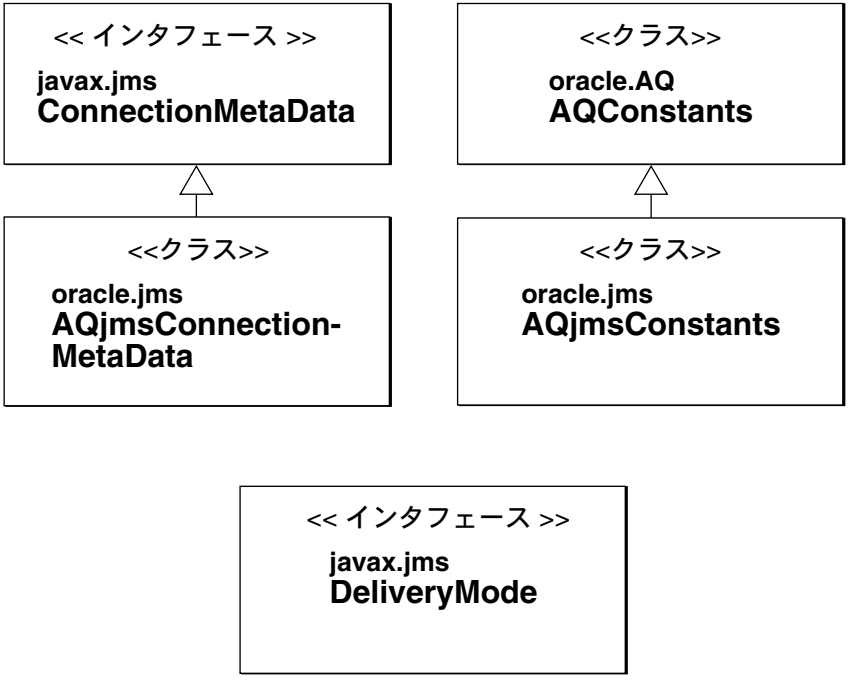


参照 :

- B-25 ページの「インタフェース - javax.jms.Connection」を参照してください。
- B-33 ページの「インタフェース - javax.jms.QueueConnection」も参照してください。
- B-38 ページの「インタフェース - javax.jms.TopicConnection」も参照してください。
- B-47 ページの「クラス - oracle.jms.AQjmsConnection」も参照してください。

Oracle JMS クラス（パート 3）

図 B-3 クラス図 : Oracle クラスのクラス（パート 3）

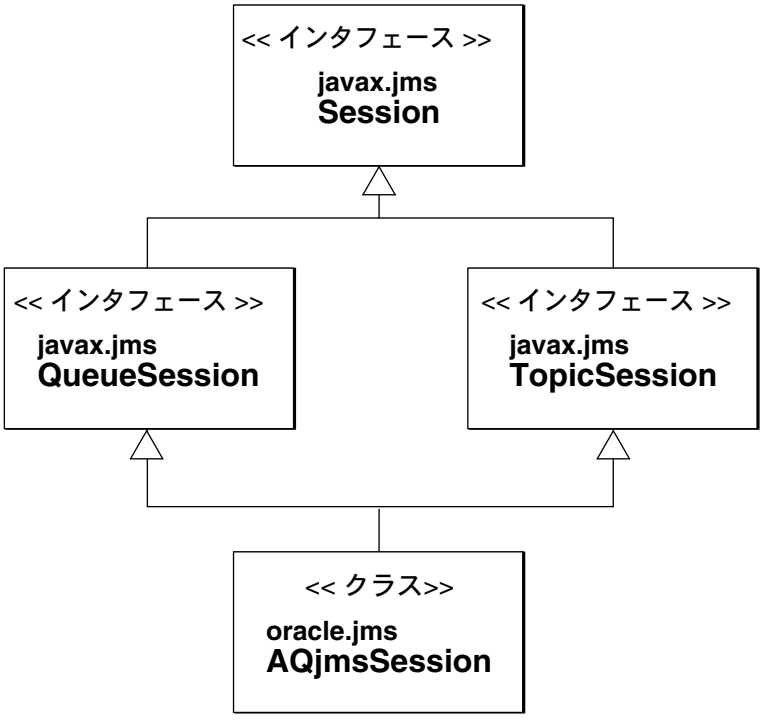


参照 :

- B-26 ページの「インタフェース - javax.jms.ConnectionMetaData」を参照してください。
- B-47 ページの「インタフェース - oracle.jms.AQjmsConnectionMetadata」も参照してください。
- B-48 ページの「クラス - oracle.jms.AQjmsConstants」も参照してください。
- B-27 ページの「インタフェース - javax.jms.DeliveryMode」も参照してください。

Oracle JMS クラス（パート 4）

図 B-4 クラス図 : Oracle クラスのクラス（パート 4）

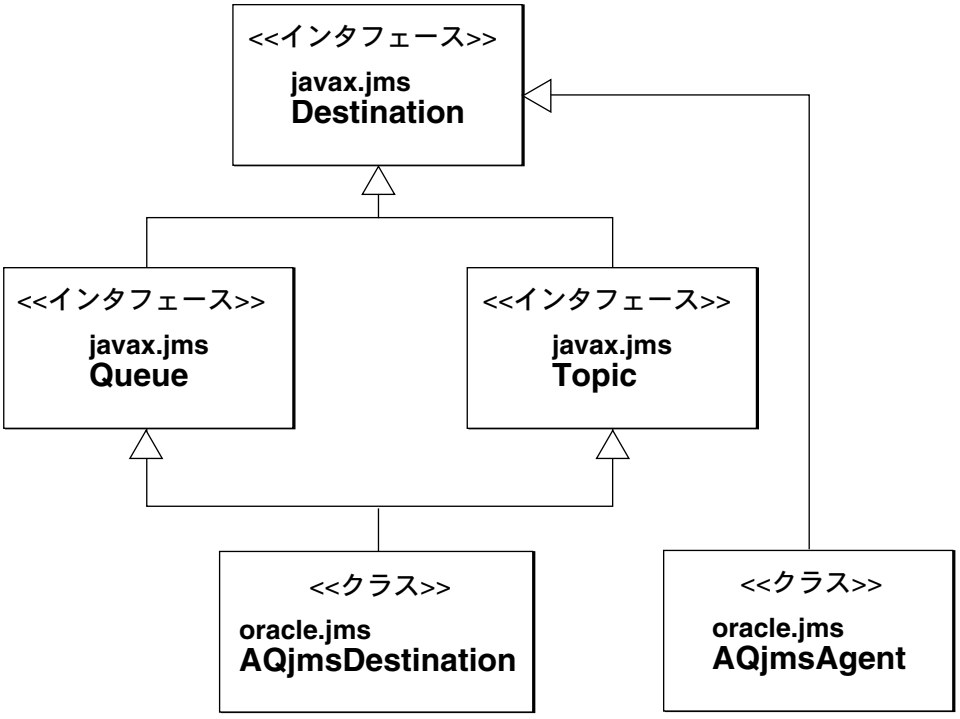


参照 :

- B-35 ページの「インタフェース - javax.jms.Session」を参照してください。
- B-35 ページの「インタフェース - javax.jms.QueueSession」も参照してください。
- B-40 ページの「インタフェース - javax.jms.TopicSession」も参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsSession」も参照してください。

Oracle JMS クラス（パート 5）

図 B-5 クラス図 : Oracle クラスのクラス（パート 5）

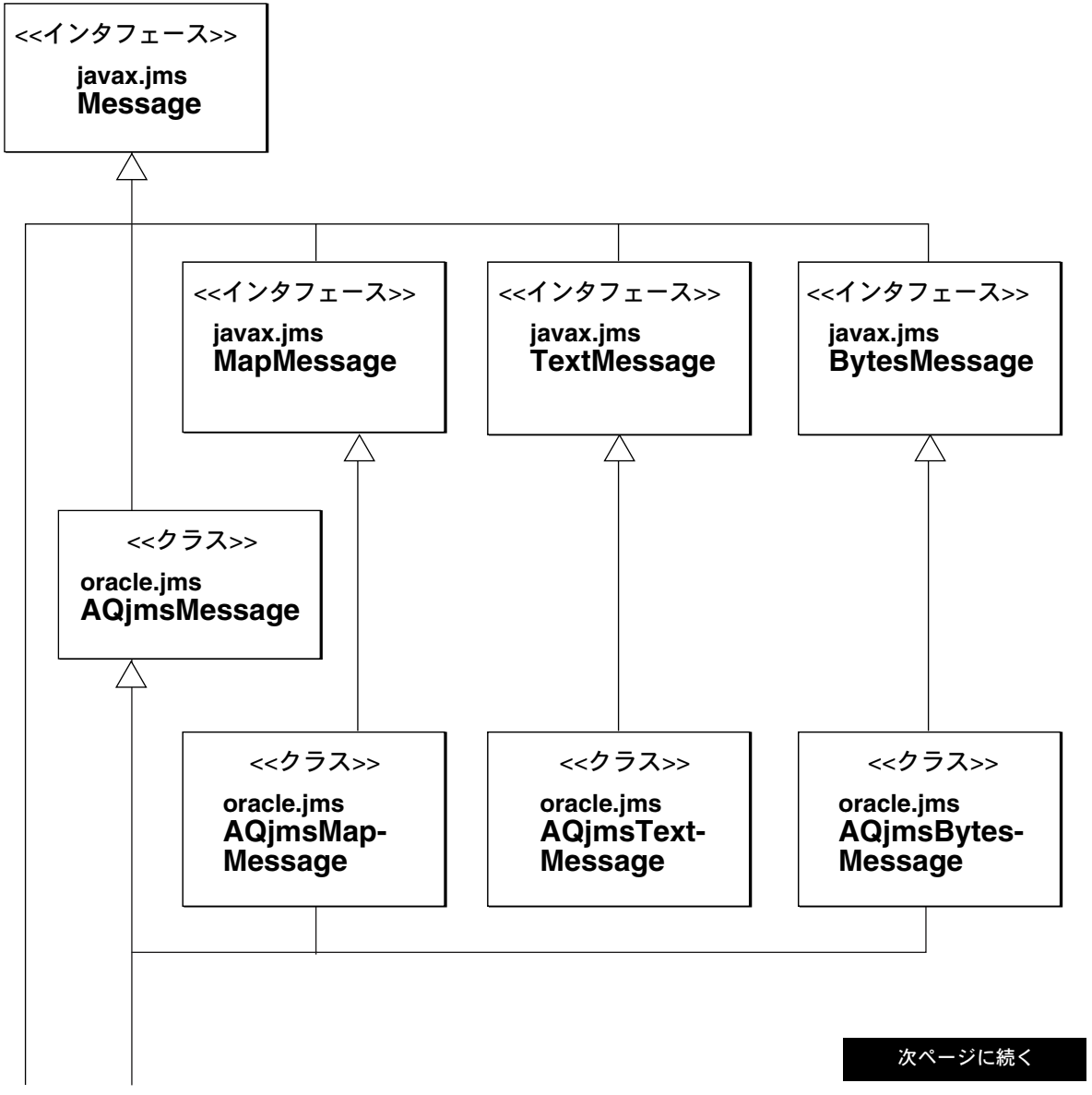


参照 :

- B-27 ページの「インタフェース - javax.jms.Destination」を参照してください。
- B-32 ページの「インタフェース - javax.jms.Queue」も参照してください。
- B-38 ページの「インタフェース - javax.jms.Topic」も参照してください。
- B-49 ページの「クラス - oracle.jms.AQjmsDestination」も参照してください。
- B-46 ページの「クラス - oracle.jms.AQjmsAgent」も参照してください。

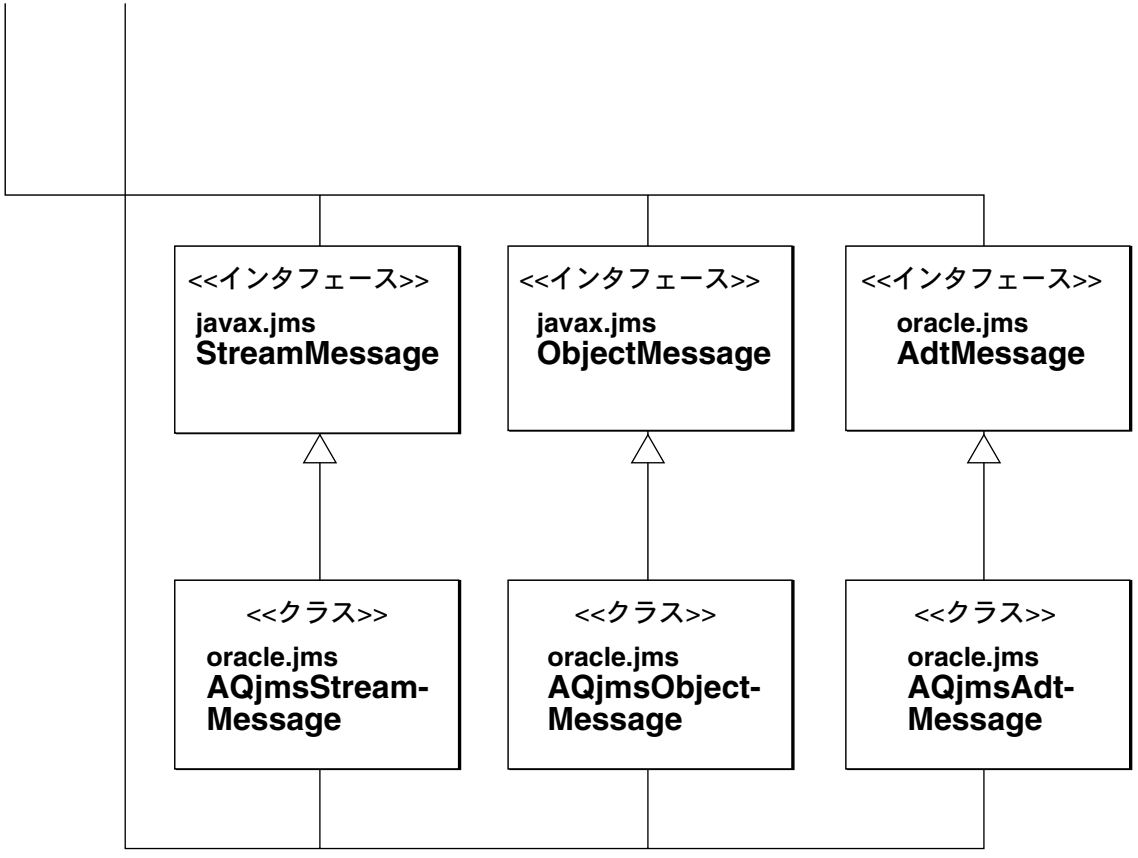
Oracle JMS クラス（パート 6）

図 B-6 クラス図 : Oracle クラスのクラス（パート 6）



Oracle JMS クラス（パート 6 の続き）

図 B-7 クラス図 : Oracle クラスのクラス（パート 6）

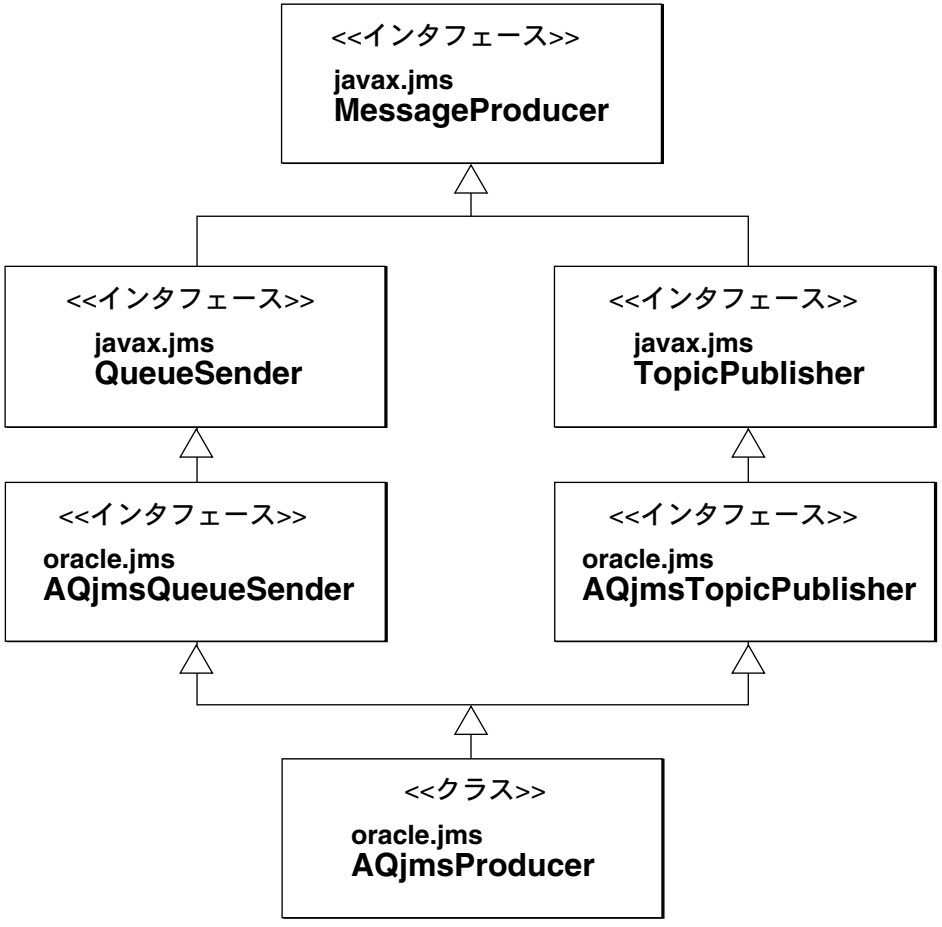


参照：

- B-28 ページの「インタフェース - javax.jms.Message」を参照してください。
- B-27 ページの「インタフェース - javax.jms.MapMessage」も参照してください。
- B-38 ページの「インタフェース - javax.jms.TextMessage」も参照してください。
- B-24 ページの「インタフェース - javax.jms.BytesMessage」も参照してください。
- B-52 ページの「クラス - oracle.jms.AQjmsMessage」も参照してください。
- B-52 ページの「クラス - oracle.jms.AQjmsMapMessage」も参照してください。
- B-57 ページの「クラス - oracle.jms.AQjmsTextMessage」も参照してください。
- B-47 ページの「クラス - oracle.jms.AQjmsBytesMessage」も参照してください。
- B-37 ページの「インタフェース - javax.jms.StreamMessage」も参照してください。
- B-32 ページの「インタフェース - javax.jms.ObjectMessage」も参照してください。
- B-44 ページの「インタフェース - oracle.jms.AdtMessage」も参照してください。
- B-57 ページの「クラス - oracle.jms.AQjmsStreamMessage」も参照してください。
- B-53 ページの「クラス - oracle.jms.AQjmsObjectMessage」も参照してください。
- B-46 ページの「クラス - oracle.jms.AQjmsAdtMessage」も参照してください。

Oracle JMS クラス（パート 7）

図 B-8 クラス図 : Oracle クラスのクラス（パート 7）

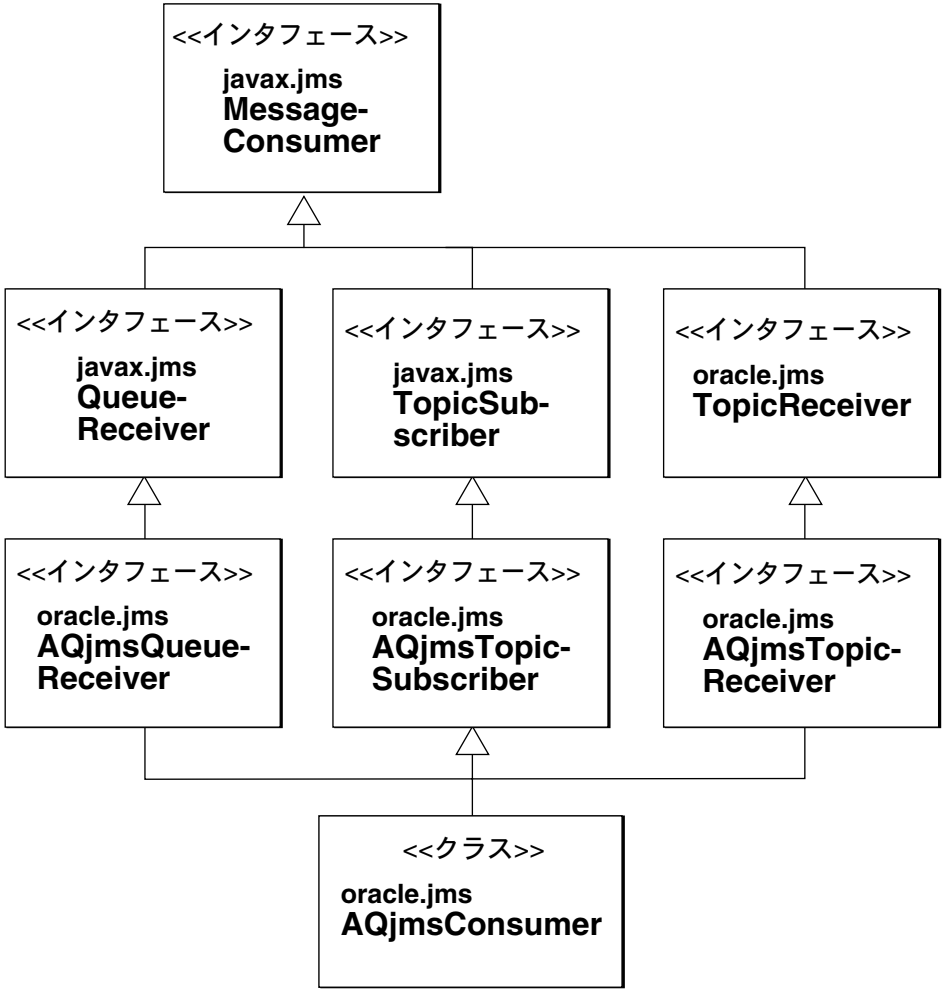


参照：

- B-31 ページの「インタフェース - javax.jms.MessageProducer」を参照してください。
- B-34 ページの「インタフェース - javax.jms.QueueSender」も参照してください。
- B-39 ページの「インタフェース - javax.jms.TopicPublisher」も参照してください。
- B-44 ページの「インタフェース - oracle.jms.AQjmsQueueSender」も参照してください。
- B-45 ページの「インタフェース - oracle.jms.AQjmsTopicPublisher」も参照してください。
- B-53 ページの「クラス - oracle.jms.AQjmsProducer」も参照してください。

Oracle JMS クラス（パート 8）

図 B-9 クラス図 : Oracle クラスのクラス（パート 8）

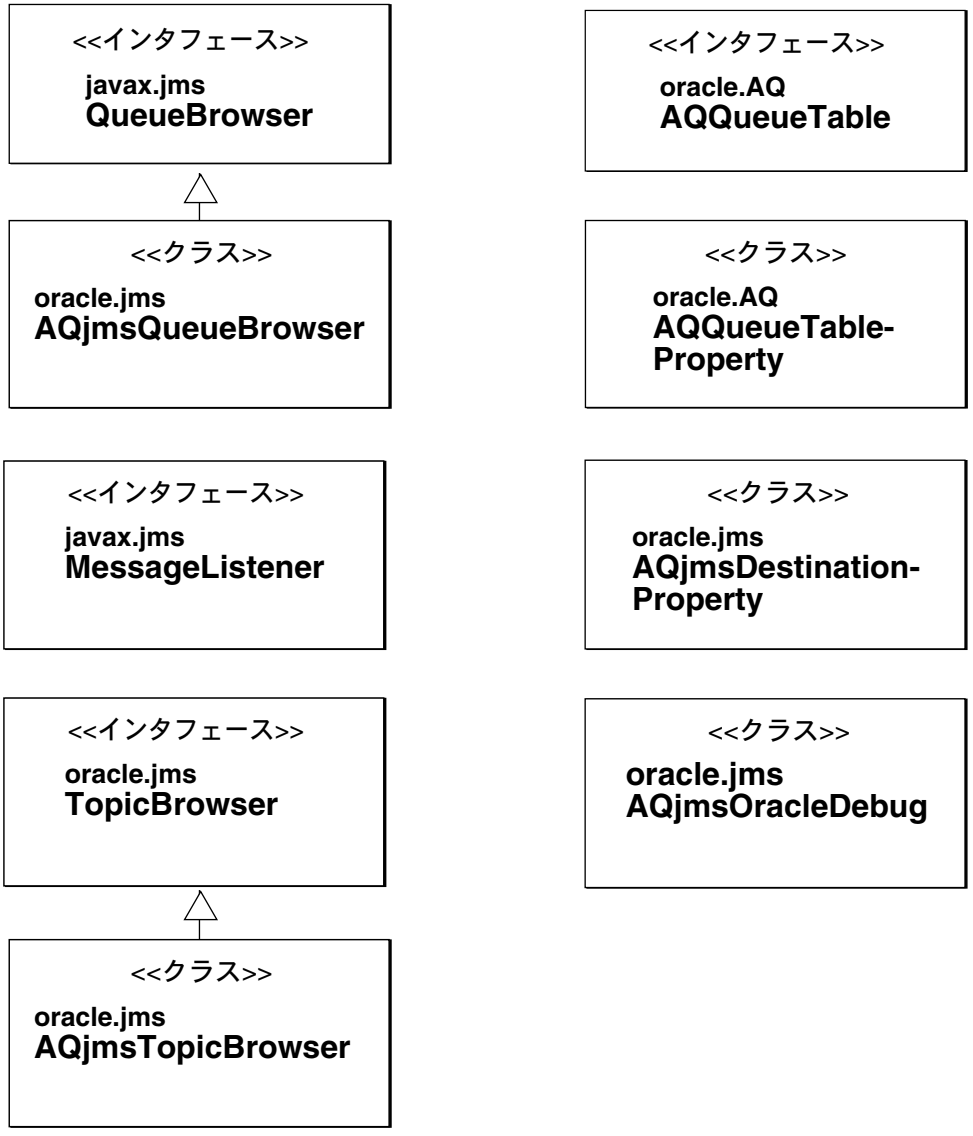


参照：

- B-30 ページの「インタフェース - javax.jms.MessageConsumer」を参照してください。
- B-34 ページの「インタフェース - javax.jms.QueueReceiver」も参照してください。
- B-40 ページの「インタフェース - javax.jms.TopicSubscriber」も参照してください。
- B-45 ページの「インタフェース - oracle.jms.TopicReceiver」も参照してください。
- B-44 ページの「インタフェース - oracle.jms.AQjmsQueueReceiver」も参照してください。
- B-45 ページの「インタフェース - oracle.jms.AQjmsTopicSubscriber」も参照してください。
- B-46 ページの「インタフェース - oracle.jms.AQjmsTopicReceiver」も参照してください。
- B-48 ページの「インタフェース - oracle.jms.AQjmsConsumer」も参照してください。

Oracle JMS クラス（パート 9）

図 B-10 クラス図 : Oracle クラスのクラス（パート 9）

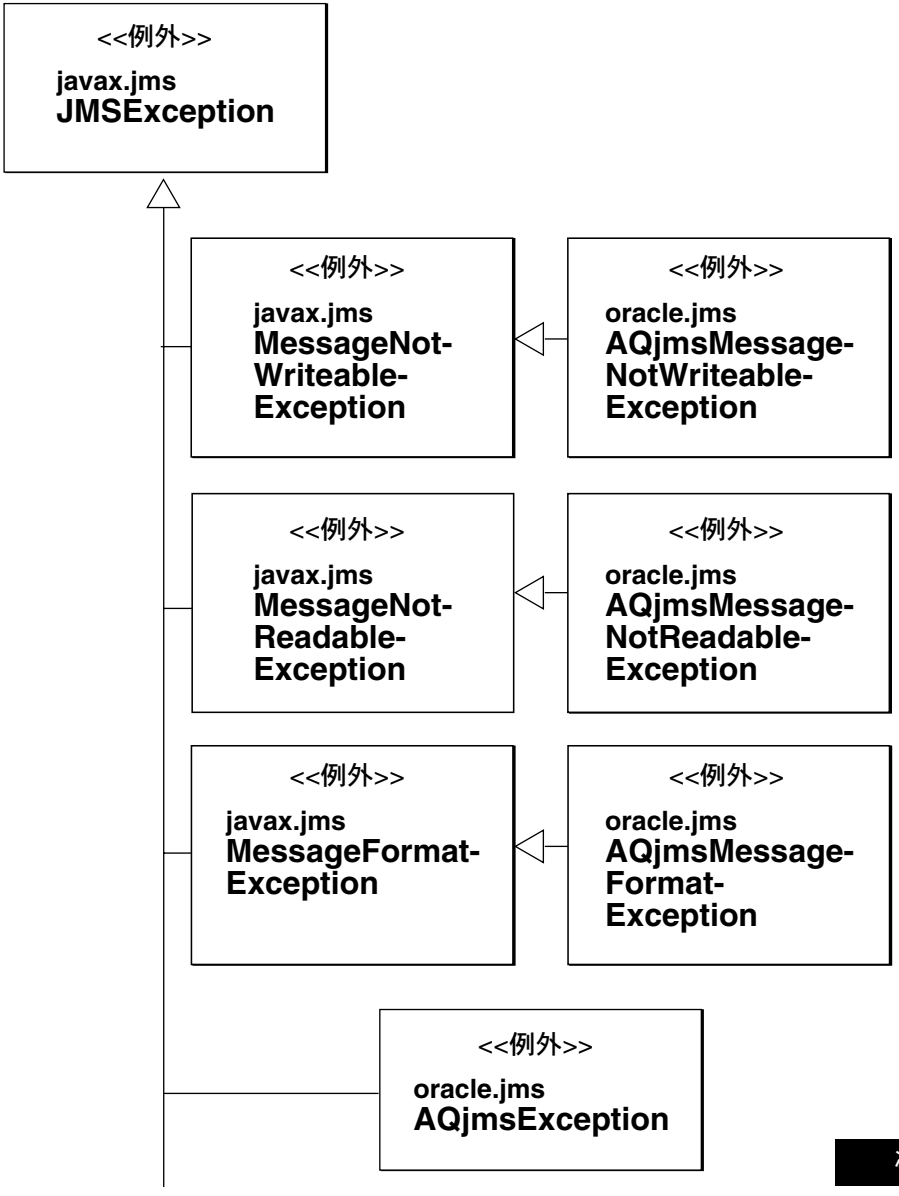


参照：

- B-32 ページの「インタフェース - javax.jms.QueueBrowser」を参照してください。
- B-54 ページの「クラス - oracle.jms.AQjmsQueueBrowser」も参照してください。
- B-31 ページの「インタフェース - javax.jms.MessageListener」も参照してください。
- B-62 ページの「インタフェース - oracle.jms.TopicBrowser」も参照してください。
- B-62 ページの「クラス - oracle.jms.AQjmsTopicBrowser」も参照してください。
- B-60 ページの「インタフェース - oracle.AQ.AQQueueTable」も参照してください。
- B-60 ページの「クラス - oracle.AQ.AQQueueTableProperty」も参照してください。
- B-50 ページの「クラス - oracle.jms.AQjmsDestinationProperty」も参照してください。
- B-53 ページの「クラス - oracle.jms.AQjmsOracleDebug」も参照してください。

Oracle JMS クラス（パート 10）

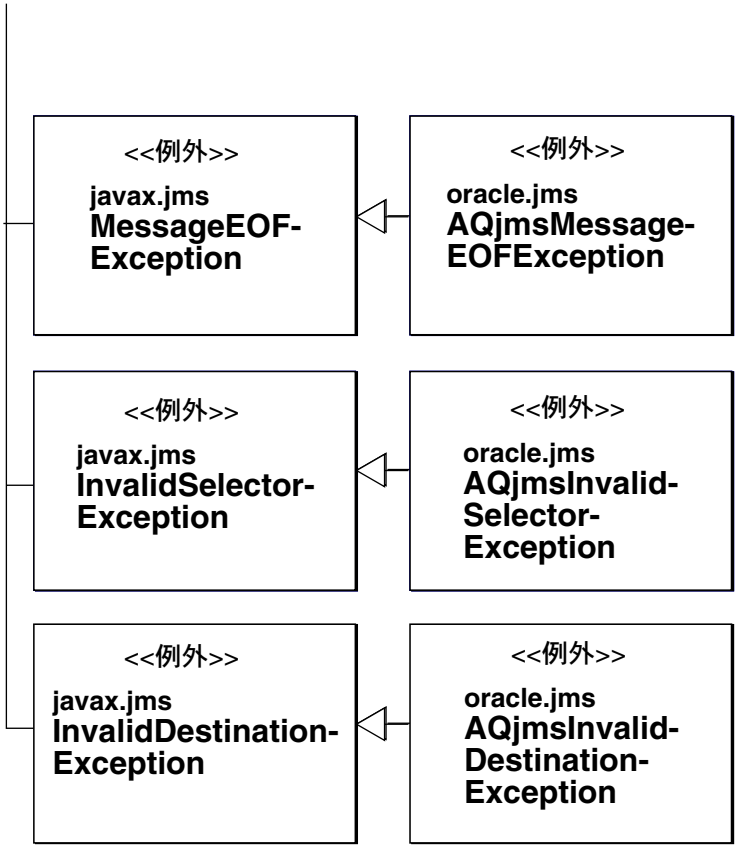
図 B-11 クラス図 : Oracle クラスのクラス（パート 10）



次ページに続く

Oracle JMS クラス（パート 10 の続き）

図 B-12 クラス図 : Oracle クラスのクラス（パート 10）



参照：

- B-42 ページの「例外 - javax.jms.JMSException」を参照してください。
- B-43 ページの「例外 - javax.jms.MesageNotWriteableException」も参照してください。
- B-59 ページの「例外 - oracle.jms.AQjmsMesssageNotWriteableException」も参照してください。
- B-43 ページの「例外 - javax.jms.MessageNotReadableException」も参照してください。
- B-59 ページの「例外 - oracle.jms.AQjmsMessageNotReadableException」も参照してください。
- B-43 ページの「例外 - javax.jms.MessageFormatException」も参照してください。
- B-59 ページの「例外 - oracle.jms.AQjmsMessageFormatException」も参照してください。
- B-58 ページの「例外 - oracle.jms.AQjmsException」も参照してください。
- B-42 ページの「例外 - javax.jms.MessageEOFException」も参照してください。
- B-59 ページの「例外 - oracle.jms.AQjmsMessageEOFException」も参照してください。
- B-41 ページの「例外 - javax.jms.InvalidSelectorException」も参照してください。
- B-59 ページの「例外 - oracle.jms.AQjmsInvalidSelectorException」も参照してください。
- B-41 ページの「例外 - javax.jms.InvalidDestinationException」も参照してください。
- B-58 ページの「例外 - oracle.jms.AQjmsInvalidDestinationException」も参照してください。

インタフェース、クラスおよび例外

インタフェース - javax.jms.BytesMessage

<< インタフェース >>
javax.jms.BytesMessage

<< メソッド >>
readBoolean()

readByte()

readBytes(byte[])

readBytes(byte[], int)

readChar()

readDouble()

readFloat()

readInt()

readLong()

readShort()

readUnsignedByte()

readUnsignedShort()

readUTF()

reset()

writeBoolean(boolean)

writeByte(byte)

writeBytes(byte[])

writeBytes(byte[], int, int)

writeChar(char)

writeDouble(double)

writeFloat(float)

writeInt(int)

writeLong(long)

writeObject(Object)

writeShort(short)

writeUTF(String)

参照： B-12 ページの「Oracle JMS クラス（パート 6）」を参照してください。

インタフェース - javax.jms.Connection

<< インタフェース >>
javax.jms.Connection

<< メソッド >>
close()

getClientID()

getMetaData()

start()

stop()

getExceptionListener()

setExceptionListener(ExceptionListener)

参照：

- B-8 ページの「Oracle JMS クラス（パート 2）」を参照してください。

利用方法：

- JMS コネクションの開始
- JMS コネクションの停止
- JMS コネクションのクローズ

インタフェース - javax.jms.ConnectionFactory

<< インタフェース >>
javax.jms.ConnectionFactory

参照 :

- B-6 ページの「Oracle JMS クラス（パート 1)」を参照してください。

利用方法 :

- キュー・コネクションの確立 : ユーザー名 / パスワードの使用
- キュー・コネクションの確立 : オープンしている JDBC コネクションの使用

インタフェース - javax.jms.ConnectionMetaData

<< インタフェース >>
javax.jms.ConnectionMetaData

<< メソッド >>
getJMSMajorVersion()

getJMSMinorVersion()

getJMSProviderName()

getJMSVersion()

getProviderMajorVersion()

getProviderMinorVersion()

getProviderVersion()

参照 : B-9 ページの「Oracle JMS クラス（パート 3)」を参照してください。

インタフェース - javax.jms.DeliveryMode

<< インタフェース >>
javax.jms.DeliveryMode

<< 定数 >>
NON_PERSISTENT（現在サポートしていません）
PERSISTENT

参照： B-9 ページの「Oracle JMS クラス（パート 3）」を参照してください。

インタフェース - javax.jms.Destination

<< インタフェース >>
javax.jms.Destination

参照： B-11 ページの「Oracle JMS クラス（パート 5）」を参照してください。

インタフェース - javax.jms.MapMessage

<< インタフェース >>
javax.jms.MapMessage

<< メソッド >>
getBoolean(String)
getByte(String)
getBytes(String)
getChar(String)
getDouble(String)
getFloat(String)
getInt(String)
getLong(String)
getMapNames()
getObject(String)
getShort(String)

getString(String)

itemExists(String)

setBoolean(String, boolean)

setByte(String, byte)

setBytes(String, byte[])

setBytes(String, byte[], int, int)

setChar(String, char)

setDouble(String, double)

setFloat(String, float)

setInt(String, int)

setLong(String, long)

setObject(String, Object)

setShort(String, short)

setString(String, String)

参照： B-12 ページの「Oracle JMS クラス（パート 6）」を参照してください。

インタフェース - javax.jms.Message

<< インタフェース >>
javax.jms.Message

<< メソッド >>
clearBody()

clearProperties()

getBooleanProperty(String)

getByteProperty(String)

getDoubleProperty(String)

getFloatProperty(String)

getIntProperty(String)

getJMSCorrelationID()

getJMSCorrelationIDAsBytes()

getJMSDeliveryMode()

getJMSDestination()

getJMSExpiration()

getJMSMessageID()

getJMSPriority()

getJMSReplyTo()

getJMSTimestamp()

getJMSType()

getLongProperty(String)

getObjectProperty(String)

getPropertyNames()

getShortProperty(String)

getStringProperty(String)

propertyExists(String)

setBooleanProperty(String, boolean)

setByteProperty(String, byte)

<< メソッド >>

setDoubleProperty(String, double)

setFloatProperty(String, float)

setIntProperty(String, int)

setJMSCorrelationID(String)

setJMSCorrelation(IDAsBytes(byte[])

setJMSReplyTo(Destination)

setJMSType(String)

setLongProperty(String, long)

setObjectProperty(String, Object)

setShortProperty(String, short)

setStringProperty(String, String)

参照：

- B-6 ページの「Oracle JMS クラス（パート 1）」を参照してください。

利用方法：

- メッセージの相関識別子の指定
- JMS メッセージ・プロパティの指定
- JMS メッセージ・プロパティを Boolean として指定
- JMS メッセージ・プロパティを String として指定
- JMS メッセージ・プロパティを Int として指定
- JMS メッセージ・プロパティを Double として指定
- JMS メッセージ・プロパティを Float として指定
- JMS メッセージ・プロパティを Byte として指定
- JMS メッセージ・プロパティを Long として指定
- JMS メッセージ・プロパティを Object として指定

インタフェース - javax.jms.MessageConsumer

<< インタフェース >>
javax.jms.MessageConsumer

<< メソッド >>
close()

getMessageListener()

getMessageSelector()

receive()

receive(long)

receiveNoWait()

setMessageListener(MessageListener)

参照：

- B-17 ページの「Oracle JMS クラス（パート 8）」を参照してください。

利用方法：

- メッセージ・コンシューマを使用したメッセージの同期受信 : タイムアウトを指定
- メッセージ・コンシューマを使用したメッセージの同期受信 : 待機なし

インタフェース - javax.jms.MessageListener

<< インタフェース >>
javax.jms.MessageListener

<< メソッド >>
onMessage(Message)

参照： B-19 ページの「Oracle JMS クラス（パート 9）」を参照してください。

インタフェース - javax.jms.MessageProducer

<< インタフェース >>
javax.jms.MessageProducer

<< メソッド >>
close()

getDeliveryMode()

getDisableMessageID()

getPriority()

getTimeToLive()

setDisableMessageID(boolean)

setPriority(int)

setTimeToLive(int)

参照：

- B-15 ページの「Oracle JMS クラス（パート 7)」を参照してください。

利用方法：

- メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの Time-To-Live の設定
- メッセージ・プロデューサが送信するすべてのメッセージに対するデフォルトの優先順位の設定

インタフェース - javax.jms.ObjectMessage

<< インタフェース >>
javax.jms.bjectMessage

<< メソッド >>
getObject()

setObject(Serializable)

参照： B-13 ページの「Oracle JMS クラス（パート 6 の続き)」を参照してください。

インタフェース - javax.jms.Queue

<< インタフェース >>
javax.jms.Queue

<< メソッド >>
getQueueName()

toString()

参照： B-19 ページの「Oracle JMS クラス（パート 9)」を参照してください。

インタフェース - javax.jms.QueueBrowser

<< インタフェース >>
javax.jms.Queue Browser

<< メソッド >>
close()

getEnumeration()

getMessageSelector()

getQueue()

> **参照：** B-19 ページの「Oracle JMS クラス（パート 9）」を参照してください。

インタフェース - javax.jms.QueueConnection

<< インタフェース >>
javax.jms.QueueConnection

<< メソッド >>
createQueueSession(boolean, int)

> **参照：**
>
> - B-8 ページの「Oracle JMS クラス（パート 2）」を参照してください。
>
> **利用方法：**
>
> - キュー・セッションの作成

インタフェース - javax.jms.QueueConnectionFactory

<< インタフェース >>
javax.jms.QueueConnectionFactory

<< メソッド >>
createQueueConnection()

createQueueConnection(String, String)

参照 :

- B-6 ページの「Oracle JMS クラス（パート 1）」を参照してください。

利用方法 :

- キュー・コネクションの確立 : ユーザー名 / パスワードの使用
- キュー・コネクションの確立 : オープンしている JDBC コネクションの使用
- キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用
- キュー・コネクションの確立 : オープンしている OracleOCIConnectionPool の使用

インタフェース - javax.jms.QueueReceiver

<< インタフェース >>
javax.jms.QueueReceiver

<< メソッド >>
getQueue()

参照 : B-17 ページの「Oracle JMS クラス（パート 8）」を参照してください。

インタフェース - javax.jms.QueueSender

<< インタフェース >>
javax.jms.QueueSender

<< メソッド >>
getQueue()

send(Message)

send(Message, int, int, long)

send(Queue, Message)

send(Queue, Message, int, int, long)

参照：

- B-15 ページの「Oracle JMS クラス（パート 7）」を参照してください。

利用方法：

- キュー・センダーの作成
- メッセージの送信 : デフォルト送信オプションのキュー・センダーの使用
- メッセージの送信 : 送信オプションを指定したキュー・センダーの使用

インタフェース - javax.jms.QueueSession

<< インタフェース >>
javax.jms.QueueSession

<< メソッド >>
createBrowser(Queue)

createBrowser(Queue, String)

createQueue(String)

createReceiver(Queue)

createReceiver(Queue, String)

createSender(Queue)

参照： B-10 ページの「Oracle JMS クラス（パート 4）」を参照してください。

インタフェース - javax.jms.Session

<< インタフェース >>
javax.jms.Session

<< 定数 >>
AUTO_ACKNOWLEDGE

CLIENT_ACKNOWLEDGE

DUPS_OK_ACKNOWLEDGE

<< メソッド >>

close()

commit()

createBytesMessage()

createMapMessage()

createMessage()

createObjectMessage()

createObjectMessage(Serializable)

createStreamMessage()

createTextMessage()

createTextMessage(StringBuffer)

getMessageListener()

getTransacted()

rollback()

setMessageListener(MessageListener)

参照： B-10 ページの「Oracle JMS クラス（パート 4）」を参照してください。

利用方法：

- キュー・センダーの作成
- Text、Stream、Object、Bytes、MapMessage を使用するキューに対するキュー・ブラウザの作成
- キュー・レシーバの作成 : 標準 JMS 型メッセージ・キュー
- キュー・コネクションの確立 : オープンしている JDBC コネクションの使用
- MapMessage の作成
- StreamMessage の作成
- ObjectMessage の作成
- TextMessage の作成

インタフェース - javax.jms.StreamMessage

<< インタフェース >>
javax.jms.StreamMessage

<< メソッド >>
readBoolean()

readByte()

readBytes(byte[])

readChar()

readDouble()

readFloat()

readInt()

readLong()

readObject()

readShort()

readString()

reset()

writeBoolean(boolean)

writeByte(byte)

writeBytes(byte[])

writeBytes(byte[], int, int)

writeChar(char)

writeDouble(double)

writeFloat(float)

writeInt(int)

writeLong(long)

writeObject(Object)

writeShort(short)

writeString(String)

参照： B-13 ページの「Oracle JMS クラス（パート 6 の続き）」を参照してください。

インタフェース - javax.jms.TextMessage

<< インタフェース >>
javax.jms.TextMessage

<< メソッド >>
getText()

setText(String)

参照： B-12 ページの「Oracle JMS クラス（パート 6）」を参照してください。

インタフェース - javax.jms.Topic

<< インタフェース >>
javax.jms.Topic

<< メソッド >>
getTopicName()

toString()

参照： B-11 ページの「Oracle JMS クラス（パート 5）」を参照してください。

インタフェース - javax.jms.TopicConnection

<< インタフェース >>
javax.jms.TopicConnection

<< メソッド >>
createTopicSession(boolean, int)

参照：

- B-8 ページの「Oracle JMS クラス（パート 2）」を参照してください。

利用方法：

- トピック・コネクションの確立 : ユーザー名 / パスワードの使用

インタフェース - javax.jms.TopicConnectionFactory

<< インタフェース >>
javax.jms.TopicConnectionFactory

<< メソッド >>
createTopicConnection()

createTopicConnection(String, String)

参照： B-6 ページの「Oracle JMS クラス（パート 1）」を参照してください。

インタフェース - javax.jms.TopicPublisher

<< インタフェース >>
javax.jms.TopicPublisher

<< メソッド >>
getTopic()

publish(Message)

publish(Message, int, int, long)

publish(Topic, Message)

publish(Topic, Message, int, int, long)

参照 :

- B-15 ページの「Oracle JMS クラス（パート 7)」を参照してください。

利用方法 :

- トピック・パブリッシャを使用したメッセージのパブリッシュ : 最小限の指定
- トピック・パブリッシャを使用したメッセージのパブリッシュ : 相関および遅延を指定
- トピック・パブリッシャを使用したメッセージのパブリッシュ : 優先順位および Time-To-Live を指定
- トピック・パブリッシャを使用したメッセージのパブリッシュ : トピック・サブスクライバをオーバーライドする受信者リストを指定

インタフェース - javax.jms.TopicSession

<< インタフェース >>
javax.jms.TopicSession

<< メソッド >>
createDurableSubscriber(Topic, String)

createDurableSubscriber(Topic, String, String, boolean)

createPublisher(Topic)

参照 :

- B-15 ページの「Oracle JMS クラス（パート 4)」を参照してください。

利用方法 :

- JMS トピックに対する永続サブスクライバの作成 : セレクタの指定なし
- JMS トピックに対する永続サブスクライバの作成 : セレクタの指定あり

インタフェース - javax.jms.TopicSubscriber

<< インタフェース >>
javax.jms.TopicSubscriber

<< メソッド >>
getNoLocal()

getTopic()

> **参照：** B-17 ページの「Oracle JMS クラス（パート 8）」を参照してください。

例外 - javax.jms.InvalidDestinationException

<< 例外 >>
javax.jms.InvalidDestination-Exception

<< コンストラクタ >>
InvalidDestinationException(String)

InvalidDestinationException(String, String)

> **参照：** B-22 ページの「Oracle JMS クラス（パート 10 の続き）」を参照してください。

例外 - javax.jms.InvalidSelectorException

<< 例外 >>
javax.jms.InvalidSelector-Exception

<< コンストラクタ >>
InvalidSelectorException(String)

InvalidSelectorException(String, String)

> **参照：** B-22 ページの「Oracle JMS クラス（パート 10 の続き）」を参照してください。

例外 - javax.jms.JMSException

<< 例外 >>
javax.jms.JMSException

<< コンストラクタ >>
JMSException(String)

JMSException(String, String)

<< メソッド >>
getErrorCode()

getLinkedException()

setLinkedException(Exception)

参照 :

- B-21 ページの「Oracle JMS クラス（パート 10)」を参照してください。

利用方法 :

- JMS 例外のエラー・コードの取得
- JMS 例外のエラー・メッセージの取得
- JMS 例外にリンクされた例外の取得
- JMS 例外のスタック・トレースの出力

例外 - javax.jms.MessageEOFException

<< 例外 >>
javax.jms.MessageEOFException

<< コンストラクタ >>
MessageEOFException(String)

MessageEOFException(String, String)

参照 : B-22 ページの「Oracle JMS クラス（パート 10 の続き)」を参照してください。

例外 - javax.jms.MessageFormatException

<< 例外 >>
javax.jms.MessageFormatException

<< コンストラクタ >>
MessageFormatException(String)

MessageFormatException(String, String)

参照： B-21 ページの「Oracle JMS クラス（パート 10）」を参照してください。

例外 - javax.jms.MessageNotReadableException

<< 例外 >>
javax.jms.MessageNotReadable-Exception

<< コンストラクタ >>
MessageNotReadableException(String)

MessageNotReadableException(String, String)

参照： B-21 ページの「Oracle JMS クラス（パート 10）」を参照してください。

例外 - javax.jms.MesageNotWriteableException

<< 例外 >>
javax.jms.MessageNotWriteable-Exception

<< コンストラクタ >>
MessageNotWriteableException(String)

MessageNotWriteableException(String, String)

参照： B-21 ページの「Oracle JMS クラス（パート 10）」を参照してください。

インタフェース - oracle.jms.AdtMessage

<< インタフェース >>
oracle.jms.AdtMessage

<< メソッド >>
getAdtPayload()

setAdtPayload(CustomDatum)

> **参照：** B-13 ページの「Oracle JMS クラス（パート 6 の続き）」を参照してください。

インタフェース - oracle.jms.AQjmsQueueReceiver

<< インタフェース >>
oracle.jms.AQjmsQueueReceiver

<< メソッド >>
getNavigationMode()

receiveNoData()

receiveNoData(long)

setNavigationMode(int)

> **参照：**
>
> - B-17 ページの「Oracle JMS クラス（パート 8）」を参照してください。
>
> **利用方法：**
>
> - メッセージの受信に対するナビゲーション・モードの指定

インタフェース - oracle.jms.AQjmsQueueSender

<< インタフェース >>
oracle.jms.AQjmsQueueSender

> **参照：** B-15 ページの「Oracle JMS クラス（パート 7）」を参照してください。

インタフェース - oracle.jms.AQjmsTopicPublisher

<< インタフェース >>
oracle.jms.AQjmsTopicPublisher

<< メソッド >>
publish(Message, AQjmsAgent[])

publish(Message, AQjmsAgent[], int, int, long)

publish(Topic, Message, AQjmsAgent[])

publish(Topic, Message, AQjmsAgent[], int, int, long)

参照： B-15 ページの「Oracle JMS クラス（パート 7）」を参照してください。

インタフェース - oracle.jms.TopicReceiver

<< インタフェース >>
oracle.jms.AQjmsTopicReceiver

<< メソッド >>
getNavigationMode()

receiveNoData()

receiveNoData(long)

setNavigationMode(int)

参照： B-17 ページの「Oracle JMS クラス（パート 8）」を参照してください。

インタフェース - oracle.jms.AQjmsTopicSubscriber

<< インタフェース >>
oracle.jms.AQjmsTopicSubscriber

<< メソッド >>
getNavigationMode()

receiveNoData()

receiveNoData(long)

setNavigationMode(int)

参照： B-17 ページの「Oracle JMS クラス（パート 8）」を参照してください。

インタフェース - oracle.jms.AQjmsTopicReceiver

<< インタフェース >>
oracle.jms.TopicReceiver

<< メソッド >>
getTopic()

参照： B-17 ページの「Oracle JMS クラス（パート 8）」を参照してください。

クラス - oracle.jms.AQjmsAdtMessage

<< クラス >>
oracle.jms.AQjmsAdtMessage

<< メソッド >>
getAdtPayload()

setAdtPayload(CustomDatum)

参照： B-13 ページの「Oracle JMS クラス（パート 6 の続き）」を参照してください。

クラス - oracle.jms.AQjmsAgent

<< クラス >>
oracle.jms.AQjmsAgent

<< コンストラクタ >>
AQjmsAgent(String, String)

AQjmsAgent(String, String, int)

<< メソッド >>
getAddress()

getName()

getProtocol()

setAddress(String)

setName(String)

setProtocol(int)

toString()

参照：

- B-11 ページの「Oracle JMS クラス（パート 5）」を参照してください。

利用方法：

- AQjms エージェントの作成

クラス - oracle.jms.AQjmsBytesMessage

<< クラス >>
oracle.jms.AQjmsBytesMessage

参照： B-12 ページの「Oracle JMS クラス（パート 6）」を参照してください。

クラス - oracle.jms.AQjmsConnection

<< クラス >>
oracle.jms.AQjmsConnection

<< メソッド >>
getCurrentJmsSession()

getOCIConnectionPool()

参照： B-8 ページの「Oracle JMS クラス（パート 2）」を参照してください。

インタフェース - oracle.jms.AQjmsConnectionMetadata

<< インタフェース >>
oracle.jms.AQjmsConnectionMeta-Data

参照： B-9 ページの「Oracle JMS クラス（パート 3）」を参照してください。

クラス - oracle.jms.AQjmsConstants

<< クラス >>
oracle.jms.AQjmsConstants

<< 定数 >>
EXCEPTION

NAVIGATION_FIRST_MESSAGE

NAVIGATION_NEXT_MESSAGE

NAVIGATION_NEXT_TRANSACTION

NONE

NORMAL

STATE_EXPIRED

STATE_PROCESSED

STATE_READY

STATE_WAITING

TRANSACTIONAL

WAIT_FOREVER

WAIT_NONE

参照： B-9 ページの「Oracle JMS クラス（パート 3）」を参照してください。

インタフェース - oracle.jms.AQjmsConsumer

<< インタフェース >>
oracle.jms.AQjmsConsumer

参照： B-17 ページの「Oracle JMS クラス（パート 8）」を参照してください。

クラス - oracle.jms.AQjmsDestination

<< クラス >>
oracle.jms.AQjmsDestination

<< メソッド >>
alter(Session, AQjmsDestinationProperty)

alterPropagationSchedule(Session, String, Double, String, Double)

disablePropagationSchedule(Session, String)

drop(Session)

enablePropagationSchedule(Session, String)

getCompleteName()

getCompleteTableName()

getQueueName()

getQueueOwner()

getTopicName()

getTopicOwner()

grantQueuePrivilege(Session, String, String, boolean)

grantTopicPrivilege(Session, String, String, boolean)

revokeQueuePrivilege(Session, String, String)

<< メソッド >>
revokeTopicPrivilege(Session, String, String)

schedulePropagation(Session, String, Date, Double, String, Double)

start(Session, boolean, boolean)

stop(Session, boolean, boolean, boolean)

toString()

unschedulePropagation(Session, String)

参照 :

- B-11 ページの「Oracle JMS クラス（パート 5)」を参照してください。

利用方法 :

- トピック権限の付与 : パブリッシュ・サブスクライブ
- 宛先の開始
- 宛先の停止
- 宛先の変更
- 宛先の削除
- 伝播のスケジューリング
- 伝播スケジュールの使用可能化
- 伝播スケジュールの変更
- 伝播スケジュールの使用不可能化
- 伝播スケジュールの解除

クラス - oracle.jms.AQjmsDestinationProperty

<< クラス >>
oracle.jms.AQjmsDestinationProperty

<< 定数 >>
EXCEPTION_QUEUE

INFINITE

NORMAL_QUEUE

<< コンストラクタ >>
AQjmsDestinationProperty()

<< メソッド >>
getComment()

getMaxRetries()

getQueueType()

getRetentionTime()

getRetryInterval()

setComment(java.lang.String qt_comment)

setMaxRetries(int retries)

setMaxRetries(java.lang.Integer retries)

setQueueType(int q_type)

setRetentionTime(double r_time)

setRetentionTime(java.lang.Double r_time)

setRetryInterval(double interval)

setRetryInterval(java.lang.Double interval)

toString()

参照 :

- B-19 ページの「Oracle JMS クラス（パート 9)」を参照してください。

利用方法 :

- 宛先プロパティの指定

クラス - oracle.jms.AQjmsFactory

<< クラス >>
oracle.jms.AQjmsFactory

<<static>>
getQueueConnectionFactory(String, Properties)

getQueueConnectionFactory(String, String, int, String)

getTopicConnectionFactory(String, Properties)

getTopicConnectionFactory(String, String, int, String)

参照 :

- B-6 ページの「Oracle JMS クラス（パート 1）」を参照してください。

利用方法 :

- キュー・コネクション・ファクトリの取得 : JDBC URL の使用
- キュー・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用
- トピック・コネクション・ファクトリの取得 : JDBC URL の使用
- トピック・コネクション・ファクトリの取得 : JDBC コネクション・パラメータの使用

クラス - oracle.jms.AQjmsMapMessage

<< クラス >>
oracle.jms.AQjmsMapMessage

参照 : B-12 ページの「Oracle JMS クラス（パート 6）」を参照してください。

クラス - oracle.jms.AQjmsMessage

<< クラス >>
oracle.jms.AQjmsMessage

<< メソッド >>
getJMSMessageIDAsBytes()

getSenderID()

setSenderID(AQjmsAgent)

参照 :

- B-12 ページの「Oracle JMS クラス（パート 6）」を参照してください。

利用方法 :

- メッセージのメッセージ ID を Byte として取得

クラス - oracle.jms.AQjmsObjectMessage

<< クラス >>
oracle.jms.AQjmsObjectMessage

参照： B-13 ページの「Oracle JMS クラス（パート 6 の続き）」を参照してください。

クラス - oracle.jms.AQjmsOracleDebug

<< クラス >>
oracle.jms.AQjmsOracleDebug

<< 定数 >>
AQ_ORA_TR1

AQ_ORA_TR2

AQ_ORA_TR3

AQ_ORA_TR4

AQ_ORA_TR5

<< メソッド >>
getLogStream()

setLogStream(OutputStream)

setTraceLevel(int)

参照： B-19 ページの「Oracle JMS クラス（パート 9）」を参照してください。

クラス - oracle.jms.AQjmsProducer

<< クラス >>
oracle.jms.AQjmsProducer

参照： B-15 ページの「Oracle JMS クラス（パート 7）」を参照してください。

クラス - oracle.jms.AQjmsQueueBrowser

<< クラス >>
oracle.jms.AQjmsQueueBrowser

参照 :

- B-19 ページの「Oracle JMS クラス（パート 9)」を参照してください。

利用方法 :

- キュー・ブラウザを使用したメッセージのブラウズ

クラス - oracle.jms.AQjmsQueueConnectionFactory

<< クラス >>
oracle.jmsAQjmsQueueConnection-Factory

<<static>>
createQueueConnection(Connection)

createQueueConnection(OracleOCIConnectionPool)

<< メソッド >>
createQueueConnection()

createQueueConnection(String, String)

参照 :

- B-6 ページの「Oracle JMS クラス（パート 1)」を参照してください。

利用方法 :

- キュー・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用

クラス - oracle.jms.AQjmsSession

<< クラス >>
oracle.jms.AQjmsSession

<< メソッド >>
createAdtMessage()

createAdtMessage(CustomDatum)

createBrowser(Queue, CustomDatumFactory)

createBrowser(Queue, String, boolean)

createBrowser(Queue, String, CustomDatumFactory)

createBrowser(Queue, String, CustomDatumFactory, boolean)

createBrowser(Topic, String)

createBrowser(Topic, String, boolean)

createBrowser(Topic, String, CustomDatumFactory)

createBrowser(Topic, String, CustomDatumFactory, boolean)

createBrowser(Topic, String, String)

createBrowser(Topic, String, String, boolean)

createBrowser(Topic, String, String, CustomDatumFactory)

createBrowser(Topic, String, String, CustomDatumFactory, boolean)

createDurableSubscriber(Topic, String, CustomDatumFactory)

createDurableSubscriber(Topic, String, String, boolean, CustomDatumFactory)

createQueue(AQQueueTable, String, AQjmsDestinationProperty)

createQueueTable(String, String, AQQueueTableProperty)

createReceiver(Queue, CustomDatumFactory)

createReceiver(Queue, String, CustomDatumFactory)

createRemoteSubscriber(Topic, AQjmsAgent, String)

createRemoteSubscriber(Topic, AQjmsAgent, String, CustomDatumFactory)

createTopic(AQQueueTable, String, AQjmsDestinationProperty)

createTopicReceiver(Topic, String, String)

createTopicReceiver(Topic, String, String, CustomDatumFactory)

getDBConnection()

getJmsConnection()

getQueue(String, String)

getQueueTable(String, String)

getTopic(String, String)

grantSystemPrivilege(String, String, boolean)

revokeSystemPrivilege(String, String)

unsubscribe(Topic, AQjmsAgent)

unsubscribe(Topic, String)

参照 :

- B-10 ページの「Oracle JMS クラス（パート 4）」を参照してください。

利用方法 :

- キュー表の作成
- キュー表の取得
- トピックの作成 : パブリッシュ・サブスクライブ
- システム権限の付与
- システム権限の取消し
- キュー権限の付与 : Point-to-Point
- キュー権限の取消し : Point-to-Point
- Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成
- Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成
- Oracle オブジェクト型（ユーザー定義型）メッセージ・キューに対するキュー・ブラウザの作成 : メッセージをロック
- キュー・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・キュー
- ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定なし
- ユーザー定義型トピックに対する永続サブスクライバの作成 : セレクタの指定あり

- リモート・サブスクライバの作成 : JMS 型メッセージ・トピック
- リモート・サブスクライバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック
- 永続サブスクリプションのサブスクライブの解除 : ローカル・サブスクライバ
- 永続サブスクリプションのサブスクライブの解除 : リモート・サブスクライバ
- トピック・レシーバの作成 : 標準 JMS 型メッセージ・トピック
- トピック・レシーバの作成 : Oracle オブジェクト型（ユーザー定義型）メッセージ・トピック
- セッションからの JMS コネクションの取得
- JMS セッションからの JDBC コネクションの取得
- ユーザー定義型メッセージの作成

クラス - oracle.jms.AQjmsStreamMessage

<< クラス >>
oracle.jms.AQjmsStreamMessage

参照 : B-13 ページの「Oracle JMS クラス（パート 6 の続き）」を参照してください。

クラス - oracle.jms.AQjmsTextMessage

<< クラス >>
oracle.jms.AQjmsTextMessage

参照 : B-12 ページの「Oracle JMS クラス（パート 6）」を参照してください。

クラス - oracle.jms.AQjmsTopicConnectionFactory

<< クラス >>
oracle.jms.AQjmsTopicConnection-Factory

<<static>>
createTopicConnection(Connection)

createTopicConnection(OracleOCIConnectionPool)

<< メソッド >>
createTopicConnection()

createTopicConnection(String, String)

参照 :

- B-6 ページの「Oracle JMS クラス（パート 1）」を参照してください。

利用方法 :

- トピック・コネクションの確立 : オープンしている JDBC コネクションの使用
- トピック・コネクションの確立 : デフォルトのコネクション・ファクトリ・パラメータの使用

例外 - oracle.jms.AQjmsException

<< 例外 >>
oracle.jms.AQjmsException

<< メソッド >>
getErrorNumber()

参照 :

- B-21 ページの「Oracle JMS クラス（パート 10）」を参照してください。

利用方法 :

- JMS 例外のエラー番号の取得

例外 - oracle.jms.AQjmsInvalidDestinationException

<< 例外 >>
oracle.jms.AQjmsInvalidDestination-Exception

参照 : B-22 ページの「Oracle JMS クラス（パート 10 の続き）」を参照してください。

例外 - oracle.jms.AQjmsInvalidSelectorException

<< 例外 >>
oracle.jms.AQjmsInvalidSelector-Exception

> **参照：** B-22 ページの「Oracle JMS クラス（パート 10 の続き）」を参照してください。

例外 - oracle.jms.AQjmsMessageEOFException

<< 例外 >>
oracle.jms.AQjmsMessageEOF-Exception

> **参照：** B-22 ページの「Oracle JMS クラス（パート 10 の続き）」を参照してください。

例外 - oracle.jms.AQjmsMessageFormatException

<< 例外 >>
oracle.jms.AQjmsMessageFormatException

> **参照：** B-21 ページの「Oracle JMS クラス（パート 10）」を参照してください。

例外 - oracle.jms.AQjmsMessageNotReadableException

<< 例外 >>
oracle.jms.AQjmsMessageNotReadableException

> **参照：** B-21 ページの「Oracle JMS クラス（パート 10）」を参照してください。

例外 - oracle.jms.AQjmsMesssageNotWriteableException

<< 例外 >>
oracle.jms.AQjmsMessageNotWriteableException

> **参照：** B-21 ページの「Oracle JMS クラス（パート 10）」を参照してください。

インタフェース - oracle.AQ.AQQueueTable

<< インタフェース >>
oracle.AQ.AQQueueTable

<< メソッド >>
alter(java.lang.String comment)

alter(java.lang.String comment, int primary_instance, int secondary_instance)

drop(boolean force)

getName()

getOwner()

getProperty()

参照： B-19 ページの「Oracle JMS クラス（パート 9）」を参照してください。

クラス - oracle.AQ.AQQueueTableProperty

<< クラス >>
oracle.AQ.AQQueueTableProperty

<< 定数 >>
NONE

TRANSACTIONAL

<< コンストラクタ >>
AQQueueTableProperty(java.lang.String p_type)

<< メソッド >>
getComment()

getCompatible()

getMessageGrouping()

getPayloadType()

getPrimaryInstance()

getSecondaryInstance()

getSortOrder()

isMulticonsumerEnabled()

setComment(java.lang.String qt_comment)

setCompatible(java.lang.String qt_compatible)

setMessageGrouping(int message_grouping)

setMultiConsumer(boolean enable)

setPayloadType(java.lang.String p_type)

setPrimaryInstance(int inst)

setSecondaryInstance(int inst)

setSortOrder(java.lang.String s_order)

setStorageClause(java.lang.String s_clause)

toString()

参照：

- B-19 ページの「Oracle JMS クラス（パート 9）」を参照してください。

利用方法：

- キュー表の作成（キュー表のプロパティの指定）

インタフェース - oracle.jms.TopicBrowser

<< インタフェース >>
oracle.jms.TopicBrowser

<< メソッド >>
close()

get Enumeration()

getTopic()

getMessageSelector()

purgeSeen()

参照： B-19 ページの「Oracle JMS クラス（パート 9）」を参照してください。

クラス - oracle.jms.AQjmsTopicBrowser

<< クラス >>
oracle.jms.AQjmsTopicBrowser

> **参照：** B-19 ページの「Oracle JMS クラス（パート 9）」を参照してください。

C

BooksOnLine 用スクリプト

この付録には、次のスクリプトが掲載されています。

- tkaqdoca.sql: ユーザー、オブジェクト、キュー表、キューおよびサブスクライバ作成用のスクリプト
- tkaqdocd.sql: 管理インタフェースおよび操作インタフェースの例
- tkaqdoce.sql: 操作例
- tkaqdocp.sql: 操作インタフェースの例
- tkaqdocc.sql: クリーンアップ・スクリプト

tkaqdoca.sql: ユーザー、オブジェクト、キュー表、キューおよびサブスクライバ作成用のスクリプト

```
Rem $Header: tkaqdoca.sql 26-jan-99.17:50:37 aquser1 Exp $
Rem
Rem tkaqdoca.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998, 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      tkaqdoca.sql - TKAQ DOCumentation Admin examples file

Rem Set up a queue admin account and individual accounts for each application
Rem
connect system/manager
set serveroutput on;
set echo on;

Rem Create a common admin account for all BooksOnLine applications
Rem
create user BOLADM identified by BOLADM;
grant connect, resource, aq_administrator_role to BOLADM;
grant execute on dbms_aq to BOLADM;
grant execute on dbms_aqadm to BOLADM;
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','BOLADM',FALSE);
execute dbms_aqadm.grant_system_privilege('DEQUEUE_ANY','BOLADM',FALSE);

Rem Create the application schemas and grant appropriate permission
Rem to all schemas

Rem Create an account for Order Entry
create user OE identified by OE;
grant connect, resource to OE;
grant execute on dbms_aq to OE;
grant execute on dbms_aqadm to OE;

Rem Create an account for WR Shipping
create user WS identified by WS;
grant connect, resource to WS;
grant execute on dbms_aq to WS;
grant execute on dbms_aqadm to WS;

Rem Create an account for ER Shipping
create user ES identified by ES;
grant connect, resource to ES;
grant execute on dbms_aq to ES;
grant execute on dbms_aqadm to ES;
```

```
Rem Create an account for Overseas Shipping
create user OS identified by OS;
grant connect, resource to OS;
grant execute on dbms_aq to OS;
grant execute on dbms_aqadm to OS;

Rem Create an account for Customer Billing
Rem Customer Billing, for security reason, has an admin schema that
Rem hosts all the queue tables and an application schema from where
Rem the application runs.
create user CBADM identified by CBADM;
grant connect, resource to CBADM;
grant execute on dbms_aq to CBADM;
grant execute on dbms_aqadm to CBADM;

create user CB identified by CB;
grant connect, resource to CB;
grant execute on dbms_aq to CB;
grant execute on dbms_aqadm to CB;

Rem Create an account for Customer Service
create user CS identified by CS;
grant connect, resource to CS;
grant execute on dbms_aq to CS;
grant execute on dbms_aqadm to CS;


Rem All object types are created in the administrator schema.
Rem All application schemas that host any propagation source
Rem queues are given the ENQUEUE_ANY system level privilege
Rem allowing the application schemas to enqueue to the destination
Rem queue.
Rem
connect BOLADM/BOLADM;

Rem Create objects

create or replace type customer_typ as object (
        custno          number,
        name            varchar2(100),
        street          varchar2(100),
        city            varchar2(30),
        state           varchar2(2),
        zip             number,
        country         varchar2(100));
/
```

```
create or replace type book_typ as object (
        title           varchar2(100),
        authors         varchar2(100),
        ISBN            number,
        price           number);
/

create or replace type orderitem_typ as object (
        quantity        number,
        item            book_typ,
        subtotal        number);
/

create or replace type orderitemlist_vartyp as varray (20) of orderitem_typ;
/

create or replace type order_typ as object (
        orderno         number,
        status          varchar2(30),
        ordertype       varchar2(30),
        orderregion     varchar2(30),
        customer        customer_typ,
        paymentmethod   varchar2(30),
        items           orderitemlist_vartyp,
        total           number);
/

grant execute on order_typ to OE;
grant execute on orderitemlist_vartyp to OE;
grant execute on orderitem_typ to OE;
grant execute on book_typ to OE;
grant execute on customer_typ to OE;
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','OE',FALSE);

grant execute on order_typ to WS;
grant execute on orderitemlist_vartyp to WS;
grant execute on orderitem_typ to WS;
grant execute on book_typ to WS;
grant execute on customer_typ to WS;
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','WS',FALSE);

grant execute on order_typ to ES;
grant execute on orderitemlist_vartyp to ES;
grant execute on orderitem_typ to ES;
grant execute on book_typ to ES;
grant execute on customer_typ to ES;
```

```
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','ES',FALSE);

grant execute on order_typ to OS;
grant execute on orderitemlist_vartyp to OS;
grant execute on orderitem_typ to OS;
grant execute on book_typ to OS;
grant execute on customer_typ to OS;
execute dbms_aqadm.grant_system_privilege('ENQUEUE_ANY','OS',FALSE);

grant execute on order_typ to CBADM;
grant execute on orderitemlist_vartyp to CBADM;
grant execute on orderitem_typ to CBADM;
grant execute on book_typ to CBADM;
grant execute on customer_typ to CBADM;

grant execute on order_typ to CB;
grant execute on orderitemlist_vartyp to CB;
grant execute on orderitem_typ to CB;
grant execute on book_typ to CB;
grant execute on customer_typ to CB;

grant execute on order_typ to CS;
grant execute on orderitemlist_vartyp to CS;
grant execute on orderitem_typ to CS;
grant execute on book_typ to CS;
grant execute on customer_typ to CS;

Rem Create queue tables, queues for OE
Rem
connect OE/OE;
begin
dbms_aqadm.create_queue_table(
        queue_table => 'OE_orders_sqtab',
        comment => 'Order Entry Single Consumer Orders queue table',
        queue_payload_type => 'BOLADM.order_typ',
        message_grouping => DBMS_AQADM.TRANSACTIONAL,
        compatible => '8.1',
        primary_instance => 1,
        secondary_instance => 2);
end;
/

Rem Create a priority queue table for OE
begin
dbms_aqadm.create_queue_table(
        queue_table => 'OE_orders_pr_mqtab',
        sort_list =>'priority,enq_time',
```

```
        comment => 'Order Entry Priority MultiConsumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1',
        primary_instance => 2,
        secondary_instance => 1);
end;
/

begin
dbms_aqadm.create_queue (
        queue_name              => 'OE_neworders_que',
        queue_table             => 'OE_orders_sqtab');
end;
/

begin
dbms_aqadm.create_queue (
        queue_name              => 'OE_bookedorders_que',
        queue_table             => 'OE_orders_pr_mqtab');
end;
/

Rem Orders in OE_bookedorders_que are being propagated to WS_bookedorders_que,
Rem ES_bookedorders_que and OS_bookedorders_que according to the region
Rem the books are shipped to.  At the time an order is placed, the customer
Rem can request Fed-ex shipping (priority 1), priority air shipping (priority
Rem 2) and ground shipping (priority 3).  A priority queue is created in
Rem each region, the shipping applications will dequeue from these priority
Rem queues according to the orders' shipping priorities, processes the orders
Rem and enqueue the processed orders into
Rem the shipped_orders queues or the back_orders queues.  Both the shipped_
Rem orders queues and the back_orders queues are FIFO queues. However,
Rem orders put into the back_orders_queues are enqueued with delay time
Rem set to 1 day, so that each order in the back_order_queues is processed
Rem only once a day until the shipment is filled.

Rem Create queue tables, queues for WS Shipping
connect WS/WS;

Rem Create a priority queue table for WS shipping
begin
dbms_aqadm.create_queue_table(
        queue_table => 'WS_orders_pr_mqtab',
        sort_list =>'priority,enq_time',
        comment => 'West Shipping Priority MultiConsumer Orders queue table',
        multiple_consumers => TRUE,
```

```
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');
end;
/

Rem Create a FIFO queue tables for WS shipping
begin
dbms_aqadm.create_queue_table(
        queue_table => 'WS_orders_mqtab',
        comment => 'West Shipping Multi Consumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');
end;
/

Rem Booked orders are stored in the priority queue table
begin
dbms_aqadm.create_queue (
        queue_name              => 'WS_bookedorders_que',
        queue_table             => 'WS_orders_pr_mqtab');
end;
/

Rem Shipped orders and back orders are stored in the FIFO queue table
begin
dbms_aqadm.create_queue (
        queue_name              => 'WS_shippedorders_que',
        queue_table             => 'WS_orders_mqtab');
end;
/


begin
dbms_aqadm.create_queue (
        queue_name              => 'WS_backorders_que',
        queue_table             => 'WS_orders_mqtab');
end;
/

Rem
Rem  In order to test history, set retention to 1 DAY for the queues
Rem  in WS

begin
dbms_aqadm.alter_queue(
         queue_name => 'WS_bookedorders_que',
```

```
         retention_time => 86400);
end;
/

begin
dbms_aqadm.alter_queue(
         queue_name => 'WS_shippedorders_que',
         retention_time => 86400);
end;
/

begin
dbms_aqadm.alter_queue(
         queue_name => 'WS_backorders_que',
         retention_time => 86400);
end;
/


Rem Create queue tables, queues for ES Shipping
connect ES/ES;

Rem Create a priority queue table for ES shipping
begin
dbms_aqadm.create_queue_table(
        queue_table => 'ES_orders_mqtab',
        comment => 'East Shipping Multi Consumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');
end;
/

Rem Create a FIFO queue tables for ES shipping
begin
dbms_aqadm.create_queue_table(
        queue_table => 'ES_orders_pr_mqtab',
        sort_list =>'priority,enq_time',
        comment => 'East Shipping Priority Multi Consumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');
end;
/

Rem Booked orders are stored in the priority queue table
begin
```

```
dbms_aqadm.create_queue (
        queue_name              => 'ES_bookedorders_que',
        queue_table             => 'ES_orders_pr_mqtab');
end;
/

Rem Shipped orders and back orders are stored in the FIFO queue table
begin
dbms_aqadm.create_queue (
        queue_name              => 'ES_shippedorders_que',
        queue_table             => 'ES_orders_mqtab');
end;
/

begin
dbms_aqadm.create_queue (
        queue_name              => 'ES_backorders_que',
        queue_table             => 'ES_orders_mqtab');
end;
/


Rem Create queue tables, queues for Overseas Shipping
connect OS/OS;

Rem Create a priority queue table for OS shipping
begin
dbms_aqadm.create_queue_table(
        queue_table => 'OS_orders_pr_mqtab',
        sort_list =>'priority,enq_time',
        comment => 'Overseas Shipping Priority MultiConsumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');
end;
/

Rem Create a FIFO queue tables for OS shipping
begin
dbms_aqadm.create_queue_table(
        queue_table => 'OS_orders_mqtab',
        comment => 'Overseas Shipping Multi Consumer Orders queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');
end;
/
```

```
Rem Booked orders are stored in the priority queue table
begin
dbms_aqadm.create_queue (
        queue_name              => 'OS_bookedorders_que',
        queue_table             => 'OS_orders_pr_mqtab');
end;
/

Rem Shipped orders and back orders are stored in the FIFO queue table
begin
dbms_aqadm.create_queue (
        queue_name              => 'OS_shippedorders_que',
        queue_table             => 'OS_orders_mqtab');
end;
/

begin
dbms_aqadm.create_queue (
        queue_name              => 'OS_backorders_que',
        queue_table             => 'OS_orders_mqtab');
end;
/


Rem Create queue tables, queues for Customer Billing
connect CBADM/CBADM;
begin
dbms_aqadm.create_queue_table(
        queue_table => 'CBADM_orders_sqtab',
        comment => 'Customer Billing Single Consumer Orders queue table',
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');

dbms_aqadm.create_queue_table(
        queue_table => 'CBADM_orders_mqtab',
        comment => 'Customer Billing Multi Consumer Service queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');

dbms_aqadm.create_queue (
        queue_name              => 'CBADM_shippedorders_que',
        queue_table             => 'CBADM_orders_sqtab');

end;
/
```

```
Rem Grant dequeue privilege on the shopped orders queue to the Customer Billing
Rem application.  The CB application retrieves shipped orders (not billed yet)
Rem from the shopped orders queue.
execute dbms_aqadm.grant_queue_privilege('DEQUEUE', 'CBADM_shippedorders_que', 'CB',
FALSE);

begin
dbms_aqadm.create_queue (
        queue_name              => 'CBADM_billedorders_que',
        queue_table             => 'CBADM_orders_mqtab');
end;
/

Rem Grant enqueue privilege on the billed orders queue to Customer Billing
Rem application.  The CB application is allowed to put billed orders into
Rem this queue.
execute dbms_aqadm.grant_queue_privilege('ENQUEUE', 'CBADM_billedorders_que', 'CB',
FALSE);


Rem Customer support tracks the state of the customer request in the system
Rem
Rem At any point, customer request can be in one of the following states
Rem  A. BOOKED  B. SHIPPED  C. BACKED   D. BILLED
Rem  Given the order number the customer support will return the state
Rem  the order is in. This state is maintained in the order_status_table

connect CS/CS;

CREATE TABLE Order_Status_Table(customer_order       boladm.order_typ,
                                status               varchar2(30));

Rem Create queue tables, queues for Customer Service

begin
dbms_aqadm.create_queue_table(
        queue_table => 'CS_order_status_qt',
        comment => 'Customer Status multi consumer queue table',
        multiple_consumers => TRUE,
        queue_payload_type => 'BOLADM.order_typ',
        compatible => '8.1');

dbms_aqadm.create_queue (
        queue_name              => 'CS_bookedorders_que',
        queue_table             => 'CS_order_status_qt');
```

```
dbms_aqadm.create_queue (
        queue_name              => 'CS_backorders_que',
        queue_table             => 'CS_order_status_qt');

dbms_aqadm.create_queue (
        queue_name              => 'CS_shippedorders_que',
        queue_table             => 'CS_order_status_qt');

dbms_aqadm.create_queue (
        queue_name              => 'CS_billedorders_que',
        queue_table             => 'CS_order_status_qt');

end;
/

Rem Create the Subscribers for OE queues
Rem Add the Subscribers for the OE booked_orders queue

connect OE/OE;

Rem Add a rule-based subscriber for West Shipping
Rem West Shipping handles Western region US orders
Rem Rush Western region orders are handled by East Shipping
declare
  subscriber     aq$_agent;
begin
  subscriber := aq$_agent('West_Shipping', 'WS.WS_bookedorders_que', null);
  dbms_aqadm.add_subscriber(queue_name => 'OE.OE_bookedorders_que',
                            subscriber => subscriber,
                            rule       => 'tab.user_data.orderregion = ''WESTERN''
AND tab.user_data.ordertype != ''RUSH''');
end;
/

Rem Add a rule-based subscriber for East Shipping
Rem East shipping handles all Eastern region orders
Rem East shipping also handles all US rush orders
declare
  subscriber     aq$_agent;
begin
  subscriber := aq$_agent('East_Shipping', 'ES.ES_bookedorders_que', null);
  dbms_aqadm.add_subscriber(queue_name => 'OE.OE_bookedorders_que',
                            subscriber => subscriber,
                            rule       => 'tab.user_data.orderregion = ''EASTERN''
OR (tab.user_data.ordertype = ''RUSH'' AND tab.user_data.customer.country = ''USA'')
');
end;
```

```
/

Rem Add a rule-based subscriber for Overseas Shipping
Rem Intl Shipping handles all non-US orders
declare
  subscriber     aq$_agent;
begin
  subscriber := aq$_agent('Overseas_Shipping', 'OS.OS_bookedorders_que', null);
  dbms_aqadm.add_subscriber(queue_name => 'OE.OE_bookedorders_que',
                            subscriber => subscriber,
                            rule       => 'tab.user_data.orderregion =
''INTERNATIONAL''');
end;
/

Rem Add the Customer Service order queues as a subscribers to the
Rem corresponding queues  in OrderEntry, Shipping and Billing

declare
  subscriber     aq$_agent;
begin
  /* Subscribe to the booked orders queue */
  subscriber := aq$_agent('BOOKED_ORDER', 'CS.CS_bookedorders_que', null);
  dbms_aqadm.add_subscriber(queue_name => 'OE.OE_bookedorders_que',
                            subscriber => subscriber);
end;
/


connect WS/WS;

declare
  subscriber     aq$_agent;
begin
  /* Subscribe to the WS back orders queue */
  subscriber := aq$_agent('BACK_ORDER', 'CS.CS_backorders_que', null);
  dbms_aqadm.add_subscriber(queue_name => 'WS.WS_backorders_que',
                            subscriber => subscriber);
end;
/

declare
  subscriber     aq$_agent;
begin
  /* Subscribe to the WS shipped orders queue */
  subscriber := aq$_agent('SHIPPED_ORDER', 'CS.CS_shippedorders_que', null);
  dbms_aqadm.add_subscriber(queue_name => 'WS.WS_shippedorders_que',
```

```
                              subscriber => subscriber);
end;
/

connect CBADM/CBADM;
declare
  subscriber     aq$_agent;
begin
  /* Subscribe to the BILLING billed orders queue */
  subscriber := aq$_agent('BILLED_ORDER', 'CS.CS_billedorders_que', null);
  dbms_aqadm.add_subscriber(queue_name => 'CBADM.CBADM_billedorders_que',
                              subscriber => subscriber);

end;
/


Rem
Rem BOLADM will Start all the queues
Rem
connect BOLADM/BOLADM
execute dbms_aqadm.start_queue(queue_name => 'OE.OE_neworders_que');
execute dbms_aqadm.start_queue(queue_name => 'OE.OE_bookedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'WS.WS_bookedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'WS.WS_shippedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'WS.WS_backorders_que');
execute dbms_aqadm.start_queue(queue_name => 'ES.ES_bookedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'ES.ES_shippedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'ES.ES_backorders_que');
execute dbms_aqadm.start_queue(queue_name => 'OS.OS_bookedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'OS.OS_shippedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'OS.OS_backorders_que');
execute dbms_aqadm.start_queue(queue_name => 'CBADM.CBADM_shippedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'CBADM.CBADM_billedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'CS.CS_bookedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'CS.CS_backorders_que');
execute dbms_aqadm.start_queue(queue_name => 'CS.CS_shippedorders_que');
execute dbms_aqadm.start_queue(queue_name => 'CS.CS_billedorders_que');

connect system/manager

Rem
Rem Start job_queue_processes to handle AQ propagation
Rem

alter system set job_queue_processes=4;
```

tkaqdocd.sql: 管理インタフェースおよび操作インタフェースの例

```
Rem
Rem $Header: tkaqdocd.sql 26-jan-99.17:51:23 aquser1 Exp $
Rem
Rem tkaqdocd.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998, 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      tkaqdocd.sql - <one-line expansion of the name>
Rem
Rem    DESCRIPTION
Rem      <short description of component this file declares/defines>
Rem
Rem    NOTES
Rem      <other useful comments, qualifications, etc.>
Rem
Rem


Rem
Rem  Schedule propagation for the shipping, billing, order entry queues
Rem

connect OE/OE;

execute dbms_aqadm.schedule_propagation(queue_name => 'OE.OE_bookedorders_que');

connect WS/WS;
execute dbms_aqadm.schedule_propagation(queue_name => 'WS.WS_backorders_que');
execute dbms_aqadm.schedule_propagation(queue_name => 'WS.WS_shippedorders_que');

connect CBADM/CBADM;
execute dbms_aqadm.schedule_propagation(queue_name => 'CBADM.CBADM_billedorders_
que');


Rem
Rem   Customer service application
Rem
Rem   This application monitors the status queue for messages and updates
Rem   the Order_Status table.


connect CS/CS
```

```
Rem
Rem  Dequeus messages from the 'queue' for 'consumer'

CREATE OR REPLACE PROCEDURE DEQUEUE_MESSAGE(
                          queue      IN   VARCHAR2,
                          consumer   IN   VARCHAR2,
                          message    OUT  BOLADM.order_typ)
IS

dopt                      dbms_aq.dequeue_options_t;
mprop                     dbms_aq.message_properties_t;
deq_msgid                 raw(16);
BEGIN
  dopt.dequeue_mode := dbms_aq.REMOVE;
  dopt.navigation := dbms_aq.FIRST_MESSAGE;
  dopt.consumer_name := consumer;

  dbms_aq.dequeue(
               queue_name => queue,
               dequeue_options => dopt,
               message_properties => mprop,
               payload => message,
               msgid => deq_msgid);
  commit;
END;
/



Rem
Rem  Updates the status of the order in the status table
Rem

CREATE OR REPLACE PROCEDURE update_status(
                                new_status    IN VARCHAR2,
                                order_msg    IN BOLADM.ORDER_TYP)
IS
 old_status    VARCHAR2(30);
 dummy         NUMBER;
BEGIN

  BEGIN
    /* query old status from the table */
    SELECT st.status INTO old_status from order_status_table st
       where st.customer_order.orderno = order_msg.orderno;

  /* Status can be 'BOOKED_ORDER', 'SHIPPED_ORDER', 'BACK_ORDER'
```

```
   *          and   'BILLED_ORDER'
   */

   IF new_status = 'SHIPPED_ORDER' THEN
      IF old_status = 'BILLED_ORDER' THEN
        return;             /* message about a previous state */
      END IF;
   ELSIF new_status = 'BACK_ORDER' THEN
      IF old_status = 'SHIPPED_ORDER' OR old_status = 'BILLED_ORDER' THEN
        return;             /* message about a previous state */
      END IF;
   END IF;

   /* update the order status  */
     UPDATE order_status_table st
        SET st.customer_order = order_msg, st.status = new_status
        where st.customer_order.orderno = order_msg.orderno;

   COMMIT;

  EXCEPTION
  WHEN OTHERS  THEN     /* change to no data found */
    /* first update for the order */
    INSERT INTO order_status_table(customer_order, status)
    VALUES (order_msg, new_status);
    COMMIT;

  END;
END;
/


Rem
Rem  Monitors the customer service queues for 'time' seconds
Rem

 CREATE OR REPLACE PROCEDURE MONITOR_STATUS_QUEUE(time  IN  NUMBER)
IS
  agent_w_message   aq$_agent;
  agent_list        dbms_aq.agent_list_t;
  wait_time         INTEGER := 120;
  no_message        EXCEPTION;
  pragma EXCEPTION_INIT(no_message, -25254);
  order_msg         boladm.order_typ;
  new_status        VARCHAR2(30);
  monitor           BOOLEAN := TRUE;
  begin_time        number;
```

```
  end_time          number;
BEGIN

 begin_time :=  dbms_utility.get_time;
 WHILE (monitor)
 LOOP
 BEGIN
  agent_list(1) := aq$_agent('BILLED_ORDER', 'CS_billedorders_que', NULL);
  agent_list(2) := aq$_agent('SHIPPED_ORDER', 'CS_shippedorders_que', NULL);
  agent_list(3) := aq$_agent('BACK_ORDER', 'CS_backorders_que', NULL);
  agent_list(4) := aq$_agent('Booked_ORDER', 'CS_bookedorders_que', NULL);

   /* wait for order status messages */
   dbms_aq.listen(agent_list, wait_time, agent_w_message);

   dbms_output.put_line('Agent' || agent_w_message.name || ' Address '|| agent_w_
message.address);
   /* dequeue the message from the queue */
   dequeue_message(agent_w_message.address, agent_w_message.name, order_msg);

   /* update the status of the order depending on the type of the message
    * the name of the agent contains the new state
    */
   update_status(agent_w_message.name, order_msg);

  /* exit if we have been working long enough */
   end_time := dbms_utility.get_time;
   IF  (end_time - begin_time > time)   THEN
     EXIT;
   END IF;

  EXCEPTION
  WHEN  no_message  THEN
    dbms_output.put_line('No messages in the past 2 minutes');
       end_time := dbms_utility.get_time;
    /* exit if we have done enough work */
    IF  (end_time - begin_time > time)   THEN
      EXIT;
    END IF;
  END;

  END LOOP;
END;
/
```

```
Rem
Rem   History queries
Rem

Rem
Rem   Average processing time for messages in western shipping:
Rem   Difference between the ship- time and book-time for the order
Rem
Rem   NOTE: we assume that order id is the correlation identifier
Rem         Only processed messages are considered.

Connect WS/WS


SELECT  SUM(SO.enq_time - BO.enq_time) / count (*) AVG_PRCS_TIME
 FROM WS.AQ$WS_orders_pr_mqtab BO , WS.AQ$WS_orders_mqtab SO
 WHERE SO.msg_state = 'PROCESSED' and BO.msg_state = 'PROCESSED'
 AND SO.corr_id = BO.corr_id and SO.queue = 'WS_shippedorders_que';


Rem
Rem  Average backed up time (again only processed messages are considered
Rem

SELECT SUM(BACK.deq_time - BACK.enq_time)/count (*) AVG_BACK_TIME
  FROM WS.AQ$WS_orders_mqtab BACK
  WHERE BACK.msg_state = 'PROCESSED' and BACK.queue = 'WS_backorders_que';
```

tkaqdoce.sql: 操作例

```
Rem
Rem $Header: tkaqdoce.sql 26-jan-99.17:51:28 aquser1 Exp $
Rem
Rem tkaqdocl.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998, 1999. All Rights Reserved.
Rem


set echo on

Rem ================================================================
Rem        Demonstrate enqueuing a backorder with delay time set
Rem        to 1 day.  This will guarantee that each backorder will
Rem        be processed only once a day until the order is filled.
Rem ================================================================

Rem Create a package that enqueue with delay set to one day
connect BOLADM/BOLADM
create or replace procedure requeue_unfilled_order(sale_region varchar2,
                                                   backorder order_typ)
as
  back_order_queue_name    varchar2(62);
  enqopt                   dbms_aq.enqueue_options_t;
  msgprop                  dbms_aq.message_properties_t;
  enq_msgid                raw(16);
begin
  -- Choose a back order queue based the the region
  IF sale_region = 'WEST' THEN
    back_order_queue_name := 'WS.WS_backorders_que';
  ELSIF sale_region = 'EAST' THEN
    back_order_queue_name := 'ES.ES_backorders_que';
  ELSE
    back_order_queue_name := 'OS.OS_backorders_que';
  END IF;

  -- Enqueue the order with delay time set to 1 day
  msgprop.delay := 60*60*24;
  dbms_aq.enqueue(back_order_queue_name, enqopt, msgprop,
                  backorder, enq_msgid);
end;
```

tkaqdocp.sql: 操作インタフェースの例

```
Rem
Rem $Header: tkaqdocp.sql 26-jan-99.17:50:54 aquser1 Exp $
Rem
Rem tkaqdocp.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998, 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      tkaqdocp.sql - <one-line expansion of the name>
Rem

set echo on;

Rem ================================================================
Rem         Illustrating Support for Real Application Clusters
Rem ================================================================

Rem Login into OE account
connect OE/OE;
set serveroutput on;

Rem check instance affinity of OE queue tables from AQ administrative view

select queue_table, primary_instance, secondary_instance, owner_instance
from user_queue_tables;

Rem alter instance affinity of OE queue tables

begin
dbms_aqadm.alter_queue_table(
        queue_table => 'OE.OE_orders_sqtab',
        primary_instance => 2,
        secondary_instance => 1);
end;
/

begin
dbms_aqadm.alter_queue_table(
        queue_table => 'OE.OE_orders_pr_mqtab',
        primary_instance => 1,
        secondary_instance => 2);
end;
/

Rem check instance affinity of OE queue tables from AQ administrative view
```

```
select queue_table, primary_instance, secondary_instance, owner_instance
from user_queue_tables;

Rem ================================================================
Rem                    Illustrating Propagation Scheduling
Rem ================================================================

Rem Login into OE account

set echo on;
connect OE/OE;
set serveroutput on;

Rem
Rem Schedule Propagation from bookedorders_que to shipping
Rem

execute dbms_aqadm.schedule_propagation(queue_name => 'OE.OE_bookedorders_que');

Rem Login into boladm account
set echo on;
connect boladm/boladm;
set serveroutput on;

Rem create a procedure to enqueue an order
create or replace procedure order_enq(book_title   in varchar2,
                                      book_qty     in number,
                                      order_num    in number,
                                      shipping_priority in number,
                                      cust_state   in varchar2,
                                      cust_country in varchar2,
                                      cust_region  in varchar2,
                                      cust_ord_typ in varchar2) as

OE_enq_order_data        BOLADM.order_typ;
OE_enq_cust_data         BOLADM.customer_typ;
OE_enq_book_data         BOLADM.book_typ;
OE_enq_item_data         BOLADM.orderitem_typ;
OE_enq_item_list         BOLADM.orderitemlist_vartyp;
enqopt                   dbms_aq.enqueue_options_t;
msgprop                  dbms_aq.message_properties_t;
enq_msgid                raw(16);

begin

        msgprop.correlation := cust_ord_typ;
```

```
        OE_enq_cust_data := BOLADM.customer_typ(NULL, NULL, NULL, NULL,
                                cust_state, NULL, cust_country);
        OE_enq_book_data := BOLADM.book_typ(book_title, NULL, NULL, NULL);
        OE_enq_item_data := BOLADM.orderitem_typ(book_qty,
                                OE_enq_book_data, NULL);
        OE_enq_item_list := BOLADM.orderitemlist_vartyp(
                                BOLADM.orderitem_typ(book_qty,
                                OE_enq_book_data, NULL));
        OE_enq_order_data := BOLADM.order_typ(order_num, NULL,
                                cust_ord_typ, cust_region,
                                OE_enq_cust_data, NULL,
                                OE_enq_item_list, NULL);

        -- Put the shipping priority into message property before
        -- enqueueing the message
        msgprop.priority := shipping_priority;
        dbms_aq.enqueue('OE.OE_bookedorders_que', enqopt, msgprop,
                        OE_enq_order_data, enq_msgid);
end;
/

show errors;

grant execute on order_enq to OE;

Rem now create a procedure to dequeue booked orders for shipment processing
create or replace procedure shipping_bookedorder_deq(
                                        consumer in varchar2,
                                        deqmode in binary_integer) as

deq_cust_data           BOLADM.customer_typ;
deq_book_data           BOLADM.book_typ;
deq_item_data           BOLADM.orderitem_typ;
deq_msgid               RAW(16);
dopt                    dbms_aq.dequeue_options_t;
mprop                   dbms_aq.message_properties_t;
deq_order_data          BOLADM.order_typ;
qname                   varchar2(30);
no_messages             exception;
pragma exception_init   (no_messages, -25228);
new_orders              BOOLEAN := TRUE;

begin

        dopt.consumer_name := consumer;
        dopt.wait := DBMS_AQ.NO_WAIT;
        dopt.dequeue_mode := deqmode;
```

```
        dopt.navigation := dbms_aq.FIRST_MESSAGE;

        IF (consumer = 'West_Shipping') THEN
                qname := 'WS.WS_bookedorders_que';
        ELSIF (consumer = 'East_Shipping') THEN
                qname := 'ES.ES_bookedorders_que';
        ELSE
                qname := 'OS.OS_bookedorders_que';
        END IF;

        WHILE (new_orders) LOOP
          BEGIN
            dbms_aq.dequeue(
                queue_name => qname,
                dequeue_options => dopt,
                message_properties => mprop,
                payload => deq_order_data,
                msgid => deq_msgid);

            deq_item_data := deq_order_data.items(1);
            deq_book_data := deq_item_data.item;
            deq_cust_data := deq_order_data.customer;

            dbms_output.put_line(' **** next booked order **** ');
            dbms_output.put_line('order_num: ' || deq_order_data.orderno ||
                        ' book_title: ' || deq_book_data.title ||
                        ' quantity: ' || deq_item_data.quantity);
            dbms_output.put_line('ship_state: ' || deq_cust_data.state ||
                        ' ship_country: ' || deq_cust_data.country ||
                        ' ship_order_type: ' || deq_order_data.ordertype);
            dopt.navigation := dbms_aq.NEXT_MESSAGE;
          EXCEPTION
            WHEN no_messages THEN
                 dbms_output.put_line (' ---- NO MORE BOOKED ORDERS ---- ');
                 new_orders := FALSE;
          END;
        END LOOP;

end;
/
show errors;

Rem now create a procedure to dequeue rush orders for shipment
create or replace procedure get_rushtitles(consumer in varchar2) as

deq_cust_data            BOLADM.customer_typ;
deq_book_data            BOLADM.book_typ;
```

```
deq_item_data            BOLADM.orderitem_typ;
deq_msgid                RAW(16);
dopt                     dbms_aq.dequeue_options_t;
mprop                    dbms_aq.message_properties_t;
deq_order_data           BOLADM.order_typ;
qname                    varchar2(30);
no_messages              exception;
pragma exception_init    (no_messages, -25228);
new_orders               BOOLEAN := TRUE;

begin

        dopt.consumer_name := consumer;
        dopt.wait := 1;
        dopt.correlation := 'RUSH';

        IF (consumer = 'West_Shipping') THEN
                qname := 'WS.WS_bookedorders_que';
        ELSIF (consumer = 'East_Shipping') THEN
                qname := 'ES.ES_bookedorders_que';
        ELSE
                qname := 'OS.OS_bookedorders_que';
        END IF;

        WHILE (new_orders) LOOP
          BEGIN
            dbms_aq.dequeue(
                queue_name => qname,
                dequeue_options => dopt,
                message_properties => mprop,
                payload => deq_order_data,
                msgid => deq_msgid);

            deq_item_data := deq_order_data.items(1);
            deq_book_data := deq_item_data.item;

            dbms_output.put_line(' rushorder book_title: ' ||
                                 deq_book_data.title ||
                        ' quantity: ' || deq_item_data.quantity);
          EXCEPTION
            WHEN no_messages THEN
                 dbms_output.put_line (' ---- NO MORE RUSH TITLES ---- ');
                 new_orders := FALSE;
          END;
        END LOOP;

end;
```

```
/
show errors;

Rem now create a procedure to dequeue orders for handling North American
Rem orders
create or replace procedure get_northamerican_orders as

deq_cust_data            BOLADM.customer_typ;
deq_book_data            BOLADM.book_typ;
deq_item_data            BOLADM.orderitem_typ;
deq_msgid                RAW(16);
dopt                     dbms_aq.dequeue_options_t;
mprop                    dbms_aq.message_properties_t;
deq_order_data           BOLADM.order_typ;
deq_order_nodata         BOLADM.order_typ;
qname                    varchar2(30);
no_messages              exception;
pragma exception_init    (no_messages, -25228);
new_orders               BOOLEAN := TRUE;

begin

        dopt.consumer_name := 'Overseas_Shipping';
        dopt.wait := DBMS_AQ.NO_WAIT;
        dopt.navigation := dbms_aq.FIRST_MESSAGE;
        dopt.dequeue_mode := DBMS_AQ.LOCKED;

        qname := 'OS.OS_bookedorders_que';

        WHILE (new_orders) LOOP
          BEGIN
            dbms_aq.dequeue(
                queue_name => qname,
                dequeue_options => dopt,
                message_properties => mprop,
                payload => deq_order_data,
                msgid => deq_msgid);

            deq_item_data := deq_order_data.items(1);
            deq_book_data := deq_item_data.item;
            deq_cust_data := deq_order_data.customer;

            IF (deq_cust_data.country = 'Canada' OR
                deq_cust_data.country = 'Mexico' ) THEN

                dopt.dequeue_mode := dbms_aq.REMOVE_NODATA;
                dopt.msgid := deq_msgid;
```

```
                dbms_aq.dequeue(
                        queue_name => qname,
                        dequeue_options => dopt,
                        message_properties => mprop,
                        payload => deq_order_nodata,
                        msgid => deq_msgid);

                dbms_output.put_line(' **** next booked order **** ');
                dbms_output.put_line('order_no: ' || deq_order_data.orderno ||
                        ' book_title: ' || deq_book_data.title ||
                        ' quantity: ' || deq_item_data.quantity);
                dbms_output.put_line('ship_state: ' || deq_cust_data.state ||
                        ' ship_country: ' || deq_cust_data.country ||
                        ' ship_order_type: ' || deq_order_data.ordertype);

              END IF;

              commit;
              dopt.dequeue_mode := DBMS_AQ.LOCKED;
              dopt.msgid := NULL;
              dopt.navigation := dbms_aq.NEXT_MESSAGE;
            EXCEPTION
              WHEN no_messages THEN
                   dbms_output.put_line (' ---- NO MORE BOOKED ORDERS ---- ');
                   new_orders := FALSE;
            END;
          END LOOP;

end;
/
show errors;

grant execute on shipping_bookedorder_deq to WS;
grant execute on shipping_bookedorder_deq to ES;
grant execute on shipping_bookedorder_deq to OS;
grant execute on shipping_bookedorder_deq to CS;

grant execute on get_rushtitles to ES;

grant execute on get_northamerican_orders to OS;

Rem Login into OE account
connect OE/OE;
set serveroutput on;

Rem
Rem Enqueue some orders into OE_bookedorders_que
```

```
Rem

execute BOLADM.order_enq('My First   Book', 1, 1001, 3,'CA', 'USA', 'WESTERN',
'NORMAL');
execute BOLADM.order_enq('My Second  Book', 2, 1002, 3,'NY', 'USA', 'EASTERN',
'NORMAL');
execute BOLADM.order_enq('My Third   Book', 3, 1003, 3, '',   'Canada',
'INTERNATIONAL', 'NORMAL');
execute BOLADM.order_enq('My Fourth  Book', 4, 1004, 2, 'NV', 'USA', 'WESTERN',
'RUSH');
execute BOLADM.order_enq('My Fifth   Book', 5, 1005, 2, 'MA', 'USA', 'EASTERN',
'RUSH');
execute BOLADM.order_enq('My Sixth   Book', 6, 1006, 3,''  , 'UK',  'INTERNATIONAL',
'NORMAL');
execute BOLADM.order_enq('My Seventh Book', 7, 1007, 1,'',   'Canada',
'INTERNATIONAL', 'RUSH');
execute BOLADM.order_enq('My Eighth  Book', 8, 1008, 3,'',   'Mexico',
'INTERNATIONAL', 'NORMAL');
execute BOLADM.order_enq('My Ninth   Book', 9, 1009, 1, 'CA', 'USA', 'WESTERN',
'RUSH');
execute BOLADM.order_enq('My Tenth   Book', 8, 1010, 3, ''  , 'UK', 'INTERNATIONAL',
'NORMAL');
execute BOLADM.order_enq('My Last    Book', 7, 1011, 3, ''  , 'Mexico',
'INTERNATIONAL', 'NORMAL');
commit;
/

Rem
Rem Wait for Propagation to Complete
Rem

execute dbms_lock.sleep(100);

Rem ================================================================
Rem                  Illustrating Dequeue Modes/Methods
Rem ================================================================


connect WS/WS;
set serveroutput on;

Rem Dequeue all booked orders for West_Shipping
execute BOLADM.shipping_bookedorder_deq('West_Shipping', DBMS_AQ.REMOVE);
commit;
/

connect ES/ES;
```

```
set serveroutput on;

Rem Browse all booked orders for East_Shipping
execute BOLADM.shipping_bookedorder_deq('East_Shipping', DBMS_AQ.BROWSE);

Rem Dequeue all rush order titles for East_Shipping
execute BOLADM.get_rushtitles('East_Shipping');
commit;
/

Rem Dequeue all remaining booked orders (normal order) for East_Shipping
execute BOLADM.shipping_bookedorder_deq('East_Shipping', DBMS_AQ.REMOVE);
commit;
/

connect OS/OS;
set serveroutput on;

Rem Dequeue all international North American orders for Overseas_Shipping
execute BOLADM.get_northamerican_orders;
commit;
/

Rem Dequeue rest of the booked orders for Overseas_Shipping
execute BOLADM.shipping_bookedorder_deq('Overseas_Shipping', DBMS_AQ.REMOVE);
commit;
/


Rem ================================================================
Rem           Illustrating Enhanced Propagation Capabilities
Rem ================================================================

connect OE/OE;
set serveroutput on;

Rem
Rem Get propagation schedule information & statistics
Rem

Rem get averages
select avg_time, avg_number, avg_size from user_queue_schedules;

Rem get totals
select total_time, total_number, total_bytes from user_queue_schedules;

Rem get status information of schedule (present only when active)
```

```
select process_name, session_id, instance, schedule_disabled
        from user_queue_schedules;

Rem get information about last and next execution
select last_run_date, last_run_time, next_run_date, next_run_time
        from user_queue_schedules;

Rem get last error information if any
select failures, last_error_msg, last_error_date, last_error_time
        from user_queue_schedules;

Rem disable propagation schedule for booked orders

execute dbms_aqadm.disable_propagation_schedule(queue_name => 'OE_bookedorders_
que');
execute dbms_lock.sleep(30);
select schedule_disabled from user_queue_schedules;

Rem alter propagation schedule for booked orders to execute every
Rem 15 mins (900 seconds) for a window duration of 300 seconds

begin
dbms_aqadm.alter_propagation_schedule(
        queue_name => 'OE_bookedorders_que',
        duration => 300,
        next_time => 'SYSDATE + 900/86400',
        latency => 25);
end;
/

execute dbms_lock.sleep(30);
select next_time, latency, propagation_window from user_queue_schedules;

Rem enable propagation schedule for booked orders

execute dbms_aqadm.enable_propagation_schedule(queue_name => 'OE_bookedorders_que');
execute dbms_lock.sleep(30);
select schedule_disabled from user_queue_schedules;

Rem unschedule propagation for booked orders

execute dbms_aqadm.unschedule_propagation(queue_name => 'OE.OE_bookedorders_que');

set echo on;

Rem =====================================================================
Rem                       Illustrating Message Grouping
```

```
Rem ================================================================

Rem Login into boladm account
set echo on;
connect boladm/boladm;
set serveroutput on;

Rem now create a procedure to handle order entry
create or replace procedure new_order_enq(book_title   in varchar2,
                                          book_qty     in number,
                                          order_num    in number,
                                          cust_state   in varchar2) as

OE_enq_order_data        BOLADM.order_typ;
OE_enq_cust_data         BOLADM.customer_typ;
OE_enq_book_data         BOLADM.book_typ;
OE_enq_item_data         BOLADM.orderitem_typ;
OE_enq_item_list         BOLADM.orderitemlist_vartyp;
enqopt                   dbms_aq.enqueue_options_t;
msgprop                  dbms_aq.message_properties_t;
enq_msgid                raw(16);

begin

        OE_enq_cust_data := BOLADM.customer_typ(NULL, NULL, NULL, NULL,
                                cust_state, NULL, NULL);
        OE_enq_book_data := BOLADM.book_typ(book_title, NULL, NULL, NULL);
        OE_enq_item_data := BOLADM.orderitem_typ(book_qty,
                                OE_enq_book_data, NULL);
        OE_enq_item_list := BOLADM.orderitemlist_vartyp(
                                BOLADM.orderitem_typ(book_qty,
                                OE_enq_book_data, NULL));
        OE_enq_order_data := BOLADM.order_typ(order_num, NULL,
                                NULL, NULL,
                                OE_enq_cust_data, NULL,
                                OE_enq_item_list, NULL);
        dbms_aq.enqueue('OE.OE_neworders_que', enqopt, msgprop,
                        OE_enq_order_data, enq_msgid);
end;
/
show errors;

Rem now create a procedure to handle order enqueue
create or replace procedure same_order_enq(book_title   in varchar2,
                                           book_qty     in number) as

OE_enq_order_data        BOLADM.order_typ;
```

```
OE_enq_book_data         BOLADM.book_typ;
OE_enq_item_data         BOLADM.orderitem_typ;
OE_enq_item_list         BOLADM.orderitemlist_vartyp;
enqopt                   dbms_aq.enqueue_options_t;
msgprop                  dbms_aq.message_properties_t;
enq_msgid                raw(16);

begin

        OE_enq_book_data := BOLADM.book_typ(book_title, NULL, NULL, NULL);
        OE_enq_item_data := BOLADM.orderitem_typ(book_qty,
                                OE_enq_book_data, NULL);
        OE_enq_item_list := BOLADM.orderitemlist_vartyp(
                                BOLADM.orderitem_typ(book_qty,
                                OE_enq_book_data, NULL));
        OE_enq_order_data := BOLADM.order_typ(NULL, NULL,
                                NULL, NULL,
                                NULL, NULL,
                                OE_enq_item_list, NULL);
        dbms_aq.enqueue('OE.OE_neworders_que', enqopt, msgprop,
                        OE_enq_order_data, enq_msgid);
end;
/
show errors;

grant execute on new_order_enq to OE;
grant execute on same_order_enq to OE;

Rem now create a procedure to get new orders by dequeuing
create or replace procedure get_new_orders as

deq_cust_data            BOLADM.customer_typ;
deq_book_data            BOLADM.book_typ;
deq_item_data            BOLADM.orderitem_typ;
deq_msgid                RAW(16);
dopt                     dbms_aq.dequeue_options_t;
mprop                    dbms_aq.message_properties_t;
deq_order_data           BOLADM.order_typ;
qname                    varchar2(30);
no_messages              exception;
end_of_group             exception;
pragma exception_init    (no_messages, -25228);
pragma exception_init    (end_of_group, -25235);
new_orders               BOOLEAN := TRUE;

begin
```

```
        dopt.wait := 1;
        dopt.navigation := DBMS_AQ.FIRST_MESSAGE;
        qname := 'OE.OE_neworders_que';
        WHILE (new_orders) LOOP
          BEGIN
            LOOP
                BEGIN
                    dbms_aq.dequeue(
                        queue_name => qname,
                        dequeue_options => dopt,
                        message_properties => mprop,
                        payload => deq_order_data,
                        msgid => deq_msgid);

                    deq_item_data := deq_order_data.items(1);
                    deq_book_data := deq_item_data.item;
                    deq_cust_data := deq_order_data.customer;

                    IF (deq_cust_data IS NOT NULL) THEN
                      dbms_output.put_line(' **** NEXT ORDER **** ');
                      dbms_output.put_line('order_num: ' ||
                                deq_order_data.orderno);
                      dbms_output.put_line('ship_state: ' ||
                                deq_cust_data.state);
                    END IF;
                    dbms_output.put_line(' ---- next book ---- ');
                    dbms_output.put_line(' book_title: ' ||
                                deq_book_data.title ||
                                ' quantity: ' || deq_item_data.quantity);
                EXCEPTION
                    WHEN end_of_group THEN
                      dbms_output.put_line ('*** END OF ORDER ***');
                      commit;
                      dopt.navigation := DBMS_AQ.NEXT_TRANSACTION;
                END;
            END LOOP;
          EXCEPTION
            WHEN no_messages THEN
                 dbms_output.put_line (' ---- NO MORE NEW ORDERS ---- ');
                 new_orders := FALSE;
          END;
        END LOOP;

end;
/

show errors;
```

```
grant execute on get_new_orders to OE;

Rem Login into OE account
connect OE/OE;
set serveroutput on;

Rem
Rem Enqueue some orders using message grouping into OE_neworders_que
Rem

Rem First Order
execute BOLADM.new_order_enq('My First   Book', 1, 1001, 'CA');
execute BOLADM.same_order_enq('My Second  Book', 2);
commit;
/

Rem Second Order
execute BOLADM.new_order_enq('My Third   Book', 1, 1002, 'WA');
commit;
/

Rem Third Order
execute BOLADM.new_order_enq('My Fourth  Book', 1, 1003, 'NV');
execute BOLADM.same_order_enq('My Fifth   Book', 3);
execute BOLADM.same_order_enq('My Sixth   Book', 2);
commit;
/

Rem Fourth Order
execute BOLADM.new_order_enq('My Seventh Book', 1, 1004, 'MA');
execute BOLADM.same_order_enq('My Eighth  Book', 3);
execute BOLADM.same_order_enq('My Ninth   Book', 2);
commit;
/

Rem
Rem Dequeue the neworders
Rem

execute BOLADM.get_new_orders;
```

tkaqdocc.sql: クリーンアップ・スクリプト

```
Rem
Rem $Header: tkaqdocc.sql 26-jan-99.17:51:05 aquser1 Exp $
Rem
Rem tkaqdocc.sql
Rem
Rem  Copyright (c) Oracle Corporation 1998, 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      tkaqdocc.sql - <one-line expansion of the name>
Rem

set echo on;
connect system/manager
set serveroutput on;

drop user WS cascade;
drop user ES cascade;
drop user OS cascade;
drop user CB cascade;
drop user CBADM cascade;
drop user CS cascade;
drop user OE cascade;
drop user boladm cascade;
```

D

JMS および AQ XML サーブレット・エラー・メッセージ

この付録では、トラブルシューティングのために、エラー・メッセージ、原因および処置のリストを掲載しています（*string* には文字列が入ります）。

JMS エラー・メッセージ

JMS-00101 無効な送達モードです : *string*

原因 : 送達モードがサポートされていません。

処置 : 有効な送達モードは、AQjmsConstants.PERSISTENT です。

JMS-00102 機能 : *string* はサポートされません。

原因 : この機能は、今回のリリースではサポートされていません。

処置 : 処置はありません。

JMS-00104 メッセージ・ペイロードの指定が必要です。

原因 : メッセージ・ペイロードが NULL です。

処置 : メッセージに、NULL 以外のペイロードを指定してください。

JMS-00105 エージェントの指定が必要です。

原因 : AQjmsAgent オブジェクトが NULL です。

処置 : リモート・サブスクライバを表す有効な AQjmsAgent を指定してください。

JMS-00106 1 つの JMSConnection で複数のオープン・セッションを保持することはできません。

原因 : コネクション上に、開かれている JMS セッションがあります。コネクション上には、2 つ以上のセッションを開くことができません。

処置 : 開いているセッションを閉じてから、新しいセッションを開いてください。

JMS-00107 *string* では操作できません。

原因 : このオブジェクトでは使用できない操作が指定されています。

処置 : 処置はありません。

JMS-00108 型 : *string* のメッセージは、型 : *string* のペイロードを含む宛先に対しては使用できません。

原因 : 使用されているメッセージ型と、宛先に指定されているペイロードの型が一致していません。

処置 : この宛先を含むキュー表に指定されているペイロードに割り当てられているメッセージ型を使用してください。

JMS-00109 クラスが見つかりません : *string*

原因 : 指定されたクラスが見つかりません。

処置 : CLASSPATH に、指定されたクラスが含まれているかどうかを確認してください。

JMS-00110 優先順位 : *string* は書込み不可です。

原因 : 読込み専用メッセージのヘッダー・フィールドまたはプロパティを更新しようとしました。

処置 : 処置はありません。

JMS-00111 接続の指定が必要です。

原因 : コネクション・オブジェクトが NULL です。

処置 : NULL 以外の JDBC コネクションを指定してください。

JMS-00112 接続が無効です。

原因 : JDBC コネクションが無効です。

処置 : NULL 以外の Oracle JDBC コネクションを指定してください。

JMS-00113 接続が停止状態です。

原因 : 停止状態のコネクション上でメッセージを受信しようとしました。

処置 : コネクションを開始してください。

JMS-00114 接続はクローズされています。

原因 : クローズされたコネクションを使用しようとしました。

処置 : 新しいコネクションを確立してください。

JMS-00115 コンシューマはクローズされています。

原因 : クローズされたコンシューマを使用しようとしました。

処置 : 新しいメッセージ・コンシューマを作成してください。

JMS-00116 サブスクライバ名の指定が必要です。

原因 : サブスクライバ名が NULL です。

処置 : NULL 以外のサブスクリプション名を指定してください。

JMS-00117 変換に失敗 - プロパティの型が無効です。

原因 : 要求された型にプロパティを変換中に、エラーが発生しました。

処置 : プロパティのデータ型に対応したメソッドを使用して、プロパティを取り出してください。

JMS-00119 無効なプロパティ値です。

原因 : 無効なプロパティの値が指定されています。

処置 : 設定中のプロパティに、適切な型の値を使用してください。

JMS-00120 デキューに失敗しました。

原因 : メッセージの受信中にエラーが発生しました。

処置 : 詳細は、JMSException およびリンクされた SQLException 内のメッセージを参照してください。

JMS-00121 DestinationProperty の指定が必要です。

原因 : キュー / トピックの作成中に、NULL の AQjmsDestinationProperty が指定されました。

処置 : 宛先に、NULL 以外の AQjmsDestinationProperty を指定してください。

JMS-00122 内部エラー : *string*

原因 : 内部エラーが発生しました。

処置 : オラクル社カスタマ・サポート・センターに連絡してください。

JMS-00123 間隔は最低 *integer* 秒以上必要です。

原因 : 無効な間隔が指定されました。

処置 : 間隔は、30 秒より大きい値に指定してください。

JMS-00124 無効なデキュー・モードです。

原因 : 無効なデキュー・モードが指定されました。

処置 : 有効なデキュー・モードは、AQConstants.DEQUEUE_BROWSE、AQConstants.DEQUEUE_REMOVE、AQConstants.DEQUEUE_LOCKED および AQConstants.DEQUEUE_REMOVE_NODATA です。

JMS-00125 無効なキューが指定されました。

原因 : 無効なキュー・オブジェクトが指定されました。

処置 : 有効なキュー・ハンドルを指定してください。

JMS-00126 無効なトピックが指定されました。

原因 : 無効なトピック・オブジェクトが指定されました。

処置 : 有効なトピック・ハンドルを指定してください。

JMS-00127 無効な宛先です。

原因 : 無効な宛先オブジェクトが指定されました。

処置 : 有効な宛先（キュー / トピック）オブジェクトを指定してください。

JMS-00128 無効なナビゲーション・モードです。

原因 : 無効なナビゲーション・モードが指定されました。

処置 : 有効なナビゲーション・モードは、AQjmsConstants.NAVIGATION_FIRST_MESSAGE、AQjmsConstants.NAVIGATION_NEXT_MESSAGE および AQjmsConstants.NAVIGATION_NEXT_TRANSACTION です。

JMS-00129 無効なペイロード型です。

原因 : 使用されているメッセージ型と、宛先に指定されているペイロードの型が一致していません。

処置 : この宛先を含むキュー表に指定されたペイロードに割り当てられているメッセージ型を使用してください。ユーザー定義型メッセージでは、適切な CustomDatumFactory を使用して、メッセージ・コンシューマを作成してください。

JMS-00130 JMS キューはマルチ・コンシューマには使用できません。

原因 : AQ マルチ・コンシューマ・キューを、JMS キューとして取得しようとしました。

処置 : JMS キューでは、マルチ・コンシューマを使用できません。

JMS-00131 セッションはクローズされています。

原因 : 閉じられたセッションを使用しようとしました。

処置 : 新しいセッションを開いてください。

JMS-00132 プロパティの最大数 *integer* を超過しました。

原因 : メッセージに対するユーザー定義のプロパティの数が、最大値を超えました。

処置 : 処置はありません。

JMS-00133 メッセージの指定が必要です。

原因 : NULL のメッセージが指定されました。

処置 : NULL 以外のメッセージを指定してください。

JMS-00134 名前の指定が必要です。

原因 : NULL のキューまたはキュー表名が指定されました。

処置 : NULL 以外の名前を指定してください。

JMS-00135 ドライバ *string* はサポートされません。

原因 : サポートされていないドライバが指定されています。

処置 : 有効な JDBC ドライバは、OCI8 および Thin です。サーバー・ドライバを使用するには、getDefaultConnection() を使用してコネクションを取得し、static メソッドである createTopicConnection および createQueueConnection メソッドを使用してください。

JMS-00136 ペイロード・ファクトリは、ADT ペイロードを持つ宛先にのみ指定できます。

原因 : ユーザー定義型ペイロードを含まない宛先のコンシューマに、CustomDatumFactory が指定されました。

処置 : ペイロード型が SYS.AQ$_JMS_TEXT_MESSAGE、SYS.AQ$_JMS_BYTES_MESSAGE、SYS.AQ$_JMS_MAP_MESSAGE、SYS.AQ$_JMS_OBJECT_MESSAGE または SYS.AQ$_JMS_STREAM_MESSAGE の宛先では、このフィールドを NULL に設定してください。

JMS-00137 ペイロード・ファクトリは、ADT ペイロードを持つ宛先に対して指定する必要があります。

原因 : ユーザー定義型ペイロードを含む宛先に、CustomDatumFactory が指定されませんでした。

処置 : ユーザー定義型メッセージを含む宛先では、ユーザー定義型の宛先に割り当てる Java クラスに対して CustomDatumFactory を指定してください。

JMS-00138 プロデューサはクローズされています。

原因 : クローズされたプロデューサを使用しようとしました。

処置 : 新しいメッセージ・プロデューサを作成してください。

JMS-00139 プロパティ名の指定が必要です。

原因 : プロパティ名が NULL です。

処置 : NULL 以外のプロパティ名を指定してください。

JMS-00140 無効な System プロパティです。

原因 : 無効なシステム・プロパティ名が指定されました。

処置 : 有効な JMS システム・プロパティを指定してください。

JMS-00142 JMS 項目はマルチ・コンシューマ対応のキュー表で作成する必要があります。

原因 : シングル・コンシューマ用のキュー表内に、JMS トピックを作成しようとしました。

処置 : JMS トピックは、マルチ・コンシューマが使用可能なキュー表にのみ作成してください。

JMS-00143 キューの指定が必要です。

原因 : NULL のキューが指定されました。

処置 : NULL 以外のキューを指定してください。

JMS-00144 JMS キューはマルチ・コンシューマ対応のキュー表では作成できません。

原因 : マルチ・コンシューマ用のキュー表内に、JMS キューを作成しようとしました。

処置 : JMS キューは、マルチ・コンシューマが使用可能でないキュー表にのみ作成してください。

JMS-00145 無効なレシピエント・リストです。

原因 : 空の受信者リストが指定されました。

処置 : 1 つ以上の受信者を含む受信者リストを指定してください。

JMS-00146 登録に失敗しました。

原因 : 型をタイプ・マップに登録中に、エラーが発生しました。

処置 : 処置はありません。

JMS-00147 無効な ReplyTo 宛先タイプです。

原因 : ReplyTo 宛先のオブジェクト型が無効です。

処置 : ReplyTo 宛先の型は、AQjmsAgent にしてください。

JMS-00148 プロパティ名のサイズが制限を超えています。

原因 : プロパティ名が最大サイズを超えています。

処置 : プロパティ名は、100 文字未満で指定してください。

JMS-00149 サブスクライバの指定が必要です。

原因 : NULL のサブスクライバが指定されました。

処置 : NULL 以外のサブスクライバを指定してください。

JMS-00150 プロパティはサポートされません。

原因 : サポートされていないプロパティを使用しようとしました。

処置 : 処置はありません。

JMS-00151 項目の型は EXCEPTION にできません。

原因 : トピックは、AQjmsConstants.EXCEPTION 型にできません。

処置 : トピックが、AQjmsConstants.NORMAL 型になるように指定してください。

JMS-00153 System プロパティの型が無効です。

原因 : 指定された値の型が、設定中のシステム・プロパティに定義された型と一致していません。

処置 : システム・プロパティの設定と一致した型を使用してください。

JMS-00154 SEQUENCE DEVIATION の値が無効です。

原因 : 順序逸脱が無効です。

処置 : 有効な値は、AQEnqueueOption.DEVIATION_BEFORE および AQEnqueueOption.DEVIATION_TOP です。

JMS-00155 AQ 例外 : *string*

原因 : AQ Java レイヤーでエラーが発生しました。

処置 : 詳細は、JMSException およびリンクされた Exception 内のメッセージを参照してください。

JMS-00156 無効なクラス : *string*

原因 : 無効なクラスが指定されています。

処置 : `CLASSPATH` に、指定されたクラスが含まれているかどうかを確認してください。

JMS-00157 IO 例外 : *string*

原因 : I/O の例外です。

処置 : 詳細は、JMSException 内のメッセージを参照してください。

JMS-00158 SQL 例外 : *string*

原因 : SQL の例外です。

処置 : 詳細は、リンクされた SQLException 内のメッセージを参照してください。

JMS-00159 無効なセレクタ : *string*

原因 : 無効なセレクタまたは長すぎるセレクタが指定されています。

処置 : セレクタの構文を確認してください。

JMS-00160 EOF 例外 : *string*

原因 : バイト・ストリームの読込み中に EOF 例外が発生しました。

処置 : 処置はありません。

JMS-00161 メッセージ形式例外 : *string*

原因 : ストリーム・データを指定された型に変換中に、エラーが発生しました。

処置 : ストリーム上で予期されるデータの型を確認して、適切な読込みメソッドを使用してください。

JMS-00162 メッセージは読込み不可です。

原因 : メッセージが書込み専用モードです。

処置 : リセット・メソッドをコールして、メッセージを読込み可能にしてください。

JMS-00163 メッセージは書込み不可です。

原因 : メッセージが読込み専用モードです。

処置 : clearBody メソッドを使用して、メッセージを書込み可能にしてください。

JMS-00164 該当する要素はありません。

原因 : 指定された名前の要素が、MapMessage 内に見つかりません。

処置 : 処置はありません。

JMS-00165 プロパティ値が最大サイズを超えています。

原因 : プロパティの値が、最大長を超えています。

処置 : JMS 定義のプロパティの値の最大長は 100 です。ユーザー定義のプロパティの値の最大長は 2000 です。

JMS-00166 項目の指定が必要です。

原因 : NULL のトピックが指定されました。

処置 : NULL 以外のトピックを指定してください。

JMS-00167 ペイロード・ファクトリまたは Sql_data_class の指定が必要です。

原因 : オブジェクト・ペイロードを含むキューに、ペイロード・ファクトリまたは Sql_data_class が指定されていません。

処置 : CustomDatumFactory、またはキューに定義されたユーザー定義型にマップする Java オブジェクトの SQLData クラスを指定してください。

JMS-00168 ペイロード・ファクトリと sql_data_class の両方は指定できません。

原因 : デキュー中に、CustomDatumFactory および SQLData クラスが指定されました。

処置 : CustomDatumFactory、またはキューに定義されたユーザー定義型にマップする Java オブジェクトの SQLData クラスのいずれかを指定してください。

JMS-00169 Sql_data_class は NULL にできません。

原因 : NULL の SQLData クラスが指定されています。

処置 : キューに定義されたユーザー定義型にマップする SQLData クラスを指定してください。

JMS-00171 *string* を含んだメッセージが定義されていません。

原因 : メッセージ内のペイロード型が無効です。

処置 : キューが RAW または OBJECT ペイロードを含むように定義されているかどうかを確認し、メッセージ内に適切なペイロード型を使用してください。

JMS-00172 複数のキュー表が問合せに一致しています : *string*

原因 : 問合せに 2 つ以上のキュー表が一致します。

処置 : 所有者およびキュー表名を指定してください。

JMS-00173 キュー表 : *string* が見つかりません。

原因 : 指定されたキュー表が見つかりません。

処置 : 有効なキュー表を指定してください。

JMS-00174 オブジェクト・ペイロードとともにキューのクラスを指定する必要があります。dequeue(deq_option, payload_fact) または dequeue(deq_option, sql_data_cl) を使用してください。

原因 : このデキュー・メソッドは、OBJECT ペイロードを含むキューからのデキューには使用できません。

処置 : dequeue(deq_option, payload_fact) または dequeue(deq_option, sql_data_cl) のいずれかを使用してください。

JMS-00175 DequeueOption の指定が必要です。

原因 : NULL のデキュー・オプションが指定されています。

処置 : NULL 以外のデキュー・オプションを指定してください。

JMS-00176 EnqueueOption の指定が必要です。

原因 : NULL のエンキュー・オプションが指定されています。

処置 : NULL 以外のエンキュー・オプションを指定してください。

JMS-00177 無効なペイロード・タイプ : ロー・ペイロード・キューには dequeue(deq_option) を使用してください。

原因 : このメソッドは、ロー・ペイロードを含むキューからのデキューには使用できません。

処置 : dequeue(deq_option) メソッドを使用してください。

JMS-00178 無効なキュー名です - *string*

原因 : NULL または無効なキュー名が指定されています。

処置 : NULL 以外のキュー名を使用してください。キュー名はスキーマ名で修飾しないでください。スキーマ名は、owner パラメータの値として指定してください。

JMS-00179 無効な表名です - *string*

原因 : NULL または無効なキュー表名が指定されています。

処置 : NULL 以外のキュー表名を使用してください。キュー表名はスキーマ名で修飾しないでください。スキーマ名は、owner パラメータの値として指定してください。

JMS-00180 無効なキュー・タイプです。

原因 : キュー・タイプが無効です。

処置 : 有効なタイプは、AQConstants.NORMAL または AQConstants.EXCEPTION です。

JMS-00181 wait_time の値が無効です。

原因 : 待機時間の値が無効です。

処置 : 待機時間は、AQDequeueOption.WAIT_FOREVER、AQDequeueOption.WAIT_NONE または 0（ゼロ）より大きいすべての値に指定できます。

JMS-00182 問合せで複数のキューが一致しています。

原因 : 問合せに 2 つ以上のキューが一致します。

処置 : キューの所有者および名前を指定してください。

JMS-00183 AQ ドライバが登録されていません。

原因 : AQDriver が登録されていません。

処置 : AQ Java ドライバが登録されているかどうかを確認してください。Class.forName("oracle.AQ.AQOracleDriver") を使用してください。

JMS-00184 キュー・オブジェクトが無効です。

原因 : キュー・オブジェクトが無効です。

処置 : 基礎となる JDBC コネクションがクローズされている可能性があります。キュー・ハンドルを再度取得してください。

JMS-00185 QueueProperty の指定が必要です。

原因 : NULL の AQQueueProperty が指定されています。

処置 : NULL 以外の AQQueueProperty を指定してください。

JMS-00186 QueueTableProperty の指定が必要です。

原因 : NULL の QueueTableProperty が指定されています。

処置 : NULL 以外の AQQueueTableProperty を指定してください。

JMS-00187 キュー表の指定が必要です。

原因 : NULL のキュー表が指定されています。

処置 : NULL 以外のキュー表を指定してください。

JMS-00188 QueueTable オブジェクトが無効です。

原因 : QueueTable オブジェクトが無効です。

処置 : 基礎となる JDBC コネクションがクローズされている可能性があります。QueueTable ハンドルを再度取得してください。

JMS-00189 バイト配列が小さすぎます。

原因 : 指定されたバイト配列が小さすぎるため、要求されたデータを保持できません。

処置 : 要求されたデータを保持できる大きいバイト配列を指定するか、またはリクエストを短くしてください。

JMS-00190 キュー : *string* が見つかりません。

原因 : 指定されたキューが見つかりません。

処置 : 有効なキューを指定してください。

JMS-00191 sql_data_cl は SQLData インタフェースを実装しているクラスであることが必要です。

原因 : java.sql.SQLData インタフェースをサポートしないクラスが指定されています。

処置 : 処置はありません。

JMS-00192 無効な可視性値です。

原因 : 無効な値の可視性が指定されています。

処置 : 有効な値は、AQConstants.VISIBILITY_ONCOMMIT および AQConstants.VISIBILITY_IMMEDIATE です。

JMS-00193 JMS キューに RAW 型のペイロードを含めることはできません。

原因 : RAW ペイロードを含む JMS キューを作成しようとしました。

処置 : JMS キュー / トピックには、RAW ペイロードを含めないでください。

JMS-00194 セッション・オブジェクトが無効です。

原因 : セッション・オブジェクトが無効です。

処置 : 基礎となる JDBC コネクションがクローズされている可能性があります。新しいセッションを確立してください。

JMS-00195 無効なオブジェクト型 : オブジェクトは CustomDatum/ORAData または SQLData インタフェースを実装している必要があります。

原因 : 無効なオブジェクト型が指定されています。

処置 : オブジェクトには、CustomDatum または SQLData インタフェースを実装してください。

JMS-00196 1 つの JMS セッションで同じ宛先のオープン QueueBrowser を複数保持することはできません。

原因 : このセッションのこのキューでは、キュー・ブラウザがすでにオープンされています。

処置 : 特定のセッションの同じキューでは、2 つ以上のキュー・ブラウザをオープンできません。既存のキュー・ブラウザをクローズしてから、新しくオープンしてください。

JMS-00197 リモート・サブスクライバに対するエージェント・アドレスの指定が必要です。

原因 : リモート・サブスクライバのアドレス・フィールドが NULL です。

処置 : アドレス・フィールドには、完全に修飾されたリモート・トピック名を含めてください。

JMS-00198 無効な操作 : セッションに特権メッセージ・リスナーを設定しました。

原因 : セッション・メッセージ・リスナーが設定されたときに、クライアントがメッセージ・コンシューマを使用して、メッセージを受信しようとしました。

処置 : セッションのメッセージ・リスナーを使用して、メッセージを使用してください。コンシューマのメッセージ受信メソッドは、使用しないでください。

JMS-00199 通知の登録に失敗しました。

原因 : 通知の登録に失敗しました。

処置 : 詳細は、リンクされた Exception 内のエラー・メッセージを参照してください。

JMS-00200 宛先の指定が必要です。

原因 : 宛先が NULL です。

処置 : NULL 以外の宛先を指定してください。

JMS-00201 すべてのレシピエントを recipient_list に指定する必要があります。

原因 : 受信者リスト内の 1 つ以上の要素が NULL です。

処置 : 受信者リスト内のすべての AQjmsAgents を指定してください。

JMS-00202 メッセージの非同期受信の登録解除に失敗しました。

原因 : データベースを使用する非同期受信用のコンシューマの登録を削除するときに、エラーが発生しました。

処置 : 詳細は、リンクされた Exception 内のエラー・メッセージを参照してください。

JMS-00203 ペイロード・ファクトリの指定が必要です。

原因 : NULL のペイロード・ファクトリが指定されました。

処置 : NULL 以外のペイロード・ファクトリを指定してください。

JMS-00204 AQ JNI レイヤーでエラーが発生しました。

原因 : JNI エラーです。

処置 : 詳細は、リンクされた Exception 内のエラー・メッセージを参照してください。

JMS-00205 ネーミング例外

原因 : ネーミング例外です。

処置 : 詳細は、リンクされた Exception 内のエラー・メッセージを参照してください。

JMS-00206 XA 例外 XAError- *string* :: OracleError- *string*

原因 : XA レイヤーでエラーが発生しました。

処置 : 詳細は、リンクされた XAException 内のメッセージを参照してください。

JMS-00207 JMS 例外 *string*
原因 : JMS レイヤーでエラーが発生しました。
処置 : 詳細は、リンクされた JMSException 内のメッセージを参照してください。

JMS-00208 XML SQL 例外
原因 : XML SQL レイヤーでエラーが発生しました。
処置 : 詳細は、リンクされた AQxmlException 内のメッセージを参照してください。

JMS-00209 XML SAX 例外
原因 : XML SAX レイヤーでエラーが発生しました。
処置 : 詳細は、リンクされた AQxmlException 内のメッセージを参照してください。

JMS-00210 XML 解析例外
原因 : XML 解析レイヤーでエラーが発生しました。
処置 : 詳細は、リンクされた AQxmlException 内のメッセージを参照してください。

JMS-00220 接続できなくなりました。
原因 : データベースに接続できなくなりました。
コメント : このエラーは、データベース / ネットワーク / マシンのいずれかがアクセス不可である場合に発生する可能性があります。一時的な障害の可能性があります。

JMS-00221 接続プールに使用可能な物理データベース接続がありません。
原因 : 指定された操作を行うために必要な、使用可能な物理データベース接続が OCI コネクション・プールにありません。
処置 : 後で、操作を再実行してください。

AQ XML サーブレット・エラー・メッセージ

JMS-00400 宛先名を指定する必要があります。
原因 : NULL の宛先が指定されています。
処置 : NULL 以外の宛先所有者を指定してください。

JMS-00402 クラスが見つかりません : *string*
原因 : 指定されたクラスが見つかりません。
処置 : `CLASSPATH` に、指定されたクラスが含まれているかどうかを確認してください。

JMS-00403 IO 例外 : *string*
原因 : I/O の例外です。
処置 : 詳細は、リンクされた AQxmlException 内のメッセージを参照してください。

JMS-00404 XML 解析例外

原因 : XML 解析レイヤーでエラーが発生しました。

処置 : 詳細は、リンクされた AQxmlException 内のメッセージを参照してください。

JMS-00405 XML SAX 例外

原因 : XML SAX レイヤーでエラーが発生しました。

処置 : 詳細は、リンクされた AQxmlException 内のメッセージを参照してください。

JMS-00406 JMS 例外 *string*

原因 : JMS レイヤーでエラーが発生しました。

処置 : 詳細は、リンクされた JMSException 内のメッセージを参照してください。

JMS-00407 *string* では操作できません。

原因 : このオブジェクトでは使用できない操作が指定されています。

処置 : 操作を実行するユーザーが必要な権限を持っていることを確認してください。

JMS-00408 変換に失敗しました - 無効なプロパティ型です。

原因 : 要求された型にプロパティを変換中に、エラーが発生しました。

処置 : プロパティのデータ型に対応したメソッドを使用して、プロパティを取り出してください。

JMS-00409 そのような要素はありません。

原因 : 指定された名前の要素が、MapMessage 内に見つかりません。

処置 : 有効な要素名を指定してください。

JMS-00410 XML SQL 例外

原因 : JDBC SQL レイヤーでエラーが発生しました。

処置 : 詳細は、リンクされた SQLException 内のメッセージを参照してください。

JMS-00411 ペイロードの主要部は null にできません

原因 : 無効な本体文字列またはドキュメントが指定されました。

処置 : ペイロードに、NULL 以外の本体文字列またはドキュメントを指定してください。

JMS-00412 バイト変換に失敗しました。

原因 : 無効なユーザー名 / パスワードが指定されました。

処置 : NULL 以外のユーザー名およびパスワードを指定してください。

JMS-00413 操作には自動コミットできません。

原因 : この操作に自動コミット・フラグは設定できません。

処置 : 自動コミット・フラグを設定しないでください。

JMS-00414 宛先の所有者を指定する必要があります。

原因 : NULL の宛先所有者が指定されています。

処置 : NULL 以外の宛先所有者を指定してください。

JMS-00145 無効な可視性値です。

原因 : 無効な値の可視性が指定されています。

処置 : 有効な値は、AQxmlConstants.VISIBILITY_ONCOMMIT および AQxmlConstants.VISIBILITY_IMMEDIATE です。

JMS-00416 無効なデキュー・モードです。

原因 : 無効なデキュー・モードが指定されました。

処置 : 有効なデキュー・モードは、AQxmlConstants.DEQUEUE_BROWSE、AQxmlConstants.DEQUEUE_REMOVE、AQxmlConstants.DEQUEUE_LOCKED および AQxmlConstants.DEQUEUE_REMOVE_NODATA です。

JMS-00417 無効なナビゲーション・モードです。

原因 : 無効なナビゲーション・モードが指定されました。

処置 : 有効なナビゲーション・モードは、AQxmlConstants.NAVIGATION_FIRST_MESSAGE、AQxmlConstants.NAVIGATION_NEXT_MESSAGE および AQxmlConstants.NAVIGATION_NEXT_TRANSACTION です。

JMS-00418 wait_time の値が無効です。

原因 : 待機タイプの値が無効です。

処置 : 待機時間は、AQDequeueOption.WAIT_FOREVER、AQDequeueOption.WAIT_NONE または 0（ゼロ）より大きいすべての値に指定できます。

JMS-00419 ConnectionPoolDataSource の値が無効です。

原因 : NULL または無効な ConnectionPoolDataSource が指定されています。

処置 : 適切な URL、ユーザー名およびパスワードで有効な OracleConnectionPoolDataSource オブジェクト指定してください。

JMS-00420 cache_size の値が無効です。

原因 : 無効なキャッシュ・サイズが指定されました。

処置 : キャッシュ・サイズは、0 より大きい値に指定してください。

JMS-00421 cache_scheme の値が無効です。

原因 : 無効なキャッシュ・スキーマが指定されました。

処置 : 有効なキャッシュ・スキーマは、
OracleConnectionCacheImpl.DYNAMIC_SCHEME および
OracleConnectionCacheImpl.FIXED_WAIT_SCHEME です。

JMS-00422 無効なタグです - {0}

原因 : XML 文書内に無効なタグがありました。

処置 : XML 文書が AQ スキーマに準拠していることを確認してください。

JMS-00423 無効な値です。

原因 : 無効な値が指定されました。

処置 : XML 文書内に指定された値が、AQ スキーマに指定されている値と一致していることを確認してください。

JMS-00424 無効なメッセージ・ヘッダーです。

原因 : NULL または無効なメッセージ・ヘッダーが指定されています。

処置 : 有効なメッセージ・ヘッダーを指定してください。

JMS-00425 プロパティ名を指定する必要があります。

原因 : プロパティ名が NULL です。

処置 : NULL 以外のプロパティ名を指定してください。

JMS-00426 プロパティが存在しません。

原因 : 無効なプロパティ名が指定されました。プロパティが存在しません。

処置 : プロパティが存在しません。

JMS-00427 サブスクライバ名を指定する必要があります。

原因 : サブスクライバ名が NULL です。

処置 : NULL 以外のサブスクリプション名を指定してください。

JMS-00428 有効なメッセージを指定する必要があります。

原因 : メッセージが NULL です。

処置 : NULL 以外のメッセージを指定してください。

JMS-00429 登録オプションを指定する必要があります。

原因 : 登録オプションが NULL です。

処置 : NULL 以外の登録オプションを指定してください。

JMS-00430 データベース・リンクを指定する必要があります。

原因 : データベース・リンクが NULL です。

処置 : NULL 以外のデータベース・リンクを指定してください。

JMS-00431 順序番号を指定する必要があります。

原因 : 順序番号が NULL です。

処置 : NULL 以外の順序番号を指定してください。

JMS-00432 ステータスを指定する必要があります。

原因 : ステータス・オプションが NULL です。

処置 : NULL 以外のステータス・オプションを指定してください。

JMS-00433 ユーザーが認証されていません。

原因 : ユーザーが認証されていません。

処置 : サーブレットに接続する前に、ユーザーが Web サーバーで認証されていることを確認してください。

JMS-00434 無効なデータ・ソースです。

原因 : データ・ソースが NULL または無効です。

処置 : データベースに接続するために有効なデータ・ソースを指定してください。

JMS-00435 スキーマの位置が無効です。

原因 : スキーマの位置が NULL または無効です。

処置 : スキーマに対して有効な URL を指定してください。

JMS-00436 AQ 例外

原因 : AQ Java レイヤーでエラーが発生しました。

処置 : 詳細は、AQxmlException およびリンクされた Exception 内のメッセージを参照してください。

JMS-00437 無効な宛先です。

原因 : 無効な宛先オブジェクトが指定されました。

処置 : 有効な宛先（キュー / トピック）オブジェクトを指定してください。

JMS-00438 AQ エージェント {0} は有効なデータベース・ユーザーにマップされていません。

原因 : 指定された AQ エージェントが、要求された操作を実行する権限を持つデータベース・ユーザーにマップしていません。

処置 : dbms_aqadm.enable_db_access を使用して、必要なキュー特権を持つデータベース・ユーザーにエージェントをマップしてください。

JMS-00439 無効なスキーマ・ドキュメントです。

原因 : 指定されたスキーマ・ドキュメントが無効です。

処置 : スキーマ・ドキュメントに対して有効な URL を指定してください。

JMS-00440 無効な操作 : エージェント {0} は 2 つ以上のデータベース・ユーザーにマップされています。

原因 : AQ エージェントが、同じセッション内で 2 つ以上のデータベース・ユーザーにマップしています。

処置 : AQ エージェントにマップするデータベース・ユーザーは 1 つのみにしてください。このエージェントにマップするデータベース・ユーザーについては、`aq$internet_users` ビューを確認してください。

E

Unified Modeling Language 図

このマニュアルでは、アドバンスト・キューイングで使用されているテクノロジの説明方法として Unified Modeling Language（UML）を使用して利用図を示しています。この付録では、利用図および UML 表記法の概略を示します。

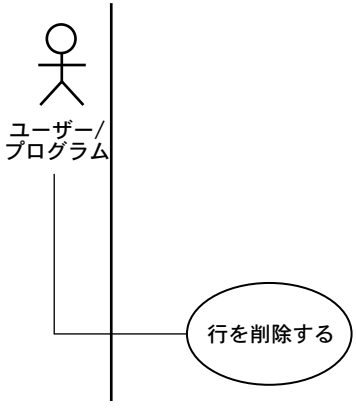
内容は次のとおりです。

- 利用図
- 状態図

利用図

利用図では、1 次利用は、それぞれアクター（人を表すマーク）によって開始します。アクターは、ユーザー、アプリケーションまたはサブプログラムのどれでもかまいません。アクターは、図 E-1 に示すとおり、アクションを囲む楕円（吹出し）で示される 1 次利用に接続されます。

図 E-1 1 次利用



1 次利用を完了するには、他の操作が必要になる場合があります。図 E-2 では、

- キュー名を指定する

が、次に示すオペレーションを完了するために必要なサブオペレーションまたは 2 次利用の 1 つです。

- メッセージをエンキューする

1 次利用から、必要な他の操作（ここでは省略されています）に向かって下向きの線が伸びています。

図 E-2 サブオペレーションが存在する 1 次利用

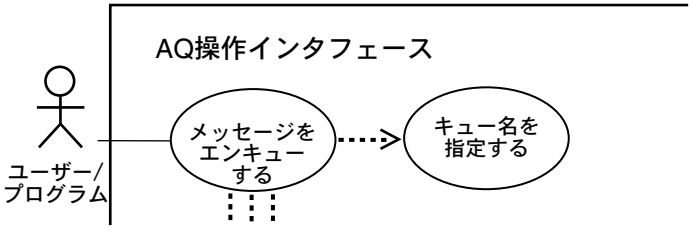


図 E-3 に示すとおり、影付きの 2 次利用は、固有の利用図に展開されます。これによって、次のことが簡単になります。

- オペレーション・ロジックの理解
- 複数ページにわたる複雑なオペレーションの継続

この例では、

- メッセージ・プロパティを指定する
- オプションを指定する
- ペイロードを追加する

というアクションは、すべてさらに詳しい利用図に展開されます。

図 E-3 2 次利用を表す影付きの利用図

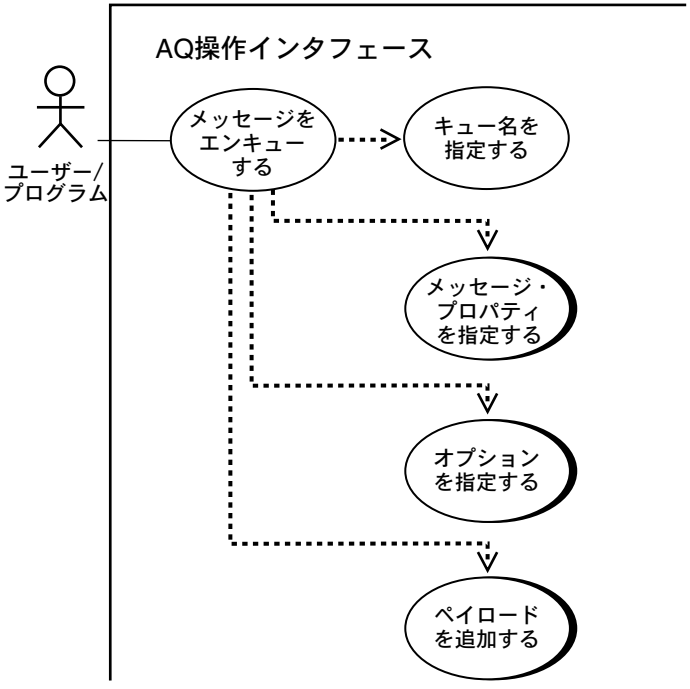


図 E-4（抜粋）には、展開された利用図が示されています。標準的な図は、通常、アクターから始まりますが、ここでは利用方法自体がサブオペレーションへの出発点になっています。この例では、

- ペイロードを追加する

の展開ビューが、次のオペレーションの構成要素を表しています。

- メッセージをエンキューする

図 E-4 展開された利用図

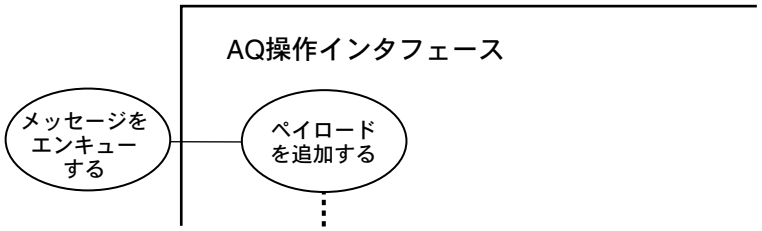
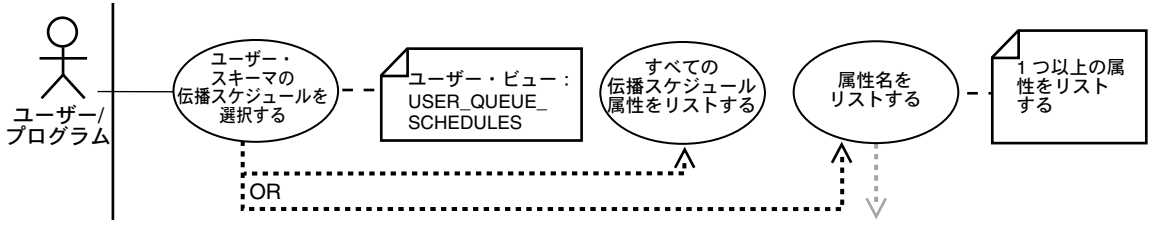


図 E-5 では、注釈ボックスの使用方法を示します。

- 注釈ボックスには、代替名を示すことができます。この場合、「ユーザー・スキーマの伝播スケジュールを選択する」というアクションは、USER_QUEUE_SCHEDULES というビューによって表されます。
- 注釈ボックスは、アクションの説明を示すことができます。この場合、「属性名をリストする」というアクションには、ユーザーに対する注釈が付けられています。注釈には、すべての伝播スケジュール属性をリストしない場合は、属性を 1 つ以上リストする必要があることが示されています。

図 E-5 注釈ボックス



利用図の破線の矢印は、依存性を示します。図 E-6 では、

- 一時 LOB を解放する

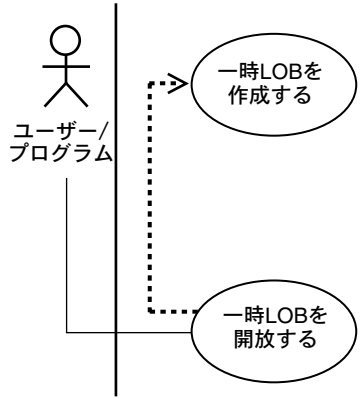
を実行するためには、まず、

- 一時 LOB を作成する

必要があることを示しています。

矢印の宛先は、最初に実行する必要があるオペレーションを示します。

図 E-6 依存性



利用方法とそのサブオペレーションは複雑な関係でリンクする可能性があります。図 E-7 の例では、まず、

- 通知を登録する

を実行した後で、

- 通知を受け取る

必要があります。

図 E-7 利用方法とサブオペレーションの関連

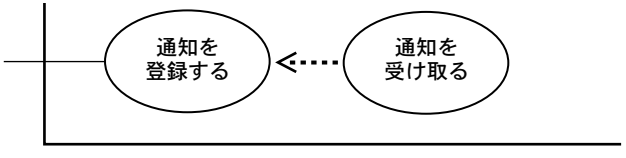


図 E-8 では、OR 条件の分岐パスが示されています。ビューを起動する際に、すべての属性のリストを選択するか、または 1 つ以上の属性を選択できます。灰色の矢印は、表示させる属性をユーザーが指定できることを示しています。

図 E-8 OR 条件の分岐パス

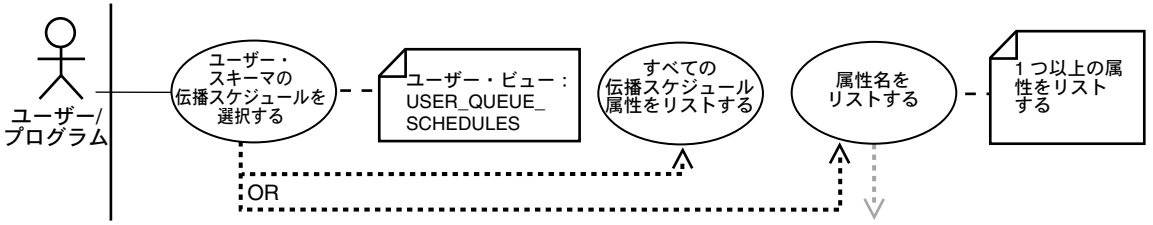


図 E-9 では、黒の破線の矢印は、宛先のオペレーションが必須であることを示しています。灰色の破線の矢印は、宛先のオペレーションがオプションであることを示しています。この例では、

- 追加を書き込む

を LOB に対して実行するには、まず

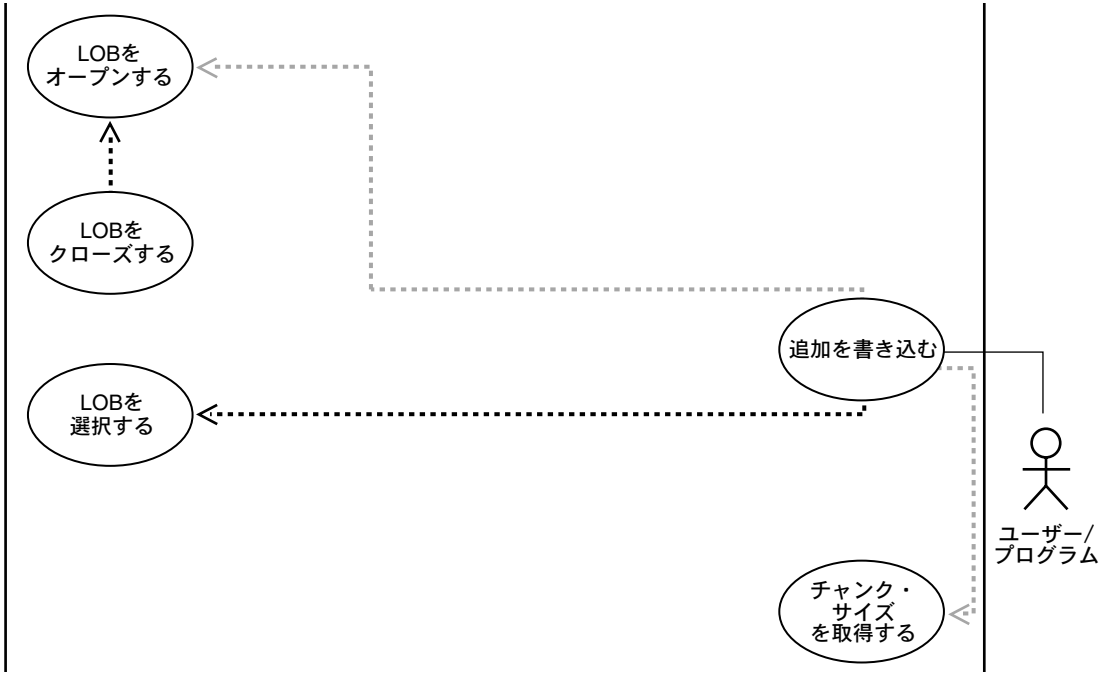
- LOB を選択する

必要があります。オプションとして、次の 2 つの操作から選択できます。

- LOB をオープンするか、またはチャンク・サイズを取得する

この図は、「LOB をオープンする」場合は、後で「LOB をクローズする」必要があることを示しています。

図 E-9 必須およびオプションの操作



状態図

状態図は、ビューの属性を示しています。ビューの属性には、可視または不可視という 2 つの状態が存在します。この例では、利用図の下に状態図（図の最下部の灰色の領域に示されているキュー、名前、アドレスおよびプロトコルのボックス）が追加され、ビューのすべての属性が示されています。

図 E-10 では、キューのサブスクライバを問い合せるためにビューが使用されることを示します。4 つの属性は、単独で、いくつか組み合せて、またはすべてを指定できます。

図 E-10 ビューの属性を示す利用図および状態図

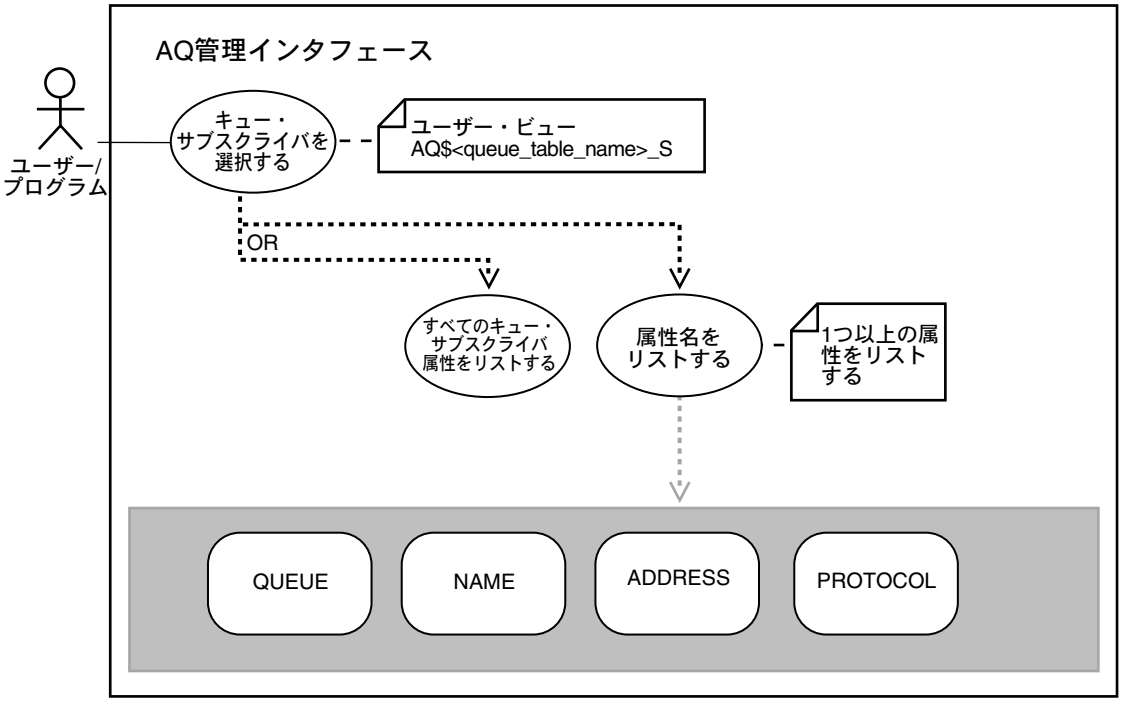
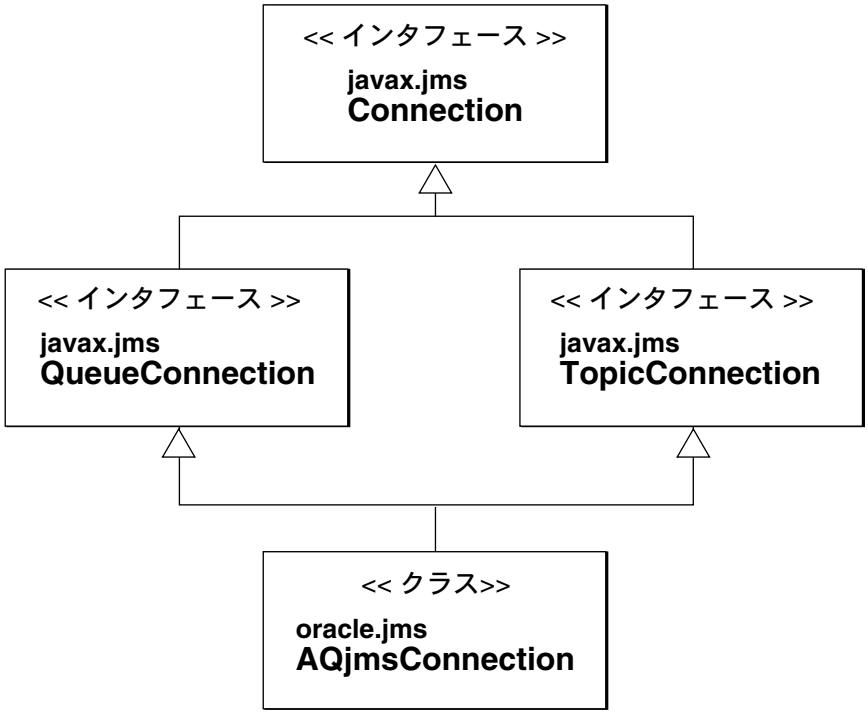


図 E-11 のクラス図は、次の項目を示しています。

- クラス、インタフェースおよび例外が相互関係を伴っているかどうか（**<< インタフェース >>** など、<<>> を使用して表します）
- クラスを含むパッケージの名前（**oracle.jms** など）
- クラスの名前（**AQjmsConnection** などなど）

図 E-11 クラス、インタフェースおよび例外を表すクラス図



索引

A

AdtMessage，12-28
AQ XML
　サーブレット，17-49，17-55
　スキーマ，17-33
AQ XML サーブレット
　通知登録，8-101
AQ_TM_PROCESSES，2-10
AQjmsQueueConnectionFactory，B-54
AQXmlPublish メソッド，17-6
AQXmlReceive メソッド，17-20
AQXmlSend メソッド，17-6
AQ エージェント
　削除，9-95
　作成，9-91
　登録，17-53
　変更，9-93
AQ オブジェクトの削除，A-64
AQ キュー
　登録，18-20
AQ サーブレット，17-2

B

BooksOnLine のサンプル・アプリケーション，8-1
　JMS の使用，12-2
BytesMessage，12-26

C

C
　「Oracle Call Interface（OCI）」を参照

D

DBA_ATTRIBUTE_TRANSFORMATIONS，10-43
DBA_QUEUE_TABLES，10-4，10-6，10-22
DBA_QUEUES，10-8
DBA_TRANSFORMATIONS，10-41
DBMS_AQADM.DROP_QUEUE，9-19
DBMS_AQADM パッケージ，4-2
DBMS_MGWADM.DB_CONNECT_INFO プロシージャ
　メッセージ・ゲートウェイの構成，18-13
　例，18-11
DBMS_MGWADM パッケージ，18-4
delay，2-9
dequeue mode，2-9

E

Enterprise Manager，1-7
expiration，2-9

F

FAQ，6-1
　JMS に関する質問，6-17
　Oracle Internet Directory，6-19
　一般的な質問，6-2
　インストールに関する質問，6-20
　インターネット・アクセスに関する質問，6-18
　パフォーマンスに関する質問，6-20
　変換に関する質問，6-20
　メッセージ・ゲートウェイに関する質問，6-7

H

HTTP，1-12，17-2，17-5，17-52，17-53，17-59
  AQ XML サーブレットへのアクセス，17-57
  AQ 操作，17-2
  伝播，17-61
  ヘッダー，17-4
  レスポンス，17-5
HTTPS
  伝播，17-61

I

iDAP
  「Internet Data Access Presentation」を参照
  インターネット上での転送，17-1
  スキーマ，17-36
  メッセージ，17-6
INIT.ORA パラメータ，2-10
Internet Data Access Presentation（iDAP），1-12，17-3

J

Java
  「JDBC」を参照
Java API，2-11
javax.jms.BytesMessage，B-24
javax.jms.Connection，B-25
javax.jms.ConnectionFactory，B-26
javax.jms.ConnectionMetaData，B-26
javax.jms.DeliveryMode，B-27
javax.jms.Destination，B-27
javax.jms.InvalidDestinationException，B-41
javax.jms.InvalidSelectorException，B-41
javax.jms.JMSException，B-42
javax.jms.MapMessage，B-27
javax.jms.MesageNotWriteableException，B-43
javax.jms.Message，B-28
javax.jms.MessageConsumer，B-30
javax.jms.MessageEOFException，B-42
javax.jms.MessageFormatException，B-43
javax.jms.MessageListener，B-31
javax.jms.MessageNotReadableException，B-43
javax.jms.MessageProducer，B-31
javax.jms.ObjectMessage，B-32
javax.jms.Queue，B-32
javax.jms.QueueBrowser，B-32
javax.jms.QueueConnection，B-33
javax.jms.QueueConnectionFactory，B-33
javax.jms.QueueReceiver，B-34
javax.jms.QueueSender，B-34
javax.jms.QueueSession，B-35
javax.jms.Session，B-35
javax.jms.StreamMessage，B-37
javax.jms.TextMessage，B-38
javax.jms.Topic，B-38
javax.jms.TopicConnection，B-38
javax.jms.TopicSession，B-40
javax.jms.TopicSubscriber，B-40
JDBC，3-6
  コネクション・パラメータ、データベースを介した登録，13-4
  コネクション・パラメータ、トピック・コネクション・ファクトリ，13-24
  コネクション・ファクトリ、LDAP を介した登録，13-8
JDBC URL
  LDAP を介した登録，13-11
  データベース登録，13-6
JMS
  サンプル・ペイロード，12-31
JMS インタフェース，B-2
JMS 型，17-10
JMS 型のキュー / トピック，17-10
JMS クラス，B-2，B-6
JMS の拡張機能，3-8
JMS 例外，B-2
JOB_QUEUE_PROCESSES パラメータ，2-10

L

LDAP
  キュー / トピック・コネクション・ファクトリ，13-26
  キュー / トピック、取得，13-28
  登録，13-11
  登録解除，13-14，13-16
LDAP サーバー
  AQ XML サーブレット，17-55
listener.ora ファイル
  変更，18-7
Listen 機能，8-86

M

MapMessage，12-27
message_grouping，2-9
MGW_ADMINISTRATOR_ROLE ロール，18-13
mgw.ora ファイル，18-9
　例，18-10

N

navigation，2-9

O

object_name，2-2
ObjectMessage，12-28
OO4O
　「Oracle Objects for OLE（OO4O）」を参照
Oracle Internet Directory，6-19
Oracle JMS クラス，B-6
Oracle Real Application Clusters，1-10，8-29，12-18
oracle.AQ.AQQueueTable，B-60
oracle.AQ.AQQueueTableProperty，B-60
oracle.jms.AdtMessage，B-44
oracle.jms.AQjmsAdtMessage，B-46
oracle.jms.AQjmsAgent，B-46
oracle.jms.AQjmsBytesMessage，B-47
oracle.jms.AQjmsConnection，B-47
oracle.jms.AQjmsConstants，B-48
oracle.jms.AQjmsConsumer，B-48
oracle.jms.AQjmsDestination，B-49
oracle.jms.AQjmsDestinationProperty，B-50
oracle.jms.AQjmsException，B-58
oracle.jms.AQjmsFactory，B-51
oracle.jms.AQjmsInvalidDestinationException，B-58
oracle.jms.AQjmsInvalidSelectorException，B-59
oracle.jms.AQjmsMapMessage，B-52
oracle.jms.AQjmsMessage，B-52
oracle.jms.AQjmsMessageEOFException，B-59
oracle.jms.AQjmsMessageFormatException，B-59
oracle.jms.AQjmsMessageNotReadableException，B-59
oracle.jms.AQjmsMesssageNotWriteableException，B-59
oracle.jms.AQjmsObjectMessage，B-53
oracle.jms.AQjmsOracleDebug，B-53
oracle.jms.AQjmsProducer，B-53
oracle.jms.AQjmsQueueBrowser，B-54
oracle.jms.AQjmsQueueReceiver，B-44
oracle.jms.AQjmsQueueSender，B-44
oracle.jms.AQjmsSession，B-54
oracle.jms.AQjmsStreamMessage，B-57
oracle.jms.AQjmsTextMessage，B-57
oracle.jms.AQjmsTopicBrowser，B-62
oracle.jms.AQjmsTopicConnectionFactory，B-57
oracle.jms.AQjmsTopicPublisher，B-45
oracle.jms.AQjmsTopicReceiver，B-46
oracle.jms.AQjmsTopicSubscriber，B-45
oracle.jms.TopicBrowser，B-61
oracle.jms.TopicReceiver，B-45
Oracle 以外のキュー
　登録，18-19
　登録解除，18-20
Oracle オブジェクト型（ユーザー定義型）のキュー，17-9
Oracle の拡張機能，3-8

P

PL/SQL，3-2

Q

QMN
　「キュー・モニター（QMN）」を参照
queue_type，2-9

R

RAW 型のキュー表およびキューの作成，A-4
RAW キュー，17-9
Real Application Clusters
　「Oracle Real Application Clusters」を参照
retention，2-9

S

sequence_deviation，2-9
SOAP，17-33
　エンベロープ，17-3
　ヘッダー，17-3
　本体，17-4
　メソッドの起動，17-4
　メッセージ構造，17-3

SOAP スキーマ，17-33
SQL アクセス，1-8
state パラメータ，2-9
StreamMessage，12-26

T

TextMessage，12-27
tnsnames.ora ファイル
　変更，18-8
　例，18-9
type_name，2-2

U

USER_ATTRIBUTE_TRANSFORMATIONS，10-40
USER_TRANSFORMATIONS，10-39

V

visibility，2-9
Visual Basic
　「Oracle Objects for OLE（OO4O）」を参照

W

wait，2-9

X

XML，17-1
　コンポーネント，17-10
　サーブレット，17-49，17-55
　サーブレット、HTTP，17-57
　スキーマ，17-33

あ

アクセス制御
　「システム・レベルのアクセス制御」を参照
宛先
　開始，13-54
　削除，13-60
　停止，13-56
　プロパティ、指定，13-36
　変更，13-58
宛先レベルのアクセス制御，12-17
アドバンスト・キューイング
　インターネット上での操作，17-2
アンロード
　メッセージ・ゲートウェイ，18-12

い

インストール
　メッセージ・ゲートウェイ
　　Oracle データベースの前提条件，18-6
　　確認，18-12
インストール後のタスク，18-7
インターネット
　AQ 操作，xxv，1-2，1-11
　アクセス，8-34
　アドバンスト・キューイング操作，17-1，17-2
インタフェース - javax.jms.BytesMessage，B-24
インタフェース - javax.jms.Connection，B-25
インタフェース - javax.jms.ConnectionFactory，B-26
インタフェース - javax.jms.ConnectionMetaData，B-26
インタフェース - javax.jms.DeliveryMode，B-27
インタフェース - javax.jms.Destination，B-27
インタフェース - javax.jms.MapMessage，B-27
インタフェース - javax.jms.Message，B-28
インタフェース - javax.jms.MessageConsumer，B-30
インタフェース - javax.jms.MessageListener，B-31
インタフェース - javax.jms.MessageProducer，B-31
インタフェース - javax.jms.ObjectMessage，B-32
インタフェース - javax.jms.Queue，B-32
インタフェース - javax.jms.QueueBrowser，B-32
インタフェース - javax.jms.QueueConnection，B-33
インタフェース - javax.jms.QueueConnectionFactory，
　B-33
インタフェース - javax.jms.QueueReceiver，B-34
インタフェース - javax.jms.QueueSender，B-34
インタフェース - javax.jms.QueueSession，B-35
インタフェース - javax.jms.Session，B-35
インタフェース - javax.jms.StreamMessage，B-37
インタフェース - javax.jms.TextMessage，B-38
インタフェース - javax.jms.Topic，B-38
インタフェース - javax.jms.TopicSession，B-40
インタフェース - javax.jms.TopicSubscriber，B-40
インタフェース - oracle.AQ.AQQueueTable，B-60
インタフェース - oracle.jms.AdtMessage，B-44
インタフェース -
　oracle.jms.AQjmsConnectionMetadata，B-47
インタフェース - oracle.jms.AQjmsConsumer，B-48

インタフェース - oracle.jms.AQjmsQueueReceiver, B-44
インタフェース - oracle.jms.AQjmsQueueSender, B-44
インタフェース - oracle.jms.AQjmsTopicPublisher, B-45
インタフェース - oracle.jms.AQjmsTopicReceiver, B-46
インタフェース - oracle.jms.AQjmsTopicSubscriber, B-45
インタフェース - oracle.jms.TopicReceiver, B-45
インタフェース、クラスおよび例外, B-2

え

永続キュー, 1-21
永続サブスクライバ, 12-45
エージェント, 1-21
「AQ エージェント」を参照
「ゲートウェイ・エージェント」を参照
識別, 2-3, 2-4
エージェント・ユーザー
作成, 18-11
例, 18-11
エクスポート
キュー表データ, 4-5
増分, 4-6
エラー・メッセージ, D-1
エンキュー, 11-4, 11-13
オプションの指定, 11-7
機能, 8-34
クライアントによるリクエスト, 17-6
サーバー・レスポンス, 17-27
メッセージ・プロパティの指定, 11-10
エンキュー、メッセージの優先順位および順序付け, 1-15

お

オブジェクト型, 4-3, 4-16
オブジェクト型のキュー表およびキューの作成, A-4

か

開始
宛先, 13-54
型
オブジェクト, 4-3, 4-16
監視
登録された Oracle 以外のキュー, 18-20
ログ・ファイル, 18-26
管理
「DBMS_MGWADM パッケージ」を参照
ゲートウェイ・エージェント, 18-13
メッセージ・ゲートウェイ, 18-4
管理インタフェース, 4-4, 9-1
基本操作, 13-2
ビュー, 10-1, 10-2
利用方法, 13-2
管理ユーザー
作成, 18-11
例, 18-11

き

期限切れ
時間指定, 8-46
起動, 18-14
機能、新規, xxxiii
キュー, 1-21
AQ
登録, 18-20
Oracle 以外
登録, 18-19
登録解除, 18-20
Point-to-Point, 12-37
Point-to-Point、作成, 13-38
開始, 9-43
削除, 9-32
作成, 9-21
作成、例, A-4
サブスクライバ, 7-6
サブスクライバ、選択, 10-29
サブスクライバのルール, 10-31
すべて選択, 10-7
選択、ユーザーがキュー権限を持つ, 10-15
選択、ユーザーがなんらかの権限を持つ, 10-13
選択、ユーザー・スキーマ, 10-23
停止, 9-46
非永続, 1-10, 6-3, 9-27
変更, 9-29
ユーザー・スキーマからの選択, 10-23
例外, 18-24
キューイング
基本, 7-3

基本、プロデューサ、コンシューマ，7-3
キュー・エンティティ
モデリング，7-2
キュー権限
取消し，9-56
取消し、Point-to-Point，13-52
付与，9-54
付与、Point-to-Point，13-50
キュー・タイプ
検証，9-79
キュー / トピック
LDAP，13-28
LDAP のコネクション・ファクトリ，13-26
コネクション・ファクトリ、LDAP を介した LDAP の登録解除，13-16
コネクション・ファクトリ、データベースを介した LDAP の登録解除，13-14
キューのサブスクライバ
選択、ルール，10-31
キューの伝播
スケジューリング，9-72
スケジュールの解除，9-76
キュー表，1-21
削除，9-18
作成，9-4，9-13，13-30，13-32
作成、例，9-8，9-23，9-24，A-4
作成、例、XMLType 属性，9-8
取得，13-34
すべて選択，10-3，10-17
変更，9-15
メッセージの選択，10-17
ユーザー・スキーマからすべて選択，10-21
ユーザーの表の選択，10-5
優先メッセージの作成，9-24
キュー表データ
エクスポート，4-5
キュー・モニター，1-23
キュー・モニター（QMN），2-10
キュー・レベルのアクセス制御，1-10，8-4

く

クラス，B-2
クラス - AQjmsQueueConnectionFactory，B-54
クラス、JMS，B-6
クラス - oracle.AQ.AQQueueTableProperty，B-60
クラス - oracle.jms.AdtMessage，B-46
クラス - oracle.jms.AQjmsBytesMessage，B-47
クラス - oracle.jms.AQjmsConnection，B-47
クラス - oracle.jms.AQjmsConstants，B-48
クラス - oracle.jms.AQjmsStreamMessage，B-57
クラス - oracle.jms.AQjmsDestination，B-49
クラス - oracle.jms.AQjmsDestinationProperty，B-50
クラス - oracle.jms.AQjmsException，B-58
クラス - oracle.jms.AQjmsFactory，B-51
クラス - oracle.jms.AQjmsMapMessage，B-52
クラス - oracle.jms.AQjmsObjectMessage，B-53
クラス - oracle.jms.AQjmsOracleDebug，B-53
クラス - oracle.jms.AQjmsProducer，B-53
クラス - oracle.jms.AQjmsQueueBrowser，B-54
クラス - oracle.jms.AQjmsSession，B-54
クラス - oracle.jms.AQjmsTextMessage，B-57
クラス - oracle.jms.AQjmsTopicConnectionFactory，B-57
グループ化
メッセージ，12-62
グローバル・イベント，6-19
グローバル・エージェント，6-19
グローバル・キュー，6-19

け

ゲートウェイ・エージェント，18-5
管理，18-13
ゲートウェイ・リンク
「メッセージ・ゲートウェイ・リンク」または「メッセージ・リンク」を参照
権限，4-4
「システム権限」、「トピック権限」などの特定の権限を参照
取消し，A-65

こ

構成
「DBMS_MGWADM.DB_CONNECT_INFO プロシージャ」を参照
接続情報，18-11
伝播ジョブ，18-20
メッセージ・ゲートウェイ・リンク，18-15
構造化ペイロード，1-9，8-11
構造化ペイロードまたはメッセージ型，12-21
コネクション・ファクトリ
LDAP を介した LDAP の登録解除，13-16

キュー / トピック、LDAP，13-26
データベースを介した LDAP の登録解除，13-14
トピック、JDBC URL，13-22
コミット・レスポンス，17-31
コンシューマ，7-5
コンポジット，7-13

さ

サーブレット
AQ XML，17-49，17-55
再開
伝播ジョブ，18-22
再試行
遅延間隔，8-74
最適化
到着待機，8-72
削除
宛先，13-60
キュー表，9-18
メッセージ・ゲートウェイ・リンク，18-18
作成
Point-to-Point キュー，13-38
エージェント・ユーザー，18-11
管理ユーザー，18-11
キュー，9-21
キュー表，13-30，13-32
キュー表およびキュー、例，A-4
サブスクライバ，18-21
メッセージ・ゲートウェイのスケジュール，18-21
メッセージ・ゲートウェイ・リンク，18-15
サブスクライバ，2-5
永続，12-45
削除，9-69
作成，18-21
選択，10-29
追加，9-59
変更，9-65，18-22
ルールベース，1-16，9-62
サブスクライバの追加，9-59
サブスクリプション，8-34
匿名，2-5
ルールベース，8-82
サブスクリプションおよび受信者リスト，1-14

し

時間指定，1-16
期限切れ，8-46，12-60
遅延，8-44，12-58
システム権限
取消し，9-52，13-44
付与，9-49，13-42
システム・レベルのアクセス制御，8-2，12-16
指定
宛先プロパティ，13-36
受信者，1-22，7-6
複数，8-60
リスト，2-4，8-34，12-49
ローカルおよびリモート，1-17，8-61
受信におけるメッセージのナビゲーション，12-69
取得
キュー表，13-34
使用可能化
伝播ジョブ，18-22
使用不可能化
伝播ジョブ，18-22
伝播スケジュール，9-88
新機能，xxxiii

す

スキーマ
AQ XML，17-33
iDAP，17-36
SOAP，17-33
スケジュール
作成，18-21
伝播，1-19，12-88，13-62
変更，18-22
スケジュールの解除
伝播，13-70

せ

静的サービス情報
例，18-8
セキュリティ，4-2，4-3
接続情報
構成，18-11
例，18-13

設定
　メッセージ・ゲートウェイ，18-6
選択ルール，18-23
前提条件
　Oracle 以外のメッセージ・システム，18-6
　Oracle データベース，18-6

そ

相関識別子，1-13
操作インタフェース
　基本操作，11-1
　利用方法，11-2
送信元の識別，1-16

た

待機
　メッセージの到着，8-72

ち

遅延
　時間指定，12-58
遅延間隔
　再試行，8-74
　時間指定，8-44
遅延を伴う再試行，1-18
チューニング
　「データベース・チューニング」を参照

つ

追跡およびイベント・ジャーナル，1-9
通知，17-32
　非同期，8-93
通知登録とリスナー，6-3

て

停止，18-14
　宛先，13-56
データベース
　設計およびモデリング，7-1
　チューニング，5-2
データベース・アクセス
　可能化，9-97
データベース・オブジェクト
　ロード，18-7
データベース・セッション，17-54
デキュー，11-44
　HTTP の使用，8-102
　機能，8-55
　クライアントによるリクエスト，17-20
　同一メッセージ、マルチ・コンシューマ，7-6
　方法，8-56
　マルチ・コンシューマによる同一メッセージのデキュー，7-6
　メッセージ・ナビゲーション，1-17，8-63
　モード，1-17，8-67
デキュー・リクエスト
　サーバー・レスポンス，17-29
伝播，1-15，2-10，8-102，12-83，17-61
　HTTP の使用，8-113
　LOB 属性，8-106
　LOB 属性を伴うメッセージ，8-106
　機能，8-102
　障害，4-12
　処理，18-5
　スケジューリング，13-62
　スケジュール，8-103，1-19，8-103，8-109，12-88
　スケジュールの解除，13-70
　スケジュールの使用可能化，13-64
　スケジュールの使用不可能化，13-68
　スケジュールの変更，13-66
　メッセージ，4-5，7-13
　問題点，4-10
　例外処理，12-92，12-94
伝播ジョブ
　構成，18-20
　再開，18-22
　使用可能化，18-22
　使用不可能化，18-22
　リセット，18-22
伝播スケジュール，12-90
　使用可能化，9-85
　使用不可能化，9-88
　すべて選択，10-9
　選択，10-9
　変更，9-82
　ユーザー・スキーマからの選択，10-25
伝播中の例外処理，12-92，12-94，12-95
伝播、例外処理，12-95

と

統計ビュー，8-33
統計ビューのサポート，12-20
登録
  AQ エージェント，17-53
  AQ キュー，18-20
  JDBC URL LDAP，13-11
  JDBC コネクション・パラメータ LDAP，13-8
  Oracle 以外のキュー，18-19
  キュー，2-5
  クライアントによるリクエスト，17-6
  データベース、JDBC URL，13-6
  データベース、JDBC コネクション・パラメータ，13-4
登録解除
  LDAP のキュー / トピック・コネクション・ファクトリ，13-14，13-16
  Oracle 以外のキュー，18-20
    例，18-20
登録された Oracle 以外のキュー
  監視，18-20
  変更，18-19
  例，18-20
登録リクエスト
  サーバー・レスポンス，17-31
トピック
  コネクション・ファクトリ、JDBC URL，13-22
  コネクション・ファクトリ、JDBC コネクション・パラメータ，13-24
  パブリッシュ・サブスクライブ、作成，13-40
トピック権限
  取消し、パブリッシュ・サブスクライブ，13-48
  付与、パブリッシュ・サブスクライブ，13-46
トピック・パブリッシャ，12-47
トランザクションのコミット，17-26
トランザクションのロールバック，17-26
トランザクション保護，1-18
取消し
  システム権限，13-44

は

パフォーマンス，5-2
パブリッシュ・サブスクライブ，7-12，8-27
  トピック，12-43
パブリッシュ・サブスクライブ・トピックの作成，13-40

ひ

非永続キュー，1-10，1-21，6-3
  作成，9-27
非同期通知，1-16，8-93
ビュー，10-1，10-2
  属性，10-1
  統計，8-33
標準キュー
  「ユーザー・キュー」を参照

ふ

複数の受信者，1-17
付与
  システム権限，9-49，13-42
プレビュー後のメッセージのデキュー，A-32
プログラム環境，2-7，3-2
プロデューサ，7-4
プロトコル・アドレス
  例，18-7

へ

ペイロード，17-8
  構造化，8-11
変換，18-23
  「メッセージ・フォーマットの変換」を参照
  例，18-24
変更
  listener.ora ファイル，18-7
  tnsnames.ora ファイル，18-8
  宛先，13-58
  サブスクライバ，18-22
  スケジュール，18-22
  登録された Oracle 以外のキュー，18-19
  メッセージ・ゲートウェイ・リンク，18-17

ほ

保存およびメッセージ履歴，1-9，8-26，12-18

め

メッセージ
  エラー、AQ XML サーブレット，D-1
  エラー、JMS，D-1
  グループ化，8-49
  受信者，7-6
  順序付け，8-36，12-55
  デキューにおけるナビゲーション，8-63
  伝播，7-13
  ファンアウト，7-13
  プロデューサおよびコンシューマ，1-21
  優先順位および順序付け，8-36，12-55
  履歴，8-26
メッセージ・エンキュー，11-4
メッセージ・ゲートウェイ，18-4，18-21
  FAQ，6-7
  アーキテクチャ，18-3
  インストール
    Oracle 以外のメッセージ・システムの前提条件，18-6
    Oracle データベースの前提条件，18-6
    確認，18-12
  管理，18-4
  機能，18-2
  サブスクライバ
    作成，18-21
メッセージ・ゲートウェイ・エージェント
  「ゲートウェイ・エージェント」を参照
メッセージ・ゲートウェイ・リンク
  構成，18-15
  削除，18-18
  作成，18-15
  変更，18-17
  例，18-18
メッセージ待機の最適化，1-17
メッセージのエンキューおよびデキュー
  Pro*C/C++ を使用した RAW 型，A-22，A-24
  Pro*C/C++ を使用した相関識別子およびメッセージ ID，A-37
  RAW 型，A-14，A-16，A-18
  オブジェクト型，A-11
  遅延および期限切れ，A-36
  マルチ・コンシューマ・キュー，A-43，A-46
  優先順位，A-14，A-16，A-18
  例，A-11
メッセージのグループ化，1-15，12-62
メッセージの受信，12-66
メッセージの定義，1-20
メッセージの非同期受信，12-76
メッセージのファネルイン
  「コンポジット」を参照
メッセージのファンアウト，7-13
メッセージの優先順位および順序付け，12-55
メッセージ・フォーマットの変換，1-7
メッセージ・プロデューサ機能，12-55，12-92
メッセージ・ペイロード，17-8
メッセージ履歴および保存，12-18

も

モデリング
  キュー・エンティティ，7-2
モデリングおよび設計，7-1

ゆ

ユーザー
  エージェント，18-11
  管理，18-11
ユーザー・キュー，1-21
ユーザー認可，17-53
ユーザー認証，17-52
ユーザー・ロール，4-2
優先順位指定メッセージのキュー表およびキューの作成，A-5

り

リセット
  伝播ジョブ，18-22
リンク
  「メッセージ・ゲートウェイ・リンク」を参照

る

ルール，1-22
  サブスクライバの選択，10-31
ルールベースのサブスクライバ，1-16
  例，9-62
ルールベースのサブスクリプション，8-82

れ

例
　AQ 操作，A-1
例外，B-2
例外 - javax.jms.InvalidDestinationException，B-41
例外 - javax.jms.InvalidSelectorException，B-41
例外 - javax.jms.JMSException，B-42
例外 - javax.jms.MesageNotWriteableException，B-43
例外 - javax.jms.MessageEOFException，B-42
例外 - javax.jms.MessageFormatException，B-43
例外 - javax.jms.MessageNotReadableException，B-43
例外 - oracle.jms.AQjmsInvalidDestinationException，
　B-58
例外 - oracle.jms.AQjmsInvalidSelectorException，B-59
例外 - oracle.jms.AQjmsMessageEOFException，B-59
例外 - oracle.jms.AQjmsMessageFormatException，
　B-59
例外 -
　oracle.jms.AQjmsMessageNotReadableException，
　B-59
例外 -
　oracle.jms.AQjmsMesssageNotWriteableException，
　B-59
例外キュー，1-21，18-24
例外処理，1-18，8-77，12-80
例外リスナーの ping 周期，16-133，16-135
列挙定数
　管理インタフェース，2-9
　操作インタフェース，2-9

ろ

ロード
　データベース・オブジェクト，18-7
　メッセージ・ゲートウェイ
　　Oracle 以外のメッセージ・システムの前提条件，
　　18-6
ロール
　取消し，A-65
　ユーザー，4-2
ロールおよび権限の取消し，A-65
ロールバック・レスポンス，17-32
ログ・ファイル
　監視，18-26