Sun Java System Message Queue 4.0 发行说明

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

文件号码 819–6961

060613@14986

目录

◆ ◆ ◆ 第 1 章

Sun Java System Message Queue 4.0 发行说明

版本 4.0

文件号码 819-6961

本发行说明包含了发行 Sun Java™ System Message Queue 4.0 时可用的重要信息。本说明主要介绍新增功能和增强功能、已知问题和限制以及其他信息。在使用 Message Queue 之前，请先阅读本文档。

本发行说明的最新版本可以在 Sun Java System Message Queue 文档 Web 站点找到。请在安装和设置软件前仔细查阅此 Web 站点，完成安装和设置后也要定期查看最新的发行说明和产品文档。

本发行说明包含以下部分：

本文档引用第三方 URL，并提供其他相关信息。

Sun 对本文档中提到的第三方 Web 站点的可用性不承担任何责任。对于此类站点或资源中的（或通过它们获得的）任何内容、广告、产品或其他材料，Sun 并不表示认可，也不承担任何责任。对于因使用或依靠此类站点或资源中的（或通过它们获得的）任何内容、产品或服务而造成的或连带产生的实际或名义损坏或损失，Sun 概不负责，也不承担任何责任。

发行说明修订历史记录

下表列出了 Message Queue 产品的所有 4.x 发行版的发行日期，并描述了与每个发行版关联的主要更改。

表 1–1 修订历史记录

日期	更改描述
2006 年 5 月	本发行说明文档的初始版本。

关于 Message Queue 4.0

Sun Java System Message Queue 是一种功能全面的消息服务，提供符合 Java Messaging Specification (JMS) 1.1 的可靠、异步的消息传送功能。此外，Message Queue 还提供 JMS 规范之外的许多功能，以满足大型企业部署的需要。

Message Queue 4.0 发行版仅支持 Application Server 9 PE。它是一个次要发行版，其中包括一些新功能，少量的增强功能和一些错误修复。本部分包含以下信息：

- 第 6 页中的 “此发行版的新增功能”
- 第 19 页中的 “硬件和软件要求”

此发行版的新增功能

Message Queue 4.0 包括以下新功能：

- 第 7 页中的 “对 C-API 和 C 客户端运行时的接口更改”
- 第 7 页中的 “显示有关持久性存储库的信息”
- 第 8 页中的 “持久性存储库格式更改”
- 第 8 页中的 “代理管理”
- 第 9 页中的 “JDBC 持久性支持”
- 第 9 页中的 “SSL 支持”
- 第 9 页中的 “JMX 支持”
- 第 13 页中的 “客户端运行时日志记录”
- 第 17 页中的 “连接事件通知”

这些功能将在以下各部分中进行描述。

注意 – 在版本 4.0 中进行了一些比较细微但可能会引起问题的更改，其中之一就是不再使用命令行选项来指定密码。此后，必须将所有密码都存储在文件中，如第 22 页中的 “过时的密码选项”中所述。

对 C-API 和 C 客户端运行时的接口更改

在 3.7 发行版中已经对 C-API 进行了更改：这些更改包括一个新函数、一个新消息类型和一个新连接属性。

- 新函数：MQGetDestinationName()

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

  使用此函数可获取目标名称。返回的 destinationName 是一个副本，调用程序通过调用 MQFreeString() 函数释放此副本。

  参数

  destinationHandle　　要了解其名称的目标的句柄。

  destinationName　　名称的输出参数。

  使用回复模式时，此函数会非常有用。可以使用 MQGetMessageReplyTo 函数获取消息发送时所在目标的句柄。然后可以使用 MQGetDestinationName 获取该目标的名称。获取目标名称之后，即可根据名称处理消息。

- 新枚举值：MQ_MESSAGE

  新的 MQMessageType MQ_MESSAGE 允许 C 客户端与其他 Message Queue 客户端（C 和 Java）交换 Message 类型的 JMS 消息：

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

  MQ_MESSAGE 类型用于标识具有标题和属性但没有消息主体的消息。使用 MQCreateMessage() 函数可创建此类型的消息。

- 新连接属性 MQ_UPDATE_RELEASE_PROPERTY，用于指定已安装的 Message Queue 的更新发行版本。使用 MQGetMetaData() 函数可获取版本信息。

显示有关持久性存储库的信息

已将 query 子命令添加到 imqdbmgr 命令。使用此子命令可显示有关持久性存储库的信息，包括存储库版本、数据库用户以及是否已创建数据库表。

以下是此命令所显示的信息的示例。

```
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.
```

持久性存储库格式更改

为了提高性能，Message Queue 3.7 UR1 对持久性存储库格式进行了两处更改。一处是对文件存储库的更改，另一处是对 JDBC 存储库的更改。

- 文件存储库中保留的事务数据的格式

  已经对事务状态信息（存储在 Message Queue 的基于文件的持久性存储库中）的格式进行了更改，以减少磁盘 I/O 和提高 JMS 事务性能。

- Oracle JDBC 存储库

  在 Message Queue 的以前版本中，适用于 Oracle 的存储库结构使用 `LONG RAW` 数据类型来存储消息数据。在 Oracle 8 中，Oracle 引入了 `BLOB` 数据类型，而不再使用 `LONG RAW` 类型。Message Queue 3.7 UR1 已转为使用 `BLOB` 数据类型，以提高性能和提供更多支持。

由于这些更改会影响存储库兼容性，因此已将 Message Queue 3.7 UR1 中文件存储库和 JDBC 存储库的版本从 350 更改为 370。

Message Queue 4.0 对 JDBC 存储库进行了一些更改，以便实现优化和支持未来的增强功能。因此，JDBC 存储库版本已提高到 400。请注意，在版本 4.0 中，基于文件的持久性存储库版本仍保持为 370，因为未对此版本进行任何更改。

Message Queue 4.0 允许将持久性存储库自动转换为最新版本的基于文件持久性存储库和 JDBC 持久性存储库。`imqbrokerd` 首次启动时，如果实用程序检测到早期版本的存储库，则会将存储库迁移到新格式，并留下早期的存储库。

- 基于文件存储库的 200 和 350 版本将被迁移到 370 版本格式。
- JDBC 存储库的 350 和 370 版本将被迁移到 400 版本格式。（如果需要升级 200 版本的存储库，需要先逐步升级到 3.5 或 3. 6 中间版本。）

如果需要回滚此升级，可以先卸载 Message Queue 4.0，然后重新安装以前运行的版本。由于未对存储库的早期副本进行任何更改，因此代理可以运行存储库的早期副本。

代理管理

命令实用程序 (`imqcmd`) 添加了一个子命令和多个选项，允许管理员停止代理、在指定的时间间隔后关闭代理或销毁连接。

- 停止代理可使其进入静默状态，以便在关闭或重新启动代理之前处理完所有消息。无法对处于停止状态的代理创建新连接。要停止代理，请输入如下命令。

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

- 要在指定的时间间隔后关闭代理，请输入如下命令。时间间隔指定了在代理关闭之前要等待的秒数。

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

- 要销毁连接，请输入如下命令。

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

  使用 `imqcmd list cxn` 或 `imqcmd query cxn` 命令可获取连接 ID。

有关 `imqcmd` 命令语法的完整信息，请参见《Sun Java System Message Queue 3 2005Q4 管理指南》中的第 14 章“命令行参考”。

JDBC 持久性支持

目前可以将 Apache Derby 版本 10.1.1 作为符合 JDBC 的持久性存储库提供程序。

SSL 支持

从发行版 4.0 开始，客户端连接工厂属性 `imqSSLIsHostTrusted` 的默认值为 `false`。如果应用程序依赖于以前的默认值 `true`，则需要对此属性进行重新配置，并将其明确设置为 `true`。

JMX 支持

已添加符合 Java Management Extension (JMX) 规范的新 API，用于配置和监视 Message Queue 代理。使用此 API，可以在 Message Queue 客户端应用程序中以编程方式来配置和监视代理函数。在 Message Queue 的早期版本中，只能从命令行或管理控制台访问这些函数。

此 API 由一组 JMX 受管理 *Bean (MBean)* 组成，用于管理以下与 Message Queue 相关的资源：

- 消息代理
- 连接服务
- 连接
- 目标
- 消息生成方
- 消息使用方
- 事务
- 代理群集
- 日志记录
- Java 虚拟机 (Java Virtual Machine, JVM)

这些 MBean 提供了一些属性和操作，用于同步轮询和处理基础资源的状态，此外还提供了通知，以使客户端应用程序能够在状态发生更改时异步侦听和响应这些更改。使用 JMX API，客户端应用程序可以执行如下配置和监视任务：

- 设置代理的端口号
- 设置代理的最大消息大小
- 暂停连接服务
- 设置连接服务的最大线程数
- 获取服务上的当前连接数

- 销毁连接
- 创建目标
- 销毁目标
- 启用或禁用自动创建目标
- 清除来自目标的所有消息
- 获取自代理启动以来目标收到的累积消息数
- 获取队列的当前状态（正在运行或已暂停）
- 获取某主题当前消息生成方的数量
- 清除来自长期订阅者的所有消息
- 获取当前 JVM 堆大小

有关 JMX API 的介绍以及完整的参考信息，请参见《Sun Java System Message Queue 4.0 Developer's Guide for JMX Clients》。

代理支持：与 JMX 有关的属性

已添加多个新代理属性，以支持 JMX API（请参见表 1–2）。无法使用 Message Queue 命令实用程序 (`imqcmd`) 从命令行设置这些属性。可以使用代理实用程序 (`imqbrokerd`) 的 `-D` 选项设置这些属性，也可以在代理的实例配置文件 (`config.properties`) 中手动编辑这些属性。此外，还可以使用表 1–3 中所述的新代理实用程序选项来设置其中某些属性（`imq.jmx.rmiregistry.start`、 `imq.jmx.rmiregistry.use` 和 `imq.jmx.rmiregistry.port`）。此表列出了每个选项，指定了选项类型，并描述了选项用途。

表 1–2 用于支持 JMX 的新代理属性

属性	类型	描述
`imq.jmx.rmiregistry.start`	`Boolean`	代理启动时是否启动 RMI 注册表? 如果为 `true`，代理将在 `imq.jmx.rmiregistry.port` 指定的端口上启动 RMI 注册表，并使用此注册表存储 JMX 连接器的 RMI 存根。请注意，在这种情况下将忽略 `imq.jmx.rmiregistry.use` 的值。 默认值：`false`
`imq.jmx.rmiregistry.use`	`Boolean`	是否使用外部 RMI 注册表? 仅当 `imq.jmx.rmiregistry.start` 为 `false` 时适用。 如果为 `true`，代理将在 `imq.jmx.rmiregistry.port` 指定的端口上使用外部 RMI 注册表，以便存储 JMX 连接器的 RMI 存根。代理启动时外部 RMI 注册表必须已处于运行状态。 默认值：`false`

表 1–2 用于支持 JMX 的新代理属性　（续）

属性	类型	描述
`imq.jmx.rmiregistry.port`	`Integer`	RMI 注册表的端口号 仅当 `imq.jmx.rmiregistry.start` 或 `imq.jmx.rmiregistry.use` 为 `true` 时适用。然后即可将 JMX 连接器配置为使用 RMI 注册表，方法是将此端口号添加到这些连接器的 JMX 服务 URL 的 URL 路径中。 默认值：`1099`
`imq.jmx.connector.list`	`String`	预配置的 JMX 连接器的名称，以逗号分隔。 默认值：`jmxrmi,ssljmxrmi`
`imq.jmx.connector.activelist`	`String`	代理启动时要激活的 JMX 连接器的名称，以逗号分隔。 默认值：`jmxrmi`
`imq.jmx.connector.`*connectorName* `.urlpath`	`String`	连接器 *connectorName* 的 JMX 服务 URL 的 *urlPath* 部分 当必须明确设置 JMX 服务 URL 路径时（例如使用共享的外部 RMI 注册表时），此选项会非常有用。 默认值：如果使用 RMI 注册表存储 JMX 连接器的 RMI 存根（即，如果 `imq.jmx.registry.start` 或 `imq.jmx.registry.use` 为 `true`）： `/jndi/rmi://`*brokerHost*`:`*rmiPort* `/`*brokerHost*`/`*brokerPort*`/`*connectorName* 如果不使用 RMI 注册表（这是默认情况，`imq.jmx.registry.start` 和 `imq.jmx.registry.use` 均为 `false`）： `/stub/`*rmiStub* 其中 *rmiStub* 是 RMI 存根自身的编码和序列化表示
`imq.jmx.connector.`*connectorName* `.useSSL`	`Boolean`	是否对连接器 *connectorName* 使用安全套接字层 (Secure Socket Layer, SSL)？ 默认值：`false`

表 1–2 用于支持 JMX 的新代理属性 （续）

属性	类型	描述
imq.jmx.connector.*connectorName* .brokerHostTrusted	Boolean	连接器 *connectorName* 是否信任代理提供的所有证书？ 仅当 imq.jmx.connector. *connectorName*.useSSL 为 true 时适用。 如果为 false，Message Queue 客户端运行时将验证提供给它的所有证书。如果证书的签名者不在客户端的信任存储库中，验证将失败。 如果为 true，则会跳过证书验证。此选项可能非常有用，例如，在使用自签名证书进行软件测试时。 默认值：false

imq.jmx.connector.list 属性用于定义一组要在代理启动时创建的命名 JMX 连接器；imq.jmx.connector.activelist 用于指定要激活其中哪些连接器。每个命名的连接器都有其自身的一组属性：

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

默认情况下，将创建两个 JMX 连接器，名称分别为 jmxrmi 和 ssljmxrmi；前者被配置为不使用 SSL 加密 (imq.jmx.connector.jmxrmi.useSSL = false)，后者则使用 SSL 加密 (imq.jmx.connector.ssljmxrmi.useSSL = true)。默认情况下，代理启动时只激活 jmxrmi 连接器；有关如何激活 ssljmxrmi 连接器以进行安全通信的信息，请参见 第 13 页中的 “针对 JMX 客户端的 SSL 支持” 。

为方便起见，还将新选项（表 1–3）添加到命令行代理实用程序 (imqbrokerd)，以控制 RMI 注册表的使用、启动和端口。这些选项的用途和效果与等效代理属性的用途和效果相同，如表 1–2 中所述。此表列出了每个选项，指定了选项的等效代理属性，并描述了选项用途。

表 1–3 用于支持 JMX 的新代理实用程序选项

选项	等效代理属性	描述
-startRmiRegistry	imq.jmx.rmiregistry.start	代理启动时是否启动 RMI 注册表?
-useRmiRegistry	imq.jmx.rmiregistry.use	是否使用外部 RMI 注册表?
-rmiRegistryPort	imq.jmx.rmiregistry.port	RMI 注册表的端口号

已将新的子命令（表 1–4）添加到命令行命令实用程序 (imqcmd)，用于列出代理启动时创建和启动的 JMX 连接器的 JMX 服务 URL。不使用 Message Queue 简便类

AdminConnectionFactory 的 JMX 客户端需要使用此信息来获取 JMX 连接器；另外，通过通用的 JMX 浏览器（例如 Java 监视和管理控制台 jconsole），还可以使用此信息来管理或监视 Message Queue。

表 1–4 新的命令实用程序子命令

子命令	描述
list jmx	列出 JMX 连接器的 JMX 服务 URL

针对 JMX 客户端的 SSL 支持

如上所述，默认情况下将使用预配置的 JMX 连接器 jmxrmi 来配置 Message Queue 消息代理，从而进行不安全的通信。需要使用安全套接字层 (Secure Socket Layer, SSL) 进行安全通信的应用程序必须激活备用的安全 JMX 连接器 ssljmxrmi。这需要执行以下步骤：

1. 获取和安装签名证书，方法与为 ssljms、ssladmin 或 cluster 连接服务获取和安装签名证书（如 Message Queue 管理指南 中所述）相同。
2. 在信任存储库中安装根证书颁发机构颁发的证书（如有必要）。
3. 将 ssljmxrmi 连接器添加到代理启动时要激活的 JMX 连接器列表：

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

4. 使用 Message Queue 代理实用程序 (imqbrokerd) 启动代理，可以将密码文件中的密钥库密码传递给此实用程序，也可以在提示时从命令行键入密码。
5. 默认情况下，ssljmxrmi 连接器（或基于 SSL 的任何其他连接器）被配置为验证提供给它的所有代理 SSL 证书。要避免进行此验证（例如，在使用自签名证书进行软件测试时），请将代理属性 imq.jmx.connector.ssljmxrmi.brokerHostTrusted 设置为 true。

在客户端，必须使用将 ssljmxrmi 指定为首选连接器的 URL 配置管理员连接工厂 (AdminConnectionFactory)：

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

如果需要，可使用系统属性 javax.net.ssl.trustStore 和 javax.net.ssl.trustStorePassword 将 JMX 客户端指向信任存储库。

客户端运行时日志记录

本部分介绍了 Message Queue 4.0 对客户端运行时日志记录的支持，记录内容包括与连接和会话有关的事件。

JDK 1.4（及更高版本）包含 java.util.logging 库。此库实现了一个标准记录程序接口，可用于特定于应用程序的日志记录。

Message Queue 客户端运行时使用 Java 日志记录 API 来实现日志记录功能。可以使用所有 J2SE 1.4 日志记录工具来配置日志记录活动。 例如，应用程序可以使用以下 Java 日志记录工具来配置 Message Queue 客户端运行时输出日志记录信息的方式：

- 日志记录处理程序
- 日志记录过滤器
- 日志记录格式化程序
- 日志记录级别

有关 Java 日志记录 API 的详细信息，请参见位于 http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html 的 Java 日志记录概述。

日志记录的名称空间、级别和活动

Message Queue 提供程序定义了一组与日志记录级别和日志记录活动相关联的日志记录名称空间，以使 Message Queue 客户端能够在正确设置日志记录配置后记录连接和会话事件。

Message Queue 客户端运行时的根日志记录名称空间被定义为 `javax.jms`。Message Queue 客户端运行时中的所有记录程序均将此名称用作父名称空间。

用于 Message Queue 客户端运行时的日志记录级别与 `java.util.logging.Level` 类中定义的级别相同。此类定义了七个标准日志级别，另外还有两个用于打开和关闭日志记录的设置。

`OFF`	关闭日志记录。
`SEVERE`	最高优先级，最大值。由应用程序定义。
`WARNING`	由应用程序定义。
`INFO`	由应用程序定义。
`CONFIG`	由应用程序定义。
`FINE`	由应用程序定义。
`FINER`	由应用程序定义。
`FINEST`	最低优先级，最小值。由应用程序定义。
`ALL`	启用所有消息的日志记录。

通常，Message Queue 客户端运行时中发生的异常和错误由具有 `javax.jms` 名称空间的记录程序进行记录。

- 从 JVM 抛出并由客户端运行时捕获的异常（例如 `IOException`）由具有 `javax.jms` 日志记录名称空间的记录程序记录为 `WARNING` 级别。
- 从客户端运行时抛出的 JMS 异常（例如 `IllegalStateException`）由具有 `javax.jms` 日志记录名称空间的记录程序记录为 `FINER` 级别。
- 从 JVM 抛出并由客户端运行时捕获的错误（例如 `OutOfMemoryError`）由具有 `javax.jms` 日志记录名称空间的记录程序记录为 `SEVERE` 级别。

下表列出了可记录的事件和日志级别，要记录 JMS 连接事件和会话事件，必须设置此级别。

下表描述了连接的日志级别和事件。

表 1–5 `javax.jms.connection` 名称空间的日志级别和事件

日志级别	事件
FINE	连接已创建
FINE	连接已启动
FINE	连接已关闭
FINE	连接已断开
FINE	已重新连接
FINER	其他连接活动，例如 `setClientID`
FINEST	消息、确认、Message Queue 操作和控制消息（例如提交事务）

对于会话，将在日志记录中记录以下信息。

- 传送给使用方的消息的每个日志记录包括 ConnectionID、SessionID 和 ConsumerID。
- 由生成方发送的消息的每个日志记录包括 ConnectionID、SessionID、ProducerID 和目标名称。

下表描述了会话的日志级别和事件。

表 1–6 `javax.jms.session` 名称空间的日志级别和事件

日志级别	事件
FINE	会话已创建
FINE	会话已关闭
FINE	生成方已创建
FINE	使用方已创建
FINE	目标已创建
FINER	其他会话活动，例如提交会话。
FINEST	已生成并使用消息。（未在日志记录中记录消息属性和主体。）

默认情况下，输出日志级别是从运行应用程序所在的 JRE 继承而来的。可查看 `JRE_DIRECTORY/lib/logging.properties` 文件以确定该级别。

可通过编程方式或使用配置文件来配置日志记录，还可以控制进行日志记录的范围。以下部分介绍了这些可行的操作。

使用 JRE 日志记录配置文件

以下示例说明了如何在 JRE_DIRECTORY/lib/logging.properties 文件（用于设置 Java 运行环境的日志级别）中设置日志记录的名称空间和级别。使用此 JRE 的所有应用程序都将具有相同的日志记录配置。下面的样例配置将 javax.jms.connection 名称空间的日志记录级别设置为 INFO，并指定将输出写入 java.util.logging.ConsoleHandler。

```
#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
```

为特定的应用程序使用日志记录配置文件

还可以从用于运行应用程序的 Java 命令行定义日志记录配置文件。该应用程序将使用在指定的日志记录文件中定义的配置。在以下示例中，configFile 使用的格式与 JRE_DIRECTORY/lib/logging.properties 文件中定义的格式相同。

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

以编程方式设置日志记录配置

以下代码使用 java.util.logging API 来记录连接事件，方法是将 javax.jms.connection 名称空间的日志级别更改为 FINE。您可以在应用程序中包含此类代码，以便通过编程方式设置日志记录配置。

```
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);
```

连接事件通知

连接事件通知允许 Message Queue 客户端侦听关闭和重新连接事件，并根据通知类型和连接状态采取适当的操作。例如，如果发生故障转移，并且客户端重新连接到了其他代理，则应用程序可能希望清除事务状态，并继续执行新事务。

如果 Message Queue 提供程序检测到某个连接存在严重问题，它将调用该连接对象的已注册的异常侦听器。Message Queue 提供程序使用以下方法完成此操作：调用侦听器的 onException 方法，并向其传递一个用于描述该问题的 JMSException 参数。同样，Message Queue 提供程序还提供一个事件通知 API，该 API 允许客户端运行时向应用程序通知有关连接状态更改的信息。通知 API 由以下元素定义：

- com.sun.messaging.jms.notification 软件包，用于定义事件侦听器和通知事件对象。
- com.sun.messaging.jms.Connection 接口，用于定义 javax.jms.Connection 接口的扩展。

以下部分介绍了可以触发通知的事件，并说明了如何创建事件侦听器。

连接事件

下表列出并介绍了事件侦听器可以返回的事件。

请注意，当发生连接事件时，不会调用 JMS 异常侦听器。仅当客户端运行时达到最大重新连接尝试次数时，才会调用异常侦听器。客户端运行时在调用异常侦听器之前，始终会调用事件侦听器。

表 1–7 通知事件

事件类型	含义
ConnectionClosingEvent	如果 Message Queue 客户端运行时从代理收到通知，指明由于管理员请求关闭而即将关闭连接，则 Message Queue 客户端运行时将生成此事件。
ConnectionClosedEvent	关闭连接时（由于代理错误或管理员请求关闭或重新启动连接），Message Queue 客户端运行时将生成此事件。 如果事件侦听器收到 ConnectionClosedEvent，则应用程序可以使用所收到的事件的 getEventCode() 方法，来获取用于指明关闭原因的事件代码。
ConnectionReconnectedEvent	Message Queue 客户端运行时已重新连接到代理。此代理可以是客户端先前连接的代理，也可以是其他代理。 应用程序可以使用所收到的事件的 getBrokerAddress 方法，来获取它所重新连接到的代理的地址。
ConnectionReconnectFailedEvent	Message Queue 客户端运行时无法重新连接到代理。每次重新连接尝试失败时，该运行时都会生成新的事件，并将该事件传送到事件侦听器。 发生连接事件时不会调用 JMS 异常侦听器。仅当客户端运行时达到最大重新连接尝试次数时，才会调用 JMS 异常侦听器。客户端运行时在调用异常侦听器之前，始终会调用事件侦听器。

创建事件侦听器

以下代码示例说明了如何设置连接事件侦听器。只要发生连接事件，客户端运行时就会调用事件侦听器的 onEvent 方法。

```
//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 );
```

事件侦听器示例

在此示例中，应用程序选择让事件侦听器将连接事件记录到应用程序的日志记录系统中：

```
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.
         }
}
```

硬件和软件要求

在《Sun Java System Application Server Platform Edition 9 Release Notes》中的“Hardware and Software Requirements”中指定了硬件和软件要求。

此发行版中修复的错误

下表介绍了在 Message Queue 4.0 中修复的错误。

表 1–8 在 Message Queue 4.0 中修复的错误

错误号	描述
4986481	在 Message Queue 3.5 中，调用 `Session.recover`可能会在自动重新连接模式中挂起。
4987325	调用 `Session.recover` 之后，会将重新传送的消息的“已重新传送”标志设置为 `false`。
6157073	将新连接消息更改为既包含总连接数，也包含服务上的连接数。

表 1–8 在 Message Queue 4.0 中修复的错误　（续）

错误号	描述
6193884	在非 C 语言环境中，Message Queue 向系统日志输出垃圾消息。
6196233	无法使用 JMSMessageID 选择消息。
6251450	在群集关闭过程中，connectList 出现 ConcurrentModificationException。
6252763	java.nio.HeapByteBuffer.putLong/Int 出现 java.nio.BufferOverflowException。
6260076	使用 Oracle 存储库，在代理启动后的第一条消息发布完后，消息发布速度变得很慢。
6260814	对 JMSXUserID 进行处理的选择器始终得到 false 值。
6264003	队列浏览器显示未提交的消息。
6271876	关闭具有未使用消息的使用方时，连接流控制无法正常工作。
6279833	Message Queue 不允许两个代理使用相同的 JDBC 表。
6293053	除非清除存储库中的内容（使用 –reset store），否则当系统的 IP 地址更改时，主代理将无法正确启动。
6294767	Message Queue 代理需要在其打开的网络套接字上设置 SO_REUSEADDR。
6304949	无法设置 TopicConnectionFactory 的 ClientID 属性。
6307056	Txn 日志成为性能瓶颈。
6320138	Message Queue C API 无法确定回复标头中的队列名称。
6320325	即使在 Solaris 上同时安装了 JDK 1.4 和 JDK 1.5，代理有时也会选取前者。
6321117	多代理群集初始化抛出 java.lang.NullPointerException。
6330053	从订户提交事务时，JMS 客户端抛出 java.lang.NoClassDefFoundError。
6340250	支持 C-API 中的 MESSAGE 类型。
6351293	添加对 Apache Derby 数据库的支持。

重要信息

本部分介绍核心产品文档中未包括的最新信息。本部分包含以下主题：

兼容性问题

本部分包含 Message Queue 4.0 中的兼容性问题。

接口稳定性

Sun Java System Message Queue 使用的许多接口可能会随着时间的变化而发生更改。《Sun Java System Message Queue 3 2005Q4 管理指南》 中的第 21 章根据接口的稳定性对接口进行了分类。 接口越稳定，在产品的后续版本中对其进行更改的可能性就越小。

与 Message Queue 的下一个主要发行版相关的问题

在 Message Queue 下一个主要发行版中进行的某些更改可能会导致您的客户端与该发行版不兼容。现在提供此信息是为了让您针对这些更改做好准备。

- 作为 Sun Java System Message Queue 的一部分进行安装的个别文件的位置可能会发生更改。如果现有的应用程序依赖于某些 Message Queue 文件的当前位置，则这些应用程序可能会中断。
- 3.5 及更早版本的代理可能无法再在具有较高版本代理的群集中运行。
- 在以后的版本中，Message Queue 客户端可能无法使用早于 1.3 的 JDK 版本。

Message Queue 4.0 的文档更新

与本发行说明文档不同，Message Queue 4.0 只包含一个新文档：《Sun Java System Message Queue 4.0 Developer's Guide for JMX Clients》 。

针对 Message Queue 3.6 SP3, 2005Q4 发布的 Message Queue 文档在有关 Application Server 9 PE 客户端需求方面包含了最新内容。可在以下位置找到此文档集：

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

`http://docs.sun.com/app/docs/coll/1391.1`

已知问题和限制

本部分包含了 Message Queue 4.0 中已知问题的列表。涵盖以下产品领域：

有关当前错误、错误的状态和解决方法的列表，Java Developer Connection™ 成员应参见 Java Developer Connection Web 站点上的 Bug Parade 页。在报告新的错误之前请先查看该页。虽然未列出所有的 Message Queue 错误，但如果您想了解是否已报告了某个问题，可以将该页作为一个很好的起点。

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

注 – 可以免费获得 Java Developer Connection 成员资格，但需要进行注册。有关如何成为 Java Developer Connection 成员的详细信息，请访问 Sun 的 "For Developers" Web 页。

要报告新错误或提交功能请求，请向 `imq-feedback@sun.com` 发送电子邮件。

过时的密码选项

在以前版本的 Message Queue 中，可以对以下命令使用 `–p` 或 `–password` 选项以交互方式来指定密码：`imqcmd`、`imqbrokerd` 和 `imdbmgr`。从版本 4.0 开始，将不再使用这些选项。必须按照以下方式提供密码。

1. 在仅用于存储密码的文件中将适当的属性设置为所需的密码值。

   使用以下语法在密码文件中指定密码。

   PasswordPropertyName =MyPassword
2. 使用 `–passfile` 选项传递密码文件的名称。

密码文件可以包含以下列出的一个或多个密码。

- 用于打开 SSL 密钥库的密钥库密码。可使用 `imq.keystore.password` 属性指定此密码。
- LDAP 系统信息库密码，用于在非匿名连接的情况下与 LDAP 目录进行安全连接。可使用 `imq.user_repository.ldap.password` 属性指定此密码。
- 用于连接到符合 JDBC 的数据库的 JDBC 数据库密码。可使用 `imq.persist.jdbc.password` 属性指定此密码。
- `imqcmd` 命令（用于执行代理管理任务）的密码。可使用 `imq.imqcmd.password` 属性指定此密码。

在以下示例中，将 JDBC 数据库的密码设置为 `abracadabra`。

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

可以将代理配置为使用密码文件，该文件可通过以下任一方法创建。

- 在代理的 `config.properties` 文件中设置以下属性。

  ```
  imq.passfile.enabled=true
  imq.passfile.dirpath=MyFileDirectory
  imq.passfile.name=MyPassfileName
  ```
- 使用 `imqbrokerd` 命令的 `–passfile` 选项。

  ```
  imqbrokerd –passfile MyPassfileName
  ```

一般问题

本部分包含 Message Queue 4.0 中的一般问题。其中某些问题是以前的 Message Queue 版本引入的。

以下问题对于 Message Queue 4.0 产品的两个版本都适用。

- SOAP 客户端。以前，用于引用 `mail.jar` 和 `mail.jar` 的 SAAJ 1.2 实现 jar 无需位于 `CLASSPATH` 中。在 SAAJ 1.3 中删除了此引用，因此 Message Queue 客户端必须将 `mail.jar` 明确放入 `CLASSPATH` 中。
- 在 Message Queue 4.0 中，在 `config.properties` 文件的注释部分给出了将 LDAP 服务器用作用户系统信息库的示例代理配置，而 `default.properties` 文件中的 LDAP 用户系统信息库示例已被注释掉。

  如果您以前使用示例 LDAP 用户系统信息库属性（在 `default.properties` 文件中指定）中的任何属性值，则当您的 JMS 应用程序客户端尝试创建 JMS 连接时，将会收到安全异常。升级到 Message Queue 4.0 之后将会发生这种情况。

  当 JMS 客户端尝试连接到 Message Queue 4.0 代理时，代理日志中会记录一个错误，且 JMS 客户端会收到以下异常：

  ```
  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
  ```

  解决方法 请按照《Sun Java System Message Queue 3 2005Q4 管理指南》第 8 章中的说明来设置代理属性 `imq.user_repository.ldap.uidattr`。

JMX 问题

以下是有关使用 JMX API 的问题。

- 在 Windows 平台上，事务管理器监视 MBean 的 `getTransactionInfo` 方法将返回具有错误事务创建时间的事务信息（错误号 *6393359*）。

  解决方法 请改用事务管理器监视 MBean 的 `getTransactionInfoByID` 方法。
- 将目标监视 MBean 的 `getActiveConsumerIDs` 方法用于主题目标时抛出异常（错误号 *6397184*）。

  解决方法 请改用目标监视 MBean 的 `getConsumerIDs` 方法。

管理/配置问题

以下是有关管理和配置 Message Queue 的问题

- 在 Windows 计算机上，当 CLASSPATH 包含双引号时，imqadmin 和 imqobjmgr 实用程序将抛出错误（错误号 *5060769*）。

  解决方法 可以忽略此错误消息；代理会正确处理，并将任何错误通知使用方。 此错误不影响系统的可靠性。

- 如果所提供的值中包含空格，则所有 Solaris 和 Windows 脚本中的 -javahome 选项都不起作用（错误号 *4683029*）。

  Message Queue 命令和实用程序使用 javahome 选项来指定要使用的备用 Java 2 兼容运行时。但是，备用 Java 运行时的路径名不能包含空格。以下是包含空格的路径示例。

  Windows：`C:/jdk 1.4`

  Solaris： `/work/java 1.4`

  解决方法 请在不包含空格的位置或路径中安装 Java 运行时。

- imqQueueBrowserMaxMessagesPerRetrieve 属性指定客户端运行时在浏览队列目标的内容时一次检索到的最大消息数。请注意，客户端应用程序始终可以获取队列上的所有消息。因此，imqQueueBrowserMaxMessagesPerRetrieve 属性会影响如何对排队的消息进行分块，以便传送到客户端运行时（消息较少时用较大的块，或消息较多时用较小的块），但不会影响所浏览的全部消息。更改此属性的值可能会对性能产生影响，但不会导致客户端应用程序获取的数据增加或减少（错误号 *6387631*）。

代理问题

以下问题将影响 Message Queue 代理。

- 当使用 Ctrl-C 关闭代理时，事务可能会在存储库关闭后被清除（错误号 *4934446*）。

  如果在处理消息或事务时关闭代理，代理可能会显示错误消息，原因是“存储库关闭后访问存储方法”。

  解决方法 可以忽略此错误消息；代理会正确处理，并将任何错误通知使用方。 此错误不影响系统的可靠性。

- 如果持久性存储库打开过多目标，将无法访问代理。（错误号 *4953354*）。

  解决方法 这种情况是由于代理达到了系统打开文件描述符限制所致。在 Solaris 和 Linux 上可使用 ulimit 命令来增加文件描述符限制。

- 目标被销毁后，使用方将被孤立（错误号 *5060787*）。

  目标被销毁后，活动的使用方将被孤立。使用方孤立后，他们将不再接收消息（即使重新创建了目标）。

  解决方法 此问题没有解决方法。

- 无法使用 JMSMessageID 选择消息（错误号 *6196233*）。

  解决方法 更改选择器，将以下表达式：

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

  更改为以下表达式：

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

- Message Queue 队列浏览器显示未提交的消息（错误号 *6264003*）。
  浏览队列内容时，已在事务中生成但尚未提交的消息可能会显示在队列浏览器枚举中。
  解决方法 此问题没有解决方法。
- 群集中某个代理的连接丢失（错误号 *6377527*）。
  产生此错误的一个原因是，丢失连接的代理的地址被解析为回送 IP 地址 (127.0.0.1)。
  解决方法 请确保不要将代理地址解析为回送 IP 地址。

可再分发的文件

Sun Java System Message Queue 4.0 中包含以下一组文件，您可以使用这些文件，并以二进制格式自由分发它们：

- `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`

此外，还可以再分发 `LICENSE` 和 `COPYRIGHT` 文件。

为残疾人士提供的辅助功能

欲获得自本介质发行以来所发布的辅助功能，请联系 Sun 索取有关 "Section 508" 法规符合性的产品评估文档，以便确定哪些版本最适合部署辅助功能解决方案。可通过以下网址获取应用程序的更新版本：

`http://sun.com/software/javaenterprisesystem/get.html`

有关 Sun 在辅助功能方面所做出的努力，请访问

`http://sun.com/access`

如何报告问题和提供反馈

如果您在使用 Sun Java System Message Queue 期间遇到问题，请通过以下方式与 Sun 客户支持部门联系：

- 位于 http://www.sun.com/service/sunone/software 的 Sun 软件支持联机服务。

  此站点上有一些链接，通过这些链接可以访问知识库、联机支持中心和 Product Tracker，还可了解维护方案以及用于联系支持部门的电话号码。
- 随维护合同一起分发的电话号码。

为了更好地帮助您解决问题，请在联系支持部门时提供以下信息：

- 问题描述，包括问题出现时的情况及其对您的操作的影响。
- 计算机类型、操作系统版本和产品版本，包括可能影响问题的所有修补程序和其他软件。
- 用来再现该问题的详细步骤。
- 所有错误日志或核心转储。

Sun Java System 软件论坛

以下位置提供了一个 Sun Java System Message Queue 论坛：

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

我们欢迎您的参与。

Java 技术论坛

您可能会对 Java 技术论坛中的 JMS 论坛感兴趣。

http://forum.java.sun.com

Sun 欢迎您提出意见

Sun 致力于提高其文档的质量，并十分乐意收到您的意见和建议。

要共享您的意见，请访问 http://docs.sun.com，然后单击“发送意见”(Send Comments)。在联机表单中提供文档标题和文件号码。文件号码包含七位或九位数字，可在书的标题页或在文档顶部找到该号码。例如，本书的标题为《Sun Java System Message Queue 4.0 发行说明》，文件号码为 819-6961。

提出意见时您还需要在表格中输入文档的英文文件号码和标题。本文档的英文文件号码是 819–5946，文档标题为《Sun Java System Message Queue 4.0 Release Notes》。

其他 Sun 资源

从以下 Internet 位置可以找到有用的 Sun Java System 信息：

- 文档

  http://docs.sun.com/prod/java.sys 和

  http://docs.sun.com/prod/java.sys?l=zh
- 专业服务

  http://www.sun.com/service/sunps/sunone
- 软件产品和服务

  http://www.sun.com/software
- 软件支持服务

  http://www.sun.com/service/sunone/software
- 支持和知识库

  http://www.sun.com/service/support/software
- Sun 支持和培训服务

  http://training.sun.com
- 咨询和专业服务

  http://www.sun.com/service/sunps/sunone
- 开发者信息

  http://developers.sun.com
- Sun 开发者支持服务

  http://www.sun.com/developers/support
- 软件培训

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