Oracle9*i* Lite

AQ Lite 開発者ガイド

リリース 5.0

2001 年 7 月

部品番号：J03838-01

ORACLE®

Oracle9*i* Lite AQ Lite 開発者ガイド , リリース 5.0

部品番号：J03838-01

原本名：Oracle9*i* Lite Developer's Guide for AQ Lite, Release 5.0

原本部品番号：A90251-01

目次

2 AQ Lite のディプロイメント

A AQ Lite Java API

B AQ Lite サンプル・アプリケーション

はじめに

ここでは、『AQ Lite 開発者ガイド』について説明します。

このマニュアルについて

Oracle AQ Lite は強力なメッセージ・キューイング・サービスで、モバイル端末と Oracle サーバーの間で信頼性の高いメッセージ通信を行います。AQ Lite は Oracle8*i* のアドバンスト・キューイング（AQ）機能に基づいています。AQ Lite は、Oracle8*i* の AQ 機能をモバイル端末に対して透過的に拡張します。

『AQ Lite 開発者ガイド』は、モバイルおよびサーバー・ベースのアプリケーションでメッセージを使用した通信を可能にする、キューイング・システムの構築方法を開発者に示しています。このガイドでは、AQ Lite の概念および機能の概要を説明し、AQ Lite を使用したキューイング・システムの開発とディプロイメントの方法を提供します。

説明する内容は、次のとおりです。

第 1 章「概要」	モバイル・エンタープライズ・アプリケーションにおけるメッセージ・キューイングの要件を考察することで、AQ Lite の概要を示します。
第 2 章「AQ Lite のディプロイメント」	AQ Lite を使用した、キューイング・システムのディプロイメント方法を説明します。
第 3 章「AQ Lite キュー・オブジェクトの管理」	AQ および AQ Lite Java API を使用した、AQ Lite キュー・オブジェクトの管理方法について説明します。
第 4 章「AQ Lite の操作インタフェース」	AQ Lite でメッセージをエンキューおよびデキューするために使用する操作インタフェースを説明します。
第 5 章「AQ Lite の伝播」	AQ Lite のメッセージ伝播の使用方法を説明します。
付録 A「AQ Lite Java API」	AQ Lite の Java application programmer's interface（API）の概要と詳細を説明します。
付録 B「AQ Lite サンプル・アプリケーション」	Oracle Lite に含まれる AQ Lite サンプル・アプリケーションの開発と使用方法を説明します。
付録 C「AQ Lite の制約」	AQ Lite の機能上の制約をリストします。

1

概要

この章では、モバイル・エンタープライズ・アプリケーションにおけるメッセージ・キューイングの要件を考察することで、AQ Lite の概要を示します。説明する内容は、次のとおりです。

概要

Oracle AQ Lite は強力なメッセージ・キューイング・サービスで、モバイル端末と Oracle サーバーの間で信頼性の高いメッセージ通信を行います。AQ Lite は Oracle8*i* のアドバンスト・キューイング（AQ）機能に基づいています。AQ Lite は、Oracle8*i* の AQ 機能をモバイル端末に対して透過的に拡張します。

開発者は、AQ Lite でアプリケーションを開発する前に、Oracle8*i* AQ の機能に習熟する必要があります。詳細は、『Oracle8*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』を参照してください。

AQ Lite の要件

Oracle8*i* AQ で、AQ Lite を使用するための要件は次のとおりです。

- オブジェクト・オプション付きの Oracle8*i* Enterprise Edition（リリース 8.1.5 以上）
- JDBC Thin クライアント
- Oracle8*i* の AQ 権限を持つユーザー名とパスワード
- Net8 Client

注意： Oracle Lite のデフォルトのインストールでは、AQ Lite はインストールされません。AQ Lite コンポーネントをインストールするためには、「Custom」インストール・オプションを使用する必要があります。インストール手順は、『Oracle9*i* Lite インストレーションおよび構成ガイド』を参照してください。

モバイル・アプリケーションにおけるメッセージ・キューイングの必要性

この項では、モバイル・エンタープライズ・アプリケーションの開発とディプロイメントに AQ Lite を使用する利点をいくつか示します。

サンプル・シナリオ

Oracle Lite に含まれる AQ Lite のサンプル・アプリケーションには、次のシナリオで説明するアプリケーションの実装例が提供されています。詳細は、付録 B「AQ Lite サンプル・アプリケーション」を参照してください。

LDB Sales は、主要ソフトウェア・メーカー数社の製品を扱う販売会社です。全国的な販売網を構成する 1200 名の社員は、それぞれの指定地域にあるソフトウェア開発会社と法人顧客を訪問して、1 日の仕事の大半を過ごします。この訪問中に、営業社員は製品の注文を取

り、また顧客に新しいソフトウェア製品の情報を提供します。各営業社員は、ラップトップ・コンピュータ上で実行される LDB Sales 外商販売用アプリケーションを使用します。LDB Sales のデータ・サーバーは、カリフォルニア州サンフランシスコの本社に置かれています。この会社の処理センターは、テキサス州ヒューストンにあります。販売アプリケーション・ソフトウェアを使用することで、営業担当者は次の作業ができます。

- 製品注文を入力し、販売データ・サーバーに送る
- 注文処理アプリケーションから受注承諾を受け取る
- 既存の注文情報を参照する

ロバートは、LDB Sales の営業社員です。彼の担当地域は、マサチューセッツ州ボストンのソフトウェア開発会社まで広がっています。通常、ロバートは顧客から毎日 30 件以上の製品注文を受けます。注文を受けるたびに、彼は自分のラップトップの販売アプリケーション (Oracle Lite データベース 1) に注文を入力します。注文を入力して保存すると、販売アプリケーションはその注文をその日の他の注文と一緒に「outbox」に格納します。1 日の仕事が終わって営業所に戻ると、ロバートは LDB Sales のデータ・サーバー（Oracle8*i* サーバー）にダイヤル接続します。サーバーに接続すると、ロバートは販売アプリケーションの「Send Orders」ボタンをクリックします。

販売アプリケーションは、ロバートのラップトップの「outbox」から、データ・サーバー上の対応する中央の注文キューに、その日の新規注文をすべて転送します。それと同時に、販売データ・サーバーは前日の注文に対する受注承諾を転送してきます。また、サーバーは更新された連絡先や製品情報の中から特にロバート宛てのものを、彼のラップトップの「inbox」に転送します。販売アプリケーションは、ロバートのラップトップのローカル・データベースを新しいデータで自動的に更新します。

注文処理センター（Oracle Lite データベース 2）では、次のような処理が行われます。注文処理アプリケーションは、ロバートの注文をデータ・サーバーから取り出します。適切なベンダーに新しい注文を送った後、注文処理アプリケーションはそれぞれの注文に対する受注承諾を生成します。次に、注文処理アプリケーションは受注承諾を販売データ・サーバーに転送します。ロバートは、次に接続したときにここから受注承諾を取り出すことができます。

この単純なシナリオでは、サーバーと複数のモバイル・クライアント端末との間でメッセージ交換を行うアプリケーションが説明されています。そのような環境で重要な問題の 1 つは、モバイル・クライアントが実際にサーバーに接続している時間が非常に短いことです。

モバイル・クライアントおよびサーバーのアプリケーションは、オフラインでメッセージを生成し、それぞれの受信者に宛先指定します。モバイル・クライアントおよびサーバーのアプリケーションは、オフラインで生成し宛先指定したメッセージを、短い接続期間中に速やかにかつ確実に交換できなければなりません。また、メッセージ送信時に、受信側アプリケーションがそのメッセージを処理可能かどうかにかかわらず、メッセージ交換が可能である必要があります。

アプリケーション・モデルとしての同期通信

従来のクライアント / サーバー・アーキテクチャでは、メッセージ交換は要求 / 応答パラダイムを基盤としていて、多くの場合「同期」、「オンライン」または「接続」と呼ばれます。このモデルでは、プログラムは別のプログラムに要求を送信し、応答が届くまで待機します(ブロックされます)。このメッセージの送信者と受信者の密接な連結は、処理を行う前に応答の受信が必要なプログラムに適しています。

このモデルの最大の欠点は、アプリケーションが機能するためにはすべてのコンポーネント・プログラムが使用可能かつ実行中になっている必要があることです。そのため、同期通信は、特にサンプル・シナリオで説明したようなモバイル・アプリケーション環境には適さないことになります。モバイル・クライアント端末がメッセージを送信および受信できるのは、サーバーに接続したときのみです。サーバー・アプリケーションが、この短時間の接続中に使用不可能になると、モバイル・クライアント・アプリケーションは処理を続けられません。

アプリケーション・モデルとしての非同期メッセージ

「非同期」、「非接続」または「遅延」モデルでは、プロデューサ役のプログラムが**キュー**にメッセージを入れてから処理を開始します。コンシューマ役のプログラムがキューからメッセージを取り出して、それを処理します。このモデルは、サンプル・シナリオで説明したタイプのモバイル・アプリケーションに最適です。クライアント・アプリケーションは、オフラインの間にメッセージを生成してキューに格納できます。モバイル端末がネットワークに接続したときに、格納されているすべてのメッセージがサーバー上の対応するキューに転送されます。同時に、モバイル・クライアント宛てのメッセージが、サーバー上のキューからクライアント端末上の指定されたキューに転送されます。

メッセージの永続性

ネットワーク、ハードウェアおよびソフトウェアには、障害が発生することがあります。障害が発生したときに非同期メッセージ機能が正しく動作するためには、メッセージは永続的に格納され、1 度は正確に処理される必要があります。つまり、メッセージ機能には永続性が必要です。

アプリケーションは、外部のクライアントまたは内部の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場合があります。このような条件では、アプリケーションの処理能力が追いつかないことがあります。同様に、データベース間の通信リンクは常に使用可能ではなく、他の用途のために確保されていることもあります。また、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こともあります。システムでメッセージを即座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メカニズムを持つ必要があります。

AQ Lite の利点

AQ Lite には、モバイル・アプリケーションに非同期の永続メッセージ・システムを実装する、開発者用のツールがあります。AQ および AQ Lite を使用することで、モバイル端末で実行中のアプリケーションは、サーバーまたは他のモバイル端末上のアプリケーションとメッセージを交換し、それらのメッセージを AQ Lite メッセージ・キューに格納できます。モバイル端末を Oracle8*i* サーバーに接続したときに、格納されているメッセージがサーバー上の対応する AQ メッセージ・キューに伝播されます。同時に、サーバー上の指定されたメッセージがモバイル端末上のメッセージ・キューに伝播されます。

次の表は、Oracle AQ Lite を使用したときの一般的な利点です。

利点	説明
統合されたデータベース・レベルの操作のサポート	AQ Lite のキューは、データベース・オブジェクトに実装されます。高可用性と信頼性という操作上の利点は、すべてキュー・データにも適用されます。さらに、データベース開発ツールや管理ツールもキューとともに使用できます。
統合されたトランザクション	メッセージ内に制御情報と内容（データ・ペイロード）が統合されることにより、アプリケーションの開発および管理が単純化されます。
Oracle8*i* AQ との統合	モバイル端末上のアプリケーションは、どの Oracle8*i* AQ アプリケーションともメッセージを交換できます。
信頼性およびリカバリ可能性	キュー・データには、データベースの標準的な特性（信頼性とリカバリ可能性）が適用されます。

AQ Lite のシステム・コンポーネント

この項では、AQ lite システムの基本コンポーネントについて説明します。

メッセージ

メッセージとは、キューに挿入され、また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 Lite 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、AQ Lite に対して透過的です。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・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

キューとは、メッセージのリポジトリです。キューは AQ Lite を使用して作成され、配布され、削除されます。キューには 3 つのタイプがあります。

- マスター・キュー
- クライアント・キュー
- スタンドアロン・キュー

マスター・キュー

マスター・キューは中心になるキューで、1 つ以上のクライアント・キューと関連付けられます。マスター・キューは Oracle8*i* サーバーに常駐します。

クライアント・キュー

クライアント・キューは Oracle Lite クライアント端末に常駐し、サーバー上のマスター・キューと同じキュー名を共有します。クライアント・キューは、対応するマスター・キューとのみ直接メッセージを交換できます。

ユーザーの視点では、クライアント・キューとマスター・キューが単一のキューのように機能します。マスター・キューに格納されたメッセージはクライアント・キューに伝播され、そこからユーザーによってデキューされます。同じように、クライアント・キューにエンキュされたメッセージはマスター・キューに伝播され、そこで別のユーザーによってアクセスされます。

スタンドアロン・キュー

スタンドアロン・キューは Oracle Lite クライアント端末に常駐し、ローカル・アプリケーションどうしのメッセージ交換に使用されます。スタンドアロン・キューには対応するマスター・キューがなく、サーバーにメッセージを伝播したり、サーバーからメッセージが伝播されることはありません。

キュー・テーブル

キューは、キュー・テーブルに格納されます。キュー・テーブルは、1 つ以上のキューを含むデータベース表です。

エージェント

エージェントとは、キュー内のメッセージにアクセスするエンド・ユーザーまたはアプリケーションです。エージェントには 2 つのタイプがあります。

- メッセージをキューに入れる（エンキュする）プロデューサ
- メッセージをキューから取り出す（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、AQ Lite の操作インタフェースを使用して、メッセージをキューに挿入し、またキューからメッセージを取り出します。

エージェントは、「名前」、「アドレス」および「プロトコル」によって識別されます。エージェントの名前には、アプリケーションの名前あるいはアプリケーションによって割り当てられたユーザーの名前を使用できます。

受信者リストおよびサブスクライバ・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されるように指定できます。これには 2 通りの方法があります。

- メッセージのエンキュ元が、メッセージを取り出せる受信者を指定できます。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。
- キュー管理者が、キューからメッセージをデキューできるサブスクライバのデフォルト・リストを指定できます。受信者を指定せずにエンキュされたメッセージは、すべてのサブスクライバに送信されます。メッセージ受信者リストは、キュー・サブスクライバ・リストをオーバーライドします。

AQ Lite の機能

この項では、開発者が AQ Lite を使用して実装できる機能の概要を示します。

ディプロイメント機能

ディプロイメントとは、Oracle Lite クライアント・キューを設定して初期化し、クライアント・キューと Oracle8*i* サーバー上のマスター・キューの間の伝播関係を確立する処理です。

最初に Oracle8*i* サーバー・データベースにキュー・オブジェクトが作成されます。ディプロイメントによって、同一のキューが Oracle Lite クライアント・データベース上に確立されます。このディプロイメント処理には、キュー間の伝播に必要な定義も含まれます。

クライアント管理は不要

AQ Lite では、Oracle Lite クライアント・サイトの管理は必要ありません。Oracle8*i* サーバー上にキュー関係が設定されると、各 AQ Lite クライアントはディプロイメントを実行して、キュー・オブジェクトをローカルの Oracle Lite データベースに確立します。それ以上のメンテナンスは必要ありません。

伝播機能

AQ Lite プロパゲータは、Oracle Lite データベースのクライアント・キューと Oracle8*i* データベースのマスター・キューの間で、メッセージを転送するユーティリティです。たとえば、ラップトップで実行される販売アプリケーションは、製品の注文メッセージをローカルの Oracle Lite データベースにある「Orders」という名前のキューに格納します。AQ Lite プロパゲータは注文メッセージをローカルの「Orders」キューからデキューし、それらを Oracle8*i* サーバー上の「Orders」マスター・キューに伝播させます。同じ伝播の間に、AQ Lite プロパゲータはサーバー上の Orders マスター・キューからクライアント端末宛ての受注応諾メッセージをデキューし、それをクライアント端末上のキューにエンキュします。

エンキュ機能

次の AQ Lite 機能は、メッセージをキューにエンキュすることで、メッセージ生成プロセスをサポートします。

enqueue および propEnqueue

AQ Lite は、`enqueue()` メソッドおよび `propEnqueue()` メソッドを使用して、ローカル・アプリケーションによってエンキュされるメッセージと AQ Lite プロパゲータによってエンキュされるメッセージを区別します。AQ Lite プロパゲータは、`propEnqueue()` を使用して、サーバーからのメッセージを Oracle Lite クライアント・キューにエンキュします。これによって、クライアント端末にエンキュされたメッセージがサーバーに伝播して戻されることはなくなります。Oracle Lite クライアント上で実行されるアプリケーションは、`enqueue()` を使用してサーバー宛てのメッセージをローカル・キューに格納します。これによって、別のローカル・アプリケーションがメッセージをデキューして処理することはなくなります。ローカル・アプリケーションは、`propEnqueue()` を使用して別のローカル・アプリケーションが使用するメッセージをエンキュします。

相関識別子

ユーザーはメッセージに識別子を割り当てて、メッセージ間の相関関係を確立できます。これによって、ユーザーは関連するメッセージのグループを取り出せます。

エンキュにおけるメッセージの優先順位と順序付け

ユーザーは、メッセージのデキュー順序を、優先順位またはソート順を使用して指定できます。優先順位による順序付けを使用すると、エンキュ時に各メッセージに優先順位が割り当てられます。デキュー時には、メッセージは優先順位の順序でキューから取り出されます。優先順位による順序付けが使用されないときは、メッセージはその位置を基にデキューされます。ユーザーは、ソート・プロパティを指定して、キューのメッセージを順序付けできます。

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、各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。別のコンシューマがデキューしているメッセージは、スキップされます。

デキュー機能

次の AQ Lite 機能は、メッセージをキューからデキューすることで、メッセージを取り出すプロセスをサポートします。

dequeue および propDequeue

AQ Lite は、dequeue() メソッドおよび propDequeue() メソッドを使用して、ローカル・アプリケーションによってデキューされるメッセージと AQ Lite プロパゲータによってデキューされるメッセージを区別します。AQ Lite プロパゲータは、propDequeue() を使用して、Oracle Lite クライアントから、サーバー上のマスター・キュー宛てのメッセージをデキューします。Oracle Lite クライアント上で実行されるアプリケーションは、dequeue() を使用してローカル・キューからメッセージをデキューします。

複数の受信者

キューにある 1 つのメッセージは、複数のコピーを作成しなくても、複数の受信者により取り出すことができます。

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、単一トランザクションに関連する集合体を形成できます。このキューは、メッセージのグループ化を使用可能にしたキュー・テーブルに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内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が含まれている場合には、そのメッセージは、ヘッダーのメッセージ、請求書の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。

デキューにおけるメッセージのナビゲーション

キューのメッセージを使用するには、いくつかの方法があります。キューの最初のメッセージを選択し、それからキュー順序に従ってナビゲートすれば、どのメッセージでも選択できます。相関識別子を指定することで、キューのメッセージをフィルタ処理できます。また、一意の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。

デキューのモード

デキュー要求は、キューのメッセージを「ブラウズ」または「削除」できます。メッセージがブラウズ・モードでデキューされた後でも、別のコンシューマによってデキュー可能です。メッセージが削除モードでデキューされた後は、キューから削除されるため、別のコンシューマはアクセスできません。

Java API

AQ Lite の Java API は、AQ Lite の管理機能および操作機能の両方をサポートします。AQ Lite の Java API は、Oracle8*i* AQ と完全な互換性があります。詳細は、付録 A「AQ Lite Java API」を参照してください。

AQ Lite の構成

AQ Lite によって、2 つの基本キューイング構成を実装できます。

- マスター / クライアント・キュー構成
- スタンドアロン・キュー構成

この構成は、AQ Lite で実装できる最も基本的なキューイング・システムです。これは、より複雑な AQ Lite キューイング・システムを作成する基本ブロックと考えることができます。この構成を再現し、多様な方法で複合し、その機能性を Oracle8*i* AQ に拡張することで、他に類がないメッセージ機能要件を満たす複合キューイング・システムを開発できます。

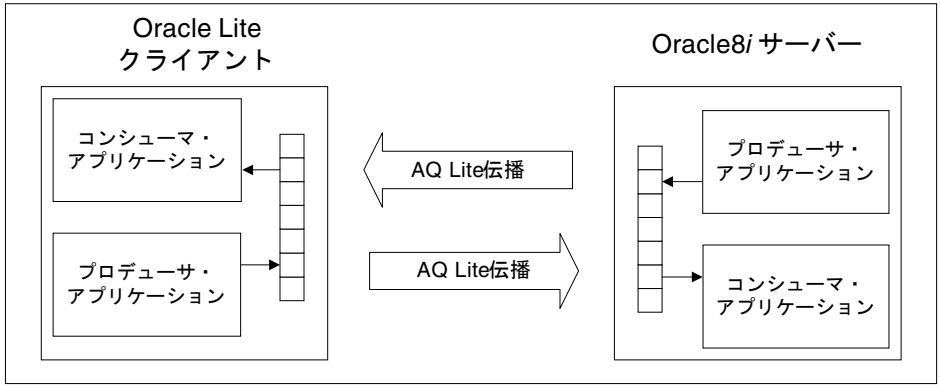
マスター / クライアント・キュー構成

マスター / クライアント・キュー構成では、Oracle8*i* サーバー上のマスター・キューと、それに対応する 1 つ以上の Oracle Lite 端末上に常駐するクライアント・キューの間で、メッセージが交換されます。メッセージは、クライアントとサーバーのキューの間を伝播によって交換されます。クライアント・キューには、対応するマスター・キューと同じキュー名が付けられます。クライアント・キューは、対応するマスター・キューとのみ直接メッセージを交換できます。

次のシナリオで、マスター / クライアント・キュー構成のバリエーションをいくつか示します。

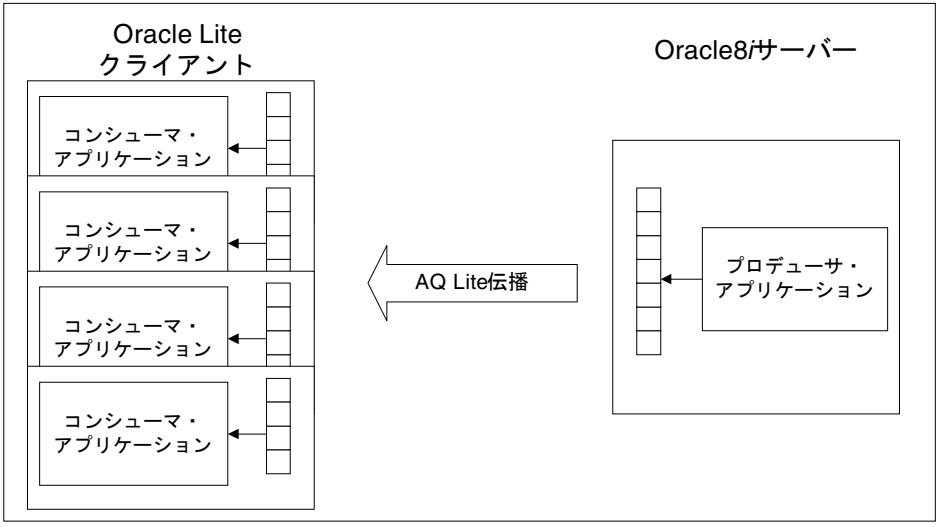
クライアント / サーバー・メッセージ機能

このシナリオでは、メッセージはサーバーとクライアント端末の間で交換されます。マスター・キューおよびクライアント・キューは、同一のキュー名が付いています。クライアント端末で生成されるメッセージは、クライアント・キューにエンキュされます。このメッセージは、次にサーバー上のマスター・キューに伝播されます。プロデューサ・アプリケーションおよびコンシューマ・アプリケーションは、クライアント端末とサーバーの両方に常駐できます。



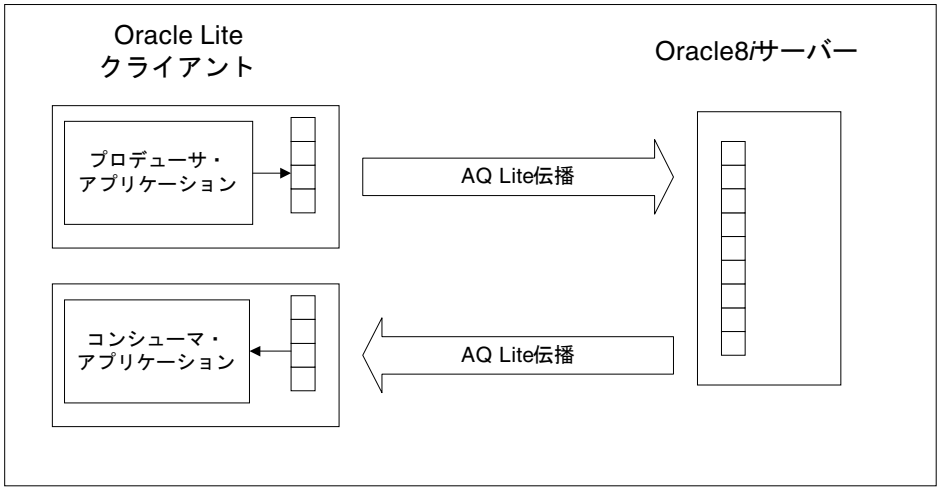
サーバー / マルチクライアント・メッセージ機能

このシナリオでは、Oracle8*i* 上のマスター・キューから複数の端末上のクライアント・キューにメッセージをブロードキャストします。キューイング・システムの各端末に共通のサブスクライバ名を使用することで、メッセージのグローバルなディプロイメントが円滑にできます。たとえば、各クライアント端末上に「globalsubscriber」という名前のコンシューマを作成したとします。このサブスクライバにパブリッシュされるメッセージは、システム内のどの AQ Lite クライアントからでもデキューできます。各コンシューマはブラウズ・モードでメッセージをデキューするため、各ユーザーが使用した後でもメッセージはパブリッシュされたキューに残ります。



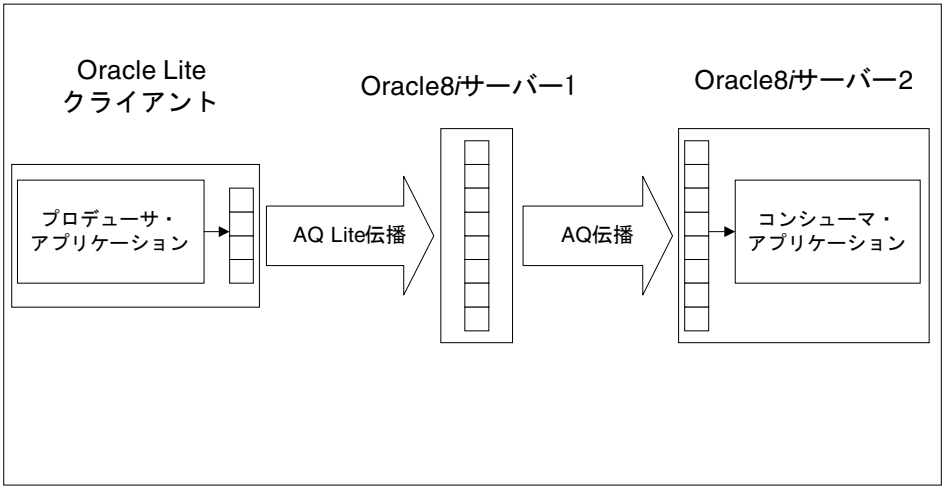
クライアント / クライアント・メッセージ機能

クライアント / クライアントのシナリオでは、メッセージはマスター・サーバーを通って 2 つ以上のクライアント端末の間で交換されます。マスター・キューおよびすべてのクライアント・キューは、同一のキュー名を持つ必要があります。クライアント端末で生成されるメッセージは、マスター・キューに伝播されます。このメッセージは、次にメッセージ受信者リストまたはキュー・サブスクライバ・リストで指定されたクライアント・キューに伝播されます。



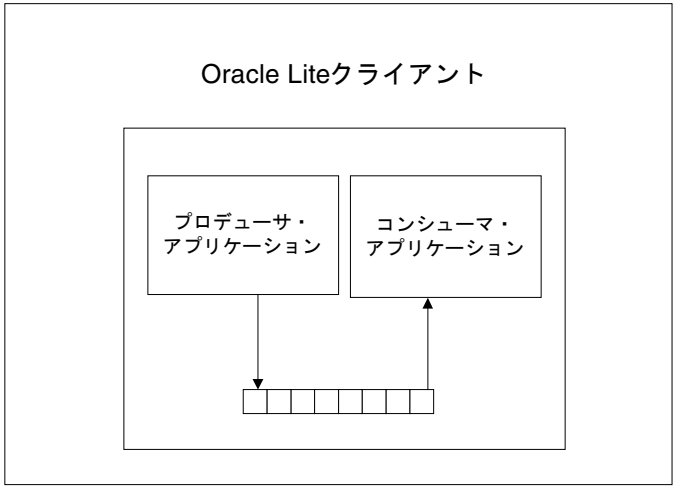
クライアント / ノンマスター・サイト・メッセージ機能

このシナリオでは、AQ Lite と AQ の両方の機能の使用方法を示します。メッセージは、クライアント端末とマスター・キューが常駐しているサーバー以外のサーバーとの間で交換されます。クライアント端末で生成されるメッセージは、AQ Lite 伝播を使用して、クライアント・キューからマスター・キューに伝播されます。メッセージがマスター・キューに送られると、AQ 機能がコールされてそのメッセージを 2 つめのサーバーに伝播します。



スタンドアロン・キュー構成

スタンドアロン・キュー構成では、同じクライアント端末に常駐するプロデューサ・アプリケーションおよびコンシューマ・アプリケーションが、単一の共通キューを通してメッセージを交換します。この構成ではただ 1 つのクライアント・キューが必要になります。マスター・キューはありません。伝播はこの構成では必要ありません。プロデューサは、キューにメッセージをエンキュします。コンシューマは、キュー・サブスクライバ・リストまたはメッセージ受信者リストに基づいてメッセージをデキューできます。


Oracle Liteクライアント
プロデューサ・
アプリケーション
コンシューマ・
アプリケーション

2

AQ Lite のディプロイメント

この章では、AQ lite を使用したマスター / クライアント・キュー構成のディプロイメントの方法を説明します。説明する内容は、次のとおりです。

- ディプロイメントの概要
- ディプロイメント API
- ディプロイメント・プロシージャ
- 既存のディプロイメントの変更
- エラー・メッセージとアドバイザ・メッセージ

ディプロイメントの概要

マスター / クライアント構成では、Oracle8*i* サーバー上のマスター・キューと、それに対応する 1 つ以上の Oracle Lite 端末上に常駐するクライアント・キューの間で、メッセージが伝播されます。クライアント・キューとマスター・キューの間でメッセージを伝播するためには、パブリッシュ / サブスクライブ・プロシージャを使用してディプロイメント時に両者の関係を確立する必要があります。

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・プロシージャによって関係が確立し、その関係によってマスター・キューとクライアント・キューの間でメッセージが伝播されるようになります。このプロシージャには、パブリッシュ・フェーズとサブスクライブ・フェーズがあります。

パブリッシュ・フェーズでは、AQ キューが Oracle8*i* サーバー上に作成され、各キューにメッセージ受信者が設定されます。メッセージ受信者は、キュー・サブスクライバ・リストから暗黙的に選択するか、メッセージ受信者リストで受信者を指定して選択することができます。

サブスクライブ・フェーズでは、サーバー上の AQ キューが Oracle Lite 端末上に対応するクライアント・キューを持ったマスター・キューに変換されます。サブスクライブ・フェーズでは、マスター・キューのどのメッセージを各 Oracle Lite 端末上の受信者に伝播するかを定義します。

上記のパブリッシュ / サブスクライブ・メカニズムに基づいて、次の AQ Lite Java API のクラスを使用してディプロイメントが行われます。

- `AQLiteDeploymentAdmin`
- `AQLiteDeploymentQueueGroup`
- `AQLiteDeploymentOliteSite`
- `AQLiteDeploymentClient`

ディプロイメント・タスク

ディプロイメントの前に、Oracle8*i* サーバー管理者が、AQ Lite クライアントにディプロイメントするマスター・キューを作成する必要があります。キュー作成の詳細は、第 3 章「AQ Lite キュー・オブジェクトの管理」を参照してください。

キューを作成した後、管理者はクラス AQLiteDeploymentAdmin をコールして、次のステップを実行します。

1. QueueGroup を作成する

もっと効率的にディプロイメントを行うために、管理者は QueueGroup にキューを割り当てます。QueueGroup は、キューの集まりです。QueueGroup を使用すると、各端末ごとにキューを割り当てなくても、共通のキュー・グループを複数のクライアント端末にディプロイメントできます。また、1 つのキューが、複数の QueueGroup のメンバーになることができます。AQLiteDeploymentQueueGroup には、QueueGroup オブジェクトの管理機能があります。

2. OliteSite オブジェクトを作成する

次に、サーバー管理者は OliteSite オブジェクトを作成します。各 OliteSite オブジェクトは、Oracle Lite 端末上の受信者を 1 人表します。AQLiteDeploymentOliteSite は、OliteSite オブジェクトを管理するために使用されます。

3. OliteSite サブスクリプションを定義する

QueueGroup オブジェクトおよび OliteSite オブジェクトが作成されると、サーバー管理者は各 OliteSite と QueueGroup の関係を定義します。1 つの QueueGroup を複数の OliteSite オブジェクトに割り当てることができます。QueueGroup オブジェクトを含む各 OliteSite オブジェクトを特定の Oracle Lite クライアントにディプロイメントできます。複数の OliteSite オブジェクトを単一の Oracle Lite クライアントにディプロイメントできます。

4. ディプロイメントのインスタンスを生成する

すべての OliteSite オブジェクトが作成され定義されると、サーバー管理者はディプロイメントのインスタンスを生成します。インスタンスの生成によって、OliteSite 定義に基づいたディプロイメント・メッセージが生成されます。ディプロイメント・メッセージは、QueueGroup と OliteSite オブジェクトの関係ごとに生成されます。各メッセージには、Oracle Lite クライアントにディプロイメントされるキューに関する情報が含まれます。

ディプロイメント・メッセージは、Oracle8*i* サーバー上のディプロイメント・キューである AQLITE$_DeplMessages にエンキュされます。ディプロイメント・キューは AQ Lite のシステム・キューで、ディプロイメントを円滑に行うために使用されます。ディプロイメント・キューは、サーバーおよび各クライアント端末に常駐します。

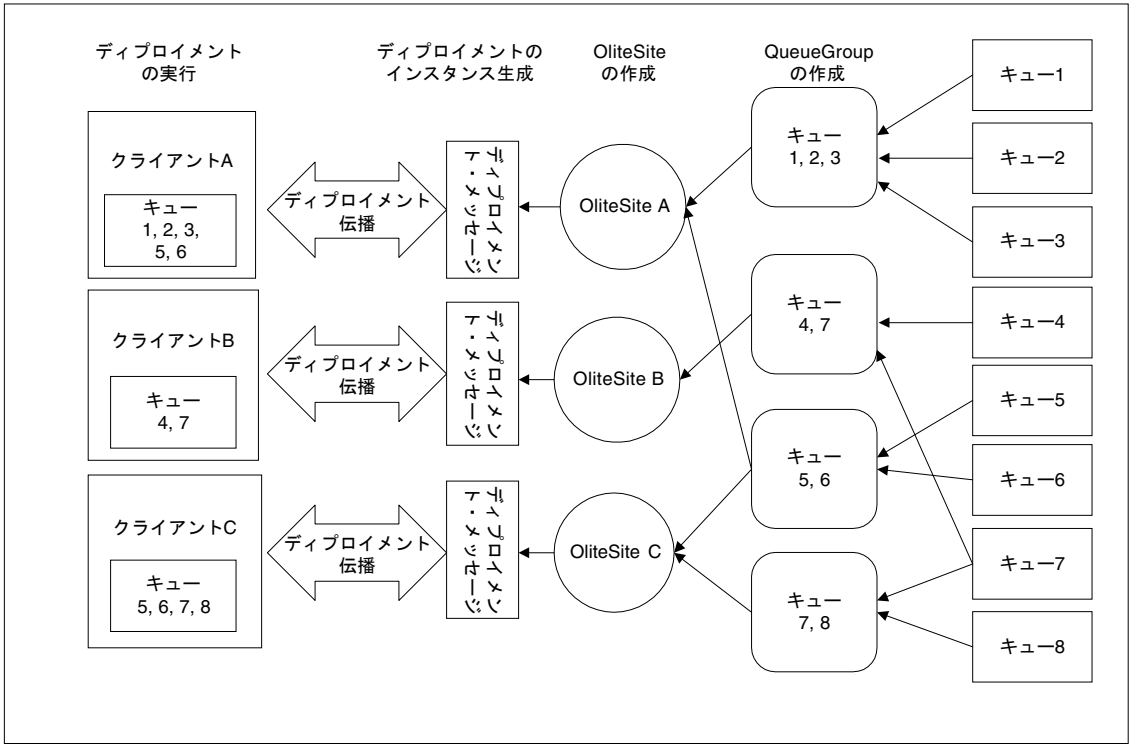
5. クライアントからディプロイメントを実行する

ディプロイメント・メッセージの準備ができると、Oracle Lite クライアントのユーザーはサーバーに接続して、クラス AQLiteDeploymentClient をコールすることでディプロイメントの要求を発行します。

AQLiteDeploymentClient は、次に示す機能を実行します。

1. Oracle Lite サイトに AQ カタログ・クラスを作成する。
2. 伝播によって Oracle8*i* サーバーとの通信を確立する。
3. エンキュされたディプロイメント・メッセージを、AQLiteDeploymentAdmin によって伝播しデキューする。
4. キュー・テーブルとキューを Oracle Lite サイトに作成する。
5. キュー内のあらゆるメッセージの伝播を初期化する。

注意： ディプロイメントは、各クライアント端末から 1 度実行する必要があります。



ディプロイメント API

この項では、AQ Lite ディプロイメントで使用される次のクラスの API について説明します。

- AQLiteDeploymentAdmin
- AQLiteDeploymentQueueGroup
- AQLiteDeploymentOliteSite
- AQLiteDeploymentClient

AQLiteDeploymentAdmin

このクラスは、ディプロイメント準備中の様々なキュー・オブジェクトを管理するために使用します。

コンストラクタ

用途

AQLiteDeploymentAdmin クラスを構成します。

構文

```
public AQLiteDeploymentAdmin(java.lang.Object obj)
```

パラメータ

パラメータ	説明
conn	Oracle データベース接続のオブジェクト

createOliteSite

用途

このメソッドは、OliteSite オブジェクトを作成します。

構文 1

```
public AQLiteDeploymentOliteSite createOliteSite(java.lang.String oliteSiteID,
                                                 java.lang.String applicationName,
                                                 java.lang.String owner,
                                                 java.lang.String queue,
                                                 boolean isLocal,
                                                 java.lang.String comments)
                                                    throws AQException
```

構文 2

```
public AQLiteDeploymentOliteSite createOliteSite(java.lang.String oliteSiteID,
                                                 java.lang.String applicationName,
                                                 boolean isLocal,
                                                 java.lang.String comments)
                                                    throws AQException
```

パラメータ

パラメータ	説明
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID。
applicationName	指定された OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ。
owner	ディプロイメント・メッセージ・キューの所有者。デフォルトを使用するときは、NULL に設定します。
queue	ディプロイメント・メッセージ・キューのキュー。デフォルトを使用するときは、NULL に設定します。
isLocal	OliteSite のキュー・タイプ。
comments	関連するコメント。

getOliteSite

用途

このメソッドは、OliteSite オブジェクトを取得します。

構文

```
public AQLiteDeploymentOliteSite getOliteSite(java.lang.String oliteSiteID,
                                              java.lang.String applicationName)
```

パラメータ

パラメータ	説明
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID
applicationName	指定された OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ

dropOliteSite

用途

このメソッドは、OliteSite オブジェクトを削除します。

構文

```
public void dropOliteSite(java.lang.String oliteSiteID,
                          java.lang.String applicationName)
```

パラメータ

パラメータ	説明
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID
applicationName	指定された OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ

createQueueGroup

用途

このメソッドは、キューの配列を伴う QueueGroup オブジェクトを作成します。

構文 1

```
public AQLiteDeploymentQueueGroup createQueueGroup
                (java.lang.String groupName,
                 java.lang.String comments,
                 java.lang.String[] owner,
                 java.lang.String[] queue)
                    throws AQException
```

構文 2

```
public AQLiteDeploymentQueueGroup createQueueGroup
                (java.lang.String groupName,
                 java.lang.String comments,
                    throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前
comments	関連するコメント
owner	キュー所有者の配列
queue	キューの配列

getQueueGroup

用途

このメソッドは、QueueGroup オブジェクトを取得します。

構文

```
public AQLiteDeploymentQueueGroup getQueueGroup(java.lang.String groupName)
```

パラメータ

パラメータ	説明
groupName	グループの名前

dropQueueGroup

用途

このメソッドは、QueueGroup オブジェクトを削除します。

構文

```
public void dropQueueGroup(java.lang.String groupName)
```

パラメータ

パラメータ	説明
groupName	グループの名前

AQLiteDeploymentQueueGroup

このクラスは、QueueGroup オブジェクトのディプロイメントを管理するために使用します。

コンストラクタ

用途

AQLiteDeploymentQueueGroup クラスを構成します。そのクラスに対応するデータベース・オブジェクトが存在しない場合、コンストラクタがインポート・パラメータに基づいてデータベースを作成します。

構文 1

```
public AQLiteDeploymentQueueGroup(java.lang.Object obj,
                                  java.lang.String queueGroup)
                                     throws AQException
```

構文 2

```
public AQLiteDeploymentQueueGroup(java.lang.Object obj,
                                  java.lang.String queueGroup,
                                  java.lang.String comments)
                                     throws AQException
```

構文 3

```
public AQLiteDeploymentQueueGroup(java.lang.Object obj,
                                  java.lang.String queueGroup,
                                  java.lang.String comments,
                                  java.lang.String[] owner,
                                  java.lang.String[] queue)
                                     throws AQException
```

パラメータ

パラメータ	説明
conn	Oracle データベース接続のオブジェクト
queueGroup	QueueGroup の名前
comments	関連するコメント
owner	キュー所有者の配列
queue	キューの配列

getName

用途

このメソッドは、QueueGroup の名前を取得します。

構文

```
public java.lang.String getName()
```

drop

用途

QueueGroup オブジェクトを削除します。

構文

```
public void drop()
   throws AQException
```

getQueues

用途

このメソッドは、すべてのキュー名の配列を取得し、ベクトル [2*i] にキュー所有者を、ベクトル [2*i+1] にキュー名を返します。

構文

```
public java.util.Vector getQueues()
   throws AQException
```

getOliteSite

用途

このメソッドは、QueueGroup オブジェクトと関連するすべての OliteSite オブジェクトの配列を取得します。

構文

```
public java.lang.String[] getOliteSite()
   throws AQException
```

defineDeployTo

用途

このメソッドは、QueueGroup オブジェクトがディプロイメントされる OliteSite オブジェクトを定義します。

構文

```
public void defineDeployTo(java.lang.String OliteSite,
                           java.lang.String user)
                              throws AQException
```

undefineDeployTo

用途

このメソッドは、QueueGroup がディプロイメントされる OliteSite オブジェクトの定義を削除します。

構文

```
public void undefineDeployTo(java.lang.String OliteSite,
                             java.lang.String user)
                                throws AQException
```

addQueue

用途

このメソッドは、QueueGroup にキューを追加します。

構文

```
public void addQueue(java.lang.String[] owner,
                     java.lang.String[] queue)
                        throws AQException
```

removeQueue

用途

このメソッドは、QueueGroup からキューを削除します。

構文

```
public void removeQueue(java.lang.String[] owner,
                        java.lang.String[] queue)
                           throws AQException
```

AQLiteDeploymentOliteSite

このクラスは、OliteSite オブジェクトをディプロイメント用に作成、準備するために使用します。

コンストラクタ

用途

このメソッドは、AQLiteDeploymentOliteSite クラスを構成します。2 番目の構文では owner、queue および isLocal パラメータにデフォルトを使用します。

構文 1

```
public AQLiteDeploymentOliteSite(java.lang.Object obj,
                                 java.lang.String oliteSiteID,
                                 java.lang.String applicationName,
                                 java.lang.String owner,
                                 java.lang.String queue,
                                 boolean isLocal,
                                 java.lang.String comments)
                                    throws AQException
```

構文 2

```
public AQLiteDeploymentOliteSite(java.lang.Object obj,
                                 java.lang.String oliteSiteID,
                                 java.lang.String applicationName)
                                    throws AQException
```

パラメータ

パラメータ	説明
conn	Oracle データベース接続のオブジェクト。
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID。
applicationName	指定された OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ。
owner	ディプロイメント・メッセージ・キューの所有者。デフォルトを使用するときは、NULL に設定します。
queue	ディプロイメント・メッセージ・キュー。デフォルトを使用するときは、NULL に設定します。
isLocal	OliteSite のキュー・タイプ。
comments	関連するコメント。

drop

用途

このメソッドは、OliteSite オブジェクトを削除します。

構文

```
public void drop()
   throws AQException
```

getQueueGroup

用途

このメソッドは、OliteSite オブジェクト内に定義された QueueGroup オブジェクトを取得します。2 番目の構文は、OliteSite オブジェクト内に定義されたすべての QueueGroup オブジェクトを取得します。

構文 1

```
public AQLiteDeploymentQueueGroup getQueueGroup(java.lang.String groupName)
   throws AQException
```

構文 2

```
public java.lang.String[] getQueueGroup()
   throws AQException
```

戻り値

QueueGroup オブジェクトが 1 つも存在しない場合、NULL を返します。

パラメータ

パラメータ	説明
groupName	グループの名前
queueGroup	QueueGroup オブジェクトの名前

defineQueueGroup

用途

このメソッドは、OliteSite オブジェクトに QueueGroup オブジェクトを定義します。

構文 1

```
public void defineQueueGroup(java.lang.String groupName)
   throws AQException
```

構文 2

```
public void defineQueueGroup(AQLiteDeploymentQueueGroup queueGroup)
   throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前
queueGroup	QueueGroup オブジェクト

isDefinedQueueGroup

用途

このメソッドは、OliteSite オブジェクトに QueueGroup オブジェクトが定義されているかチェックします。

構文

```
public boolean isDefinedQueueGroup(java.lang.String groupName)
   throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前

undefineQueueGroup

用途

このメソッドは、OliteSite オブジェクトの QueueGroup オブジェクトの定義を解除します。

構文 1

```
public void undefineQueueGroup(java.lang.String groupName)
   throws AQException
```

構文 2

```
public void undefineQueueGroup(AQLiteDeploymentQueueGroup queueGroup)
   throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前
queueGroup	QueueGroup オブジェクト

instantiate

用途

このメソッドは、OliteSite オブジェクトのディプロイメントのインスタンスを生成します。groupName オブジェクトが指定されないときは、OliteSite オブジェクトに定義されているすべてのグループのインスタンスが生成されます。

構文 1

```
public void instantiate(java.lang.String groupName)
   throws AQException
```

構文 2

```
public void instantiate()
   throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前

deInstantiate

用途

このメソッドは、OliteSite オブジェクトに対するディプロイメント・インスタンスを削除します。groupName オブジェクトが指定されないときは、OliteSite オブジェクトに定義されているすべてのグループのインスタンスが削除されます。

構文 1

```
public void deInstantiate(java.lang.String groupName)
   throws AQException
```

構文 2

```
public void deInstantiate()
   throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前

AQLiteDeploymentClient

このクラスは、Oracle Lite クライアント端末上でディプロイメントを実行するために使用します。

コンストラクタ

用途

このメソッドは、AQLiteDeploymentClient クラスを構成します。

構文 1

```
public AQLiteDeploymentClient(oracle.lite.AQ.AQLiteSession aqConn,
                              java.lang.String applName,
                              AQLitePropOraConn ora_svr_db,
                              AQLitePropagation propagator,
                              java.lang.String oliteSiteID)
                                 throws AQException
```

パラメータ

パラメータ	説明
aqConn	AQ Lite 接続
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID
applName	指定された OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ
ora_svr_db	Oracle8*i* サーバーのデータベース接続オブジェクト
propagator	伝播オブジェクト

execDeployment

用途

このメソッドは、AQ Lite クライアント・サイトのディプロイメントを実行します。

構文

```
public void execDeployment(boolean creCat,
                           boolean propMsg)
                              throws AQException
```

パラメータ

パラメータ	説明
creCat	AQ Lite のカタログ情報を作成します。
propMsg	すでに存在するメッセージをディプロイメント時に伝播します。

ディプロイメント・プロシージャ

この項では、マスター / クライアント・キュー構成に基づいて AQ Lite キューイング・システムをディプロイメントするために必要なステップを詳細に説明します。サンプル・コードを使用して、各ステップを示します。

サーバー側のディプロイメント・タスク

次の作業は、Oracle Lite クライアント端末にディプロイメントする前に、Oracle8*i* サーバー上で実行しておく必要があります。

AQ Lite のディプロイメントを実行するための Oracle8*i* サーバーを準備する

1. Oracle8*i* AQ がサーバーにインストールされ、使用可能になっていることを確認する。

> **注意： AQLiteO8.sql スクリプトを実行すると、既存の AQ Lite 情報がすべて Oracle8*i* データベースから削除されます。**

2. **AQLiteO8.sql** スクリプトを Oracle8*i* サーバー上で実行する。このスクリプトは、Oracle8*i* Lite インストレーション CD-ROM の **ORACLE_HOME/Mobile/SDK/AQLite/SQL** にあります。

3. 次の定数を **AQLiteCreDeplMsgQ.java** 用にカスタマイズする。

```
public static final String connstr:  Oracle8i connection string,
                                     "jdbc:oracle:thin:@dlsun368:5521:aqj"
public static final String orauser:  Oracle8i user ID, "system"
public static final String orapswd:  Oracle8i user password, "manager"
```

4. **AQLiteCreDeplMsgQ.java** をコンパイルして、実行する。

キュー・テーブルおよびマスター・キューを作成する

キュー・テーブルとマスター・キューを Oracle8*i* サーバーに作成します。キュー・テーブルとキューの作成の詳細は、第 3 章「AQ Lite キュー・オブジェクトの管理」を参照してください。

マスター・キューにサブスクライバとパブリッシュの関係を指定する

Oracle8*i* サーバーのマスター・キューに、サブスクライバ名を追加します。サブスクライバの追加の詳細は、第 3 章「AQ Lite キュー・オブジェクトの管理」を参照してください。

ディプロイメント関係を定義する

1. 次のように AQLiteDeploymentAdmin の createQueueGroup メソッドを使用して、QueueGroup オブジェクトを作成し、キューを割り当てる。

```
AQLiteDeploymentAdmin admin = new AQLiteDeploymentAdmin(jdbc_conn);
AQLiteDeploymentQueueGroup qg = admin.createQueueGroup
   ("Queue Group 1", "this is for order queue", q_owner_array,
      q_name array);
```

2. OliteSite オブジェクトを作成して、OliteSite-QueueGroup 関係を定義する。

```
AQLiteDeploymentOliteSite os =
admin.createOliteSite("OliteSite123",  // OliteSite ID
                      "sampUser",    // Consumer name of Oracle Lite App
                       null, null, true, "for AQLite Sample Application" );
os.defineQueueGroup(qg);
```

各 OliteSite-QueueGroup 関係ごとに個別にコールして、定義する必要があります。

サーバーにディプロイメント・インスタンスを生成する

次のように、ディプロイメント・インスタンスを生成します。

```
AQLiteDeploymentOliteSite os;
.
.
.
os.instantiate();
```

これによって、各 OliteSite オブジェクトのすべてのディプロイメント情報が連結され、1 つのディプロイメント・メッセージになります。このメッセージは、事前に作成された「AQLITE$_DeplMessages」キューに格納されます。これで、サーバーは各クライアント端末からの接続とディプロイメントの実行に対する準備ができました。

クライアントからのディプロイメントの実行

ディプロイメントは、キューイング・システムの各 Oracle Lite クライアント端末から 1 度実行する必要があります。

1. クライアント端末と Oracle8*i* サーバーの接続を確立する。
2. 次のように、クライアント端末からのディプロイメントを実行する。

```
AQLitePropOraConn o8_conn =
    new AQLitePropOraConn("Oracle8i connection", "scott", "tiger",
    "dlsun368.us.oracle.com:5521:aqj");

AQLitePropagation propagator = new AQLitePropagator();

    AQLiteDeploymentClient dc =
       new AQLiteDeploymentClient((AQSession) aqConn,
                              "sampUser", // consumer application name
                               o8_conn,
                               propagator,
                               "OliteSite123" // Oracle Lite Site ID
                             );
        // create a new database on the client side.
        dc.execDeployment(true, false);
```

DeploymentClient オブジェクトは、次に示す機能を実行します。

- 初めてのディプロイメントの場合は、AQ カタログ情報を Oracle Lite データベースに作成する。
- Oracle8*i* サーバーから Oracle Lite クライアントに、キュー・テーブルとキューをディプロイメントする。
- メッセージがあれば、Oracle8*i* サーバーから Oracle Lite クライアントに伝播する。

execDeployment() の最初の引数が TRUE のときは、次に示すデータベース・カタログ表が作成されます。

表	説明
AQ$_AGENT	サブスクライバと受信者の記憶領域クラス
AQ$_QUEUES	AQ Lite キューのカタログ・クラス
AQ$_QUEUE_TABLES	AQ Lite キュー・テーブルのカタログ・クラス
AQ$_SEQ	AQ Lite で使用されるシーケンス・クラス
AQLITE$_DEPLMSGQT	ディプロイメント・メッセージのキュー・テーブルのクラス
AQ$_DBCONN	Oracle8*i* サーバー・サイト情報の記憶領域クラス
AQLITE$_PROPCAT	AQ Lite 伝播のカタログ・クラス
AQLITE$_PROPAGATIONLOG	AQ Lite 伝播のログ表
AQLITE$_PROPMSGLOG	AQ Lite 伝播メッセージのログ表
AQ$_QUEUESJAC (AQ$_QUEUES のサブクラス)	内部で使用するサブクラス

既存のディプロイメントの変更

キューイング・システムをディプロイメントした後で、キューやコンシューマを追加または削除してキューイング・システムを変更することが必要な場合があります。これらの変更を行うためには、再度ディプロイメントを実行する必要があります。

既存のディプロイメントを変更するためには、一般に次のステップが必要になります。

1. ディプロイメント定義を変更する。
2. すべての OliteSite オブジェクトのインスタンスを再生成する。
3. キューを削除する場合は、キューまたはキュー・テーブルを Oracle Lite クライアントから削除する。
4. 各クライアント端末からディプロイメントを再度実行する。

例 : 新しいキューを追加する

次の例は、既存のディプロイメントに新しいキューを追加する方法です。

ディプロイメント定義を変更する

1. キュー名の配列とキュー所有者の配列に要素を 1 つ追加する。

```
String[] qn = new String [2];
String[] qo = new String [2];
  qn[0] = "queue1";
  qn[1] = "newQueue2";
  qo[0] = "scott";
  qo[1] = "scott";
```

2. 新しいキューと新しい所有者によって、QueueGroup を作成する。

```
AQLiteDeploymentQueueGroup qg =  admin.getQueueGroup("QueueGroup 1");
   qg.addQueue("Queue Group 2",            // assign Queue group name
               "this is for order queue", // Comment
                qo,                       // Queue owner name(s)
                qn                        // Queue name(s)
              );
```

すべての OliteSite オブジェクトのインスタンスを再生成する

次のように、OliteSite オブジェクトのインスタンスを再生成します。

```
new_os1.deinstantiate();
new_os1.instantiate();
new_os1.deinstantiate();
new_os2.instantiate();
```

各クライアント端末からディプロイメントを再度実行する

新しいキュー情報によってキューイング・システムを更新するために、各クライアント端末からディプロイメントを再度実行する必要があります。詳細は、「クライアントからのディプロイメントの実行」を参照してください。

例 : 新しいコンシューマを追加する

次の例は、既存のディプロイメントに新しいコンシューマを追加する方法です。

ディプロイメント定義を変更する

新しいコンシューマ名によって、新しい OliteSite オブジェクトを作成します。

```
AQLiteDeploymentQueueGroup qg = admin.getQueueGroup("QueueGroup 1");

AQLiteDeploymentOliteSite old_os = admin.getOliteSite("OliteSite123");
  old_os.deinstantiate(qg);

AQLiteDeploymentOliteSite new_os1 =
   admin.createOliteSite("OliteSite123",  // OliteSite ID
                         "Consumer1",    // Consumer name of Olite App
                          null,
                          null,
                          true,
                          "for AQLite Sample Application"
                        );

 AQLiteDeploymentOliteSite new_os2 =
   admin.createOliteSite("OliteSite123",  // OliteSite ID
                        "newConsumer",    // Consumer name of Olite App
                         null,
                         null,
                         true,
                         "for AQLite Sample Application"
                        );

 new_os1.defineQueueGroup(qg);
 new_os2.defineQueueGroup(qg);
```

すべての OliteSite オブジェクトのインスタンスを生成する

次のように、新しい OliteSite オブジェクトのインスタンスを生成します。

```
new_os1.instantiate();
new_os2.instantiate();
```

新しいコンシューマを追加した後は、ディプロイメントを再度実行する必要はありません。

エラー・メッセージとアドバイザ・メッセージ

AQ Lite 使用中のエラー・メッセージおよびアドバイザ・メッセージの詳細は、『Oracle9*i* Lite メッセージ・リファレンス』を参照してください。

3

AQ Lite キュー・オブジェクトの管理

この章では、AQ および AQ Lite Java API を使用して、AQ Lite キュー・オブジェクトを管理する方法について説明します。この章で取り上げるメソッドの使用方法の詳細は、付録 A「AQ Lite Java API」を参照してください。

この章で説明するタスクは、Oracle8*i* サーバー上の PL/SQL または Oracle コール・インタフェース（OCI）でも実行できます。詳細は、『Oracle8*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』を参照してください。

この章で説明する内容は、次のとおりです。

- AQ セッションの作成
- キュー・オブジェクトの概要
- キュー・オブジェクトの管理
- キュー・テーブルの作成
- キュー・テーブルの削除
- キューの作成
- キューの削除
- サブスクライバの追加
- サブスクライバの削除
- Queue オブジェクトおよび QueueTable オブジェクトの取得

AQ セッションの作成

この章で説明する操作は、必ず AQ セッションを作成してから実行する必要があります。AQ Lite に AQ セッションを作成する方法は、次のとおりです。

1. Oracle Lite Java アクセス・クラス（JAC）接続を確立する。
2. `AQLiteDriver` を取得する。

次の例では、AQ Lite に AQ セッションを作成します。

```
// Establish Oracle Lite Connection
POLConnection olite_conn =
POLConnection.getConnection("C:¥orant¥oldb¥polite.odb",0);

// Initiate AQLite Driver
AQLiteDriver aq_driver = new AQLiteDriver();

// Create AQ Session
AQSession aq_session = aq_driver.createAQSession(olite_conn);
```

Oracle8*i* AQ では、Oracle JDBC 接続を使用できます。Oracle8*i* `AQDriver` を取得するには、`AQOracleDriver` クラスを使用します。また、`AQDriverManager` クラスを使用して、異なる AQ ドライバを取り扱ったり、AQ セッションを確立することもできます。

キュー・オブジェクトの概要

キューとは、メッセージのリポジトリです。キューは、キュー・テーブルに格納されます。キュー・テーブルは、1 つ以上のキューを含むデータベース表です。どのキューにもサブスクライバ・リストがあります。サブスクライバ・リストには、そのキューのメッセージにアクセスできるアプリケーションまたはユーザーのリストが含まれています。

AQ Lite のキューとキュー・テーブルの情報は、次の、Oracle Lite 内のカタログ表に格納されています。

AQ$_Queue_Tables

列の名前	説明	NULL は可能か？	型
OWNER	キューのスキーマ名		VARCHAR2(30)
QUEUE_TABLE	キュー・テーブルの名前		VARCHAR2(30)
TYPE	ペイロードの型		VARCHAR2(7)
OBJECT_TYPE	オブジェクトの型		VARCHAR2(61)

列の名前	説明	NULL は可能か？	型
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)
USER_COMMENT	キュー・テーブルに対するユーザー・コメント		VARCHAR2(50)

AQ$_Queues

列の名前	説明	NULL は可能か？	型
OWNER	キューのスキーマ名	NOT NULL	VARCHAR2(30)
NAME	キューの名前	NOT NULL	VARCHAR2(30)
QUEUE_TABLE	このキューが格納されるキュー・テーブル	NOT NULL	VARCHAR2(30)
QID	一意のキュー識別子	NOT NULL	NUMBER
QUEUE_TYPE	キューのタイプ		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)

Oracle8*i* キューおよびキュー・テーブルの情報

Oracle8*i* では、次のビューでキュー情報およびキュー・テーブル情報にアクセスできます。

- DBA_queue_tables
- USER_queue_tables
- ALL_queue_tables
- DBA_queues
- USER_queues
- ALL_queues

Oracle8*i* では、サブスクライバ情報は AQ$<*q_tab_name*>_s に格納されます。Oracle Lite では、サブスクライバ情報は AQ$_QueuesJAC Class に格納されます。

キュー・オブジェクトの管理

この項では、AQ Lite Java API を使用してキュー・オブジェクトおよびキュー・テーブル・オブジェクトを作成したり管理したりするために必要な情報を示します。キューおよびキュー・テーブルを管理するために使用するメソッドは、AQ Lite 構成に依存します。マスター / クライアント構成を管理する場合は、一般的に管理タスクを Oracle8*i* サーバー上で実行し、Oracle Lite クライアント端末にディプロイメントします。スタンドアロン構成を管理する場合は、キュー・オブジェクトおよびキュー・テーブル・オブジェクトを Oracle Lite クライアントから管理する必要があります。次の各項を参照してください。

スタンドアロン・キュー・オブジェクト作成の準備

Oracle Lite クライアントにスタンドアロンのキューおよびキュー・テーブルを作成する前に、次のコードに示されているような、サーバーを使用しないディプロイメントをクライアント端末上で実行して、必要なカタログを作成する必要があります。

```
AQLIteDeploymentClient deplc =
AQLIteDeploymentClient(aqConn,anyName", null, null, "anyID");
deplc.execDeployment(true, false);
```

後で、スタンドアロン・キューをマスター / クライアント・キュー構成の中で使用する場合は、`execDeployment` メソッドの最初のパラメータを「FALSE」に設定してコールし、再度サーバーを使用しないディプロイメントを実行します。

キュー・テーブルの作成

この項では、AQ および AQ Lite Java API を使用して、キュー・テーブルを作成する方法について説明します。キュー・テーブルは、Oracle8*i* サーバー上の PL/SQL または Oracle コール・インタフェース（OCI）でも作成できます。詳細は、『Oracle8*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』を参照してください。

マスター / クライアント構成を管理する場合は、Oracle8*i* サーバー上にキュー・テーブルを作成し、各クライアント端末にディプロイメントする必要があります。詳細は、第 2 章「AQ Lite のディプロイメント」を参照してください。

スタンドアロン・クライアント構成を管理する場合は、AQ Lite Java API を使用して、キュー・テーブルをクライアント端末上に作成する必要があります。

createQueueTable

用途

Oracle8*i* サーバーまたは Oracle Lite クライアント上にキュー・テーブルを作成します。

構文

```
public AQQueueTable createQueueTable(java.lang.String owner,
                                      java.lang.String name,
                                      AQQueueTableProperty property)
                                         throws AQException
```

メソッド

次の表は、`AQQueueTableProperty` に表プロパティを設定するために使用するメソッドです。

メソッド	説明
`setMultiConsumer`	キューに複数のコンシューマを設定できるかどうかを指定します。次のオプションが指定できます。TRUE: 設定可能、FALSE: 設定不可（デフォルト）
`setStorageClause`	記憶領域パラメータを設定します。Oracle8*i* サーバー・キューでのみサポートされます。このキュー・プロパティは、ディプロイメント時には Oracle Lite キュー・テーブルに設定されません。
`setCompatible`	Oracle8 の下位互換性オプションを指定します。Oracle8*i* サーバー・キューでのみサポートされます。このキュー・プロパティは、ディプロイメント時には Oracle Lite キュー・テーブルに設定されません。

メソッド	説明
setMessageGrouping	メッセージのグループ化に使用されるオプションを指定します。TRANSACTIONAL の場合は、同一トランザクションにエンキューされたメッセージのみがデキューされます。NONE の場合は、メッセージがなくなるまで、すべてのメッセージがデキューされます。デフォルトは NONE です。
setPrimaryInstance	そのキュー・テーブルをプライマリ・インスタンスとして設定するかどうかを指定します。Oracle8*i* サーバー・キューでのみサポートされます。ディプロイメントでは、このプロパティを Oracle Lite キュー・テーブルに設定しません。
setSecondaryInstance	複数インスタンスを認めるかどうかを指定します。Oracle8*i* サーバー・キューでのみサポートされます。ディプロイメントでは、このプロパティを Oracle Lite キュー・テーブルに設定しません。
setSortOrder	キュー内のメッセージのソート順を設定します。メッセージは、エンキュ時刻（デフォルト）または優先順位によってソートされます。
setComment	コメントを指定します。

例

```
AQSession aq_conn;
.
.
.
AQQueueTableProperty qt_property = new AQQueueTableProperty("RAW");
qt_prop.setMultiConsumer(true);   // allow more than one consumer
qt_prop.setStorageClause("");     // use default storage parameter
qt_prop.setCompatible("");        // backward compatibility
qt_prop.setMessageGrouping
   (AQQueueTableProperty.NONE);    // no message grouping
qt_prop.setPrimaryInstance(1);    // set as primary instance
qt_prop.setSecondaryInstance(0);  // do not allow multiple instance
qt_prop.setSortOrder("ENQ_TIME"); // order messages by enqueue time
qt_prop.setComment("AQLite Sample Application");  // comments

AQQueueTable q_table = aq_conn.createQueueTable("Scott", // queue user
                                                "Q_Table1", // queue name
                                                 qt_property);
```

注意

AQ Lite では、次のメソッドはサポートされていません。

- `setStorageClause();`
- `setCompatible();`
- `setPrimaryInstance();`
- `setSecondaryInstance();`

キュー・テーブルのプロパティを取り出すために使用するメソッドの詳細は、付録 A「AQ Lite Java API」を参照してください

キュー・テーブルの削除

この項では、AQ Lite Java API を使用して、キュー・テーブルを削除する方法について説明します。キュー・テーブルは、Oracle8*i* サーバー上の PL/SQL または Oracle コール・インタフェース（OCI）でも削除できます。詳細は、『Oracle8*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』を参照してください。

キュー・テーブルの構成にかかわらず、クライアントとサーバーの両方でキュー・テーブルを削除する必要があります。クライアント端末に常駐しているキュー・テーブルは、Oracle8*i* サーバー・インタフェースからは削除できません。キュー・テーブルを削除する前に、その表に属しているすべてのキューを削除する必要があります。

drop()

用途

Oracle8*i* サーバーまたは Oracle Lite クライアント上のキュー・テーブルを削除します。

構文

```
public void drop(boolean force) throws AQException
```

例

```
AQQueueTable queue_table;
AQQueue queue;
.
.
.
queue.drop();
queue_table.drop(true);
```

キューの作成

この項では、AQ Lite Java API を使用して、キューを作成する方法について説明します。キューは、Oracle8*i* サーバー上の PL/SQL または Oracle コール・インタフェース（OCI）でも作成できます。詳細は、『Oracle8*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』を参照してください。

マスター / クライアント構成を管理する場合は、Oracle8*i* サーバー上にキューを作成し、各クライアント端末にディプロイメントする必要があります。詳細は、第 2 章「AQ Lite のディプロイメント」を参照してください。

メッセージをエンキュおよびデキューできるようにするためには、Oracle8*i* メソッドを使用してキューを開始する必要があります。`start()` メソッドは、Oracle Lite では操作できません。このメソッドを使用しても、クライアント・キューには何の効果もありません。

スタンドアロン・キュー構成を管理する場合は、AQ Lite Java API を使用して、キューをクライアント端末上に作成する必要があります。

createQueue

用途

Oracle8*i* サーバーまたは Oracle Lite クライアント上にキューを作成します。

構文

`AQSession` の場合

```
public AQQueue createQueue(AQQueueTable q_table,
                           java.lang.String q_name,
                           AQQueueProperty q_property)
                              throws AQException
```

`AQQueueTable` の場合

```
public AQQueue createQueue(java.lang.String queue_name,
                           AQQueueProperty q_property)
                              throws AQException
```

例

```
AQQueueTable q_table;
.
.
.
AQQueueProperty q_property = new AQQueueProperty();
q_property.setQueueType(AQQueueProperty.NORMAL_QUEUE);
q_property.setRetentionTime(0.0) ;
```

```
q_property.setComment("AQLite Sample Application");

AQQueue queue = aq_conn.createQueue (q_table,      // Queue table object
                                     "Queue1",     // Queue name
                                     q_property);  // Queue property
queue.start();
```

メソッド

次の表では、AQQueuePropertyにキュー・プロパティを設定するために使用するメソッドを説明します。

メソッド	説明
setQueueType	キューのタイプを設定します。次のオプションが指定できます。NORMAL_QUEUE または EXCEPTION_QUEUE 例外キューは、Oracle Lite キューでは使用できません。例外キューは、Oracle Lite クライアントにはディプロイメントできません。
setComment	コメントを指定します。

注意

AQ Lite では、次のメソッドはサポートされていません。

- setQueueType()
- setRentionTime()
- setMaxRetries
- setRetryInterval()

キューの削除

この項では、AQ Lite Java API を使用して、キューを削除する方法について説明します。キューは、Oracle8*i* サーバー上の PL/SQL または Oracle コール・インタフェース（OCI）でも削除できます。詳細は、『Oracle8*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』を参照してください。

キューの構成にかかわらず、クライアントとサーバーの両方でキューを削除する必要があります。クライアント端末に常駐しているキューは、Oracle8*i* サーバー・インタフェースからは削除できません。Oracle8*i* サーバーからキューを削除する前に、そのキューを停止する必要があります。stop() メソッドは、Oracle Lite では操作できません。このメソッドを使用しても、クライアント・キューには何の効果もありません。

dropQueue

用途

Oracle8*i* サーバーまたは Oracle Lite クライアント上のキューを削除します。

構文

AQQueueTable の場合

```
public void dropQueue(java.lang.String queue_name)
   throws AQException
```

AQQueue の場合

```
public void drop()
   throws AQException
```

例

```
AQQueue queue;
.
.
.
queue.stop();
queue.drop();
```

サブスクライバの追加

この項では、AQ Lite Java API を使用して、サブスクライバを追加する方法について説明します。サブスクライバは、Oracle8*i* サーバー上の PL/SQL または Oracle コール・インタフェース（OCI）でも追加できます。詳細は、『Oracle8*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』を参照してください。

サブスクライバは、キュー内の全メッセージのデフォルトの受信者です。各キューは、複数のサブスクライバを持つことができます。Oracle8*i* サーバー上またはクライアント端末上のアプリケーションが、サブスクライバになることができます。サブスクライバ・リストは、エンキュ時にメッセージに受信者リストを追加することでオーバーライドできます。

Oracle8*i* サーバー・キューのサブスクライバは、どの Oracle8*i* サーバーまたは Oracle Lite クライアントにも常駐できます。Oracle Lite クライアントから Oracle8*i* サーバー・キューに伝播されたメッセージは、別の Oracle8*i* サーバー・キューまたは Oracle Lite クライアントにもディプロイメントできます。

マスター / クライアント構成では、サーバーからのメッセージはプロパゲータによって受信者に割り当てられます。クライアントからサーバーに伝播されたメッセージは、エンキュ時に受信者が設定されていない場合は、Oracle8*i* マスター・キューのサブスクライバ・リストを使用します。

スタンドアロン構成では、ローカル端末上のすべてのメッセージ・コンシューマがキュー・サブスクライバ・リストに含まれている必要があります。

サブスクライバは AQAgent 型です。サブスクライバを追加するには、最初に AQAgent オブジェクトを作成します。

addSubscriber

用途

サブスクライバを Oracle Lite キューに追加します。

構文

```
public void addSubscriber(AQAgent subscriber,
                          java.lang.String rule)
                             throws AQException
```

例

```
//Create AQAgent objects
AQAgent subscriber1 = new AQAgent("order_entry_consumer", null, 0);
AQAgent subscriber2 = new AQAgent("delivery_consumer", null, 0);

//Add subscribers
AQQueue queue;
.
.
.
queue.addSubscriber(subscriber1, null);
queue.addSubscriber(subscriber2, null);
```

サブスクライバの削除

この項では、AQ Lite Java API を使用して、キューからサブスクライバを削除する方法について説明します。サブスクライバは、Oracle8*i* サーバー上の PL/SQL または Oracle コール・インタフェース（OCI）でも削除できます。詳細は、『Oracle8*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』を参照してください。

removeSubscriber

用途

サブスクライバを Oracle Lite キューから削除します。

構文

```
public void removeSubscriber(AQAgent subscriber)
   throws AQException
```

例

```
AQQueue queue;
.
.
.
AQAgent[] subscriber = queue.getSubscribers();
queue.removeSubscriber(subscriber[0]);
```

Queue オブジェクトおよび QueueTable オブジェクトの取得

次の表は、Queue および QueueTable オブジェクトを取得するためのメソッドの位置をまとめたものです。これらのメソッドの詳細は、付録 A「AQ Lite Java API」を参照してください。

メソッド	インタフェース
getQueueTable	AQSession
getQueue	AQSession および AQQueueTable
getSubscribers	AQQueue

4

AQ Lite の操作インタフェース

この章では、AQ Lite のメッセージをエンキュおよびデキューするために使用する操作インタフェースを説明します。説明する内容は、次のとおりです。

- AQ セッションの作成
- メッセージの作成
- メッセージのエンキュ
- エンキュ・オプションの指定
- メッセージ・プロパティの指定
- エンキュ用の RAW 型ペイロードの準備
- enqueue() と propEequeue()
- エンキュの例
- メッセージのデキュー
- デキュー・オプションの指定
- デキューしたメッセージからの RAW 型ペイロードの取得
- dequeue() と propDequeue()
- メッセージ終了の際の例外処理

AQ セッションの作成

この章で説明する操作は、必ず AQ セッションを作成してから実行する必要があります。AQ Lite に AQ セッションを作成する方法は、次のとおりです。

1. Oracle Lite Java アクセス・クラス（JAC）接続を確立する。
2. AQLiteDriver をロードする。

次の例では、AQ Lite に AQ セッションを作成します。

```
// Establish Oracle Lite Connection
POLConnection olite_conn =
   POLConnection.getConnection("C:/orant/oldb35/polite.odb",0);

// Initiate AQLiteDriver
AQLiteDriver aq_driver = new AQLiteDriver();

// Create AQ Session
AQSession aq_session = aq_driver.createAQSession(olite_conn);
```

Oracle8*i* AQ では、Oracle JDBC 接続を使用できます。Oracle8*i* の AQDriver を取得するには、AQOracleDriver クラスを使用します。また、AQDriverManager クラスを使用して、異なる AQ ドライバを取り扱ったり、対応する AQ セッションを確立することもできます。

メッセージの作成

用途

エンキュ操作用のメッセージ・オブジェクトを準備します。メッセージの作成には、AQQueue インタフェースを使用します。

構文

```
public AQMessage createMessage()
   throws AQException
```

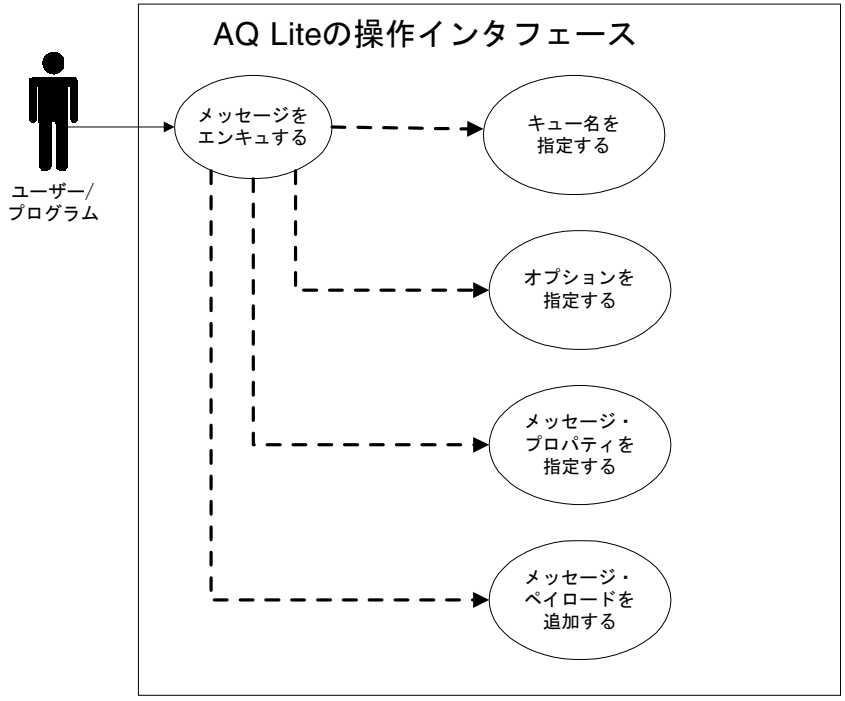
メソッドの概要

次のメソッドが AQMessage インタフェースに実装されています。

メソッド	説明
getMessageId()	メッセージ ID を取得します。
getMessageProperty()	メッセージ・プロパティを取得します。

メソッド	説明
getObjectPayload()	オブジェクト型ペイロードを取得します。
getRawPayload()	RAW 型ペイロードを取得します。
setMessageProperty	メッセージ・プロパティを設定します。
setObjectPayload()	オブジェクト型ペイロードを設定します。
setRawPayload()	RAW 型ペイロード・オブジェクト・ペイロードの設定は、現在のリリースではサポートされません。付録 C「AQ Lite の制約」を参照してください。

メッセージのエンキュ



用途

指定したキューにメッセージを追加します。メッセージのエンキュには、AQQueue インタフェースを使用します。

構文

```
public byte[] enqueue(AQEnqueueOption enq_option,
                      AQMessage message) throws AQException
```

パラメータ

パラメータ	説明
enqueue_options	エンキュ操作で使用できるオプションを設定します。
message	メッセージ・プロパティを指定します。
Return byte[]:	システムで生成されたメッセージの識別子。これは、デキュー時にメッセージを識別するためのグローバルな一意識別子です。

エンキュ・オプションの指定

用途

エンキュ操作で使用できるオプションを指定します。エンキュ・オプションの指定には、AQEnqueueOption クラスを使用します。

構文

```
java.lang.Object
    |
    +----oracle.AQ.AQEnqueueOption

public class AQEnqueueOption
extends java.lang.Object
```

パラメータ

パラメータ	説明
visibility	エンキュ要求のトランザクション動作を指定します。 ON_COMMIT: エンキュはカレント・トランザクションの一部です。トランザクションがコミットされると、操作は完了です。これがデフォルトです。 IMMEDIATE: エンキュはカレント・トランザクションの一部ではありません。この操作は、それ自体で 1 つのトランザクションになります。
relative_msgid	AQLite ではサポートされません。関連するメソッドは無視されます。
sequence_deviation	AQLite ではサポートされません。関連するメソッドは無視されます。

フィールドの概要

- static int VISIBILITY_IMMEDIATE
- static int VISIBILITY_ONCOMMIT

コンストラクタの概要

- AQEnqueueOption(int visibility, byte[] relative_msgid, int sequence_deviation)
- AQEnqueueOption(int visibility, byte[] relative_msgid, java.lang.Integer sequence_deviation)
- AQEnqueueOption()

メソッドの概要

メソッド	説明
getRelMessageId()	関連するメッセージ ID を取得します。
getSequenceDeviation()	順序逸脱を取得します。

メソッド	説明
getVisibility()	可視性を取得します。戻り値: AQConstants.VISIBILITY_IMMEDIATE または AQConstants.VISIBILITY_ONCOMMIT
setSequenceDeviation()	順序逸脱および関連するメッセージ ID を設定します。
setVisibility()	可視性を設定します。

メッセージ・プロパティの指定

用途

メッセージ・プロパティには、AQ Lite が個々のメッセージを管理するために使用する情報が記述されています。このプロパティは、AQMessageProperty クラスを使用してエンキュ時に設定されます。メッセージ・プロパティの値はデキューのときに返されます。

構文

```
java.lang.Object
    |
    +----oracle.AQ.AQMessageProperty

public class AQMessageProperty
extends java.lang.Object
```

パラメータ

パラメータ	説明
priority	メッセージの優先順位の値を指定または返します。数値が小さいほど、優先順位が高いことを表します。優先順位には、負数も含め、任意の数値を指定できます。
delay	AQLite ではサポートされません。関連するメソッドは無視されます。
expiration	AQLite ではサポートされません。関連するメソッドは無視されます。
correlation	メッセージ間の関連付けのためにプロデューサ（エンキュ元）によって設定された識別子

パラメータ	説明
attempts	AQLite ではサポートされません。関連するメソッドは無視されます。
recipient_list	このパラメータは、複数コンシューマに対応したキューに対してのみ有効です。デフォルトの受信者は、キューのサブスクライバです。このパラメータは、デキュー時にはコンシューマに対して返されません。
exception_queue	AQLite ではサポートされません。関連するメソッドは無視されます。
enqueue_time	メッセージがエンキュされた時刻を返します。この値はシステムによって決定され、ユーザーは設定できません。この値は、エンキュ時には設定できません。
state	AQLite ではサポートされません。関連するメソッドは無視されます。
sender_id	アプリケーションによって指定された送信者識別子を指定、または返します。 デフォルトは NULL です。
original_msgid	AQLite ではサポートされません。関連するメソッドは無視されます。 デフォルトは NULL です。

コンストラクタの概要

- AQMessageProperty()

メソッドの概要

メソッド	説明
getAttempts()	試行回数を取得します。
getCorrelation()	相関関係を取得します。
getDelay()	遅延を取得します。
getEnqueueTime()	エンキュ時刻を取得します。
getExceptionQueue()	例外キューの名前を取得します。
getExpiration()	期限切れを取得します。
getOrigMessageId()	元のメッセージ ID を取得します。

メソッド	説明
getPriority()	メッセージの優先順位を取得します。
getRecipientList()	受信者リストを取得します。
getSender()	送信者を取得します。
getState()	メッセージの状態を取得します。 戻り値：STATE_READY、STATE_WAITING、STATE_PROCESSED または STATE_EXPIRED
setCorrelation()	相関関係を設定します。
setDelay()	遅延を設定します。
setExceptionQueue()	例外キューの名前を設定します。
setExpiration()	期限切れを設定します。
setPriority()	メッセージの優先順位を設定します。
setRecipientList()	受信者リストを設定します。
setSender()	送信者を設定します。

使用上の注意

受信者リストを設定すると、そのキューのサブスクライバ・リストは上書きされます。

エンキュ用の RAW 型ペイロードの準備

AQ Lite のこのリリースでは、RAW 型メッセージ・ペイロードのみをサポートします。たとえば Java オブジェクトのような、非バイナリ・ストリームをエンキュする場合は、それをバイナリ・ストリームに変換して、RAW 型ペイロードに設定する必要があります。これを簡単に行うには、次の例のようにそのオブジェクトを直列化します。

```
//serializing the object
   ByteArrayOutputStream bArrOut = new ByteArrayOutputStream();
   ObjectOutputStream p = new ObjectOutputStream(bArrOut);
    p.writeObject((Object) mp);  //MP is a java object
    byte[] bArr = bArrOut.toByteArray();
    bArrOut.close();
    p.close();
   .
   .
   .
   msg.setRawPayload(msg_payload);
```

enqueue() と propEequeue()

AQ Lite キュー内のメッセージは、ローカル・アプリケーションでも、AQ Lite プロパゲータでもエンキュすることができます。AQ Lite プロパゲータは、サーバー上のマスター・キューとクライアント・キューの間でメッセージを交換します。AQ Lite は enqueue() および propEnqueue() を使用して、ローカル・アプリケーションがエンキュしたメッセージと AQ Lite プロパゲータがエンキュしたものとを区別します。enqueue() と propEnqueue() は AQQueue インタフェースに実装されます。

通常、AQ Lite プロパゲータは PropEnqueue() を使用して、サーバーからのメッセージをクライアント・キューにエンキュします。Oracle Lite クライアント上で実行されるアプリケーションは、enqueue() を使用してサーバー宛てのメッセージをローカル・キューに格納します。例外が 1 つあります。別のローカル・アプリケーションで使用されるメッセージをエンキュするためには、ローカル・アプリケーションは propEnqueue() を使用する必要があります。

構文

```
public byte[] propEnqueue(AQEnqueueOption enq_option,
                          AQMessage message)
                             throws AQException
```

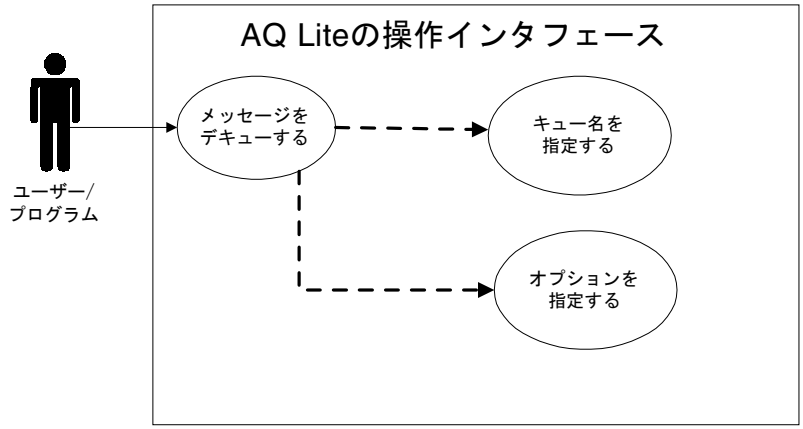
エンキュの例

次に、メッセージ・エンキュの例を示します。

```
void enq_order()
 {
        float[] up ={200, 800, 100, 900};
        try {
      AQQueue q = parent1.aqConn.getQueue(parent1.oliteUser,
                                          "AQLite_SampOrderQ");

      //get the order
      aqlOrder mp = new aqlOrder();
            /*
                byte                Order#[16];
                String              Customer;
                String              Item;
                int                 Qty;
                float               UnitPrice;
                float               Total;
                java.util.Date      OrderDate;
                String              Shipping;
                String              Payment;
                java.util.Date      ProcessingDate;
```

```
            String                Status; //NEW, PROCESSED, REJECT
          */

       // assign attribute values to the object mp
      .
      .
      .
      //serializing the object
      ByteArrayOutputStream bArrOut = new ByteArrayOutputStream();
      ObjectOutputStream p = new ObjectOutputStream(bArrOut);
       p.writeObject((Object) mp);
       byte[] bArr = bArrOut.toByteArray();
       bArrOut.close();
       p.close();

      AQEnqueueOption enq_option = new AQEnqueueOption();
      AQMessageProperty msg_prop = new AQMessageProperty();
      java.util.Vector v = new java.util.Vector(1);
      v.addElement((Object) new AQAgent("sampProc", null, 0));
      msg_prop.setRecipientList(v);

      AQMessage msg = q.createMessage();
      AQRawPayload msg_payload = msg.getRawPayload();
      msg_payload.setStream(bArr, bArr.length);
      msg.setMessageProperty(msg_prop);
      msg.setRawPayload(msg_payload);

      byte[] msg_id = q.enqueue(enq_option , msg);

    }
        catch (AQException e)
   {
        System.out.println(e);
   }
        catch (Exception e)
  {
        System.out.println(e);
   }
}
```

メッセージのデキュー



用途

指定したキューからメッセージを取り出します。メッセージのデキューには、AQQueue インタフェースを使用します。

構文

```
public AQMessage dequeue(AQDequeueOption dequeue_option) throws AQException
```

パラメータ

パラメータ	説明
queue_name	キュー名を指定します。
dequeue_options	定義については、「デキュー・オプションの指定」を参照してください。
Return_AQMessage	実際のメッセージです。

デキュー・オプションの指定

用途

デキュー操作で使用できるオプションを指定します。デキュー・オプションの指定には、AQDequeueOption クラスを使用します。

構文

```
java.lang.Object
     |
     +----oracle.AQ.AQDequeueOption
public class AQDequeueOption  extends java.lang.Object
```

パラメータ

パラメータ	説明
consumer_name	コンシューマの名前です。コンシューマ名が一致するメッセージのみがアクセスされます。複数コンシューマ用のキューでない場合には、このフィールドを NULL にする必要があります。
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。 BROWSE: メッセージをロックせずに読み取ります。これは「SELECT」文と等価です。 LOCKED: メッセージを読み取って、書込みロックを取得します。ロックは、そのトランザクションが終了するまで継続します。これは「SELECT FOR UPDATE」文と等価です。 REMOVE: メッセージを読み取って、更新または削除します。これがデフォルトです。メッセージは、retention プロパティに基づいて、キュー・テーブルに保存できます。

パラメータ	説明
navigation	取り出すメッセージの位置を指定します。まず最初に、位置が決まります。次に、検索基準が適用されます。最後に、メッセージが取り出されます。
	NEXT_MESSAGE: 取出し可能で検索基準に適合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・グループに属している場合、AQ では、検索基準に適合し、同じメッセージ・グループに属する、次の取出し可能メッセージを取り出します。これがデフォルトです。
	NEXT_TRANSACTION: 現行のトランザクション・グループの残りのメッセージ（存在する場合）をスキップして、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使用可能な場合にのみ使用できます。
	FIRST_MESSAGE: 取出し可能で検索基準に適合する、最初のメッセージを取り出します。このオプションを使用すると、位置がキューの先頭にリセットされます。
visibility	新規メッセージを、カレント・トランザクションの一部としてデキューするかどうかを指定します。visibility パラメータは、BROWSE モードを使用している場合には無視されます。
	ON_COMMIT: デキューはカレント・トランザクションの一部です。これがデフォルトです。
	IMMEDIATE: デキューされたメッセージは、カレント・トランザクションの一部ではありません。このメッセージは、それ自体が 1 つのトランザクションになります。
wait	NO_WAIT: 現時点では、唯一使用可能なオプションは「待機しない」です。
msgid	デキューされるメッセージのメッセージ識別子を指定します。
correlation	メッセージ間の関連付けのためにプロデューサ（エンキュ元）によって設定された識別子

使用上の注意

デキューするメッセージを選択するには 3 つの方法があります。

- ナビゲーション・モードによるデキュー（デキューの指定を、最初のメッセージ、次のメッセージ、または TRANSACTIONAL キュー・タイプでは次のトランザクション、にします）。
- メッセージ ID によるデキュー（一意の ID によってメッセージを指定します）。
- 相関 ID によるデキュー（相関 ID を指定します）。

ナビゲーション・モードでは、キューの最初のメッセージを選択し、「次のメッセージ」を選択して、キューの順序に従ってどのメッセージにでもナビゲートできます。

そのキューでトランザクションのグループ化が使用可能な場合は、ナビゲーション・オプションは次のようになります。

- 「最初のメッセージ」を選択すると、デキュー位置はキューの最初に設定されます。
- 順序に従った「次のメッセージ」を要求すると、そのトランザクションの次のメッセージが選択されます。
- 「次のトランザクション」を要求すると、次のトランザクションの最初のメッセージが選択されます。

相関識別子またはメッセージ識別子によってメッセージを選択すると、トランザクションのグループ化プロパティは無視されます。また、トランザクション内でメッセージのサブセットを選択してコミットしても、トランザクションのグループ化は無視されます。

フィールドの概要

- `static int DEQUEUE_BROWSE`
- `static int DEQUEUE_LOCKED`
- `static int DEQUEUE_REMOVE`
- `static int DEQUEUE_REMOVE_NODATA`
- `static int NAVIGATION_FIRST_MESSAGE`
- `static int NAVIGATION_NEXT_MESSAGE`
- `static int NAVIGATION_NEXT_TRANSACTION`
- `static int VISIBILITY_IMMEDIATE`
- `static int VISIBILITY_ONCOMMIT`
- `static int WAIT_FOREVER`
- `static int WAIT_NONE`

コンストラクタの概要

- `AQDequeueOption()`

メソッドの概要

メソッド	説明
getConsumerName()	コンシューマの名前を取得します。
getCorrelation()	相関 ID を取得します。
getDequeueMode()	デキュー・モードを取得します。戻り値 : DEQUEUE_BROWSE、DEQUEUE_LOCKED、DEQUEUE_REMOVE または DEQUEUE_REMOVE_NODATA
getMessageId()	メッセージ ID を取得します。
getNavigationMode()	ナビゲーション・モードを取得します。戻り値 : NAVIGATION_FIRST_MESSAGE、NAVIGATION_NEXT_MESSAGE または NAVIGATION_NEXT_TRANSACTION
getVisibility()	可視性を取得します。戻り値 : AQConstants.VISIBILITY_IMMEDIATE または AQConstants.VISIBILITY_ONCOMMIT
setConsumerName()	コンシューマの名前を設定します。
setCorrelation()	相関関係を設定します。
setDequeueMode()	デキュー・モードを設定します。
setMessageId()	メッセージ ID を設定します。
setNavigationMode()	ナビゲーション・モードを設定します。
setVisibility()	可視性を設定します。次の値を指定できます。AQConstants.VISIBILITY_IMMEDIATE または VISIBILITY_ONCOMMIT

デキューしたメッセージからの RAW 型ペイロードの取得

AQ Lite のこのリリースでは、RAW 型メッセージ・ペイロードのみをサポートします。デキューされたメッセージが、たとえば Java オブジェクトのような、非バイナリ・ストリームの場合は、それを RAW 型ペイロードからバイナリ・ストリームに再変換する必要があります。

```
AQMessage msg = q.dequeue(deq_option);
AQRawPayload raw_payload = msg.getRawPayload();
byte[] byte_array = raw_payload.getBytes();
ByteArrayInputStream b_stream = new ByteArrayInputStream(byte_array);
ObjectInputStream p = new ObjectInputStream((InputStream) b_stream);
MyJavaClass obj=p.read.Object();
```

dequeue() と propDequeue()

AQ Lite キューにあるメッセージは、ローカル・アプリケーションでも、サーバー上のマスター・キューとクライアント・キューの間でメッセージを交換するプロパゲータでもデキューできます。AQ Lite は dequeue() および propDequeue() を使用して、ローカル・アプリケーションによるデキューとプロパゲータによるデキューを区別します。

通常、AQ Lite プロパゲータは propDequeue() を使用して、クライアント・キューからのメッセージをデキューしてサーバーに伝播します。Oracle Lite クライアント上で実行されるアプリケーションは、dequeue() を使用してローカル・キューからメッセージを取り出します。propDequeue() は dequeue() と同じ構文を使用します。

構文

```
public AQMessage propDequeue(AQDequeueOption deq_option)
   throws AQException
```

メッセージ終了の際の例外処理

dequeue() と propDequeue() を使用しているときに、キューにメッセージがなくなると次の例外が発生します。

```
AQException (error_code AQLiteConstants.AQLiteNoMoreMessage)
```

したがって、この例に示すように AQException を検出して処理する必要があります。

```
AQSession aq_conn;
.
.
.
AQQueue q = aqSession.getQueue("scott",    // queue owner
                               "Queue1");  // queue name
AQDequeueOption deq_option = new AQDequeueOption();

deq_option.setNavigationMode(AQDequeueOption.NAVIGATION_FIRST_MESSAGE);
  deq_option.setDequeueMode(AQDequeueOption.DEQUEUE_REMOVE);
  deq_option.setConsumerName("SampUser");

  try {
    AQMessage msg2 = q.dequeue(deq_option);
  }
  catch (AQException e) {
    if (e.getErrorCode() == AQLiteConstants.AQLiteNoMoreMessage)
   q_message = null;
    else
   throw new AQException (e.getErrorCode(), e.getMessage());
  }
```

```
deq_option.setNavigationMode(AQDequeueOption.NAVIGATION_NEXT_MESSAGE);

while (true) {
  try {
     AQMessage msg2 = q.dequeue(deq_option);
  }
  catch (AQException e) {
    if (e.getErrorCode() == AQLiteConstants.AQLiteNoMoreMessage)
    q_message = null;
    else
   throw new AQException (e.getErrorCode(), e.getMessage());
  }
  // handle the message
  ..
  ..
  ..
}
```

5

AQ Lite の伝播

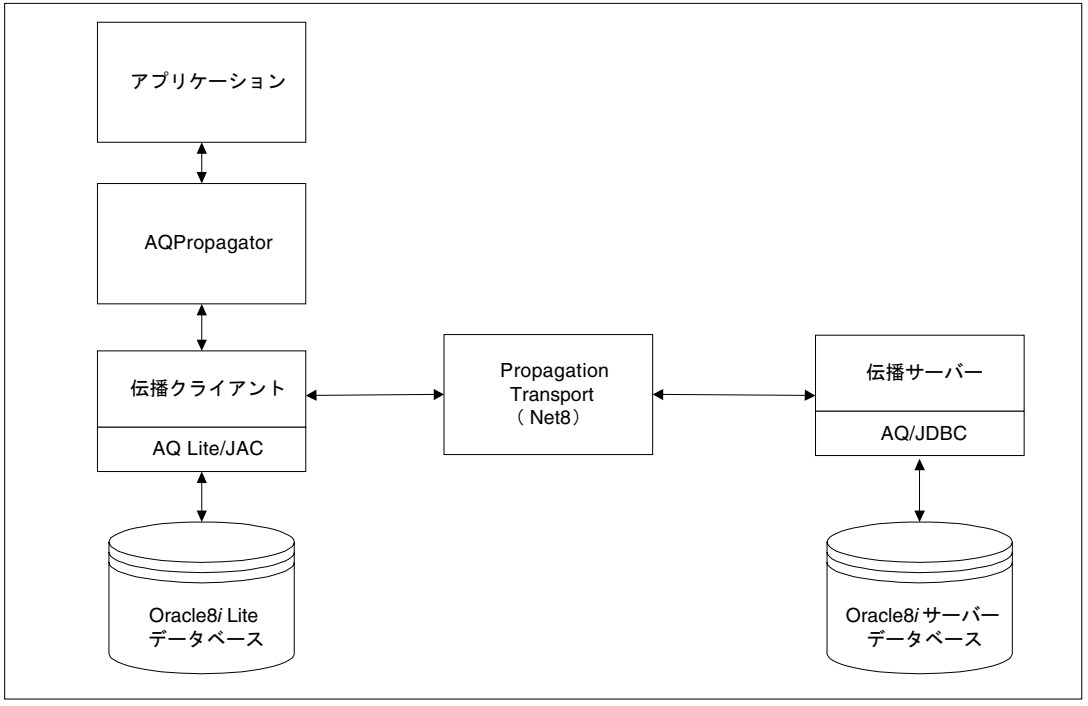
この章では、Oracle AQ Lite の伝播の使用方法を説明します。説明する内容は、次のとおりです。

伝播の概要

伝播はマスター / クライアント・キュー構成で使用され、Oracle Lite キューと Oracle8*i* サーバー・キューの間で AQ Lite メッセージを転送します。

伝播のアーキテクチャ

`AQLitePropagator` を起動することで、ユーザー・アプリケーションは Propagation Client を通して Propagation Transport と通信します。Propagation Transport はメッセージの伝播を管理し、伝播ロジックに関する情報をメンテナンスします。Propagation Transport は、その機能を AQ Lite Java API および Java アクセス・クラス（JAC）を使用して実行します。Propagation Server と通信して、クライアント・キューとマスター・キューとの間でメッセージを移動します。Propagation Server は、Oracle8*i* AQ API および JDBC を使用して Oracle8*i* サーバー・データベース上の伝播動作を管理します。



伝播キューのマッピング

Oracle Lite データベース内のクライアント・キューには、それぞれ対応するマスター・キューが 1 つだけ Oracle8*i* サーバー・データベース上にあります。クライアント・キューは、対応するマスター・キューと同じキュー名を持っています。伝播は、常に Oracle Lite クライアント・デバイスから開始します。クライアント・キューは、対応する Oracle8*i* サーバー上のマスター・キューとの間でのみメッセージを伝播します。

クライアント・キューとマスター・キューの関連は、ディプロイメント中に定義されます。クライアント / マスター・キュー・マッピング情報はカタログ表 AQLite$_Propcat に格納されます。詳細は、「AQLite$_Propcat」を参照してください。

AQLite$_Propcat

この表は、Oracle Lite データベースと Oracle8*i* データベースに常駐します。表には次のフィールドが含まれています。

フィールド	説明	型
Site_ID	Oracle8*i* サーバー・サイトの ID。	Varchar2(22) NOT NULL
Consumer	コンシューマ名。	Varchar2(30) NOT NULL
Q_Name	キューの名前。	Varchar2(30) NOT NULL
Q_Owner	キューの所有者。	Varchar2(30) NOT NULL
Last	最終のディプロイメントのタイムスタンプ。	Date
Enabled	ディプロイメントが使用可能かどうかを示すフラグ。Y: ディプロイメント可能、N: ディプロイメント不可。デフォルトは N です。	Char(1) NOT NULL
Queue_Type	AQ Lite のキューのタイプ。C: クローン・キュー、L: ローカル・キュー。デフォルトは L です。	Char(1)
User_Comment	ユーザー・コメントがある場合は、このフィールドを使用します。	Varchar2(50)

伝播の動作

Oracle8*i* サーバー上では、Oracle Lite クライアント宛てに指定されたすべてのメッセージが、伝播によってデキューされます。同時に、クライアントから送信されたあらゆるメッセージがサーバーにエンキュされます。

Oracle Lite クライアント上のアプリケーションは、`enqueue()` メソッドを使用して、マスター・キューに伝播するメッセージをエンキュします。伝播には次のものを使用します。

- `propEnqueue()` は、サーバーからクライアント・キューにメッセージをエンキュします。
- `propDequeue()` は、クライアント・キューからメッセージをデキューしてマスター・キューに移動します。
- キューからメッセージを削除する、削除デキュー・オプション。

`propEnqueue()` によってエンキュされたメッセージが、次の伝播でサーバーに戻されることはありません。Oracle Lite クライアント上のアプリケーションは、`dequeue()` メソッドを使用してサーバーから伝播されたメッセージをデキューします。これらのメソッドの使用方法の詳細は、第 4 章「AQ Lite の操作インタフェース」を参照してください。

伝播のトランザクションおよびリカバリ

クライアントが伝播を開始すると、システムは次のようなタスクを実行します。

1. 伝播はクライアント・キューからすべてのメッセージをデキューして、それをサーバー上のマスター・キューに移送します。
2. マスター・キューへのエンキュが成功すると、Oracle8*i* サーバーはトランザクションをコミットします。
3. サーバーがトランザクションをコミットした後で、AQ Lite がコミットします。サーバーがコミットしないと、トランザクションはロールバックされ、伝播は終了します。
4. 伝播は、サーバーから Oracle Lite クライアントを指定したすべてのメッセージをデキューして、それをクライアント・キューにエンキュします。
5. エンキュが成功すると、AQ Lite はトランザクションをコミットします。
6. AQ Lite がトランザクションをコミットした後で、サーバーがコミットします。AQ Lite がコミットしないと、トランザクションはロールバックされ、伝播は終了します。

伝播は 2 フェーズ・コミット・メカニズムというより 2 セットのコミットを使用します。

AQ Lite プロパゲータは、起動されるたびにトランザクションごとに一意の伝播 ID を生成します。宛先がコミットに失敗すると、クライアントおよびサーバー上の全トランザクションが取り消されロールバックされます。

伝播元が失敗してロールバックしても宛先がコミットすると、伝播ログ表にはトランザクションが記録されます。次の伝播の間に、重複したメッセージが伝播元から削除されます。

メッセージが再び宛先にエンキュされることはありません。伝播は 2 つのログ表を管理します。

- AQLite$_PropagationLog
- AQLite$_PropMsgLog

AQLite$_PropagationLog

AQLite$_PropagationLog には、次のフィールドがあります。

フィールド	説明	型
Prop_ID	伝播 ID。各伝播に一意で、0 より大きくしてください。	Number NOT NULL
Site_ID	Oracle8*i* Server サイトの ID。	Varchar2(22) NOT NULL
Q_Name	キューの名前。	Varchar2(30) NOT NULL
Q_Owner	キューの所有者。	Varchar2(30) NOT NULL
Log_ID	ログ ID。各キューに固有です。	Number NOT NULL

AQLite$_PropMsgLog

AQLite$_PropMsgLog には、次のフィールドがあります。

フィールド	説明	型
Log_ID	ログ ID。	Number NOT NULL
Msg_ID	メッセージ ID。各メッセージに一意です。	Raw(16) NOT NULL

伝播の実行

この項では、伝播の実行に必要なステップおよび AQLitePropagator クラスについて説明します。

マスター / クライアント・キューをマップする

伝播が機能するためには、マスター / クライアント・キュー構成を正しくディプロイメントする必要があります。各 Oracle Lite クライアントには、AQLiteDeploymentClient() によって一意のサイト ID が与えられる必要があります。

マスター / クライアント・キュー構成のキューのマッピングの詳細は、第 2 章「AQ Lite のディプロイメント」を参照してください。

プロパゲータを起動する

プロパゲータは、起動されるとクライアントおよびサーバー側の定義済キューをすべてチェックします。キュー名は、ディプロイメントの間にデータベース表 AQLite$_PropCat に格納されます。AQ Lite プロパゲータの起動手順は次のとおりです。

1. Oracle Lite クライアントへの接続を確立する。

```
// Get Oracle Lite POL Connection
POLConnection olite_dbconn =
   POLConnection.getConnection("C:¥orant¥oldb40¥polite.odb", 0);
```

2. Oracle8*i* データベースのログインおよび JDBC 情報を集める。

```
// Gather info for Oracle8i JDBC Connection
AQLitePropOraConn o8_dbconn =
new AQLitePropOraConn("oracle815database1",  // assign it a name
                      "scott",               // Oracle8i username
                      "tiger",               // Oracle8i password
                      "my_hostname:5521:my_sid"); // jdbc thin connect
                                                  // string
```

3. メソッド AQLitePropagation.start_queue_prop() をコールする。

```
// Initiate Propagator
AQLitePropagation Propagator = new AQLitePropagator();

// Start Propagation - both directions
Propagator.start_queue_prop(olite_dbconn,// POLConnection object
                            o8_dbconn,  // Oracle8i connection info
                            true,   // allow propagate from Oracle8i to
                                    //Oracle Lite
                            true,   // allow propagate from Oracle Lite to
```

```
                                        //Oracle8i
                                true);  // this argument has
                                        //no effect on
                                        // Propagator in current release
```

AQLitePropagator

この項では、伝播を実行するときに使用する AQLitePropagator クラスについて説明します。

コンストラクタ

用途

AQLitePropagator クラスを構成します。

構文

```
AQLitePropagator()
```

start_queue_prop

用途

このメソッドは伝播を起動します。

構文

```
public void start_queue_prop(oracle.pol.jac.POLConnection olitedb,
                             AQLitePropOraConn ora_svr_db,
                             boolean enqueue_enable,
                             boolean dequeue_enable,
                             boolean long_trans_enable)
```

パラメータ

パラメータ	説明
olitedb	Oracle Lite データベース接続。
ora_svr_db	Oracle8*i* データベース接続を開始するための必須フィールド。
enqueue_enable	Oracle8*i* から Oracle Lite への伝播を使用可能にします。

パラメータ	説明
dequeue_enable	Oracle Lite から Oracle8*i* への伝播を使用可能にします。
long_trans_enable	AQ Lite の現在のリリースでは使用されません。

Propagator Transport の実装

AQ Lite のこのリリースでは、伝播は Net8 のみをサポートします。ただし、開発者は他のプロトコル、たとえば Oracle Mobile Agent（OMA）または HTTP を使用して、インタフェース AQLitePropTransport を実装し、AQLitePropServer をコールすることで、専用のトランスポートを作成できます。

AQLitePropTransport インタフェース

この項では、AQ Lite Propagation Transport の共通インタフェースである AQLitePropTransport を説明します。

クラスの実装

次のクラスが AQLitePropTransport インタフェースの実装に使用されます。

- AQLitePropTransportNet8
- AQLitePropServer

openConnection

用途

このメソッドは、Oracle8*i* サーバーへの接続を確立し、サーバーの OliteSite ID を検証します。

構文

```
public int openConnection(AQLitePropOraConn ora_svr_db,
                          java.lang.String olite_id,
                          int outbound_prop_id,
                          java.lang.String propcat_table,
                          int dequeue_mode)
```

パラメータ

パラメータ	説明
ora_svr_db	Oracle8*i* データベースに接続するために必要な情報
olite_id	Oracle Lite サイトの ID
outbound_prop_id	アウトバウンド（クライアントからサーバー方向）伝播 ID
propcat_table	propcat（伝播カタログ）表の名前
dequeue_mode	AQ デキュー・オプションのデキュー・モード

戻り値

インバウンド（サーバーからクライアント方向）伝播 ID

propEnqueue

用途

AQ Lite プロパゲータは、このメソッドを使用してローカル・クライアント・デバイスへのメッセージをエンキュします。また、ローカル・アプリケーションでは、このメソッドを使用して別のローカル・アプリケーションが使用するメッセージをエンキュします。

構文

```
public void propEnqueue(AQMessage msg,
                        java.lang.String in_qOwner,
                        java.lang.String in_qName,
                        boolean new_queue)
```

propDequeue

用途

このメソッドは、Oracle8*i* サーバーからメッセージをデキューします。

構文

```
public AQLitePropQueue propDequeue()
```

propCommit

用途

このメソッドは、伝播トランザクションをコミットします。

構文

```
public void propCommit(int option)
```

propRollback

用途

このメソッドは、伝播トランザクションをロールバックします。

構文

```
public void propRollback(int option)
```

closeConnection

用途

このメソッドは、Oracle8*i* サーバーへの接続をクローズします。

構文

```
pubilic void closeConnection();
```

AQLitePropTransport インタフェースの実装

次の例では、Net8 を使用してトランスポート・インタフェースを実装します。

```
package oracle.lite.AQ;
import oracle.AQ.*;
/**
  This is the common interface for AQ Lite Propagation Transport.
*/
public class AQLitePropTransportNet8 implements AQLitePropTransport{

   AQLitePropServer prop_server;
```

```
   /** Constructor */
   public AQLitePropTransportNet8 () {
      AQLitePropServer prop_server = new AQLitePropServer();
      this.prop_server = prop_server;
   }

   /** Establish connection to Oracle8i server, verify Oracle Lite Site ID at
   Oracle8i side.*/
   public int openConnection(AQLitePropOraConn ora_svr_db,
                             String olite_site_id,
                             int outbound_prop_id,
                             String propcat_table,
                             int dequeue_mode) {
   return prop_server.openConnection(ora_svr_db, olite_site_id,
                                     outbound_prop_id,
                                     propcat_table,
                                     dequeue_mode);
   }

   /** Enqueue messages to Oracle8i server */
   public void propEnqueue(AQMessage msg,
                           String in_qOwner,
                           String in_qName,
                           boolean new_queue) {
      prop_server.propEnqueue(msg, in_qOwner, in_qName, new_queue);
   }
   /** Dequeue messages from Oracle8i server */
   public AQLitePropQueue propDequeue() {
      return prop_server.propDequeue();
   }

   /** Commit transaction */
   public void propCommit(int option) {
   prop_server.propCommit(option);
   }

   /** Rollback transaction */
   public void propRollback(int option) {
      prop_server.propRollback(option);
   }
}
```

トランスポートのプラグイン

トランスポートを実装した後、それを AQ Lite 伝播にプラグインする必要があります。そのために、**AQLite.jar** が常駐しているクラスパスにある **AQLite.ini** 構成ファイルを変更します。

次のように、**AQLite.ini** 構成ファイルを変更します。

1. 元の `transportName` 文をコメント化する。
2. クラスパスにトランスポート・クラス名を挿入して、`transportName` 文にユーザーのトランスポート・クラスを設定する。

例を次に示します。

```
#
# This is the AQLite configuration file
#
#transportName=oracle.lite.AQ.AQLitePropTransportNet8
transportName=oracle.lite.AQ.AQLiteMyPropTransport
```

A

AQ Lite Java API

この章では、Oracle AQ Lite の Java application programmer's interface（API）の概要と詳細を説明します。説明する内容は、次のとおりです。

- AQLiteDeploymentClient
- AQLiteDeploymentOliteSite
- AQLiteDeploymentQueueGroup
- AQLitePropagator
- AQLitePropTransport

概要

AQ Lite の Java API は、AQ Lite の管理機能および操作機能の両方をサポートします。

次の項では、共通インタフェースとクラスについて説明します。共通インタフェースは、Oracle Lite と Oracle8*i* で実装が異なります。

AQ Lite Java クラスは、次の場所にあります。

- *Oracle_Home*/Mobile/classes/AQCommon.jar
- *Oracle_Home*/Mobile/classes/AQOracle.jar
- *Oracle_Home*/Mobile/classes/AQLite.jar

API とクラス

この項では、AQ Lite に実装されているインタフェースとクラスをリストします。

AQ と AQ Lite の実装に共通したインタフェースおよびクラスを使用するようにしてください。これにより、アプリケーションは Oracle8*i* AQ と Oracle AQ Lite 間で移植可能になります。

AQ インタフェース

この表は、AQ Lite アプリケーションに使用される AQ インタフェースをまとめたものです。

注意： `AQQueue` インタフェースは `AQQueueAdmin` を継承します。したがって、すべての管理および操作機能を `AQQueue` 経由で使用できます。

インタフェースの概要	説明
`AQSession`	キューイング・システムに対するセッションをオープンします
`AQQueueTable`	AQ のキュー・テーブル・インタフェース
`AQQueueAdmin`	AQ のキュー管理インタフェース
`AQQueue`	AQ のキュー操作インタフェース
`AQMessage`	AQ のメッセージ
`AQRaw Payload`	AQ の RAW 型ペイロード
`AQDriver`	様々な AQ ドライバ用のインタフェース

AQ Common Classes

この表は、AQ Lite アプリケーションを実装するために使用される AQ 共通クラスをまとめたものです。

クラスの概要	説明
AQConstants	AQ 操作で使用される定数
AQAgent	AQ のエージェント
AQDriverManager	様々な AQ ドライバ用のドライバ・マネージャ
AQEnqueueOption	AQ のエンキュ・オプション
AQDequeueOption	AQ のデキュー・オプション
AQMessageProperty	AQ のメッセージ・プロパティ
AQQueueProperty	AQ のキューのプロパティ
AQQueueTableProperty	AQ のキュー・テーブルのプロパティ

Oracle AQ Lite クラス

この表は、AQ Lite アプリケーションを実装するために使用される AQ Lite のクラスをまとめたものです。

クラスの概要	説明
AQLiteSession	Oracle Lite での AQSession の実装
AQLiteMessage	Oracle Lite での AQMessage の実装
AQLiteDriver	Oracle Lite での AQDriver の実装
AQLiteQueue	Oracle Lite での AQQueue の実装
AQLiteQueueTable	Oracle Lite での AQQueueTable の実装
AQLiteRawPayload	Oracle Lite での AQRawPayload の実装

AQLiteDeployment クラス

この表は、AQ Lite アプリケーションのディプロイメントに使用されるクラスをまとめたものです。

クラスの概要	説明
AQLiteDeploymentAdmin	ディプロイメントの管理
AQLiteDeploymentClient	クライアント端末からのディプロイメント
AQLiteDeploymentOliteSite	OliteSite オブジェクトのディプロイメント
AQLiteDeploymentQueueGroup	QueueGroup オブジェクトのディプロイメント

AQLitePropagation のクラスとインタフェース

この表は、AQ Lite の伝播を実装するために使用されるクラスをまとめたものです。

クラスとインタフェースの概要	説明
AQLitePropagator	Oracle Lite と Oracle8*i* サーバー間のメッセージ伝播
AQLitePropTransport	AQ Lite Propagation Transport の共通インタフェース

Oracle8*i* の AQ クラス

クラスの概要	説明
AQOracleDriver	Oracle サーバーでの AQDriver の実装

AQDriverManager

Java AQ API の様々な実装は、AQDriverManager を使用して管理されます。Oracle Lite と Oracle8*i* は、どちらも、AQDriverManager に登録されている AQDriver を使用します。このドライバ・マネージャを使用して、メッセージ・タスクの実行に使用する AQSession を作成します。

AQDriverManager.createAQSession() メソッドが起動されると、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。

Oracle Lite AQDriver では、AQSession を作成するためのパラメータとして、有効な Oracle Lite JAC 接続が渡されるものと想定します。

また、JAC 接続を確立し AQLiteDriver を起動する方法でも、AQSession を作成できます。AQLiteDriver を使用した AQSession の作成の詳細は、第 4 章「AQ Lite の操作インタフェース」を参照してください。

getDrivers

用途

このメソッドは、ドライバ・マネージャに登録されているドライバのリストを返します。登録済ドライバの名前を含む文字列のベクトルを返します。

構文

```
public static java.util.Vector getDrivers()
```

createAQSession

用途

このメソッドは、AQSession を作成します。

構文

```
public static AQSession createAQSession (java.lang.Object conn)
  throws AQException
```

パラメータ

パラメータ	説明
conn	ユーザーが AQOracleDriver を使用している場合は、渡されるオブジェクトは有効な JDBC 接続にしてください。

使用上の注意

AQ の Java オブジェクトは、現時点ではスレッド・セーフではありません。AQSession、AQQueueTable、AQQueue およびその他の AQ オブジェクトに対するメソッドを、複数の異なるスレッドから同時にコールしないでください。これらのオブジェクトをスレッド間で渡すことはできますが、これらの AQ オブジェクトに対するメソッドが同時に起動されないようにプログラムで保証する必要があります。

マルチスレッド・プログラムの場合は、(同一または別のデータベース接続を使用して) 各スレッド内に個別の AQSession を作成し、AQSession の getQueueTable メソッドと getQueue メソッドを使用して、新規のキュー・テーブルとキューのハンドルを取得する必要があります。

registerDriver

用途

このメソッドは、AQ ドライバの様々な実装によって使用され、AQ ドライバ自体をドライバ・マネージャに登録します (このメソッドは、クライアント・プログラムからは直接コールされません)。

構文

```
public static void registerDriver(AQDriver aq_driver)
```

AQLiteDriver

このクラスは、Oracle AQ Lite ドライバを Oracle Lite データベースに提供します。各ドライバには、システム固有の Oracle8*i* Lite データベースへの接続オブジェクトが供給されている必要があります。AQ Lite は、現在 JAC 接続を使用します。詳細は、AQDriver インタフェースを参照してください。AQDriver 型のオブジェクトを取得するクラスを構成するだけで済みます。

注意： Oracle8*i* AQ にアクセスするには、AQOracleDriver を使用する必要があります。

AQSession を作成するには、まずデータベース接続をオープンする必要があります。その後、アプリケーションで必要な AQLiteDriver をロードする必要があります。ドライバをロードする必要があるのは、(最初の createAQSession コールの前の) 1 回のみです。ドライバを複数回ロードしても何の効果もありません。

例

```
POLConnection olitedb_conn;    /* POL connection */
AQLiteDriver aq_driver         /* AQ Lite driver */
AQSession aq_sess              /* AQ Session */

olitedb_conn = POLConnection.getConnection("c:¥orant¥oldb40¥polite.odb", 0);
```

```
aq_driver = new AQLiteDriver();
aq_sess = aq_driver.createAQSession(olitedb_conn);
```

コンストラクタ

用途

AQ Lite Driver クラスを構成します。

構文

```
public AQLiteDriver() throws AQException
```

createAQSession

用途

このメソッドは、AQSession を作成します。

構文

```
public createAQSession(java.lang.Object obj) throws AQException
```

パラメータ

パラメータ	説明
obj	Oracle Lite データベースへの JAC 接続オブジェクト

acceptsObject

用途

このメソッドは接続オブジェクトをアクセプトします。

構文

```
public acceptsObject(java.lang.Object conn) throws AQException
```

getMajorVersion

用途

このメソッドは、バージョン番号を取得します。

構文

```
public string getMajorVersion() throws AQException
```

getMinorVersion

用途

このメソッドは、リリース番号を取得します。

構文

```
public string getMinorVersion() throws AQException
```

AQSession

AQSession インタフェースはキューイング・システムへのセッションをオープンするために使用されます。

createQueueTable

用途

このメソッドは、AQQueueTableProperty オブジェクト内に渡されるプロパティに従って、特定ユーザーのスキーマ内に新規キュー・テーブルを作成します。

構文

```
public AQQueueTable createQueueTable(java.lang.String owner,
                                      java.lang.String name,
                                      AQQueueTableProperty property)
                                      throws AQException
```

パラメータ

パラメータ	説明
owner	キュー・テーブルを作成するスキーマ（ユーザー）
q_name	キュー・テーブルの名前
property	キュー・テーブルのプロパティ

戻り値

AQQueueTable オブジェクト

getQueueTable

用途

このメソッドは、既存のキュー・テーブルに対するハンドルを取得します。

構文

```
public AQQueueTable getQueueTable(java.lang.String owner,
                                  java.lang.String name)
```

パラメータ

パラメータ	説明
owner	キュー・テーブルが存在するスキーマ（ユーザー）
name	キュー・テーブルの名前

戻り値

AQQueueTable オブジェクト

createQueue

用途

このメソッドは、指定されたキュー・プロパティを使用してキュー・テーブルの中にキューを作成します。キュー・テーブルの作成に使用されるスキーマ名を使用します。

構文

```
public AQQueue createQueue(AQQueueTable q_table,
                           java.lang.String q_name,
                           AQQueueProperty q_property) throws AQException
```

パラメータ

パラメータ	説明
q_table	キューを作成するキュー・テーブル
name	作成されるキューの名前
q_property	キューのプロパティ

戻り値

AQQueue オブジェクト

getQueue

用途

このメソッドは、既存のキューに対するハンドルを取得します。

構文

```
public AQQueue getQueue(java.lang.String owner,
                        java.lang.String name)
```

パラメータ

パラメータ	説明
owner	キュー・テーブルが存在するスキーマ（ユーザー）
name	キューの名前

戻り値

AQQueue オブジェクト

使用上の注意

AQ Java API は、現時点では、RAW 型ペイロードのキューのみサポートしています。オブジェクト型ペイロードを使用するキューを含むキュー・テーブルにアクセスすると、AQException とメッセージ「ペイロードの型がサポートされていません」を受け取ります。

AQLiteConstants

このクラスには、AQ Lite Java API で使用される定数が含まれています。

```
public static intAQLClassExists
public static java.lang.StringAQLiteDriverName
public static int AQLiteMajorVersion
public static int AQLiteMinorVersion
public static intAQLiteNoMoreMessage
public static java.lang.StringAQLitePOLConnection
public static intAQLObjectExists
```

AQAgent

このオブジェクトは、メッセージのプロデューサまたはコンシューマを指定します。

コンストラクタ

用途

コンストラクタには 2 種類の実装がありますが、いずれの場合も、指定されたパラメータを使用して新規の AQAgent が割り当てられます。

構文 1

```
public AQAgent(java.lang.String name,
               java.lang.String address,
               double protocol)
```

構文 2

```
public AQAgent(java.lang.String name,
               java.lang.String address)
```

パラメータ

パラメータ	説明
name	エージェント名。
address	エージェント・アドレス。
protocol	エージェントのプロトコル（最初のコンストラクタにのみ必要）。デフォルトは 0 です。

getName

用途

このメソッドは、エージェント名を取得します。

構文

```
public java.lang.String getName() throws AQException
```

setName

用途

このメソッドは、エージェント名を設定します。

構文

```
public void setName(java.lang.String name) throws AQException
```

パラメータ

パラメータ	説明
name	エージェント名

getAddress

用途

このメソッドは、エージェントのアドレスを取得します。

構文

```
public java.lang.String getAddress() throws AQException
```

setAddress

用途

このメソッドは、エージェントのアドレスを設定します。

構文

```
public void setAddress(java.lang.String address) throws AQException
```

パラメータ

パラメータ	説明
address	特定の宛先にあるキュー

getProtocol

用途

このメソッドは、エージェントのプロトコルを取得します。

構文

```
public int getProtocol() throws AQException
```

setProtocol

用途

このメソッドは、エージェントのプロトコルを設定します。

構文

```
public void setProtocol(int protocol) throws AQException
```

パラメータ

パラメータ	説明
protocol	エージェント・プロトコル

AQQueueTableProperty

このクラスは、キュー・テーブルのプロパティを表します。

メッセージのグループ化のための定数

```
public static final int NONE
public static final int TRANSACTIONAL
```

コンストラクタ

用途

このメソッドは、デフォルトのプロパティ値と指定されたペイロード型を使用して、AQQueueTableProperty オブジェクトを作成します。

構文

```
public AQQueueTableProperty(java.lang.String p_type)
```

パラメータ

パラメータ	説明
p_type	ペイロードの型。現時点では、RAW 型のペイロードのみサポートされています。

getPayloadType

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setPayloadType

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setStorageClause

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getSortOrder

用途

このメソッドは、キュー・テーブルで使用されるソート順序を取得します。

構文

```
public java.lang.String getSortOrder() throws AQException
```

戻り値

キュー・テーブルで使用されるソート順序

setSortOrder

用途

このメソッドは、キュー・テーブルで使用されるソート順序を設定します。

構文

```
public void setSortOrder(java.lang.String s_order) throws AQException
```

パラメータ

パラメータ	説明
s_order	昇順で sort_key として使用する列を指定します。この文字列のフォーマットは、<*sort_column1, sort_column2*> です。 優先順位によるソート：AQLiteConstants.AQLiteSortOrderPriority エンキュ時刻によるソート： AQLiteConstants.AQLiteSortOrderEnqTime 最初にエンキュ時刻、次に優先順位によるソート： AQLiteConstants.AQLiteSortOrderEP 最初に優先順位、次にエンキュ時刻によるソート： AQLiteConstants.AQLiteSortOrderPE

isMulticonsumerEnabled

用途

このメソッドは、表内に作成されたキューが、1 つのメッセージにつき複数のコンシューマを含むことができるかどうかを問い合せます。

構文

```
public boolean isMulticonsumerEnabled() throws AQException
```

戻り値

表に作成されたキューが、1 つのメッセージにつき複数のコンシューマを含める場合は、TRUE が返されます。

表に作成されたキューが、1 つのメッセージにつきコンシューマを 1 つしか含めない場合は、FALSE が返されます。

setMultiConsumer

用途

このメソッドは、表に作成されたキューが、1 つのメッセージにつき複数のコンシューマを含むことができるどうかを決定します。

構文

```
public void setMultiConsumer(boolean enable) throws AQException
```

パラメータ

パラメータ	説明
enable	FALSE を指定すると、表に作成されたキューは、1 つのメッセージにつきコンシューマを 1 つしか含みません。 TRUE を指定すると、表に作成されたキューが、1 つのメッセージにつき複数のコンシューマを含めます。

getMessageGrouping

用途

このメソッドは、キュー・テーブル内のキューのメッセージ・グループ化の方法を取得します。

構文

```
public int getMessageGrouping() throws AQException
```

戻り値

各メッセージが個別に扱われる場合は、NONE が返されます。

1 つのトランザクションの一部としてエンキュされたすべてのメッセージが同一グループに属するものとみなされ、デキューの際に 1 つの関連メッセージ・グループとしてデキューできる場合は、TRANSACTIONAL が返されます。

setMessageGrouping

用途

このメソッドは、キュー・テーブル内に作成されたキューのメッセージ・グループ化の方法を設定します。

構文

```
public void setMessageGrouping(int m_grouping) throws AQException
```

パラメータ

パラメータ	説明
m_grouping	NONE または TRANSACTIONAL

getComment

用途

このメソッドは、キュー・テーブルのコメントを取得します。

構文

```
public java.lang.String getComment() throws AQException
```

setComment

用途

このメソッドは、コメントを設定します。

構文

```
public void setComment(java.lang.String qt_comment) throws AQException
```

パラメータ

パラメータ	説明
qt_comment	キュー・テーブルのコメント

getCompatible

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setCompatible

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getPrimaryInstance

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setPrimaryInstance

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getSecondaryInstance

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setSecondaryInstance

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

AQQueueProperty

このクラスは、キューのプロパティを表します。

定数

```
public static final int NORMAL_QUEUE
public static final int EXCEPTION_QUEUE
public static final int INFINITE   // infinite retention
```

コンストラクタ

用途

このメソッドは、デフォルトのプロパティ値を使用して新規の AQQueueProperty オブジェクトを作成します。

構文

```
public AQQueueProperty()
```

getQueueType

用途

このメソッドは、キューのタイプを取得します。

構文

```
public int getQueueType() throws AQException
```

戻り値

NORMAL_QUEUE または EXCEPTION_QUEUE

setQueueType

用途

このメソッドは、キューのタイプを設定するために使用します。

構文

```
public void setQueueType(int q_type) throws AQException
```

パラメータ

パラメータ	説明
q_type	NORMAL_QUEUE または EXCEPTION_QUEUE

getMaxRetries

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setMaxRetries

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setRetryInterval

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getRetryInterval

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getRetentionTime

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setRetentionTime

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getComment

用途

このメソッドは、キューのコメントを取得します。

構文

```
public java.lang.String getComment() throws AQException
```

setComment

用途

このメソッドは、キューのコメントを設定します。

構文

```
public void setComment(java.lang.String qt_comment) throws AQException
```

パラメータ

パラメータ	説明
qt_comment	キューのコメント

AQQueueTable

AQQueueTable インタフェースには、キュー・テーブルの管理のためのメソッドが含まれています。

getOwner

用途

このメソッドは、キュー・テーブルの所有者を取得します。

構文

```
public java.lang.String getOwner() throws AQException
```

getName

用途

このメソッドは、キュー・テーブルの名前を取得します。

構文

```
public java.lang.String getName() throws AQException
```

getProperty

用途

このメソッドは、キュー・テーブルのプロパティを取得します。

構文

```
public AQQueueTableProperty getProperty() throws AQException
```

戻り値

AQQueueTableProperty オブジェクト

drop

用途

このメソッドは現在のキュー・テーブルを削除します。

構文

```
public void drop(boolean force) throws AQException
```

パラメータ

パラメータ	説明
force	FALSE を指定すると、キュー・テーブル内にキューが存在する場合、操作は失敗します。これがデフォルトです。 TRUE を指定すると、キュー・テーブル内のすべてのキューが自動的に停止および削除されます。

alter

このメソッドは、AQ Lite の現在のリリースではサポートされていません。

createQueue

用途

このメソッドは、キュー・テーブル内にキューを作成します。

構文

```
public AQQueue createQueue(java.lang.String queue_name,
                           AQQueueProperty q_property) throws AQException
```

パラメータ

パラメータ	説明
queue_name	作成されるキューの名前
q_property	キューのプロパティ

戻り値

AQQueue オブジェクト

dropQueue

用途

このメソッドは、キュー・テーブル内のキューを削除します。

構文

```
public void dropQueue(java.lang.String queue_name) throws AQException
```

パラメータ

パラメータ	説明
queue_name	削除されるキューの名前

AQQueueAdmin

start

用途

このメソッドによって、キューに対するエンキュおよびデキューを可能にします。

構文

```
public void start(boolean enqueue,
                  boolean dequeue) throws AQException
```

パラメータ

パラメータ	説明
enqueue	このキューに対するエンキュを可能にする場合は、TRUE を設定します。 現在の設定を変更しない場合は、FALSE を設定します。
dequeue	このキューに対するデキューを可能にする場合は、TRUE を設定します。 現在の設定を変更しない場合は、FALSE を設定します。

startEnqueue

用途

このメソッドによって、キューに対するエンキュを可能にします。これは、start(TRUE, FALSE) と等価です。

構文

```
public void startEnqueue() throws AQException
```

startDequeue

用途

このメソッドによって、キューに対するデキューを可能にします。これは、start(FALSE, TRUE) と等価です。

構文

```
public void startDequeue() throws AQException
```

stop

用途

このメソッドによって、キューに対するエンキュとデキューを禁止にします。

構文

```
public void stop(boolean enqueue,
                 boolean dequeue,
                 boolean wait) throws AQException
```

パラメータ

パラメータ	説明
enqueue	このキューに対するエンキュを禁止にする場合は、TRUE を設定します。 現在の設定を変更しない場合は、FALSE を設定します。
dequeue	このキューに対するデキューを禁止にする場合は、TRUE を設定します。 現在の設定を変更しない場合は、FALSE を設定します。
wait	処理中のトランザクションが完了するまで待機する場合は TRUE を設定します。 正常またはエラー状態で即時に復帰させる場合は FALSE を設定します。

stopEnqueue

用途

このメソッドによって、キューに対するエンキュを禁止にします。これは、stop(TRUE, FALSE,wait) と等価です。

構文

```
public void stopEnqueue(boolean wait) throws AQException
```

パラメータ

パラメータ	説明
wait	処理中のトランザクションが完了するまで待機する場合は TRUE を設定します。 正常またはエラー状態で即時に復帰させる場合は FALSE を設定します。

stopDequeue

用途

このメソッドによって、キューに対するデキューを禁止にします。これは、stop(FALSE, TRUE, wait) と等価です。

構文

```
public void stopDequeue(boolean wait) throws AQException
```

パラメータ

パラメータ	説明
wait	処理中のトランザクションが完了するまで待機する場合は TRUE を設定します。 正常またはエラー状態で即時に復帰させる場合は FALSE を設定します。

drop

用途

このメソッドは、キューを削除します。

構文

```
public void drop() throws AQException
```

alterQueue

このメソッドは、AQ Lite の現在のリリースではサポートされていません。

addSubscriber

用途

このメソッドは、キューにサブスクライバを追加します。

構文

```
public void addSubscriber(AQAgent subscriber,
                          java.lang.String rule) throws AQException
```

パラメータ

パラメータ	説明
subscriber	サブスクリプションを定義する対象の AQAgent。
rule	AQ Lite はルールベースのサブスクライバをサポートしていません。この値は常に NULL です。

removeSubscriber

用途

このメソッドは、キューからサブスクライバを削除します。

構文

```
public void removeSubscriber(AQAgent subscriber) throws AQException
```

パラメータ

パラメータ	説明
subscriber	削除対象の AQAgent

alterSubscriber

このメソッドは、AQ Lite の現在のリリースではサポートされていません。

grantQueuePrivilege

このメソッドは、AQ Lite の現在のリリースではサポートされていません。

revokeQueuePrivilege

このメソッドは、AQ Lite の現在のリリースではサポートされていません。

AQQueue

このインタフェースは、キューの操作インタフェースをサポートします。AQQueue は AQQueueAdmin を継承しているため、このインタフェースで管理機能も使用できます。

getOwner

用途

このメソッドは、キューの所有者を取得します。

構文

```
public java.lang.String getOwner() throws AQException
```

getName

用途

このメソッドは、キューの名前を取得します。

構文

```
public java.lang.String getName() throws AQException
```

getQueueTableName

用途

このメソッドは、キューが存在するキュー・テーブルの名前を取得します。

構文

```
public java.lang.String getQueueTableName() throws AQException
```

getProperty

用途

このメソッドは、キューのプロパティを取得します。

構文

```
public AQQueueProperty getProperty() throws AQException
```

戻り値

AQQueueProperty オブジェクト

createMessage

用途

このメソッドは、エンキュされるデータを格納できる新規 AQMessage オブジェクトを作成します。

構文

```
public AQMessage createMessage() throws AQException
```

戻り値

AQMessage オブジェクト

enqueue

用途

このメソッドは、キューにメッセージをエンキュします。

構文

```
public byte[] enqueue(AQEnqueueOption enq_option,
                      AQMessage message) throws AQException
```

パラメータ

パラメータ	説明
enq_option	AQEnqueOption オブジェクト
message	エンキュされる AQMessage オブジェクト

戻り値

エンキュされたメッセージのメッセージ ID。このコールの完了後、AQMessage オブジェクトの messageId フィールドにも格納されます。

propEnqueue

用途

このメソッドは、プロパゲータのためのメッセージをエンキュします。propEnqueue の使用方法の詳細は、第 4 章「AQ Lite の操作インタフェース」の「enqueue() と propEequeue()」を参照してください。

構文

```
public byte[] propEnqueue(AQEnqueueOption enq_option,
                      AQMessage message) throws AQException
```

dequeue

用途

このメソッドは、キューからメッセージをデキューします。

構文

```
public AQMessage dequeue(AQDequeueOption deq_option) throws AQException
```

パラメータ

パラメータ	説明
deq_option	AQDequeueOption オブジェクト

戻り値

デキューされたメッセージである AQMessage

propDequeue

用途

このメソッドは、プロパゲータ用のキューからメッセージをデキューします。propDequeue の使用方法の詳細は、第 4 章「AQ Lite の操作インタフェース」の「dequeue() と propDequeue()」を参照してください。

構文 1

```
public AQMessage propDequeue(AQDequeueOption deq_option)
                      throws AQException
```

構文 2

```
public AQMessage propDequeue(AQDequeueOption deq_option,
                      java.lang.Class payload_class)
                      throws AQException
```

getSubscribers

用途

このメソッドは、キューのサブスクライバ・リストを取得します。

構文

```
public AQAgent[] getSubscribers() throws AQException
```

戻り値

AQAgents の配列

AQEnqueueOption

このクラスは、エンキュ操作に使用できるオプションを指定するために使用します。

定数

```
public static final int DEVIATION_NONE
public static final int DEVIATION_BEFORE
public static final int DEVIATION_TOP
public static final int VISIBILITY_ONCOMMIT
public static final int VISIBILITY_IMMEDIATE
```

コンストラクタ

用途

使用可能なコンストラクタは 2 種類あります。1 番目のコンストラクタは、指定されたオプションでオブジェクトを作成し、2 番目のコンストラクタは、デフォルトのオプションでオブジェクトを作成します。

構文 1

```
public AQEnqueueOption(int visibility,
                       byte[] relative_msgid,
                       int sequence_deviation)
```

構文 2

```
public AQEnqueueOption()
```

パラメータ

パラメータ	説明
relative_msgid	このパラメータは、AQLite ではサポートされていません。
sequence_deviation	このパラメータは、AQLite ではサポートされていません。 デフォルトは DEVIATION_NONE です。

getVisibility

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setVisibility

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getRelMessageId

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getSequenceDeviation

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setSequenceDeviation

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

AQDequeueOption

このクラスは、デキュー操作に使用できるオプションを指定するために使用します。

定数

```
public static final int NAVIGATION_FIRST_MESSAGE
public static final int NAVIGATION_NEXT_TRANSACTION
public static final int NAVIGATION_NEXT_MESSAGE
public static final int DEQUEUE_BROWSE
public static final int DEQUEUE_LOCKED
public static final int DEQUEUE_REMOVE
public static final int DEQUEUE_REMOVE_NODATA
```

```
public static final int WAIT_FOREVER
public static final int WAIT_NONE
public static final int VISIBILITY_ONCOMMIT
public static final int VISIBILITY_IMMEDIATE
```

コンストラクタ

用途

このメソッドは、デフォルトのオプションでオブジェクトを作成します。

構文

```
public AQDequeueOption()
```

getConsumerName

用途

このメソッドは、コンシューマ名を取得します。

構文

```
public java.lang.String getConsumerName() throws AQException
```

setConsumerName

用途

このメソッドは、コンシューマ名を設定します。

構文

```
public void setConsumerName(java.lang.String consumer_name) throws AQException
```

パラメータ

パラメータ	説明
consumer_name	エージェント名

getDequeueMode

用途

このメソッドは、デキュー・モードを取得します。

構文

```
public int getDequeueMode() throws AQException
```

戻り値

DEQUEUE_BROWSE、DEQUEUE_LOCKED、DEQUEUE_REMOVE または DEQUEUE_REMOVE_NODATA

setDequeueMode

用途

このメソッドは、デキュー・モードを設定します。

構文

```
public void setDequeueMode(int dequeue_mode) throws AQException
```

パラメータ

パラメータ	説明
dequeue_mode	DEQUEUE_BROWSE、DEQUEUE_LOCKED、DEQUEUE_REMOVE または DEQUEUE_REMOVE_NODATA

getNavigationMode

用途

このメソッドは、ナビゲーション・モードを取得します。

構文

```
public int getNavigationMode() throws AQException
```

戻り値

NAVIGATION_FIRST_MESSAGE、NAVIGATION_NEXT_MESSAGE または NAVIGATION_NEXT_TRANSACTION

setNavigationMode

用途

このメソッドは、ナビゲーション・モードを設定します。

構文

```
public void setNavigationMode(int navigation) throws AQException
```

パラメータ

パラメータ	説明
navigation	NAVIGATION_FIRST_MESSAGE、NAVIGATION_NEXT_MESSAGE または NAVIGATION_NEXT_TRANSACTION

getVisibility

用途

このメソッドは、可視性を取得します。

構文

```
public int getVisibility() throws AQException
```

戻り値

VISIBILITY_IMMEDIATE または VISIBILITY_ONCOMMIT

setVisibility

用途

このメソッドは、可視性を設定します。

構文

```
public void setVisibility(int visibility) throws AQException
```

パラメータ

パラメータ	説明
visibility	VISIBILITY_IMMEDIATE または VISIBILITY_ONCOMMIT

getWaitTime

用途

このメソッドは、待機時間を取得します。

構文

```
public int getWaitTime() throws AQException
```

戻り値

WAIT_NONE または 0 秒

setWaitTime

用途

このメソッドは、待機時間を設定します。

構文

```
public void setWaitTime(int wait_time) throws AQException
```

パラメータ

パラメータ	説明
wait_time	待機時間。常に WAIT_NONE または 0 秒。

getMessageId

用途

このメソッドは、メッセージ ID を取得します。

構文

```
public byte[] getMessageId() throws AQException
```

setMessageId

用途

このメソッドは、メッセージ ID を設定します。

構文

```
public void setMessageId(byte[] message_id) throws AQException
```

パラメータ

パラメータ	説明
message_id	メッセージ ID

getCorrelation

用途

このメソッドは、相関 ID を取得します。

構文

```
public java.lang.String getCorrelation() throws AQException
```

setCorrelation

用途

このメソッドは、相関 ID を設定します。

構文

```
public void setCorrelation(java.lang.String correlation) throws AQException
```

パラメータ

パラメータ	説明
correlation	ユーザー指定情報

AQMessage

このインタフェースには、RAW 型ペイロードまたはオブジェクト型ペイロードの AQ メッセージ用のメソッドが含まれています。

getMessageId

用途

このメソッドは、メッセージ ID を取得します。

構文

```
public byte[] getMessageId() throws AQException
```

getRawPayload

用途

このメソッドは、RAW 型ペイロードを取得します。

構文

```
public AQRawPayload getRawPayload() throws AQException
```

戻り値

AQRawPayload オブジェクト

setRawPayload

用途

このメソッドは、RAW 型ペイロードを設定します。オブジェクト型のキューから作成されたメッセージに対してこのメソッドをコールすると、AQException が発生します。

構文

```
public void setRawPayload(AQRawPayload message_payload) throws AQException
```

パラメータ

パラメータ	説明
message_payload	RAW 型ユーザー・データを含む AQRawPayload オブジェクト

getMessageProperty

用途

このメソッドは、メッセージ・プロパティを取得します。

構文

```
public AQMessageProperty getMessageProperty() throws AQException
```

戻り値

AQMessageProperty オブジェクト

setMessageProperty

用途

このメソッドは、メッセージ・プロパティを設定します。

構文

```
public void setMessageProperty(AQMessageProperty property) throws AQException
```

パラメータ

パラメータ	説明
property	AQMessageProperty オブジェクト

AQMessageProperty

AQMessageProperty クラスには、AQ で個々のメッセージを管理するために使用される情報が含まれています。プロパティはエンキュ時に設定され、デキュー時にその値が返されます。

定数

```
public static final int DELAY_NONE
public static final int EXPIRATION_NEVER
public static final int STATE_READY
public static final int STATE_WAITING
public static final int STATE_PROCESSED
public static final int STATE_EXPIRED
```

コンストラクタ

用途

このメソッドは、デフォルトのプロパティ値を使用して AQMessageProperty オブジェクトを作成します。

構文

```
public AQMessageProperty()
```

getPriority

用途

このメソッドは、メッセージの優先順位を取得します。

構文

```
public int getPriority() throws AQException
```

setPriority

用途

このメソッドは、メッセージの優先順位を設定します。

構文

```
public void setPriority(int priority) throws AQException
```

パラメータ

パラメータ	説明
priority	メッセージの優先順位。負数も含め、任意の数値を指定できます。数値が小さいほど、優先順位が高いことを表します。

getDelay

用途

このメソッドは、遅延値を取得します。

構文

```
public int getDelay() throws AQException
```

setDelay

用途

このメソッドは、遅延値を設定します。

構文

```
public void setDelay(int delay) throws AQException
```

パラメータ

パラメータ	説明
delay	遅延値は、メッセージがデキュー可能になるまでの秒数を表します。NO_DELAY の場合、メッセージはただちにデキューできます。

getExpiration

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

setExpiration

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

getCorrelation

用途

このメソッドは、相関関係を取得します。

構文

```
public java.lang.String getCorrelation() throws AQException
```

setCorrelation

用途

このメソッドは相関関係を設定します。

構文

```
public void setCorrelation(java.lang.String correlation) throws AQException
```

パラメータ

パラメータ	説明
correlation	ユーザー指定情報

getAttempts

用途

このメソッドは、試行回数を取得します。

構文

```
public int getAttempts() throws AQException
```

getRecipientList

用途

このメソッドは、受信者リストを取得します。

構文

```
public java.util.Vector getRecipientList() throws AQException
```

戻り値

AQAgent オブジェクトのベクトル。このパラメータは、デキュー時にはコンシューマに対して返されません。

setRecipientList

用途

このメソッドは、受信者リストを設定します。

構文

```
public void setRecipientList(java.util.Vector r_list) throws AQException
```

パラメータ

パラメータ	説明
r_list	AQAgents のベクトル。デフォルトの受信者は、キューのサブスクライバです。

getOrigMessageId

用途

このメソッドは、元のメッセージ ID を取得します。

構文

```
public byte[] getOrigMessageId() throws AQException
```

getSender

用途

このメソッドは、メッセージの送信者を取得します。

構文

```
public AQAgent getSender() throws AQException
```

setSender

用途

このメソッドは、メッセージの送信者を設定します。

構文

```
public void setSender(AQAgent sender) throws AQException
```

パラメータ

パラメータ	説明
sender	メッセージをエンキュする AQAgent オブジェクト

getEnqueueTime

用途

このメソッドは、エンキュ時刻を取得します。

構文

```
public java.util.Date getEnqueueTime() throws AQException
```

getState

このメソッドは、Oracle AQ Lite の現在のリリースではサポートされていません。

AQRawPayload

このオブジェクトは、AQMessage オブジェクトに含まれる RAW 型のユーザー・データを表します。

getStream

用途

このメソッドは、RAW 型ペイロード・データの一部を、指定されたバイト配列に読み取ります。

構文

```
public int getStream(byte[] value, int len) throws AQException
```

パラメータ

パラメータ	説明
value	RAW 型データを保持するバイト配列
len	読み取られるバイト数

戻り値

読み取られたバイト数

getBytes

用途

このメソッドは、RAW 型ペイロード・データ全体をバイト配列として取り出します。

構文

```
public byte[] getBytes() throws AQException
```

戻り値

byte[]。バイト配列として表された RAW 型ペイロードです。

setStream

用途

このメソッドは、RAW 型ペイロードの値を設定します。

構文

```
public void setStream(byte[] value,
                      int len) throws AQException
```

パラメータ

パラメータ	説明
value	RAW 型ペイロードを含むバイト配列
len	RAW 型ストリームに書き込むバイト数

AQException

この例外は、ユーザーが AQ Java API を使用中にエラーが検出された場合に発生します。

```
public class AQException extends java.lang.RuntimeException
```

このインタフェースは、Java の例外処理によってサポートされているすべてのメソッドの他、いくつかの追加メソッドをサポートします。

getErrorCode

用途

このメソッドは、エラー番号（エラー・コード）を取得します。

構文

```
public int getErrorCode()
```

getNextException

用途

このメソッドは、スタック内に次の例外があればその例外を取得します。

構文

```
public java.lang.Exception getNextException()
```

AQLiteDeploymentAdmin

このクラスは、ディプロイメント準備中の様々なキュー・オブジェクトを管理するために使用します。

コンストラクタ

用途

AQLiteDeploymentAdmin クラスを構成します。

構文

```
public AQLiteDeploymentAdmin(java.lang.Object obj)
```

パラメータ

パラメータ	説明
conn	Oracle データベース接続のオブジェクト

createOliteSite

用途

このメソッドは、OliteSite オブジェクトを作成します。

構文 1

```
public AQLiteDeploymentOliteSite createOliteSite(java.lang.String oliteSiteID,
                                                 java.lang.String applicationName,
                                                 java.lang.String owner,
                                                 java.lang.String queue,
                                                 boolean isLocal,
                                                 java.lang.String comments)
                                                    throws AQException
```

構文 2

```
public AQLiteDeploymentOliteSite createOliteSite(java.lang.String oliteSiteID,
                                                 java.lang.String applicationName,
                                                 boolean isLocal,
                                                 java.lang.String comments)
                                                    throws AQException
```

パラメータ

パラメータ	説明
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID。
applicationName	指定された OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ。
owner	ディプロイメント・メッセージ・キューの所有者。デフォルトを使用するときは、NULL に設定します。
queue	ディプロイメント・メッセージ・キューのキュー。デフォルトを使用するときは、NULL に設定します。
isLocal	OliteSite キューのタイプ。
comments	関連するコメント。

getOliteSite

用途

このメソッドは、OliteSite オブジェクトを取得します。

構文

```
public AQLiteDeploymentOliteSite getOliteSite(java.lang.String oliteSiteID,
                                              java.lang.String applicationName)
```

パラメータ

パラメータ	説明
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID
applicationName	この OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ

dropOliteSite

用途

このメソッドは、OliteSite オブジェクトを削除します。

構文

```
public void dropOliteSite(java.lang.String oliteSiteID,
                          java.lang.String applicationName)
```

パラメータ

パラメータ	説明
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID
applicationName	この OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ

createQueueGroup

用途

このメソッドは、キューの配列によって QueueGroup オブジェクトを作成します。

構文 1

```
public AQLiteDeploymentQueueGroup createQueueGroup
               (java.lang.String groupName,
                java.lang.String comments,
                java.lang.String[] owner,
                java.lang.String[] queue)
                   throws AQException
```

構文 2

```
public AQLiteDeploymentQueueGroup createQueueGroup
                (java.lang.String groupName,
                 java.lang.String comments,
                    throws AQException)
```

パラメータ

パラメータ	説明
groupName	グループの名前
comments	関連するコメント
owner	キュー所有者の配列
queue	キューの配列

getQueueGroup

用途

このメソッドは、QueueGroup オブジェクトを取得します。

構文

```
public AQLiteDeploymentQueueGroup getQueu(java.lang.String groupName)
```

パラメータ

パラメータ	説明
groupName	グループの名前

dropQueueGroup

用途

このメソッドは、QueueGroup を削除します。

構文

```
public void dropQueueGroup(java.lang.String groupName)
```

パラメータ

パラメータ	説明
groupName	グループの名前

AQLiteDeploymentClient

このクラスは、Oracle Lite クライアント端末上でディプロイメントを実行するために使用します。

コンストラクタ

用途

AQLiteDeploymentClient クラスを構成します。

構文

```
public AQLiteDeploymentClient(oracle.lite.AQ.AQLiteSession aqConn,
                              java.lang.String applName,
                              AQLitePropOraConn ora_svr_db,
                              AQLitePropagation propagator,
                              java.lang.String oliteSiteID)
                                 throws AQException
```

パラメータ

パラメータ	説明
aqConn	AQ Lite 接続
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID
applName	この OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ
ora_svr_db	Oracle サーバーのデータベース接続オブジェクト
propagator	伝播オブジェクト

execDeployment

用途

このメソッドは、AQ Lite クライアント・サイトのディプロイメントを実行します。

構文

```
public void execDeployment(boolean creCat,
                           boolean propMsg)
                              throws AQException
```

パラメータ

パラメータ	説明
creCat	AQ Lite のカタログ情報を作成します。
propMsg	すでに存在するメッセージをディプロイメント時に伝播します。

AQLiteDeploymentOliteSite

このクラスは、OliteSite オブジェクトをディプロイメント用に作成、準備するために使用します。

コンストラクタ

用途

AQLiteDeploymentOliteSite クラスを構成します。2 番目の構文では owner、queue および isLocal パラメータにデフォルトを使用します。

構文 1

```
public AQLiteDeploymentOliteSite(java.lang.Object obj,
                                 java.lang.String oliteSiteID,
                                 java.lang.String applicationName,
                                 java.lang.String owner,
                                 java.lang.String queue,
                                 boolean isLocal,
                                 java.lang.String comments)
                                    throws AQException
```

構文 2

```
public AQLiteDeploymentOliteSite(java.lang.Object obj,
                                 java.lang.String oliteSiteID,
                                 java.lang.String applicationName)
                                    throws AQException
```

パラメータ

パラメータ	説明
conn	Oracle データベース接続のオブジェクト。
oliteSiteID	Oracle Lite クライアント・データベース・サイトの ID。
applicationName	この OliteSite オブジェクトのディプロイメントと伝播に使用されるコンシューマ。
owner	ディプロイメント・メッセージ・キューの所有者。デフォルトを使用するときは、NULL に設定します。
queue	ディプロイメント・メッセージ・キュー。デフォルトを使用するときは、NULL に設定します。
isLocal	OliteSite のキュー・タイプ。
comments	コメント。

drop

用途

このメソッドは、OliteSite オブジェクトを削除します。

構文

```
public void drop()
            throws AQException
```

getQueueGroup

用途

このメソッドは、OliteSite オブジェクト内に定義された QueueGroup オブジェクトを取得します。2 番目の構文は、OliteSite オブジェクト内に定義されたすべての QueueGroup オブジェクトを取得します。

構文 1

```
public AQLiteDeploymentQueueGroup getQueueGroup(java.lang.String groupName)
   throws AQException
```

構文 2

```
public java.lang.String[] getQueueGroup() throws AQException
```

戻り値

QueueGroup オブジェクトが 1 つも存在しない場合、NULL を返します。

パラメータ

パラメータ	説明
groupName	グループの名前
queueGroup	QueueGroup オブジェクト

defineQueueGroup

用途

このメソッドは、OliteSite オブジェクトに QueueGroup オブジェクトを定義します。

構文 1

```
public void defineQueueGroup(java.lang.String groupName)
   throws AQException
```

構文 2

```
public void defineQueueGroup(AQLiteDeploymentQueueGroup queueGroup)
   throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前
queueGroup	QueueGroup オブジェクト

isDefinedQueueGroup

用途

このメソッドは、OliteSite オブジェクトに QueueGroup オブジェクトが定義されているかチェックします。

構文

```
public boolean isDefinedQueueGroup(java.lang.String groupName)
   throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前

undefineQueueGroup

用途

このメソッドは、OliteSite オブジェクトの QueueGroup オブジェクトの定義を解除します。

構文 1

```
public void undefineQueueGroup(java.lang.String groupName)
   throws AQException
```

構文 2

```
public void undefineQueueGroup(AQLiteDeploymentQueueGroup queueGroup)
   throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前
queueGroup	QueueGroup オブジェクト

instantiate

用途

このメソッドは、OliteSite オブジェクトのディプロイメントのインスタンスを生成します。groupName が指定されない場合、OliteSite オブジェクトに定義されているすべてのグループのインスタンスが生成されます。

構文 1

```
public void instantiate(java.lang.String groupName) throws AQException
```

構文 2

```
public void instantiate() throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前

deInstantiate

用途

このメソッドは、OliteSite オブジェクトに対するディプロイメント・インスタンスを削除します。groupName が指定されない場合、OliteSite オブジェクトに定義されているすべてのグループのインスタンスが削除されます。

構文 1

```
public void deInstantiate(java.lang.String groupName) throws AQException
```

構文 2

```
public void deInstantiate() throws AQException
```

パラメータ

パラメータ	説明
groupName	グループの名前

AQLiteDeploymentQueueGroup

このクラスは、QueueGroup オブジェクトのディプロイメントを管理するために使用します。

コンストラクタ

用途

AQLiteDeploymentQueueGroup クラスを構成します。そのクラスに対応するデータベース・オブジェクトが存在しない場合、コンストラクタがインポート・パラメータに基づいてデータベースを作成します。

構文 1

```
public AQLiteDeploymentQueueGroup(java.lang.Object obj,
                                  java.lang.String queueGroup)
                                     throws AQException
```

構文 2

```
public AQLiteDeploymentQueueGroup(java.lang.Object obj,
                                  java.lang.String queueGroup,
                                  java.lang.String comments)
                                     throws AQException
```

構文 3

```
public AQLiteDeploymentQueueGroup(java.lang.Object obj,
                                  java.lang.String queueGroup,
                                  java.lang.String comments,
                                  java.lang.String[] owner,
                                  java.lang.String[] queue)
                                     throws AQException
```

パラメータ

パラメータ	説明
conn	Oracle データベース接続のオブジェクト
queueGroup	QueueGroup の名前
comments	関連するコメント

パラメータ	説明
owner	キュー所有者の配列
queue	キューの配列

getName

用途

このメソッドは、QueueGroup の名前を取得します。

構文

```
public java.lang.String getName()
```

drop

用途

QueueGroup オブジェクトを削除します。

構文

```
public void drop() throws AQException
```

getQueues

用途

このメソッドはすべてのキュー名の配列を取得し、次のベクトルに返します。

- ベクトル [2*i] にはキュー所有者を返します。
- ベクトル [2*i+1] にはキュー名を返します。

構文

```
public java.util.Vector getQueues() throws AQException
```

getOliteSite

用途

このメソッドは、QueueGroup オブジェクトと関連するすべての OliteSite オブジェクトの配列を取得します。

構文

```
public java.lang.String[] getOliteSite() throws AQException
```

defineDeployTo

用途

このメソッドは、QueueGroup がディプロイメントされる OliteSite オブジェクトを定義します。

構文

```
public void defineDeployTo(java.lang.String OliteSite,
                           java.lang.String user)
                              throws AQException
```

undefineDeployTo

用途

このメソッドは、QueueGroup がディプロイメントされる OliteSite オブジェクトの定義を削除します。

構文

```
public void undefineDeployTo(java.lang.String OliteSite,
                             java.lang.String user)
                                throws AQException
```

addQueue

用途

このメソッドは、QueueGroup にキューを追加します。

構文

```
public void addQueue(java.lang.String[] owner,
                     java.lang.String[] queue)
                        throws AQException
```

removeQueue

用途

このメソッドは、QueueGroup からキューを削除します。

構文

```
public void removeQueue(java.lang.String[] owner,
                        java.lang.String[] queue)
                           throws AQException
```

AQLitePropagator

このクラスは、AQ Lite プロパゲータを実行するために使用します。クラスには次のものが含まれます。

- AQLitePropagator

start_queue_prop

用途

このメソッドは伝播を起動します。

構文

```
public void start_queue_prop(oracle.pol.jac.POLConnection olitedb,
                             AQLitePropOraConn ora_svr_db,
                             boolean enqueue_enable,
                             boolean dequeue_enable,
                             boolean long_trans_enable)
```

パラメータ

パラメータ	説明
olitedb	Oracle Lite データベース接続。
ora_svr_db	Oracle8*i* データベース接続をオープンするための必須フィールド。
enqueue_enable	Oracle8*i* から Oracle Lite への伝播を使用可能にします。
dequeue_enable	Oracle Lite から Oracle8*i* への伝播を使用可能にします。
long_trans_enable	AQ Lite の現在のリリースでは使用されません。

start_queue_depl

用途

このメソッドはディプロイメントを起動します。

構文

```
public void start_queue_depl(oracle.pol.jac.POLConnection olitedb,
                             AQLitePropOraConn ora_svr_db,
                             boolean enqueue_enable,
                             boolean dequeue_enable,
                             boolean long_trans_enable)
```

パラメータ

パラメータ	説明
olitedb	Oracle Lite データベース接続。
ora_svr_db	Oracle8*i* データベース接続をオープンするための必須フィールド。
enqueue_enable	Oracle8*i* から Oracle Lite への伝播を使用可能にします。
dequeue_enable	Oracle Lite から Oracle8*i* への伝播を使用可能にします。
long_trans_enable	AQ Lite の現在のリリースでは使用されません。

stop_queue_prop

このメソッドは、AQ Lite の現在のリリースではサポートされていません。

pause_queue_prop

このメソッドは、AQ Lite の現在のリリースではサポートされていません。

resume_queue_prop

このメソッドは、AQ Lite の現在のリリースではサポートされていません。

AQLitePropTransport

これは、AQ Lite Propagation Transport の共通インタフェースです。クラスには次のものが含まれます。

- `AQLitePropTransportNet8`
- `AQLitePropServer`

openConnection

用途

このメソッドは、Oracle8*i* サーバーへの接続を確立し、サーバー上の `OliteSite` ID を検証します。

構文

```
public int openConnection(AQLitePropOraConn ora_svr_db,
                          java.lang.String olite_id,
                          int outbound_prop_id,
                          java.lang.String propcat_table,
                          int dequeue_mode)
```

パラメータ

パラメータ	説明
`ora_svr_db`	Oracle8*i* データベース接続をオープンするための必須フィールド
`olite_id`	Oracle Lite サイトの ID
`outbound_prop_id`	アウトバウンド（クライアントからサーバー方向）伝播 ID

パラメータ	説明
propcat_table	propcat（伝播カタログ）表の名前
dequeue_mode	AQ のデキュー・モード

戻り値

インバウンド（サーバーからクライアント方向）伝播 ID

propEnqueue

用途

このメソッドは、AQ Lite プロパゲータによってローカル・クライアント端末へのメッセージをエンキュするために使用されます。また、ローカル・アプリケーションでは、このメソッドを使用して別のローカル・アプリケーションが使用するメッセージをエンキュします。

構文

```
public void propEnqueue(AQMessage msg,
                        java.lang.String in_qOwner,
                        java.lang.String in_qName,
                        boolean new_queue)
```

propDequeue

用途

このメソッドは、Oracle8*i* サーバーからメッセージをデキューします。

構文

```
public AQLitePropQueue propDequeue()
```

propCommit

用途

このメソッドは、伝播トランザクションをコミットします。

構文

```
public void propCommit(int option)
```

propRollback

用途

このメソッドは、伝播トランザクションをロールバックします。

構文

```
public void propRollback(int option)
```

closeConnection

用途

このメソッドは Oracle8*i* サーバーへの接続をクローズします。

構文

```
public void closeConnection()
```

B

AQ Lite サンプル・アプリケーション

この章では、Oracle AQ Lite サンプル・アプリケーションの開発と使用方法を説明します。説明する内容は、次のとおりです。

- AQ Lite サンプル・アプリケーションの設定
- JFC Swing パッケージのインストール
- AQ Lite サンプル・アプリケーションの使用

AQ Lite サンプル・アプリケーションの設定

AQ Lite サンプル・アプリケーションをセットアップするために、次のプロシージャを実行します。

AQ Lite をインストールする

AQ Lite のインストール手順は、『Oracle9*i* Lite インストレーションおよび構成ガイド』を参照してください。AQ Lite をインストールすると、*ORACLE_HOME*¥Mobile¥SDK¥AQLite¥AQLSAMP ディレクトリにあるサンプル・アプリケーションが使用可能になります。

Oracle8*i* サーバーをセットアップする

1. Oracle8*i* AQ がサーバーにインストールされ、使用可能になっていることを確認します。
2. SQL スクリプト **AQLiteO8.sql** をまだ実行していない場合は、ユーザーの SYSTEM（デフォルトのパスワードは MANAGER）として Oracle8*i* サーバー上で実行します。このスクリプトは、*ORACLE_HOME*/Mobile/SDK/AQLite/SQL ディレクトリにあります。

> **警告：** SQL スクリプト AQLiteO8.sql を実行すると、あらゆる既存の AQ Lite 情報が Oracle8*i* データベースから削除されます。

3. AQLiteSamp ディレクトリにある Java プログラム **setupDeplMsgQ.class**（**aqsample.jar** ファイルに含まれています）を実行します。

```
java setupDeplMsg [o8UserID]  [o8Passwd]  [o8ConnStr]
```

例： `java setupDeplMsgQ AQUSER AQUSER my_machine.mydomain.com:1521:orcl`

引数	説明	例
o8UserId	Oracle8*i* のユーザー ID	"system"
o8Passwd	Oracle8*i* のユーザー・パスワード	"manager"
o8ConnStr	Oracle8*i* の接続文字列	"my_machine.mydomain.com:1521:orcl"

Oracle Lite サイトをセットアップする

1. 有効な Oracle Lite データベースを作成します。
2. AQLiteSamp ディレクトリにある Java プログラム **setupServer.class**（**aqlsample.jar** ファイルに含まれています）を実行します。

```
java setupServer [o8UserId] [o8Passwd] [o8ConnStr]
```

例： `java setupServer AQUSER AQUSER my_machine.my domain.com:1521:orcl`

引数	説明	例
o8UserId	Oracle8*i* のユーザー ID	"system"
o8Passwd	Oracle8*i* のユーザー・パスワード	"manager"
o8ConnStr	Oracle8*i* の接続文字列	"my_machine.mydomain.com:1521:orcl"

3. AQLiteSamp ディレクトリにある Java プログラム **setupClient.class**（**aqlsample.jar** ファイルに含まれています）を実行します。

```
java SetupClient [o8UserId]  [o8Passwd]  [o8ConnStr]  [option]  [oliteSiteName]
```

例： `java setup client AQUSER AQUSER my_machine.mydomain.com:1521:orcl ORD C:¥ORANT¥OLDB40¥politeProc.odb oliteSite 1`
または
`java setupClient AQUSER AQUSER my_machine.mydomain.com:1521:orcl PROC C:¥ORANT¥OLDB40¥politeProc.odb oliteSite 2`

引数	説明	例
o8UserId	Oracle8*i* のユーザー ID	"system"
o8Passwd	Oracle8*i* のユーザー・パスワード	"manager"
オプション	"ORD" は、ローカル・データベースを注文入力用にセットアップします。	"ORD"
	"PROC" は、ローカル・データベースを注文処理用にセットアップします。	" PROC"
o8ConnStr	Oracle Lite データベース・ファイルのパス	"C:¥ORANT¥OLDB40¥politeOrd.odb"（注文入力用）
		"C:¥ORANT¥OLDB¥politePROC.odb"（注文処理用）

引数	説明	例
oliteSiteName	Oracle Lite のデータベース ID	"oliteSite 1" は注文入力用、"oliteSite 2" は注文処理用

JFC Swing パッケージのインストール

Java Foundation Class (JFC) Swing パッケージをインストールしていない場合は、CLASSPATH に jar ファイル **swing.jar**（AQLiteSamp ディレクトリにあります）を組み込みます。

AQ Lite サンプル・アプリケーションの使用

AQ Lite サンプル・アプリケーションは、次のモジュールで構成されています。

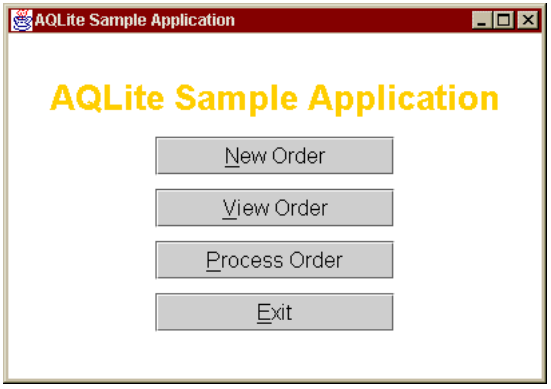
- New Order
- View Order
- Process Order

New Order および View Order モジュールは、AQ Lite クライアントに常駐する注文入力アプリケーションの 2 つのコンポーネントを表します。New Order モジュールを使用して売上注文を作成し、Process Order モジュールが取り出せるようにそれらをデータ・センター（Oracle8*i* サーバー）に送信できます。View Order モジュールを使用して、処理済の注文の情報を参照できます。

Process Order モジュールは、別の AQLite クライアントに常駐する注文処理アプリケーションを表します。このモジュールは、データ・センター（Oracle8*i* サーバー）から新規の注文を取り出して処理します。

サンプル・アプリケーションの起動

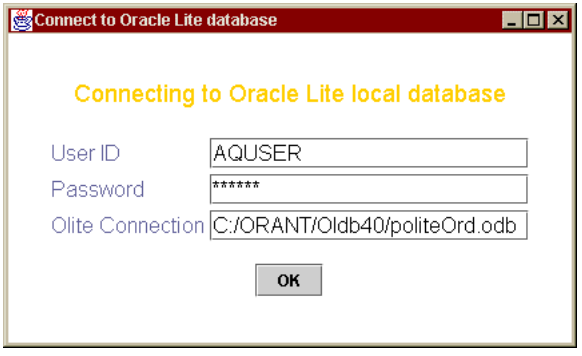
サンプル・アプリケーションを起動するときは、`java AQLiteSample` と入力して **AQLiteSample.class** を実行します。メイン・ウィンドウが表示されます。



New Order モジュールの使用

次のようにして、New Order モジュールをオープンします。

1. サンプル・アプリケーションのメイン・ウィンドウの「New Order」ボタンをクリックします。Oracle Lite データベースに接続するためのダイアログ・ボックスが表示されます。

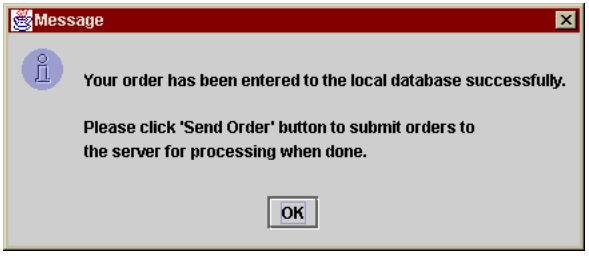


2. ユーザー ID、パスワードおよび Oracle Lite データベースのパスを入力して、「OK」ボタンをクリックします。「Enter and Send New Order」ウィンドウが表示されます。

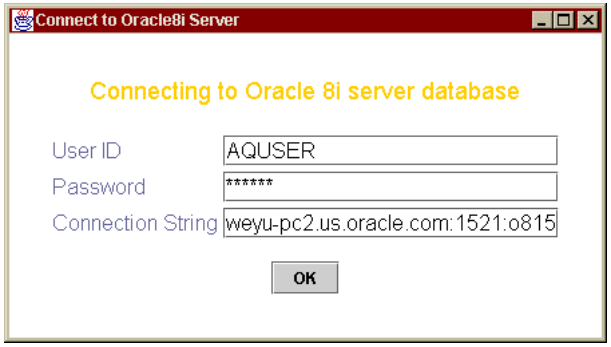
注意： New Order への Oracle Lite 接続文字列を正確に入力する必要があります（たとえば、c:¥ORANT¥OLDB¥politeORD.odb）。

次のようにして、新規注文を作成します。

1. 「Customer」フィールドに顧客名を入力します。
2. 「Item」ドロップダウン・リストから、製品を選択します。
3. 「Quantity」フィールドに数量を入力します。
4. 「Payment」ドロップダウン・リストから、支払方法を選択します。
5. 「Shipping」ドロップダウン・リストから、発送方法を選択します。
6. 「Enter Order」ボタンをクリックして、注文を完了します。新規注文が作成されて、AQ Lite クライアント・キューにエンキュされます。
7. すべての注文がローカル・データベースに正常に入力されたというメッセージが表示されます。「OK」ボタンをクリックします。



8. 「Enter and Send Order」ウィンドウの「Send Order」ボタンをクリックします。Oracle8*i* サーバーに接続するためのダイアログ・ボックスが表示されます。



9. Oracle8*i* サーバーへの接続ダイアログ・ボックスで、ユーザー ID、パスワードおよび接続文字列を対応するフィールドへ入力して、「OK」ボタンをクリックします。

注意： Oracle8*i* サーバー・データベースの正確な接続文字列を [hostname] : [port number] : [SID] の形式で入力する必要があります（たとえば、mymachine.mydomain.com:1521:orcl）。

---

10. 新規注文を作成すると、それを Process Order モジュールに送って処理できます。新規注文を Process Order モジュールに送るためには、「Send Order」ボタンをクリックします。AQ Lite は、クライアント・キューにあるすべての新規注文をマスター・キューに伝播します。マスター・キューに入った注文は、Process Order モジュールによってデキュー可能になります。

11. すべての注文が処理のためにサーバーに転送されたというメッセージが表示されます。「OK」ボタンをクリックします。

12. これで、すべての新規注文が販売部門（Oracle Lite Database 1）から処理のためにデータ・センター（Oracle8*i* サーバー）に転送されました。終了してください。

Process Order モジュールの使用

次のようにして、Process Order モジュールをオープンします。

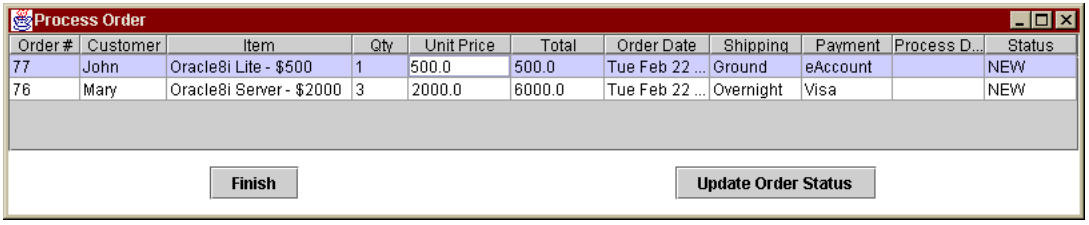
1. サンプル・アプリケーションのメイン・ウィンドウの「Process Order」ボタンをクリックします。Oracle Lite データベースに対する接続ダイアログ・ボックスが表示されます。
2. ユーザー ID、パスワードおよび接続文字列を対応するフィールドへ入力して、「OK」ボタンをクリックします。Oracle8*i* サーバーへの接続ダイアログ・ボックスが表示されます。

注意： Process Order に対するデータベース接続文字列を正確に入力する必要があります（たとえば、c:¥ORANT¥OLDB40¥politePROC.odb）。

---

3. Oracle8*i* サーバーへの接続ダイアログ・ボックスで、ユーザー ID、パスワードおよび接続文字列を対応するフィールドへ入力して、「OK」ボタンをクリックします。「Process Order」ウィンドウが表示されます。

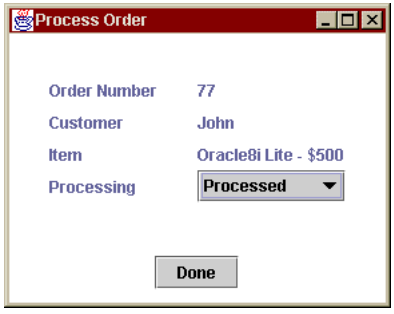
注意： Oracle8*i* サーバー・データベースの正確な接続文字列を [hostname] : [port number] : [SID] の形式で入力する必要があります（たとえば、mymachine.mydomain.com:1521:orcl）。



Process Order モジュールが起動すると、すべての新規注文メッセージをデキューし、その情報を「Process Order」ウィンドウに表示します。

次のようにして、注文を処理します。

1. 処理する注文を選択して、「Update Order Status」ボタンをクリックします。「Process Order」ダイアログ・ボックスが表示されます。



2. 「Processing」ドロップダウン・リストから注文の状態を選択して「Done」ボタンをクリックします。
3. 処理する各注文について、手順 1 および 2 を繰り返します。
4. 「Process Order」ウィンドウの「Finish」ボタンをクリックします。処理済の注文がマスター・キュー（Oracle Lite Database 2）に伝播されます。
5. これで、処理センターで注文が処理され、データ・センター（Oracle8*i* サーバー）に送り返されました。終了してください。

View Order モジュールの使用

次のようにして、View Order モジュールをオープンします。

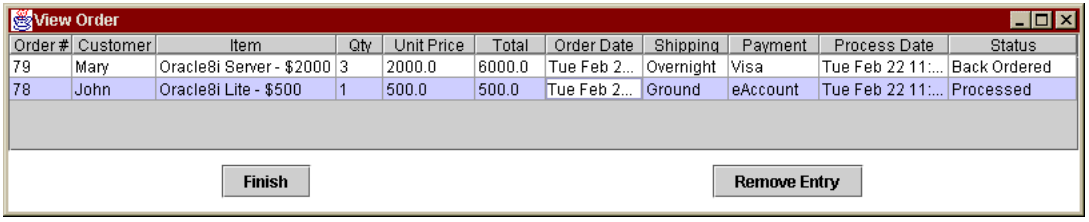
1. メイン・ウィンドウの「View Order」ボタンをクリックします。Oracle Lite データベースへの接続ダイアログ・ボックスが表示されます。
2. ユーザー ID、パスワードおよび接続文字列を対応するフィールドへ入力して、「OK」ボタンをクリックします。

注意： View Order の場合、New Order と同じデータベース接続文字列を正確に入力する必要があります（たとえば、c:¥ORANT¥OLDB40¥politeORD.odb）。

---

3. Oracle8*i* サーバーへの接続ダイアログ・ボックスが表示されます。ユーザー ID、パスワードおよび接続文字列を対応するフィールドへ入力して、「OK」ボタンをクリックします。「View Order」ウィンドウが表示されます。

注意： Oracle8*i* サーバー・データベースの正確な接続文字列を [hostname] : [port number] : [SID] の形式で入力する必要があります（たとえば、mymachine.mydomain.com:1521:orcl）。



View Order モジュールが起動されると、AQ Lite は処理済の注文メッセージをマスター・キュー（データ・センターの Oracle8*i* サーバー）からクライアント・キュー（販売 / 注文入力部門の Oracle LiteDatabase 1）に伝播します。View Order モジュールは、処理済の注文メッセージをデキューしてその情報を表示します。

次のようにして、注文を削除します。

削除する注文をクリックして、「Remove Entry」ボタンをクリックします。注文がローカル・データベースからデキューされます。

サンプル・アプリケーション・コード

この項には、AQ Lite サンプル・アプリケーションのソース・コードがあります。開発者は、Oracle AQ Lite のキューイング・アプリケーションを開発するときにこのコードを参考できます。完全なサンプル・アプリケーション・ソース・コードは、*ORACLE_HOME*/Mobile/SDK/AQLite/AQLSAMP ディレクトリにあります。

サンプル・アプリケーション・ユーザー・インタフェースの実装 : AQLiteSample.java

```
/*
  DESCRIPTION
    Main class - draws AQLite Sample App's main window

  NOTES
    Other files in the sample app:

    AQLiteSampQUtil.java     - all AQLite queue operations
    AQLiteSampUtil.java      - miscellaneous GUI and other operations
    AQLiteSampNewOrder.java  - handles "New Order" operations
    AQLiteSampProcOrder.java - handles "Process Order" operations
    AQLiteSampViewOrder.java - handles "View Order" operations


  */

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class AQLiteSample implements ActionListener{

  static JFrame frame = null;

  // Create Components
  public Component createComponents() {


    // GUI initialization
    final JLabel label = new JLabel("AQLite Sample Application");

    // create buttons
    JButton button1 = new JButton("New Order");
    JButton button2 = new JButton("View Order");
    JButton button3 = new JButton("Process Order");
    JButton button4 = new JButton("Exit");


    label.setFont(AQLiteSampUtil.fontb);
    button1.setFont(AQLiteSampUtil.fonts);
    button2.setFont(AQLiteSampUtil.fonts);
    button3.setFont(AQLiteSampUtil.fonts);
    button4.setFont(AQLiteSampUtil.fonts);
```

```
// set mnemonic keys for users to stimulate button click.
button1.setMnemonic(KeyEvent.VK_N); //(Alt-n)
button2.setMnemonic(KeyEvent.VK_V); //(Alt-v)
button3.setMnemonic(KeyEvent.VK_P); //(Alt-p)
button4.setMnemonic(KeyEvent.VK_E); //(Alt-e)

// set unique identifier to button
button1.setActionCommand(AQLiteSampUtil.bn_main_NO);
button2.setActionCommand(AQLiteSampUtil.bn_main_VO);
button3.setActionCommand(AQLiteSampUtil.bn_main_PO);
button4.setActionCommand(AQLiteSampUtil.bn_main_EX);

button1.addActionListener(this);
button2.addActionListener(this);
button3.addActionListener(this);
button4.addActionListener(this);
// set button colours
//button1.setOpaque(true);
//button1.setBackground(Color.blue);


// put the contents in a JPanel that has an "empty" border
// to put space between a top-level container and its contents
JPanel pane = new JPanel();
pane.setOpaque(true);
pane.setBackground(Color.white);
pane.setBorder(BorderFactory.createEmptyBorder(30, //top 30 pixels
                                               30, //left 30 pixels
                                               30, //bottom
                                               30) //right
               );

// set layout manager
pane.setLayout(new GridBagLayout());

GridBagConstraints c = new GridBagConstraints();
c.gridx = 0;
c.gridy = 0;
c.gridwidth = 3;
c.gridheight = 1;
c.insets = new Insets(3,2,5,2);
c.weightx = 1.0;
c.weighty = 1.0;

// add label to panel
pane.add(label, c);
```

```
    c.gridy = GridBagConstraints.RELATIVE;
    c.fill = GridBagConstraints.HORIZONTAL;
c.gridx = 2;
    c.anchor = GridBagConstraints.CENTER;
    c.gridwidth = 1;
    c.weightx = 0.0;
    c.weighty = 1.0;
    c.insets = new Insets(3,2,3,2);
    // add buttons to panel
    pane.add(button1, c);
    pane.add(button2, c);
    pane.add(button3, c);
    pane.add(button4, c);

    return pane;
  }


  public void actionPerformed (ActionEvent evt) {
    String cmd = evt.getActionCommand();
    // "New Order" button click
    if (cmd.equals(AQLiteSampUtil.bn_main_NO)) {
      AQLiteSampNewOrder newOrder = new AQLiteSampNewOrder();
      newOrder.start();
    }
    // "View Order" button click
    else if (cmd.equals(AQLiteSampUtil.bn_main_VO)) {
      AQLiteSampViewOrder viewOrder = new AQLiteSampViewOrder();
      viewOrder.start();
    }
    // "Process Order" button click
    else if (cmd.equals(AQLiteSampUtil.bn_main_PO)) {
      AQLiteSampProcOrder procOrder = new AQLiteSampProcOrder();
      procOrder.start();
    }
    // "Exit" button click
    else if (cmd.equals(AQLiteSampUtil.bn_main_EX)) {
      String dialog_txt = "Do you really want to quit?";
      String dialog_title = "AQlite Sample Application - Quit";
      int response = JOptionPane.showConfirmDialog(frame,
                                                   dialog_txt,
                                                   dialog_title,
                                                   JOptionPane.YES_NO_OPTION);
```

```
        switch (response) {
        case JOptionPane.YES_OPTION:
          System.exit(0);
          break;
        case JOptionPane.NO_OPTION:
          break;
        case JOptionPane.CLOSED_OPTION:
          break;
        }
      }

    }


    ////////////////////////////////////////////////
    public static void main(String[] args) {
      try {
        // Set Look and Feel
        UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
      }
      catch (Exception e) { }

      frame = new JFrame("AQLite Sample Application");

      AQLiteSample app = new AQLiteSample();

      // create the components to go into the frame
      // and stick them in a container named contents
      Component contents = app.createComponents();
      frame.getContentPane().add(contents, BorderLayout.CENTER);

      // Window close
      frame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {System.exit(0);}
      });
      // Size
      frame.pack();
      //Finish setting up the frame, and show it.
      frame.setVisible(true);
    }
  }
```

ディプロイメントのためのメッセージ・キューの準備 : setupDeplMsgQ.java

次のコードは、サンプル・アプリケーションの中で、ディプロイメントのためのメッセージ・キューの準備に使用されます。

```
import oracle.AQ.*;
import oracle.lite.AQ.*;
import java.util.*;
import java.sql.*;
import java.math.*;

public class setupDeplMsgQ {

  public static final String AQLDeplMsgQT = "AQLITE$_DeplMsgQT";
  public static final String AQLDeplMessages = "AQLITE$_DeplMessages";

  public static void main(String[] args) {

    if (args.length !=3 || args[0] == null || args[1] == null ||
        args[2] == null) {
      System.err.println("Usage: java setupDeplMsgQ [o8UserId] [o8Passwd]
[o8ConnStr]");
      System.err.println("¥n¥n Example:");
      System.err.println("java setupDeplMsgQ AQUSER AQUSER my_machine.my_
domain.com:1521:orcl");
      System.exit(0);
    }
    String orauser = args[0];
    String orapswd = args[1];
    String connstr = "jdbc:oracle:thin:@" + args[2];
    AQSession aqConn = null;

    try {
      // JDBC Thin Driver

      DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver());
      Connection conn =
        DriverManager.getConnection(connstr,orauser, orapswd);
      conn.setAutoCommit(false);
      AQOracleDriver aqD =   new AQOracleDriver();

      // get connection to Olite database
      aqConn = aqD.createAQSession((Object) conn);
```

```
//prepare a queuetable property
AQQueueTableProperty AQqtp = new AQQueueTableProperty("RAW");
AQqtp.setStorageClause("");
//AQqtp.setAutoCommit(false);
AQqtp.setComment("AQLITE Queue for deployment messages");
AQqtp.setCompatible("");
AQqtp.setMessageGrouping(AQqtp.NONE);
AQqtp.setMultiConsumer(true);
AQqtp.setPrimaryInstance(1);
AQqtp.setSecondaryInstance(0);
AQqtp.setSortOrder("ENQ_TIME");

//createQT
AQQueueTable tqt = null;
try {
  tqt = aqConn.getQueueTable(orauser, AQLDeplMsgQT);
}
catch (AQException e)
{
  //System.out.println(e);
}
try {
  if (tqt != null)
    tqt.drop(true);
}
catch (AQException e)
{
  System.err.println("Ooops: " + e.getMessage());
  e.printStackTrace();
}
tqt = aqConn.createQueueTable(orauser, AQLDeplMsgQT, AQqtp);


//prepare a queue property
AQQueueProperty AQqp = new AQQueueProperty();
//AQqp.setAutoCommit(false);
AQqp.setComment("test Queue");
//AQqp.setDepTracking(false);
//AQqp.setMaxRetries(100);
AQqp.setQueueType(AQqp.NORMAL_QUEUE);
AQqp.setRetentionTime(0.0) ;
//AQqp.setRetryInterval(0.0);

// create Q on the QT
AQQueue q = aqConn.createQueue(tqt, AQLDeplMessages, AQqp);
q.start();
```

```
      // commit changes
      conn.commit();
    }
    catch (AQException e)
    {
      System.err.println("Ooops: " + e.getMessage());
      e.printStackTrace();
    }
    catch (SQLException e)
    {
      System.err.println("Ooops: " + e.getMessage());
      e.printStackTrace();
    }
  }
}
```

ディプロイメントのための Oracle8*i* サーバー側の準備 : setupSever.java

次のコードは、サンプル・アプリケーションの中で、ディプロイメントのための Oracle Lite サーバー・デバイス側の準備に使用されています。

```
import java.util.*;
import java.sql.*;
//import java.math.*;
import oracle.AQ.*;
import oracle.lite.AQ.*;

public class setupServer{

  public static void main(String[] args) {

    if (args.length !=3 || args[0] == null || args[1] == null ||
        args[2] == null) {
      System.err.println("Usage: java setupServer [o8UserId] [o8Passwd]
[o8ConnStr]");
      System.err.println("¥n¥n Example:");
      System.err.println("java setupServer AQUSER AQUSER my_machine.my_
domain.com:1521:orcl");
      System.exit(0);
    }
```

```
String o8UserID = args[0];
String o8Passwd = args[1];
String o8ConnStr = "jdbc:oracle:thin:@" + args[2];

AQSession aqConn;

try {
  // connect to O8 via JDBC
  DriverManager.registerDriver (new oracle.jdbc.driver.OracleDriver());
  Connection conn = DriverManager.getConnection(o8ConnStr,o8UserID,o8Passwd);
  conn.setAutoCommit(false);


  // do not change the following if you are not sure
  String[] qn = new String[1];
  qn[0] = "AQLite_SampOrderQ";
  String[] qo = new String[1];
  qo[0] = o8UserID;
  String qt = "AQLite_SampOrderQT";

  //create Queue Table and Queue
  AQOracleDriver aqD =   new AQOracleDriver();
  aqConn = aqD.createAQSession((Object) conn);

  //prepare a queuetable property
  AQQueueTableProperty AQqtp = new AQQueueTableProperty("RAW");
  AQqtp.setStorageClause("");
  //AQqtp.setAutoCommit(false);
  AQqtp.setComment("AQLite Sample Application");
  AQqtp.setCompatible("");
  AQqtp.setMessageGrouping(AQqtp.NONE);
  AQqtp.setMultiConsumer(true);
  AQqtp.setPrimaryInstance(1);
  AQqtp.setSecondaryInstance(0);
  AQqtp.setSortOrder("ENQ_TIME");

  //createQT
  AQQueueTable tqt = null;
  try {
    tqt = aqConn.getQueueTable(qo[0], qt);
  }
  catch (AQException e) {
    System.err.println("Oooops. " + e.getMessage());
    e.printStackTrace();
```

```
}
try {
  if (tqt != null)
  {
    //cleaning
    AQLiteDeploymentAdmin admin = new AQLiteDeploymentAdmin(conn);
    AQLiteDeploymentOliteSite os =admin.getOliteSite("OliteSite1",
                                                     "sampUser");
    //os.deInstantiate("AQLSampQG1");
    try
    {//test here
      AQLiteDeploymentQueueGroup qg = os.getQueueGroup("AQLSampQG1");
      qg.drop();
    }
    catch (AQException e){}
    tqt.drop(true);
  }
}
catch (AQException e) {
  System.err.println("Oooops. " + e.getMessage());
  e.printStackTrace();
}
tqt = aqConn.createQueueTable(qo[0], qt, AQqtp);


//prepare a queue property
AQQueueProperty AQqp = new AQQueueProperty();
//AQqp.setAutoCommit(false);
AQqp.setComment("AQLite Sample Application");
//AQqp.setDepTracking(false);
AQqp.setQueueType(AQqp.NORMAL_QUEUE);
AQqp.setRetentionTime(0.0) ;

// create Q on the QT
AQQueue q = aqConn.createQueue(tqt, qn[0], AQqp);
q.start();


//deployment
AQLiteDeploymentAdmin admin = new AQLiteDeploymentAdmin(conn);

AQLiteDeploymentQueueGroup qg = admin.createQueueGroup("AQLSampQG1",
                                                       "order queue",
                                                       qo,
                                                       qn);
```

```
      AQLiteDeploymentOliteSite os = admin.createOliteSite("OliteSite1",
                                                           "sampUser",
                                                           null,
                                                           null,
                                                           true,
                                                           "AQLite Sample
Application");

      os.defineQueueGroup(qg);
      os.instantiate();
      /*
        AQLiteDeploymentAdmin admin = new AQLiteDeploymentAdmin(conn);

        AQLiteDeploymentOliteSite os = admin.getOliteSite("OliteSite2",
        "sampProc");
        //os.deInstantiate();
        os.instantiate();
      */
      os =admin.createOliteSite("OliteSite2",
                                "sampProc",
                                null,
                                null,
                                true,
                                "AQLite Sample Application");

      os.defineQueueGroup(qg);
      os.instantiate();

      // commit changes
      conn.commit();
    }//try
    catch (AQException e) {
      System.out.println("Oooops. " + e.getMessage());
      e.printStackTrace();
    }
    catch (java.sql.SQLException e) {
      System.out.println("Oooops. " + e.getMessage());
      e.printStackTrace();
    }
  }
}
```

ディプロイメントのための Oracle Lite クライアント・デバイス側の準備 : SetupClient.java

次のコードは、サンプル・アプリケーションの中で、ディプロイメントのための Oracle Lite クライアント・デバイス側の準備に使用されています。

```
import oracle.AQ.*;
import oracle.lite.jac.*;
import oracle.lite.AQ.*;

public class setupClient{
  public static void main(String[] args) {

    if (args.length !=6 || args[0] == null || args[1] == null ||
        args[2] == null || args[3] == null || args[4] == null ||
        args[5] == null) {
      show_usage();

    }
    String o8UserID = args[0];
    String o8Passwd = args[1];
    String o8ConnStr = args[2];
    boolean newDB = true;

    String o8lConnStr = args[4];
    String deqConsumer = null;
    if (args[3].toUpperCase().equals ("ORD"))
      deqConsumer = "sampUser";
    else if (args[3].toUpperCase().equals ("PROC"))
      deqConsumer = "sampProc";
    else {
      System.err.println("ERROR: option must be either 'ORD' or 'PROC'!!¥n¥n");
      show_usage();
    }
    String olitesite = args[5];


    AQSession aqConn;

    try {
      // connect to Olite via Jac
      POLConnection pConn = POLConnection.getConnection(o8lConnStr,
POLConnection.CONNECT_READ_WRITE);
      AQDriver aqD =  (AQDriver) new AQLiteDriver();
```

```
      // get connection to Olite database
      aqConn = aqD.createAQSession((Object) pConn);

      //make an O8 connection Object
      AQLitePropOraConn o8_conn =
        new AQLitePropOraConn("O8AQ",
                              o8UserID,
                              o8Passwd,
                              o8ConnStr);

      AQLitePropagator propagator = new AQLitePropagator();

      //make a deployment object
      AQLiteDeploymentClient dc = new AQLiteDeploymentClient((AQSession) aqConn,
                                                             deqConsumer,
                                                             o8_conn,
                                                             propagator,
                                                             olitesite
                                                             ) ;

      //do a deploymnet
      dc.execDeployment(newDB, false);


      // commit changes
      pConn.commit();
    }

    catch (AQException e) {
        System.err.println("Oooops. " + e.getMessage());
        e.printStackTrace();
    }
    catch (POLException e) {
        System.err.println("Oooops. " + e.getMessage());
        e.printStackTrace();
    }
  }


  static void show_usage() {
    System.err.println("Usage: java setupClient [o8UserId] [o8Passwd] [o8ConnStr]
[option] [oliteConnStr] [oliteSiteName]");
    System.err.println("");
    System.err.println("option - 'ORD' for Order entry db, 'PROC' for Process order
db");
```

```
    System.err.println("¥n¥n Example:");
    System.err.println("java setupClient AQUSER AQUSER my_machine.my_
domain.com:1521:orcl ORD C:¥¥ORANT¥¥OLDB40¥¥PoliteOrd.odb OliteSite1¥n");
    System.err.println("java setupClient AQUSER AQUSER my_machine.my_
domain.com:1521:orcl PROC C:¥¥ORANT¥¥OLDB40¥¥PoliteProc.odb OliteSite2");
    System.err.println("");
    System.exit(0);
  }

}
```

C

AQ Lite の制約

この章では、AQ Lite の機能上の制約をリストします。説明する内容は、次のとおりです。

- キュー・プロパティの制約
- API の制約

キュー・プロパティの制約

- AQ Lite には例外キューはありません。
- AQ Lite は自由なキュー挿入をサポートしません。順序逸脱はありません。
- 保存期間は 0。DEQUEUE_REMOVE モードでデキューした場合、影響を受けるキューのメッセージは完全に削除されます（キュー・テーブルに残りません）。
- AQ Lite は、エンキュおよびデキュー操作の履歴記録をサポートしません。
- キューに追加されたサブスクリプションで、キューの既存のメッセージが更新されます。
- AQ Lite キュー内のメッセージは、Oracle8*i* サーバー上のキューにマップされている、対応するマスター・キューに対してしか伝播されません。メッセージを任意のキューに伝播できません。
- AQ Lite はエラー・カウントをサポートしません。
- RAW 型のペイロードのみサポートされます。リリース 8.1.5 に含まれる Oracle8*i* AQ Java クラスでは、オブジェクト型ペイロードのエンキュおよびデキューはできません。

API の制約

- AQ Lite のドライバは `AQLiteDriver` です。
- 「デキュー待機」オプションは常に NONE です。
- 再試行遅延は常に 0 です。
- AQ Lite はルールベースのサブスクライバをサポートしません。
- AQ Lite ではキューの `start()` および `stop()` は使用できません。
- 同一の名前を持つキューは、同一の所有者が所有する必要があります。
- AQ Lite のこのリリースでは、ADT、SQLData または StructData をサポートしません。
- Oracle Lite クライアントに常駐しているクライアント・キューまたはキュー・テーブルに対応するキューまたはキュー・テーブルを Oracle8*i* サーバーから削除するときは、ディプロイメント `QueueGroup` からキューを削除するプロシージャを実行してクライアント側を更新するか、ディプロイメント `QueueGroup` を削除するか、または単に `deInstantiate()` をコールします。
- 第 1 章「概要」の「スタンドアロン・キュー構成」で説明したように Oracle Lite データベースでのみアプリケーションを実行する場合は、フットプリントが大きくならないように、Java クラス **AQLite.jar** および **AQCommon.jar** のみを使用します。第 1 章「概要」の「マスター / クライアント・キュー構成」で説明したように Oracle Lite データベースと Oracle サーバーの両方でアプリケーションを実行する場合は、Java クラス **AQLite.jar** および **AQOracle.jar** を使用します。

索引

A

C

D

E

G

I

J

O

P

Q

R

S

U

え

き

く

こ

さ

し

す

せ

そ