Sun Java System Message Queue 4.0 リリースノート

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

Part No: 819–6960

060612@14986

目次

◆ ◆ ◆ 第 1 章

Sun Java System Message Queue 4.0 リリースノート

Version 4.0

Part Number 819-6960

このリリースノートには、Sun Java™ System Message Queue 4.0 のリリース時点で得られる重要な情報が含まれています。新しい機能や拡張機能、既知の問題と制限、およびその他の情報が記載されています。Message Queue を使用する前に、このドキュメントをよくお読みください。

このリリースノートの最新版は、Sun Java System Message Queue ドキュメント Web サイトから入手できます。ソフトウェアのインストールおよび設定前だけでなく、以後も定期的にこの Web サイトをチェックして、最新版のリリースノートやマニュアルをご覧ください。

このリリースノートには次の節があります。

このマニュアルには、その他の関連情報の参照先としてサードパーティーの URL が記載されています。

Sun は、このリリースノートに記載されたサードパーティーの Web サイトの有効性および有用性に関して責任を負いません。こうしたサイトやリソース上またはこれらを通じて利用できるコンテンツ、広告、製品、その他の資料について Sun は保証するものではなく、Sun はいかなる責任も負いません。こうしたサイトやリソース上で、またはこれら

を経由して利用できるコンテンツ、製品、サービスを利用または信頼したことに伴って発生した (あるいは発生したと主張される) いかなる損害や損失についても、Sun は直接的にも間接的にも、一切の責任を負いません。

リリースノートの改訂履歴

次の表は、Message Queue 製品のすべての 4.x リリースの日付と、各リリースに関連する主な変更内容を示しています。

表 1–1 改訂履歴

日付	変更内容の説明
2006 年 5 月	このリリースノートの初回リリース

Message Queue 4.0 について

Sun Java System Message Queue は、多くの機能を備えるメッセージサービスで、Java Messaging Specification (JMS) 1.1 に準拠する信頼性の高い非同期メッセージングを提供します。Message Queue では、JMS 仕様を超える機能も用意され大規模企業のシステム配備のニーズにも対応できるようになっています。

Message Queue 4.0 は、Application Server 9 PE のサポートに限定されたリリースです。マイナーリリースには、いくつかの新機能と機能拡張、およびバグ修正が含まれています。この節の構成は次のとおりです。

- 6 ページの「このリリースでの新機能」
- 22 ページの「ハードウェアおよびソフトウェア要件」

このリリースでの新機能

Message Queue 4.0 に含まれる新機能は次のとおりです。

- 7 ページの「C-API および C クライアントランタイムのインタフェースの変更」
- 8 ページの「持続ストアに関する情報の表示」
- 8 ページの「持続ストアの形式の変更」
- 9 ページの「ブローカ管理」
- 9 ページの「JDBC 持続サポート」
- 10 ページの「SSL サポート」
- 10 ページの「JMX サポート」
- 15 ページの「クライアントランタイムログ」
- 19 ページの「接続イベント通知」

以降の節で、これらの機能について説明しています。

注意 – version 4.0 には、軽度とはいえ、状況によって十分な対応が必要になる変更が導入されています。その 1 つとして、パスワードを指定するコマンド行オプションが使用されなくなったことが挙げられます。今後は、25 ページの「使用されなくなったパスワードオプション」で説明するように、すべてのパスワードをファイルに保存する必要があります。

C-API および C クライアントランタイムのインタフェースの変更

C-API への変更は Release 3.7 で導入されました。この変更には、新しい関数、新しいメッセージタイプ、新しいコネクションプロパティーが含まれます。

- 新しい関数: `MQGetDestinationName()`

```
MQGetDestinationName (const MQDestinationHandle destinationHandle,
                            MQString * destinationName);
```

この関数は、送信先の名前を取得するために使用します。戻り値の `destinationName` はコピーで、呼び出し元が `MQFreeString()` 関数を呼び出すことで解放を担当します。

パラメータ

destinationHandle　名前を取得したい送信先へのハンドル。

destinationName　名前の出力パラメータ。

この関数は、reply-to パターンを使用する場合に便利です。`MQGetMessageReplyTo` 関数を使用するとメッセージの送信先へのハンドルを取得できます。次に `MQGetDestinationName` を使用して、その送信先の名前を取得できます。送信先の名前を取得したら、名前に基づいてメッセージを処理できます。

- 新しい列挙値: `MQ_MESSAGE`

新しい `MQMessageType` と `MQ_MESSAGE` を使用すると、C クライアントはタイプが `Message` である JMS メッセージを、他の Message Queue クライアント (C と Java の両方) と交換することができます。

```
typedef enum _MQMessageType {MQ_TEXT_MESSAGE = 0,
                              MQ_BYTES_MESSAGE = 1,
                              MQ_MESSAGE = 3,
                              MQ_UNSUPPORTED_MESSAGE = 2} MQMessageType;
```

`MQ_MESSAGE` タイプは、ヘッダーとプロパティーはあるが、メッセージ本体が存在しないメッセージを識別します。このタイプのメッセージを作成するには、`MQCreateMessage ()` 関数を使用します。

- 新しいコネクションプロパティー `MQ_UPDATE_RELEASE_PROPERTY` は、Message Queue のインストールバージョンに対する更新リリースバージョンを指定します。`MQGetMetaData()` 関数を使用してバージョン情報を取得します。

持続ストアに関する情報の表示

imqdbmgr コマンドに query サブコマンドが追加されました。このサブコマンドは、持続ストアの情報 (ストアバージョン、データベースユーザー、データベーステーブルが作成されたかどうかなど) を表示するために使用します。

次に、このコマンドによって表示される情報の例を示します。

```
dbmgr query

[04/Oct/2005:15:30:20 PDT] Using plugged-in persistent store:
        version=400
        brokerid=Mozart1756
        database connection url=jdbc:oracle:thin:@Xhome:1521:mqdb
        database user=scott
Running in standalone mode.
Database tables have already been created.
```

持続ストアの形式の変更

Message Queue の Version 3.7 UR1 では、パフォーマンスを向上するために持続ストアの形式に 2 つの変更が加えられました。1 つはファイルストアに対する変更で、もう 1 つは JDBC ストアに対する変更です。

- ファイルストア内で持続されるトランザクションデータの形式

  ディスク I/O を軽減し、JMS トランザクションのパフォーマンスを向上させるために、Message Queue ファイルベースの持続ストア内に格納されているトランザクション状態情報の形式が変更されました。

- Oracle JDBC ストア

  以前のバージョンの Message Queue では、Oracle のストアスキーマでは、メッセージデータを格納するために LONG RAW データ型が使用されていました。Oracle 8 では、Oracle によって BLOB データ型が導入され、LONG RAW 型は使用されなくなりました。Message Queue 3.7 UR1 では、パフォーマンスとサポート性を向上するために、データ型が BLOB に変更されました。

これらの変更がストアの互換性にも影響し、Message Queue version 3.7 UR1 では、ファイルストアと JDBC ストアの両方のストアバージョンが 350 から 370 に変更されました。

Message Queue version 4.0 では、最適化と将来の機能拡張をサポートするために、JDBC ストアが変更されました。このため、JDBC ストアのバージョンが 400 になりました。ただし Version 4.0 では、ファイルベースの持続ストアには変更が行われていないため、このファイルストアのバージョンは 370 のままです。

Message Queue 4.0 は、ファイルベースの持続ストアと JDBC の持続ストアの最新バージョンへの持続ストアの自動変換をサポートしています。imqbrokerd を最初に起動した

ときに、ユーティリティーが古いバージョンのストアを検出した場合、古いストアはそのままで、ストアを新しい形式に移行します。

- ファイルベースのストアのバージョン 200 および 350 は、バージョン 370 の形式に移行されます。
- JDBC ストアのバージョン 350 および 370 は、バージョン 400 の形式に移行されます。バージョン 200 ストアをアップグレードする必要がある場合は、いったん中間的な release 3.5 または 3. 6 にアップグレードする必要があります。

このアップグレードをロールバックする必要のある場合は、Message Queue 4.0 をいったんアンインストールして、以前実行していたバージョンを再インストールします。ストアの古いコピーはそのまま残っているので、ブローカはストアの古いコピーを実行できます。

ブローカ管理

コマンドユーティリティー (`imqcmd`) に 1 つのサブコマンドと複数のオプションが追加され、管理者がブローカを休止したり、指定した間隔の後ブローカをシャットダウンしたり、接続を破棄できるようになりました。

- ブローカを休止すると休止状態になり、ブローカをシャットダウンまたは再起動する前に、メッセージを排出してしまうことができます。休止状態にあるブローカに新しく接続が作成されることはありません。ブローカを休止するには、次のようなコマンドを入力します。

  ```
  imqcmd quiesce bkr -b Wolfgang:1756
  ```

- 指定した間隔の後でブローカをシャットダウンするには、次のようなコマンドを入力します。時間間隔は、ブローカをシャットダウンするまでの待機を秒数で指定します。

  ```
  imqcmd shutdown bkr -b Hastings:1066 -time 90
  ```

- 接続を破棄するには、次のようなコマンドを入力します。

  ```
  imqcmd destroy cxn -n 2691475382197166336
  ```

  コマンド `imqcmd list cxn` または `imqcmd query cxn` を使用してコネクション ID を取得します。

`imqcmd` コマンドの構文の詳細は、『Sun Java System Message Queue 3 2005Q4 管理ガイド』の第 14 章「コマンド行のリファレンス」を参照してください。

JDBC 持続サポート

Apache Derby Version 10.1.1 が、JDBC 準拠の持続ストアプロバイダとしてサポートされるようになりました。

SSL サポート

release 4.0 から、クライアントコネクションファクトリのプロパティー `imqSSLIsHostTrusted` のデフォルト値が `false` になりました。使用しているアプリケーションが以前のデフォルト値の `true` に依存している場合は、再設定を行い、プロパティーを明示的に `true` に設定する必要があります。

JMX サポート

Java Management Extensions (JMX) 仕様に準拠して、Message Queue ブローカを設定および監視するための新しい API が追加されました。この API を使用すると、Message Queue クライアントアプリケーション内部からプログラムによってブローカ関数を設定および監視することができます。以前のバージョンの Message Queue では、これらの関数にはコマンド行または管理コンソールからしかアクセスできませんでした。

この API は、一連の JMX *Managed Bean (MBean)* によって設定されており、次の Message Queue 関連のリソースを管理します。

- メッセージブローカ
- 接続サービス
- 接続
- 送信先
- メッセージプロデューサ
- メッセージコンシューマ
- トランザクション
- ブローカクラスタ
- ログ
- Java Virtual Machine (JVM)

これらの MBean によって、基礎となるリソースの状態を、同期をとりながらポーリングおよび操作するための属性と操作が設定されます。また、状態が変更されたときに、クライアントアプリケーションが非同期で待機して応答できるようにする通知も設定されます。JMX API を使用して、クライアントアプリケーションは次のようなタスクを設定および監視することができます。

- ブローカのポート番号を設定する
- ブローカのメッセージの最大サイズを設定する
- 接続サービスを一時停止する
- 接続サービスの最大スレッド数を設定する
- サービス上の現在の接続数を取得する
- 接続を破棄する
- 送信先を作成する
- 送信先を破棄する
- 送信先の自動作成を有効または無効にする
- 送信先からすべてのメッセージを削除する
- ブローカの起動後に送信先の受信したメッセージの累積数を取得する
- キューの現在の状態 (実行中または一時停止) を取得する

- トピックのメッセージプロデューサの現在数を取得する
- 永続サブスクライバからすべてのメッセージを削除する
- 現在の JVM ヒープサイズを取得する

JMX API の紹介と詳細情報については、『Sun Java System Message Queue 4.0 Developer's Guide for JMX Clients』を参照してください。

ブローカサポート: JMX 関連のプロパティー

JMX API をサポートするために、いくつかの新しいブローカプロパティーが追加されました (表 1–2 を参照)。これらのプロパティーはいずれも Message Queue コマンドユーティリティー (imqcmd) によるコマンド行からは設定できません。その代わりに、ブローカユーティリティー (imqbrokerd) の -D オプションを使用するか、ブローカのインスタンス設定ファイル (config.properties) を手作業で編集することで設定できます。さらに、これらのプロパティーのいくつか (imq.jmx.rmiregistry.start、imq.jmx.rmiregistry.use、 imq.jmx.rmiregistry.port) は、表 1–3 に示されている新しいブローカユーティリティーオプションを使用して設定できます。次の表は、各オプションとその型、それぞれの使用方法について示しています。

表 1–2 JMX サポートのための新しいブローカプロパティー

プロパティー	型	説明
imq.jmx.rmiregistry.start	ブール型	ブローカの起動時に RMI レジストリを起動しますか ? true の場合、ブローカは imq.jmx.rmiregistry.port によって指定されたポートで RMI レジストリを起動し、これを使用して JMX コネクタ用の RMI スタブを格納します。この場合、imq.jmx.rmiregistry.use の値は無視されます。 デフォルト値: false
imq.jmx.rmiregistry.use	ブール型	外部の RMI レジストリを使用しますか ? imq.jmx.rmiregistry.start が false の場合のみ適用されます。 true の場合、ブローカは imq.jmx.rmiregistry.port によって指定されたポートで外部の RMI レジストリを使用し、JMX コネクタ用の RMI スタブを格納します。この外部の RMI レジストリは、ブローカの起動時にすでに実行されている必要があります。 デフォルト値: false

表 1–2 JMX サポートのための新しいブローカプロパティー　(続き)

プロパティー	型	説明
`imq.jmx.rmiregistry.port`	整数	RMI レジストリのポート番号 `imq.jmx.rmiregistry.start` または `imq.jmx.rmiregistry.use` が `true` の場合のみ適用されます。このポート番号を JMX サービス URL の URL パスに追加することによって、JMX コネクタが RMI レジストリを使用するように設定できます。 デフォルト値: `1099`
`imq.jmx.connector.list`	文字列	事前設定された JMX コネクタの名前。コンマで区切られます。 デフォルト値: `jmxrmi,ssljmxrmi`
`imq.jmx.connector.activelist`	文字列	ブローカの起動時にアクティブになる JMX コネクタの名前。コンマで区切られます。 デフォルト値: `jmxrmi`
`imq.jmx.connector.`*connectorName* `.urlpath`	文字列	コネクタ *connectorName* の、JMX サービス URL の *urlPath* コンポーネント JMX サービス URL パスが明示的に指定されている場合 (共有の外部 RMI レジストリが使用されている場合など) に有効です。 デフォルト値: JMX コネクタ用の RMI スタブを格納するために RMI レジストリが使用されている場合 (つまり `imq.jmx.registry.start` または `imq.jmx.registry.use` が `true` の場合): `/jndi/rmi://`*brokerHost*`:`*rmiPort* `/`*brokerHost*`/`*brokerPort*`/`*connectorName* RMI レジストリが使用されていない場合 (デフォルトの場合。つまり、 `imq.jmx.registry.start` と `imq.jmx.registry.use` の両方が `false` の場合): `/stub/`*rmiStub* ここで *rmiStub* は、RMI スタブそのものをエンコードおよび直列化した表現です。
`imq.jmx.connector.`*connectorName* `.useSSL`	ブール型	コネクタ *connectorName* に対して SSL (Secure Socket Layer) を使用しますか？ デフォルト値: `false`

表 1–2 JMX サポートのための新しいブローカプロパティー　(続き)

プロパティー	型	説明
imq.jmx.connector.*connectorName* .brokerHostTrusted	ブール型	コネクタ *connectorName* 用のブローカによって提示された証明書をすべて信頼しますか？ imq.jmx.connector. *connectorName*.useSSL が true の場合のみ適用されます。 false の場合、Message Queue クライアントランタイムは提示されたすべての証明書を検証します。証明書の署名者がクライアントのトラストストア内に存在しない場合、検証は失敗します。 true の場合、証明書の検証はスキップされます。これは、ソフトウェアのテスト中で自己署名した証明書が使用されている場合などに便利です。 デフォルト値: false

imq.jmx.connector.list プロパティーは、ブローカの起動時に作成される一連の名前付き JMX コネクタを定義します。imq.jmx.connector.activelist では、これらの中でアクティブにするものを指定します。名前付きのコネクタには、それぞれ独自のプロパティーセットがあります。

imq.jmx.connector.*connectorName* .urlpath
imq.jmx.connector.*connectorName* .useSSL
imq.jmx.connector.*connectorName* .brokerHostTrusted

デフォルトでは、 jmxrmi および ssljmxrmi という名前の 2 つの JMX コネクタが作成されます。前者は SSL 暗号化を使用しないように設定 (imq.jmx.connector.jmxrmi.useSSL = false) され、後者は使用するように設定 (imq.jmx.connector.ssljmxrmi.useSSL = true) されています。デフォルトでは、ブローカの起動時には jmxrmi コネクタのみがアクティブになります。通信のセキュリティー保護のために ssljmxrmi コネクタをアクティブにする方法については、14 ページの「JMX クライアント用の SSL サポート」を参照してください。

使いやすくするために、コマンド行のブローカユーティリティー (imqbrokerd) にも、RMI レジストリの使用方法、起動、ポートを制御するための新しいオプション (表 1–3) が追加されています。これらのオプションの使用方法と効果は、表 1–2 に示されているブローカプロパティーの中の対応するものと同じです。次の表では、各オプションとともに対応するブローカプロパティーを示し、その使用方法についても説明します。

表 1–3 JMX サポートのための新しいブローカユーティリティーオプション

オプション	対応するブローカプロパティー	説明
`-startRmiRegistry`	`imq.jmx.rmiregistry.start`	ブローカの起動時に RMI レジストリを起動しますか？
`-useRmiRegistry`	`imq.jmx.rmiregistry.use`	外部の RMI レジストリを使用しますか？
`-rmiRegistryPort`	`imq.jmx.rmiregistry.port`	RMI レジストリのポート番号

ブローカの起動時に作成および起動される JMX コネクタの JMX サービス URL を表示するために、コマンド行のコマンドユーティリティー (`imqcmd`) に新しいサブコマンド (表 1–4) が追加されました。この情報は、JMX コネクタを取得するために Message Queue の簡易クラス `AdminConnectionFactory` を使用しない JMX クライアントにとって必要になります。また、Java Monitoring and Management Console (`jconsole`) などの汎用の JMX ブラウザを介して Message Queue を管理または監視するために使用することもできます。

表 1–4 新しいコマンドユーティリティーのサブコマンド

サブコマンド	説明
`list jmx`	JMX コネクタの JMX サービス URL を表示します

JMX クライアント用の SSL サポート

これまでに説明したように、Message Queue のメッセージブローカは、デフォルトでは、事前に設定済みの JMX コネクタ `jmxrmi` を使用して、セキュリティー保護されていない通信用に設定されています。通信のセキュリティー保護のために SSL (Secure Socket Layer) を使用する必要のあるアプリケーションでは、代わりにセキュリティー保護された JMX コネクタである `ssljmxrmi` をアクティブにする必要があります。このためには次の手順を実行する必要があります。

1. 『Message Queue 管理ガイド』に示されている `ssljms`、`ssladmin`、または `cluster` 接続サービスと同じ方法で、署名済みの証明書を取得およびインストールします。
2. 必要に応じて、ルート CA 証明書をトラストストア内にインストールします。
3. ブローカの起動時にアクティブになるように、JMX コネクタのリストに `ssljmxrmi` コネクタを追加します。

   ```
   imq.jmx.connector.activelist=jmxrmi,ssljmxrmi
   ```

4. Message Queue ブローカユーティリティー (`imqbrokerd`) をパスワードファイル内のキーストアパスワードに渡すか、プロンプト表示されるコマンド行から入力して、ブローカを起動します。

5. デフォルトでは、ssljmxrmi コネクタ、またはその他の SSL ベースのコネクタは、提示されるすべてのブローカ SSL 証明書を検証するように設定されています。この検証を回避する場合、ソフトウェアのテスト中で自己署名の証明書を使用している場合などは、ブローカプロパティー imq.jmx.connector.ssljmxrmi.brokerHostTrusted を true に設定します。

クライアント側では、ssljmxrmi を優先コネクタとして指定した URL を使用して管理者のコネクションファクトリ (AdminConnectionFactory) を設定する必要があります。

```
AdminConnectionFactory  acf = new AdminConnectionFactory();
acf.setProperty(AdminConnectionConfiguration.imqAddress, "mq://myhost:7676/ssljmxrmi");
```

必要に応じて、システムプロパティー javax.net.ssl.trustStore および javax.net.ssl.trustStorePassword を使用して、JMX クライアントをトラストストアにポイントします。

クライアントランタイムログ

この節では、Message Queue 4.0 による接続のクライアントランタイムログのサポートと、セッション関連のイベントについて説明します。

JDK 1.4 以上には、java.util.logging ライブラリが含まれています。このライブラリは、アプリケーション固有のログに使用される標準のロガーインタフェースを実装しています。

Message Queue のクライアントランタイムは Java Logging API を使用してログ機能を実装します。ログ活動を設定するために、すべての J2SE 1.4 ロギング機能を使用できます。たとえば、Message Queue クライアントランタイムがログ情報を出力する方法を設定するために、アプリケーションは次の Java ロギング機能を使用することができます。

- ログハンドラ
- ログフィルタ
- ログフォーマッタ
- ログレベル

Java Logging API の詳細は、http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html にある Java ロギングの概要を参照してください。

ログの名前空間、レベル、活動

Message Queue プロバイダは、ログレベルやログ活動に関連付けて一連のログの名前空間を定義します。Message Queue クライアントは、ログ設定が適切であれば、この名前空間を使用して接続やセッションイベントをログに記録することができます。

Message Queue クライアントランタイムのルートのログ名前空間は、javax.jms と定義されます。Message Queue クライアントランタイム内のすべてのロガーが、この名前を親の名前空間として使用します。

Message Queue クライアントランタイムに使用されるログレベルは、`java.util.logging.Level` クラス内で定義されるものと同じです。このクラスでは、7 つの標準ログレベルと、ログのオン/オフに使用できる 2 つの追加設定が定義されます。

`OFF`	ログをオフにします。
`SEVERE`	優先順位が最高である最高値。アプリケーションで定義します。
`WARNING`	アプリケーションで定義します。
`INFO`	アプリケーションで定義します。
`CONFIG`	アプリケーションで定義します。
`FINE`	アプリケーションで定義します。
`FINER`	アプリケーションで定義します。
`FINEST`	優先順位が最低である最低値。アプリケーションで定義します。
`ALL`	すべてのメッセージのログを有効にします。

一般に、Message Queue クライアントランタイム内で発生した例外やエラーは、名前空間 `javax.jms` を使用するロガーによってログ記録されます。

- JVM からスローされクライアントランタイムによってキャッチされる例外 (`IOException` など) は、ログ名前空間 `javax.jms` を使用するロガーによって `WARNING` レベルでログ記録されます。
- クライアントランタイムからスローされる JMS 例外 (`IllegalStateException` など) は、ログ名前空間 `javax.jms` を使用するロガーによって `FINER` レベルでログ記録されます。
- JVM からスローされクライアントランタイムによってキャッチされるエラー (`OutOfMemoryError` など) は、ログ名前空間 `javax.jms` を使用するロガーによって `SEVERE` レベルでログ記録されます。

以下の表は、JMS コネクションとセッションについて、ログ記録できるイベントとログイベントに対して設定する必要のあるログレベルを示しています。

次の表は、コネクションのログレベルとイベントについて示したものです。

表 1–5 名前空間 `javax.jms.connection` のログレベルとイベント

ログレベル	イベント
`FINE`	接続が作成されました
`FINE`	接続が起動しました
`FINE`	接続が閉じました
`FINE`	接続が破棄されました

表 1–5 名前空間 javax.jms.connection のログレベルとイベント　(続き)

ログレベル	イベント
FINE	再接続されました
FINER	その他の接続アクティビティー (setClientID など)
FINEST	メッセージ、通知、Message Queue アクション、制御メッセージ (トランザクションのコミットなど)

セッションの場合、次の情報がログレコードに記録されます。

- コンシューマに配信されるメッセージの各ログレコードには、ConnectionID、SessionID、ConsumerID が含まれています。
- プロデューサによって送信されるメッセージの各ログレコードには、ConnectionID、SessionID、ProducerID、送信先名が含まれています。

次の表は、セッションのログレベルとイベントについて示したものです。

表 1–6 名前空間 javax.jms.session のログレベルとイベント

ログレベル	イベント
FINE	セッションが作成されました
FINE	セッションが閉じました
FINE	プロデューサが作成されました
FINE	コンシューマが作成されました
FINE	送信先が作成されました
FINER	その他のセッションアクティビティー (セッションのコミットなど)。
FINEST	メッセージがプロデュースされ、消費されました。(メッセージのプロパティーと本文はログレコードに記録されません。)

デフォルトでは、出力ログレベルはアプリケーションの実行されている JRE から継承されます。JRE_DIRECTORY/lib/logging.properties ファイルを参照してレベルを確認してください。

ログはプログラムや設定ファイルを使用して設定できます。また、ログの発生する範囲を制御することもできます。次の節では、これらの機能について説明します。

JRE ロギング設定ファイルの使用

次の例は、JRE_DIRECTORY/lib/logging.properties ファイル内でログの名前空間とログレベルを設定する方法を示しています。この方法は Java ランタイム環境のログレベルの設

定に使用されます。JRE を使用するすべてのアプリケーションが同じログ設定になります。次に示すサンプル設定では、名前空間 javax.jms.connection のログレベルを INFO に設定し、その出力が java.util.logging.ConsoleHandler に書き込まれるように指定しています。

```
#logging.properties file.
# "handlers" specifies a comma separated list of log Handler
# classes. These handlers will be installed during VM startup.
# Note that these classes must be on the system classpath.
# By default we only configure a ConsoleHandler, which will only
# show messages at the INFO and above levels.

    handlers= java.util.logging.ConsoleHandler

# Default global logging level.
# This specifies which kinds of events are logged across
# all loggers. For any given facility this global level
# can be overriden by a facility-specific level.
# Note that the ConsoleHandler also has a separate level
# setting to limit messages printed to the console.

    .level= INFO

# Limit the messages that are printed on the console to INFO and above.

    java.util.logging.ConsoleHandler.level = INFO
    java.util.logging.ConsoleHandler.formatter =
                                    java.util.logging.SimpleFormatter

# The logger with javax.jms.connection name space will write
# Level.INFO messages to its output handler(s). In this configuration
# the ouput handler is set to java.util.logging.ConsoleHandler.

    javax.jms.connection.level = INFO
```

特定のアプリケーションに対するログ設定ファイルの使用

アプリケーションの実行に使用する Java コマンド行からログ設定ファイルを定義することもできます。このアプリケーションは、指定のログファイル内で定義された設定を使用します。次の例では、configFile は JRE_DIRECTORY/lib/logging.properties ファイル内で定義されているものと同じ形式を使用します。

```
java -Djava.util.logging.config.file=configFile MQApplication
```

プログラムによるログ設定

次のコードでは、java.util.logging API を使用して、javax.jms.connection 名前空間のログレベルを FINE に変更することで、接続イベントをログ記録しています。このようなコードをアプリケーションに追加して、プログラムによるログ設定を行うことができます。

```
import java.util.logging.*;
//construct a file handler and output to the mq.log file
//in the system's temp directory.

    Handler fh = new FileHandler("%t/mq.log");
    fh.setLevel (Level.FINE);

//Get Logger for "javax.jms.connection" domain.

    Logger logger = Logger.getLogger("javax.jms.connection");
    logger.addHandler (fh);

//javax.jms.connection logger would log activities
//with level FINE and above.

    logger.setLevel (Level.FINE);
```

接続イベント通知

接続イベント通知を使用すると、Message Queue クライアントは接続のクローズおよび再接続イベントを待機して、通知タイプと接続状態に基づいて適切なアクションを起こすことができます。たとえば、フェイルオーバーが発生してクライアントが別のブローカに再接続された場合、アプリケーションはそのトランザクションの状態をクリーンアップしてから新しいトランザクションに進む必要があるかもしれません。

Message Queue プロバイダは、接続について深刻な問題を検出した場合、接続オブジェクトの登録済みの例外リスナーを呼び出します。この呼び出しは、リスナーの onException メソッドを呼び出し、問題を記述する JMSException 引数にこれを渡すことで実行されます。同様に、Message Queue プロバイダはイベント通知 API を提供し、これを使用してクライアントランタイムは、接続状態の変更をアプリケーションに通知できます。通知 API は次の要素によって定義されます。

- com.sun.messaging.jms.notification パッケージ。イベントリスナーと通知イベントオブジェクトを定義します。
- com.sun.messaging.jms.Connection インタフェース。javax.jms.Connection インタフェースへのエクステンションを定義します。

次の節では、通知をトリガーできるイベントと、イベントリスナーの作成方法について説明します。

接続イベント

次の表は、イベントリスナーによって返されるイベントについて説明したものです。

接続イベントの発生時に JMS 例外リスナーは呼び出されません。例外リスナーが呼び出されるのは、クライアントランタイムの再接続の試行回数が上限に達してしまった場合のみです。クライアントランタイムは、常に例外リスナーの前にイベントリスナーを呼び出します。

表 1–7 通知イベント

イベントタイプ	意味
`ConnectionClosingEvent`	Message Queue クライアントランタイムは、管理者のシャットダウン要求によって接続がクローズされようとしているという通知をブローカから受信したときに、このイベントを生成します。
`ConnectionClosedEvent`	Message Queue クライアントランタイムは、ブローカのエラーによって接続がクローズされたか、管理者のシャットダウン要求または再起動要求によって接続がクローズされたときに、このイベントを生成します。 イベントリスナーが `ConnectionClosedEvent` を受信すると、アプリケーションは受信したイベントの `getEventCode()` メソッドを使用して、クローズの原因を指定するイベントコードを取得します。
`ConnectionReconnectedEvent`	Message Queue クライアントランタイムがブローカに再接続されました。このブローカは、このクライアントが以前接続していたものと同じ場合もありますが、別のブローカの場合もあります。 アプリケーションは、受信したイベントの `getBrokerAddress` メソッドを使用して、再接続先のブローカのアドレスを取得することができます。

表 1–7 通知イベント　(続き)

イベントタイプ	意味
ConnectionReconnectFailedEvent	Message Queue クライアントランタイムがブローカへの再接続に失敗しました。再接続の試みに失敗するたびに、ランタイムは新しいイベントを生成し、それをイベントリスナーに配信します。 接続イベントの発生時に JMS 例外リスナーは呼び出されません。呼び出されるのは、クライアントランタイムの再接続の試行回数が上限に達してしまった場合のみです。クライアントランタイムは、常に例外リスナーの前にイベントリスナーを呼び出します。

イベントリスナーの作成

次のコード例は、接続イベントリスナーの設定方法を示したものです。接続イベントが発生するたびに、イベントリスナーの onEvent メソッドがクライアントランタイムによって呼び出されます。

```
//create an MQ connection factory.

com.sun.messaging.ConnectionFactory factory =
        new com.sun.messaging.ConnectionFactory();

//create an MQ connection.

com.sun.messaging.jms.Connection connection =
       (com.sun.messaging.jms.Connection )factory.createConnection();

//construct an MQ event listener.  The listener implements
//com.sun.messaging.jms.notification.EventListener interface.

com.sun.messaging.jms.notification.EventListener eListener =
       new ApplicationEventListener();

//set event listener to the MQ connection.

connection.setEventListener ( eListener );
```

イベントリスナーの例

この例では、アプリケーションが、そのイベントリスナーが接続イベントをアプリケーションのロギングシステムに記録するように選択します。

```
public class ApplicationEventListener implements
               com.sun.messaging.jms.notification.EventListener {
```

```
public void onEvent ( com.sun.messaging.jms.notification.Event connEvent ) {
          log (connEvent);
}
private void log ( com.sun.messaging.jms.notification.Event connEvent ) {
           String eventCode = connEvent.getEventCode();
          String eventMessage = connEvent.getEventMessage();
          //write event information to the output stream.
         }
}
```

ハードウェアおよびソフトウェア要件

ハードウェアおよびソフトウェア要件は、『Sun Java System Application Server Platform Edition 9 Release Notes』の「Hardware and Software Requirements」で指定されています。

このリリースで修正されたバグ

次の表では、Message Queue 4.0 で修正されたバグを説明しています。

表 1–8 Message Queue 4.0 で修正されたバグ

バグの番号	説明
4986481	Message Queue 3.5 では、自動再接続モードで `Session.recover` を呼び出すとハングしてしまうことがあります。
4987325	Session.recover の呼び出し後、再配信されたメッセージに対する再配信されたフラグが `false` に設定されました。
6157073	新しく接続メッセージが変更され、接続の合計数のほかにサービス上の接続数も表示されるようになりました。
6193884	Message Queue は、C 以外のロケールで文字化けしたメッセージを syslog に出力します。
6196233	`JMSMessageID` を使用したメッセージ選択が機能しません。
6251450	クラスタのシャットダウン時、 `connectList` に `ConcurrentModificationException` が発生します。
6252763	`java.nio.HeapByteBuffer.putLong/Int` に `java.nio.BufferOverflowException` が発生します。
6260076	Oracle ストレージを使用すると、起動後に発行される最初のメッセージが遅くなります。

表 1–8 Message Queue 4.0 で修正されたバグ　(続き)

バグの番号	説明
6264003	キューブラウザにコミットされていないメッセージが表示されます。
6260814	`JMSXUserID` 上のセレクタ処理が、常に `false` と評価されてしまいます。
6264003	キューブラウザにコミットされていないメッセージが表示されます。
6271876	消費されていないメッセージを含むコンシューマを閉じようとすると、接続フロー制御が正しく動作しなくなります。
6279833	Message Queue では、2 つのブローカが同じ JDBC テーブルを使用することはできません。
6293053	システム IP アドレスが変更された場合、`–reset store` を使用してストアをクリアしていないと、マスターブローカが正しく起動しません。
6294767	Message Queue ブローカがネットワークソケットを開く場合、そのネットワークソケット上に `SO_REUSEADDR` を設定する必要があります。
6304949	`TopicConnectionFactory` に `ClientID` プロパティーを設定することができません。
6307056	txn ログがパフォーマンス上のボトルネックになっています。
6320138	Message Queue C API では reply-to ヘッダーからキューの名前を決定することができません。
6320325	Solaris 上に JDK 1.4 と JDK 1.5 の両方がインストールされている場合、ブローカが JDK 1.5 より前に JDK 1.4 を検出してしまうことがあります。
6321117	マルチブローカクラスタの初期化で `java.lang.NullPointerException` が発生します。
6330053	サブスクライバからトランザクションをコミットしている場合、JMS クライアントで `java.lang.NoClassDefFoundError` が発生します。
6340250	C-API で `MESSAGE` タイプをサポート。
6351293	Apache Derby データベースへのサポートを追加。

重要な情報

この節には主な製品のドキュメントに含まれていない最新の情報が含まれています。この節で説明する項目は次のとおりです。

互換性の問題

この節では、Message Queue 4.0 の互換性の問題を説明しています。

インタフェースの安定性

Sun Java System Message Queue で使用される多くのインタフェースは、時間の経過につれて変更される可能性があります。『Sun Java System Message Queue 3 2005Q4 管理ガイド』の第 21 章では、インタフェースをそれらの安定性に従って分類しています。インタフェースの安定性が高いほど、後続バージョンの製品で変更される可能性が低くなります。

Message Queue の次回のメジャーリリースに関係する問題

Message Queue の次回のメジャーリリースでは、クライアントがこのリリースとの互換性がなくなるような変更が導入される可能性があります。この情報は、このような変更に備えていただく目的で今回提供しています。

- Sun Java System Message Queue の一部としてインストールされる各ファイルの場所が変更される可能性があります。これによって、特定の Message Queue ファイルの現在の場所に依存する既存のアプリケーションの動作が中断する可能性があります。
- 3.5 以前のブローカは、これより新しいブローカのクラスタ内では動作できなくなる可能性があります。
- 今後のリリースでは、Message Queue クライアントは 1.3 より前のバージョンの JDK を使用できなくなる可能性があります。

Message Queue 4.0 に関するマニュアルの更新

この『リリースノート』のほかに、Message Queue 4.0 にはもう 1 つ新しいマニュアルが追加されています。それは、『Sun Java System Message Queue 4.0 Developer's Guide for JMX Clients』です。

Message Queue 3.6 SP3, 2005Q4 用に発行された Message Queue のマニュアルが、Application Server 9 PE クライアントの要件に関連して更新されました。このマニュアルセットは次の場所で入手できます。

http://docs.sun.com/app/docs/coll/1307.1

既知の問題および制約

この節には、Message Queue 4.0 の既知の問題についてのリストが含まれています。製品に関する次のトピックが記載されています。

現時点のバグ、その状態、および回避策の一覧については、Java Developer Connection™ メンバーは、Java Developer Connection Web サイトの「Bug Parade」ページを参照してください。新しいバグを報告する前に、このページをチェックしてください。すべての Message Queue バグがリストされているわけではありませんが、このページはある問題が報告済みかどうかを知りたい場合に活用できます。

`http://bugs.sun.com/bugdatabase/index.jsp`

注 – Java Developer Connection のメンバーになるのは無料ですが、登録が必要です。Java Developer Connection のメンバーになる方法についての詳細は、Sun の「For Developers」Web ページを参照してください。

新しいバグの報告や機能に関する要求を行うには、`imq-feedback@sun.com` 宛てにメールを送信してください。

使用されなくなったパスワードオプション

以前のバージョンの Message Queue では、`–p` または `–password` オプションを使用して、次のようなコマンドのパスワードを対話形式で指定することができました。`imqcmd`、`imqbrokerd`、`imdbmgr` 今回のバージョンの 4.0 から、これらのオプションは使用できなくなりました。パスワードを指定するには次の手順に従います。

1. パスワードのみを格納するために使用するファイルで、適切なプロパティーを必要なパスワード値に設定します。

   パスワードファイル内で、次の構文を使用してパスワードを指定します。

   PasswordPropertyName =MyPassword

2. `–passfile` オプションを使用してパスワードファイルの名前を渡します。

パスワードファイルには、次に示すパスワードを 1 つ以上格納することができます。

- SSL キーストアを開くために使用するキーストアパスワード。このパスワードを指定するには `imq.keystore.password` プロパティーを使用します。
- 接続が匿名でない場合に LDAP ディレクトリとの接続のセキュリティーを確保するために使用する LDAP リポジトリパスワード。このパスワードを指定するには `imq.user_repository.ldap.password` プロパティーを使用します。

- JDBC 準拠のデータベースとの接続に使用する JDBC データベースパスワード。このパスワードを指定するには imq.persist.jdbc.password プロパティーを使用します。
- imqcmd コマンド (ブローカ管理タスクを実行する) に対するパスワード。このパスワードを指定するには imq.imqcmd.password プロパティーを使用します。

次の例では、JDBC データベースに対するパスワードに abracadabra を設定しています。

```
imq.persist.jdbc.password=abracadabra
```

ブローカがユーザー作成のパスワードファイルを使用するように設定するには、次のいずれかの方法を実行します。

- ブローカの config.properties ファイルに次のプロパティーを設定する。

  ```
  imq.passfile.enabled=true
  imq.passfile.dirpath=MyFileDirectory
  imq.passfile.name=MyPassfileName
  ```

- imqbrokerd コマンドの –passfile オプションを使用する。

  ```
  imqbrokerd –passfile MyPassfileName
  ```

一般的な問題

この節では Message Queue 4.0 の一般的な問題を説明しています。中には以前のバージョンの Message Queue で発生した問題もあります。

次に示す問題は Message Queue 4.0 製品のどちらの版にも影響します。

- SOAP クライアント。これまでは、mail.jar および mail.jar を参照するために使用する SAAJ 1.2 実装 JAR が CLASSPATH 内に存在する必要はありませんでした。SAAJ 1.3 では、この参照が削除されたため、Message Queue クライアントが mail.jar を明示的に CLASSPATH 内に配置する必要があります。
- Message Queue 4.0 では、LDAP サーバーをユーザーリポジトリとして使用するためのブローカ設定の例が、config.properties ファイルのコメント領域に用意されています。default.properties ファイルにある LDAP ユーザーリポジトリの例は、コメントアウトされています。

  従来、default.properties ファイルにある、LDAP ユーザーリポジトリプロパティー設定例のプロパティー値に依存していた場合、JMS アプリケーションクライアントは、JMS 接続を作成しようとするとセキュリティーの例外を受け取ります。これは、Message Queue 4.0 へのアップグレード後に発生します。

  JMS クライアントが Message Queue 4.0 ブローカへの接続を作成しようとすると、ブローカログでエラーとなり、JMS クライアントは次の例外を受け取ります。

  ```
  SecurityException.
  20/Aug/2004:11:16:41 PDT] ERROR [B4064]: Ldap repository ldap property
  .uidattr not defined for authentication type
  ```

```
basic:com.sun.messaging.jmq.auth.LoginException:
[B4064]: Ldap repository ldap property .uidattr not defined
for authentication type basic
```

回避策 『Sun Java System Message Queue 3 2005Q4 管理ガイド』の第 8 章にある指示に従い、ブローカプロパティー `imq.user_repository.ldap.uidattr` を設定します。

JMX の問題

次に示す問題は JMX API の使用に関係するものです。

- Windows プラットフォームでは、トランザクションマネージャーの監視 MBean の `getTransactionInfo` メソッドが不正なトランザクションの作成時間を含むトランザクション情報を返します (バグ *ID 6393359*)。

  回避策 代わりにトランザクションマネージャーの監視 MBean の `getTransactionInfoByID` メソッドを使用します。

- 送信先監視 MBean の `getActiveConsumerIDs` メソッドをトピック送信先と一緒に使用すると例外をスローします (バグ *ID 6397184*)。

  回避策 代わりに送信先監視 MBean の `getConsumerIDs` メソッドを使用します。

管理および設定上の問題

次に示す問題は Message Queue の管理および設定に関係するものです。

- Windows コンピュータで `CLASSPATH` に二重引用符が含まれている場合、`imqadmin` および `imqobjmgr` ユーティリティーはエラーをスローします (バグ *ID 5060769*)。

  回避策 エラーメッセージは無視できます。ブローカはコンシューマへのエラーの通知を正しく処理します。このエラーは、システムの信頼性には影響を与えません。

- すべての Solaris および Windows スクリプトで、`-javahome` オプションの値に空白文字が含まれると動作しない (バグ *ID 4683029*)。

  `javahome` オプションは Message Queue コマンドおよびユーティリティーで使用し、使用する代替の Java 2 互換のランタイムを指定します。ただし、代替の Java 2 互換のランタイムへのパスには、空白文字を含めることはできません。空白文字を含むパスの例は次のとおりです。

  Windows の場合: `C:/jdk 1.4`

  Solaris の場合: `/work/java 1.4`

  回避策 Java ランタイムを、空白文字が含まれない場所またはパスにインストールします。

- `imqQueueBrowserMaxMessagesPerRetrieve` 属性は、クライアントランタイムがキュー送信先の内容を検索するときに一度に取得することのできるメッセージの最大数を指定します。ただし、クライアントアプリケーションは常にキューにあるすべてのメッセージを取得します。このため `imqQueueBrowserMaxMessagesPerRetrieve` 属性は、

キューに入れられたメッセージがチャンクされ、クライアントランタイムに配信される方法 (より少ない大型チャンク、またはより多い小型チャンク) には影響しますが、参照されるメッセージの総数には影響しません。この属性の値を変更するとパフォーマンスに影響が出る可能性はありますが、クライアントアプリケーションの取得するデータ量が変動することはありません (バグ *ID 6387631*)。

ブローカの問題

次に示す問題は Message Queue ブローカに影響します。

- Ctrl-C を使用してブローカをシャットダウンする場合、ストアが閉じられたあとにトランザクションがクリーンアップされることがある (バグ *ID 49344466*)。

  メッセージまたはトランザクションの処理中にブローカがシャットダウンされた場合、「ストアが閉じられた後に、ストアメソッドがアクセスされました。」という内容のエラーをブローカが表示することがあります。

  回避策 エラーメッセージは無視できます。ブローカはコンシューマへのエラーの通知を正しく処理します。このエラーは、システムの信頼性には影響を与えません。

- 持続ストアがあまりにも多くの送信先を開く場合、ブローカがアクセス不可能になる (バグ *ID 4953354*)。

  回避策 この状態はブローカがシステムのオープンファイル記述子の制限に達したことが原因です。Solaris や Linux では、`ulimit` コマンドを使って、ファイル記述子の制限を増やします。

- 送信先が破棄された場合、コンシューマが孤立する (バグ *ID 5060787*)。

  送信先が破棄された場合、アクティブコンシューマが孤立します。いったんコンシューマが孤立すると、送信先が再作成された場合でもメッセージを受信しなくなります。

  回避策 この問題には、回避策がありません。

- `JMSMessageID` を使用したメッセージ選択が機能しない (バグ *ID 6196233*)。

  回避策 次のセレクタを変更します。

  ```
  JMSMessageID = "ID:message-id-string"
  ```

  次のように変更します。

  ```
  JMSMessageID IN ('ID:message-id-string', 'message-id-string')
  ```

- Message Queue のキューブラウザが、コミットされていないメッセージも表示する (バグ *ID 6264003*)。

  キューの内容を検索するとき、キューブラウザの列挙に、トランザクション内で生成されたがまだコミットされていないメッセージが表示されることがあります。

  回避策 この問題には、回避策がありません。

- クラスタ内のブローカの接続が切断されます (バグ *ID 6377527*)。

  この問題の原因の 1 つとして、接続が切断されたブローカのアドレスがループバック IP アドレス (127.0.0.1) として解決されていることが考えられます。

回避策 ブローカのアドレスがループバック IP アドレスとして解決されないようにします。

再配布可能ファイル

Sun Java System Message Queue 4.0 には、バイナリ形式で使用でき自由に配布可能な次のファイルのセットが含まれています。

- `fscontext.jar`
- `imq.jar`
- `imqxm.jar`
- `jaas.jar`
- `jcert.jar`
- `jms.jar`
- `jndi.jar`
- `jnet.jar`
- `jsse.jar`
- `ldap.jar`
- `ldapbpjar`
- `libmqcrt.so` (HPUX)
- `libmqcrt.so` (UNIX)
- `mqcrt1.dll` (Windows)
- `providerutil.jar`

さらに、`LICENSE` ファイルおよび `COPYRIGHT` ファイルも再配布できます。

障害を持つ方のためのアクセシビリティー

このメディアの出版後にリリースされたアクセシビリティーを入手する場合は、請求に応じて Sun から提供される 508 条に関する製品評価資料を参照し、使いやすいソリューションの配備にもっとも適したバージョンを調べてください。更新バージョンのアプリケーションは、`http://sun.com/software/javaenterprisesystem/get.html` にあります。

ユーザー補助に対する Sun のコミットメントについては、`http://sun.com/access` を参照してください。

問題の報告方法とフィードバックの提供方法

Sun Java System Message Queue に関する問題を発見した場合は、次の方法のいずれかを使って Sun カスタマサポートにご連絡ください。

- Sun Software Support オンラインサービスの Web サイトをご利用ください。`http://www.sun.com/service/sunone/software`

  このサイトには、メンテナンスプログラムおよびサポート連絡先番号だけでなく、Knowledge Base、オンラインサポートセンター、および ProductTracker へのリンクがあります。
- 保守契約を結んでいるお客様の場合は、専用ダイヤルをご利用ください。

テクニカルサポートスタッフが問題解決のお手伝いをいたします。カスタマサポートをご利用の際は、次の情報をご用意ください。

- 問題が発生した箇所や動作への影響など、問題の具体的な説明
- マシン機種、OS バージョン、および、問題に影響すると思われるパッチやそのほかのソフトウェアなどの製品バージョン
- 問題を再現するために行なった操作に関する詳しい説明
- エラーログやコアダンプ

Sun Java System Software Forum

次のサイトでは、Sun Java System Message Queue フォーラムが利用できます。

http://swforum.sun.com/jive/forum.jspa?forumID=24

ご参加を歓迎いたします。

Java Technology Forum

Java Technology Forums には、関連する JMS のフォーラムがあります。

http://forum.java.sun.com

コメントの送信先

Sun ではマニュアルの改善に関心を払っており、お客様のコメントおよびご提案を必要としています。

コメントを送るには、http://docs.sun.com にアクセスして「コメントの送信」をクリックしてください。オンラインフォームにマニュアルのタイトルと Part No. を入力してください。Part No. は書籍のタイトルのページまたはマニュアルの最上部に記されており、通常、7 桁または 9 桁の数字です。このマニュアルのタイトルは『Sun Java System Message Queue 4.0 リリースノート』で、Part No. は 819-6960 です。

Sun の追加情報

次のサイトにも、Sun Java System に関する有益な情報が掲載されています。

- マニュアル

  http://docs.sun.com/prod/java.sys
- プロフェッショナルサービス

  http://www.sun.com/service/sunps/sunone
- ソフトウェア製品とサービス

http://www.sun.com/software

- ソフトウェアサポートサービス

  http://www.sun.com/service/sunone/software

- サポートと Knowledge Base

  http://www.sun.com/service/support/software

- Sun サポートとトレーニングサービス

  http://training.sun.com

- コンサルティングおよびプロフェッショナルサービス

  http://www.sun.com/service/sunps/sunone

- 開発者向け情報

  http://developers.sun.com

- Sun 開発者サポートサービス

  http://www.sun.com/developers/support

- ソフトウェアトレーニング

  http://www.sun.com/software/training