Sun Java™ System

Message Queue 3
管理指南

2005Q4

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

文件号码：819-3561

目录

图

表

过程

前言

Sun Java™ System Message Queue 管理指南介绍了 Sun Java System Message Queue 3 2005Q4 (Message Queue 3.6)，并提供了管理 Message Queue 消息传送系统所需的信息。

本前言包含以下各节：

目标读者

本手册的目标读者为需要执行 Message Queue 管理任务的管理员和应用程序开发者。 Message Queue **管理员**负责设置和管理 Message Queue 消息传送系统，尤其是处于该系统核心的消息服务器。

阅读本书之前

阅读本手册之前，应阅读 Message Queue 技术概述，以熟悉 Java 消息规范的 Message Queue 实现、Message Queue 服务的组件以及开发、部署和管理 Message Queue 应用程序的基本过程。

本书的结构

表 1 简要介绍了本手册的内容。

表 1 本手册的内容

部分 / 章节	描述
第 I 部分 “Message Queue 管理简介”	
第 1 章 “管理任务和工具”	介绍 Message Queue 管理任务和工具。
第 2 章 “快速入门教程”	提供一个实用教程，以帮助您熟悉 Message Queue 管理控制台。
第 II 部分 “管理任务”	
第 3 章 “启动代理和客户端”	介绍如何启动 Message Queue 代理和客户端。
第 4 章 “配置代理”	介绍如何设置和读取配置属性，并简要介绍了代理的可配置部分。此外，还介绍了如何设置文件或数据库以执行持久性功能。
第 5 章 “管理代理”	介绍代理管理任务。
第 6 章 “管理物理目标”	介绍与物理目标相关的管理任务。
第 7 章 “管理安全性”	介绍与安全相关的任务，例如管理密码文件、验证、授权和加密。
第 8 章 “管理受管理对象”	介绍对象存储，并说明如何执行与受管理对象 （连接工厂和目标）相关的任务。
第 9 章 “使用代理群集”	介绍如何设置和管理 Message Queue 代理群集。
第 10 章 “监视消息服务器”	介绍如何设置和使用 Message Queue 监视工具。

表 1 本手册的内容 （续）

部分 / 章节	描述
第 11 章 “分析和调整消息服务”	介绍分析和优化消息服务器性能的技术。
第 12 章 “问题疑难解答”	提供有关确定 Message Queue 常见问题的原因以及可以采取哪些措施来解决这些问题的建议。
第 III 部分 “参考”	
第 13 章 “命令行参考”	提供 Message Queue 命令实用程序的语法和说明。
第 14 章 “代理属性参考”	列出并说明可用来配置代理的属性。
第 15 章 “物理目标属性参考”	列出并说明可用来配置物理目标的属性。
第 16 章 “受管理对象属性参考”	列出并说明可用来配置受管理对象 （连接工厂和目标）的属性。
第 17 章 “JMS 资源适配器属性参考”	列出并说明可用来配置 Message Queue 资源适配器 （与应用服务器一起使用）的属性。
第 18 章 “度量参考”	列出并说明 Message Queue 代理生成的度量。
第 IV 部分 “附录”	
附录 A “Message Queue 数据在特定平台上的位置”	列出 Message Queue 文件在每个支持的平台上的位置。
附录 B “Message Queue 接口的稳定性”	说明不同 Message Queue 接口的稳定性。
附录 C “HTTP/HTTPS 支持”	介绍如何设置和使用用于 Message Queue 通信的超文本事务处理协议 (Hypertext Transaction protocol, HTTP)。
附录 D “常用命令实用程序命令”	列出一些常用的 Message Queue 命令实用程序 (imqcmd) 命令。

本书使用的约定

此节介绍本手册中使用的约定。

文本约定

表 2 概述了本手册中使用的文本约定。

表 2 文本约定

格式	描述
Italics (斜体文本)	斜体文本表示占位符，可以使用相应的子句或值替换斜体文本。斜体文本还可用于表示保留未译的关键词或短语。
monospace （等宽文本）	等宽文本表示示例代码、在命令行输入的命令、目录、文件、路径名、错误消息文本、类名、方法名、（包括签名中的所有元素）、软件包名、保留字和 URL。
[]	方括号用来表示命令行语法语句中的可选值。
全部大写文本	全部大写文本表示环境变量 (如 IMQ_HOME) 或首字母缩略词 (如 JMS, GIF、 HTML 等）。
新词术语强调	新词或术语以及要强调的词。
《书名》	书名。
键名 + 键名	用加号连接的一组同时按下的键：Ctrl+A 表示同时按下这两个键。
键名 - 键名	用连字符连接的一组依次按下的键：Esc-S 表示先按 Esc 键，然后松开它，再按 S 键。

目录变量约定

Message Queue 使用三个目录变量；它们的设置方法因平台而异。表 3 介绍了这些变量以及如何在 Solaris™、 Linux 和 Windows 平台上使用这些变量。

注 在本手册中，所显示的目标变量并没有采用通常的特定于平台的语法（例如 UNIX 上的 $IMQ_HOME）。路径名通常采用 UNIX 目录分隔符表示法 (/)。

表 3 Message Queue 目录变量

变量	描述
IMQ_HOME	表示 Message Queue 基目录 （根安装目录）： • 在 Solaris 和 Linux 上不使用；这两个平台上没有 Message Queue 基目录。 • 在 Windows 上，由 Message Queue 安装程序进行设置 （默认情况下设置为 `C:\Program Files\Sun\MessageQueue3`）。 • 对于 Solaris 和 Windows 上的 Sun Java System Application Server，设置为 Application Server 基目录下的 `/imq`。
IMQ_VARHOME	存储 Message Queue 临时或动态创建的配置和数据文件的目录；可以设置为指向任何目录的环境变量。 • 在 Solaris 上，默认设置为 `/var/imq`。 • 在 Linux 上，默认设置为 `/var/opt/sun/mq` 目录。 • 在 Windows 上，默认设置为 `IMQ_HOME\var`。 • 对于 Solaris 上的 Sun Java System Application Server Evaluation Edition，默认设置为 `IMQ_HOME/var`。 • 对于 Windows 上的 Sun Java System Application Server，默认设置为 `IMQ_HOME\var`。

IMQ_JAVAHOME	Message Queue 可执行文件所需的 Java™ 运行时 (JRE) 的位置。默认设置为按显示的顺序在下列位置中查找，但用户可以选择将它设置为驻留所需的 JRE 的位置。
	• 在 Solaris 8 或 9 上：
	`/usr/jdk/entsys-j2se`
	`/usr/jdk/jdk1.5.*`
	`/usr/jdk/j2sdk1.5.*`
	`/usr/j2se`
	• 在 Solaris 10 上：
	`/usr/jdk/entsys-j2se`
	`/usr/java`
	`/usr/j2se`
	• 在 Linux 上：
	`/usr/jdk/entsys-j2se`
	`/usr/java/jre1.5.*`
	`/usr/java/jdk1.5.*`
	`/usr/java/jre1.4.2*`
	`/usr/java/j2sdk1.4.2*`
	• 在 Windows 上：
	`IMQ_HOME\jre*`

相关文档

本节所列的信息资源是对本手册的补充，它们提供了有关 Message Queue 的进一步信息。

Message Queue 文档集

Message Queue 文档集包括表 4 中列出的文档。

表 4 Message Queue 文档集

文档	读者	描述
Message Queue Installation Guide	开发者和管理员	介绍如何在 Solaris、 Linux 和 Windows 平台上安装 Message Queue 软件。
Message Queue 发行说明	开发者和管理员	包含新功能、局限性、已知错误以及技术说明的介绍。
Message Queue 技术概述	开发者和管理员	介绍了 Message Queue 的概念、功能和组件。
Message Queue 管理指南	管理员和开发者	提供使用 Message Queue 管理工具执行管理任务所需的背景和信息。
Message Queue Developer's Guide for Java Clients	开发者	提供有关使用 JMS 和 SOAP/JAXM 规范的 Message Queue 实现开发 Java 客户端程序的信息。
Message Queue Developer's Guide for C Clients	开发者	提供有关使用 Message Queue 消息服务的 C 应用程序编程接口 (C-API) 开发 C 客户端程序的信息。

Java 消息服务规范

Message Queue 消息服务符合 **Java 消息服务规范**中介绍的 Java 消息服务 (Java Message Service, JMS) 应用程序编程接口。可以在以下 URL 找到该文档：

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

联机帮助

Message Queue 命令行实用程序有联机帮助；有关详细信息，请参见第 13 章 “命令行参考”。 Message Queue 图形用户界面 (Graphical User Interface, GUI) 管理工具 （即管理控制台）也包括一个与上下文相关的联机帮助工具；请参见第 37 页上的 “管理控制台联机帮助”。

JavaDoc

表 5 中显示的位置提供了 JavaDoc 格式的 JMS 和 Message Queue API 文档：可以在任何 HTML 浏览器 （如 Netscape 或 Internet Explorer）中查看此文档。它包括标准 JMS API 文档以及特定于 Message Queue 的 API。

表 5 JavaDoc 位置

平台	位置
Solaris	`/usr/share/javadoc/imq/index.html`
Linux	`/opt/sun/mq/javadoc/index.html/`
Windows	`IMQ_HOME/javadoc/index.html`

示例客户端应用程序

安装 Message Queue 时会自动安装一个目录，其中包含几个示例客户端应用程序。有关因特定平台而异的准确位置，请参见附录 A “Message Queue 数据在特定平台上的位置”。位于该目录及其每个子目录中的 `README` 文件提供了有关示例应用程序的描述性信息。

相关的第三方 Web 站点

在适当的情况下，本手册引用了一些提供更多相关信息的第三方 URL。

注 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。

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。

提出意见时您还需要在表格中输入文件的英文文件号码和标题。本文件的英文文件号码是 819-2571，文件标题为 Sun Java System Message Queue 3 Administration Guide。

Message Queue 管理简介

第 1 章

管理任务和工具

本章概述了 Sun Java™ System Message Queue 管理任务和用于执行这些任务的工具，其中重点介绍了命令行管理实用程序的常用功能。本章包含以下各节：

管理任务

要执行的典型管理任务取决于 Message Queue 运行环境的性质。开发和测试 Message Queue 应用程序的软件开发环境的要求与部署这些应用程序 （以完成有用的工作）的生产环境不同。以下各节概述了这两种不同类型环境的典型管理要求。

开发环境中的管理

在开发环境中强调的是灵活性。 Message Queue 消息服务器主要用于测试处于开发过程中的应用程序。管理任务通常非常少，一般由程序员来管理自己的系统。此类环境通常具有以下特点：

- 启动代理只是为了用于测试
- 在客户端代码中实例化受管理对象，而不是以管理方式创建它们
- 自动创建的目标
- 文件系统对象存储
- 基于文件的持久性
- 基于文件的用户系统信息库

- 多代理群集中没有主代理

生产环境中的管理

在生产环境中，由于必须可靠地部署和运行应用程序，因此管理更为重要。要执行的管理任务取决于消息传送系统的复杂性以及它必须支持的应用程序的复杂性。这些任务可以分为两个普通类别：设置操作和维护操作。

设置操作

生产环境中的管理设置操作通常包括下面的部分或全部操作：

管理员安全性

- 设置默认管理用户 (admin) 的密码 （第 126 页上的 “更改默认的管理员密码”）
- 控制个人或组对管理连接服务 （第 132 页上的 “用于连接服务的访问控制”）和停用消息队列 （第 134 页上的 “对物理目标的访问控制”）的访问权限
- 控制管理组对基于文件的用户系统信息库或轻量目录访问协议 (Lightweight Directory Access Protocol, LDAP) 用户信息库的访问权限 （第 124 页上的 “组”、第 128 页上的 “为管理员设置访问控制”）

一般安全性

- 管理基于文件的用户系统信息库的内容 （第 125 页上的 “填充和管理用户系统信息库”），或配置代理以使用现有的 LDAP 用户系统信息库 （第 127 页上的 “编辑实例配置文件”）
- 控制单个用户或组被授权执行的操作 （第 129 页上的 “授权用户：访问控制属性文件”）
- 使用安全套接字层 (Secure Socket Layer, SSL) 设置加密服务 （第 135 页上的 “使用基于 SSL 的服务”）

受管理对象

- 设置和配置 LDAP 对象存储 （第 148 页上的 “LDAP 服务器对象存储”）
- 创建连接工厂和目标 （第 157 页上的 “添加受管理对象”）

代理群集

- 创建群集配置文件 （第 166 页上的 “使用群集配置文件”）
- 指定主代理 （第 170 页上的 “主代理”）

持久性

- 配置代理以使用持久性存储 （第 86 页上的 “配置持久性数据存储”）

内存管理

- 设置目标的配置属性以优化其内存使用 （第 114 页上的 “更新物理目标属性”、第 15 章 “物理目标属性参考”）

维护操作

由于应用程序性能、可靠性和安全性是生产环境中至关重要的因素，因此，必须通过持续的管理维护操作来严格监视和控制消息服务器资源，这些操作包括：

代理管理和调整

- 使用代理度量来调整和重新配置代理 （第 11 章 “分析和调整消息服务”）
- 管理代理内存资源 （第 74 页上的 “路由服务”）
- 创建和管理代理群集以平衡消息负载 （第 9 章 “使用代理群集”）
- 恢复出现故障的代理 （第 64 页上的 “启动代理”）

受管理对象

- 调整连接工厂属性以确保客户端应用程序能够正常运行 （第 150 页上的 “连接工厂属性”）
- 监视和管理物理目标 （第 6 章 “管理物理目标”）
- 控制用户对目标的访问权限 （第 134 页上的 “对物理目标的访问控制”）

客户端管理

- 监视和管理长期订阅 （请参见第 104 页上的 “管理长期订阅”）
- 监视和管理事务 （请参见第 105 页上的 “管理事务”）

管理工具

Message Queue 管理工具分为两类：

- 命令行实用程序
- 图形管理控制台

命令行实用程序

所有 Message Queue 实用程序都可以通过命令行接口访问。实用程序命令具有通用的格式、语法约定和选项。其中包括以下内容：

- **代理实用程序** (`imqbrokerd`) 用于启动代理并指定它们的配置属性，包括将它们连接成一个**群集**。
- **命令实用程序** (`imqcmd`) 用于控制代理及其资源，并管理物理目标。
- **对象管理器实用程序** (`imqobjmgr`) 用于管理对象存储中与提供者无关的**受管理对象**，可通过 Java 命名和目录接口 (Java Naming and Directory Interface, JNDI) 来访问这些受管理对象。
- **数据库管理器实用程序** (`imqdbmgr`) 用于为持久性存储器创建和管理符合 Java 数据库连接 (Java Database Connectivity, JDBC) 标准的数据库。
- **用户管理器实用程序** (`imqusermgr`) 用于填充基于文件的用户系统信息库，该系统信息库用于用户验证和授权。
- **服务管理器实用程序** (`imqsvcadmin`) 用于将代理作为 Windows 服务进行安装和管理。
- **密钥工具实用程序** (`imqkeytool`) 用于为安全套接字层 (Secure Socket Layer, SSL) 验证生成自签名证书。

有关这些实用程序的用法的详细信息，请参见第 13 章 “命令行参考”。

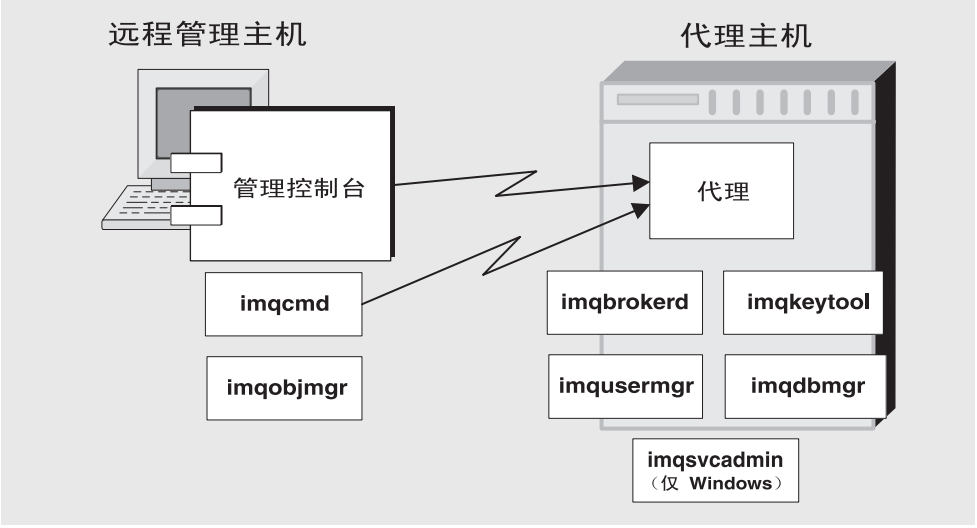
管理控制台

Message Queue **管理控制台**结合了命令实用程序和对象管理器实用程序的某些功能。您可以使用它执行以下任务：

- 连接到代理并对其进行远程控制
- 创建和管理物理目标
- 在 JNDI 对象存储中创建和管理受管理对象

但是，您不能使用管理控制台执行以下任务：启动代理，创建代理群集，管理 JDBC 数据库或用户系统信息库，将代理作为 Windows 服务安装，以及生成 SSL 证书。对于这些任务，您需要使用其他命令行实用程序 （代理、数据库管理器、用户管理器、服务管理器以及密钥工具），这些实用程序不能远程操作，必须与所管理的代理在同一个主机上运行 （见图 1-1）。

图 1-1 本地和远程管理实用程序



有关管理控制台的简明实用的介绍，请参见第 2 章 “快速入门教程”。要获取有关管理控制台用法的更多详细信息，请使用其自身的帮助工具。

第 2 章

快速入门教程

本快速入门教程通过指导您使用 Message Queue 管理控制台（用于管理消息代理及对象存储的图形界面）完成某些基本管理任务，对 Message Queue 的管理进行了简要介绍。本章包含以下各节：

本教程建立了运行符合 JMS 的简单应用程序 `HelloWorldMessageJNDI` 所需的物理目标和受管理对象。该应用程序位于示例应用程序目录（在 Solaris 和 Windows 平台上为 `demo`，在 Linux 平台上为 `examples`；请参见附录 A “Message Queue 数据在特定平台上的位置”）的 `helloworld` 子目录中。在本教程的最后一部分，您将运行此应用程序。

注 您必须安装了 Message Queue 产品才能学习本教程。如有必要，请参见 Message Queue Installation Guide 中的说明。

本教程仅介绍基本信息，因此不能替代文档。通过执行教程中介绍的步骤，您将学会以下操作：

- 启动消息代理
- 连接到代理并使用管理控制台对其进行管理

- 在代理上创建物理目标
- 创建对象存储，并使用管理控制台连接到对象存储
- 将受管理对象添加到对象存储并查看其属性

注 本教程中提供的说明特定于 Windows 平台。在必要情况下，也为使用其他平台的用户提供了补充说明。

某些管理任务无法使用管理控制台完成。您必须使用命令行实用程序按如下所示的过程来执行此类任务：

- 启动代理
- 创建代理群集
- 配置某些物理目标属性
- 管理用于持久性存储器的 JDBC 数据库
- 管理用户系统信息库
- 将代理作为 Windows 服务安装
- 生成 SSL 证书

在本手册的后面几章对上述所有任务进行了介绍。

启动管理控制台

要启动管理控制台，请使用以下方法之一：

- 在 Solaris 上，输入以下命令：

  ```
  /usr/bin/imqadmin
  ```

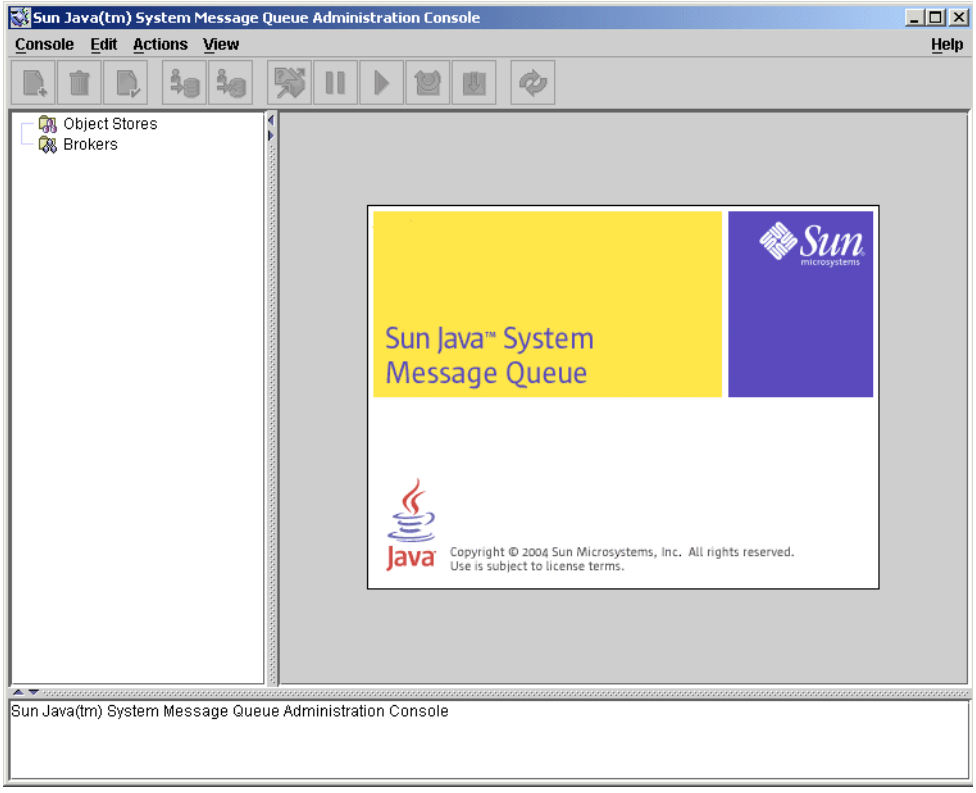
- 在 Linux 上，输入以下命令：

  ```
  /opt/sun/mq/bin/imqadmin
  ```

- 在 Windows 上，选择 “开始” > “程序” > "Sun Microsystems" > "Sun Java System Message Queue 3.6" > "Administration"。

您可能需要等待几秒钟，才会显示 “管理控制台” 窗口 （见图 2-1）。

图 2-1 管理控制台窗口



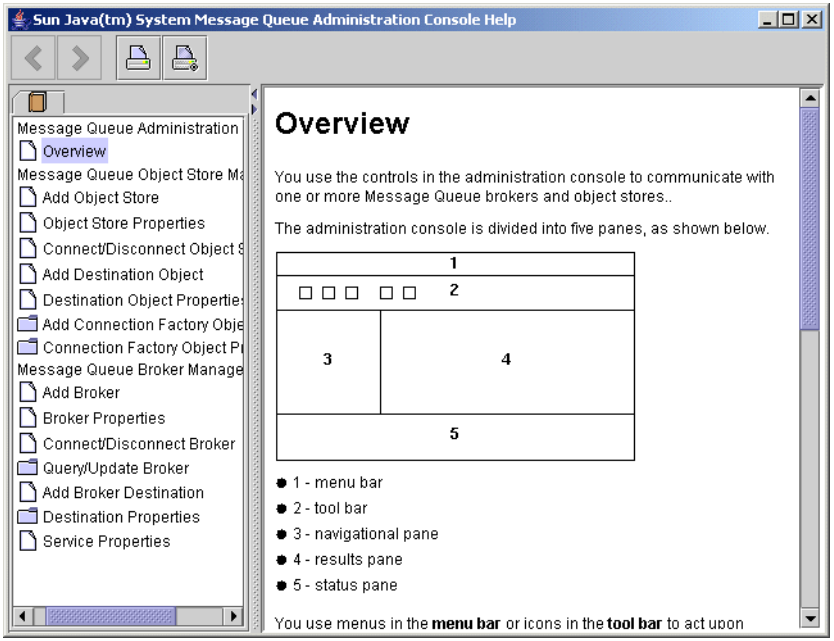
请花几秒钟时间查看一下管理控制台窗口。该窗口包括位于顶部的菜单栏、紧挨在菜单栏下面的工具栏、左侧的浏览窗格、右侧的结果窗格 （目前显示标识 Sun Java System Message Queue 产品的图形）以及底部的状态窗格。

注 使用管理控制台时，可以通过 “查看” 菜单上的 “刷新” 命令来更新任何元素或元素组 （如代理或对象存储列表）的显示。

管理控制台联机帮助

管理控制台提供帮助工具，其中包含有关如何使用控制台执行管理任务的完整信息。要使用帮助工具，请展开菜单栏右端的 “帮助” 菜单，然后选择 “概述”。将显示管理控制台的 “帮助” 窗口 （图 2-2）。

图 2-2 管理控制台帮助窗口



“帮助”窗口左侧的浏览窗格将主题划分为三个区域：Message Queue 管理控制台、 Message Queue 对象存储管理和 Message Queue 代理管理。每个区域中都包含若干文件和文件夹。文件夹为包含多个选项卡的对话框提供帮助，而文件则用于简单的对话框或单个选项卡。在浏览窗格中选择某个项之后，将在右侧的结果窗格中显示该项的内容。选中 “概述”项之后，结果窗格中会显示管理控制台窗口的概貌，并对窗口中的每个窗格进行了标识，如图所示。

使用管理控制台执行的第一个任务是创建代理引用。但在开始之前，请查阅 “帮助”窗口以了解相关信息。在 “帮助”窗口的浏览窗格中单击 “添加代理”项；结果窗格的内容将改为显示相应的文本，该文本说明了添加代理后的情况以及 “添加代理”对话框中每个字段的用途。请仔细阅读帮助文本，然后关闭 “帮助”窗口。

使用代理

本节说明如何使用管理控制台连接和管理消息代理。

启动代理

使用管理控制台无法启动代理，而应使用以下方法之一：

- 在 Solaris 上，输入以下命令：

  ```
  /usr/bin/imqbrokerd
  ```

- 在 Linux 上，输入以下命令：

  ```
  /opt/sun/mq/bin/imqbrokerd
  ```

- 在 Windows 上，选择 “开始” > “程序” > "Sun Microsystems" > "Sun Java System Message Queue 3.6" > "Message Broker"。

如果使用 Windows 的 “开始”菜单，则会出现一个显示类似以下文本行的命令窗口，以表明代理已经准备就绪：

```
正在加载持久性数据 . . .
代理 "imqbroker@stan:7676" 就绪。
```

重新激活 “管理控制台”窗口。您现在可以将代理添加到控制台并连接到代理。在管理控制台中添加对代理的引用之前不必启动代理，但在连接代理之前必须启动它。

将代理添加到管理控制台

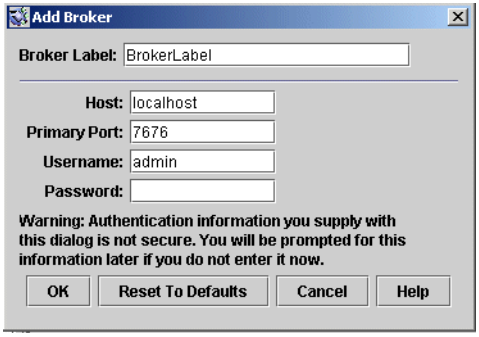
添加代理即在管理控制台中创建了对该代理的引用。添加代理之后，您就可以连接到它。

➤ 将代理添加到管理控制台

1. 在 “管理控制台”窗口的浏览窗格中单击 “代理”项，然后从 “操作”菜单中选择 “添加代理”。

   或者，也可以在 “代理”上单击鼠标右键，然后从弹出的上下文菜单中选择 “添加代理”。无论在哪种情况下，都会显示 “添加代理”对话框 （图 2-3）。

图 2-3 “添加代理”对话框



2. 在“代理标签”字段中输入代理的名称。

   这样便提供了一个在管理控制台中标识代理的标签。

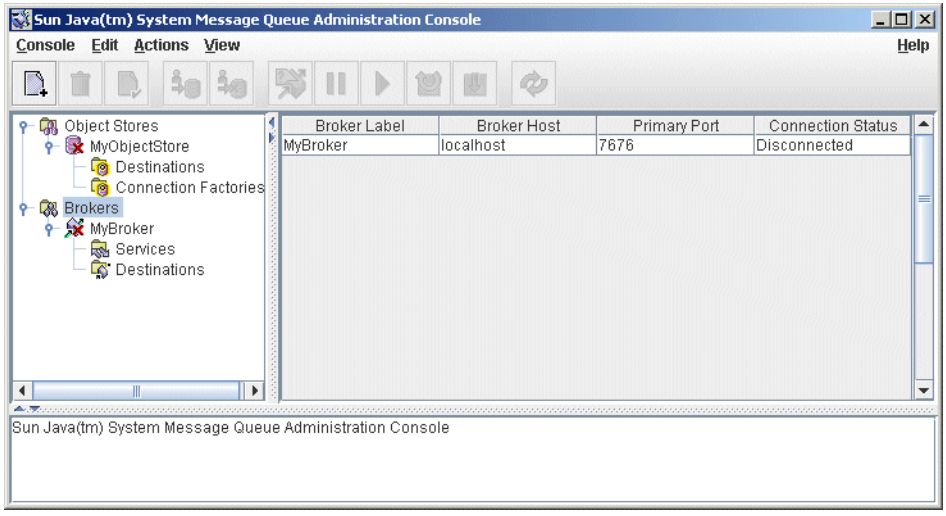
   请记下在对话框中指定的默认主机名 (`localhost`) 和主端口 (`7676`)。以后在配置客户端用于创建此代理连接的连接工厂时，您必须指定这些值。

   在本练习中，请在“代理标签”字段中键入名称 `MyBroker`。将“密码”字段保留为空；如果在连接时指定密码，则密码将会更安全。

3. 单击“确定”添加代理并关闭该对话框。

   新代理将显示在浏览窗格中的“代理”之下，如图 2-4 所示。代理图标上的红色 X 表示该代理当前未连接到管理控制台。

图 2-4 管理控制台窗口中显示的代理



添加代理之后，可以使用 “操作” 菜单 （或弹出的上下文菜单）上的 “属性” 命令来显示 “代理属性” 对话框 （类似于图 2-3 中显示的 “添加代理” 对话框），以便查看或修改代理的任意属性。

连接到代理

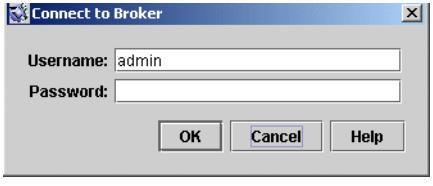
将代理添加到管理控制台之后，即可连接该代理。

➤ **连接到代理**

1. 在 “管理控制台” 窗口的浏览窗格中单击该代理的名称，然后从 “操作” 菜单中选择 “连接代理”。

   或者，也可以在该代理的名称上单击鼠标右键，然后从弹出的上下文菜单中选择 “连接代理”。无论在哪种情况下，都会显示 “连接代理” 对话框 （图 2-5）。

图 2-5 “连接代理” 对话框



2. 输入连接代理时要使用的用户名和密码。

   该对话框最初显示默认用户名 admin。在实际环境中，应尽快设置安全的用户名和密码 （请参见第 121 页上的 “验证用户”）；在本练习中，使用默认值即可。

   与默认用户名关联的密码也是 admin；请在对话框的 “密码”字段中键入它。这样，您就能以管理权限连接到该代理。

3. 单击 “确定”连接代理并关闭对话框。

连接到代理之后，可以使用 “操作”菜单 （或上下文菜单）上的命令对选定代理执行下列操作：

- “暂停代理”会暂停正在运行的代理的操作。
- “恢复代理”会恢复已被暂停的代理的操作。
- “重新启动代理”会重新初始化并重新启动代理。
- “关闭代理”会终止代理的操作。
- “查询 / 更新代理”会显示或修改代理的配置属性。
- “断开代理”会终止代理与管理控制台之间的连接。

查看连接服务

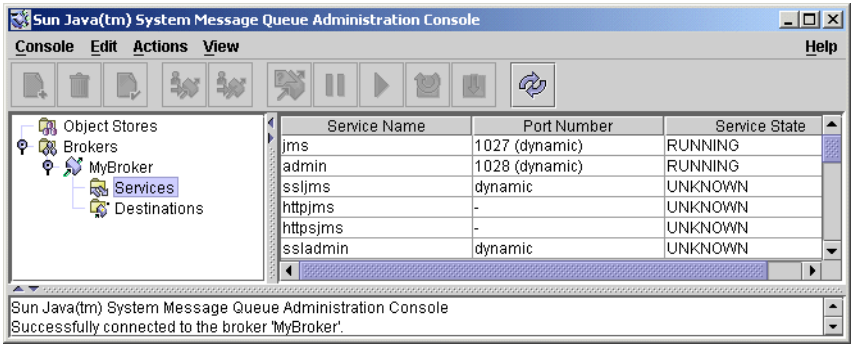
根据代理提供的连接服务及其支持的物理目标可以区分代理。

➤ 查看可用连接服务

1. 在 “管理控制台”窗口浏览窗格的代理名称下选择 “服务”。

   结果窗格中会显示可用服务的列表 （见图 2-6），其中显示了每个服务的名称、端口号及当前状态。

图 2-6 查看连接服务



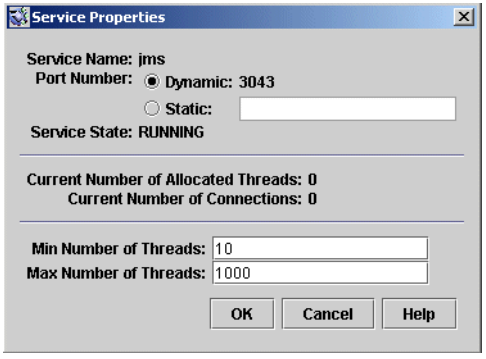
2. 通过在结果窗格中单击某个服务的名称来选择该服务。

   在本练习中，请选择名称 jms。

3. 从 “操作” 菜单中选择 “属性”。

   将显示 “服务属性” 对话框 （图 2-7）。可以使用此对话框为该服务指定静态端口号，并更改为其分配的最大线程数及最小线程数。

图 2-7 “服务属性” 对话框



   在本练习中，不要更改连接服务的任何属性。

4. 单击 “确定” 接受新的属性值并关闭对话框。

“操作” 菜单中还包含用于暂停和恢复服务的命令。但是，如果选择管理服务并展开 “操作” 菜单，您会看到 “暂停服务” 命令已被禁用。这是因为管理服务是管理控制台与代理之间的链接：如果暂停该服务，您将无法再访问该代理。

使用物理目标

物理目标是消息代理中的一个位置，从消息生成方收到的消息先保存在此处，稍后再传送给一个或多个消息使用方。根据所使用的**消息传送域**，目标可以分为以下两种：**队列**（点对点域）和**主题**（发布 / 订阅域）。有关消息传送域及其关联目标的进一步讨论，请参见 Message Queue 技术概述。

创建物理目标

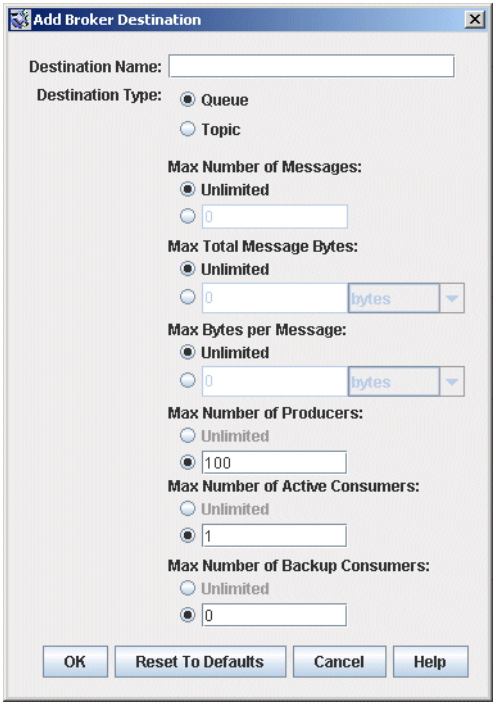
默认情况下对消息代理进行如下配置：只要消息生成方或使用方试图访问不存在的目标，即自动创建新的物理目标。在软件开发环境中测试客户端代码时，这种**自动创建的目标**非常便于使用。但在生产设置中，建议您禁用目标自动创建功能，而要求所有目标都由管理员明确创建。以下过程说明了如何将这种**自动创建的目标**添加到代理。

➤ 将物理目标添加到代理

1. 在“管理控制台”窗口的浏览窗格中，单击代理名称下的“目标”项，然后从“操作”菜单中选择“添加代理目标”。

   或者，也可以在“目标”上单击鼠标右键，然后从弹出的上下文菜单中选择“添加代理目标”。无论在哪种情况下，都会显示“添加代理目标”对话框（图 2-8）。

图 2-8 “添加代理目标”对话框



2. 在“目标名称”字段中输入物理目标的名称。

   请记下为目标指定的名称，以后在创建与此物理目标对应的受管理对象时需要用到此名称。

   在本练习中，请键入名称 `MyQueueDest`。

3. 选择“队列”或“主题”单选按钮，以指定要创建的目标的类型。

   在本练习中，请选择“队列”（如果它未处于选中状态）。

4. 单击“确定”添加物理目标并关闭对话框。

   新目标将显示在结果窗格中。

查看物理目标属性

可以使用管理控制台 “操作”菜单上的 “属性”命令来查看或修改物理目标的属性。

➤ 查看或修改物理目标的属性

1. 在 “管理控制台”窗口的浏览窗格中，选择代理名称下的 “目标”。

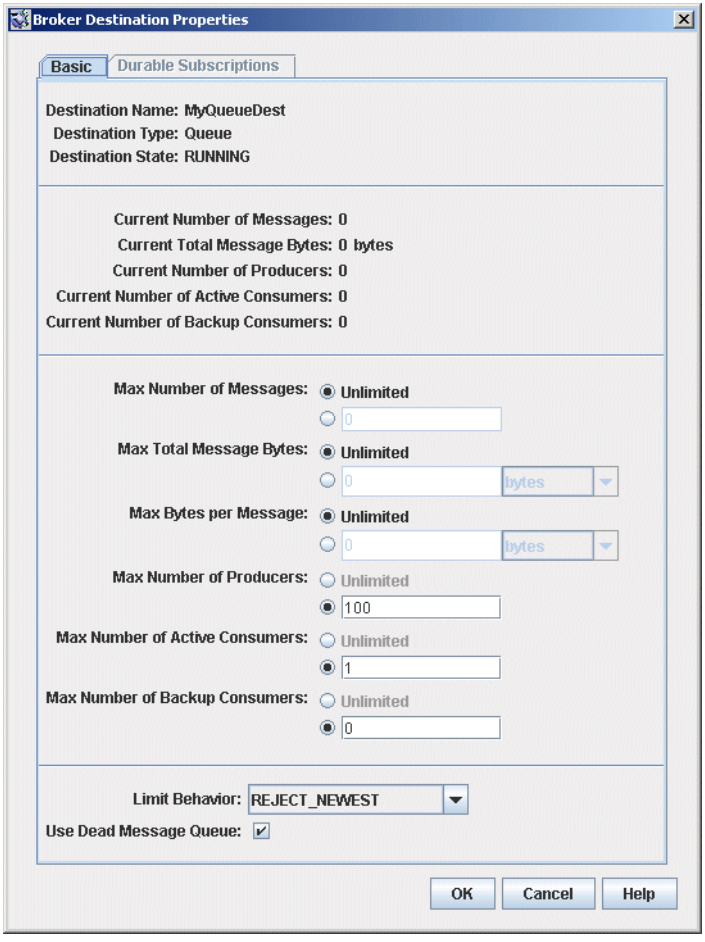
   结果窗格中会显示可用物理目标的列表，其中包括每个目标的名称、类型及当前状态。

2. 通过在结果窗格中单击某个物理目标的名称来选择该物理目标。

3. 从 “操作”菜单中选择 “属性”。

   将显示 “代理目标属性”对话框 （图 2-9），其中显示了有关选定物理目标的当前状态和配置信息。可以使用此对话框更改各种配置属性，如该目标可以容纳的消息、生成方和使用方的最大数量。

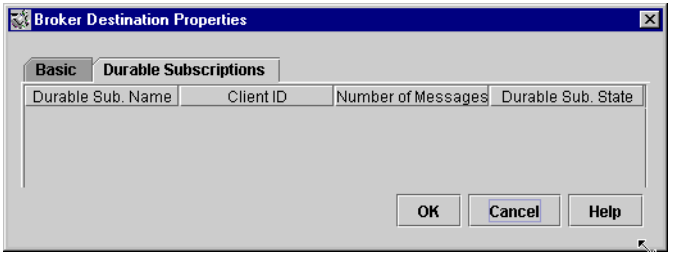
图 2-9 “代理目标属性”对话框



在本练习中，不要更改目标的任何属性。

对于主题目标，“代理目标属性”对话框中包含一个额外的选项卡“长期订阅”。单击此选项卡会显示“长期订阅”面板（图 2-10），其中列出了当前与给定主题关联的所有长期订阅的信息。

图 2-10 “长期订阅”面板



可以使用 “长期订阅”面板的 “清除”和 “删除”按钮执行下列操作：

- 清除与长期订阅关联的所有待处理消息
- 从主题中删除长期订阅

“长期订阅”选项卡对于队列目标是禁用的。

4. 单击 “确定”接受新的属性值并关闭对话框。

清除物理目标中的消息

清除物理目标中的消息会删除与目标关联的所有待处理消息，从而使目标为空。

➤ **清除物理目标中的消息**

1. 在 “管理控制台”窗口的浏览窗格中，选择代理名称下的 “目标”。

   结果窗格中会显示可用物理目标的列表，其中包括每个目标的名称、类型及当前状态。

2. 通过在结果窗格中单击某个目标的名称来选择该目标。
3. 从 “操作”菜单中选择 “清除消息”。

   将显示一个确认对话框，要求您确认是否要继续执行该操作。

4. 单击 “是”确认操作并关闭确认对话框。

删除物理目标

删除目标会清除它的所有消息，然后销毁目标本身，从而将其从所属的代理中永久删除。

➤ **删除物理目标**

1. 在 “管理控制台”窗口的浏览窗格中，选择代理名称下的 “目标”。

   结果窗格中会显示可用目标的列表，其中包括每个目标的名称、类型及当前状态。

2. 通过在结果窗格中单击某个目标的名称来选择该目标。

3. 从 “编辑”菜单中选择 “删除”。

   将显示一个确认对话框，要求您确认是否要继续执行该操作。

4. 单击 “是”确认操作并关闭确认对话框。

   在本练习中，不要删除您之前创建的目标 `MyQueueDest`，请单击 “否”关闭确认对话框，而不执行删除操作。

使用对象存储

对象存储用于存储 Message Queue **受管理对象**，这类对象封装特定于具体 Message Queue 提供者的实现及配置信息。对象存储可以是轻量目录访问协议 (Lightweight Directory Access Protocol, LDAP) 目录服务器，也可以是本地文件系统中的目录。

虽然可以在客户端应用程序的代码中直接实例化和配置受管理对象，但一般最好是由管理员来创建和配置这些对象并将其存储在对象存储中，以便客户端应用程序可以使用 Java 命名和目录接口 (Java Naming and Directory Interface, JNDI) 来访问它们。这使得客户端代码本身可以与提供者无关。

添加对象存储

虽然可以使用管理控制台来**管理**对象存储，但不能使用它来**创建**对象存储；将作为对象存储的 LDAP 服务器或文件系统目录必须已事先存在。之后，可以将此现有对象存储添加到管理控制台中，同时创建对它的引用，以便可以在控制台中对它执行操作。

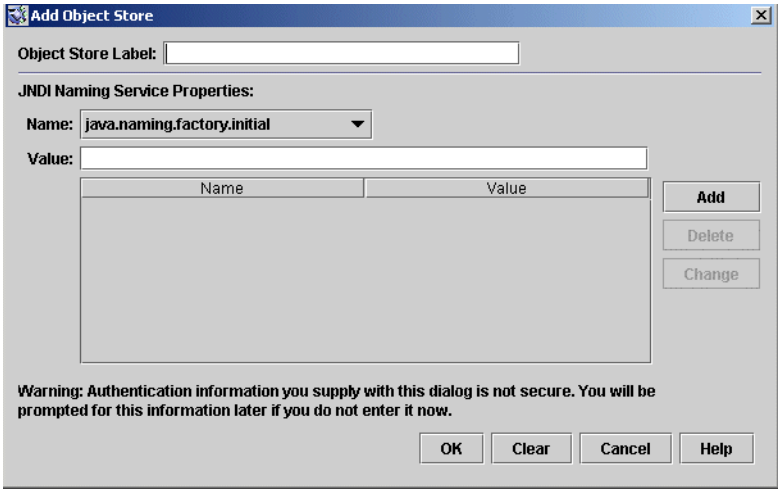
注 本章中使用的样例应用程序假定对象存储位于 C 驱动器上一个名为 `Temp` 的目录中。如果 C 驱动器上尚不存在名为 `Temp` 的文件夹，请先创建该文件夹，然后再继续完成以下练习。（在 Windows 以外的平台上，可以使用 `/tmp` 目录，该目录应该已经存在。）

➤ 将对象存储添加到管理控制台

1. 在 “管理控制台” 窗口的浏览窗格中单击 “对象存储” 项，然后从 “操作” 菜单中选择 “添加对象存储”。

   或者，也可以在 “对象存储” 上单击鼠标右键，然后从弹出的上下文菜单中选择 “添加对象存储”。无论在哪种情况下，都会显示 “添加对象存储” 对话框（图 2-11）。

图 2-11 “添加对象存储” 对话框



2. 在 “对象存储标签” 字段中输入对象存储的名称。

   这样便提供了一个用于在管理控制台中标识该对象存储的标签。

   在本练习中，请键入名称 `MyObjectStore`。

3. 输入要用于查找受管理对象的 JNDI 属性值：

   a. 从 “名称” 下拉菜单中选择要指定的属性名称。

   b. 在 “值” 字段中键入属性的值。

   c. 单击 “添加” 按钮添加指定的属性值。

      该属性及其值将显示在属性概要窗格中。

   重复步骤 a 至 c，根据需要设置任意多个属性。

   在本练习中，请将 `java.naming.factory.initial` 属性设置为：

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

将 `java.naming.provider.url` 属性设置为：

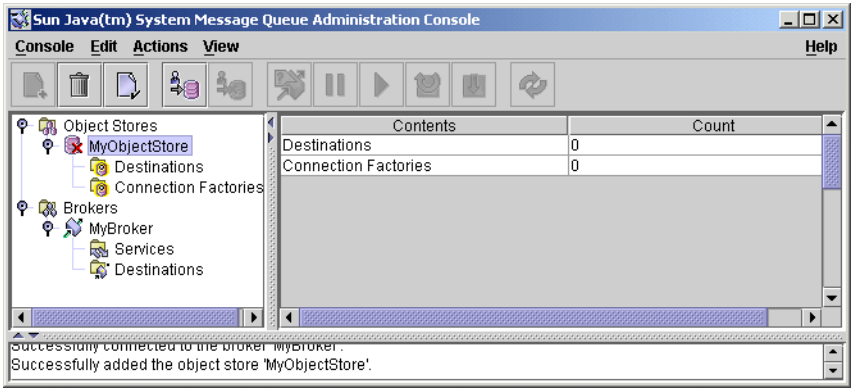
```
file:///C:/Temp
```

（或者，在 Solaris 或 Linux 平台上设置为 `file:///tmp`）。对于文件系统对象存储，只需设置以上属性；有关 LDAP 存储所需的属性值的信息，请参见第 148 页上的 “LDAP 服务器对象存储”。

4. 单击 “确定” 添加对象存储并关闭对话框。

   新的对象存储将显示在浏览窗格中的 “对象存储” 下，如图 2-12 所示。对象存储图标上的红色 X 表示该对象存储当前未连接到管理控制台。

图 2-12 管理控制台窗口中显示的对象存储



在浏览窗格中单击对象存储时，将在结果窗格中列出其内容。由于您尚未将任何受管理对象添加到对象存储，因此对于目标和连接工厂，“计数” 列中都显示 0。

添加对象存储之后，可以使用 “操作” 菜单 （或弹出的上下文菜单）上的 “属性” 命令来显示 “对象存储属性” 对话框 （类似于图 2-11 中显示的 “添加对象存储” 对话框），以便查看或修改对象存储的任意属性。

连接到对象存储

现在，您已经将对象存储添加到管理控制台，必须连接到该对象存储才能向其中添加受管理对象。

➤ **连接到对象存储**

1. 在 “管理控制台”窗口的浏览窗格中单击该对象存储的名称，然后从 “操作”菜单中选择 “连接到对象存储”。

   或者，也可以在该对象存储的名称上单击鼠标右键，然后从弹出的上下文菜单中选择 “连接到对象存储”。无论在哪种情况下，对象存储图标上的红色 X 都会消失，这表示它现在已经连接到管理控制台。

使用受管理对象

将对象存储连接到管理控制台之后，即可向其中添加受管理对象 （连接工厂和目标）。本节将介绍具体操作。

注 管理控制台仅显示 Message Queue 受管理对象。如果对象存储中包含与要添加的受管理对象具有相同查找名称的非 Message Queue 对象，则您在尝试执行添加操作时将会收到一条错误消息。

添加连接工厂

连接工厂供客户端应用程序用来创建代理连接。通过配置连接工厂，您可以控制它所创建的连接的属性。

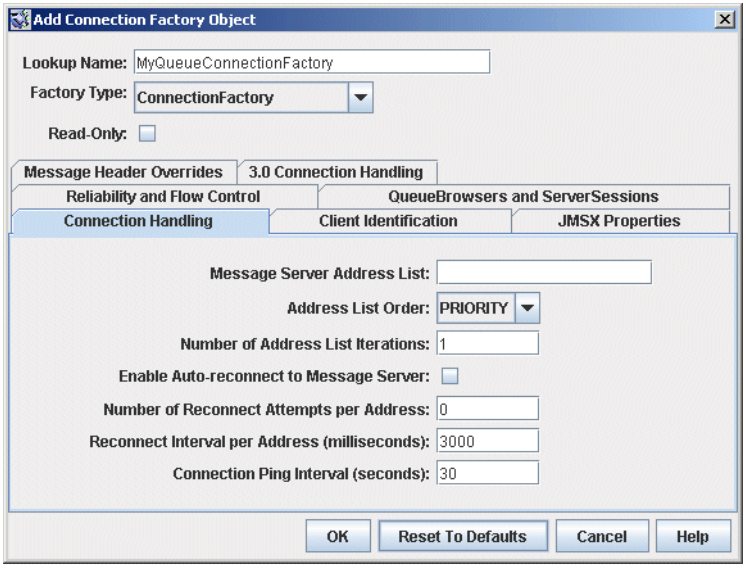
➤ **将连接工厂添加到对象存储**

1. 确保该对象存储已连接到管理控制台 （请参见第 51 页上的 “连接到对象存储”）。

2. 在 “管理控制台”窗口的浏览窗格中，单击对象存储名称下的 “连接工厂”项，然后从 “操作”菜单中选择 “添加连接工厂对象”。

   或者，也可以在 “连接工厂”上单击鼠标右键，然后从弹出的上下文菜单中选择 “添加连接工厂对象”。无论在哪种情况下，都会显示 “添加连接工厂对象”对话框 （图 2-13）。

图 2-13 “添加连接工厂对象”对话框



3. 在“查找名称”字段中输入连接工厂的名称。

   客户端应用程序在通过 JNDI 查找连接工厂时将会使用该名称。

   在本练习中，请键入名称 `MyQueueConnectionFactory`。

4. 从“工厂类型”下拉菜单中选择要创建的连接工厂的类型。

   在本练习中，请选择 "QueueConnectionFactory"。

5. 单击“连接处理”选项卡。

   将显示“连接处理”面板，如图 2-13 所示。

6. 在“消息服务器地址列表”字段中，填入此连接工厂将为其创建连接的代理的地址。

   地址列表可能包含一个代理，也可能包含多个代理（对于代理群集）。对于每个代理，它指定了代理的连接服务、主机名和端口号等信息。要指定的信息的确切性质及语法各不相同，这取决于要使用的连接服务；有关具体信息，请参见第 288 页上的“连接处理”。

   在本练习中，无需在“消息服务器地址列表”字段中键入任何内容，因为样例应用程序 `HelloWorldMessageJNDI` 要求连接工厂使用默认情况下自动为其配置的标准地址列表属性（连接服务 `jms`、主机名 `localhost` 和端口号 `7676`）。

7. 根据需要配置连接工厂的任何其他属性。

   除 “连接处理”外，“添加连接工厂对象”对话框还包含许多其他面板，用于配置连接工厂的不同属性。

   在本练习中，不要更改任何其他设置。但是您可能会发现，依次单击其他选项卡来了解可以指定的配置信息种类很有好处。使用 “帮助”按钮可以了解与这些配置面板的内容有关的详细信息。

8. 如果适用，单击 “只读”复选框。

   这会锁定连接工厂对象的配置属性，使其只具有在创建时指定的值。无论是在客户端代码中通过编程方式，还是在命令行中通过管理方式，都不能覆盖受管理对象的只读属性。

   在本练习中，不要选中 “只读”。

9. 单击 “确定”创建连接工厂、将其添加到对象存储并关闭对话框。

   新的连接工厂将显示在结果窗格中。

添加目标

目标受管理对象表示代理上的物理目标，它使得客户端在向该物理目标发送消息时，可以不必考虑特定于提供者的配置和命名语法。当客户端发送通过受管理对象寻址的消息时，代理会将该消息传送到对应的物理目标 （如果存在）。如果不存在对应的物理目标，则在启用了自动创建功能的情况下，代理会自动创建一个物理目标 （如第 44 页上的 “创建物理目标”中所述），并将消息传送到该目标；否则，它会生成一个错误，指出无法传送该消息。

以下过程说明了如何将目标受管理对象添加到与现有物理目标对应的对象存储中。

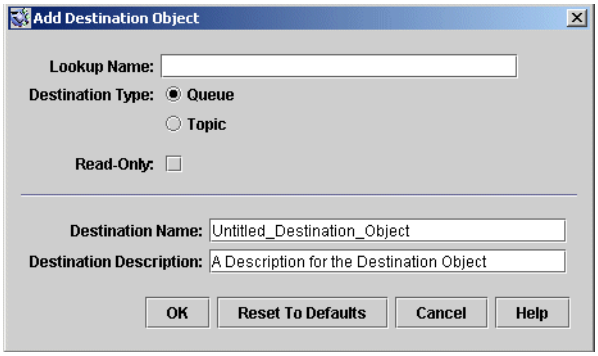
➤ 将目标添加到对象存储

1. 确保该对象存储已经连接到管理控制台 （请参见第 51 页上的 “连接到对象存储”）。

2. 在 “管理控制台”窗口的浏览窗格中，单击对象存储名称下的 “目标”项，然后从 “操作”菜单中选择 “添加目标对象”。

   或者，也可以在 “目标”上单击鼠标右键，然后从弹出的上下文菜单中选择 “添加目标对象”。无论在哪种情况下，都会显示 “添加目标对象”对话框 （图 2-14）。

图 2-14 “添加目标对象”对话框



3. 在“查找名称”字段中输入目标受管理对象的名称。

   客户端应用程序在通过 JNDI 查找目标时将会使用该名称。

   在本练习中，请键入名称 `MyQueue`。

4. 选择“队列”或“主题”单选按钮，以指定要创建的目标对象的类型。

   在本练习中，请选择“队列”（如果它未处于选中状态）。

5. 在“目标名称”字段中输入对应物理目标的名称。

   这是您向代理中添加物理目标时指定的名称（请参见第 44 页上的“使用物理目标”）。

   在本练习中，请键入名称 `MyQueueDest`。

6. 或者，在“目标描述”字段中输入目标的简短描述。

   此字段的内容只是为了方便用户使用，对客户端操作不会产生任何影响。

   在本练习中，您可以删除“目标描述”字段的内容，或者键入一些描述性文本，例如：

   ```
   Example destination for MQ Admin Guide tutorial
   ```

7. 如果适用，单击“只读”复选框。

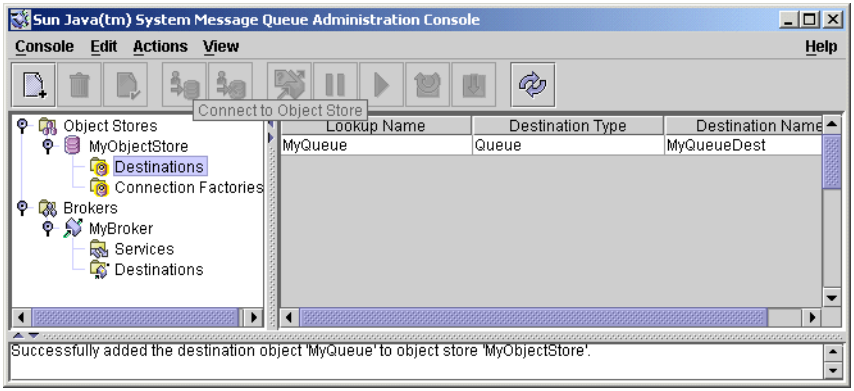
   这会锁定目标对象的配置属性，使其只具有在创建时指定的值。无论是在客户端代码中通过编程方式，还是在命令行中通过管理方式，都不能覆盖受管理对象的只读属性。

   在本练习中，不要选中“只读”。

8. 单击 “确定”创建目标对象、将其添加到对象存储并关闭对话框。

   新的目标对象将显示在结果窗格中，如图 2-15 中所示。

图 2-15 管理控制台中显示的目标对象



查看受管理对象的属性

可以使用管理控制台 “操作”菜单上的 “属性”命令来查看或修改受管理对象的属性。

➤ 查看或修改受管理对象的属性

1. 在 “管理控制台”窗口的浏览窗格中，选择对象存储名称下的 “连接工厂”或 “目标”。

   结果窗格中将显示可用连接工厂或目标受管理对象的列表，其中显示了每一项的查找名称和类型 （对于目标受管理对象，还会显示目标名称）。

2. 通过在结果窗格中单击某个受管理对象的名称来选择该受管理对象。

3. 从 “操作”菜单中选择 “属性”。

   将显示 “连接工厂对象属性”或 “目标对象属性”对话框，它们与 “添加连接工厂对象”（第 53 页上的图 2-13）或 “添加目标对象”（第 55 页上的图 2-14）对话框类似。可以使用此对话框更改选定对象的配置属性。但请注意，您不能更改对象的查找名称；更改此名称的唯一方法是删除该对象，然后添加一个新的具有所需查找名称的受管理对象。

4. 单击 “确定”接受新的属性值并关闭对话框。

删除受管理对象

删除受管理对象会将其从所属的对象存储中永久删除。

➤ **删除受管理对象**

1. 在 “管理控制台”窗口的浏览窗格中，选择对象存储名称下的 “连接工厂”或 “目标”。

   结果窗格中将显示可用连接工厂或目标受管理对象的列表，其中显示了每一项的查找名称和类型 （对于目标受管理对象，还会显示目标名称）。

2. 通过在结果窗格中单击某个受管理对象的名称来选择该受管理对象。

3. 从 “编辑”菜单中选择 “删除”。

   将显示一个确认对话框，要求您确认是否要继续执行该操作。

4. 单击 “是”确认操作并关闭确认对话框。

   在本练习中，不要删除您之前创建的受管理对象 `MyQueue` 或 `MyQueueConnectionFactory`，请单击 “否”关闭确认对话框，而不执行删除操作。

运行样例应用程序

样例应用程序 `HelloWorldMessageJNDI` 是为了与本教程配套使用而提供的。它使用您创建的物理目标和受管理对象：

- 名为 `MyQueueDest` 的队列物理目标
- JNDI 查找名称为 `MyQueueConnectionFactory` 的队列连接工厂受管理对象
- JNDI 查找名称为 `MyQueue` 的队列受管理对象

该代码创建了一个简单的队列发送者和接收者，并发送和接收一条 `Hello World` 消息。

在运行应用程序之前，请打开源文件 `HelloWorldMessageJNDI.java` 并通读代码。该程序很简短，但注释非常详细，您可以很容易地了解它的工作原理。

➤ **运行样例应用程序**

1. 根据您使用的平台，使用以下命令之一将包含 `HelloWorldmessageJNDI` 应用程序的目录作为当前目录：

   - 在 Solaris 上：

```
cd /usr/demo/imq/helloworld/helloworldmessagejndi
```

- 在 Linux 上：

```
cd /opt/sun/mq/examples/helloworld/helloworldmessagejndi
```

- 在 Windows 上：

```
cd IMQ_HOME\demo\helloworld\helloworldmessagejndi
```

您会发现 HelloWorldMessageJNDI.class 文件已经存在。（如果对应用程序进行更改，则必须使用 Message Queue Developer's Guide for Java Clients 中介绍的客户端应用程序编译过程来重新编译它。）

2. 设置 CLASSPATH 变量，以包括含有 HelloWorldMessageJNDI.class 文件的当前目录以及 Message Queue 产品附带的以下 .jar 文件：

```
jms.jar
imq.jar
jndi.jar
fscontext.jar
```

有关设置 CLASSPATH 变量的信息，请参见 Message Queue Developer's Guide for Java Clients。

注 文件 jndi.jar 与 JDK 1.4 捆绑在一起。除非您使用的是早期版本的 JDK，否则无需将此文件添加到 CLASSPATH 中。

3. 根据您使用的平台，通过执行以下命令之一来运行 HelloWorldMessageJNDI 应用程序：

- 在 Solaris 或 Linux 上：

```
% java HelloWorldMessageJNDI file:///tmp
```

- 在 Windows 上：

```
java HelloWorldMessageJNDI
```

如果应用程序成功运行，您会看到代码示例 2-1 中显示的输出：

代码示例 2-1 样例应用程序的输出

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


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


Creating connection to broker.
Connection to broker created.

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

第 II 部分

管理任务

第 3 章

启动代理和客户端

安装 Sun Java System Message Queue 并执行一些准备步骤后，即可开始启动代理和客户端。代理的配置是由一组配置文件控制的，传递给代理实用程序 (`imqbrokerd`) 的命令行选项可以覆盖这些配置文件；有关更多信息，请参见第 4 章 “配置代理”。

本章包含以下各节：

准备系统资源

在启动代理之前，需要执行两项系统级别的准备任务：同步系统时钟，以及 （在 Solaris 或 Linux 平台上）设置文件描述符限制。以下各节介绍了这些任务。

同步系统时钟

在启动任何代理或客户端之前，同步将要与 Message Queue 系统进行交互的所有主机的时钟至关重要。如果使用消息到期 （生存时间）功能，则同步尤为重要。来自未同步时钟的时间戳可能会使消息到期功能无法按预期方式工作，并且可能会使消息无法传送。同步对于代理群集同样至关重要。

应该对系统进行配置以运行时间同步协议，如简单网络时间协议 (Simple Network Time Protocol, SNTP)。通常，Solaris 和 Linux 中的 xntpd 守护进程以及 Windows 中的 W32Time 服务支持时间同步。（有关配置此服务的信息，请参见操作系统文档。）代理运行后，要避免往回设置系统时钟。

设置文件描述符限制

在 Solaris 和 Linux 平台上，运行客户端或代理的 shell 对进程可以使用的文件描述符数量的限制不是很严格。在 Message Queue 中，客户端创建的每个连接或代理接受的每个连接都使用其中一个文件描述符。每个具有持久性消息的物理目标也使用文件描述符。

因此，文件描述符限制限定了代理或客户端可以具有的连接数。默认情况下，Solaris 最多可具有 256 个连接， Linux 最多可具有 1024 个连接。（在实践中，由于将文件描述符用于持久性，因此连接限制实际上低于此值。）如果您需要的连接数高于此值，则必须提高将要执行客户端或代理的每个 shell 中的文件描述符限制。有关如何执行此操作的信息，请参见 ulimit 手册页。

启动代理

可以使用 Message Queue 命令行实用程序或 Windows 的 “开始”菜单以交互方式启动代理，也可以安排在系统启动时自动启动代理。以下各节介绍了操作方法。

以交互方式启动代理

可以使用代理实用程序 (imqbrokerd) 从命令行中以交互方式启动代理。（或者，也可以从 Windows 的 “开始”菜单中启动代理。）不能使用管理控制台 (imqadmin) 或命令实用程序 (imqcmd) 启动代理；只有在代理已经运行后才能使用这些工具。

在 Solaris 和 Linux 平台上，代理实例必须始终由最初启动该实例的用户启动。每个代理实例都有其自身的一组配置属性和基于文件的消息存储。首次启动代理实例时， Message Queue 将使用用户的文件创建模式掩码 (umask) 来设置该代理实例的配置信息和持久性数据所在目录的权限。

默认情况下，代理实例具有实例名称 imqbroker。要从命令行中使用此名称和默认配置启动代理，只需使用以下命令：

```
imqbrokerd
```

此命令使用端口映射器的默认端口 7676 启动本地计算机上名为 imqbroker 的代理实例 （请参见第 72 页上的 “端口映射器”）。

要指定非默认的实例名称，请使用 imqbrokerd 命令的 -name 选项。以下命令启动实例名称为 myBroker 的代理：

```
imqbrokerd -name myBroker
```

在 imqbrokerd 命令行中，还可以使用其他选项来控制代理操作的各个方面。下面的示例使用 -tty 选项向命令窗口发送错误消息和警告 （标准输出）：

```
imqbrokerd -name myBroker -tty
```

也可以在命令行中使用 -D 选项来覆盖在代理实例配置文件 (config.properties) 中指定的属性值。下面的示例设置 imq.jms.max_threads 属性，将 jms 连接服务可用的最大线程数提高到 2000：

```
imqbrokerd -name myBroker -Dimq.jms.max_threads=2000
```

有关 imqbrokerd 命令的语法、子命令和选项的完整信息，请参见第 242 页上的 “代理实用程序”。要了解此信息的简要概述，请输入以下命令：

```
imqbrokerd -help
```

注 如果您具有 Sun Java System Message Queue Platform Edition 许可证，则可以使用 imqbrokerd 命令的 -license 选项来激活试用版 Enterprise Edition 许可证，它允许您试用 Enterprise Edition 功能 90 天。指定 try 作为许可证名称：

```
imqbrokerd -license try
```

每次启动代理时都必须使用此选项，否则，代理将默认使用标准 Platform Edition 许可证。

自动启动代理

可以将代理设置为在系统启动时自动启动，而不是在命令行中明确启动它。具体操作方法取决于运行代理的平台 （Solaris、 Linux 或 Windows）。

在 Solaris 和 Linux 上自动启动

在 Solaris 和 Linux 系统上，使代理可以自动启动的脚本在 Message Queue 安装期间放在 /etc/rc* 目录树中。要允许使用这些脚本，您必须按如下所示编辑配置文件 /etc/imq/imqbrokerd.conf (Solaris) 或 /etc/opt/sun/mq/imqbrokerd.conf (Linux)：

- 要在系统启动时自动启动代理，请将 AUTOSTART 属性设置为 YES。
- 要使代理在异常退出后自动重新启动，请将 RESTART 属性设置为 YES。
- 要为代理设置启动命令行参数，请为 ARGS 属性指定一个或多个值。

在 Windows 上自动启动

要在 Windows 系统启动时自动启动代理，必须将代理定义为 Windows 服务。代理将在系统启动时启动并在后台运行，直到系统关闭。因此，不要使用 imqbrokerd 命令启动代理，除非您希望启动其他实例。

系统最多只能有一个作为 Windows 服务运行的代理。任务管理器将此类代理作为两个可执行进程列出：

- 本地 Windows 服务包装 imqbrokersvc.exe
- 正在运行代理的 Java 运行时

在 Windows 系统上，可以在安装 Message Queue 时将代理作为服务安装。安装后，可以使用服务管理器实用程序 (imqsvcadmin) 执行以下操作：

- 将代理作为 Windows 服务添加
- 确定代理服务的启动选项
- 删除作为 Windows 服务运行的代理

要将启动选项传递给代理，请使用 imqsvcadmin 命令的 -args 参数。其工作方式与 imqbrokerd 命令的 -D 选项相同，如第 64 页上的 “启动代理”中所述。可以照常使用命令实用程序 (imqcmd) 来控制代理操作。

有关 imqsvcadmin 命令的语法、子命令和选项的完整信息，请参见第 257 页上的 “服务管理器实用程序”。

重新配置代理服务

重新配置作为 Windows 服务安装的代理的过程如下：

➤ **重新配置作为 Windows 服务运行的代理**

1. 停止服务。

a. 在 Windows “开始”菜单的 “设置”子菜单中选择 “控制面板”。

b. 打开 “管理工具”控制面板。

c. 使用以下方法运行 “服务”工具：选择该工具的图标，然后从 “文件”菜单或弹出的上下文菜单中选择 “打开”，或者仅双击该图标即可。

d. 在 “服务 (本地)”下，选择 "Message Queue Broker" 服务，然后从 “操作”菜单中选择 “属性”。

   或者，也可以在 "Message Queue Broker" 上单击鼠标右键，然后从弹出的上下文菜单中选择 “属性”，或者仅双击 "Message Queue Broker" 即可。无论在哪种情况下，都会显示 “Message Queue Broker 属性”对话框。

e. 在 “属性”对话框中的 “常规”选项卡下，单击 “停止”以停止代理服务。

2. 删除服务。

   在命令行中输入以下命令：

   ```
   imqsvcadmin remove
   ```

3. 重新安装服务，并使用 -args 选项指定不同的代理启动选项，或使用 -vmargs 选项指定不同的 Java 版本参数。

   例如，要将服务的主机名和端口号分别更改为 broker1 和 7878，可以使用以下命令：

   ```
   imqsvcadmin install -args "-name broker1 -port 7878"
   ```

使用可选 Java 运行时

可以使用 imqsvcadmin 命令的 -javahome 或 -jrehome 选项指定可选 Java 运行时的位置。（还可以在服务 “属性”对话框 “常规”选项卡下的 “启动参数”字段中指定这些选项。）

注 “启动参数”字段将反斜杠字符 (\) 视为转义符，因此在将反斜杠用作路径分隔符时必须键入两次，例如：

```
-javahome c:\\j2sdk1.4.0
```

显示代理服务启动选项

要确定代理服务的启动选项，请使用 imqsvcadmin 命令的 query 选项，如代码示例 3-1 中所示。

代码示例 3-1 显示代理服务启动选项

```
imqsvcadmin query

Service Message Queue Broker is installed.
Display Name:Message Queue Broker
Start Type:Automatic
Binary location:C:\Sun\MessageQueue\bin\imqbrokersvc.exe
JavaHome:c:\j2sdk1.4.0
Broker Args:-name broker1 -port 7878
```

服务启动问题疑难解答

如果试图启动作为 Windows 服务的代理时出现错误，您可以查看记录的错误事件：

➤ **查看记录的服务错误事件**

1. 打开 Windows 的 “管理工具”控制面板。
2. 启动 “事件查看器”工具。
3. 选择 “应用程序事件日志”。
4. 从 “操作”菜单中选择 “刷新”以显示所有错误事件。

删除代理

删除代理的过程也因平台而异，如以下各节所述。

删除 Solaris 或 Linux 上的代理

要删除 Solaris 或 Linux 平台上的代理实例，可以使用具有 `-remove` 选项的 `imqbrokerd` 命令。命令的格式如下：

```
imqbrokerd [options...] -remove instance
```

例如，如果代理名称为 `myBroker`，则此命令应为：

```
imqbrokerd -name myBroker -remove instance
```

此命令将删除指定代理的整个实例目录。

如果将代理设置为在系统启动时自动启动，请编辑配置文件 /etc/imq/imqbrokerd.conf (Solaris) 或 /etc/opt/sun/mq/imqbrokerd.conf (Linux)，将 AUTOSTART 属性设置为 NO。

有关 imqbrokerd 命令的语法、子命令和选项的完整信息，请参见第 242 页上的“代理实用程序”。要了解此信息的简要概述，请输入以下命令：

删除 Windows 代理服务

要删除作为 Windows 服务运行的代理，请使用以下命令：

```
imqcmd shutdown bkr
```

关闭代理，随后使用以下命令：

```
imqsvcadmin remove
```

删除服务。

或者，也可以使用 Windows 的服务工具 （可通过管理工具控制面板访问）来停止和删除代理服务。

删除代理服务后请重新启动计算机。

启动客户端

启动客户端应用程序之前，请向应用程序开发者了解有关如何设置系统的信息。如果要启动 Java 客户端应用程序，则必须正确设置 CLASSPATH 变量，并确保安装了正确的 .jar 文件。 Message Queue Developer's Guide for Java Clients 中包含有关系统设置常规步骤的信息，但开发者可能会提供额外信息。

要启动 Java 客户端应用程序，请使用以下命令行格式：

```
java clientAppName
```

要启动 C 客户端应用程序，请使用应用程序开发者提供的格式。

应用程序文档应提供有关应用程序设置的属性值的信息；您可能希望通过命令行覆盖其中某些属性值。您可能还希望在命令行中为使用 Java 命名和目录接口 (Java Naming and Directory Interface, JNDI) 查找来查找连接工厂的任何 Java 客户端指定属性。如果查找返回的连接工厂比应用程序旧，则该连接工厂可能不支持较新的属性。在这种情况下， Message Queue 将这些属性设置为默认值；如有必要，可以使用命令行覆盖这些默认值。

要从命令行中为 Java 应用程序指定属性值，请使用如下语法：

```
java [[-Dattribute=value]...] clientAppName
```

attribute 的值必须是连接工厂受管理对象属性，如第 16 章 “受管理对象属性参考”。中所述；如果值中包含空格，请在命令行 *attribute=value* 部分的两端加上引号。

以下示例将启动一个名为 `MyMQClient` 的客户端应用程序，该示例连接到主机 `OtherHost` 上位于端口 `7677` 的代理：

```
java -DimqAddressList=mq://OherHost:7677/jms MyMQClient
```

在命令行上指定的主机名和端口会覆盖应用程序自身设置的任何其他主机名和端口。

在某些情况下无法使用命令行来指定属性值。管理员可以将受管理对象设置为只允许读取访问，应用程序开发者也可以通过对客户端应用程序进行编码来实现此目的。与应用程序开发者进行沟通非常必要，这有助于了解启动客户端程序的最佳途径。

第 4 章

配置代理

代理配置由一组配置文件以及在启动时传递给 imqbrokerd 命令的选项控制。本章描述了可用的配置属性以及如何使用它们来配置代理。

本章包含以下各节：

有关代理配置属性的完整参考信息，请参见第 14 章 “代理属性参考”。

代理服务

代理配置属性可以分为几种类别，具体取决于它们所影响的服务或代理组件：

- **连接服务**管理代理与其客户端之间的物理连接，这些连接用于传输传入和传出的消息。
- **路由服务**路由和传送 JMS 有效负荷消息，以及消息服务用于支持可靠传送的控制消息。
- **持久性服务**管理持久性存储器中的数据写入和检索。
- **安全服务**对连接到代理的用户进行验证，并授予他们执行相应操作的权限。
- **监视服务**生成有关代理性能的度量和诊断信息。

以下各节介绍了上述每个服务，并介绍了可用于根据特定需要自定义这些服务的属性。

连接服务

消息代理可以提供各种**连接服务**，这些连接服务使用各种传输协议来支持应用程序客户端和管理客户端。第 259 页上的 “连接属性”中列出了与连接服务相关的代理配置属性。

表 4-1 显示了可用的连接服务，这些服务可以通过以下两个特性进行区分：

- **服务类型**指定服务提供的是 JMS 消息传送 (NORMAL) 还是 Message Queue 管理服务 (ADMIN)。
- **协议类型**指定底层传输协议。

表 4-1 Message Queue 连接服务

服务名称	服务类型	协议类型
jms	NORMAL	TCP
ssljms (Enterprise Edition)	NORMAL	TLS （基于 SSL 的安全性）
httpjms (Enterprise Edition)	NORMAL	HTTP
httpsjms (Enterprise Edition)	NORMAL	HTTPS （基于 SSL 的安全性）
admin	ADMIN	TCP
ssladmin	ADMIN	TLS （基于 SSL 的安全性）

通过设置代理的 imq.service.activelist 属性，可以将其配置为运行上述任意或全部连接服务。此属性的值是一个连接服务列表，当代理启动时，会激活该列表中的连接服务；如果未明确指定此属性，则默认情况下将激活 jms 和 admin 服务。

每个连接服务还支持特定的验证和授权功能；有关更多信息，请参见第 78 页上的 “安全服务”。

端口映射器

每个连接服务仅在由主机名 （或 IP 地址）和端口号指定的特定端口上可用。您可以明确地为服务指定静态端口号，也可以让代理的**端口映射器**动态指定端口号。端口映射器自身驻留在代理的**主端口**上，该端口通常位于标准端口号 7676 上。（如果需要，您可以通过代理配置属性 imq.portmapper.port，用其他端口号来覆盖标准端口号。）默认情况下，每个连接服务在启动时都在端口映射器中注册自身。当客户端创建与代理的连接时， Message Queue 客户端运行时首先与端口映射器联系，为所需的连接服务请求端口号。

或者，也可以使用 imq.*serviceName*.*protocolType*.port 配置属性 （其中 *serviceName* 和 *protocolType* 标识特定的连接服务，如表 4-1 中所示）来覆盖端口映射器的设置，并明确地为连接服务指定一个静态端口号。（只有 jms、 ssljms、 admin 和 ssladmin 连接服务可以通过这种方式进行配置； httpjms 和 httpsjms 服务使用不同的配置属性，如附录 C “HTTP/HTTPS 支持”中所述。）但是，静态端口通常仅在特殊情况下 （例如要穿过防火墙建立连接）使用，建议不要在一般情况下使用静态端口。

注 如果有两个或更多的主机可用 （例如，在一台计算机中安装了多个网卡），则可以使用代理属性来指定连接服务应该绑定到哪个主机。imq.hostname 属性为所有连接服务指定一个默认主机；如果需要，以后可以使用 imq.*serviceName*.*protocolType*.hostname （对于 jms、ssljms、 admin 或 ssladmin 服务）或 imq.portmapper.hostname （对于端口映射器自身）来覆盖此默认值。

如果同时收到多个端口管理器请求，则它们将存储在操作系统待办事项中等待操作。 imq.portmapper.backlog 属性指定待办事项中的请求的最大数量。如果超过此限制，随后的任何请求都将被拒绝，直到待办事项中的请求减少。

线程池管理

每个连接服务都是多线程的，因此可以支持多个连接。这些连接所需的线程由代理在每个服务的单独**线程池**中维护。当连接需要某些线程时，这些线程即添加到支持该连接的服务的线程池中。

您选择的线程模型指定了线程是专用于单个连接还是由多个连接共享：

- 在**专用模型**中，与代理的每个连接都需要两个线程：一个用于传入消息，另一个用于传出消息。这限制了可以支持的连接数，但却提高了性能。
- 在**共享模型**中，当发送或接收消息时，连接由共享线程处理。由于每个连接都不需要专用线程，因此这种模型增加了可能的连接数，但却降低了性能，因为线程管理需要额外的开销。

代理的 imq.*serviceName*.threadpool_model 属性指定了为给定连接服务使用两个模型中的哪一个。此属性接受两个字符串值中的一个：dedicated 或 shared。如果未明确设置此属性，则默认情况下采用 dedicated。

还可以通过设置代理属性 imq.*serviceName*.min_threads 和 imq.*serviceName*.max_threads 来指定服务线程池中的最小线程数和最大线程数。当可用线程数超过指定的最小阈值时， Message Queue 将在线程变为空闲状态时将其关闭，直到再次达到最小阈值，以此来节省内存资源。如果负载较重，线程数可能会增加，直到达到线程池的最大数量；此后，新的连接将被拒绝，直到某个线程变得可用。

共享线程模型使用**分配器线程**为活动连接指定线程。代理属性 imq.shared.connectionMonitor_limit 指定单个分配器线程可以监视的最大连接数。此属性的值越小，为连接指定线程的速度越快。 imq.ping.interval 属性指定代理定期测试 ("ping") 连接 (以验证该连接是否仍然处于活动状态) 的时间间隔（以秒为单位）。通过这种定期测试，可以尽早检测出连接故障，以免在尝试传输消息时失败。

路由服务

客户端连接到代理之后，即可开始路由和传送消息。在这个阶段，代理负责创建和管理不同类型的物理目标，确保消息流畅通，以及有效地使用资源。您可以使用第 261 页上的 “路由属性”中介绍的代理配置属性，根据应用程序的需要以相应方式来管理这些任务。

代理的性能和稳定性取决于可用的系统资源 （如内存）以及资源的使用效率。可以设置配置属性，以防止代理因传入消息太多而过载，或者耗尽内存。这些属性在三种不同的级别上起作用，使消息服务在资源不足时仍可正常运行：

- **系统范围的消息限制**共同应用于系统中的所有物理目标。这些限制包括代理保存的最大消息数 (imq.system.max_count) 以及这些消息占用的最大总字节数 (imq.system.max_size)。如果达到这两个限制中的任何一个，代理将会拒绝所有新消息，直到待处理消息低于该限制。在单个消息的最大大小 (imq.message.max_size) 以及回收过期消息的时间间隔 (imq.message.expiration.interval) 方面也存在限制。
- **单个目标限制**控制发送到特定物理目标的消息流。在第 15 章 “物理目标属性参考”中对控制这些限制的配置属性进行了介绍。其中包括以下几个方面的限制：目标可以保存的消息的数量和大小、可以为目标创建的消息生成方和使用方的数量，以及可以作为一个批次一起传送到目标的消息数。

  可以对目标进行配置以响应内存限制：降低消息生成方传送消息的速度、拒绝新的传入消息，或者丢弃时间最长或优先级最低的现有消息。对于以这种方式从目标中删除的消息，可以选择将其移动到停用消息队列，而不是彻底丢弃；代理属性 imq.destination.DMQ.truncateBody 可以控制是将整个消息主体保存在停用消息队列中，还是仅仅将消息头和属性数据保存在停用消息队列中。

为了在应用程序开发和测试期间提供方便，可以将消息代理配置为只要消息生成方或使用方尝试访问的目标不存在，即自动创建新的物理目标。第 262 页上的表 14-3 中汇总的代理属性与上面介绍的代理属性类似，但它们适用于这种**自动创建的目标**，而不是以管理方式创建的目标。

- **系统内存阈值**定义了几个内存使用级别，代理将根据这些级别采取越来越严格的措施以防止内存过载。共定义了四个这样的使用级别：
  - **Green**：有大量内存可用。
  - **Yellow**：代理内存开始变得不足。
  - **Orange**：代理内存非常少。
  - **Red**：代理已没有可用内存。

  定义这些级别的内存占用百分比分别由代理属性 `imq.green.threshold`、`imq.yellow.threshold`、 `imq.orange.threshold` 和 `imq.red.threshold` 指定；默认值为 0% (green)、 80% (yellow)、 90% (orange) 和 98% (red)。

  当内存使用从一个级别上升到另一个级别时，代理将采取渐进的响应措施：首先将消息从活动内存交换到持久性存储器中，然后限制非持久性消息的生成方，最后阻止消息流入代理。（这两种措施都会降低代理的性能。）代理使用以下方法来限制消息生成：将传送的每个批次的大小限制为由属性 `imq.`*resourceState*`.count` 指定的消息数，其中 *resourceState* 分别为 `green`、`yellow`、 `orange` 或 `red`。

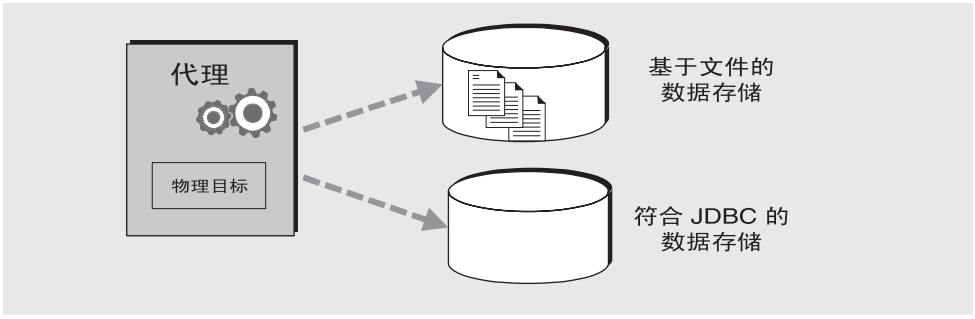
注 触发这些系统内存阈值表明系统范围和目标级别的消息限制设置得过高。由于内存阈值无法始终及时地捕获潜在的内存过载，因此不应该依赖它们来控制内存使用，而应重新配置系统范围和目标级别的限制以优化内存资源。

持久性服务

代理在发生故障后进行恢复时，需要重新创建消息传送操作的状态。要执行此操作，代理必须将状态信息保存到**持久性数据存储**中。代理在重新启动后，将使用保存的数据重新创建目标和长期订阅，恢复持久性消息，回滚打开的事务，并为未传送的消息重新生成路由表。然后它才能恢复消息传送。

Message Queue 既支持基于文件的持久性模块，也支持基于 JDBC 的持久性模块（见图 4-1）。基于文件的持久性使用单个文件来存储持久性数据；基于 JDBC 的持久性使用 Java 数据库连接 (Java Database Connectivity, JDBCTM) 接口将代理连接到符合 JDBC 的数据存储。虽然基于文件的持久性通常比基于 JDBC 的持久性快，但某些用户更希望使用 JDBC 存储所提供的冗余和管理控制功能。代理配置属性 `imq.persist.store`（请参见第 266 页上的表 14-4）指定使用两种持久性形式中的哪一种。

图 4-1 持久性数据存储



基于文件的持久性

默认情况下， Message Queue 使用基于文件的持久性数据存储，在这种存储中使用单个文件来存储持久性数据，如消息、目标、长期订阅和事务。第 266 页上的 “基于文件的持久性”中列出了与基于文件的持久性相关的代理配置属性。

基于文件的存储位于某个目录中，该目录使用数据存储所属的代理实例的名称 (*instanceName*) 进行标识：

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

（有关 `instances` 目录的位置，请参见附录 A “Message Queue 数据在特定平台上的位置”。）代理中的每个目标都有其自身的子目录，用于保存传送到该目标的消息。

注 由于持久性数据存储可能会包含敏感或专用的消息，因此应该保护 .../instances/*instanceName*/fs350/ 目录不受未经授权的访问；请参见第 88 页上的 “保护持久性数据”。

消息以外的所有持久性数据都存储在单独的文件中：一个文件用于目标，另一个文件用于长期订阅，第三个文件用于事务状态信息。大多数消息都存储在由大小可变的记录组成的单个文件中。可以压缩此文件，以减少添加和删除消息时产生的碎片（请参见第 116 页上的 “压缩物理目标”）。此外，超过特定阈值大小的消息将存储在其各自的文件中，而不是存储在大小可变的记录文件中。可以使用代理属性 `imq.persist.file.message.max_record_size` 来配置此阈值的大小。

代理为这些单个的消息文件维护一个文件池：当不再需要某个文件时，并不会将其删除，而是重新放入目标目录的空闲文件池中，以便日后其他消息可以重新使用该文件。代理属性 `imq.persist.file.destination.message.filepool.limit` 指定池中的最大文件数。当某个目标的单个消息文件数超过此限制时，将删除不再需要的文件，而不是将其重新放入文件池。

如果将文件重新放入文件池，则代理可以节省时间，但会占用存储器空间，因为它只是将文件标记为可重新使用，而没有删除文件以前的内容。可以使用 `imq.persist.file.message.filepool.cleanratio` 代理属性来指定每个目标的文件池中应保持 “清除”（空）状态 （而不是仅仅标记为可重新使用）的文件数百分比。此值设置得越高，文件池所需的空间越少，但清空文件内容 （当文件重新放入文件池时）所需的开销也越大。如果代理的 `imq.persist.file.message.cleanup` 属性为 `true`，则在代理关闭时将清空池中的所有文件，使其保持清除状态；这样可以节省存储器空间，但会减慢关闭过程。

将数据写入持久性存储时，操作系统在以同步还是 “延迟”（异步）方式写入数据方面存在一定的不准确性。延迟存储可能会导致数据在系统崩溃时丢失 （如果代理认为数据已经写入持久性存储器，但实际并未写入）。为了确保绝对的可靠性 （以牺牲性能为代价），可以将代理属性 `imq.persist.file.sync.enabled` 设置为 `true`，以要求同步写入所有数据。在这种情况下，当系统在崩溃后恢复运行时，可以保证数据是可用的，并且代理可以可靠地恢复运行。但请注意，虽然数据并未丢失，但却不能用于群集中的任何其他代理，因为群集中的代理当前并未共享该数据。

基于 JDBC 的持久性

可以不使用基于文件的持久性，而将代理设置为访问可通过 JDBC 驱动程序进行访问的任何数据存储。这需要设置适当的与 JDBC 相关的代理配置属性，以及使用数据库管理器实用程序 (`imqdbmgr`) 创建具有正确结构的数据库。有关详细信息，请参见第 87 页上的 “配置基于 JDBC 的存储”。

第 267 页上的 “基于 JDBC 的持久性”中列出了用于将代理配置为使用 JDBC 数据库的属性。可以在每个代理实例的实例配置文件 (`config.properties`) 中指定这些属性，也可以使用代理实用程序 (`imqbrokerd`) 或数据库管理器实用程序 (`imqdbmgr`) 的 `-D` 命令行选项来指定这些属性。

imq.persist.jdbc.driver 属性提供了连接到数据库时使用的 JDBC 驱动程序的 Java 类名。还有一些属性用于指定具有以下功能的 URL：连接到现有数据库 (imq.persist.jdbc.opendburl)、创建新的数据库 (imq.persist.jdbc.createdburl) 和关闭数据库连接 (imq.persist.jdbc.closedburl)。

imq.persist.jdbc.user 和 imq.persist.jdbc.password 属性提供了用于访问数据库的用户名和密码； imq.persist.jdbc.needpassword 是用于指定是否需要密码的布尔标志。为了安全起见，应该仅在通过 -passfile 命令行选项指定的密码文件中指定密码；如果未指定此类密码文件，则 imqbrokerd 和 imqdbmgr 命令将以交互方式提示您指定密码。同样，可以通过在命令行中使用 imqbrokerd 命令的 -dbuser 选项或 imqdbmgr 命令的 -u 选项来提供用户名。

在由多个代理实例共享的 JDBC 数据库中，配置属性 imq.persist.jdbc.brokerid 指定每个实例的唯一实例标识符，该标识符将被附加到数据库表名的后面。（嵌入式数据库只存储一个代理实例的数据，因此通常不需要此属性。）与 JDBC 相关的其余配置属性用于自定义创建数据库结构的 SQL 代码，每个数据库表对应一个属性。例如， imq.persist.jdbc.table.IMQSV35 属性提供用于创建版本表的 SQL 命令， imq.persist.jdbc.table.IMQCCREC35 属性提供用于创建配置更改记录表的 SQL 命令， imq.persist.jdbc.table.IMQDEST35 属性提供用于创建目标表的 SQL 命令，等等；有关完整列表，请参见第 268 页上的表 14-6。

注 由于各个数据库系统所要求的具体 SQL 语法不同，因此请务必查看数据库供应商提供的相应文档以了解详细信息。

安全服务

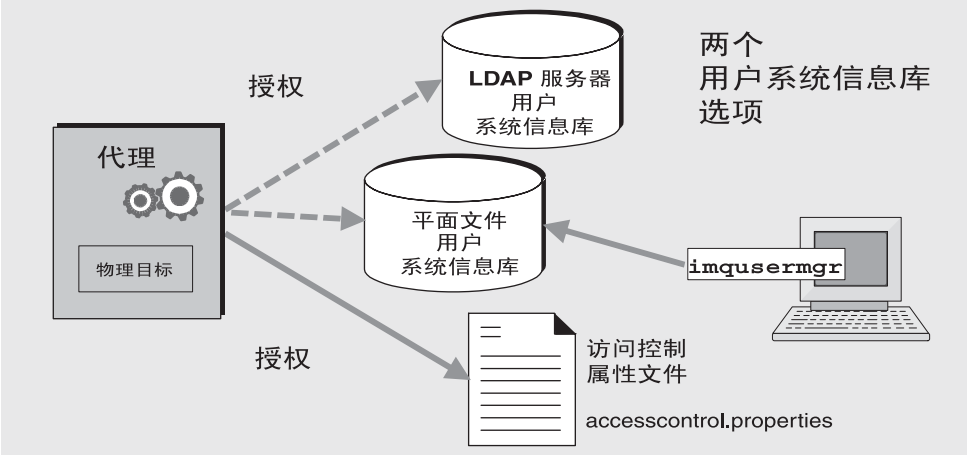
Message Queue 为用户访问控制 （验证和授权）和加密提供了安全服务。

- **验证**确保只有通过验证的用户才能与代理建立连接。
- **授权**指定哪些用户或组具有访问资源以及执行特定操作的权限。
- **加密**防止消息在通过连接传送时被篡改。

Message Queue 管理员负责设置代理在验证用户以及为用户授予操作权限时所需的信息。第 270 页上的 “安全属性”中列出了与安全服务有关的代理属性。布尔属性 `imq.accesscontrol.enabled` 充当一个主开关，它控制是否在代理范围内应用访问控制；要进行更精细的控制，可以通过设置 `imq.`*serviceName*`.accesscontrol.enabled` 属性（其中 *serviceName* 是连接服务的名称，如第 72 页上的表 4-1 所示）来覆盖特定连接服务的此设置：例如，`imq.httpjms.accesscontrol.enabled`。

图 4-2 显示了代理在提供验证和授权服务时所需的组件。这些服务依赖于**用户系统信息库**，该系统信息库包含有关消息传送系统用户的以下信息：用户名、密码和组成员资格。此外，为了给用户或组授予执行特定操作的权限，代理还会查询**访问控制属性文件**，该文件指定了用户或组可以执行的操作。可以使用配置属性 `imq.accesscontrol.file.filename` 为整个代理指定一个访问控制属性文件，也可以使用 `imq.`*serviceName*`.accesscontrol.file.filename` 为单个连接服务指定访问控制属性文件。

图 4-2 安全支持



如图 4-2 所示，您可以将用户数据存储在随 Message Queue 服务一起提供的平面文件用户系统信息库中，也可以将它们插入原有的轻量目录访问协议 (Lightweight Directory Access Protocol, LDAP) 系统信息库中：

- 如果选择平面文件系统信息库，则必须使用 Message Queue 用户管理器实用程序 (`imqusermgr`) 来管理系统信息库。此选项是内置的，非常易于使用。

- 如果您希望使用现有的 LDAP 服务器，请使用 LDAP 供应商提供的工具来填充和管理用户系统信息库。您还必须在代理的实例配置文件中设置一些属性，以使代理能够在 LDAP 服务器中查询有关用户和组的信息。

代理的 imq.authentication.basic.user_repository 属性指定要使用的系统信息库的类型。通常，如果可伸缩性很重要，或者您需要让不同的代理共享系统信息库（例如，如果您使用的是代理群集），则最好使用 LDAP 系统信息库。有关设置平面文件或 LDAP 用户系统信息库的更多信息，请参见第 121 页上的 “验证用户”。

验证

请求与代理建立连接的客户端必须提供用户名和密码，代理会将该用户名和密码与存储在用户系统信息库中的用户名和密码进行比较。从客户端传输到代理的密码是使用 Base-64 编码 （对于平面文件系统信息库）或消息摘要 (MD5) 散列 （对于 LDAP 系统信息库）进行编码的。具体选择哪一种编码由 imq.authentication.type 属性 （对于整个代理）或 imq.*serviceName*.authentication.type 属性 （对于特定连接服务）控制。 imq.authentication.client.response.timeout 属性设置验证请求的超时时间间隔。

如第 144 页上的 “使用密码文件”中所述，您可以选择将密码放在**密码文件**中，而不是让系统以交互方式提示您指定密码。此选项由布尔型代理属性 imq.passfile.enabled 控制。如果此属性为 true，则 imq.passfile.dirpath 和 imq.passfile.name 属性提供密码文件的目录路径和文件名。 imq.imqcmd.password 属性 （可以嵌入密码文件中）指定密码，该密码用于验证管理用户是否有权使用命令实用程序 (imqcmd) 来管理代理、连接服务、连接、物理目标、长期订阅和事务。

如果您使用的是基于 LDAP 的用户系统信息库，则可以使用所有代理属性来配置 LDAP 查找的各个方面。 LDAP 服务器自身的地址 （主机名和端口号）由 imq.user_repository.ldap.server 指定。 imq.user_repository.ldap.principal 属性提供用于绑定到 LDAP 系统信息库的标识名，而 imq.user_repository.ldap.password 则提供关联的密码。其他属性为单个的用户和组搜索指定目录库和可选 JNDI 过滤器，为用户和组名指定特定于提供者的属性标识符，等等；有关详细信息，请参见第 270 页上的 “安全属性”。

授权

用户通过验证后，即可被授权执行与 Message Queue 相关的各种活动。 Message Queue 管理员可以定义用户组，并指定组中各个用户的成员资格。默认的访问控制属性文件只明确地引用一个组，即 admin （请参见第 124 页上的 “组”）。此组中的用户具有 admin 连接服务的连接权限，该权限允许用户执行诸如创建目标以及监视和控制代理等管理功能。默认情况下，您定义的任何其他组中的用户都无法获取 admin 服务连接。

当用户尝试执行某个操作时，代理将对照访问控制属性文件中指定的允许执行该操作的用户名和组成员资格，来检查用户系统信息库中该用户的用户名和组成员资格。访问控制属性文件为用户或组指定了执行以下操作的权限：

- 连接到代理
- 访问目标：为任意给定目标或所有目标创建使用方、生成方或队列浏览器
- 自动创建目标

加密

要对客户端与代理之间发送的消息进行加密，需要使用基于安全套接字层 (Secure Socket Layer, SSL) 标准的连接服务。SSL 通过在启用 SSL 的代理与客户端之间建立加密连接来提供连接级别的安全性。

要使用基于 SSL 的 Message Queue 连接服务，需要使用密钥工具实用程序 (`imqkeytool`) 生成专用密钥 / 公共密钥对。此实用程序将公共密钥嵌入自签名的证书中，然后将此证书放入 Message Queue 密钥库中。密钥库自身受密码保护；要解除对密钥库的锁定，必须在启动时提供由 `imq.keystore.password` 属性指定的密钥库密码。解除对密钥库的锁定后，代理即可将证书传递给请求连接的任何客户端。然后，客户端可以使用此证书建立与代理的加密连接。

`imq.audit.enabled` 代理属性控制将审计记录记录到 Message Queue 代理日志文件的行为；有关更多信息，请参见第 145 页上的 “创建审计日志”。

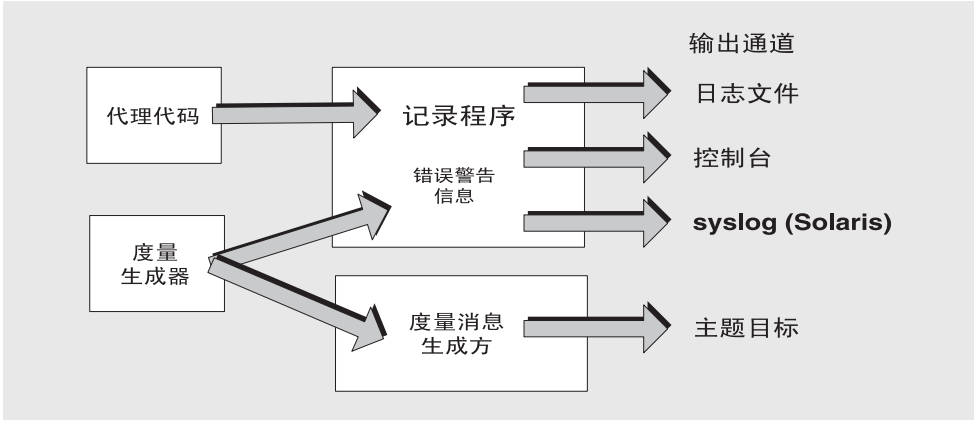
监视服务

代理中包含一些用于监视和诊断应用程序及代理性能的组件。其中包括：

- 生成数据的组件，包括度量生成器和记录事件的代理代码
- 将输出信息写入多个输出通道的记录程序组件
- 度量消息生成方，将包含度量信息的 JMS 消息发送到主题目标以供 JMS 监视客户端使用

图 4-3 中显示的是常规模式。第 274 页上的 “监视属性”中列出了用于配置监视服务的代理属性。

图 4-3 监视支持



度量生成器

度量生成器提供有关代理活动的信息，如流入流出代理的消息、代理内存中的消息数及其使用的内存量、打开的连接数以及正在使用的线程数。布尔型代理属性 imq.metrics.enabled 控制是否记录此类信息； imq.metrics.interval 指定记录的频率。

记录程序

出错时，记录程序获取代理代码和度量生成器生成的信息，并将这些信息写入标准输出 （控制台）、日志文件以及 syslog 守护进程 （在 Solaris 平台上）中。要使用的日志文件由 imq.log.file.dirpath 和 imq.log.file.filename 代理属性标识；imq.log.console.stream 指定将控制台输出定向到 stdout 还是 stderr。

imq.log.level 属性控制记录程序收集的度量信息的类别：ERROR、 WARNING 或 INFO。每个级别都包括高于它的级别，因此，如果您指定 WARNING 作为日志记录级别，则将同时记录错误消息。 imq.log.console.output 和 imq.log.file.output 属性分别控制将哪些指定类别写入控制台和日志文件。但在这种情况下，类别**并不**包括高于它的类别；因此，如果您要将错误和警告写入日志文件，而将信息性消息写入控制台，则必须明确地将 imq.log.file.output 设置为 ERROR|WARNING，将 imq.log.console.output 设置为 INFO。在 Solaris 平台上，使用另一个属性 imq.log.syslog.output 来指定要写入 syslog 守护进程的度量信息的类别。此外还有一个 imq.destination.logDeadMsgs 属性，该属性指定当停用消息被丢弃或移动到停用消息队列时是否进行记录。

对于日志文件，可以指定何时关闭文件并将输出转移到新文件。当日志文件达到指定的大小 (imq.log.file.rolloverbytes) 或生存期 (imq.log.file.rolloversecs) 之后，将保存该文件并创建一个新的日志文件。

有关与日志记录相关的其他代理属性，请参见第 274 页上的 “监视属性”；有关如何配置记录程序以及如何使用它来获取性能信息的更多详细信息，请参见第 175 页上的 “配置和使用代理日志记录”。

度量消息生成方 (Enterprise Edition)

度量消息生成方以一定的时间间隔从度量生成器接收信息，并将该信息写入**度量消息**，然后根据消息中包含的度量信息类型，将度量消息发送到多个度量主题目标之一（见表 4-2）。订阅这些度量主题目标的 Message Queue 客户端可以使用这些消息并处理消息中包含的度量数据。这样，开发者就可以创建自定义的监视工具来支持消息传送应用程序。有关在每种类型的度量消息中报告的度量数量的详细信息，请参见 Message Queue Developer's Guide for Java Clients。

表 4-2 度量主题目标

主题名称	度量信息类型
mq.metrics.broker	代理度量
mq.metrics.jvm	Java 虚拟机度量
mq.metrics.destination_list	目标及其类型的列表
mq.metrics.destination.queue.*queueName*	指定队列的目标度量
mq.metrics.destination.topic.*topicName*	指定主题的目标度量

代理属性 imq.metrics.topic.enabled 和 imq.metrics.topic.interval 分别控制是否将消息发送到度量主题目标以及发送的频率。imq.metrics.topic.timetolive 和 imq.metrics.topic.persist 属性分别指定此类消息的生命周期和持久性。

除了包含在度量消息主体中的信息之外，每个消息头中还包含提供以下附加信息的属性：

- 消息类型
- 发送消息的代理的地址（主机名和端口号）
- 度量样例的取样时间

以下属性对于处理不同类型（或来自不同代理）的度量消息的客户端应用程序非常有用。

设置代理属性

可以通过以下两种方式之一来指定代理的配置属性：

- 编辑代理的配置文件
- 直接从命令行提供属性值

以下两节介绍了这两种配置代理的方法。

配置文件

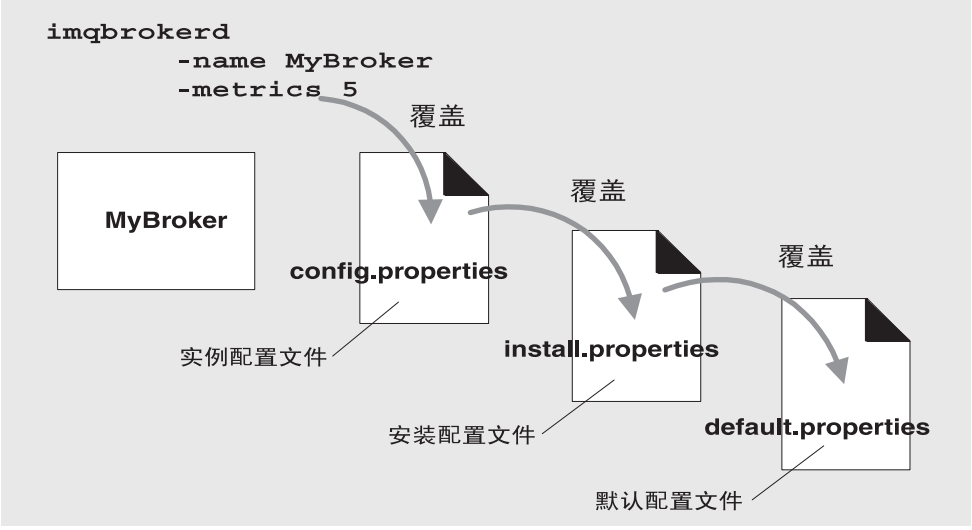
代理配置文件包含用于配置代理的属性设置。它们保存在一个目录中，该目录的位置取决于使用的操作系统平台；有关详细信息，请参见附录 A “Message Queue 数据在特定平台上的位置”。该目录存储以下文件：

- 启动时加载的**默认配置文件** `default.properties`。此文件不可编辑，但您可以通过读取该文件来确定默认设置，以及查找要更改的属性的确切名称。
- 包含安装 Message Queue 时指定的全部属性的**安装配置文件** `install.properties`。此文件在安装后无法进行编辑。

此外，每个单独的代理实例都有其自身的**实例配置文件**，如下所述。如果连接群集中的代理实例，您可能还需要使用**群集配置文件**来指定群集的配置信息；有关更多信息，请参见第 277 页上的 “群集配置属性”。

启动时，代理会合并各个配置文件中的属性值。如图 4-4 中所示，这些文件构成了一个分层结构，在此结构中，实例配置文件中指定的值将覆盖安装配置文件中的值，而安装配置文件中的值又将覆盖默认配置文件中的值。在分层结构的顶部，您可以通过使用 `imqbrokerd` 命令的命令行选项来手动覆盖配置文件中指定的任何属性值。

图 4-4 代理配置文件



编辑实例配置文件

首次运行代理时，将创建一个实例配置文件，其中包含该特定代理实例的配置属性。该实例配置文件被命名为 config.properties，并存储在由所属代理实例的名称标识的目录中：

…/instances/*instanceName*/props/config.properties

（有关 instances 目录的位置，请参见附录 A “Message Queue 数据在特定平台上的位置”。）如果该文件不存在，则必须在启动代理时使用 -name 选项 （请参见第 242 页上的 “代理实用程序”）指定一个实例名， Message Queue 可以使用该实例名来创建文件。

注 instances/*instanceName* 目录和实例配置文件由创建相应代理实例的用户所有。代理实例必须始终由该用户重新启动。

实例配置文件由代理实例维护，并在您使用 Message Queue 管理实用程序更改配置时进行相应的更改。您也可以手动编辑实例配置文件，以便自定义代理的行为和资源使用。要执行此操作，您必须是 instances/*instanceName* 目录的拥有者，或者以 root 身份登录以更改目录的访问权限。

代理仅在启动时读取实例配置文件。要对代理配置进行永久性更改，您必须关闭代理，编辑该文件，然后重新启动代理。该文件 （或任何配置文件）中的属性定义使用以下语法：

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

例如，下面的条目指定，代理最多可以在内存和持久性存储器中保存 50,000 条消息，超过此限制后，将拒绝其他消息：

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

下面的条目指定，将每天 （86,400 秒）创建一个新的日志文件：

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

有关可用代理配置属性及其默认值的信息，请参见第 71 页上的 “代理服务”和第 14 章 “代理属性参考”。

从命令行设置配置选项

在启动代理时 （或之后），可以在命令行中输入代理配置选项。

在启动时，可使用代理实用程序 (`imqbrokerd`) 启动代理实例。通过使用该命令的 `-D` 选项，可以指定任何代理配置属性及其值；有关更多信息，请参见第 64 页上的 “启动代理”和第 242 页上的 “代理实用程序”。如果使用服务管理器实用程序 (`imqsvcadmin`) 将代理作为 Windows 服务启动，则可使用 `-args` 选项指定启动配置属性；请参见第 257 页上的 “服务管理器实用程序”。

在代理实例运行时，还可以更改某些代理属性。要修改正在运行的代理的配置，请使用命令实用程序的 `imqcmd update bkr` 命令；请参见第 95 页上的 “更新代理属性”和第 247 页上的 “代理管理”。

配置持久性数据存储

代理的持久性数据存储保存有关物理目标、长期订阅、消息、事务和确认的信息。默认情况下， Message Queue 代理被配置为使用基于文件的持久性存储，但您可以将其重新配置为插入可通过 JDBC 驱动程序访问的任何数据存储。代理配置属性 `imq.persist.store` （请参见第 266 页上的表 14-4）指定使用两种持久性形式中的哪一种。

本节介绍了如何设置代理以使用持久性存储。包括以下主题：

- 第 87 页上的 “配置基于文件的存储”

- 第 87 页上的 “配置基于 JDBC 的存储”
- 第 88 页上的 “保护持久性数据”

配置基于文件的存储

创建代理实例时，将自动创建基于文件的数据存储。存储位于代理的实例目录中；有关确切位置，请参见附录 A “Message Queue 数据在特定平台上的位置”。

默认情况下， Message Queue 执行异步的磁盘写入操作。操作系统可以缓冲这些操作以获取高性能。但是，如果在两次写入操作之间出现意外的系统故障，则消息可能会丢失。为了提高可靠性 （但会降低性能），可以设置代理属性 `imq.persist.file.sync` 以改为同步写入数据。有关此属性的进一步讨论，请参见第 76 页上的 “基于文件的持久性”和第 266 页上的表 14-5。

启动代理实例时，可以使用 `imqbrokerd` 命令的 `-reset` 选项清除文件系统存储。有关此选项及其子选项的更多信息，请参见第 242 页上的 “代理实用程序”。

配置基于 JDBC 的存储

要配置代理以使用基于 JDBC 的持久性，请在代理的实例配置文件中设置与 JDBC 相关的属性，并创建相应的数据库结构。 Message Queue 数据库管理器实用程序 (`imqdbmgr`) 使用 JDBC 驱动程序和代理配置属性来创建和管理数据库。如果数据库表已损坏或者您希望使用其他数据库作为数据存储，还可以使用数据库管理器从数据库中删除损坏的表或删除数据库。有关更多信息，请参见第 254 页上的 “数据库管理器实用程序”。

注 系统提供了 Oracle 和 PointBase 数据库产品的示例配置。这些文件的位置因平台而异，在附录 A “Message Queue 数据在特定平台上的位置”中相关表的 “示例应用程序和配置”下列出了此位置。此外，在实例配置文件 `config.properties` 中以注释值的形式提供了 PointBase 嵌入式版本、 PointBase 服务器版本和 Oracle 的示例。

➤ 配置基于 JDBC 的数据存储

1. 在代理的配置文件中设置与 JDBC 相关的属性。

   在第 77 页上的 “基于 JDBC 的持久性”中讨论了这些相关属性，并在第 268 页上的表 14-6 中列出了这些属性。需要特别指出的是，您必须将代理的 `imq.persist.store` 属性设置为 `jdbc` （请参见第 266 页上的表 14-4）。

2. 将 JDBC 驱动程序的 .jar 文件的副本或符号链接放入以下位置：

   /usr/share/lib/imq/ext/ (Solaris)
   /opt/sun/mq/share/lib/ (Linux)
   IMQ_VARHOME\lib\ext (Windows)

   例如，如果您在 Solaris 系统上使用 PointBase，则以下命令将驱动程序的 .jar 文件复制到相应位置：

   ```
   % cp j2eeSDKInstallDirectory/pointbase/lib/pointbase.jar
   /usr/share/lib/imq/ext
   ```

   以下命令则创建一个符号链接：

   ```
   % ln -s j2eeSDKInstallDirectory/lib/pointbase/pointbase.jar
   /usr/share/lib/imq/ext
   ```

3. 创建 Message Queue 持久性所需的数据库结构。

   使用 imqdbmgr create all 命令 （对于嵌入式数据库）或 imqdbmgr create tbl 命令 （对于外部数据库）；请参见第 254 页上的 “数据库管理器实用程序”。

   a. 转到 imqdbmgr 所在的目录：

      cd /usr/bin (Solaris)
      cd /opt/sun/mq/bin (Linux)
      cd IMQ_HOME\bin (Windows)

   b. 输入 imqdbmgr 命令：

      ```
      imqdbmgr create all
      ```

注 如果使用嵌入式数据库，则最好在以下目录中创建它：

…/instances/*instanceName*/dbstore/*databaseName*

如果嵌入式数据库未设置用户名和密码保护，则可能设置了文件系统权限保护。要确保代理能够对数据库进行读写访问，则运行该代理的用户应该是使用 imqdbmgr 命令创建该嵌入式数据库的用户。

保护持久性数据

持久性存储可以包含临时存储的消息文件以及其他信息。由于这些消息可能包含专用信息，因此保护数据存储以防止未经授权的访问非常重要。本节介绍了如何保护基于文件或基于 JDBC 的数据存储中的数据。

保护基于文件的存储

使用基于文件持久性的代理将持久性数据写入平面文件数据存储，该数据存储的位置因平台而异 （请参见附录 A “Message Queue 数据在特定平台上的位置”）：

…/instances/*instanceName*/fs350/

其中 *instanceName* 是标识代理实例的名称。

instanceName/fs350/ 目录是在第一次启动代理实例时创建的。保护此目录的过程取决于运行代理的操作系统平台：

- 在 Solaris 和 Linux 上，目录的权限由启动代理实例的用户的文件模式创建掩码 (umask) 确定。因此，可以通过适当地设置掩码来限制启动代理实例及读取其持久性文件的权限。或者，管理员 （超级用户）也可以通过将 instances 目录的权限设为 700 来保护持久性数据。
- 在 Windows 上，可以使用 Windows 操作系统提供的机制来设置目录的权限。这通常涉及打开目录的 “属性”对话框。

保护基于 JDBC 的存储

使用基于 JDBC 持久性的代理将持久性数据写入符合 JDBC 的数据库。对于由数据库服务器管理的数据库 （如 Oracle），建议创建一个用户名和密码来访问 Message Queue 数据库表 （名称以 IMQ 开头的表）。如果数据库不允许保护单个表，请创建一个仅由 Message Queue 代理使用的专用数据库。请参见数据库供应商提供的文档，以了解有关如何创建用户名 / 密码访问的信息。

代理打开数据库连接所需的用户名和密码可以作为代理配置属性来提供。但是，一种更安全的方法是在启动代理时将它们作为命令行选项 （imqbrokerd 命令的 -dbuser 和 -dbpassword 选项）提供 （请参见第 242 页上的 “代理实用程序”）。

对于代理通过数据库 JDBC 驱动程序直接访问的嵌入式数据库，通常通过在要存储持久性数据的目录上设置文件权限来提供安全性，如前面的 “保护基于文件的存储”。中所述。但是，要确保代理和数据库管理器实用程序均可读写数据库，二者应该由同一个用户来运行。

第 5 章

管理代理

本章介绍如何使用 imqcmd 实用程序来管理代理及其服务。本章包含以下各节：

本章并未涵盖与管理代理相关的全部主题。其他主题将在后面的各章中进行介绍：

- 管理代理上的物理目标。有关如何创建、显示、更新和销毁物理目标，以及如何使用停用消息队列等主题的信息，请参见第 6 章 “管理物理目标”。
- 设置代理的安全性。有关用户验证、访问控制、加密、密码文件和审计日志等主题的信息，请参见第 7 章 “管理安全性”。

前提条件

使用 imqcmd 和 imqusermgr 命令行实用程序来管理代理。在管理代理前，必须执行以下操作：

- 使用 imqbrokerd 实用程序命令启动代理。在代理运行前，不能使用其他命令行实用程序。
- 确定要设置 Message Queue 管理用户，还是使用默认帐户。要使用管理命令，必须指定用户名和密码。

  安装 Message Queue 时，会安装默认的用户系统信息库，这是一个平面文件。该系统信息库包含两个默认条目：admin 用户和 guest 用户。如果要测试 Message Queue，可以使用默认用户名和密码 (admin/admin) 来运行 imqcmd 实用程序。

  如果要设置生产系统，则必须为管理用户设置验证和授权。有关设置基于文件的用户系统信息库或配置 LDAP 目录服务器的使用的信息，请参见第 7 章 “管理安全性”。在生产环境中，使用非默认用户名和密码是一种比较安全的做法。
- 如果要使用安全的代理连接，请在目标代理实例中设置并启用 ssladmin 服务。有关更多信息，请参见第 135 页上的 “使用基于 SSL 的服务”。

使用 imqcmd 实用程序

imqcmd 实用程序用于管理代理及其服务。

有关 imqcmd 命令的语法、子命令和选项的参考信息，请参见第 241 页上的第 13 章 “命令行参考”。在单独的第 283 页上的第 15 章 “物理目标属性参考”中，介绍了管理物理目标时可使用的参考信息。

显示帮助

要显示有关 imqcmd 实用程序的帮助，请使用 -h 或 -H 选项，而不要使用子命令。您无法获取有关特定子命令的帮助。

例如，以下命令显示有关 imqcmd 的帮助：

```
imqcmd -H
```

如果输入的命令行除了包含 -h 或 -H 选项外，还包含子命令或其他选项外，则 imqcmd 实用程序只处理 -h 或 -H 选项。命令行中的其他所有项均被忽略。

显示产品版本

要显示 Message Queue 产品版本，请使用 -v 选项。例如：

```
imqcmd -v
```

如果输入的命令行除了包含 -v 选项外，还包含子命令或其他选项，则 imqcmd 实用程序只处理 -v 选项。命令行中的其他所有项均被忽略。

指定用户名和密码

因为将对照用户系统信息库验证每个 imqcmd 子命令，所以每个 imqcmd 子命令都要求提供用户名和密码。唯一的例外是使用 -h 或 -H 选项来显示帮助的命令以及使用 -v 选项来显示产品版本的命令。

指定用户名

使用 -u 选项可指定管理用户名。如果省略了用户名，该命令会提示您输入它。例如，以下命令显示有关默认代理的信息：

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

为使本章中的示例便于阅读，我们将默认用户名 admin 显示为 -u 选项的参数。在生产环境中，应该使用自定义的用户名。

指定密码

使用以下方法之一指定密码：

- 创建密码文件 (passfile) 并在该文件中输入密码。在命令行中，使用 -passfile 选项提供密码文件的名称。
- 让命令提示您输入密码。

在 Message Queue 的以前版本中，可以使用 -p 选项在 imqcmd 命令行中指定密码。不赞成使用此选项，在今后的版本中会将其删除。

指定代理名和端口

imqcmd 的默认代理是本地主机上运行的代理，默认端口是 7676。

如果对在远程主机上运行的代理或监听非默认端口的代理执行命令，则必须使用 -b 选项指定代理的主机和端口。

示例

本节中的示例说明了 imqcmd 的用法。

第一个示例列出了在 localhost 端口 7676 上运行的代理的属性，因此不需要使用 -b 选项。该命令使用默认管理用户名 (admin) 并省略了密码，因此命令会提示您输入密码。

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

以下示例列出了在主机 myserver 端口 1564 上运行的代理的属性。用户名是 aladdin。（要使该命令起作用，需要更新用户系统信息库，将用户名 aladdin 添加到 admin 组中。）

```
imqcmd query bkr -b myserver:1564 -u aladdin
```

以下示例列出了在 localhost 端口 7676 上运行的代理的属性。命令的初始超时时间限制设置为 20 秒，超时后的重试次数设置为 7。用户密码在名为 myPassfile 的密码文件中，该文件位于调用命令时的当前目录中。

```
imqcmd query bkr -u admin -passfile myPassfile -rtm 20 -rtr 7
```

对于安全的代理连接，这些示例可能会包括 -secure 选项。 -secure 选项使 imqcmd 使用 ssladmin 服务 （如果已配置并启动了该服务）。

显示代理信息

要查询并显示某个代理的信息，请使用 query bkr 子命令。

下面是 query bkr 子命令的语法：

```
imqcmd query bkr -b hostName:portNumber
```

该子命令列出默认代理或指定主机和端口上的代理的当前属性设置。它还列出与指定代理连接且正在运行的代理 （在多代理群集中）。

例如：

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

提示输入密码后，此命令将产生类似如下内容的输出：

```
版本                                    3.6
实例名                                  imqbroker
主端口                                  7676
```

```
系统中当前的消息数                          0
系统中当前的消息大小（字节）                0

停用消息队列中的当前消息数量                0
停用消息队列中的当前消息字节总数            0

记录停用消息                                true
截断停用消息队列中的消息主体                false

系统中的最大消息数                          无限制 (-1)
系统中的最大消息大小                        无限制 (-1)
最大消息大小                                70m

自动创建队列                                true
自动创建主题                                true
自动创建队列的最大活动用户数    1
自动创建队列的最大备份用户数    0

群集代理列表（处于活动状态）
群集代理列表（处于已配置状态）
群集主代理
群集 URL

日志等级                                    INFO
日志清空并重新记录间隔（秒）               604800
日志清空并重新记录大小（字节）             无限制 (-1)
```

更新代理属性

可以使用 update bkr 子命令更新以下代理属性：

- imq.autocreate.queue
- imq.autocreate.topic
- imq.autocreate.queue.maxNumActiveConsumers
- imq.autocreate.queue.maxNumBackupConsumers
- imq.cluster.url
- imq.destination.DMQ.truncateBody
- imq.destination.logDeadMsgs
- imq.log.level
- imq.log.file.rolloversecs
- imq.log.file.rolloverbytes

- imq.system.max_count
- imq.system.max_size
- imq.message.max_size
- imq.portmapper.port

下面是 update bkr 子命令的语法：

```
imqcmd update bkr [-b hostName:portNumber]-o attribute=value [[-o
attribute=value1]...]
```

该子命令更改默认代理或指定主机和端口上的代理的指定属性。例如，以下命令禁止自动创建队列目标：

```
imqcmd update bkr -o "imq.autocreate.queue=false" -u admin
```

这些属性在第 14 章 “代理属性参考”中介绍。

暂停和恢复代理

启动代理后，可以使用 imqcmd 子命令控制代理的状态。

暂停代理

暂停代理时暂停的是代理的连接服务线程，从而使代理停止侦听连接端口。其结果是代理将无法再接受新连接、接收消息或分发消息。

但是，暂停代理不会暂停 admin 连接服务，因此您可以执行控制发送到代理的消息流所需的管理任务。暂停代理也不会暂停 cluster 连接服务。但是，群集中的消息传送依赖于群集中的不同代理所执行的传送功能。因此，暂停群集中的代理可能导致某些消息流的速度变慢。

下面是 pause bkr 子命令的语法：

```
imqcmd pause bkr [-b hostName:portNumber]
```

此命令暂停默认代理或指定主机和端口上的代理。

以下命令暂停在 myhost 端口 1588 上运行的代理。

```
imqcmd pause bkr -b myhost:1588 -u admin
```

也可以暂停单个连接服务和单个物理目标。有关更多信息，请参见第 102 页上的 “暂停和恢复连接服务”和第 114 页上的 “暂停和恢复物理目标”。

恢复代理

恢复代理将重新激活代理的服务线程，使代理恢复侦听端口。

下面是 resume bkr 子命令的语法：

```
imqcmd resume bkr [-b hostName:portNumber]
```

此子命令恢复默认代理或指定主机和端口上的代理。

以下命令将恢复在 localhost 端口 7676 上运行的代理。

```
imqcmd resume bkr -u admin
```

关闭并重新启动代理

关闭代理将正常终止代理进程。代理将停止接受新的连接和消息，并在完成现有消息的传送后终止代理进程。

下面是 shutdown bkr 子命令的语法：

```
imqcmd shutdown bkr [-b hostName:portNumber]
```

此子命令关闭默认代理或指定主机和端口上的代理。

以下命令将关闭在 ctrlsrv 端口 1572 上运行的代理：

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

使用 restart bkr 子命令可关闭并重新启动代理。下面是 restart bkr 子命令的语法：

```
imqcmd restart bkr [-b hostName:portNumber]
```

该子命令关闭并重新启动默认代理或指定主机和端口上的代理，重新启动时使用首次启动代理时指定的选项。要选择其他选项，请关闭代理，然后通过指定所需的选项来重新启动它。

显示代理度量

要显示有关代理的度量信息，请使用 metrics bkr 子命令。

下面是 metrics bkr 子命令的语法：

```
imqcmd metrics bkr [-b hostName:portNumber]
        [-m metricType] [-int interval] [-msp numSamples]
```

此子命令显示默认代理或指定主机和端口上的代理的代理度量。

可以使用 -m 选项来指定显示以下度量类型之一：

- **ttl** 显示流入和流出代理的消息和包的度量 （默认度量类型）。
- **rts** 显示消息和包流入和流出代理的速率的度量 （以一秒为衡量单位）。
- **cxn** 显示连接、虚拟内存堆和线程。

可以使用 -int 选项来指定显示度量的时间间隔 （以秒为单位）。默认值为 5 秒。

可以使用 -msp 选项来指定在输出中显示的样例的数量。默认为无限制 （无穷多）。

例如，要获得消息在 10 秒间隔内流入和流出代理的速率，请使用以下命令：

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

此命令产生的输出如下：

```
---------------------------------------------------------
 消息数 / 秒      消息字节 / 秒      数据包 / 秒      数据包字节 / 秒
 传入   传出      传入     传出      传入   传出      传入     传出
---------------------------------------------------------
 0      0         27       56        0      0         38       66
 10     0       7365       56        10     10      7457     1132
 0      0         27       56        0      0         38       73
 0      10        27     7402        10     20      1400     8459
 0      0         27       56        0      0         38       73
```

有关代理收集和报告的数据的更详细描述，请参见第 304 页上的 “代理范围内的度量”。

管理连接服务

imqcmd 实用程序包含可用于执行以下连接服务管理任务的子命令：

- 列出连接服务
- 显示连接服务信息
- 更新连接服务属性
- 显示连接服务度量
- 暂停和恢复连接服务

代理既支持来自应用程序客户端的连接，也支持来自管理客户端的连接。表 5-1 显示了 Message Queue 代理中当前可用的连接服务。如该表所示，每项服务都与它所使用的服务类型（对于应用程序客户端是 NORMAL，对于管理客户端是 ADMIN）和底层传输协议关联。

表 5-1 代理支持的连接服务

服务名称	服务类型	协议类型
jms	NORMAL	tcp
ssljms (Enterprise Edition)	NORMAL	tls（基于 SSL 的安全性）
httpjms (Enterprise Edition)	NORMAL	http
httpsjms (Enterprise Edition)	NORMAL	https（基于 SSL 的安全性）
admin	ADMIN	tcp
ssladmin (Enterprise Edition)	ADMIN	tls（基于 SSL 的安全性）

您可以使用 imqcmd 子命令将连接服务作为一个整体进行管理，也可以管理某项特定的连接服务。如果子命令的目标是某项特定服务，请使用 -n 选项来指定表 5-1 的“服务名”列中列出的某个名称。

列出连接服务

要列出代理中可用的连接服务，请使用 `list svc` 子命令。

下面是 `list svc` 子命令的语法：

```
imqcmd list svc [-b hostName:portNumber]
```

此子命令列出默认代理或指定主机和端口上的代理中的所有连接服务。

以下命令列出在 `localhost` 端口 `7676` 上运行的代理中的所有服务：

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

该命令将输出如下信息：

```
-----------------------------------------------
服务名          端口号            服务状态
-----------------------------------------------
admin           41844（动态）     正在运行
httpjms         -                 未知
httpsjms        -                 未知
jms             41843（动态）     正在运行
ssladmin        动态              未知
ssljms          动态              未知
```

显示连接服务信息

要查询并显示某项服务的信息，请使用 `query` 子命令。

下面是 `query svc` 子命令的语法：

```
imqcmd query svc -n serviceName [-b hostName:portNumber]
```

`query svc` 子命令显示有关在默认代理或指定主机和端口上的代理上运行的指定服务的信息。

例如：

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

提示输入密码后，此命令将产生类似如下内容的输出：

```
服务名                          jms
服务状态                        正在运行
端口号                          60920（动态）

当前已分配的线程数              0
当前的连接数                    0

最小线程数                      10
最大线程数                      1000
```

更新连接服务属性

可以使用 update 子命令更改表 5-2 中列出的一个或多个服务属性的值。

表 5-2 imqcmd 更新的连接服务属性

属性	描述
port	为要更新的服务指定的端口（不适用于 httpjms 和 httpsjms）。值为 0 表示端口由端口映射器动态分配。
minThreads	至少应分配给服务的线程数。
maxThreads	最多可分配给服务的线程数。

下面是 update 子命令的语法：

```
imqcmd update svc -n serviceName [-b hostName:portNumber]
        -o attribute=value  [-o attribute=value1]...
```

该子命令更新在默认代理或指定主机和端口上的代理中运行的指定服务的指定属性。有关服务属性的说明，请参见第 259 页上的 “连接属性”。

以下命令将至少应分配给 jms 服务的线程数更改为 20。

```
imqcmd update svc -n jms -o "minThreads=20" -u admin
```

显示连接服务度量

要显示有关某个服务的度量信息，请使用 metrics 子命令。

下面是 metrics 子命令的语法：

```
imqcmd metrics svc -n serviceName [-b hostName:portNumber] [-m metricType]
  [-int interval] [-msp numSamples]
```

此子命令显示默认代理或指定主机和端口上的代理中指定服务的度量。

使用 -m 选项指定要显示的度量类型：

- **ttl** 显示有关通过指定的连接服务流入和流出代理的消息和包的度量。（默认度量类型。）
- **rts** 显示有关消息和包通过指定的连接服务流入和流出代理的速率的度量（以一秒为衡量单位）。
- **cxn** 显示连接、虚拟内存堆和线程。

可以使用 -int 选项来指定显示度量的时间间隔 （以秒为单位）。默认值为 5 秒。

可以使用 -msp 选项来指定在输出中显示的样例的数量。默认为无限制 （无穷多）。

例如，要获得 jms 连接服务处理的消息和包的累计总数，请使用以下命令：

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

提示输入密码后，此命令将产生类似如下内容的输出：

```
-----------------------------------------------
消息          消息字节          数据包          数据包字节
传入   传出    传入     传出    传入   传出    传入     传出
-----------------------------------------------
164    100    120704   73600   282    383    135967   102127
657    100    483552   73600   775    876    498815   149948
```

有关使用 imqcmd 来报告连接服务度量的详细说明，请参见第 305 页上的 “连接服务度量”。

暂停和恢复连接服务

要暂停除管理服务之外的其他任何服务（不能暂停管理服务），请使用 pause svc 和 resume svc 子命令。

下面是 pause svc 子命令的语法：

```
imqcmd pause svc -n serviceName [-b hostName:portNumber]
```

此子命令暂停在默认代理或指定主机和端口上的代理中运行的指定服务。例如，以下命令暂停在默认代理上运行的 httpjms 服务。

```
imqcmd pause svc -n httpjms -u admin
```

暂停服务有如下影响：

- 代理将停止在已暂停的服务上接受新的客户端连接。如果 Message Queue 客户端尝试打开新的连接，将出现异常。
- 已暂停的服务上的所有现有连接都将保持活动状态，但是代理将暂停这些连接上的所有消息处理，直到服务恢复。（例如，在服务恢复之前，将禁止客户端尝试使用 send 方法来发送消息。）
- 代理已接收的任何消息的消息传送状态都将保留。（例如，事务不会中断，消息传送将在服务恢复之后恢复。）

要恢复服务，请使用 resume svc 子命令。

下面是 resume svc 子命令的语法：

```
imqcmd resume svc -n serviceName[-b hostName:portNumber]
```

此子命令恢复在默认代理或指定主机和端口上的代理中运行的指定服务。

获取有关连接的信息

imqcmd 实用程序包含可用来列出并获取连接信息的子命令。

list cxn 子命令列出指定服务名的全部连接。下面是 list cxn 子命令的语法：

```
imqcmd list cxn [-svn serviceName] [-b hostName:portNumber]
```

此子命令列出默认代理或指定主机和端口上的代理中指定服务名的所有连接。如果未指定服务名，将列出所有连接。

例如，以下命令列出默认代理上的所有连接：

```
imqcmd list cxn -u admin
```

提示输入密码后，此命令将产生类似如下内容的输出：

```
正在列出指定的代理上的所有连接：
-------------------------------------
主机                  主端口
-------------------------------------
localhost               7676
```

```
--------------------------------------------------------------------------------
连接 ID                  用户     服务       生成方       使用方         主机
--------------------------------------------------------------------------------
1964412264455443200   guest   jms        0          1            127.0.0.1
1964412264493829311   admin   admin      1          1            127.0.0.1

成功列出连接。
```

要查询并显示某项连接服务的信息，请使用 query 子命令。

```
query cxn -n connectionID [-b hostName:portNumber]
```

此子命令显示默认代理或指定主机和端口上的代理中的指定连接的信息。

例如：

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

提示输入密码后，此命令将产生类似如下内容的输出：

```
连接 ID         421085509902214374
用户            guest
服务            jms
生成方          0
使用方          1
主机            111.22.333.444
端口            60953
客户端 ID
客户端平台
```

管理长期订阅

使用 imqcmd 子命令可以通过执行下面一项或多项操作来管理代理的长期订阅：

- 列出长期订阅
- 清除长期订阅的所有消息
- 销毁长期订阅

长期订阅是指客户端注册为长期项的主题订阅；长期订阅有唯一标识，它要求代理保留该订阅的消息，即使订阅使用方变为非活动状态也是如此。通常情况下，代理只能在消息已过期的情况下删除为长期订户保留的消息。

要列出指定物理目标的长期订阅，请使用 list dur 子命令。下面是 list dur 子命令的语法：

```
imqcmd list dur -d destName
```

例如，以下命令使用本地主机默认端口上的代理，列出 SPQuotes 主题的所有长期订阅：

```
imqcmd list dur -d SPQuotes
```

对于主题的每个长期订阅，list dur 子命令都返回长期订阅的名称、用户的客户端 ID、排队发往该主题的消息数量以及长期订阅的状态 （活动 / 非活动）。例如：

```
名称          客户端 ID          消息数量       长期订户
                                              状态
-----------------------------------------------------------------
myDurable    myClientID        1              非活动
```

可以使用 list dur 子命令返回的信息标识您要销毁或要清除其消息的长期订阅。

purge dur 子命令清除带有指定客户端标识符的指定长期订阅的所有消息。下面是 purge dur 子命令的语法：

```
imqcmd purge dur -n subscrName -c clientID
```

destroy dur 子命令销毁带有指定客户端标识符的指定长期订阅。下面是 destroy dur 子命令的语法：

```
imqcmd destroy dur -n subscrName -c clientID
```

例如，以下命令销毁长期订阅 myDurable 和客户端 ID myClientID。

```
imqcmd destroy dur -n myDurable -c myClientID
```

管理事务

客户端应用程序启动的所有事务都由代理来跟踪。这些事务可以是简单 Message Queue 事务，也可以是分布式事务 （XA 资源）管理器管理的分布式事务。

每个事务都有一个 Message Queue 事务 ID，这是一个 64 位数字，用于唯一标识代理上的事务。分布式事务也有一个分布式事务 ID (XID)，长度可达 128 字节，由分布式事务管理器指定。Message Queue 负责维护 Message Queue 事务 ID 与 XID 之间的关联。

对于分布式事务而言，当失败时，事务可能停留在 PREPARED 状态，而不会提交。因此，作为管理员，您需要监视事务的状态，并回滚或提交那些停留在 PREPARED 状态的事务。

要列出代理跟踪的所有事务，请使用 `list txn` 命令。下面是 `list tx` 子命令的语法：

```
imqcmd list txn
```

例如，以下命令列出某个代理中的所有事务。

```
imqcmd list txn
```

对于每个事务， `list` 子命令将返回事务 ID、状态、用户名、消息数或确认数以及创建时间。例如：

```
-----------------------------------------------------------------
事务 ID           状态       用户名      # Msgs/     创建时间
                                        # Acks
-----------------------------------------------------------------

64248349708800  PREPARED  guest      4/0         1/30/02 10:08:31 AM
64248371287808  PREPARED  guest      0/4         1/30/02 10:09:55 AM
```

该命令显示了代理中的所有事务，包括本地事务和分布式事务。只能提交或回滚处于 PREPARED 状态的事务。只有当您知道该事务由于失败而停留在 PREPARED 状态，而且分布式事务管理器当前没有提交该事务时才可以这样做。

例如，如果代理的 auto-rollback 属性设置为 false （请参见第 261 页上的表 14-2），则必须在代理启动时手动提交或回滚处于 PREPARED 状态的事务。

`list` 子命令还分别显示事务中生成和确认的消息数 (`#Msgs/#Acks`)。提交事务之前不会传送这些消息，也不会处理确认。

`query` 子命令可以显示上述信息以及其他许多值：客户端 ID、连接标识和分布式事务 ID (XID)。下面是 `query txn` 子命令的语法：

```
imqcmd query txn -n transactionID
```

例如，以下示例生成如下所示的输出：

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

```
客户端 ID
连接                guest@192.18.116.219:62209->jms:62195
创建时间            1/30/02 10:08:31 AM
确认数              0
消息数              4
状态                PREPARED
事务 ID             64248349708800
用户名              guest
XID
6469706F6C7369646577696E6465723130313234313431313030373230
```

使用 commit 和 rollback 子命令可以提交和回滚分布式事务。正如前文所述，只能提交和回滚处于 PREPARED 状态的事务。

下面是 commit 子命令的语法：

```
imqcmd commit txn -n transactionID
```

例如：

```
imqcmd commit txn -n 64248349708800
```

下面是 rollback 子命令的语法：

```
imqcmd rollback txn -n transactionID
```

有关更多信息，请参见第 261 页上的表 14-2 中的 imq.transaction.autorollback 属性。

也可以配置代理，使它在启动时自动回滚处于 PREPARED 状态的事务。

第 6 章

管理物理目标

本章介绍如何使用 imqcmd 实用程序管理物理目标。 Message Queue 消息通过代理上的物理目标路由到使用方客户端。代理管理与物理目标关联的内存和持久性存储器并设置它们的行为。

在群集中，您在某个代理上创建一个物理目标后，群集会将该物理目标传播到所有代理。应用程序客户端可以订阅某一主题，或者使用群集中任何代理上的队列，因为这些代理会协同工作，在整个群集中路由消息。但是，该消息的持久性和确认只由最初生成消息的代理负责管理。

本章包含以下主题：

表 13-5 中提供了用于管理物理目标和完成上述任务的 imqcmd 子命令的完整参考信息。

有关物理目标的介绍，请参见 Message Queue 技术概述。

注 每当与物理目标交互时，客户端应用程序都会使用 Destination 对象。为了实现提供者无关性和可移植性，客户端通常使用管理员创建的目标对象，它们被称为目标受管理对象。可以配置受管理对象，使其可供客户端应用程序使用，如第 8 章 “管理受管理对象”中所述。

使用命令实用程序

可以使用 Message Queue 命令实用程序 (imqcmd) 来管理物理目标。 imqcmd 命令的语法与管理其他代理服务时的语法一样。

有关 imqcmd 及其子命令和选项的完整参考信息，请参见第 241 页上的第 13 章 “命令行参考”。

子命令

表 6-1 列出了 imqcmd 子命令，本章介绍了这些子命令的用法。有关这些子命令的参考信息，请参见第 250 页上的 “物理目标管理”。

表 6-1 命令实用程序的物理目标子命令

子命令和参数	描述
compact dst	压缩一个或多个物理目标的基于文件的数据存储。
create dst	创建物理目标。
destroy dst	销毁物理目标。
list dst	列出代理中的物理目标。
metrics dst	显示物理目标度量。
pause dst	暂停代理中的一个或多个物理目标。
purge dst	清除物理目标中的所有消息，但不销毁该物理目标。
query dst	查询并显示有关物理目标的信息。
resume dst	恢复代理中的一个或多个暂停的物理目标。
update dst	更新目标属性。

创建物理目标

要创建物理目标，请使用 imqcmd create 子命令。下面是 create 子命令的语法：

```
create dst -t destType -n destName [-o property=value] [-o property=value1]...
```

创建物理目标时，请指定以下内容：

- 物理目标类型， t （主题）或 q （队列）。
- 物理目标名称。命名规则如下：
  - 名称必须仅包含字母数字字符。不能包含空格。
  - 名称可以字母字符、下划线字符 (_) 或美元符号 ($) 开头。名称不能以字符串 "mq" 开头。
- 物理目标属性的非默认值。

也可以在更新物理目标时设置属性。

许多物理目标属性都影响代理的内存资源和消息流。例如，可以指定能发送到物理目标的生成方数量，它们可以发送的消息数量和大小，以及达到物理目标限制时代理应做出的响应。这些限制与由代理配置属性控制的代理范围的限制类似。

以下属性既适用于队列目标，也适用于主题目标：

- maxNumMsgs。指定物理目标中允许的未使用消息的最大数量。
- maxTotalMsgBytes。指定物理目标中允许用于未使用消息的内存的最大总量（以字节为单位）。
- limitBehavior。指定当达到内存限制阈值时代理的响应方式。
- maxBytesPerMsg。指定物理目标中允许的任何单个消息的最大大小 （以字节为单位）。
- maxNumProducers。指定物理目标的生成方的最大数量。
- consumerFlowLimit。指定在一批中传送给使用方的消息的最大数量。
- isLocalOnly。仅适用于代理群集。指定物理目标不能在其他代理上复制，因而限定只能将消息传送到本地使用方 （连接到创建物理目标的代理的使用方）。
- useDMQ。指定是丢弃物理目标的停用消息，还是将其放在停用消息队列中。

以下属性只适用于队列目标：

- maxNumActiveConsumers。指定来自队列目标的负载平衡传送中可以处于活动状态的最大使用方数。

- maxNumBackupConsumers。指定当来自队列目标的负载平衡传送中出现任何错误时，可以代替活动使用方的最大备份使用方数。
- localDeliveryPreferred。仅应用于代理群集中的负载平衡队列传送。指定仅当本地代理中没有使用方时才将消息传送到远程使用方。

有关物理目标属性的完整参考信息，请参见第 283 页上的第 15 章 “物理目标属性参考”。

对于自动创建的目标，在代理的实例配置文件中设置默认属性值。有关自动创建的属性的参考信息，请参见第 262 页上的表 14-3。

➤ 创建物理目标

- 要创建队列目标，请输入以下命令：

```
imqcmd create dst -n myQueue -t q -o "maxNumActiveConsumers=5"
```

- 要创建主题目标，请输入以下命令：

```
imqcmd create dst -n myTopic -t t -o "maxBytesPerMsg=5000"
```

列出物理目标

可以获得有关物理目标的当前属性值、与物理目标关联的生成方和使用方的数量以及消息传送度量 （例如，物理目标中消息的数量和大小）的信息。

要找到您希望获取其信息的物理目标，请使用 list dst 子命令列出代理中的所有物理目标。下面是 list dst 子命令的语法：

```
list dst [-t destType] [-tmp]
```

该命令列出指定类型的物理目标。目标类型 (-t) 选项的值可以是 q （queue，队列）或 t （topic，主题）。

如果省略目标类型，将列出所有类型的物理目标。

list dst 子命令可以选择指定要列出的目标的类型或包含临时目标 （使用 -tmp 选项）。临时目标由客户端创建，通常用于接收发送到其他客户端的消息的回复。

例如，要获得在 myHost 端口 4545 上运行的代理中的所有物理目标列表，请输入以下命令：

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

除非指定目标类型 t （表示只包括主题），否则，除其他任何物理目标外，还始终会显示停用消息队列 mq.sys.dmq 的信息。

显示有关物理目标的信息

要获得有关物理目标当前属性的信息，请使用 query dst 子命令。下面是 query dst 子命令的语法：

```
query dst -t destType -n destName
```

此命令列出有关指定类型和名称的目标的信息。例如，以下命令显示有关队列目标 XQueue 的信息：

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

此命令将产生类似如下内容的输出：

```
------------------------------------
目标名称              目标类型
------------------------------------
XQueue                队列

在指定的代理上：

-------------------------
主机          主端口
-------------------------
localhost     7676

目标名称                          XQueue
目标类型                          Queue
目标状态                          正在运行
以管理方式创建                    true

当前的消息数                      0
当前的消息大小总计                0
当前的生成方数                    0
当前的活动用户数                  0
当前的备份用户数                  0

最大消息数                        无限制 (-1)
消息的最大总计大小（字节）        无限制 (-1)
每条消息的最大大小                无限制 (-1)
最大生成方数                      100
最大活动用户数                    1
最大备份用户数                    0

限制行为                          REJECT_NEWEST
用户流限制                        1000
是本地目的地                      false
首选本地传递                      false
使用停用消息队列                  true
```

输出还显示与目标关联的生成方和使用方的数量。对于队列目标，该数量包括活动使用方和备份使用方。

可以使用 update dst 子命令更改一个或多个属性的值 （请参见第 114 页上的 “更新物理目标属性”）。

更新物理目标属性

可以使用 update dst 子命令更改物理目标的属性，并使用 -o 选项指定要更新的属性。下面是 update dst 子命令的语法：

```
update dst -t destType -n destName -o property=value [[-o property=value1]...]
```

此命令更新指定目标上指定属性的值。属性名可以是表 15-1 中列出的任何属性。

可以使用 -o 选项多次来更新多个属性。例如，以下命令将 maxBytesPerMsg 属性更改为 1000，同时将 MaxNumMsgs 属性更改为 2000：

```
imqcmd update dst -t q -n myQueue -o "maxBytesPerMsg=1000"
            -o "maxNumMsgs=2000" -u admin
```

有关可以更新的属性列表，请参见第 15 章 “物理目标属性参考”。

不能使用 update dst 子命令更新物理目标的**类型**或更新 isLocalOnly 属性。

注 停用消息队列是专用的物理目标，其属性与其他目标的属性稍有不同。有关更多信息，请参见第 118 页上的 “配置停用消息队列的使用”。

暂停和恢复物理目标

可暂停物理目标以控制从生成方到目标的消息传送，或从目标到使用方的消息传送，或者同时控制二者。特别是，可暂停到目标的消息流，以防止当消息的生成速度明显快于使用速度时，目标由于所包含的消息过多而过载。压缩物理目标之前，必须先暂停它。

要暂停流入或流出物理目标的消息传送，请使用 pause dst 子命令。下面是 pause dst 子命令的语法：

```
pause dst [-t destType -n destName] [-pst pauseType]
```

对于指定类型和名称的目标，此子命令暂停将消息传送给使用方 (-pst CONSUMERS)，或暂停从生成方传送消息 (-pst PRODUCERS)，或同时暂停二者 (-pst ALL)。如果未指定目标类型和名称，则暂停所有物理目标。默认值为 ALL。

示例：

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

imqcmd pause dst -n myTopic -t t -pst CONSUMERS -u admin
```

要恢复向暂停目标的传送，请使用 resume dst 子命令。下面是 resume dst 子命令的语法：

```
resume dst [-t destType -n destName]
```

此子命令恢复向指定类型和名称的暂停目标传送消息。如果未指定目标类型和名称，则恢复所有目标。

示例：

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

在代理群集中，物理目标实例位于群集的各个代理中。必须逐一暂停各个目标。

清除物理目标

可以清除物理目标上当前排队的所有消息。清除物理目标意味着目标上存储的所有消息都将被删除。

当堆积的消息占用了过多的系统资源时，可能需要清除消息。当某个队列没有注册的使用方客户端，但仍然接收大量消息时，可能会发生这种情况。如果某个主题的长期订户始终处于非活动状态也可能发生这种情况。在上述两种情况下，都没有必要保留消息。

要清除物理目标中的消息，请使用 purge dst 子命令。下面是 purge dst 子命令的语法：

```
purge dst -t destType -n destName
```

此子命令清除指定类型和名称的物理目标中的消息。

示例：

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

imqcmd purge dst -n myTopic -t t -u admin
```

如果关闭代理后不希望在重新启动代理时传送过时消息，请使用 -reset messages 选项清除过时消息；例如：

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

这样可以避免重新启动代理后清理目标的麻烦。

在代理群集中，物理目标实例位于群集的各个代理中。必须分别清理每个目标。

销毁物理目标

要销毁物理目标，请使用 destroy dst 子命令。下面是 destroy dst 子命令的语法：

```
destroy dst -t destType -n destName
```

该子命令销毁指定类型和名称的物理目标。

示例：

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

销毁物理目标将清除该目标中的所有消息并将该目标从代理中删除，此操作是不可逆的。

不能销毁停用消息队列。

压缩物理目标

如果使用基于文件的数据存储作为消息的持久性存储，则可以监视磁盘使用情况并在必要时压缩磁盘。

可构建基于文件的消息存储，以便将消息存储在与保存它的物理目标对应的目录中。在每个物理目标的目录中，大多数消息都存储在一个由大小可变的记录组成的文件 （即大小可变的记录文件）中。（为减少文件碎片，大小超过可配置阈值的消息将存储在单独的文件中。）

由于各种大小的消息先持久化，然后从大小可变的记录文件中删除，因此可能会在文件中形成漏洞，从而使得文件中的空闲记录无法再次使用。

为管理未使用的空闲记录，命令实用程序包含用于监视每个物理目标的磁盘利用率的子命令，以及在磁盘利用率降低时用于回收空闲磁盘空间的子命令。

监视物理目标的磁盘利用率

要监视物理目标的磁盘利用率，请使用如下所示的命令：

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

此命令产生如下所示的输出：

```
---------------------------------------
保留的       已用的      利用率
---------------------------------------
806400      804096     99
1793024     1793024    100
2544640     2518272    98
```

子命令输出中的各列具有以下含义：

表 6-2 物理目标磁盘占用度量

度量	描述
保留的	所有记录使用的磁盘空间 （以字节为单位），其中包括保存活动消息的记录以及等待再次使用的空闲记录。
已用的	保存活动消息的记录使用的磁盘空间 （以字节为单位）。
利用率	已用的磁盘空间除以保留的磁盘空间所得的商。比率越高，可用于保存活动消息的磁盘空间就越多。

回收未使用的物理目标磁盘空间

磁盘利用模式取决于使用特定物理目标的消息发送应用程序的特征。根据流入和流出物理目标的消息的相对数量，以及消息的相对大小，保留的磁盘空间可能会随时间而增加。

如果消息生产率大于消息使用率，则通常应该重新使用空闲记录，且应提高利用率。但是，如果消息生产率等于或小于消息使用率，则应降低利用率。

通常，应尽量使保留的磁盘空间保持一个稳定的量，并使磁盘利用率保持较高水平。一般规则是：如果系统达到稳定状态 （其中保留的磁盘空间量较为稳定），且磁盘利用率较高 （大于 75%），则不必回收未使用的磁盘空间。如果系统达到稳定状态，而利用率较低 （低于 50%），则可压缩磁盘以回收空闲记录占用的磁盘空间。

使用 compact dst 子命令压缩数据存储。下面是 compact dst 子命令的语法：

```
compact dst [-t destType -n destName]
```

此子命令为指定类型和名称的物理目标压缩基于文件的数据存储。如果未指定目标类型和名称，则会压缩所有目标。压缩之前必须先暂停物理目标。

如果保留的磁盘空间随时间持续增加，请通过设置目标内存限制属性和限制行为来重新配置目标的内存管理 （请参见第 283 页上的表 15-1）。

➤ 回收未使用的物理目标磁盘空间

1. 暂停目标。

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

2. 压缩磁盘。

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

3. 恢复物理目标。

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

如果未指定目标类型和名称，则会为**所有**物理目标执行此操作。

配置停用消息队列的使用

停用消息队列 mq.sys.dmq 是系统创建的物理目标，它保存代理的停用消息及其他物理目标。停用消息队列是一个工具，用于监视、调整系统效率以及故障排除。有关术语“停用消息”的定义以及停用消息队列的更详细介绍，请参见 Message Queue 技术概述。

代理在启动时会自动创建停用消息队列。如果代理无法处理消息或者消息的有效期到期，则代理会将消息放入该队列中。另外，其他物理目标也可以使用停用消息队列来保存丢弃的消息。通过使用停用消息队列，可以提供有利于排除系统故障的信息。

配置停用消息队列的使用

默认情况下，物理目标配置为使用停用消息队列。可以通过设置物理目标属性 useDMQ 来禁止或允许物理目标使用停用消息队列。

下面的示例创建一个名为 myDist 的队列，该队列在默认情况下使用停用消息队列：

```
imqcmd create dst -n -myDist -t q
```

下面的示例禁止上述队列使用停用消息队列：

```
imqcmd update dst -n myDist -t q -o useDMQ=false
```

通过设置 imq.autocreate.destination.useDMQ 代理属性，可以允许或禁止代理中所有自动创建的物理目标使用停用消息队列。

配置和管理停用消息队列

可以使用 Message Queue 命令实用程序 (imqcmd) 像管理其他队列那样来管理停用消息队列，但有一些不同之处。例如，由于停用消息队列是系统创建的，因此您不能创建、暂停或销毁它。另外，如表 6-3 中所示，停用消息队列的默认值有时与普通队列的默认值不同。

停用消息队列属性

配置停用消息队列与配置其他队列相似，但某些物理目标属性不应用默认值或具有不同的默认值。表 6-3 列出了停用消息队列以独特方式处理的队列属性。

表 6-3 标准物理目标属性的停用消息队列处理

属性	停用消息队列的独特处理
limitBehavior	停用消息队列的默认值为 REMOVE_OLDEST。（其他队列的默认值为 REJECT_NEWEST。）停用消息队列不支持流控制。
localDeliveryPreferred	不适用于停用消息队列。
maxNumMsgs	停用消息队列的默认值为 1000。其他队列的默认值为 -1（无限制）。
maxNumProducers	不适用于停用消息队列。
maxTotalMsgBytes	对于停用消息队列，默认值为 10 MB。对于其他队列，默认值为 -1（无限制）。
isLocalOnly	在代理群集中，停用消息队列始终是本地物理目标，此属性永久性地设置为 true。但是，本地代理的停用消息队列可包含由群集中其他代理的客户端生成的消息（如果本地代理将这些消息标记为停用）。

消息内容

代理可以将完整的消息放入停用消息队列中，也可以丢弃消息主体内容，而只保留标题和属性数据。默认情况下，停用消息队列存储完整的消息。

如果要减小停用消息队列的大小并且不打算恢复停用消息，请考虑将 imq.destination.DMQ.truncateBody 代理属性设置为 true：

```
imqcmd update bkr -o imq.destination.DMQ.truncateBody=true
```

这将丢弃消息主体，而只保留标题和属性数据。

启用停用消息日志记录

默认情况下将禁用停用消息日志记录。启用停用消息日志记录使代理可以记录以下事件：

- 代理将消息移至停用消息队列中
- 代理丢弃来自停用消息队列以及来自不使用停用消息队列的任何物理目标的消息
- 物理目标达到其限制

以下命令启用停用消息日志记录：

```
imqcmd update bkr -o imq.destination.logDeadMsgs=true
```

停用消息日志记录适用于使用停用消息队列的所有物理目标。不能对单个物理目标启用或禁用日志记录。

第 7 章

管理安全性

作为管理员，您可以配置用户系统信息库来验证用户、定义访问控制、配置用来加密客户端 / 代理通信的安全套接字层 (Secure Socket Layer, SSL) 连接服务，并设置代理启动时使用的密码文件。

本章包含以下各节：

验证用户

当用户尝试连接到代理时，代理将通过检查所提供的名称和密码来对用户进行验证。如果名称和密码与特定于代理的 （每个代理均配置为需要查阅）用户系统信息库中的名称和密码相匹配，则允许该用户与代理建立连接。

您负责维护系统信息库中用户、用户组及用户密码的列表。各个代理实例可以使用不同的用户系统信息库。本节将说明如何创建、填充和管理系统信息库。

系统信息库可以是以下类型之一：

- Message Queue 附带的平面文件系统信息库

  此类型的用户系统信息库非常易于使用。您可以使用用户管理器实用程序 (`imqusermgr`) 来填充和管理系统信息库。要启用验证，您需要用每个用户的名称和密码以及用户组的名称填充用户系统信息库。

有关设置和管理用户系统信息库的详细信息，请参见 “使用平面文件用户系统信息库”。

- LDAP 服务器

  这可以是现有的或新的 LDAP 目录服务器，这种服务器使用 LDAP v2 或 v3 协议。它并不像平面文件系统信息库那样易于使用，但它的可伸缩性更好，因此更适用于生产环境。

  如果您目前使用的是 LDAP 用户系统信息库，请使用 LDAP 供应商提供的工具来填充和管理用户系统信息库。有关更多信息，请参见第 127 页上的 “使用 LDAP 服务器管理用户系统信息库”。

使用平面文件用户系统信息库

Message Queue 提供了一个平面文件用户系统信息库和一个命令行工具，即，用户管理器实用程序 (`imqusermgr`)，您可以使用它来填充和管理平面文件用户系统信息库。以下各节介绍平面文件用户系统信息库以及如何使用用户管理器实用程序来填充和管理该系统信息库。

创建用户系统信息库

平面文件用户系统信息库是特定于实例的。默认的用户系统信息库 （名为 `passwd`）是为启动的每个代理实例自动创建的。此用户系统信息库所在的目录由与该系统信息库相关联的代理实例的名称标识 （请参见附录 A “Message Queue 数据在特定平台上的位置”）：

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

创建的系统信息库默认有两个条目。表 7-1 中的每一行显示一个条目。

表 7-1 用户系统信息库中的初始条目

用户名	密码	组	状态
admin	admin	admin	处于活动状态
guest	guest	anonymous	处于活动状态

这些初始条目使得 Message Queue 代理安装后即可使用，而不需要管理员的干预：

- 初始 `guest` 用户条目使客户端可以使用默认的 `guest` 用户名和密码连接到代理实例。

- 利用初始 admin 用户条目，您可以通过 imqcmd 命令使用默认的 admin 用户名和密码管理代理实例。您应更新此初始条目以更改密码 （请参见第 126 页上的“更改默认的管理员密码”）。

以下各节说明如何填充和管理平面文件用户系统信息库。

用户管理器实用程序

使用 Message Queue 用户管理器实用程序 (imqusermgr)，您可以编辑或填充平面文件用户系统信息库。本节介绍了用户管理器实用程序。下文说明如何使用 imqusermgr 子命令完成特定任务。

有关 imqusermgr 命令的完整参考信息，请参见第 13 章 “命令行参考”。

使用用户管理器之前，请谨记以下事项：

- 如果特定于代理的用户系统信息库不存在，您必须启动相应的代理实例来创建此系统信息库。
- 必须在安装了代理的主机上运行 imqusermgr 命令。
- 您必须具有写入系统信息库的适当权限：即，在 Solaris 和 Linux 上，您的身份必须是 root 用户或首次创建代理实例的用户。

注 以下各节中的示例采用默认代理实例。

子命令

imqusermgr 命令包含子命令 add、 delete、 list 和 update。

- add 子命令将用户和关联的密码添加到指定的 （或默认的）代理实例系统信息库中，并可以选择指定用户所属的组。子命令语法如下所示：

  add [-i *instanceName*] -u *userName* -p *passwd* [-g *group*] [-s]

- delete 子命令从指定的 （或默认的）代理实例系统信息库中删除指定用户。子命令语法如下所示：

  delete [-i *instanceName*] -u *userName* [-s] [-f]

- list 子命令显示有关指定的 （或默认的）代理实例系统信息库中指定用户或所有用户的信息。子命令语法如下所示：

  list [-i *instanceName*] [-u *userName*]

- update 子命令更新指定的 （或默认的）代理实例系统信息库中指定用户的密码和 / 或状态。子命令语法如下所示：

```
update [-i instanceName] -u userName -p passwd [-a state] [-s] [-f]
update [-i instanceName] -u userName -a state [-p passwd] [-s] [-f]
```

命令选项

表 7-2 列出了 imqusermgr 命令的选项。

表 7-2 imqusermgr 选项

选项	描述
-a *activeState*	指定用户是否处于活动状态 (true/false)。true 表示处于活动状态。这是默认值。
-f	执行操作，无需用户确认。
-h	显示用法帮助。不执行命令行上的其他选项。
-i *instanceName*	指定命令要应用到的代理实例名。如果未指定，则采用默认实例名 imqbroker。
-p *passwd*	指定用户密码。
-g *group*	指定用户组。有效值包括 admin、user 和 anonymous。
-s	设置无提示模式。
-u *userName*	指定用户名。
-v	显示版本信息。不执行命令行上的其他选项。

组

在代理实例的用户系统信息库中添加用户条目时，您可以指定以下三个预定义组之一：admin、user 或 anonymous。如果不指定任何组，则默认属于 user 组。应按如下方式指定组：

- **admin 组。** 用于代理管理员。默认情况下，指定到该组中的用户可以配置和管理代理。您可以为 admin 组指定多个用户。
- **user 组。** 用于普通（非管理）Message Queue 客户端用户。多数客户端用户都属于 user 组。默认情况下，该组中的用户可以生成发往所有主题和队列的消息，使用来自所有主题和队列的消息，以及浏览任意队列中的消息。
- **anonymous 组。** 用于那些不想使用代理已知的用户名（可能是因为客户端应用程序不知道要使用的实际用户名）的 Message Queue 客户端。此帐户类似于大多数 FTP 服务器中的匿名帐户。每次只能为 anonymous 组分配一个用户。与 user 组不同，应限定 anonymous 组的访问权限，或者在部署时删除该组中的用户。

要更改用户所属的组，必须删除该用户的条目，然后为该用户添加另一个条目并为其指定新组。

您不能重命名或删除这些系统创建的组，也不能创建新组。但是，您可以指定访问规则，定义该组的成员可以执行的操作。有关更多信息，请参见第 129 页上的 “授权用户：访问控制属性文件”。

用户状态

向系统信息库中添加用户时，用户的默认状态是活动的。要使用户处于非活动状态，您必须使用 update 命令。例如，以下命令将使用户 JoeD 处于非活动状态：

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

处于非活动状态的用户条目将保留在系统信息库中，但不能打开新连接。当某个用户处于非活动状态时，如果您试图添加具有相同名称的另一个用户，操作将失败。必须删除处于非活动状态的用户条目，或者更改新用户的名称，或者为新用户指定一个不同的名称。这样可以防止添加重复的用户名。

用户名和密码的格式

用户名和密码必须遵循以下原则：

- 用户名不能包含星号 (*)、逗号 (,)、冒号 (:) 或换行符或回车符。
- 用户名或密码必须至少包含一个字符。
- 如果用户名或密码包含空格，必须将整个用户名或密码用引号引起来。
- 除了命令 shell 限定的命令行中最多可输入的字符数之外，密码或用户名的长度没有限制。

填充和管理用户系统信息库

要在系统信息库中添加用户，可以使用 add 子命令。例如，以下命令向默认代理实例用户系统信息库添加用户名为 Katharine、密码为 sesame 的用户。

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

要从系统信息库中删除用户，可以使用 delete 子命令。例如，以下命令删除用户 Bob：

```
imqusermgr delete -u Bob
```

要更改用户的密码或状态，可以使用 update 子命令。例如，以下命令将 Katharine 的密码更改为 aladdin：

```
imqusermgr update -u Katharine -p aladdin
```

要列出一个或所有用户的相关信息，可以使用 list 命令。以下命令显示名为 isa 的用户的相关信息：

```
imqusermgr list -u isa
```

```
% imqusermgr list -u isa

代理实例的用户系统信息库：imqbroker
---------------------------------
用户名      组      活动状态
---------------------------------
isa        admin    true
```

以下命令列出所有用户的相关信息：

```
imqusermgr list
```

```
% imqusermgr list

代理实例的用户系统信息库：imqbroker
--------------------------------------
用户名         组             活动状态
--------------------------------------
admin         admin         true
guest         anonymous     true
isa           admin         true
testuser1     user          true
testuser2     user          true
testuser3     user          true
testuser4     user          false
testuser5     user          false
```

更改默认的管理员密码

为安全起见，应该将 admin 的默认密码更改为只有您自己知道的密码。以下命令将 mybroker 代理实例的默认管理员密码从 admin 更改为 grandpoobah。

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

要快速确认此更改是否已生效，可以在代理实例运行时运行任何命令行工具。例如，以下命令将提示您输入密码：

```
imqcmd list svc mybroker -u admin
```

输入新密码 (grandpoobah) 将会被接受；而输入旧密码则会失败。

更改密码后，使用任何 Message Queue 管理工具 （包含管理控制台）时都应该提供新密码。

使用 LDAP 服务器管理用户系统信息库

要使用 LDAP 服务器来管理用户系统信息库，请执行以下任务：

- 编辑实例配置文件
- 为管理员设置访问控制

编辑实例配置文件

要让代理使用目录服务器，请设置代理实例配置文件 config.properties 中某些属性的值。设置这些属性后，每当用户试图连接到代理实例或执行信息传送操作时，代理实例都会在 LDAP 服务器中查询有关用户和组的信息。

实例配置文件位于代理实例目录下的某个目录中。路径的格式如下：

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

有关实例目录在特定操作系统中的位置的信息，请参见附录 A “Message Queue 数据在特定平台上的位置”。

➤ 编辑配置文件以使用 LDAP 服务器

1. 通过设置以下属性，指定您使用的是 LDAP 用户系统信息库：

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

2. 设置 imq.authentication.type 属性，确定密码是以 Base64 (basic) 还是 MD5 (digest) 编码的形式从客户端传递给代理。使用 LDAP 目录服务器管理用户系统信息库时，必须将验证类型设置为 basic。例如，

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

3. 还必须设置控制 LDAP 访问的代理属性。这些属性存储在代理的实例配置文件中。第 78 页上的 “安全服务”对这些属性进行了讨论，第 270 页上的 “安全属性”对这些属性进行了总结。

   Message Queue 使用 JNDI API 与 LDAP 目录服务器进行通信。有关这些属性的语法及其所引用的术语的详细信息，请查阅 JNDI 文档。 Message Queue 使用 Sun JNDI LDAP 提供者并使用简单验证。

   Message Queue 支持 LDAP 验证故障转移：可以指定要尝试进行验证的 LDAP 目录服务器列表 （请参见 `imq.user.repos.ldap.server` 属性的参考信息）。

   有关说明如何设置 LDAP 用户系统信息库相关属性的示例，请参见代理的 `config.properties` 文件。

4. 必要时还需要编辑访问控制属性文件中的用户 / 组和规则。有关使用访问控制属性文件的详细信息，请参见第 129 页上的 “授权用户：访问控制属性文件”。

5. 如果您希望代理在连接验证和组搜索时通过 SSL 与 LDAP 目录服务器进行通信，您需要在 LDAP 服务器中激活 SSL，然后在代理配置文件中设置以下属性：

   - 指定 LDAP 服务器进行 SSL 通信时所使用的端口。例如：

     ```
     imq.user_repository.ldap.server=myhost:7878
     ```

   - 将代理属性 `imq.user_repository.ldap.ssl.enabled` 设置为 `true`。

     使用多个 LDAP 目录服务器时，可使用 `ldap://` 指定每个额外的目录服务器。例如：

     ```
     imq.user_repository.ldap.server=myHost:7878
     ldap://otherHost:7878 ...
     ```

     用空格分隔每个额外的目录服务器。对于其他与 LDAP 相关的属性，列表中的所有目录服务器必须使用相同的值。

为管理员设置访问控制

要创建管理用户，请使用访问控制属性文件指定能够创建 `ADMIN` 连接的用户和组。必须在 LDAP 目录中预定义这些用户和组。

能够创建 `ADMIN` 连接的任何用户或组都可以执行管理命令。

➤ **设置管理用户**

1. 通过将代理属性 `imq.accesscontrol.enabled` 设置为 `true` （这是默认值）来允许使用访问控制文件。

   `imq.accesscontrol.enabled` 属性允许使用访问控制文件。

2. 打开访问控制文件 `accesscontrol.properties`。附录 A “Message Queue 数据在特定平台上的位置”中列出了该文件的位置。

   该文件包含一个如下所示的条目：

```
服务连接访问控制
##################################
connection.NORMAL.allow.user=*
connection.ADMIN.allow.group=admin
```

   列出的条目是示例。请注意， `admin` 组存在于基于文件的用户系统信息库中，而非默认存在于 LDAP 目录中。您必须将在 LDAP 目录中定义的组的名称替换为要为其授予 Message Queue 管理员权限的组的名称。

3. 要为用户授予 Message Queue 管理员权限，请按如下所示输入用户名：

   `connection.ADMIN.allow.user=`*userName*[[`,`*userName2*]...]

4. 要为组授予 Message Queue 管理员权限，请按如下所示输入组名称：

   `connection.ADMIN.allow.group=`*groupName*[[`,`*groupName2*]...]

授权用户：访问控制属性文件

访问控制属性文件 （Access Control Properties File， ACL 文件）包含一些规则，用来指定用户和用户组可以执行的操作。您可以编辑 ACL 文件，将操作限定给某些用户和组。各个代理实例可以使用不同的 ACL 文件。

无论用户信息是放在平面文件用户系统信息库中还是放在 LDAP 用户系统信息库中，都会使用 ACL 文件。当客户端应用程序执行以下操作之一时，代理将检查其 ACL 文件：

- 创建连接
- 创建生成方

- 创建使用方
- 浏览队列

代理将检查 ACL 文件以确定是授权生成请求的用户还是用户所属的组来执行该操作。

如果对 ACL 文件进行编辑，新的设置将在下一次代理检查该文件以验证授权时生效。编辑完文件后，不需要重新启动代理。

创建访问控制属性文件

ACL 文件是特定于实例的。每次启动代理实例时，都会在该实例目录中创建一个名为 `accesscontrol.properties` 的默认文件。该文件的路径具有如下格式（请参见附录 A “Message Queue 数据在特定平台上的位置”）：

```
.../instances/brokerInstanceName/etc/accesscontrol.properties
```

ACL 文件的格式与 Java 属性文件类似。首先定义文件的版本，然后指定访问控制规则，规则分为三部分：

- 连接访问控制
- 物理目标访问控制
- 物理目标自动创建访问控制

`version` 属性定义 ACL 属性文件的版本，不能更改此条目。

```
version=JMQFileAccessControlModel/100
```

下面介绍指定访问控制的 ACL 文件的三个组成部分，然后说明访问规则的基本语法并介绍如何计算权限。

访问规则语法

在 ACL 属性文件中，访问控制用于定义特定用户或组对受保护的资源（如物理目标和连接服务）具有哪些访问权限。访问控制由一个规则或一组规则组成，每个规则都由一个 Java 属性表示：

这些规则的基本语法如下：

resourceType.*resourceVariant*.*operation*.*access*.*principalType*=*principals*

表 7-3 介绍了语法规则的元素。

表 7-3 访问规则的语法元素

元素	描述
resourceType	以下选项之一：`connection`、`queue` 或 `topic`。
resourceVariant	*resourceType* 指定的类型的一个实例。例如，myQueue。通配符 (*) 可用于表示所有连接服务类型或所有物理目标。
operation	值取决于所设置的访问规则的类型。
access	以下选项之一：`allow` 或 `deny`。
principalType	以下选项之一：`user` 或 `group`。有关更多信息，请参见第 124 页上的“组”。
principals	可能具有规则左侧指定的访问权限的人。如果 `principalType` 为 `user`，则可能是单个用户或以逗号分隔的用户列表；如果 `principalType` 为 `group`，则可能是单个组或以逗号分隔的组列表。通配符 (*) 可用于表示所有用户或所有组。

下面是一些访问规则示例：

- 以下规则表示所有用户都可以向名为 q1 的队列发送消息。

  ```
  queue.q1.produce.allow.user=*
  ```

- 以下规则表示任何用户都可以向任何队列发送消息。

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

注 要指定非 ASCII 的用户、组或目标名称，请使用 Unicode 转义符 (\uXXXX) 表示法。如果在您编辑并保存的 ACL 文件中，这些名称采用了非 ASCII 编码，则可以通过 Java `native2ascii` 工具将此文件转换为 ASCII。有关更多详细信息，请参见

http://java.sun.com/j2se/1.4/docs/guide/intl/faq.html

权限的计算方式

当文件中存在多个访问规则时，将按如下方式计算权限：

- 具体的访问规则将覆盖一般访问规则。应用以下两条规则之后，所有用户都可以向所有队列发送消息，但是 Bob 不能向 `tq1` 发送消息。

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

  queue.tq1.produce.deny.user=Bob
  ```

- 指定给显式 *principal* 的访问权限将覆盖指定给 * *principal* 的访问权限。以下规则将拒绝 Bob 向 tq1 发送消息，但允许其他人发送。

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

  queue.tq1.produce.deny.user=Bob
  ```

- 用户的 * *principal* 规则将覆盖组的对应 * *principal* 规则。例如，以下两条规则允许所有通过验证的用户向 tq1 发送消息。

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

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

- 授予用户的访问权限将覆盖授予用户所属组的访问权限。在以下示例中，即使 Bob 是 User 的成员，他也不能生成发送到 tq1 的消息。User 的其他所有成员都可以生成此类消息。

  ```
  queue.tq1.produce.allow.group=User

  queue.tq1.produce.deny.user=Bob
  ```

- 任何未通过访问规则明确授予的访问权限均默认为被拒绝。例如，如果 ACL 文件不包含任何访问规则，则所有用户的所有操作都将被拒绝。

- 如果同时允许和拒绝同一个用户或组的访问权限，则访问权限将相互抵消。例如，以下两条规则将使 Bob 无法浏览 q1：

  ```
  queue.q1.browse.allow.user=Bob

  queue.q1.browse.deny.user=Bob
  ```

  以下两条规则阻止 User 组使用 q5 中的消息。

  ```
  queue.q5.consume.allow.group=User

  queue.q5.consume.deny.group=User
  ```

- 如果多条规则等号左侧的内容都相同，则只有最后一条规则起作用。

用于连接服务的访问控制

ACL 属性文件中的连接访问控制部分包含代理连接服务的访问控制规则。连接访问控制规则的语法如下：

```
connection.resourceVariant.access.principalType=principals
```

为 *resourceVariant* 定义了两个值：NORMAL 和 ADMIN。这些预定义的值是您唯一能够授予访问权限的连接服务类型。

默认的 ACL 属性文件授予所有用户访问 NORMAL 连接服务的权限，并授予 admin 组中的用户访问 ADMIN 连接服务的权限：

```
connection.NORMAL.allow.user=*
connection.ADMIN.allow.group=admin
```

如果您使用的是基于文件的用户系统信息库，则默认组由用户管理器实用程序创建 admin。如果您使用的是 LDAP 用户系统信息库，则可以通过执行以下操作之一来使用默认 ACL 属性文件：

- 在 LDAP 目录中定义一个名为 admin 的组。
- 使用在 LDAP 目录中定义的一个或多个组的名称来替换 ACL 属性文件中的名称 admin。

您可以对连接访问权限加以限定。例如，以下规则拒绝 Bob 访问 NORMAL，但允许其他人访问：

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

可以使用星号 (*) 指定所有通过验证的用户或组。

使用 ACL 属性文件来授权访问 ADMIN 连接的方式与使用基于文件的用户系统信息库和 LDAP 用户系统信息库不同，如下所示：

- **基于文件的用户系统信息库**
  - 如果禁用访问控制， admin 组中的用户将具有 ADMIN 连接权限。
  - 如果启用访问控制，请编辑 ACL 文件。明确授予用户或组访问 ADMIN 连接服务的权限。
- **LDAP 用户系统信息库。**如果您使用的是 LDAP 用户系统信息库，请执行下列所有操作：
  - 启用访问控制。
  - 编辑 ACL 文件并提供可以建立 ADMIN 连接的用户或组的名称。指定在 LDAP 目录服务器中定义的任何用户或组。

对物理目标的访问控制

访问控制属性文件的目标访问控制部分包含基于物理目标的访问控制规则。这些规则决定谁 （用户 / 组）可以在哪里 （物理目标）执行什么 （操作）。这些规则控制的访问类型包括向队列发送消息、向主题发布消息、从队列接收消息、订阅主题以及浏览队列中的消息。

默认情况下，任何用户或组都可以对任何物理目标进行任意类型的访问。您可以添加更多特定的目标访问规则或编辑默认的规则。本节的余下部分介绍物理目标访问规则的语法，您必须理解这些语法才能编写自己的规则。

目标规则的语法如下：

resourceType.*resourceVariant*.*operation*.*access*.*principalType*=*principals*

表 7-4 介绍了这些元素：

表 7-4 物理目标访问控制规则的元素

组件	描述
resourceType	可以是 `queue` 或 `topic`。
resourceVariant	某个物理目标名或所有物理目标 (*)，星号表示所有队列或所有主题。
operation	可以是 `produce`、 `consume` 或 `browse`。
access	可以是 `allow` 或 `deny`。
principalType	可以是 `user` 或 `group`。

可以将访问权限授予一个或多个用户和 / 或一个或多个组。

以下示例说明了不同类型的物理目标访问控制规则：

- 允许所有用户向任意 queue 目标发送消息。

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

- 拒绝 `user` 组的任何成员订阅 `Admissions` 主题的能力。

  ```
  topic.Admissions.consume.deny.group=user
  ```

对自动创建的物理目标的访问控制

ACL 属性文件最后一部分所包括的访问规则指定代理将为哪些用户和组自动创建物理目标。

当用户在尚不存在的物理目标上创建生成方或使用方时，如果已启用代理的自动创建属性，则代理将会创建该目标。

默认情况下，任何用户或组都有权让代理为其自动创建一个物理目标。此权限由以下规则指定：

```
queue.create.allow.user=*

topic.create.allow.user=*
```

您可以编辑 ACL 文件以限制此类访问权限。

物理目标自动创建访问规则的一般语法如下：

resourceType.create.*access*.*principalType*=*principals*

其中 *resourceType* 为 queue 或 topic。

例如，以下规则允许代理为除 Snoopy 之外的每个用户自动创建 topic 目标。

```
topic.create.allow.user=*

topic.create.deny.user=Snoopy
```

请注意，物理目标自动创建规则的效果必须与物理目标访问规则的效果一致。例如， 1) 如果您更改目标访问规则，禁止任何用户向目标发送消息； 2) 但是启用了目标的自动创建，则如果目标不存在，代理**将**创建一个物理目标，但**不会**向该目标发送任何消息。

使用基于 SSL 的服务

基于安全套接字层 (Secure Socket Layer, SSL) 标准的连接服务在客户端与代理之间发送加密的消息。本节将说明如何设置基于 SSL 的连接服务。

Message Queue 支持以下基于安全套接字层 (Secure Socket Layer, SSL) 标准的连接服务：

- ssljms、 ssladmin 和 cluster 均通过 TCP/IP 使用。
- httpsjms 通过 HTTP 使用。

这些连接服务允许对客户端与代理之间发送的消息进行加密。 Message Queue 支持基于自签名服务器证书或签名证书的 SSL 加密。

要使用基于 SSL 的连接服务，需要使用密钥工具实用程序 (imqkeytool) 生成专用密钥 / 公共密钥对。此实用程序将公共密钥嵌入自签名证书，此证书将传递给请求连接到代理的任何客户端，客户端需要使用该证书建立加密连接。

尽管 Message Queue 的基于 SSL 的连接服务在概念上很相似，但它们的设置方法却不尽相同。

本节后面部分将介绍如何通过 TCP/IP 建立安全连接。

基于 SSL 的连接服务（适用于通过 HTTP 连接的用户）(`httpsjms`) 使客户端和代理可以通过 HTTPS 隧道 Servlet 方式建立安全连接。有关通过 HTTP 建立安全连接的信息，请参见第 321 页上的附录 C “HTTP/HTTPS 支持”。

TCP/IP 的安全连接服务

以下基于 SSL 的连接服务通过 TCP/IP 提供直接、安全的连接：

- `ssljms` 服务可通过安全的加密连接在客户端与代理之间传送消息。
- `ssladmin` 服务将在 Message Queue 命令实用程序 (`imqcmd`) 与代理之间创建一个安全的加密连接。不支持管理控制台 (`imqadmin`) 安全连接。
- `cluster` 服务可通过安全的加密连接在群集中的代理之间传送消息并提供代理之间的通信（请参见第 168 页上的 “代理之间的安全连接”）。

配置自签名证书的使用

本节介绍如何使用自签名证书来设置基于 SSL 的服务。

如果需要更严密的验证，您可以使用由证书颁发机构验证的签名证书。先按照本节中的步骤执行操作，然后转至第 141 页上的 “配置签名证书的使用”执行其他步骤。

➤ 设置基于 SSL 的连接服务

1. 生成自签名证书。
2. 在代理中启用 `ssljms`、`ssladmin` 或 `cluster` 连接服务。
3. 启动代理。
4. 配置并运行客户端（仅适用于 `ssljms` 连接服务）。

设置 `ssljms` 和 `ssladmin` 连接服务的过程基本相同，不同之处在于步骤 4，即配置并运行客户端的步骤。

下文详细介绍了每个步骤。

步骤 1：生成自签名证书

带有自签名证书的 Message Queue SSL 支持用于保护所传输数据的安全性，并假定客户端正在与已知且可信任的服务器进行通信。

运行密钥工具实用程序为代理生成自签名证书。在 UNIX® 系统上，可能需要以超级用户 (root) 身份运行密钥工具，以便具有创建密钥存储的权限。

可以对 ssljms、 ssladmin 或 cluster 连接服务使用相同的证书。

在命令提示符下输入以下内容：

```
imqkeytool -broker
```

实用程序会提示您输入密钥存储密码。

```
Generating keystore for the broker ...
Enter keystore password:
```

然后，实用程序将提示您提供信息，用于标识拥有该证书的代理。您提供的信息将构成 X.500 标识名。下表列出并说明了一些提示，并为每个提示提供了一个示例。值区分大小写，并且可以包含空格。

表 7-5 自签名证书所需的标识名信息

提示	描述	示例
您的姓名是什么？	X.500 commonName (CN)。输入运行代理的服务器的全限定名称。	myhost.sun.com
您所在部门的名称是什么？	X.500 organizationalUnit (OU)。输入部门或分部的名称。	purchasing
您的工作单位的名称是什么？	X.500 organizationName (ON)。大型工作单位的名称，如公司或政府机构。	My Company, Inc.
您所在城市或地区的名称是什么？	X.500 localityName (L)。	San Francisco
您所在国家或省市的名称是什么？	X.500 stateName (ST)。输入国家或省市的全称，不要使用缩写。	California
此单位的两字母国家 / 地区代码是什么？	X.500 country (C)。	US

输入信息后，密钥工具将显示这些信息以便于您确认。例如：

```
Is CN=mqserver.sun.com, OU=purchasing, O=My Company, Inc., L=San
Francisco, ST=California, C=US correct?
```

要重新输入值，请接受默认值或输入 no ；要接受当前值并继续，请输入 yes。确认后，密钥工具将生成密钥对而暂停其他操作。

接下来，密钥工具要求您输入密码以锁定特定密钥对 （密钥密码）。按 Return 键响应此提示，以使用同一密码作为密钥密码和密钥存储密码。

注 请记住您提供的密码。启动代理时必须提供此密码，代理才能打开密钥存储。可以将密钥存储密码存储到密码文件中 （请参见第 144 页上的 “使用密码文件”）。

imqkeytool 命令运行 JDK keytool 实用程序生成自签名证书并将生成的证书置于 Message Queue 的密钥存储中。密钥存储所在的目录因操作系统而异，如附录 A “Message Queue 数据在特定平台上的位置”中所示。

密钥存储的格式与 JDK1.2 keytool 实用程序支持的格式相同。

以下是 Message Queue 密钥存储的可配置属性：

- imq.keystore.file.dirpath。适用于基于 SSL 的服务：指定包含密钥存储文件的目录的路径。有关默认值的信息，请参见附录 A “Message Queue 数据在特定平台上的位置”。
- imq.keystore.file.name。适用于基于 SSL 的服务：指定密钥存储文件的名称。
- imq.keystore.password。适用于基于 SSL 的服务：指定密钥存储密码。

您可能需要重新生成密钥对才能解决某些问题，例如：

- 您忘记了密钥存储密码。
- 启动代理时，基于 SSL 的服务初始化失败，并出现异常 java.security.UnrecoverableKeyException:Cannot recover key。

  出现异常的原因可能是因为您提供的密钥密码与您在第 137 页上的 “步骤 1：生成自签名证书”中生成自签名证书时的密钥存储密码不同。

➤ 重新生成密钥对

1. 删除代理的密钥存储。有关密钥存储的位置，请参见附录 A “Message Queue 数据在特定平台上的位置”。
2. 再次运行 imqkeytool，按第 137 页上的 “步骤 1：生成自签名证书”中所述生成密钥对。

步骤 2：在代理中启用基于 SSL 的服务

要在代理中启用基于 SSL 的服务，您需要将 ssljms （或 ssladmin）添加到 imq.service.activelist 属性中。

注 基于 SSL 的 cluster 连接服务是使用 imq.cluster.transport 属性启用的，而不是使用 imq.service.activelist 属性启用的。请参见第 168 页上的 “代理之间的安全连接”。

➤ 在代理中启用基于 SSL 的服务

1. 打开代理的实例配置文件。

   实例配置文件位于一个由代理实例名称 (*instanceName*) 标识的目录中，而该实例名称与此配置文件是相关联的 （请参见附录 A “Message Queue 数据在特定平台上的位置”）：

   .../instances/*instanceName*/props/config.properties

2. 为 imq.service.activelist 属性添加一个条目 （如果此条目尚不存在），并将基于 SSL 的服务包括在列表中。

   默认情况下，此属性包括 jms 和 admin 连接服务。您需要添加 ssljms 或 ssladmin 连接服务或同时添加这两者 （取决于您要激活的服务）：

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

步骤 3：启动代理

启动代理并提供密钥存储密码。可以通过以下方法之一提供密码：

- 让代理启动时提示您输入密码：

  imqbrokerd
  Please enter Keystore password:*myPassword*

- 将密码放在密码文件中，如第 144 页上的 “使用密码文件”中所述。将密码放在密码文件中并设置属性 imq.passfile.enabled=true 后，执行以下操作之一：

  - 将密码文件的位置传递给 imqbrokerd 命令：

    imqbrokerd -passfile /*tmp*/*myPassfile*

  - 启动代理时不使用 -passfile 选项，但使用以下两个代理配置属性指定密码文件的位置：

    imq.passfile.dirpath=/*tmp*

    imq.passfile.name=*myPassfile*

启用带有 SSL 的代理或客户端时，您可能会注意到它在几秒钟内使用了大量的 CPU 资源。这是因为 Message Queue 使用 JSSE （Java Secure Socket Extension， Java 安全套接扩展）来实现 SSL。 JSSE 使用 `java.security.SecureRandom` 生成随机数。此方法需要大量时间生成初始随机数初始化向量，这就是您看到 CPU 使用率增加的原因。生成初始化向量后， CPU 的使用率将降到正常水平。

步骤 4：配置并运行基于 SSL 的客户端

最后，配置客户端以使用安全连接服务。有两类在 TCP/IP 之上实现的安全连接方案：

- 使用 `ssljms` 的应用程序客户端
- 使用 `ssladmin` 的 Message Queue 管理客户端 （如 `imqcmd`）

下面分别介绍这两种客户端。

使用 ssljms 的应用程序客户端

必须确保客户端的类路径中有必要的安全套接扩展 (Secure Socket Extension, JSSE) `.jar` 文件，还需要告知该文件使用 `ssljms` 连接服务。

1. 如果您的客户端不使用 J2SDK1.4 （它内置 JSSE 和 JNDI 支持），请确保客户端的类路径中有以下 `.jar` 文件：

   `jsse.jar`、`jnet.jar`、`jcert.jar` 和 `jndi.jar`

2. 请确保客户端的类路径中有以下 Message Queue `.jar` 文件：

   `imq.jar` 和 `jms.jar`

3. 启动客户端并连接到代理的 ssljms 服务。执行上述操作的方法之一是输入以下命令：

   `java -DimqConnectionType=TLS` *clientAppName*

   设置 `imqConnectionType`，通知连接使用 SSL。

   有关在客户端应用程序中使用 `ssljms` 连接服务的详细信息，请参见 Message Queue Developer's Guide for Java Clients 中有关使用受管理对象的章节。

使用 ssladmin 的管理客户端 (imqcmd)

您可以通过在使用 `imqcmd` 时包含 `-secure` 选项来建立一个安全的管理连接。例如：

imqcmd list svc -b *hostName*:*portNumber* -u *adminName* -secure

其中， *adminName* 是 Message Queue 用户系统信息库中的有效条目，该命令将提示您输入密码。（如果使用的是平面文件系统信息库，请参见第 126 页上的 “更改默认的管理员密码”。）

列出连接服务是查看 ssladmin 服务是否正在运行的一种方法，您可以通过此方法成功建立安全管理连接，如以下输出所示：

```
列出指定的代理上的所有服务：

主机                    主端口
localhost               7676

服务名                  端口号              服务状态
admin                   33984（动态）       正在运行
httpjms                 -                   未知
httpsjms                -                   未知
jms                     33983（动态）       正在运行
ssladmin                35988（动态）       正在运行
ssljms                  动态                未知

成功列出服务。
```

配置签名证书的使用

签名证书可以提供比自签名证书更严密的服务器验证。要实现签名证书，请将其安装到密钥存储中，然后对 Message Queue 客户端进行配置，使其在建立与 imqbrokerd 的 SSL 连接时请求签名证书。

您只能在客户端与代理之间实现签名证书，而不能在群集中的多个代理之间实现签名证书。

后面的说明假定您已经执行了第 136 页上的 “配置自签名证书的使用”中所述的步骤。在按照这些说明执行操作时，访问 http://java.sun.com 中有关 J2SE keytool 和 X.509 证书的信息可能会有所帮助。

步骤 1：获取和安装签名证书

➤ 获取签名证书

1. 使用 J2SE keytool 为刚刚生成的自签名证书生成一个证书签名请求 (Certificate Signing Request, CSR)。

   下面是一个示例：

```
keytool -certreq -keyalg RSA -alias imq -file certreq.csr
        -keystore /etc/imq/keystore -storepass myStorePassword
```

CSR 现在将证书封装到文件 `certreq.csr` 中。

2. 通过以下方法之一生成或请求签名证书：

   - 由众所周知的证书颁发机构 (Certificate Authority, CA) 签署证书，如 Thawte 或 Verisign。有关此过程的详细信息，请参见 CA 文档。
   - 使用 SSL 签名软件包亲自对证书进行签名。

   最终的签名证书是一个 ASCII 字符序列。如果从 CA 收到签名证书，它可能是电子邮件附件或消息文本。

3. 在收到签名证书后，请将其保存到文件中。

   以下说明使用示例名 `broker.cer` 来表示代理证书。

➤ 安装签名证书

1. 检查 `$JAVA_HOME/lib/security/cacerts` 以查明 J2SE 是否默认支持您的 CA，如下所示：

   ```
   keytool -v -list -keystore $JAVA_HOME/lib/security/cacerts
   ```

   此命令列出系统密钥存储中的根 CA。

   如果您的 CA 已列出，请跳过下一步。

2. 如果 J2SE 不支持您的 CA，请将证书颁发机构的根证书导入到 `imqbrokerd` 密钥存储中。

   下面是一个示例：

   ```
   keytool -import -alias ca -file ca.cer -noprompt -trustcacerts
           -keystore /etc/imq/keystore -storepass myStorePassword
   ```

   `ca.cer` 值是从 CA 获取的 CA 根证书。

   如果您使用的是 CA 测试证书，则可能需要导入测试 CA 根证书。您的 CA 应提供有关如何获取测试 CA 根副本的说明。

3. 将签名证书导入到密钥存储中以替换原来的自签名证书。

   例如：

   ```
   keytool -import -alias imq -file broker.cer -noprompt -trustcacerts
           -keystore /etc/imq/keystore -storepass myStorePassword
   ```

   `broker.cer` 值是包含从 CA 收到的签名证书的文件。

`imqbrokerd` 密钥存储现在包含一个用于 SSL 连接的签名证书。

步骤 2：配置客户端运行时以请求签名证书

➤ 配置 Java 客户端运行时

默认情况下， Message Queue 客户端运行时信任 imqbrokerd 并接受提供给它的任何证书。现在，您必须将客户端运行时配置为请求签名证书，并确保客户端信任对该证书进行签名的 CA。

1. 要对客户端进行配置以使其向 imqbrokerd 请求有效的签名证书，请将客户端的 ConnectionFactory 对象的 imqSSLIsHostTrusted 属性设置为 false。
2. 尝试建立与 imqbrokrd 的 SSL 连接，如第 140 页上的 “步骤 4：配置并运行基于 SSL 的客户端”中所述。

   如果代理的证书是由众所周知的 CA 签名的，则连接很可能会成功，您可以跳过下一步。如果连接由于证书验证错误而失败，请执行下一步。
3. 在客户端的信任存储中安装签名 CA 的根证书，如以下各节所述。

有三个选项可用于配置具有信任存储的客户端：

- 将根 CA 安装到默认系统 cacerts 文件中。
- 将根 CA 安装到可选系统文件 jssecacerts 中。这是推荐选项。
- 将根 CA 安装到任何密钥存储文件中，并将客户端配置为使用该文件作为其信任存储。

下面几部分包含有关如何使用这些选项来安装 Verisign Test Root CA 的示例。根 CA 包含在名为 testrootca.cer 的文件中。这些示例假定 J2SE 安装在 /usr/j2se 中。

安装到默认的系统 cacerts 文件中

本示例将根 CA 安装到文件 $JAVA_HOME/usr/jre/lib/security/cacerts 中。

```
keytool -import -keystore /usr/j2se/jre/lib/security/cacerts
        -alias VerisignTestCA -file testrootca.cer -noprompt
        -trustcacerts -storepass myStorePassword
```

默认情况下，客户端搜索此密钥存储，因此无需对客户端进行进一步的配置。

安装到 jssecacerts 中

本示例将根 CA 安装到文件 $JAVA_HOME/usr/jre/lib/security/jssecacerts 中。

```
keytool -import -keystore /usr/j2se/jre/lib/security/jssecacerts
        -alias VerisignTestCA -file testrootca.cer -noprompt
        -trustcacerts -storepass myStorePassword
```

默认情况下，客户端搜索此密钥存储，因此无需对客户端进行进一步的配置。

安装到其他文件中

本示例将根 CA 安装到文件 /home/smith/.keystore 中。

```
keytool -import -keystore /home/smith/.keystore
         -alias VerisignTestCA -file testrootca.cer -noprompt
         -trustcacerts -storepass myStorePassword
```

默认情况下，客户端不搜索此密钥存储，因此您必须向客户端提供信任存储的位置。要执行此操作，请在客户端运行后设置 Java 系统属性 javax.net.ssl.trustStore。例如：

```
javax.net.ssl.trustStore=/home/smith/.keystore
```

使用密码文件

多种类型的命令都需要密码。在表 7-6 中，第一列列出了需要密码的命令，第二列列出了需要密码的原因。

表 7-6 使用密码的命令

命令	用途	密码的用途
imqbrokerd	启动代理	访问基于 JDBC 的持久性数据存储、 SSL 证书密钥存储或 LDAP 用户系统信息库
imqcmd	管理代理	验证被授权使用此命令的管理用户
imqdbmgr	管理基于 JDBC 的数据存储	访问数据存储

可以在**密码文件**中指定这些密码，并使用 -passfile 选项指定该文件的名称。下面是 -passfile 选项的格式：

```
imqbrokerd -passfile myPassfile
```

注 在以前的版本中，您可以使用 -p、 -password、 -dbpassword 和 -ldappassword 选项在命令行指定密码。不赞成使用这些选项，在今后的版本中会将它们删除。在当前的发行版中，其中某个选项在命令行中的值将取代密码文件中相关联的值。

安全性问题

以交互方式指定密码以响应提示是最安全的密码指定方法，除非他人也可以看到您的监视器。您也可以在命令行中指定密码文件。但是，以非交互方式使用命令时，必须使用密码文件。

密码文件是未加密的，因此，您必须设置其权限以防止未经授权的访问。设置权限以限制可以查看该文件的用户，但为启动代理的用户提供读取权限。

密码文件内容

密码文件是包含一组属性和值的简单文本文件。每个值都是由某一命令使用的密码。

密码文件可包含表 7-7 中所示的密码：

表 7-7 密码文件中的密码

密码	受影响的命令	描述
imq.imqcmd.password	imqcmd	为 imqcmd 命令行指定管理员密码。需要对每个命令验证此密码。
imq.keystore.password	imqbrokerd	为基于 SSL 的服务指定密钥存储密码。
imq.persist.jdbc.password	imqbrokerd imdbmgr	指定用于打开数据库连接的密码（如有必要）。
imq.user_repository.ldap.password	imqbrokerd	指定与代理的标识名（用于绑定到已配置的 LDAP 用户系统信息库）相关联的密码。

样例密码文件是 Message Queue 产品的一部分。有关样例文件位置的信息，请参见附录 A “Message Queue 数据在特定平台上的位置”。

创建审计日志

Message Queue 仅在 Enterprise Edition 中支持审计日志记录。启用审计日志记录后， Message Queue 将为下列类型的事件生成记录：

- 启动、关闭、重新启动以及删除代理实例
- 用户验证和授权

- 重置持久性存储
- 创建、清除以及销毁物理目标
- 以管理方式销毁长期订户

要将审计记录记录到 Message Queue 代理日志文件中，请将 `imq.audit.enabled` 代理属性设置为 `true`。日志中的所有审计记录都包含关键字 `AUDIT`。

第 8 章

管理受管理对象

受管理对象封装特定于提供者的配置和命名信息，以便开发可以从一个 JMS 提供者移植到另一个 JMS 提供者的客户端应用程序。通常，Message Queue 管理员为客户端应用程序创建受管理对象，以便在获取用于发送和接收消息的代理连接时使用。

本章介绍了如何使用对象管理器实用程序 (`imqobjmgr`) 来创建和管理受管理对象。本章包含以下各节：

对象存储

受管理对象放在易于使用的对象存储中，客户端应用程序可以通过 Java 命名和目录接口 (Java Naming and Directory Interface, JNDI) 从该对象存储中访问这些对象。可以使用两种类型的对象存储：标准的轻量目录访问协议 (Lightweight Directory Access Protocol, LDAP) 目录服务器或本地文件系统中的目录。

LDAP 服务器对象存储

对于生产消息传送系统，建议将 LDAP 服务器用作对象存储。 LDAP 服务器针对分布式系统而设计，它提供了在生产环境中非常有用的安全功能。

许多供应商都提供 LDAP 实现。要使用 Message Queue 管理工具管理 LDAP 服务器上的对象存储，可能首先需要配置服务器，使其可以存储 Java 对象并执行 JNDI 查找；有关详细信息，请参见随 LDAP 实现一起提供的文档。

要将 LDAP 服务器用作对象存储，必须指定表 8-1 中所示的属性。这些属性分为以下几个类别：

- **初始上下文。** `java.naming.factory.initial` 属性指定在服务器上执行 JNDI 查找的初始上下文。此属性值对于给定的 LDAP 对象存储是固定的。
- **位置。** `java.naming.provider.url` 属性指定 LDAP 服务器的 URL 和目录路径。必须验证指定的目录路径是否存在。
- **安全性。** `java.naming.security.principal`、`java.naming.security.credentials` 和 `java.naming.security.authentication` 属性管理尝试访问对象存储的呼叫者的验证。这些属性的准确格式和值因 LDAP 服务提供者而异；请参见随 LDAP 实现一起提供的文档以了解详细信息，并确定是所有操作都需要安全信息，还是只有更改存储数据的操作需要安全信息。

表 8-1 LDAP 对象存储属性

属性	描述
`java.naming.factory.initial`	JNDI 查找的初始上下文 示例： `com.sun.jndi.ldap.LdapCtxFactory`
`java.naming.provider.url`	服务器 URL 和目录路径 示例： `ldap://mydomain.com:389/ou=mqobjs,o=myapp` 其中受管理对象存储在目录 `/myapp/mqobjs` 中。
`java.naming.security.principal`	用于验证呼叫者的主体标识 此属性的格式取决于验证方案：例如， `uid=homerSimpson,ou=People,o=mq` 如果未指定此属性，行为将由 LDAP 服务提供者决定。

表 8-1 LDAP 对象存储属性 （续）

属性	描述
java.naming.security.credentials	验证主体的凭证 此属性的值取决于验证方案：例如，它可能是散列密码、纯文本密码、密钥或证书。 如果未指定此属性，行为将由 LDAP 服务提供者决定。
java.naming.security.authentication	验证的安全级别 此属性的值为 none、 simple 或 strong 关键字中的一个。例如，如果指定 simple，则当缺少任何主体或凭证值时，系统都会提示您。这样您可以使用一种更安全的方法来提供身份信息。 如果未指定此属性，行为将由 LDAP 服务提供者决定。

文件系统对象存储

Message Queue 也支持将本地文件系统中的目录用作受管理对象的对象存储。虽然不建议在生产系统中使用这种方法，但该方法的优点是非常易于在开发环境中使用。但是，请注意，如果要将目录用作部署于多个计算机节点上的客户端的集中式对象存储，则所有这些客户端都必须能够访问该目录。此外，可以访问该目录的所有用户均可使用 Message Queue 管理工具来创建和管理受管理对象。

要将文件系统目录用作对象存储，必须指定表 8-2 中所示的属性。这些属性与上述 LDAP 对象存储属性的一般含义相同；不同的是， java.naming.provider.url 属性指定保存该对象存储的目录的路径。此目录必须存在，并且 Message Queue 管理工具用户以及将访问该存储的客户端应用程序用户必须对该目录具有正确的访问权限。

表 8-2 文件系统对象存储属性

属性	描述
java.naming.factory.initial	JNDI 查找的初始上下文 示例： com.sun.jndi.fscontext.RefFSContextFactory
java.naming.provider.url	目录路径 示例： file:///C:/myapp/mqobjs

受管理对象的属性

Message Queue 受管理对象有两种基本类型：

- **连接工厂**供客户端应用程序在创建代理连接时使用。
- **目标**表示客户端应用程序可以与之交换 （发送和检索）消息的代理中的位置。

上述每种类型的受管理对象都具有一些特定属性，用于确定对象的属性和行为。本节介绍了如何使用对象管理器命令行实用程序 (imqobjmgr) 来设置这些属性；也可以使用 GUI 管理控制台进行设置，如第 2 章中所述 （请参见第 52 页上的 “使用受管理对象”）。

连接工厂属性

客户端应用程序使用**连接工厂**受管理对象来创建与代理交换消息时使用的连接。连接工厂的属性定义了它所创建的所有连接的属性。创建连接之后，将无法更改其属性；因此，配置连接属性的唯一方法就是设置用于创建连接的连接工厂的属性。

Message Queue 定义了两类连接工厂对象：

- `ConnectionFactory` 对象支持标准的消息传送和非分布式事务。
- `XAConnectionFactory` 对象支持分布式事务。

这两类对象共享相同的配置属性，可以通过这些属性来优化资源、性能和消息吞吐量。第 16 章 “受管理对象属性参考”详细列出并说明了这些属性，以下各节将对这些属性进行讨论：

- 第 150 页上的 “连接处理”
- 第 153 页上的 “客户端身份验证”
- 第 154 页上的 “可靠性和流控制”
- 第 155 页上的 “队列浏览器和服务器会话”
- 第 155 页上的 “标准消息属性”
- 第 156 页上的 “消息头覆盖”

连接处理

连接处理属性指定了要连接到的消息服务器地址，以及如何检测连接故障并尝试重新连接 （如果要求）。第 288 页上的表 16-1 概要介绍了这些属性。

消息服务器地址列表

最重要的连接处理属性是 imqAddressList，该属性指定了要与之建立连接的一个或多个代理。该属性的值是一个字符串，它包含一个消息服务器地址或者以逗号分隔的多个地址 （如果是代理群集）。服务器地址可以使用各种寻址方案，具体情况取决于要使用的连接服务 （请参见第 72 页上的 “连接服务”）和建立连接的方法：

- mq 使用代理的端口映射器为 jms 或 ssljms 连接服务动态指定端口。
- 使用 jms 连接服务， mqtcp 可以绕过端口映射器，而直接连接到指定端口。
- 使用 ssljms 连接服务， mqssl 可以与指定端口建立安全套接字层 (Secure Socket Layer, SSL) 连接。
- 使用 httpjms 连接服务， http 可以与指定 URL 处的 Message Queue 隧道 Servlet 建立超文本传输协议 (Hypertext Transport Protocol, HTTP) 连接。
- 使用 httpsjms 连接服务， https 可以与指定 URL 处的 Message Queue 隧道 Servlet 建立安全超文本传输协议 (Secure Hypertext Transport Protocol, HTTPS) 连接。

第 289 页上的表 16-2 概要介绍了这些寻址方案。

每个消息服务器地址的通用格式为：

scheme://*address*

其中 *scheme* 是上面列出的寻址方案之一，而 *address* 表示服务器地址本身。用于指定地址的准确语法因寻址方案而异，如表 16-2 中的最后一列所示。第 290 页上的表 16-3 提供了各种地址格式的示例。

在多代理群集环境中，地址列表可能包含多个服务器地址。如果第一次连接尝试失败， Message Queue 客户端运行时将尝试连接到列表中的另一个地址，依此类推，直到尝试完列表中的所有地址为止。其他两个连接工厂属性控制上述操作的执行方式：

- imqAddressListBehavior 指定了尝试连接指定地址的顺序。如果将此属性设置为字符串 PRIORITY，将按地址在地址列表中的显示顺序尝试连接。如果属性值为 RANDOM，将按随机顺序尝试连接地址；这种方式非常有用，例如，当许多 Message Queue 客户端共享同一个连接工厂对象时，这种方式有助于防止所有客户端尝试连接到同一个服务器地址。
- imqAddressListIterations 指定了循环访问列表的次数，超过该次数后，将放弃尝试并报告故障。值为 -1 表示不限制重复次数：客户端运行时将一直尝试下去，直到成功建立连接或到达结束时间为止 （以最先出现的情况为准）。

自动重新连接

通过将连接工厂的 imqReconnectEnabled 属性设置为 true，可以使客户端在连接失败时自动重新连接到代理。 imqReconnectAttempts 属性控制尝试重新连接到给定服务器地址的次数； imqReconnectInterval 指定了两次尝试之间等待的时间间隔（以毫秒为单位）。

在消息服务器地址列表 (imqAddressList) 指定了多个地址的代理群集中，不但可以在原始代理上恢复失败的连接，而且还可以在群集中的其他代理上恢复失败的连接。如果重新连接到原始代理失败，则客户端运行时将尝试连接列表中的其他地址。 imqAddressListBehavior 和 imqAddressListIterations 属性控制尝试连接地址的顺序和循环访问列表的次数，如前面一节所述。将以 imqReconnectInterval 毫秒的时间间隔反复尝试连接每个地址，直到达到 imqReconnectAttempts 指定的最大尝试次数为止。

自动重新连接支持消息使用的所有客户端确认模式。重新建立连接之后，代理将重新传送以前传送过的所有未确认消息，并使用重新传送标志对其进行标记。应用程序代码可以使用此标志确定消息是否已使用但尚未得到确认。（但是，对于非长期订户，消息服务器在关闭连接后不会保留消息。因此，在关闭连接时为这些订户生成的所有消息都将丢失，而不能在重新连接后进行传送。）在自动重新连接过程中将禁止生成消息；消息生成方无法向服务器发送消息，直到重新建立连接为止。

自动重新连接提供连接故障转移，但不提供数据故障转移：客户端重新连接到其他代理实例时，出现故障或断开连接的代理所保存的持久性消息和其他状态信息会丢失。尝试重新建立连接时， Message Queue 确实会维护客户端运行时提供的对象（如会话、消息使用方和消息生成方）。当连接失败时，系统也会维护临时目标一段时间，因为客户端可能会重新连接并再次访问它们；在客户端重新连接并使用这些目标之后，代理将会删除它们。如果重新连接时无法在代理上完全恢复客户端状态（例如，在使用事务会话时，客户端状态只在连接期间存在），将不会进行自动重新连接，而是调用该连接的异常处理程序。然后由应用程序代码捕获异常、重新连接并恢复状态。

定期测试 (ping) 连接

可以将 Message Queue 客户端运行时配置成定期测试或 "ping" 连接，从而可以尽早检测出连接故障，以免在尝试传输消息时失败。对于只使用消息而不生成消息的客户端应用程序而言，这种测试尤为重要，因为如果不进行测试，这些应用程序将无法检测到连接失败。只是偶尔生成消息的客户端也可以利用此功能。

连接工厂属性 imqPingInterval 指定了 ping 连接的频率 （以秒为单位）。默认情况下将此时间间隔设置为 30 秒；如果值为 -1，则表示禁用 ping 操作。

对失败的 ping 操作的响应因操作系统平台而异。在某些操作系统上，会立即向客户端应用程序的异常侦听器抛出异常。（如果客户端没有异常侦听器，则当它下次尝试使用连接时将会失败。）其他系统可能会继续尝试与代理建立连接，并缓冲后续的 ping 操作，直到某次尝试成功或缓冲区溢出为止。

客户端身份验证

第 291 页上的表 16-4 中列出的连接工厂属性支持长期订户的客户端验证和客户端标识符设置。

客户端验证

必须对所有的代理连接尝试进行验证，方法是将用户名和密码与消息服务器维护的用户系统信息库进行对照。连接工厂属性 `imqDefaultUsername` 和 `imqDefaultPassword` 指定了创建连接时使用的默认用户名和密码 （如果客户端未明确提供它们）。

对于不希望在应用程序开发和测试期间填充用户系统信息库的开发者，为方便起见， Message Queue 提供了用户名和密码均为 `guest` 的 guest 用户帐户。这也是 `imqDefaultUsername` 和 `imqDefaultPassword` 属性的默认值，这样，如果未明确指定这些属性，客户端可以一直使用临时帐户获取连接。在生产环境中，只有在用户系统信息库中明确注册的用户，才应该访问代理连接。

客户端标识符

根据 **Java 消息服务规范**的要求，当消息服务器必须为客户端维护持久性状态时，连接必须提供一个唯一的**客户端标识符**。 Message Queue 使用这些客户端标识符来跟踪主题目标的长期订户。如果长期订户变为非活动状态，则代理会保留与该主题有关的所有传入消息，并在订户再次处于活动状态时传送这些消息。代理通过客户端标识符来标识订户。

客户端应用程序可以使用连接对象的 `setClientID` 方法以编程方式设置其自身的客户端标识符，这使得很难协调客户端标识符，从而不能确保每个标识符都是唯一的。通常，为客户端创建连接时，最好让 Message Queue 自动指定一个唯一的标识符。具体方法是，将连接工厂的 `imqConfiguredClientID` 属性设置为具有以下格式的值

`${u}`*factoryID*

字符 `${u}` 必须是该属性值的前四个字符。（大括号之间如果是 u 以外的任何字符，都会在创建连接时抛出异常；在任何其他位置，这些字符都没有特殊含义，将被视为纯文本。） *factoryID* 的值是唯一与此连接工厂对象关联的字符串。

为特定客户端创建连接时， Message Queue 通过将字符 ${u} 替换为 u:*userName* 来构建客户端标识符，其中 *userName* 是通过连接验证的用户名。这可以确保给定连接工厂创建的每个连接都有其自身的唯一客户端标识符，即使它们在其他所有方面都完全相同。例如，如果用户名为 Calvin，为连接工厂的 imqConfiguredClientID 属性指定的字符串为 ${u}Hobbes，则指定的客户端标识符将为 u:CalvinHobbes。

注 如果两个客户端都尝试使用默认用户名 guest 获取连接，则此方案将不起作用，因为这两个客户端的客户端标识符具有相同的组成部分 ${u}。在这种情况下，只有请求连接的第一个客户端才能获取连接；第二个客户端的连接尝试将会失败，因为 Message Queue 不能创建两个具有相同客户端标识符的连接。

即使通过 imqConfiguredClientID 指定客户端标识符，客户端应用程序也可以使用连接方法 setClientID 来覆盖此设置。为了防止出现这种情况，可以将连接工厂的 imqDisableSetClientID 属性设置为 true。请注意，对于使用长期订户的应用程序，**必须**使用以下两种方法之一设置客户端标识符：使用 imqConfiguredClientID 以管理方式设置，或者使用 setClientID 以编程方式设置。

可靠性和流控制

由于客户端发送和接收的 “有效负荷”消息和 Message Queue 自身使用的控制消息 （如代理确认消息）通过同一个客户端 / 代理连接进行传递，因此有效负荷流量过多将会干扰控制消息的传送。为了帮助缓解这个问题，可以使用第 291 页上的表 16-5 中列出的连接工厂属性来管理两种消息的相对流量。这些属性分为以下四个类别：

- **确认超时**指定等待代理确认的最长时间 (imqAckTimeout)，超出此时间后将会抛出异常。
- **连接流计量**通过将有效负荷消息拆分成具有指定大小 (imqConnectionFlowCount) 的若干批消息来限制有效负荷消息的传输，从而确保可以定期传送任何堆积的控制消息。
- **连接流控制**限制有效负荷消息的数量 (imqConnectionFlowLimit)，这些消息可以在连接上保持待处理状态，以等待使用。达到限制之后，将会暂停向连接传送有效负荷消息，直到等待使用的消息数低于该限制为止。此功能的使用受布尔标志 (imqConnectionFlowLimitEnabled) 控制。

- **使用方流控制**限制有效负荷消息的数量 (imqConsumerFlowLimit)，这些消息可以针对任何单个使用方保持待处理状态，以等待使用。（也可以将此限制指定为特定队列目标的属性 consumerFlowLimit。）达到该限制时，将会暂停向使用方传送有效负荷消息，直到等待使用的消息数 （以 imqConsumerFlowLimit 的百分比表示）低于 imqConsumerFlowThreshold 属性指定的限制为止。这有助于防止同一连接上的任一使用方抢占其他使用方的流量，从而改善多个使用方之间的负载平衡。

使用上述流控制技术中的任何一种都需要在可靠性和吞吐量之间进行权衡；有关详细论述，请参见第 209 页上的 “客户端运行时消息流调整”。

队列浏览器和服务器会话

第 292 页上的表 16-6 列出了影响客户端队列浏览和服务器会话的连接工厂属性。imqQueueBrowserMaxMessagesPerRetrieve 属性指定了浏览队列目标的内容时一次可以检索的最大消息数； imqQueueBrowserRetrieveTimeout 指定了等待检索消息的最长时间。布尔属性 imqLoadMaxToServerSession 管理应用服务器会话中连接使用方的行为：如果此属性的值是 true，客户端将向服务器会话中装入多个消息 （消息数量不超过最大消息数）；如果此属性的值是 false，则客户端一次只装入一条消息。

标准消息属性

Java 消息服务规范定义了某些标准消息属性， JMS 提供者 （如 Message Queue）可以选择是否支持这些属性。根据惯例，所有这些标准属性的名称均以字母 JMSX 开头。第 293 页上的表 16-7 中列出的连接工厂属性控制 Message Queue 客户端运行时是否设置这些标准属性中的某些属性。对于生成的消息而言，这些属性包括：

JMSXUserID	发送消息的用户的标识
JMSXAppID	发送消息的应用程序的标识
JMSXProducerTXID	生成消息时所在的事务的标识

对于使用的消息而言，这些属性包括：

JMSXConsumerTXID	使用消息时所在的事务的标识
JMSXRcvTimestamp	消息传送到使用方的时间

消息头覆盖

对于某些 JMS 消息头字段，可以使用第 293 页上的表 16-8中列出的连接工厂属性来覆盖客户端设置的值。指定的设置将用于从该连接工厂获取的连接所生成的全部消息。可以采用这种方式覆盖的头字段包括：

- `JMSDeliveryMode` 传送模式（持久性或非持久性）
- `JMSExpiration` 到期时间
- `JMSPriority` 优先级

上述每个字段都有两个属性：一个是布尔属性，用于控制是否可以覆盖字段，另一个用于指定字段的值。例如，用于设置优先级的属性是 `imqOverrideJMSPriority` 和 `imqJMSPriority`。此外还有一个属性 `imqOverrideJMSHeadersToTemporaryDestinations`，该属性控制覆盖值是否适用于临时目标。

注 由于覆盖消息头可能会影响特定应用程序的需求，因此只有在咨询了应用程序的设计者或用户之后才应使用这些属性。

目标属性

标识物理队列或主题目标的**目标**受管理对象只有两个属性，如第 294 页上的表 16-9中所示。一个重要属性是 `imqDestinationName`，该属性指定了此受管理对象表示的物理目标的名称；此名称是使用 `imqcmd create dst` 命令（用于创建物理目标）的 `-n` 选项指定的。此外，还有一个可选的描述性字符串 `imqDestinationDescription`，使用该字符串有助于标识目标对象，将其与可能已经创建的其他对象区分开来。

使用对象管理器实用程序

使用 Message Queue 对象管理器实用程序 (`imqobjmgr`) 可以创建和管理受管理对象。`imqobjmgr` 命令提供了以下子命令，用于对受管理对象执行各种操作：

`add` 将受管理对象添加到对象存储中

`delete` 从对象存储中删除受管理对象

`list` 列出对象存储中现有的受管理对象

`query` 显示与受管理对象有关的信息

`update` 修改受管理对象的属性

有关 `imqobjmgr` 命令的语法、子命令和选项的参考信息，请参见第 253 页上的 “对象管理器实用程序”。

大部分对象管理器操作都要求将以下信息指定为 `imqobjmgr` 命令的选项：

- 受管理对象的 **JNDI 查找名称** （`-l` 选项）

  这是客户端应用程序使用 Java 命名和目录接口在对象存储中查找受管理对象时可以依据的逻辑名称。

- **JNDI 对象存储的属性** （`-j` 选项）

  有关可能属性及其值的信息，请参见第 147 页上的 “对象存储”。

- **受管理对象的类型** （`-t` 选项）

  可能的类型包括：

  `q` 队列目标

  `t` 主题目标

  `cf` 连接工厂

  `qf` 队列连接工厂

  `tf` 主题连接工厂

  `xcf` 分布式事务的连接工厂

  `xqf` 分布式事务的队列连接工厂

  `xtf` 分布式事务的主题连接工厂

  `e` SOAP 端点

- **受管理对象的属性** （`-o` 选项）

  有关可能属性及其值的信息，请参见第 150 页上的 “受管理对象的属性”。

添加受管理对象

`imqobjmgr` 命令的 `add` 子命令将连接工厂和主题或队列目标的受管理对象添加到对象存储中。存储在 LDAP 对象存储中的受管理对象的查找名称必须以前缀 `cn=` 开头；文件系统对象存储中的查找名称不必以任何特定前缀开头，但是不能包含正斜杠字符 (/)。

注 对象管理器仅列出并显示 Message Queue 受管理对象。如果对象存储中应该包含一个非 Message Queue 对象，并且该对象与要添加的受管理对象具有相同的查找名称，则当您尝试执行添加操作时，将会收到一条错误消息。

添加连接工厂

要使客户端应用程序能够创建代理连接，请为要创建的连接类型添加连接工厂受管理对象：队列连接工厂或主题连接工厂。代码示例 8-1 显示了一条用于将队列连接工厂（受管理对象类型 `qf`）添加到 LDAP 对象存储的命令。该对象具有查找名称 `cn=myQCF`，并使用 `jms` 连接服务连接到在端口号为 `7272` 的主机 `myHost` 上运行的代理。连接服务

代码示例 8-1 添加连接工厂

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

添加目标

创建表示目标的受管理对象时，最好先创建物理目标，然后将受管理对象添加到对象存储中。可以使用命令实用程序 (`imqcmd`) 创建物理目标，如第 111 页上的 “创建物理目标”中所述。

代码示例 8-2 中显示的命令将受管理对象添加到表示某个主题目标的 LDAP 对象存储，该主题目标的查找名称为 `myTopic`，物理目标名称为 `physTopic`。用于添加队列目标的命令会很相似，但是受管理对象的类型（`-t` 选项）应该是 `q`（对应于 "queue destination"）而不是 `t`（对应于 "topic destination"）。

代码示例 8-2 将目标添加到 LDAP 对象存储

```
imqobjmgr add
   -l "cn=myTopic"
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
```

代码示例 8-2 将目标添加到 LDAP 对象存储 （续）

```
    -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
    -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
    -j "java.naming.security.credentials=doh"
    -j "java.naming.security.authentication=simple"
     -t t
    -o "imqDestinationName=physTopic"
```

代码示例 8-3 显示的是同一个命令，但是受管理对象存储在 Solaris 文件系统而不是 LDAP 服务器中。

代码示例 8-3 将目标添加到文件系统对象存储

```
imqobjmgr add
   -l "cn=myTopic"
   -j "java.naming.factory.initial=
          com.sun.jndi.fscontext.RefFSContextFactory"
   -j "java.naming.provider.url=file:///home/foo/imq_admin_objects"
    -t t
   -o "imqDestinationName=physTopic"
```

删除受管理对象

要从对象存储中删除受管理对象，可以使用 `imqobjmgr` 命令的 `delete` 子命令，并指定要删除对象的查找名称、类型和位置。代码示例 8-4 中显示的命令用于删除在上述代码示例 8-2 中添加的对象。

代码示例 8-4 删除受管理对象

```
imqobjmgr delete
   -l "cn=myTopic"
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
    -t t
```

列出受管理对象

使用对象管理器的 `list` 子命令，可以获取对象存储中所有受管理对象或特定类型受管理对象的列表。代码示例 8-5 说明了如何列出 LDAP 服务器上的所有受管理对象。

代码示例 8-5 列出所有受管理对象

```
imqobjmgr list
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
```

代码示例 8-6 列出了所有队列目标 （类型为 `q`）。

代码示例 8-6 列出特定类型的受管理对象

```
imqobjmgr list
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
   -t q
```

查看受管理对象的信息

`query` 子命令显示与指定的受管理对象有关的信息，该对象使用查找名称及包含该对象的对象存储的属性来标识。代码示例 8-7 显示了与查找名称 `cn=myTopic` 的对象有关的信息。

代码示例 8-7 查看受管理对象的信息

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

代码示例 8-7 查看受管理对象的信息 （续）

```
    -j "java.naming.security.credentials=doh"
    -j "java.naming.security.authentication=simple"
```

修改受管理对象的属性

要修改受管理对象的属性，可以使用 imqobjmgr update 子命令。您应该提供该对象的查找名称和位置，并使用 -o 选项指定新的属性值。

代码示例 8-8 更改在第 158 页上的代码示例 8-1 中添加到对象存储的队列连接工厂的 imqReconnectAttempts 属性值。

代码示例 8-8 修改受管理对象的属性

```
imqobjmgr update
    -l "cn=myQCF"
   -j "java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory"
   -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
   -j "java.naming.security.principal=uid=homerSimpson,ou=People,o=imq"
   -j "java.naming.security.credentials=doh"
   -j "java.naming.security.authentication=simple"
   -t qf
   -o "imqReconnectAttempts=3"
```

使用命令文件

使用 imqobjmgr 命令的 -i 选项可以指定命令文件的名称，命令文件使用 Java 属性文件语法来表示全部或部分子命令子句。此功能在指定对象存储属性时尤为有用，通常，指定对象存储属性时需要执行大量的键入操作，并且在多次调用 imqobjmgr 时指定属性的操作可能相同。使用命令文件还可以避免超出命令行所允许的最大字符数。

代码示例 8-9 说明了对象管理器命令文件的一般语法。请注意， version 属性不是命令行选项：它指的是命令文件自身的版本 （而不是 Message Queue 产品的版本），因此必须将其值设置为 2.0。

代码示例 8-9 对象管理器命令文件语法

```
version=2.0
cmdtype=[ add | delete | list | query | update ]
```

代码示例 8-9 对象管理器命令文件语法 （续）

```
obj.lookupName=lookup name
objstore.attrs.objStoreAttrName1=value1
objstore.attrs.objStoreAttrName2=value2
   . . .
objstore.attrs.objStoreAttrNameN=valueN
obj.type=[ q | t | cf | qf | tf | xcf | xqf | xtf | e ]
obj.attrs.objAttrName1=value1
obj.attrs.objAttrName2=value2
   . . .
obj.attrs.objAttrNameN=valueN
```

例如，请考虑前面第 158 页上的代码示例 8-1 中显示的对象管理器命令，该命令将队列连接工厂添加到 LDAP 对象存储。此命令可以封装在命令文件中，如代码示例 8-10 所示。如果将此命令文件命名为 `MyCmdFile`，则可以使用以下命令行来执行该命令：

```
imqobjmgr -i MyCmdFile
```

代码示例 8-10 示例命令文件

```
version=2.0
cmdtype=add
obj.lookupName=cn=myQCF
objstore.attrs.java.naming.factory.initial=com.sun.jndi.ldap.LdapCtxFactory
objstore.attrs.java.naming.provider.url=ldap://mydomain.com:389/o=imq
objstore.attrs.java.naming.security.principal=\
                                        uid=homerSimpson,ou=People,o=imq
objstore.attrs.java.naming.security.credentials=doh
objstore.attrs.java.naming.security.authentication=simple
obj.type=qf
obj.attrs.imqAddressList=mq://myHost:7272/jms
```

还可以仅使用命令文件指定 `imqobjmgr` 子命令子句部分，而在命令行上明确提供其余部分。例如，代码示例 8-12 中显示的命令文件仅指定了 LDAP 对象存储的属性值。

代码示例 8-11 部分命令文件

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

代码示例 8-11 部分命令文件 （续）

```
                                                uid=homerSimpson,ou=People,o=imq
objstore.attrs.java.naming.security.credentials=doh
objstore.attrs.java.naming.security.authentication=simple
```

然后，可以使用此命令文件在 imqobjmgr 命令中指定对象存储，同时明确提供其余选项，如代码示例 8-12 所示。

代码示例 8-12 使用部分命令文件

```
imqobjmgr add
    -l "cn=myQCF"
   -i MyCmdFile
   -t qf
   -o "imqAddressList=mq://myHost:7272/jms"
```

您可以根据使用的平台，从以下相应位置找到命令文件的其他示例：

Solaris： /usr/demo/imq/imqobjmgr
Linux： /opt/sun/mq/examples/imqobjmgr
Windows： IMQ_HOME/demo/imqobjmgr

第 9 章

使用代理群集

Message Queue Enterprise Edition 支持使用**代理群集**。代理群集是一组协同工作为客户端提供消息传送服务的代理。通过使用群集，消息服务器可以将客户端连接分布在多个代理上，从而根据消息流量相应地调整自己的操作。有关群集及其工作原理的概述，请参见 Message Queue 技术概述。

本章介绍如何管理代理群集，如何将代理连接到群集以及如何配置代理群集。本章包含以下各节：

群集配置属性

可通过为群集中的每个成员代理指定**群集配置属性**来定义群集。可以为群集中的每个代理单独设置这些属性，但是，通常较为方便的一种做法是将这些属性收集到所有代理均引用的一个中心**群集配置文件**中。这样可防止设置出现不一致的情况，并确保群集中的所有代理都共享相同、一致的配置信息。

第 277 页上的表 14-9 中详细介绍了群集配置属性。其中包括以下内容：

- `imq.cluster.brokerlist` 给出了属于群集的所有代理的主机名和端口号。
- `imq.cluster.masterbroker` 指定跟踪状态变化的主代理 （如果有）。
- `imq.cluster.url` 指定群集配置文件 （如果有）的位置。
- `imq.cluster.hostname` 给出 `cluster` 连接服务 （用于群集中代理之间的内部通信）的主机名或 IP 地址。如果有多个主机可用，该设置会很有用：例如，如果一台计算机中安装了多个网络接口卡。

- imq.cluster.port 给出 cluster 连接服务的端口号。
- imq.cluster.transport 指定 cluster 连接服务所使用的传输协议，如 tcp 或 ssl。

可以为每个代理单独设置 hostname 和 port 属性，但群集中所有代理的 brokerlist、masterbroker、url 和 transport 值必须相同。

以下部分介绍了如何设置代理的群集配置属性。可以对群集中的每个代理单独进行设置，也可以使用群集配置文件来集中进行设置。

分别为各个代理设置群集属性

可以在代理的实例配置文件 （或在启动代理时的命令行）中设置代理的群集配置属性。例如，要创建由 host1 端口 9876 上的代理、host2 端口 5000 上的代理以及 ctrlhost 默认端口 (7676) 上的代理组成的群集，应在这三个代理的实例配置文件中包含以下属性：

```
imq.cluster.brokerlist=host1:9876,host2:5000,ctrlhost
```

请注意，如果采用此方法，则当群集配置需要更改时，您必须更新群集中每个代理的实例配置文件。

使用群集配置文件

为保持一致性和易维护性，建议将所有共享群集配置属性收集到一个群集配置文件中，而不是分别为每个代理设置这些属性。采用此方法时，每个代理的实例配置文件必须将 imq.cluster.url 属性设置为指向该群集配置文件的位置：例如，

```
imq.cluster.url=file:/home/cluster.properties
```

然后群集配置文件定义群集中所有代理的共享配置属性，例如要连接的代理的列表 (imq.cluster.brokerlist)、用于 cluster 连接服务的传输协议 (imq.cluster.transport) 以及 （可选）主代理的地址 (imq.cluster.masterbroker)。下面的代码定义了与前面的示例相同的群集，其中在 ctrlhost 上运行的代理充当主代理：

```
imq.cluster.brokerlist=host1:9876,host2:5000,ctrlhost
imq.cluster.masterbroker=ctrlhost
```

管理群集

本节介绍如何将一组代理连接成一个群集，如何在现有群集中添加新代理，以及如何从群集中删除代理。

连接代理

将代理连接成群集一般有两种方法：一种方法是通过命令行 （使用 -cluster 选项）来实现，一种方法是通过在群集配置文件中设置 imq.cluster.brokerlist 属性来实现。无论采用哪种方法，启动的每个代理都每隔五秒钟就尝试与其他代理连接；在主代理 （如果已配置）启动后，连接就会成功。如果群集中的某个代理在主代理之前启动，它将保持暂停状态，并拒绝客户端连接，直到主代理启动为止；之后，暂停的代理将自动进入正常运行状态。

要通过命令行配置代理群集，请在启动群集中的每个代理时使用 imqbrokerd 命令的 -cluster 选项指定群集中代理的完整列表。例如，以下命令启动一个新代理，并将它连接到在 host1 默认端口 (7676) 上运行的代理、在 host2 端口 5000 上运行的代理以及在默认主机 (localhost) 端口 9876 上运行的代理：

```
imqbrokerd -cluster host1,host2:5000,:9876
```

有一种更适用于生产系统的替代方法，即创建一个群集配置文件，它使用 imq.cluster.brokerlist 属性来指定要连接的代理的列表。然后群集中的每个代理必须将其自身的 imq.cluster.url 属性设置为指向该群集配置文件。

Linux 前提条件：设置 IP 地址

在 Linux 系统中，要将代理连接为群集，有一个特殊的前提条件。某些 Linux 安装程序自动将 localhost 条目设置为网络回送 IP 地址 (127.0.0.1)。您必须设置系统的 IP 地址，以便为群集中的所有代理设置正确的地址。

在设置群集的过程中，应针对加入群集的所有 Linux 系统检查 /etc/hosts 文件。如果系统使用静态 IP 地址，请编辑 /etc/hosts 文件以指定 localhost 的正确地址。如果地址已向域名服务 (Domain Name Service, DNS) 注册，请编辑 /etc/nsswitch.conf 文件以更改条目的顺序，使系统在查阅本地 hosts 文件前先执行 DNS 查找。 /etc/nsswitch.conf 文件中的行应显示为如下顺序：

```
hosts:dns files
```

代理之间的安全连接

如果要在群集中的代理之间以安全加密的方式传送消息，请配置 cluster 连接服务以使用基于 SSL 的传输协议。请按照第 135 页上的 “使用基于 SSL 的服务”中的说明，为群集中的各个代理设置基于 SSL 的连接服务。然后，将每个代理的 imq.cluster.transport 属性设置为 ssl。可以在群集配置文件中进行此设置，也可以为每个代理单独进行此设置。

在群集中添加代理

在群集中添加新代理的步骤取决于该群集是否使用群集配置文件。

➤ 向使用群集配置文件的群集添加新代理

1. 将新代理添加到群集配置文件中的 imq.cluster.brokerlist 属性。
2. 对群集中的各个代理执行以下命令：

   ```
   imqcmd reload cls
   ```

   该命令强制每个代理重新加载群集配置，从而确保群集中代理的所有持久性信息都是最新的。
3. **（可选）**在代理的 config.properties 文件中将 imq.cluster.url 属性的值设置为指向群集配置文件。
4. 启动新代理。

   如果未执行步骤 3，请使用 imqbrokerd 命令行中的 -D 选项设置 imq.cluster.url 的值。

➤ 向未使用群集配置文件的群集添加新代理

通过编辑 config.properties 文件或者使用 imqbrokerd 命令行中的 -D 选项设置以下属性的值：

- imq.cluster.brokerlist
- imq.cluster.masterbroker （如有必要）
- imq.cluster.transport （如果使用安全的 cluster 连接服务）

从群集中删除代理

从群集中删除代理的方法取决于群集最初是通过命令行创建的，还是通过中央群集配置文件来创建的。

使用命令行删除代理

如果群集是通过在命令行使用 imqbrokerd 命令由代理连接而成的，则您必须停止每个代理，然后通过在命令行中指定一组新的群集成员来重新启动它们。过程如下所述：

➤ 使用命令行从群集中删除代理

1. 使用 imqcmd 命令停止群集中的每个代理。
2. 重新启动仍然保留在群集中的代理 （使用 imqbrokerd 命令的 -cluster 选项，并仅仅指定那些要保留的代理）。

   例如，假设最初创建的群集包括代理 *A*、 *B*、 *C*，而且这三个代理是通过以下命令启动的：

   ```
   imqbrokerd -cluster A,B,C
   ```

   要从群集中删除代理 *A*，请使用以下命令重新启动代理 *B* 和 *C*：

   ```
   imqbrokerd -cluster B,C
   ```

使用群集配置文件删除代理

如果群集最初是通过在中央群集配置文件中使用 imq.cluster.brokerlist 属性指定其成员代理来创建的，则不必停止这些代理即可删除其中的某个代理。只需编辑配置文件来排除要删除的代理，强制群集中的其他代理成员重新加载群集配置，并重新配置被排除的代理，使它不再指向原来的群集配置文件。过程如下：

➤ 使用群集配置文件从群集中删除代理

1. 编辑群集配置文件，从为 imq.cluster.brokerlist 属性指定的列表中删除要排除的代理。
2. 对保留在群集中的每个代理执行以下命令：

   ```
   imqcmd reload cls
   ```

   该命令强制代理重新加载群集配置。
3. 停止要从群集中删除的代理。
4. 编辑该代理的 config.properties 文件，删除 imq.cluster.url 属性或为该属性指定其他值。

主代理

可以选择为群集指定一个**主代理**。主代理维护**配置更改记录**以跟踪群集的持久性状态所发生的任何变化。主代理是使用 imq.cluster.masterbroker 配置属性来标识的。该属性在群集配置文件中或在单个代理的实例配置文件中。

配置更改记录包含与群集关联的持久性实体 （例如长期订阅和管理员创建的物理目标）的相关更改信息。群集中的所有代理在启动时都会咨询主代理，以更新与这些持久性实体有关的信息。如果主代理出现故障，就无法实现上述同步；有关更多信息，请参见第 170 页上的 “当主代理不可用时”。

管理配置更改记录

因为配置更改记录中包含的信息很重要，所以要定期备份配置更改记录，以便在发生故障的情况下能够恢复此记录。这一点非常重要。虽然从备份恢复的记录不包含群集持久性状态自备份以来发生的任何更改，但经常进行备份有助于将丢失的信息降到最少。备份和恢复操作还有利于压缩和优化配置更改记录中保留的更改历史，因为该记录会随时间的推移迅速膨胀。

➤ 备份配置更改记录

使用 imqbrokerd 命令的 -backup 选项，并指定备份文件的名称。例如：

```
imqbrokerd -backup mybackuplog
```

➤ 恢复配置更改记录

1. 关闭群集中的所有代理。
2. 使用以下命令从备份文件中恢复主代理的配置更改记录：

   ```
   imqbrokerd -restore mybackuplog
   ```

3. 如果为主代理分配了新的名称或端口号，请在群集配置文件中相应地更新 imq.cluster.brokerlist 和 imq.cluster.masterbroker 属性。
4. 重新启动群集中的所有代理。

当主代理不可用时

因为群集中的所有代理都需要利用主代理来执行持久性操作，因此，当主代理不可用时，群集中任一代理的以下 imqcmd 子命令都将返回一条错误消息：

- create dst

- `destroy dst`
- `update dst`
- `destroy dur`

自动创建的物理目标和临时目标均不受影响。

在没有主代理时，任何尝试创建长期订户或取消长期订阅的客户端应用程序都会收到一条错误消息。但是，客户端可以成功地指定现有长期订阅，并与其进行交互。

第 10 章

监视消息服务器

本章介绍可用来监视消息服务器的工具，以及如何获得度量数据。本章包含以下各节：

有关特定度量的参考信息可从第 18 章 “度量参考”获得。

监视工具简介

Message Queue 信息有三种监视接口：日志文件、交互式命令以及可获取度量的客户端 API。每一种接口都有其各自的优缺点，如下所述：

- 日志文件提供长期的度量数据记录，但不易于解析。
- 使用命令可以根据需要快速获取采样信息，但不能查看历史信息或通过编程方式操作数据。
- 使用客户端 API 可以提取和处理信息、操作数据、提供图形或发送警报。但是，要使用它，您必须编写自定义应用程序来捕获和分析数据。

表 10-1 比较了各种不同的工具。

表 10-1 度量监视工具的优势和局限性

度量监视工具	优势	局限性
`imqcmd metrics`	远程监视 便于抽样检查 在命令选项中设置报告的时间间隔；可以随时更改此时间间隔 易于选择感兴趣的特定数据 数据以简单的表格形式提供	无法通过单条命令获得所有数据 难以通过编程方式分析数据 不创建历史记录 难以看到历史趋势
日志文件	定期采样 创建历史记录	需要配置代理属性；必须关闭再重新启动代理才能生效 仅限本地监视 数据格式非常难以读取或解析；没有解析工具 报告时间间隔不能随时更改；所有度量数据均是如此 在数据的选择方面不提供灵活性 仅限代理度量；目标和连接服务度量不包括在内 如果时间间隔设置得过短，性能可能会受影响
客户端 API	远程监视 易于选择感兴趣的特定数据 数据可以通过编程方式分析，并能够以任何格式呈现	需要配置代理属性；必须关闭再重新启动代理才能生效 需要编写您自己的度量监视客户端 报告时间间隔不能随时更改；所有度量数据均是如此

除了表中给出的不同外，每种工具收集到的具体信息（代理生成的度量信息的一部分）也稍有不同。有关每种监视工具收集的度量数据的信息，请参见第 303 页上的第 18 章 “度量参考”。

配置和使用代理日志记录

Message Queue 记录程序获取代理代码、调试器和度量生成器等生成的信息，并将这些信息写入多个输出通道：标准输出 （控制台）、日志文件以及 Solaris™ 操作系统上的 syslog 守护进程。

您可以指定记录程序收集的信息类型以及写入每个输出通道的信息类型。特别是，您可以指定将度量信息写出到日志文件。

本节介绍了代理的默认日志记录配置，并说明了如何将日志信息重定向到替代的输出通道，如何更改日志文件转移条件，以及如何将度量数据发送到日志文件。

默认日志记录配置

代理自动配置为将日志输出保存到一组循环的日志文件中。这些日志文件位于由关联代理的实例名称标识的目录中 （请参见附录 A “Message Queue 数据在特定平台上的位置”）：

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

日志文件是纯文本文件。它们按如下方式命名 （从最早生成到最近生成）：

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

默认情况下，日志文件每星期转移一次；系统维护九个备份文件。

- 要更改保存日志文件的目录，请将 imq.log.file.dirpath 属性设置为所需的路径。
- 要将日志文件的根名称从 log 改为其他名称，请设置 imq.log.file.filename 属性。

代理支持三种日志级别：ERROR、 WARNING 和 INFO。表 10-2 说明了每一种级别。

表 10-2 日志记录级别

级别	描述
ERROR	指出可能导致系统故障的问题的消息。
WARNING	需要注意但不会导致系统故障的警报。
INFO	度量及其他信息性消息的报告。

设置日志记录级别后，将收集所有高于和等于该级别的消息。默认的日志级别是 INFO，因此，默认情况下会记录所有 ERROR、 WARNING 和 INFO 消息。

日志消息格式

记录的消息由时间戳、消息代码和消息本身构成。信息量取决于所设置的日志级别。下面是一个 INFO 消息的示例。

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

要更改时间戳时区，请参见有关 imq.log.timezone 属性的信息，第 274 页上的表 14-8 中介绍了该属性。

更改记录程序配置

第 274 页上的表 14-8 中介绍了与日志相关的属性。

➤ **更改代理的记录程序配置**

1. 设置日志级别。
2. 为一个或多个日志记录类别设置输出通道 （文件和 / 或控制台）。
3. 如果要将输出记录到文件中，请为该文件配置转移条件。

可以通过设置记录程序属性来完成上述步骤。可以使用以下两种方法之一来设置记录程序属性：

- 启动代理之前，在该代理的 config.properties 文件中更改或添加记录程序属性。
- 在启动代理的 imqbrokerd 命令中指定记录程序命令行选项。也可以使用代理选项 -D 来更改记录程序属性 （或**任何**代理属性）。

通过命令行传递的选项将覆盖代理实例配置文件中指定的属性。表 10-3 中列出了影响日志记录的 imqbrokerd 选项。

表 10-3 imqbrokerd 记录程序选项及对应的属性

imqbrokerd 选项	描述
-metrics *interval*	指定度量信息写入记录程序的时间间隔 （以秒为单位）。
-loglevel *level*	将日志级别设置为以下三项之一：ERROR、 WARNING 和 INFO。
-silent	禁止向控制台记录日志信息。
-tty	将所有消息发送到控制台。默认情况下，只显示 WARNING 和 ERROR 级别的消息。

下面几节介绍了如何更改默认配置以执行下列操作：

- 更改输出通道 （日志消息的目标）
- 更改转移条件

更改输出通道

默认情况下，错误和警告消息除了记录到日志文件中以外，还会显示在终端上。（在 Solaris 中，错误消息还会写入到系统的 syslog 守护进程中。）

可以用以下方式更改日志消息的输出通道：

- 要在屏幕上显示**所有**日志类别 （对于给定的级别）输出，请在 imqbrokerd 命令中使用 -tty 选项。
- 要禁止在屏幕上显示日志输出，请在 imqbrokerd 命令中使用 -silent 选项。
- 使用 imq.log.file.output 属性指定将哪些类别的日志记录信息写入到日志文件中。例如，

  ```
  imq.log.file.output=ERROR
  ```

- 使用 imq.log.console.output 属性指定将哪些类别的日志记录信息写入到控制台。例如，

  ```
  imq.log.console.output=INFO
  ```

- 在 Solaris 中，使用 imq.log.syslog.output 属性指定将哪些类别的日志记录信息写入到 Solaris syslog 中。例如，

  ```
  imq.log.syslog.output=NONE
  ```

注 在更改记录程序输出通道前，必须确保日志记录级别已设置为支持映射到输出通道的信息。例如，如果将日志级别设置为 ERROR，然后将 imq.log.console.output 属性设置为 WARNING，那么将不会记录任何消息，因为没有启用 WARNING 消息的日志记录。

更改日志文件转移条件

转移日志文件有两个条件：时间和大小。默认情况下使用时间标准，每七天转移一次文件。

- 要更改时间间隔，需要更改 imq.log.file.rolloversecs 属性。例如，以下属性定义将时间间隔更改为十天：

  ```
  imq.log.file.rolloversecs=864000
  ```

- 要将转移条件更改为取决于文件大小，需要设置 imq.log.file.rolloverbytes 属性。例如，以下定义将在文件达到 500,000 字节的限制时将代理定向到转移文件。

  ```
  imq.log.file.rolloverbytes=500000
  ```

如果同时设置与时间和大小相关的转移属性，则转移将由最先达到的限制触发。前面已指出，代理最多维护九个转移文件。

可以在代理运行时设置或更改日志文件转移属性。要设置这些属性，请使用 imqcmd update bkr 命令。

将度量数据发送到日志文件

本节讲述使用代理日志文件报告度量信息的过程。有关配置记录程序的一般信息，请参见第 175 页上的 “配置和使用代理日志记录”。

➤ 使用日志文件报告度量信息

1. 配置代理的度量生成功能：

   a. 确认 imq.metrics.enabled=true

      默认情况下，用于日志记录的度量生成功能处于启用状态。

   b. 将度量生成时间间隔设置为合适的秒数。

      imq.metrics.interval=*interval*

      该值可以在 config.properties 文件中设置，也可以在启动代理时使用 -metrics *interval* 命令行选项进行设置。

2. 确认记录程序是否收集度量信息：

   imq.log.level=INFO

   这是默认值。该值可以在 config.properties 文件中设置，也可以在启动代理时使用 -loglevel *level* 命令行选项进行设置。

3. 确认记录程序是否已设置为将度量信息写入日志文件：

   imq.log.file.output=INFO

   这是默认值。它可以在 config.properties 文件中设置。

4. 启动代理。

下面显示了输出到日志文件的样例代理度量：

```
[21/Jul/2004:11:21:18 PDT]
连接：0     JVM 堆：8323072 字节 （7226576 可用）线程： 0 (14-1010)
    流入：0 个消息 （0 字节） 0 个包 （0 字节）
    流出：0 个消息 （0 字节） 0 个包 （0 字节）
流入速率：0 消息 / 秒 （0 字节 / 秒） 0 包 / 秒 （0 字节 / 秒）
流出速率：0 消息 / 秒 （0 字节 / 秒） 0 包 / 秒 （0 字节 / 秒）
```

有关度量数据的参考信息，请参见第 18 章 “度量参考”。

记录停用消息

可以通过对代理启用停用消息日志记录来监视物理目标。无论是否使用停用消息队列，都可以记录停用消息。

如果启用了停用消息日志记录，代理会记录以下类型的事件：

- 物理目标超出最大大小。
- 代理由于如下原因从物理目标中删除消息：
  - 到达目标大小限制。
  - 消息已过期。
  - 消息太大。
  - 代理试图处理消息时出错。

如果使用了停用消息队列，日志记录还会包括以下类型的事件：

- 代理将消息移动到停用消息队列中。

- 代理从停用消息队列中删除某条消息并将它丢弃。

默认情况下将禁用停用消息日志记录。要启用它，请设置代理属性 `imq.destination.logDeadMsgs`。

以交互方式显示度量

Message Queue 代理可以报告以下类型的度量：

- **Java 虚拟机 (Java Virtual Machine, JVM) 度量**。有关 JVM 堆大小的信息。
- **代理范围内的度量**。有关代理中存储的消息、流入或流出代理的消息以及内存使用的信息。按消息数和字节数跟踪消息。
- **连接服务度量**。有关连接和连接线程资源的信息，以及有关特定连接服务的消息流的信息。
- **目标度量**。有关流入和流出特定物理目标的消息的信息、有关物理目标的使用方的消息以及有关内存和磁盘空间使用情况的信息。

`imqcmd` 命令可以获取整个代理、单个连接服务以及单个物理目标的度量信息。要获得度量数据，通常应该使用 `imqcmd` 的 `metrics` 子命令。度量数据将按照您指定的时间间隔或次数写到控制台屏幕上。

也可以使用 `query` 子命令查看也包含配置信息的类似数据。有关更多信息，请参见第 183 页上的 “imqcmd query”。

imqcmd metrics

`imqcmd metrics` 的语法和选项分别如表 10-4 和表 10-5 所示。

表 10-4 `imqcmd metrics` 子命令语法

子命令语法	提供的度量数据

`metrics bkr`
`[-b` *hostName*`:`*portNumber*`]`
`[-m` *metricType*`]`
`[-int` *interval*`]`
`[-msp` *numSamples*`]`
	显示默认代理的代理度量，或显示指定主机和端口上的代理的代理度量。

或

表 10-4 imqcmd metrics 子命令语法 （续）

子命令语法	提供的度量数据
metrics svc -n *serviceName* [-b *hostName*:*portNumber*] [-m *metricType*] [-int *interval*] [-msp *numSamples*]	显示默认代理或指定主机和端口上的代理中指定服务的度量。
或	
metrics dst -t *destType* -n *destName* [-b *hostName*:*portNumber*] [-m *metricType*] [-int *interval*] [-msp *numSamples*]	显示指定类型和名称的物理目标的度量信息。

表 10-5 imqcmd metrics 子命令选项

子命令选项	描述
-b *hostName*:*portNumber*	指定报告的度量数据所对应的代理的主机名和端口。默认为 localhost:7676。
-int *interval*	指定显示度量的时间间隔 （以秒为单位）。默认值为 5 秒。
-m *metricType*	指定要显示的度量类型。
	ttl 显示流入和流出代理、服务或目标的消息和包的度量 （默认度量类型）。
	rts 显示消息和包流入和流出代理、连接服务或目标的速率的度量 （以一秒为衡量单位）。
	cxn 显示连接、虚拟内存堆和线程（仅适用于代理和连接服务）。
	con 显示与使用方有关的度量 （仅适用于目标）。
	dsk 显示磁盘使用情况度量 （仅适用于目标）。
-msp *numSamples*	指定输出中显示的样例数。默认为无限制 （无穷多）。
-n *destName*	指定报告的度量数据所对应的物理目标 （如果有）的名称。没有默认值。
-n *serviceName*	指定报告的度量数据所对应的连接服务 （如果有）。没有默认值。
-t *destType*	指定报告的度量数据所对应的物理目标 （如果有）的类型 （队列或主题）。没有默认值。

使用 metrics 子命令显示度量数据

本节讲述使用 metrics 子命令报告度量信息的过程。

➤ **使用 metrics 子命令**

1. 启动需要度量信息的代理。

   请参见第 64 页上的 “启动代理”。

2. 使用正确的 imqcmd metrics 子命令和选项，如表 10-4 和表 10-5 中所示。

度量输出：imqcmd metrics

本节包含 imqcmd metrics 子命令的输出示例。这些示例显示了代理范围、连接服务和物理目标度量。

代理范围内的度量

要获取在 10 秒的时间间隔内流入和流出代理的消息和包的速率，请使用 metrics bkr 子命令：

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

此命令产生类似如下内容的输出 （请参见第 304 页上的表 18-2 中的数据说明）：

```
------------------------------------------------------------
 消息数 / 秒      消息字节 / 秒      数据包 / 秒      数据包字节 / 秒
 传入   传出      传入     传出      传入   传出      传入     传出
------------------------------------------------------------
 0      0         27       56        0      0         38       66
 10     0       7365       56        10     10      7457     1132
 0      0         27       56        0      0         38       73
 0      10        27     7402        10     20      1400     8459
 0      0         27       56        0      0         38       73
```

连接服务标准

要获取 jms 连接服务所处理的消息和包的累计总数，请使用 metrics svc 子命令：

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

此命令产生类似如下内容的输出 （请参见第 306 页上的表 18-3 中的数据说明）：

```
-------------------------------------------------
消息数          消息字节          数据包          数据包字节
传入   传出     传入     传出     传入   传出     传入     传出
-------------------------------------------------
164    100     120704   73600    282    383     135967   102127
657    100     483552   73600    775    876     498815   149948
```

物理目标度量

要获取有关物理目标的度量信息，请使用 `metrics dst` 子命令：

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

此命令产生类似如下内容的输出 （请参见第 307 页上的表 18-4 中的数据说明）：

```
-------------------------------------------------------------------------------
   消息            消息字节              消息计数              消息总字节数 (k)      最大消息
传入   传出     传入     传出      当前值  峰值   平均值   当前值  峰值   平均值    字节数 (k)
-------------------------------------------------------------------------------
200    200    147200   147200      0     200    0        0     143    71          0
300    200    220800   147200    100     200    10       71    143    64          0
300    300    220800   220800      0     200    0        0     143    59          0
```

要获取有关物理目标的使用方的信息，请使用下面的 `metrics dst` 子命令：

```
imqcmd metrics dst -t q -n SimpleQueue -m con -u admin
```

此命令产生类似如下内容的输出 （请参见第 307 页上的表 18-4 中的数据说明）：

```
-----------------------------------------------------------------
  活动用户                  备份用户                  消息计数
当前值  峰值  平均值      当前值  峰值  平均值     当前值  峰值  平均值
-----------------------------------------------------------------
  1      1     0           0      0     0        944    1000  525
```

imqcmd query

表 10-6 中显示了 `imqcmd query` 的语法和选项以及该命令所提供的度量数据的说明。

表 10-6 imqcmd query 子命令语法

子命令语法	提供的度量数据
`query bkr [-b hostName:portNumber]`	有关当前存储在代理内存和持久性存储中的消息个数和消息字节数的信息 （请参见第 94 页上的 “显示代理信息”）。
或	
`query svc -n serviceName [-b hostName:portNumber]`	有关指定的连接服务的当前已分配线程数和连接数的信息 （请参见第 100 页上的 “显示连接服务信息”）。
或	
`query dst -t destType -n destName [-b hostName:portNumber]`	有关指定目标当前的生成方数目、活动和备份使用方的数目以及存储在内存和持久性存储中的消息数和消息字节数的信息 （请参见第 113 页上的 “显示有关物理目标的信息”）。

注 由于 imqcmd query 提供的度量数据有限，第 303 页上的第 18 章 “度量参考”的表中未提到此工具。

编写应用程序来监视代理

Message Queue 提供度量监视功能，通过该功能，代理可以将度量数据写入 JMS 消息，然后根据消息中包含的度量信息类型将它们发送到众多度量主题目标中的一个。

您可以通过编写具有下列功能的客户端应用程序来访问这些度量信息：订阅度量主题目标、使用这些目标中的消息以及处理消息中包含的度量信息。

共有五个度量主题目标，它们的名称以及传送到各个目标的度量消息的类型显示在表 10-7 中。

表 10-7 度量主题目标

主题名称	度量消息的类型
mq.metrics.broker	代理度量
mq.metrics.jvm	Java 虚拟机度量
mq.metrics.destination_list	目标及其类型的列表

表 10-7 度量主题目标 （续）

主题名称	度量消息的类型
mq.metrics.destination.queue.*monitoredDestinationName*	具有指定名称的队列的目标度量
mq.metrics.destination.topic.*monitoredDestinationName*	具有指定名称的主题的目标度量

设置基于消息的监视

本节讲述使用基于消息的监视功能来收集度量信息的过程。此过程包括客户端开发和管理任务这两方面。

➤ 设置基于消息的监视

1. 编写度量监视客户端。

   有关以编程方式编写订阅度量主题目标、使用度量消息并从这些消息中提取度量数据的客户端的说明，请参见 Message Queue Developer's Guide for Java Clients。

2. 通过在 `config.properties` 文件中设置代理属性值来配置代理的度量消息生成方：

   a. 启用度量消息生成。

      设置 `imq.metrics.topic.enabled=true`。

      默认值为 `true`。

   b. 设置生成度量消息的时间间隔 （以秒为单位）。

      设置 `imq.metrics.topic.interval=`*interval*。

      默认值为 60 秒。

   c. 指定是否需要让度量消息成为持久性消息 （即，当代理发生故障时它们是否能保留下来）。

      设置 `imq.metrics.topic.persist`。

      默认值为 `false`。

   d. 指定在删除度量消息之前它们能在各自的目标中保留的时间。

      设置 `imq.metrics.topic.timetolive`。

      默认值为 300 秒。

3. 设置需要对度量主题目标设置的任何访问控制。

   请参见下面 “安全性和访问注意事项”中的讨论。

4. 启动度量监视客户端。

   当使用方订阅度量主题时，系统会自动创建度量主题目标。创建度量主题后，代理的度量消息生成方就会开始向度量主题发送度量消息。

安全性和访问注意事项

出于以下两个原因，需要限制对度量主题目标的访问：

- 度量数据可能包含有关代理及其资源的敏感信息。
- 对度量主题目标的过多订阅可能会增大代理开销，从而给性能带来负面影响

出于这些方面的考虑，建议限制对度量主题目标的访问。

用于监视目的的客户端的验证和授权控制与其他任何客户端都相同。只有用户信息保留在 Message Queue 用户系统信息库中的用户才能连接到代理。

您可以通过访问控制属性文件来限制对特定度量主题目标的访问，从而提供更多保护，如第 129 页上的 “授权用户：访问控制属性文件”中所述。

例如， `accesscontrol.properties` 文件中的下列条目将拒绝除 user1 和 user2 之外的其他任何用户访问 mq.metrics.broker 度量主题。

```
topic.mq.metrics.broker.consume.deny.user=*
topic.mq.metrics.broker.consume.allow.user=user1,user2
```

下列条目仅允许用户 user3 监视主题 t1。

```
topic.mq.metrics.destination.topic.t1.consume.deny.user=*
topic.mq.metrics.destination.topic.t1.consume.allow.user=user3
```

根据度量数据的敏感度不同，您也可以使用加密连接将度量监视客户端连接至代理。有关使用加密连接的信息，请参见第 135 页上的 “使用基于 SSL 的服务”。

度量输出：度量消息

使用基于消息的监视 API 获得的度量数据输出是您编写的度量监视客户端的一个功能。您只会受到代理中度量生成器提供的数据的限制。有关这些数据的完整列表，请参见第 303 页上的 “度量参考”。

第 11 章

分析和调整消息服务

本章包括大量有关如何分析和调整 Message Queue 服务以优化消息传送应用程序性能的主题。本章包含以下主题：

关于性能

本节提供有关性能调整的一些背景信息。

性能调整过程

消息传送应用程序的性能取决于该应用程序与 Message Queue 服务之间的交互。因此，要获得最佳性能，需要应用程序开发者和管理员的共同努力。

优化性能的过程从应用程序设计开始，一直持续到部署应用程序之后对消息服务的调整阶段。性能调整过程包括下列阶段：

- 定义应用程序的性能要求
- 设计应用程序时要考虑到影响性能的因素 （特别是可靠性与性能之间的权衡）
- 建立性能衡量基线
- 调整或重新配置消息服务以优化性能

上述过程常常需要反复进行。在应用程序的部署过程中， Message Queue 管理员应该评估消息服务器是否适用于应用程序的总体性能要求。如果基准测试程序测试符合这些要求，管理员就可以如本章所述调整系统。但是，如果基准检验测试不符合性能要求，则可能需要重新设计应用程序或者修改部署体系结构。

性能方面

通常，性能是对消息服务将消息从生成方传送到使用方时的速度和效率的一种衡量。但是，根据您的需要，可能会有几个不同的性能方面对您特别重要。

连接负载 系统所能支持的消息生成方、消息使用方或并行连接的数量。

消息吞吐量 每秒钟能通过消息传送系统抽取的消息数或消息字节数。

等待时间 特定消息从消息生成方传送到消息使用方所需的时间。

稳定性 消息服务的总体可用性，或者当负载较重或发生故障时性能的下降程度。

效率 消息的传送效率，这是一种与使用的计算资源相关的消息吞吐量的衡量。

这些不同的性能方面通常是相辅相成的。如果消息吞吐量很高，则意味着消息在消息服务器上作为待办事项累积的可能性就会变小，因此延迟时间也应该较低 （单条消息可以很快地传送）。但是，等待时间可能取决于许多因素：通信链接的速度、消息服务器的处理速度和客户端的处理速度，以及其他许多方面。

在任何情况下，性能都有几个不同方面。哪些方面对您最重要？这通常取决于特定应用程序的要求。

基准测试程序

基准测试是为消息传送应用程序创建测试套件，并针对此测试套件衡量消息吞吐量或其他性能方面的一种过程。

例如，您可以创建这样一个测试套件：一定数量的生成方客户端使用一定数量的连接、会话和消息生成方，将标准大小的持久性或非持久性消息按照某个特定速率发送至队列或主题 （这完全取决于消息传送应用程序的设计）。同样，测试套件中还包括一定数量的使用方客户端，它们使用一定数量的连接、会话和特定类型的消息使用方 （通过特定的确认模式来使用测试套件的物理目标中的消息）。

使用标准的测试套件可以衡量消息的生成与使用之间所花的时间或者消息的平均吞吐速率，您还可以监视系统以观察连接线程使用情况、消息存储数据、消息流数据以及其他相关度量。这样就可以提高消息的生成速率、消息生成方的数量或者其他变量，直到性能受到负面影响为止。可能达到的最大吞吐量就是消息服务配置的基准。

使用此基准可以修改测试套件的某些特征。控制所有可能影响性能的因素时请小心（请参见第 194 页上的 “影响性能的应用程序设计因素”），您可以记录这些因素的更改如何影响基准。例如，您可以将连接数或消息大小增加五倍或十倍，并记录对性能产生的影响。

相反，您也可以将基于应用程序的因素保持不变，而以某种控制方式更改代理配置（例如，更改连接属性、线程池属性、 JVM 内存限制、限制行为、基于文件与基于 JDBC 的持久性等），并记录这些更改如何影响性能。

当您需要通过调整消息服务来增加所部署的应用程序的性能时，这种应用程序基准检验可以提供非常有价值的信息。基准测试程序可用于更准确地预测所做的更改或一系列更改产生的影响。

通常，基准测试程序应该在受控制的测试环境下运行，并且运行足够长的时间，以使消息服务能够稳定。（性能在启动时会受到实时编译的负面影响，因为要将 Java 代码转换为机器代码。）

基线使用模式

部署并运行消息传送应用程序后，建立基线使用模式是很重要的。您需要知道何时发生峰值需求，并且需要能够量化该需求。例如，需求通常会随最终用户的数量、活动级别、一天当中的时间，或者以上所有这些因素而发生波动。

要建立基线使用模式，您需要在一段较长的时间内监视消息服务器，并查看如下数据：

- 连接数
- 在代理 （或特定物理目标）中存储的消息数
- 流入和流出代理 （或特定物理目标）的消息
- 活动使用方数

还可以使用度量数据中提供的平均值和峰值。

将这些基线度量数据与设计时的期望值进行比较是非常重要的。通过执行此操作，可以检查客户端代码是否运行正常：例如，检查连接是否未保持打开或者已使用的消息是否仍然保留在未确认状态。这些编码错误会消耗消息服务器资源，并可能极大地影响性能。

基线使用模式有助于确定如何调整系统以优化性能。例如：

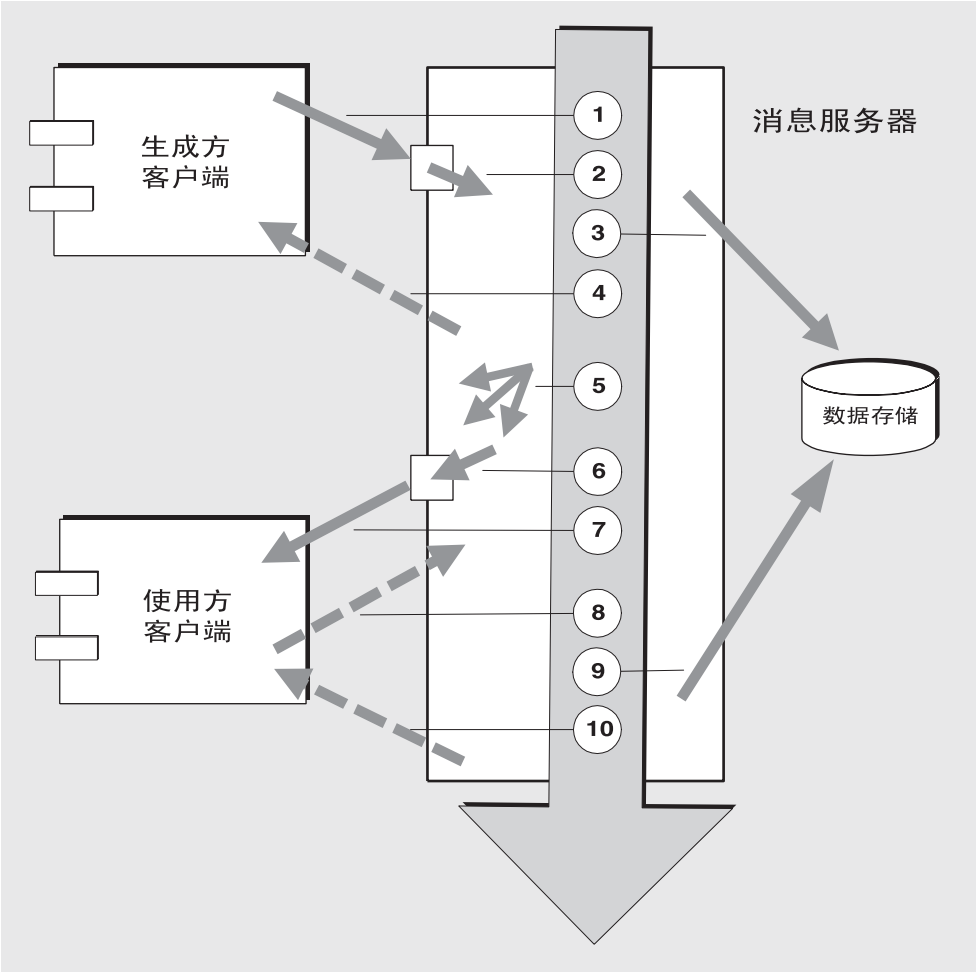
- 如果某个物理目标的使用频率明显高于其他目标，则可能需要对该物理目标设置比其他目标更高的内存限制，或者相应地调整限制行为。
- 如果需要的连接数明显大于允许的最大线程池大小，则可能需要增大线程池的大小，或者采用共享线程模型。
- 如果峰值消息流远远大于平均消息流，这可能影响您在内存不足时采取的限制行为。

通常，您对使用模式了解得越多，就能越好地将系统调整为这些模式，并计划将来的需要。

影响性能的因素

消息等待时间和消息吞吐量是两个主要的性能指示器，它们通常取决于典型消息在完成消息传送过程中的各个步骤时所需的时间。下列步骤显示了如何以持久、可靠的方式传送消息。这些步骤将在插图之后介绍。

图 11-1 通过 Message Queue 服务传送消息



1. 消息从生成方客户端传送到消息服务器。
2. 消息服务器读取消息。
3. 消息被放置到持久性存储器当中 （出于可靠性的考虑）。
4. 消息服务器确认收到消息 （出于可靠性的考虑）。
5. 消息服务器确定消息的路由。
6. 消息服务器写出消息。

7. 消息从消息服务器传送到使用方客户端。
8. 使用方客户端确认收到消息 （出于可靠性的考虑）。
9. 消息服务器处理客户端确认 （出于可靠性的考虑）。
10. 消息服务器确定已经处理客户端确认。

因为这些步骤是连续的，所以任何步骤都可能成为消息从生成方客户端到使用方客户端的传送过程的瓶颈。这些步骤中的大多数都取决于消息传送系统的物理特征：网络带宽、计算机处理速度和消息服务器体系结构等等。但是，有一些步骤还取决于消息传送应用程序的特征和该应用程序要求的可靠性级别。

以下各节讨论应用程序设计因素和消息传送系统因素这二者对性能的影响。尽管应用程序设计和消息传送系统因素在消息传送过程中紧密交互，但它们彼此是独立的。

影响性能的应用程序设计因素

应用程序的设计决策对消息传送的总体性能有很大影响。

对性能影响最大的主要是那些影响消息传送的可靠性因素。其中包括下列因素：

- 传送模式 （持久性 / 非持久性消息）
- 使用事务
- 确认模式
- 长期订阅与非长期订阅

其他影响性能的应用程序设计因素有：

- 使用选择器 （消息过滤）
- 消息大小
- 消息主体类型

接下来的几节讲述其中的每个因素对消息传送性能所产生的影响。通常，应当在性能与可靠性之间进行权衡。提高可靠性的因素可能会导致性能降低。

表 11-1 显示各个应用程序设计因素大体上如何影响消息传送性能。该表显示了两个方案 （一个高可靠性、低性能方案和一个高性能、低可靠性方案）和分别对应于这两个方案的应用程序设计因素选项。在这两种极端情况之间，有很多可以同时影响可靠性和性能的选项和折衷。

表 11-1 高可靠性和高性能方案比较

应用程序设计因素	高可靠性 低性能方案	高性能 低可靠性方案
传送模式	持久性消息	非持久性消息
使用事务	事务会话	非事务
确认模式	AUTO_ACKNOWLEDGE 或 CLIENT_ACKNOWLEDGE	DUPS_OK_ACKNOWLEDGE
长期 / 非长期订阅	长期订阅	非长期订阅
使用选择器	消息过滤	无消息过滤
消息大小	大量的小消息	少量的大消息
消息主体类型	复杂主体类型	简单主体类型

注 在接下来的图中，性能数据是在一个使用基于文件的持久性的双 CPU、 1002 Mhz Solaris 8 系统上生成的。性能测试首先预热 Message Queue 代理，使实时编译器可以优化系统，并使持久性数据库做好准备。

代理预热后，会创建单个生成方和单个使用方，并在 30 秒钟内生成消息。系统将记录使用方接收所有生成的消息所需的时间，并计算吞吐率 （每秒收到的消息数）。对于表 11-1 中所示的不同应用程序设计因素组合，此情形会重复出现。

传送模式 （持久性 / 非持久性消息）

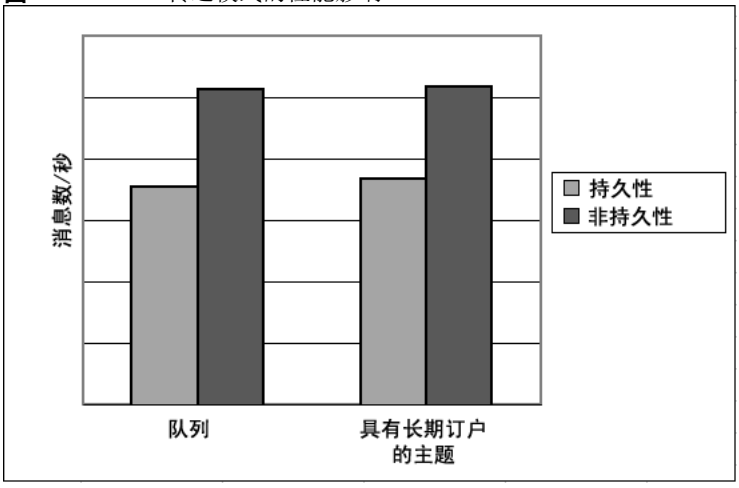
持久性消息能够在消息服务器发生故障的情况下保证消息的传送。代理会将消息存储在持久性存储中，直到所有预期的使用方都确认已使用消息为止。

代理处理持久性消息比处理非持久性消息要慢，原因如下：

- 代理必须可靠地存储持久性消息，以使其即使在代理出现故障时也不会丢失。
- 代理必须确认它所收到的每一条持久性消息。如果生成消息的方法正常返回，则说明向代理的传送成功。
- 根据客户端确认模式的不同，代理可能需要确认使用方客户端对持久性消息的确认。

持久性和非持久性模式的性能之间可以有很大的差别。图 11-2 比较了在两种可靠的传送情况下持久性和非持久性消息的吞吐量：同时将 10k 大小的消息传送至队列和具有长期订阅的主题。两种情况下都使用 AUTO_ACKNOWLEDGE 确认模式。

图 11-2 传送模式的性能影响



使用事务

事务是一种保证，保证在一个事务会话中生成和使用的所有消息将作为一个单元进行处理或不进行处理 （回滚）。

Message Queue 支持本地事务和分布式事务。

消息在事务会话中的生成或确认比在非事务会话中要慢，原因如下：

- 其他信息必须随每个生成的消息存储。
- 在某些情况下，事务中通常不该存储的消息也存储了 （例如，传送至没有订阅的主题目标的持久性消息通常应该删除，但在事务开始时，却没有关于订阅方面的信息）。
- 提交事务时，必须存储并处理该事务中有关消息的使用和确认的信息。

确认模式

确保可靠传送 JMS 消息的一种机制是，客户端确认使用了由 Message Queue 消息服务器传送给它的消息。

如果在客户端尚未确认消息之前就关闭了会话，或者如果消息服务器在处理确认之前发生了故障，代理将重新传送该消息，并设置一个 JMSRedelivered 标志。

对于非事务会话，客户端可以选择三种确认模式之一，这三种模式有其各自的性能特性：

- AUTO_ACKNOWLEDGE。使用方处理消息后，系统会自动确认该消息。此模式可确保在提供者发生故障后至少重新传送一次消息。
- CLIENT_ACKNOWLEDGE。应用程序控制确认消息的时间点。该会话中自上次确认以来处理的所有消息都将被确认。如果消息服务器在处理一组确认时发生故障，则该组中的一个或多个消息将被重新传送。
- DUPS_OK_ACKNOWLEDGE。此模式指示系统以一种惰性方式确认消息。在提供者发生故障后，可以重新传送多条消息。

（CLIENT_ACKNOWLEDGE 模式的用法与事务的用法类似，不同之处在于：它不能保证当提供者在处理过程中发生故障时，所有确认都将一起处理）。

确认模式影响性能的原因如下：

- AUTO_ACKNOWLEDGE 和 CLIENT_ACKNOWLEDGE 模式需要在代理与客户端之间有额外的控制消息。额外的控制消息会带来额外的处理开销，并干扰 JMS 有效负荷消息，从而导致处理延迟。
- 在 AUTO_ACKNOWLEDGE 和 CLIENT_ACKNOWLEDGE 模式中，客户端必须等到代理确认它已处理客户端的确认后，才能使用其他消息。（这种代理确认可确保代理不会无意地重新传送这些消息。）
- Message Queue 持久性存储必须用使用方收到的所有持久性消息的确认信息进行更新，因而降低了性能。

长期订阅与非长期订阅

主题目标的订户可以归为两类，即长期订阅的订户和非长期订阅的订户。

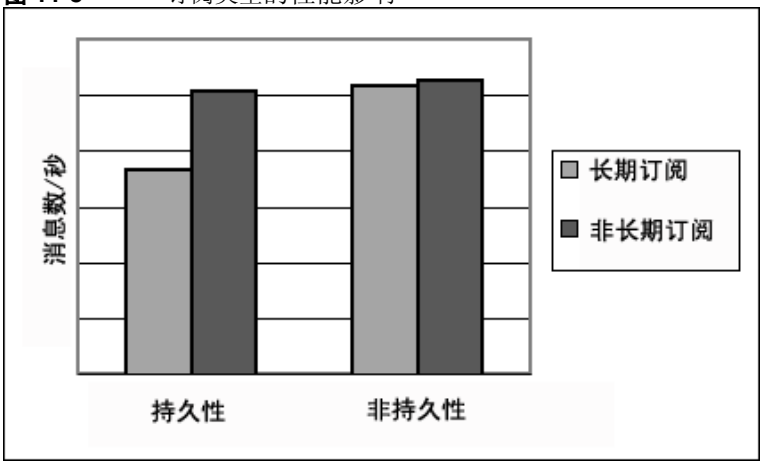
长期订阅的可靠性较高，但吞吐量较低，原因如下：

- Message Queue 消息服务器必须永久地存储指定给每个长期订阅的消息的列表，以便当消息服务器发生故障时，该列表可以在恢复之后使用。
- 长期订阅的持久性消息是永久存储的，以便当消息服务器发生故障时，消息仍可以在恢复后传送 （只要相应的使用方处于活动状态）。与此相反，非长期订阅的持久性消息不是永久存储的 （如果消息服务器发生故障，相应的使用方的连接就会断开，消息不再被传送）。

图 11-3 比较了两种情况下的长期订阅和非长期订阅主题目标的吞吐量：10K 大小的持久性和非持久性消息。两种情况下都使用 AUTO_ACKNOWLEDGE 确认模式。

从图 11-3 中可以看到，只有在持久性消息情况下，使用长期订阅的性能影响才会很明显，出现这种影响是因为持久性消息只有在长期订阅时才会永久存储，如上文所述。

图 11-3 订阅类型的性能影响



使用选择器 （消息过滤）

应用程序的开发者常常需要将消息组的目标定为特定使用方。实现此任务的方法有两种：将每组消息分别指向一个唯一的物理目标；或者是仅使用单个物理目标，并为每个使用方注册一个或多个选择器。

选择器是一种只请求特定消息的字符串，这些消息的属性值应该与传送到特定使用方的字符串匹配。例如，选择器 `NumberOfOrders >1` 仅传送 `NumberOfOrders` 属性值为 2 或更大值的消息。

将使用方注册到选择器会降低性能 （与使用多个物理目标相比），因为需要额外的处理来处理每个消息。如果使用选择器，则必须对它进行解析，以使它与将来的消息匹配。另外，路由每个消息时，都必须检索该消息的属性，并与选择器进行比较。但是，使用选择器为消息传送应用程序提供了更大的灵活性。

消息大小

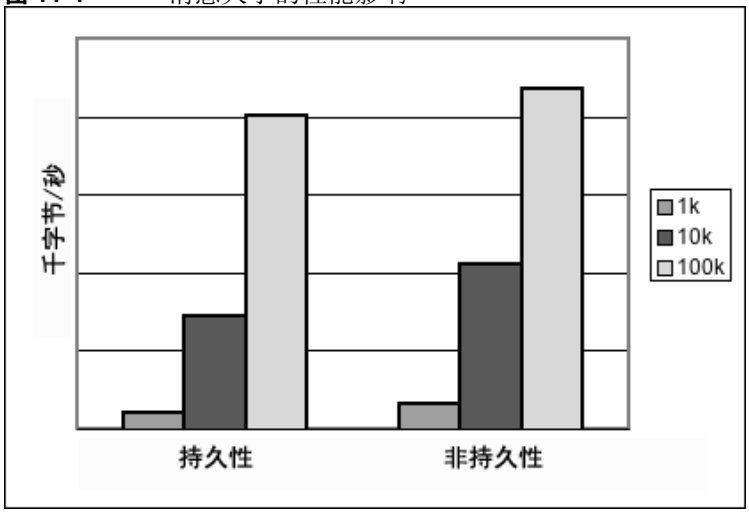
消息大小会影响性能，因为从生成方客户端到代理以及从代理到使用方客户端之间必须传递更多的数据，并且对于持久性消息，必须存储更大的消息。

但是，通过将多个较小的消息组织成一个消息在一批中传送，对各个消息的路由和处理就可以变得尽可能地简单，从而获得总体性能的提高。此时，就会丢失有关各个消息的状态的信息。

图 11-4 比较了在两种情况下， 1k、 10k 和 100k 大小的消息的吞吐量 （KB/ 秒）：持久性消息和非持久性消息。每种情况都将消息发送至队列目标，并使用 AUTO_ACKNOWLEDGE 确认模式。

图 11-4 显示在这两种情况下，传送较大消息的开销比传送较小消息的开销要小。您还可以看到在 1k 和 10k 大小的消息上出现的情况：非持久性消息较持久性消息接近 50% 的性能提高在 100k 大小的消息上并未得到体现，可能是因为在这种情况下，网络带宽已经成为消息吞吐量的瓶颈。

图 11-4 消息大小的性能影响



消息主体类型

JMS 支持五种消息主体类型，下面大致按复杂性顺序显示这些类型：

- BytesMessage：包含一组字节，格式由应用程序确定。
- TextMessage：一种简单的 java.lang.String。
- StreamMessage：包含 Java 基元值流。
- MapMessage：包含一组名称 / 值对。
- ObjectMessage：包含 Java 序列化对象。

尽管通常情况下消息类型是由应用程序的需要所决定的，但较复杂的类型（MapMessage 和 ObjectMessage）会增加性能成本 - 对数据进行序列化和反序列化的成本。性能成本取决于数据的简单或复杂程度。

影响性能的消息服务因素

消息传送应用程序的性能不但受应用程序设计的影响，而且还受执行消息路由和传送的消息服务的影响。

以下各节讨论影响性能的各个消息服务因素。了解这些因素的影响对于评估消息服务以及诊断并解决在部署的应用程序中可能发生的性能瓶颈来说至关重要。

Message Queue 服务中影响性能的最重要的因素有：

- 硬件
- 操作系统
- Java 虚拟机 (Java Virtual Machine, JVM)
- 连接
- 代理限制和行为
- 消息服务器体系结构
- 数据存储性能
- 客户端运行时配置

以下各节讲述其中的每个因素对消息传送性能所产生的影响。

硬件

对于 Message Queue 消息服务器和客户端应用程序而言， CPU 处理速度和可用内存都是消息服务性能的主要决定因素。大多数软件限制都可以通过增加处理能力来消除，而添加内存则可以同时提高处理速度和容量。但是，仅通过升级硬件来克服瓶颈通常过于昂贵。

操作系统

由于不同操作系统的效率不同，因此即使硬件平台相同，性能也会各不相同。例如，操作系统使用的线程模型会对消息服务器可以支持的并行连接数产生重要影响。在所有硬件都相同的情况下， Solaris 通常比 Linux 快，而后者通常又比 Windows 快。

Java 虚拟机 (Java Virtual Machine, JVM)

消息服务器是受主机 JVM 支持并运行在其中的一种 Java 进程。因此， JVM 处理是决定消息服务器路由和传送消息的速度和效率的重要因素。

特别是 JVM 的内存资源管理至关重要。必须为 JVM 分配足够的内存以适应不断增大的内存负载。另外， JVM 将定期回收未使用的内存，而这种内存回收会延迟消息的处理。 JVM 内存堆越大，内存回收过程中可能遇到的延迟就越长。

连接

客户端和代理之间的连接数和速度可能影响消息服务器可以处理的消息数以及消息的传送速度。

消息服务器连接限制

对消息服务器的所有访问都是通过连接进行的。对并行连接数的任何限制都会影响可以同时使用消息服务器的生成方或使用方客户端的数目。

与消息服务器的连接数通常受可用线程数的限制。可以对 Message Queue 进行配置以支持专用线程模型或共享线程模型 （请参见第 73 页上的 “线程池管理”）。

专用线程模型的速度非常快，因为每个连接都有专用的线程，但是连接数目受可用线程数的限制 （每个连接都有一个输入线程和一个输出线程）。共享线程模型对连接数不加任何限制，但是在大量连接之间共享线程会导致明显的开销和吞吐量延迟，特别是当这些连接都很繁忙时。

传输协议

Message Queue 软件允许客户端使用各种低级别的传输协议与消息服务器进行通信。 Message Queue 支持第 72 页上的 “连接服务”中所述的连接服务 （及相应协议）。

协议的选择基于应用程序的要求 （加密、可通过防火墙访问等），但是所作的选择会影响总体性能。

图 11-5 传输协议速度

图 11-5 反映不同协议技术的性能特征：

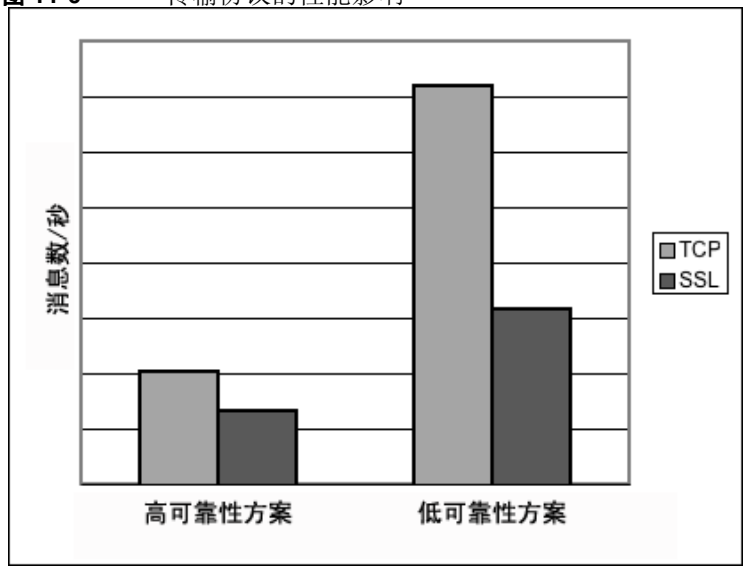
- TCP 提供与代理通信的最快方法。

- SSL 在收发消息时比 TCP 要慢 50% 到 70% （对于持久性消息是 50%，对于非持久性消息则接近 70%）。另外， SSL 在建立初始连接时比较慢 （可能需要数秒钟），因为客户端和代理 （在使用 HTTPS 的情况下则为 Web 服务器）需要建立专用密钥，以供加密要传输的数据时使用。性能的下降源自于加密和解密每个低级别 TCP 包时所需的额外处理。

  图 11-6 比较了两种情况下的 TCP 和 SSL 吞吐量：一个高可靠性方案 （将 1k 大小的持久性消息发送至长期订阅主题目标，并使用 AUTO_ACKNOWLEDGE 确认模式）和一个高性能方案 （将 1k 大小的非持久性消息发送至非长期订阅主题目标，并使用 DUPS_OK_ACKNOWLEDGE 确认模式）。

  图 11-6 显示了在高可靠性情况下具有较小影响的协议。这可能是因为在高可靠性情况下所需的持久性开销在限制吞吐量方面是比协议速度更重要的因素。

图 11-6 传输协议的性能影响



- HTTP 比 TCP 或 SSL 都要慢。它使用 Servlet，该 Servlet 在 Web 服务器上作为客户端与代理之间的代理运行。封装 HTTP 请求中的包以及消息经过两个跃点（客户端到 Servlet， Servlet 到代理）后才到达代理的要求均涉及性能开销。
- HTTPS 比 HTTP 慢是因为需要额外的开销以加密客户端和 Servlet 之间以及 Servlet 和代理之间的包。

消息服务器体系结构

Message Queue 消息服务器可以作为单个代理实现，也可以作为多个互相连接的代理实例 （即代理群集）实现。

随着连接到代理的客户端数量以及所传送的消息数量的不断增加，代理最终将超出资源限制 （例如文件描述符、线程和内存限制）。要适应不断增加的负载，方法之一是将更多的代理实例添加到 Message Queue 消息服务器，从而将客户端连接以及消息路由和传送分布到多个代理。

通常，如果客户端 （特别是消息生成方客户端）均匀地分布在群集中，则这种调整最为有效。由于在群集中的代理之间传送消息涉及到开销，因此对于连接数有限或消息传送速率有限的群集而言，其性能可能会比单个代理要低。

您也可以使用代理群集来优化网络带宽。例如，您可能希望在群集内的一组远程代理之间使用低速的长途网络链路，而在客户端与其各自的代理实例之间使用高速链接进行连接。

有关群集的详细信息，请参见第 9 章 “使用代理群集”。

代理限制和行为

消息服务器可能需要处理的消息吞吐量是消息服务器所支持的消息传送应用程序的使用模式的一个函数。但是，消息服务器在以下资源上有限：内存、 CPU 周期等。因此，消息服务器可能会在无响应或不稳定的位置发生崩溃。

Message Queue 消息服务器具有管理内存资源并防止代理用尽内存的内部机制。这些机制包括可以配置代理或其各自物理目标可以拥有的消息数或消息字节数的限制，以及当达到物理目标限制时可以实施的一组行为。

通过仔细的监视和调整，这些可配置机制可以用于平衡消息的内流和外流，使得不会发生系统过载。尽管这些机制会造成开销并限制消息的吞吐量，但它们可以维护操作的完整性。

数据存储性能

Message Queue 既支持基于文件的持久性模块，也支持基于 JDBC 的持久性模块。基于文件的持久性使用单独的文件存储持久性数据。基于 JDBC 的持久性使用 Java 数据库连接 (Java Database Connectivity, JDBC™) 接口，并需要符合 JDBC 的数据存储。基于文件的持久性通常比基于 JDBC 的持久性快；但是，某些用户对于符合 JDBC 的存储所提供的冗余和管理控制更感兴趣。

如果是基于文件的持久性，您可以通过指定让持久性操作将内存中的状态与数据存储同步，来最大限度地提高可靠性。这有助于消除因系统崩溃而导致的数据丢失，但代价是性能的下降。

客户端运行时配置

Message Queue 客户端运行时提供客户端应用程序及其与 Message Queue 消息服务的接口。它支持客户端向物理目标发送消息以及接收来自这些目标的消息所需的所有操作。客户端运行时是可配置的（通过设置连接工厂属性值），您可以控制其行为的各个方面（例如连接流度量、使用方流限制和连接流限制），从而提高性能和消息吞吐量。有关这些功能和用来配置这些功能的属性的详细信息，请参见第 209 页上的 “客户端运行时消息流调整”。

调整配置以提高性能

系统调整

以下各节讲述您可以对操作系统、 JVM 和通信协议所做的调整。

Solaris 调整：CPU 利用率、分页 / 交换 / 磁盘 I/O

有关对操作系统的调整，请参见系统文档。

Java 虚拟机调整

默认情况下，代理使用大小为 192MB 的 JVM 堆。这对于较大的消息负载来说通常太小，应该增大。

当代理快要耗尽 Java 对象使用的 JVM 堆空间时，它将使用各种技术（如流控制和消息交换）来释放内存。在极端情况下，代理甚至关闭客户端连接以释放内存并减少消息内流。所以最好将最大 JVM 堆空间设置得足够大，以避免这种情况。

但是，与系统的物理内存相比，如果最大 Java 堆空间设置过大，代理将继续增大 Java 堆空间，直至整个系统耗尽内存。这会导致性能的降低、不可预计的代理崩溃和 / 或影响系统中运行的其他应用程序和服务的行为。通常，需要有足够的物理内存以供操作系统和其他应用程序在计算机上运行。

总的说来，好的方法是：估算正常和峰值系统内存容量，并配置 Java 堆大小，使其足以提供良好性能，但同时不应过大，以免引起系统内存问题。

要更改代理的最小和最大堆大小，请在启动代理时使用 `-vmargs` 命令行选项。例如：

```
/usr/bin/imqbrokerd -vmargs "-Xms256m -Xmx1024m"
```

此命令将启动 Java 堆大小设置为 256MB，将最大 Java 堆大小设置为 1GB。

- 在 Solaris 或 Linux 上，如果通过 `/etc/rc*`（即 `/etc/init.d/imq`）启动代理，请在 `/etc/imq/imqbrokerd.conf` (Solaris) 或 `/etc/opt/sun/mq/imqbrokerd.conf` (Linux) 文件中指定代理的命令行参数。有关更多信息，请参见该文件中的注释。
- 在 Windows 上，如果将代理作为 Window 服务启动，请使用 `imqsvcadmin install` 命令的 `-vmargs` 选项指定 JVM 参数。请参见第 13 章 “命令行参考”中的 “服务管理器实用程序”。

在任何情况下，都应当通过检查代理的日志文件或通过使用 `imqcmd metrics bkr -m cxn` 命令来验证设置。

调整传输协议

选择了符合应用程序需要的协议后，基于该协议进行其他调整可能有助于提高性能。

可以使用下面三个代理属性修改协议的性能：

- `imq.protocol.`*protocolType*`.nodelay`
- `imq.protocol.`*protocolType*`.inbufsz`
- `imq.protocol.`*protocolType*`.outbufsz`

对于 TCP 和 SSL 协议，这些属性会影响客户端与代理之间的消息传送速度。对于 HTTP 和 HTTPS 协议，这些属性会影响 Message Queue 隧道 Servlet （在 Web 服务器上运行）与代理之间的消息传送速度。对于 HTTP/HTTPS 协议，还有其他可以影响性能的属性 （请参见第 207 页上的 “HTTP/HTTPS 调整”）。

协议调整属性将在以下各节中讲述。

nodelay

`nodelay` 属性影响给定协议的 Nagle 算法 （TCP/IP 上的 `TCP_NODELAY` 套接字级选项的值）。 Nagle 算法用于提高使用慢速连接 （例如广域网 (Wide-Area Network, WAN)）的系统上的 TCP 性能。

如果使用了此算法， TCP 将通过把多个数据捆绑为较大的包来尝试防止将多个小块数据发送到远程系统。如果写入套接字中的数据没有填满需要的缓冲区大小，协议将延迟发送包，直到缓冲区被填满，或者经过了特定的延迟时间为止。填满了缓冲区或者发生了超时后，包将被发送。

对于大多数消息传送应用程序，如果包发送过程中没有延迟 （Nagle 算法未启用），则性能是最佳的。这是因为客户端与代理之间的大多数交互都是请求 / 响应交互：客户端向代理发送数据包，并等待响应。例如，典型的交互包括：

- 创建连接
- 创建生成方或使用方
- 发送持久性消息 （代理确认收到消息）
- 在 AUTO_ACKNOWLEDGE 或 CLIENT_ACKNOWLEDGE 会话中发送客户端确认 （代理确认对客户端确认的处理）

对于这些交互，大多数包都比缓冲区大小要小。这意味着如果使用 Nagle 算法，代理会在向使用方发送响应之前延迟几毫秒。

但是，在连接较慢以及不需要代理响应的情况下， Nagle 算法可以提高性能。例如，客户端发送非持久性消息或者客户端确认未被代理确认（DUPS_OK_ACKNOWLEDGE 会话）就属于这样的情况。

inbufsz/outbufsz

inbufsz 属性用于在读取来自套接字的数据的输入流上设置缓冲区大小。同样，outbufsz 用于设置代理用来将数据写入套接字的输出流的缓冲区大小。

通常，这两个参数都应该设置为比收发的平均包稍大的值。一个很好的经验是将这些属性值设为平均包的大小再加上 1k （舍入为最接近的 k 倍值）。

例如，如果代理正在接收主体大小为 1k 的包，则该包的总体大小 （消息主体 + 标题 + 属性）约为 1200 字节。大小为 2k （2048 字节）的 inbufsz 可以提供合理的性能。

增大 inbufsz 或 outbufsz （使其大于该值）可以稍微提高性能，但这样会增加每个连接所需的内存。

图 11-7 显示了对大小为 1K 的包更改 inbufsz 值的结果。

图 11-7 对大小为 1K （1024 字节）的包更改 inbufsz 值的结果

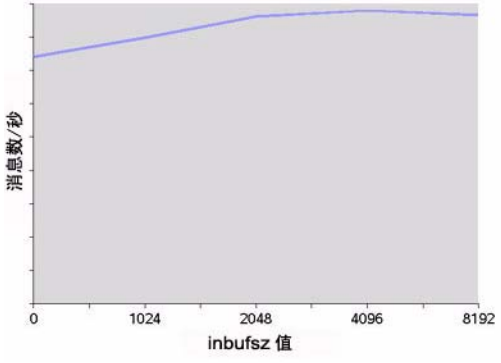
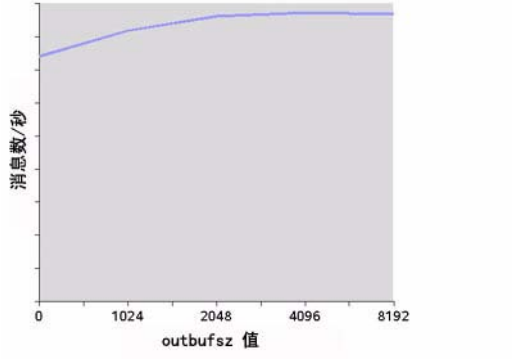


图 11-8 显示了对大小为 1K 的包更改 outbufsz 值的结果。

图 11-8 对大小为 1K （1024 字节）的包更改 outbufsz 值的结果



HTTP/HTTPS 调整

除了前面两节讨论的一般属性之外， HTTP/HTTPS 的性能还受到客户端向作为 Message Queue 隧道 Servlet 宿主的 Web 服务器发出 HTTP 请求的速度的限制。

Web 服务器可能需要优化，以处理单个套接字上的多个请求。在 JDK 1.4 版及更高版本中，与 Web 服务器的 HTTP 连接会一直保持 （与 Web 服务器的套接字保持打开），以尽量减少 Web 服务器在处理多个 HTTP 请求时使用的资源。如果使用 JDK 1.4 版的客户端应用程序的性能比运行早期 JDK 版本的同一应用程序要低，则可能需要调整 Web 服务器的 keep-alive 配置参数以提高性能。

除了这样的 Web 服务器调整之外，您还可以调整客户端轮询 Web 服务器的频率。HTTP 是一种基于请求的协议。这意味着使用基于 HTTP 的协议的客户端需要定期检查 Web 服务器，以查看是否有消息在等待。 imq.httpjms.http.pullPeriod 代理属性 （以及相应的 imq.httpsjms.https.pullPeriod 属性）指定 Message Queue 客户端运行时轮询 Web 服务器的频率。

如果 pullPeriod 值为 -1 （默认值），客户端运行时将在前一个请求返回后立即轮询服务器，从而最大限度地提高各个客户端的性能。结果，每个客户端连接都会在 Web 服务器中独占一个请求线程，这样可能会耗费 Web 服务器资源。

如果 pullPeriod 值为正数，客户端运行时将定期向 Web 服务器发送请求，以查看是否有挂起的数据。在这种情况下，客户端不会独占 Web 服务器中的请求线程。因此，如果有大量客户端在使用 Web 服务器，您可以通过将 pullPeriod 设为正值来节省 Web 服务器资源。

调整基于文件的持久性存储

有关调整基于文件的持久性存储的信息，请参见第 75 页上的 “持久性服务”。

代理调整

以下各节介绍为了提高性能可以对代理属性进行的调整。

内存管理：提高代理在负载下的稳定性

内存管理可以分别在各个目标上配置，也可以在系统范围级别内 （对于所有目标）配置。

使用物理目标限制

有关物理目标限制的信息，请参见第 6 章 “管理物理目标”。

使用系统范围的限制

如果消息生成方的数目超过消息使用方的数目，则消息可能在代理中堆积。代理包含限制生成方以及在内存很低的情况下将消息交换出活动内存的机制，但最好还是对代理可以保持的消息总数和消息字节总数进行严格限制。

可以通过设置 `imq.system.max_count` 和 `imq.system.max_size` 代理属性来控制这些限制。

例如：

```
imq.system.max_count=5000
```

上面定义的值表示代理最多只能保存 5000 条未传送 / 未确认的消息。如果发送了其他消息，它们将被代理拒绝。如果消息是持久性的，当生成方尝试发送该消息时，会收到一个异常。如果消息是非持久性的，代理将在不给出任何提示的情况下丢弃该消息。

如果在发送消息的过程中返回了异常，客户端应该暂停片刻，然后再次尝试发送。（请注意，异常绝不会是由于代理未能接收消息而引发的；所引发的异常都是由发送方客户端检测到的。）

多使用方队列性能

多队列使用方处理队列目标中的消息的效率取决于下列可配置的队列目标属性：

- 活动使用方数 (`maxNumActiveConsumers`)
- 可以在一批中传送给使用方的消息的最大数量 (`consumerFlowLimit`)

要达到最优的消息吞吐量，必须有足够数量的活动使用方以适应队列的消息生成方的速率，并且队列中的消息必须以最大化使用速率的方式路由和传送给活动使用方。 Sun Java System Message Queue 技术概述中介绍了在多个使用方之间平衡消息传送的一般机制。

如果消息在队列中堆积，这可能是因为没有足够的活动使用方来处理消息负载。也可能是每批传送给使用方的消息太多，导致消息在使用方堆积。例如，如果每批的大小 (consumerFlowLimit) 太大，某个使用方就可能会收到一个队列中的所有消息，而其他活动使用方则一个也没有收到。如果使用方速度特别快，这也不会成为问题。

但是如果使用方相对较慢，而您又希望将消息均匀地分配给它们，则需要将每一批的大小减小。每一批的大小越小，将消息传送到使用方所需的开销就越多。但是，对于较慢的使用方，使用较小的批大小通常能获得网络性能的提升。

客户端运行时消息流调整

本节讨论影响性能的流控制行为 （请参见第 204 页上的 “客户端运行时配置”）。这些行为可配置为连接工厂受管理对象的属性。有关设置连接工厂属性的信息，请参见第 8 章 “管理受管理对象”。

消息流度量

客户端收发的消息 （**有效负荷消息**）以及 Message Queue 控制消息通过同一客户端 / 代理连接传递。如果控制消息 （例如代理确认）被有效负荷消息的传送阻挡，则控制消息的传送会发生延迟。为防止此类拥塞， Message Queue 会度量通过连接进行的有效负荷消息流。

有效负荷消息是成批的 （由连接工厂属性 imqConnectionFlowCount 指定），以便只传送数目设定的一组消息。传送完一批后，将暂停有效负荷消息的传送，而只传送暂挂的控制消息。当另一批有效负荷消息传送后，接着又传送暂挂的控制消息，如此循环往复。

如果客户端执行的是需要代理作出大量响应的操作，则 imqConnectionFlowCount 应保持较低的值：例如，当客户端使用的是 CLIENT_ACKNOWLEDGE 或 AUTO_ACKNOWLEDGE 模式、持久性消息、事务或队列浏览器，或者正在添加或删除使用方时。从另一方面来说，如果客户端仅在使用 DUPS_OK_ACKNOWLEDGE 模式的连接上有简单的使用方，则可以增大 imqConnectionFlowCount 而不会降低性能。

消息流限制

在遇到本地资源 （例如内存）限制之前，存在 Message Queue 客户端运行时可以处理的有效负荷消息数限制。如果达到了此限制，性能将受影响。因此， Message Queue 允许您限制每个使用方 （或每个连接）能够通过连接传送和能够在客户端运行时中缓冲以等待使用的消息数。

使用方流限制

当传送到客户端运行时的有效负荷消息数超过任意使用方的 imqConsumerFlowLimit 值时，将停止传送该使用方的消息。仅当该使用方的未使用消息数下降至低于 imqConsumerFlowThreshold 设置的值时才会恢复消息传送。

下例说明了如何使用这些限制，以主题使用方的默认设置为例：

```
imqConsumerFlowLimit=1000

imqConsumerFlowThreshold=50
```

创建使用方后，代理将向此使用方传送第一批 1000 条消息 （前提是有这么多），中间不会暂停。发送 1000 条消息后，代理将停止传送，除非客户端运行时要求更多消息。客户端运行将保存这些消息，直到应用程序处理它们为止。在要求代理发送下一批消息之前，客户端运行时会允许应用程序使用至少 50% (imqConsumerFlowThreshold) 的消息缓冲区容量 （即 500 条消息）。

在同等情况下，如果阈值为 10%，则客户端运行时会等待应用程序使用至少 900 条消息，然后才会要求发送下一批消息。

下一批消息的大小按如下计算：

imqConsumerFlowLimit - （**缓冲区中当前暂挂的消息数**）

因此，如果 imqConsumerFlowThreshold 为 50%，则下一批的大小会在 500 和 1000 之间波动，这取决于应用程序处理消息的速度。

如果 imqConsumerFlowThreshold 设置得过高 （接近 100%），代理就会发送较小的分批消息，这样会降低消息的吞吐量。如果该值设置过低 （接近 0%），则客户端可以在代理传送下一组消息之前处理完剩余的缓冲消息，从而再次导致消息吞吐量下降。通常，除非您有特别的性能或可靠性考虑，否则不需要更改 imqConsumerFlowThreshold 属性的默认值。

基于使用方的流控制 （特别是 imqConsumerFlowLimit）是管理客户端运行时中的内存的最好方法。通常，根据客户端应用程序的不同，您应该知道在任意连接上需要支持的使用方数、消息的大小以及可用于客户端运行时的内存总量。

连接流限制

但是在某些客户端应用程序中，使用方数量可能是不确定的，这取决于最终用户所作的选择。在这些情况下，您仍可以使用连接级流限制来管理内存。

连接级流控制可以限制针对一个连接上的**所有**使用方而缓冲的消息总数。如果该数目超过了 imqConnectionFlowLimit 的值，则通过该连接进行的消息传送将停止，直到消息总数降到连接限制以下为止。（只有当 imqConnectionFlowLimitEnabled 设置为 true 时， imqConnectionFlowLimit 属性才会启用。）

在会话中排队的消息数是使用该会话的消息使用方数量以及每个使用方的消息负载的函数。如果客户端在生成或使用消息时表现出延迟，您通常可以通过下列操作来提高性能：重新设计应用程序，以便在更大数量的会话之间分布消息生成方和使用方，或者在更大数量的连接之间分布会话。

第 12 章

问题疑难解答

本章介绍如何了解及解决以下问题：

出现问题时，检查所安装 Message Queue 软件的版本号会很有帮助。可以使用版本号来确保目前正在使用的文档版本与软件版本相匹配。向 Sun 报告问题时，也需要用到版本号。要检查版本号，请执行以下命令：

```
imqcmd -v
```

客户端无法建立连接

此问题的症状如下：

- 客户端无法建立新连接。
- 客户端无法在连接失败时自动重新连接。

本节探讨了如下可能的原因：

- 客户端应用程序不关闭连接，导致连接数超出资源限制。
- 代理未运行或者网络连接有问题。
- 连接服务处于非活动状态或者已暂停。
- 相对于所需的连接数而言，可用线程数不足。
- 相对于 Solaris 或 Linux 操作系统上需要的连接数而言，文件描述符不足。
- TCP 待办事项限制了可以同时建立的新连接请求的数目。
- 操作系统限制了并行连接数。
- 对用户的验证或授权失败。

客户端应用程序不关闭连接，导致连接数超出资源限制

确认这是否就是问题的起因

列出代理的所有连接：

```
imqcmd list cxn
```

输出结果将列出所有连接以及发起每个连接的主机，从而显示具体是哪些客户端的打开连接数超出限制。

解决此问题

重写有问题的客户端，以关闭未使用的连接。

代理未运行或者网络连接有问题

确认这是否就是问题的起因

- 远程登录到代理的主端口 （例如，默认端口 7676），并验证代理是否以端口映射器输出作为响应。
- 验证代理进程是否正在主机上运行。

解决此问题

- 启动代理。
- 修复网络连接问题。

连接服务处于非活动状态或者已暂停

确认这是否就是问题的起因

检查所有连接服务的状态：

```
imqcmd list svc
```

如果某个连接服务的状态显示为 unknown 或 paused，客户端将无法使用该服务建立连接。

解决此问题

- 如果连接服务的状态显示为 unknown，它将不会出现在活动服务列表 (imq.service.active) 中。如果是基于 SSL 的服务，则还可能是因为服务未正确配置，导致代理在代理日志中生成下面的条目：错误 [B3009]：无法启动服务 ssljms：[B4001]：无法打开 ssljms 服务的协议 tls...，后面说明了引起此异常的根源。

  要正确配置 SSL 服务，请参见第 135 页上的 “使用基于 SSL 的服务”。

- 如果连接服务的状态显示为 paused，请恢复该服务 （请参见第 102 页上的 “暂停和恢复连接服务”）。

相对于所需的连接数而言，可用线程数不足

确认这是否就是问题的起因

在代理日志中检查下面的条目：

```
警告 [B3004]：服务上没有可以用来处理新连接的线程 ... 关闭新连接。
```

此外，请检查连接服务上的连接数以及当前使用的线程数 （使用以下格式之一）：

```
imqcmd query svc -n serviceName
```

```
imqcmd metrics svc -n serviceName -m cxn
```

每个连接都需要两个线程：一个用于传入消息，另一个用于传出消息（请参见第 73 页上的 “线程池管理”）。

解决此问题

- 如果使用专用线程池模型 (`imq.`*`serviceName`*`.threadpool_model=dedicated`)，则最大连接数是该线程池中的最大线程数的一半。因此，要增加连接数，请增加线程池的大小 (`imq.`*`serviceName`*`.max_threads`) 或切换到共享线程池模型。
- 如果使用共享线程池模型 (`imq.`*`serviceName`*`.threadpool_model=shared`)，则最大连接数是下面两个属性的乘积的一半：连接监视限制 (`imq.`*`serviceName`*`.connectionMonitor_limit`) 和最大线程数 (`imq.`*`serviceName`*`.max_threads`)。因此，要增加连接数，可以增加线程池的大小或增大连接监视限制。
- 最终，可支持的连接数 （或连接上的吞吐量）将达到输出 / 输出限制。这种情况下，可以使用多代理群集在群集内的代理实例之间分布连接。

相对于 Solaris 或 Linux 操作系统上需要的连接数而言，文件描述符不足

有关此问题的详细信息，请参见第 64 页上的 “设置文件描述符限制”。

确认这是否就是问题的起因

在代理日志中查找与下面显示的条目类似的条目：打开了太多文件。

解决此问题

增大文件描述符限制，如 `ulimit` 手册页中所述。

TCP 待办事项限制了可以同时建立的新连接请求的数目

TCP 待办事项限制可以同时存储在系统待办事项 (`imq.portmapper.backlog`) 中的连接请求数，超过此限制后，端口映射器将拒绝额外的请求。（在 Windows 操作系统上，有一种硬编码的待办事项限制：Windows 台式机限制为 5，而 Windows 服务器限制为 200。）

出于待办事项限制而拒绝请求通常是一种由于同时连接请求数过多而导致的瞬态现象。

确认这是否就是问题的起因

检查代理日志。首先，检查代理是否在接受某些连接的同时拒绝其他连接。其次，检查说明拒绝连接原因的消息。如果找到此类消息，则说明问题可能不是由 TCP 待办事项引起的，因为代理不记录由于 TCP 待办事项而引起的连接拒绝事件。

如果记录了一些成功连接，但未记录任何连接拒绝事件，则问题可能是由 TCP 待办事项引起的。

解决此问题

以下方法可用于解决 TCP 待办事项限制：

- 对客户端进行编程，使其在较短的时间间隔后重试所尝试的连接 （此方法之所以生效通常是由于此问题的瞬态性）。
- 提高 `imq.portmapper.backlog` 的值。
- 检查客户端是否过于频繁地反复关闭和打开连接。

操作系统限制了并行连接数

Windows 操作系统许可证对支持的并行远程连接数进行了限制。

确认这是否就是问题的起因

检查是否有可用于连接的足够线程 （使用 `imqcmd query svc`），并检查您的 Windows 许可协议的条款。如果您可以从本地客户端建立连接，但不能从远程客户端建立连接，则操作系统的限制可能就是问题的起因。

解决此问题

- 升级 Windows 许可证，以允许更多的连接。
- 通过设置多代理群集在多个代理实例之间分布连接。

对用户的验证或授权失败

验证可能因为密码错误而失败，原因是在用户系统信息库中没有该用户的条目，或者该用户没有对连接服务的访问权限。

确认这是否就是问题的起因

检查代理日志中的条目，查看是否有 `Forbidden` 错误消息。此消息指明存在验证错误，但不会指明该错误的原因。

- 如果使用的是基于文件的用户系统信息库，请输入下面的命令：

  `imqusermgr list -i` *instanceName* `-u` *userName*

- 如果输出结果显示的是用户，说明可能提交了错误的密码。如果输出显示以下错误，则说明用户系统信息库中不包含任何条目：

  `错误 [B3048]：密码文件中不存在用户，`

- 如果使用的是 LDAP 服务器用户系统信息库，请使用相应的工具检查是否存在该用户的条目。
- 检查访问控制文件以查看是否对连接服务有访问限制。

解决此问题

- 如果在用户系统信息库中没有该用户的条目，则将该用户添加到用户系统信息库中 （请参见第 125 页上的 “填充和管理用户系统信息库”）。
- 如果使用了错误的密码，请提供正确的密码。
- 如果访问控制属性设置不当，则编辑访问控制属性文件以授予连接服务权限（请参见第 132 页上的 “用于连接服务的访问控制”）。

连接的吞吐量太低

此问题的症状如下：

- 消息的吞吐量与预期不符。
- 支持的代理连接数受到限制不是如第 214 页上的 “客户端无法建立连接”中所述的原因，而是由于消息的输入 / 输出速率而导致的。

本节探讨了如下可能的原因：

- 网络连接或 WAN 太慢。
- 与 TCP 相比，连接服务协议本身就慢。
- 连接服务协议未进行优化调整。
- 消息太大，以致于占用了过多的带宽。
- 使连接吞吐量变低的可能原因也就是消息传送过程中某个步骤的瓶颈。

网络连接或 WAN 太慢

确认这是否就是问题的起因

Ping 网络，查看返回 ping 需要的时间，然后咨询网络管理员。另外还可以使用本地客户端发送并接收消息，并将传送时间与远程客户端 （使用网络链路）的传送时间相比。

解决此问题

如果连接太慢，则升级网络链路。

与 TCP 相比，连接服务协议本身就慢

例如，基于 SSL 或基于 HTTP 的协议要比 TCP 慢 （请参见第 201 页上的图 11-5）。

确认这是否就是问题的起因

如果您使用的是基于 SSL 或基于 HTTP 的协议，请尝试使用 TCP，然后比较传送时间。

解决此问题

应用程序的要求通常会限定使用的协议，因此您可以做的事情就很少了，无非是按第 205 页上的 “调整传输协议”中所述尝试调整协议而已。

连接服务协议未进行优化调整

确认这是否就是问题的起因

尝试调整协议，确定是否发生了变化。

解决此问题

尝试按第 205 页上的 “调整传输协议”中所述调整协议。

消息太大，以致于占用了过多的带宽

确认这是否就是问题的起因

尝试使用较小的消息运行基准测试程序。

解决此问题

- 请应用程序开发者对应用程序进行修改以使用消息压缩功能，请参见 Message Queue Developer's Guide for Java Clients。
- 将消息作为要发送的数据的通知来使用，但使用其他协议来移动数据。

使连接吞吐量变低的可能原因也就是消息传送过程中某个步骤的瓶颈

确认这是否就是问题的起因

如果上述每一条看来都不是造成连接吞吐量变低的原因，请参见第 193 页上的图 11-1 了解其他可能的瓶颈，并检查与以下问题相关的症状：

- 第 221 页上的 “消息的生成过程延迟或速度减慢”
- 第 223 页上的 “消息堆积”
- 第 227 页上的 “消息服务器吞吐量呈间歇性”

解决此问题

遵循前面有关问题疑难解答的各节中提供的问题解决方针。

客户端无法创建消息生成方

此问题的症状如下：

- 无法为物理目标创建消息生成方；客户端收到异常。

本节探讨了如下可能的原因：

- 物理目标被配置为仅允许有限数目的生成方。
- 由于访问控制属性文件中的设置，用户未获得创建消息生成方的授权。

物理目标被配置为仅允许有限数目的生成方

限制某个物理目标所支持的生成方 (maxNumProducers) 数目是避免消息在该物理目标上堆积的方法之一。

确认这是否就是问题的起因

检查物理目标 （请参见第 113 页上的 “显示有关物理目标的信息”）：

```
imqcmd query dst
```

输出结果将显示当前的生成方数目以及 maxNumProducers 的值。如果这两个值相同，则说明生成方的数目已达到所配置的限制。如果新的生成方被代理拒绝，代理将返回 “ResourceAllocationException [C4088]：已达到 JMS 目的地限制”消息，且在代理日志中生成如下条目：[B4183]：无法将生成方添加到目的地。

解决此问题

增大 maxNumProducers 属性的值 （请参见第 114 页上的 “更新物理目标属性”）。

由于访问控制属性文件中的设置，用户未获得创建消息生成方的授权

确认这是否就是问题的起因

如果新的生成方被代理拒绝，代理将返回以下消息：

JMSSecurityException [C4076]：客户端没有在目的地上创建生成方的权限

代理还在代理日志中记录以下条目：

[B2041]：目标上的生成方被拒绝并且 [B4051]：禁用 guest。

解决此问题

更改访问控制属性，允许用户生成消息 （请参见第 134 页上的 “对物理目标的访问控制”）。

消息的生成过程延迟或速度减慢

此问题的症状如下：

- 发送持久性消息时， send 方法不返回，并且客户端发生阻塞。
- 发送持久性消息时，客户端收到异常。
- 生成方客户端速度变慢。

本节探讨了如下可能的原因：

- 消息服务器上堆满了待办事项，使得消息生成方速度减慢。
- 代理无法将持久性消息保存到数据存储中。
- 代理的确认超时太短。
- 生成方客户端遇到了 JVM 限制。

消息服务器上堆满了待办事项，使得消息生成方速度减慢

堆满待办事项的服务器将消息堆积在代理内存中。

当物理目标内存中的消息数或消息字节数达到配置的限制时，代理会尝试根据指定的限制行为来节省内存资源。以下限制行为会使消息生成方速度减慢：

- FLOW_CONTROL：代理不会立即确认收到持久性消息 （这样就会阻塞生成方客户端）。
- REJECT_NEWEST: 代理将拒绝新的持久性消息。

同样，如果代理范围的内存 （对于所有物理目标）中消息数或消息字节数达到配置的限制，代理将尝试通过拒绝最新的消息来节省内存资源。

另外，如果达到了系统内存限制 （由于物理目标或代理范围限制设置不正确），代理将采取愈加严格的操作来防止内存过载。这些操作包括限制消息生成方。

确认这是否就是问题的起因

如果某个消息因为达到了配置的消息限制而被代理拒绝，代理将返回以下消息：

```
JMSException [C4036]：发生服务器错误
```

代理还在代理日志中记录以下条目：

```
警告 [B2011]：存储来自 IMQconn 的 JMS 消息失败
```

该消息后面接有一条表明已达到限制的消息。如果物理目标上存在消息限制，则代理将生成如下所示的条目：

```
[B4120]：无法在目标 destName 上存储消息，因为会超出 maxNumMsgs 的容量。
```

如果消息限制是代理范围的，则代理将生成如下所示的条目：

[B4024]：已经超出目前系统中的最大消息数，正在拒绝消息。

通常，您可以在发生拒绝之前按如下方式检查消息限制情况：

- 查询物理目标和代理，并检查其配置的消息限制设置。
- 使用相应的 imqcmd 命令，监视当前物理目标或代理 （作为一个整体）中的消息数或消息字节数。有关可以监视的度量以及用来获取它们的命令的信息，请参见第 18 章 “度量参考”。

解决此问题

有很多方法可以解决由于消息堆积而导致生成方变慢的问题。

- 修改物理目标上的 （或代理范围的）消息限制，请小心不要超出内存资源。

  通常，应该根据每个目标来管理内存，这样才不会达到代理范围的消息限制。有关更多信息，请参见第 208 页上的 “代理调整”。

- 将目标上的限制行为更改为当达到消息限制时并不减慢消息的生成，而是在内存中丢弃消息。

  例如，您可以指定 REMOVE_OLDEST 和 REMOVE_LOW_PRIORITY 限制行为，这些行为可以删除在内存中堆积的消息 （请参见第 283 页上的表 15-1）。

代理无法将持久性消息保存到数据存储中

如果代理无法访问数据存储或者无法将持久性消息写入数据存储，则生成方客户端将阻塞。如前所述，如果达到了目标或代理范围的消息限制，也会发生这种情况。

确认这是否就是问题的起因

如果代理无法写入数据存储，它将在代理日志中生成以下条目之一：
[B2011]：存储来自 connectionID 的 JMS 消息失败 . . . 或 [B4004]：无法持续消息 messageID. . .

解决此问题

- 如果是基于文件的持久性，则尝试增大基于文件的数据存储的磁盘空间。
- 如果是符合 JDBC 的数据存储，则检查基于 JDBC 的持久性是否正确配置 （请参见第 4 章 “配置代理”）。如果是这样，请向数据库管理员咨询，以解决其他数据库问题。

代理的确认超时太短

如果连接太慢或消息服务器反应迟缓 （由于 CPU 占用率太高或者内存资源不足导致），代理用于确认收到持久性消息的时间比连接工厂的 imqAckTimeout 属性值所允许的时间要长。

确认这是否就是问题的起因

如果超出了 imqAckTimeout 值，代理将返回以下消息：

```
JMSException [C4000]：包确认失败
```

解决此问题

更改 imqAckTimeout 连接工厂属性的值 （请参见第 154 页上的 “可靠性和流控制”）。

生成方客户端遇到了 JVM 限制

确认这是否就是问题的起因

- 查明客户端应用程序是否收到了 “内存不足”错误。
- 使用诸如 freeMemory、 MaxMemory 和 totalMemory 等运行时方法检查 JVM 堆中的可用内存。

解决此问题

调整 JVM （请参见第 204 页上的 “Java 虚拟机调整”）。

消息堆积

此问题的症状如下：

- 代理 （或特定目标）中的消息数或消息字节数随着时间的推移稳定增加。

  要查看消息是否在堆积，请检查代理中的消息数或消息字节数如何随时间的推移而改变，并与配置的限制进行比较。首先检查配置的限制：

  ```
  imqcmd query bkr
  ```

  （**注：**imqcmd metrics bkr 子命令不会显示此信息）。

  然后检查每个目标中的消息堆积情况。

  ```
  imqcmd list dst
  ```

  要检查消息是否已超出配置的目标或代理范围限制，请在代理日志中检查如下条目：警告 [B2011]：存储来自 ... 的 JMS 消息失败。此条目后面接有另一个说明已超出限制的条目。

- 消息的生成发生了延迟，或者生成的消息被代理拒绝。
- 消息到达使用方的时间过长。

本节探讨了如下可能的原因：

- 主题目标上有非活动的长期订阅。
- 队列中可以使用消息的使用方太少。
- 消息使用方的处理速度太慢，跟不上消息生成方的速度。
- 客户端确认处理减慢了消息的使用。
- 代理无法适应生成消息的速度。
- 客户端代码缺陷：使用方不确认消息。

主题目标上有非活动的长期订阅

如果长期订阅是非活动的，则消息会存储在目标中，直到相应的使用方变为活动状态且能够使用这些消息为止。

确认这是否就是问题的起因

检查每个主题目标上的长期订阅的状态：

```
imqcmd list dur -d destName
```

解决此问题

可以采用下列任意操作：

- 清除所有存在问题的长期订阅的消息 （请参见第 104 页上的 “管理长期订阅”）。
- 指定主题的消息限制以及限制行为属性 （请参见第 283 页上的表 15-1）。例如，您可以指定 REMOVE_OLDEST 和 REMOVE_LOW_PRIORITY 限制行为，这些行为删除堆积在内存中的消息。
- 清除来自相应目标的所有消息 （请参见第 115 页上的 “清除物理目标”）。
- 限制消息可以在内存中保留的时间。您可以重写生成方客户端以便对每个消息都设置一个生存时间值。您可以通过设置 imqOverrideJMSExpiration 和 imqJMSExpiration 连接工厂属性来覆盖共享一个连接的所有生成方的任何此类设置 （请参见第 293 页上的 “消息头覆盖”）。

队列中可以使用消息的使用方太少

如果消息可以传送到的活动使用方太少，队列目标可能会随着消息的堆积而堆满了待办事项。只要有下列任何原因，都会发生此情况：

- 目标的活动使用方太少。
- 使用方客户端建立连接失败。

- 活动使用方没有使用与队列中的消息匹配的选择器。

确认这是否就是问题的起因

要确定使用方不可用的原因，请检查目标上活动使用方的数目：

```
imqcmd metrics dst -n destName -t q -m con
```

解决此问题

您可以根据使用方不可用的原因采取下面相应的操作：

- 通过启动其他使用方客户端来为队列创建更多的活动使用方。
- 调整 imq.consumerFlowLimit 代理属性以优化向多个使用方的队列传送 （请参见第 208 页上的 “多使用方队列性能”）。
- 指定队列的消息限制以及限制行为属性 （请参见第 283 页上的表 15-1）。例如，您可以指定 REMOVE_OLDEST 和 REMOVE_LOW_PRIORITY 限制行为，这些行为删除堆积在内存中的消息。
- 清除来自相应目标的所有消息 （请参见第 115 页上的 “清除物理目标”）。
- 限制消息可以在内存中保留的时间。您可以重写生成方客户端以对每个消息都设置一个生存时间值，也可以通过设置 imqOverrideJMSExpiration 和 imqJMSExpiration 连接工厂属性来覆盖共享一个连接的所有生成方的任何此类设置 （请参见第 293 页上的 “消息头覆盖”）。

消息使用方的处理速度太慢，跟不上消息生成方的速度

在这种情况下，主题的订阅者或队列的接收者使用消息的速度要比生成方发送消息的速度慢。有一个或多个目标因为这种不平衡而堆满了消息。

确认这是否就是问题的起因

检查消息流入和流出代理的速率：

```
imqcmd metrics bkr -m rts
```

然后检查每个单独目标的流速：

```
imqcmd metrics bkr -t destType -n destName -m rts
```

解决此问题

- 优化使用方客户端代码。
- 对于队列目标，增大活动使用方的数目 （请参见第 208 页上的 “多使用方队列性能”）。

客户端确认处理减慢了消息的使用

有两个因素影响客户端确认处理：

- 在处理客户端确认的过程中会消耗大量的代理资源。因此，如果使用方客户端会一直阻塞到代理对客户端确认进行确认时为止，则对于这样的确认模式，消息的使用会变慢。
- JMS 有效负荷消息和 Message Queue 控制消息 （例如客户端确认）共享同一连接。因此，控制消息可能会被 JMS 有效负荷消息阻挡，从而使消息的使用变慢。

确认这是否就是问题的起因

- 检查与包流相关的消息流。如果每秒包数与消息数不成比例，则客户端确认可能有问题。
- 检查客户端是否收到以下消息：

  `JMSException [C4000]`：包确认失败

解决此问题

- 修改客户端使用的确认模式，例如切换到 `DUPS_OK_ACKNOWLEDGE` 或 `CLIENT_ACKNOWLEDGE`。
- 如果使用 `CLIENT_ACKNOWLEDGE` 或事务会话，则将更多数目的消息组合到一个确认中。
- 调整使用方和连接流控制参数 （请参见第 209 页上的 “客户端运行时消息流调整”）。

代理无法适应生成消息的速度

在这种情况下，消息流入代理的速度比代理可以将它们路由并发送到使用方的速度要快。代理的迟缓可能由下列任一或全部限制所导致：CPU、网络套接字读 / 写操作、磁盘读 / 写操作、内存分页、持久性存储或 JVM 内存限制。

确认这是否就是问题的起因

检查有无其他原因导致此问题。

解决此问题

- 升级计算机或数据存储的速度。
- 使用代理群集在多个代理实例之间分布负载。

客户端代码缺陷：使用方不确认消息

消息会保留在目标中，直到消息所发送到的所有使用方都进行了确认为止。如果客户端没有确认已使用消息，则该消息会在目标中堆积，而不会被删除。

例如，客户端代码可能存在以下缺陷：

- 使用 CLIENT_ACKNOWLEDGEacknowledgment 或事务会话的使用方可能没有定期调用 Session.acknowledge 或 Session.commit。
- 使用 AUTO_ACKNOWLEDGE 会话的使用方可能因为某种原因而挂起。

确认这是否就是问题的起因

首先检查本节中列出的其他所有可能的原因。其次，使用以下命令列出目标：

```
imqcmd list dst
```

请注意 UnAcked 标题下列出的消息数目是否与目标中的消息数目相同。 UnAcked 标题下的消息已发送到使用方但未得到确认。如果此数目与消息总数相同，则说明代理已发送所有消息，正在等待确认。

解决此问题

请求应用程序开发者帮助调试此问题。

消息服务器吞吐量呈间歇性

此问题的症状如下：

- 消息的吞吐量间歇性地下降，然后又恢复正常性能。

本节探讨了如下可能的原因：

- 代理的内存资源非常低。
- 正在发生 JVM 内存回收 （垃圾收集）。
- JVM 使用实时编译器来提高性能。

代理的内存资源非常低

由于目标和代理限制设置不当，代理采取了越来越严格的措施以防止内存过载，这样就导致代理变得非常迟缓，直到堆积的消息得到清除为止。

确认这是否就是问题的起因

在代理日志中检查内存低的情况 （[B1089]：内存不足，代理正在尝试释放资源），该情况后面会接有一个描述新内存状态和已用内存总量的条目。

另外请检查 JVM 堆中的可用内存：

```
imqcmd metrics bkr -m cxn
```

当总 JVM 内存接近 JVM 内存最大值时，可用内存就会很低。

解决此问题

- 调整 JVM （请参见第 204 页上的 “Java 虚拟机调整”）。
- 增大系统交换空间。

正在发生 JVM 内存回收 （垃圾收集）

内存回收会定期清扫整个系统，以释放内存。发生此操作时，所有的线程都会阻塞。要释放的内存量以及 JVM 堆的大小越大，因内存回收而导致的延迟就越长。

确认这是否就是问题的起因

监视计算机上的 CPU 使用率。发生内存回收时， CPU 使用率会下降。

另外，使用以下命令行选项启动代理：

```
-vmargs -verbose:gc
```

其标准输出指明发生内存回收的时间。

解决此问题

在多个 CPU 的计算机中，将内存回收设置为并行发生：

```
-XX:+UseParallelGC=true
```

JVM 使用实时编译器来提高性能

确认这是否就是问题的起因

检查有无其他原因导致此问题。

解决此问题

让系统运行一段时间，性能应该会有所改善。

消息无法到达使用方

此问题的症状如下：

- 使用方未收到生成方发送的消息。

本节探讨了如下可能的原因：

- 限制行为导致消息在代理上被删除。
- 消息超时值即将到期。

- 时钟不同步。
- 使用方客户端未能在某个连接上启动消息传送。

限制行为导致消息在代理上被删除

如果目标内存中的消息数或消息字节数达到了配置限制，代理将尝试节省内存资源。当达到限制时，代理将采取下列三个可配置的行为，从而导致消息丢失：

- REMOVE_OLDEST：删除最旧的消息。
- REMOVE_LOW_PRIORITY：根据消息存在的时间删除优先级最低的消息。
- REJECT_NEWEST: 拒绝新的持久性消息。

如果代理内存中的消息数或消息字节数达到配置的限制，代理将尝试通过拒绝最新的消息来节省内存资源。

确认这是否就是问题的起因

检查停用消息队列，如第 232 页上的 “停用消息队列包含消息”中所述。具体地说，是使用第 233 页上的 “消息的数目或者其大小超出目标限制”中的说明。查找 REMOVE_OLDEST 或 REMOVE_LOW_PRIORITY 原因。

解决此问题

增加目标限制。例如：

```
imqcmd update dst -n MyDest -o maxNumMsgs=1000
```

消息超时值即将到期

代理将删除超时值已过期的消息。如果目标上完全堆满了消息，生存时间值过短的消息将被删除。

确认这是否就是问题的起因

检查停用消息队列以查看消息是否超时。

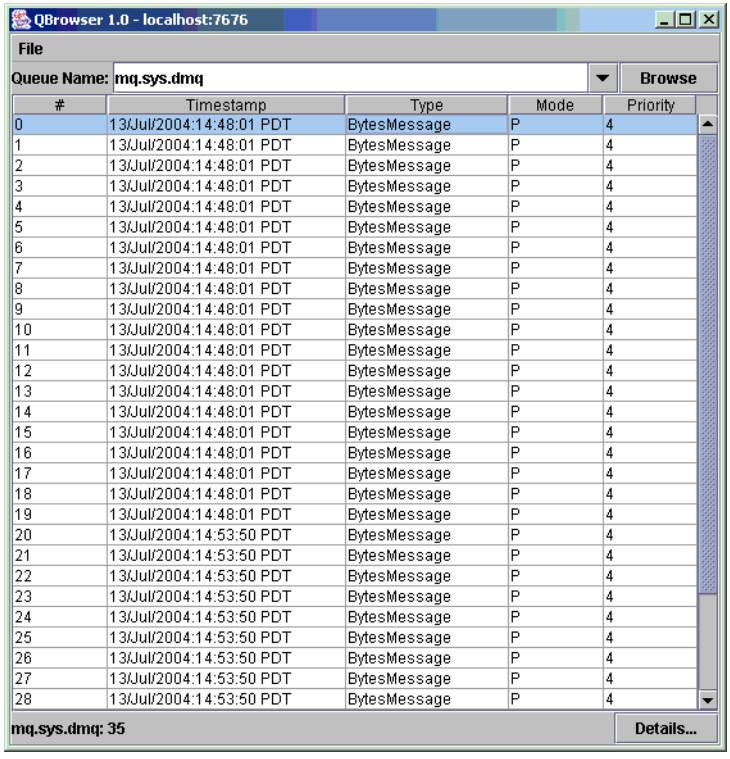
使用 QBrowser 演示应用程序来查看 DMQ 内容。QBrowser 演示程序保存在特定于操作系统的位置；要了解该位置，请参见附录 A “Message Queue 数据在特定平台上的位置”并查看 “示例应用程序和位置”表。

下面是 Windows 中的一个调用示例：

```
cd \MessageQueue3\demo\applications\qbrowser java QBrowser
```

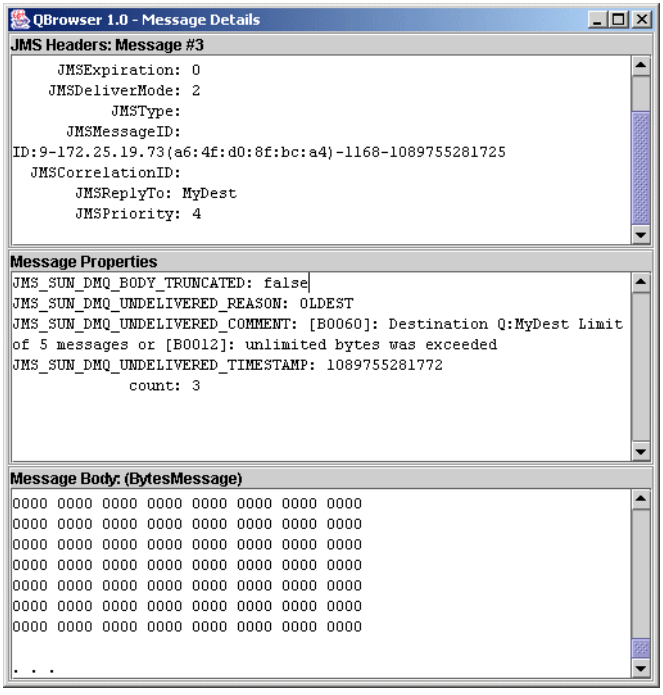
QBrowser 主窗口出现后，选择队列名称 mq.sys.dmq，然后单击 “浏览”。将出现如下所示的列表。

图 12-1 QBrowser 窗口



双击消息可显示该消息的详细信息。

图 12-2 QBrowser 消息的详细信息



请注意消息的 JMS_SUN_DMQ_UNDELIVERED_REASON 属性的值是否为 EXPIRED。

解决此问题

联系应用程序开发者，请他们提高生存时间值。

时钟不同步

如果时钟之间不同步，则代理对消息生命周期的计算可能有错误，从而导致消息超过它们的到期时间而被删除。

确认这是否就是问题的起因

在代理日志文件中，查找下列任一消息：B2102、 B2103、 B2104。这些消息均报告检测到可能的时钟脉冲相位差。

解决此问题

检查您是否正在运行时间同步程序，如第 63 页上的 “准备系统资源”中所述。

使用方客户端未能在某个连接上启动消息传送

除非客户端代码建立了连接，并在该连接上启动了消息传送，否则消息将无法传送。

确认这是否就是问题的起因

检查客户端代码是否能建立连接并启动消息传送。

解决此问题

重写客户端代码，以建立连接并启动消息传送。

停用消息队列包含消息

此问题的症状如下：

- 列出目标后，发现停用消息队列包含消息。例如，执行如下所示的命令：

  ```
  imqcmd list dst
  ```

  在提供用户名和密码后，将显示类似以下内容的输出：

```
列出指定的代理上的所有目的地：
----------------------------------
主机          主端口
----------------------------------
localhost    7676
--------------------------------------------------------------------------
   名称       类型    状态      生成方    使用方   消息总数计数   未确认  平均大小
--------------------------------------------------------------------------

MyDest      Queue  RUNNING   0       0          5          0     1177.0
mq.sys.dmq  Queue  RUNNING   0       0          35         0     1422.0
成功列出目标。
```

  在本示例中，停用消息队列 mq.sys.dmq 包含 35 条消息。

本节探讨了如下可能的原因：

- 消息的数目或者其大小超出目标限制。
- 代理时钟和生成方时钟不同步。
- 消息超时前，使用方未接收到消息。
- 相对使用方数目而言，生成方太多。
- 生成方比使用方的速度快。

- 使用方太慢。
- 客户端不提交消息。
- 长期使用方处于非活动状态。
- 发生意外的代理错误。

消息的数目或者其大小超出目标限制

确认这是否就是问题的起因

使用 QBrowser 演示应用程序来查看停用消息队列的内容。 QBrowser 演示程序位于操作系统特定的位置；要了解该位置，请参见附录 A “Message Queue 数据在特定平台上的位置”并查看 “示例应用程序和位置”表。

下面是 Windows 中的一个调用示例：

```
cd \MessageQueue3\demo\applications\qbrowser java QBrowser
```

QBrowser 主窗口出现后，选择队列名称 mq.sys.dmq，然后单击 “浏览”。将会出现如第 230 页上的图 12-1 所示的列表。

双击消息可显示该消息的详细信息。将会出现如第 231 页上的图 12-2 所示的窗口。

请注意下列消息属性的值：

- JMS_SUN_DMQ_UNDELIVERED_REASON
- JMS_SUN_DMQ_UNDELIVERED_COMMENT
- JMS_SUN_DMQ_UNDELIVERED_TIMESTAMP

请注意 JMS 标题下面的 JMSDestination 值，以确定消息将停用的目标。

解决此问题

增加目标限制。例如：

```
imqcmd update dst -n MyDest -o maxNumMsgs=1000
```

代理时钟和生成方时钟不同步

确认这是否就是问题的起因：

使用 QBrowser 应用程序来查看停用消息队列中各消息的详细信息。检查 JMS_SUN_DMQ_UNDELIVERED_REASON 的值，查找原因为 EXPIRED 的消息。

在代理日志文件中，查找下列任一消息：B2102、 B2103、 B2104。这些消息均报告检测到可能的时钟脉冲相位差。

解决此问题

检查您是否正在运行时间同步程序，如第 63 页上的 “准备系统资源”中所述。

消息超时前，使用方未接收到消息

确认这是否就是问题的起因

使用 QBrowser 应用程序来查看停用消息队列中各消息的详细信息。检查 JMS_SUN_DMQ_UNDELIVERED_REASON 的值，查找原因为 EXPIRED 的消息。

检查目标中是否有任何使用方。例如：

```
imqcmd query dst -t q -n MyDest
```

检查列出的 “当前活动使用方数”值。如果有活动使用方，则下面的某一项为真：

- 使用方的连接暂停。
- 相对使用方执行速度而言，消息超时时间太短。

解决此问题

请求应用程序开发者提高消息的生存时间值。

相对使用方数目而言，生成方太多

确认这是否就是问题的起因

使用 QBrowser 应用程序来查看停用消息队列中各消息的详细信息。检查 JMS_SUN_DMQ_UNDELIVERED_REASON 的值。

如果原因是 REMOVE_OLDEST 或 REMOVE_LOW_PRIORITY，请使用 imqcmd query dst 命令来检查目标中的生成方和使用方的数目。如果生成方的数目超过使用方的数目，则生成率可能会远远超出使用率。

解决此问题

添加更多的使用方客户端，或者将目标设置为使用 FLOW_CONTROL 限制行为。FLOW_CONTROL 限制行为通过使用率来控制生成率。

使用如下例所示的命令来启动流控制行为：

```
imqcmd update dst -n myDst -t q -o consumerFlowLimit=FLOW_CONTROL
```

生成方比使用方的速度快

确认这是否就是问题的起因

要确定较慢的使用方是否会导致生成方速度降低，请将目标限制行为设置为 FLOW_CONTROL。 FLOW_CONTROL 限制行为通过使用率来控制生成率。

使用如下例所示的命令来启动流控制行为：

```
imqcmd update dst -n myDst -t q -o consumerFlowLimit=FLOW_CONTROL
```

通过执行如下例所示的命令，使用度量来检查目标的输入和输出：

```
imqcmd metrics dst -n myDst -t q -m rts
```

在度量输出中，检查以下值：

- `Msgs/sec Out`

  该值显示代理每秒删除多少条消息。在所有使用方确认收到消息后，代理将删除这些消息，因此，该度量反映了使用率。

- `Msgs/sec In`

  此值显示代理每秒从生成方接收多少条消息。此度量反映了生成率。

由于流控制使生成与使用协调一致，因此请注意生成是否减慢或停止。如果速率减慢或停止，则说明生成方和使用方的处理速度不一致。

也可以使用 `imqcmd list dst` 命令，检查未确认的 (UnAcked) 发送消息的数目。如果未确认的消息数目小于目标大小，则表明目标尚有额外的容量，它受到客户端流控制的抑制。

解决此问题

如果生成率始终高于使用率，可以考虑有规律地使用流控制以使系统保持协调一致。

此外，使用后面各节的内容，考虑并尝试消除下列可能的因素：

- 使用方太慢。
- 客户端不提交消息。
- 使用方未能确认消息。
- 长期使用方处于非活动状态。
- 发生意外的代理错误。

使用方太慢

确认这是否就是问题的起因

使用度量来确定生成和使用的速率，如第 234 页上的 “生成方比使用方的速度快”中所述。

解决此问题

尝试下列一项或多项：

- 将目标设置为使用 FLOW_CONTROL 限制行为。使用如下所示的命令：

  ```
  imqcmd update dst -n myDst -t q -o consumerFlowLimit=FLOW_CONTROL
  ```

  使用流控制将生成率降低到使用率，防止消息在代理中堆积。生成方应用程序保留消息，直到目标可以及时处理它们，从而降低过期风险。

- 向应用程序开发者了解生成方是以稳定的速率发送消息，还是周期性成批发送。

  如果应用程序发送成批消息，请按照下一项中的说明增加目标限制。

- 根据消息数和 / 或字节数增加目标限制。

  要更改目标中的消息数，请输入如下格式的命令：

  ```
  imqcmd update dst -n destName -t {q/t} -o maxNumMsgs=number
  ```

  要更改目标的大小，请输入如下格式的命令：

  ```
  imqcmd update dst -n destName -t {q/t} -o maxTotalMsgBytes=number
  ```

  请注意，增加限制会增加代理使用的内存数。如果限制过高，代理可能会耗尽内存，从而无法处理消息。

- 考虑您是否可以在高级别负载生成期间接受消息丢失。

客户端不提交消息

确认这是否就是问题的起因

与应用程序开发者进行确认以查明应用程序是否使用事务。如果应用程序使用事务，则按如下所示列出活动事务：

```
imqcmd list txn
```

下面是一个命令输出示例：

```
-------------------------------------------------------------------------
事务 ID               状态     用户名   # Msgs/# Acks   创建时间
-------------------------------------------------------------------------
6800151593984248832  STARTED  guest       3/2         7/19/04 11:03:08 AM
```

请注意消息的数目和确认的数目。

如果消息的数目很高，则生成方可能正在发送个别的消息，而未能提交事务。代理在收到提交之前，无法路由并传送该事务的消息。

如果确认的数目很高，则使用方可能正在发送个别消息的确认，而未能提交事务。代理在收到提交之前，无法删除该事务的确认。

解决此问题

联系应用程序开发者以修复编码错误。

使用方未能确认消息

确认这是否就是问题的起因

联系应用程序开发者以确定应用程序使用的是基于系统的确认还是基于客户端的确认。如果应用程序使用基于系统的确认，则跳过本节。

如果应用程序使用基于客户端的确认 （CLIENT_ACKNOWLEDGE 类型），请先减少客户端中存储的消息数。使用如下所示的命令：

```
imqcmd update dst -n myDst -t q -o consumerFlowLimit=1
```

其次，确定是因为代理由于使用方速度慢而缓冲消息，还是因为使用方处理消息的速度很快但未对其进行确认。

使用以下命令列出目标：

```
imqcmd list dst
```

在提供用户名和密码后，将显示类似以下内容的输出：

```
列出指定的代理上的所有目的地：
----------------------------------
主机          主端口
----------------------------------
localhost    7676
-------------------------------------------------------------------------
 名称        类型      状态      生成方   使用方  消息总数计数  未确认  平均大小
-------------------------------------------------------------------------
MyDest       Queue    RUNNING    0       0        5         200     1177.0
mq.sys.dmq   Queue    RUNNING    0       0        35         0      1422.0
成功列出目标。
```

未确认数值表示代理已发送且正在等待确认的消息数。如果未确认数值较高或不断增加，则说明代理正在发送消息，因此不等待速度较慢的使用方。还说明使用方未确认消息。

解决此问题

联系应用程序开发者以修复编码错误。

长期使用方处于非活动状态

确认这是否就是问题的起因

使用以下命令格式查看主题的长期订户：

```
imqcmd list dur -d topicName
```

解决此问题

- 使用 `imqcmd purge dur` 命令清除长期使用方。
- 重新启动使用方应用程序。

发生意外的代理错误

确认这是否就是问题的起因

使用 QBrowser 对消息进行检查，如第 234 页上的 “生成方比使用方的速度快”中所述。

如果 `JMS_SUN_DMQ_UNDELIVERED_REASON` 的值是 `ERROR`，则说明代理发生错误。

解决此问题

- 检查代理日志文件以查找相关错误。
- 联系 Sun 技术支持以报告代理问题。

第 III 部分

参考

第 13 章

命令行参考

本章提供了有关使用 Message Queue 命令行管理实用程序的参考信息。本章包含以下各节：

命令行语法

Message Queue 命令行实用程序是 shell 命令。实用程序的名称是命令，其子命令或选项是传递给该命令的参数。不需要使用单独的命令来启动或退出实用程序。

以下命令语法适用于所有命令行实用程序：

utilityName [*subcommand*] [*commandArgument*] [[*-optionName* [*-optionArgument*]]...]

其中 *utilityName* 是以下实用程序之一：

- `imqbrokerd` （代理实用程序）
- `imqcmd` （命令实用程序）
- `imqobjmgr` （对象管理器实用程序）

- imqdbmgr （数据库管理器实用程序）
- imqusermgr （用户管理器实用程序）
- imqsvcadmin （服务管理器实用程序）
- imqkeytool （密钥工具实用程序）

子命令和命令级别的参数 （如果有）必须放在所有选项及其参数之前；而选项自身可以按任何顺序显示。所有子命令、命令参数、选项和选项参数都是以空格分隔的。如果选项参数的值包含空格，则必须将整个值置于引号中。（将所有属性 - 值对置于引号中通常是最安全的做法。）

以下命令用于启动默认代理，它是不带子命令子句的命令行的示例：

```
imqbrokerd
```

下面是更完整的示例：

```
imqcmd destroy dst -t q -n myQueue -u admin -f -s
```

此命令销毁名为 myQueue 的队列目标 （目标类型为 q）。验证是针对用户名 admin 执行的；此命令将提示用户输入密码。执行此命令时不提示用户进行确认 （-f 选项），并且处于无提示模式，而不显示任何输出 （-s 选项）。

代理实用程序

代理实用程序 (imqbrokerd) 用于启动代理。命令行选项将覆盖代理配置文件中的值，但仅对当前代理会话有效。

表 13-1 显示了 imqbrokerd 命令的选项以及被每个选项覆盖的配置属性 （如果有）。

表 13-1 代理实用程序选项

选项	被覆盖的属性	描述
-name *instanceName*	imq.instancename	**代理的实例名称** **在同一主机上运行的多个代理实例必须具有不同的实例名称。** **默认值：** imqbroker

表 13-1 代理实用程序选项 （续）

选项	被覆盖的属性	描述
-port *portNumber*	imq.portmapper.port	代理端口映射器的端口号 Message Queue 客户端使用此端口号连接到代理。在同一主机上运行的多个代理实例必须具有不同的端口映射器端口号。 默认值：7676
-cluster *broker1* [[,*broker2*]...]	imq.cluster.brokerlist	将代理连接到群集[1] 将指定的代理与 imq.cluster.brokerlist 属性中的列表合并。每个代理参数具有以下形式之一： *hostName*:*portNumber* *hostName* :*portNumber* 如果省略 *hostName*，则默认值为 localhost ； 如果省略 *portNumber*，则默认值为 7676。
-D*property*=*value*	实例配置文件中对应的属性	设置配置属性 有关代理配置属性的信息，请参见第 14 章 “代理属性参考”。 **注意：**请仔细检查使用此选项设置的属性的拼写和格式。不正确的值将被忽略，且系统不会给出任何通知或警告。
-reset props	无	重置配置属性 用空文件替换代理的现有实例配置文件 (config.properties) ； 所有属性都采用默认值。
-reset store	无	重置持久性数据存储 从数据存储中清除所有持久性数据 （包括持久性消息、长期订阅和事务信息），以便您可以启动一个无任何数据记录的代理实例。为了防止在以后重新启动时重置持久性存储，请不要使用 -reset 选项重新启动代理实例。 如果仅清除持久性消息或长期订阅，请改用 -reset messages 或 -reset durables。
-reset messages	无	从数据存储中清除持久性消息
-reset durables	无	从数据存储中清除长期订阅
-backup *fileName*	无	将配置更改记录备份到文件中[1] 有关更多信息，请参见第 170 页上的 “管理配置更改记录”。

表 13-1 代理实用程序选项 （续）

选项	被覆盖的属性	描述
`-restore` *fileName*	无	从备份文件中恢复配置更改记录[1] 该备份文件必须是以前使用 `-backup` 选项创建的。 有关更多信息，请参见第 170 页上的 “管理配置更改记录”。
`-remove instance`	无	删除代理实例[2] 删除与实例关联的实例配置文件、日志文件、持久性存储以及其他文件和目录。
`-password` *keyPassword*	`imq.keystore.password`	SSL 证书密钥库的密码[3]
`-dbuser` *userName*	`imq.persist.jdbc.user`	基于 JDBC 的持久性数据存储的用户名
`-dbpassword` *dbPassword*	`imq.persist.jdbc.password`	基于 JDBC 的持久性数据存储的密码[3]
`-ldappassword` *ldapPassword*	`imq.user_repository.ldap.password`	LDAP 用户系统信息库的密码[3]

`-passfile` *filePath*	`imq.passfile.enabled`
	`imq.passfile.dirpath`
	`imq.passfile.name`
		密码文件的位置
		将代理的 `imq.passfile.enabled` 属性设置为 `true`，将 `imq.passfile.dirpath` 属性设置为包含密码文件的路径，将 `imq.passfile.name` 属性设置为文件名自身。
		有关更多信息，请参见第 144 页上的 “使用密码文件”。

`-shared`	`imq.jms.threadpool_model`	使用共享线程池模型实现 `jms` 连接服务 将在连接之间共享执行线程，以增加支持的连接数。 将代理的 `imq.jms.threadpool_model` 属性设置为 `shared`。
`-javahome` *path*	无	可选 Java 运行时的位置 默认值：使用系统上安装的运行时或 Message Queue 附带的运行时。
`-vmargs` *arg1* [[*arg2*] ...]	无	将参数传递给 Java 虚拟机 参数是以空格分隔的。要传递多个参数或包含空格的参数，请将参数列表置于引号中。 只能从命令行中传递 VM 参数；在实例配置文件中没有关联的配置属性。

表 13-1 代理实用程序选项 （续）

选项	被覆盖的属性	描述
-license [*licenseName*]	无	要装入的许可证 （如果与所安装的 Message Queue 产品版本的默认许可证不同）： pe 具有基本功能的 Platform Edition try 具有企业功能的 Platform Edition （90 天试用版） unl Enterprise Edition 如果未指定许可证名称，此选项将列出系统上安装的所有许可证。
-upgrade-store-nobackup	无	从不兼容版本升级到 Message Queue 3.5 或 3.5 SP*x* 时自动删除旧的数据存储[2] 有关更多信息，请参见 Message Queue Installation Guide。
-force	无	执行操作时无需用户确认 此选项仅适用于 -remove instance 和 -upgrade-store-nobackup 选项，这两个选项通常要求确认。
-loglevel *level*	imq.broker.log.level	日志记录级别： NONE ERROR WARNING INFO 默认值：INFO
-metrics *interval*	imq.metrics.interval	代理度量的日志记录时间间隔 （以秒为单位）
-tty	imq.log.console.output	将所有消息记录到控制台 将代理的 imq.log.console.output 属性设置为 ALL。 如果未指定，将只记录错误和警告消息。
-s | -silent	imq.log.console.output	无提示模式 （不向控制台记录消息） 将代理的 imq.log.console.output 属性设置为 NONE。
-version	无	显示版本信息[4]
-h | -help	无	显示使用帮助[4]

1. 此选项仅适用于代理群集。

2. 除非还指定了 -force，否则此选项将要求用户进行确认。

3. 不建议使用此选项，它最终将被删除。应完全省略密码（这样，命令将以交互方式提示用户输入密码）或使用 -passfile 选项来指定包含密码的密码文件。
4. 忽略命令行中指定的任何其他选项。

命令实用程序

命令实用程序 (imqcmd) 用于管理代理、连接服务、连接、物理目标、长期订阅和事务。

所有 imqcmd 命令都必须包含一个子命令 （使用 -v 或 -h 选项显示产品版本信息或使用帮助的命令除外）。下面列出了一些可用的子命令，并在后面的相应章节中进行了详细介绍。在任何情况下，如果子命令接受代理地址 （-b 选项），并且未指定主机名或端口号，则默认情况下都将采用值 localhost 和 7676。

代理管理	
shutdown bkr	关闭代理
restart bkr	重新启动代理
pause bkr	暂停代理
resume bkr	恢复代理
update bkr	设置代理属性
reload cls	重新装入群集配置
query bkr	列出代理属性值
metrics bkr	显示代理度量
连接服务管理	
pause svc	暂停连接服务
resume svc	恢复连接服务
update svc	设置连接服务属性
list svc	列出代理中可用的连接服务
query svc	列出连接服务属性值
metrics svc	显示连接服务度量
连接管理	
list cxn	列出代理中的连接
query cxn	显示连接信息

物理目标管理	
create dst	创建物理目标
destroy dst	销毁物理目标
pause dst	暂停物理目标的消息传送
resume dst	恢复物理目标的消息传送
update dst	设置物理目标属性
purge dst	清除来自物理目标的所有消息
compact dst	压缩物理目标
list dst	列出物理目标
query dst	列出物理目标属性值
metrics dst	显示物理目标度量
长期订阅管理	
destroy dur	销毁长期订阅
purge dur	清除来自长期订阅的所有消息
list dur	列出主题的长期订阅
事务管理	
commit txn	提交事务
rollback txn	回滚事务
list txn	列出代理正在跟踪的事务
query txn	显示事务信息

代理管理

无法使用命令实用程序启动代理，而应使用代理实用程序 (imqbrokerd)。代理启动后，您可以使用表 13-2 中列出的 imqcmd 子命令来管理和控制代理。

表 13-2 用于代理管理的命令实用程序子命令

语法	描述
shutdown bkr [-b *hostName*:*portNumber*]	关闭代理
restart bkr [-b *hostName*:*portNumber*]	重新启动代理 关闭代理，再使用最初启动代理时指定的选项重新启动代理。

表 13-2 用于代理管理的命令实用程序子命令 （续）

语法	描述
`pause bkr` [`-b` *hostName*`:`*portNumber*]	暂停代理 有关更多信息，请参见第 96 页上的 “暂停代理”。
`resume bkr` [`-b` *hostName*`:`*portNumber*]	恢复代理
`update bkr` [`-b` *hostName*`:`*portNumber*] `-o` *property1=value1* [[`-o` *property2=value2*] ...]	设置代理属性 有关代理属性的信息，请参见第 14 章 “代理属性参考”。
`reload cls`	重新装入群集配置[1] 将所有持久性信息强制更新为最新状态。
`query bkr -b` *hostName*`:`*portNumber*	列出代理属性值 还列出连接到群集中指定代理的所有正在运行的代理。
`metrics bkr` [`-b` *hostName*`:`*portNumber*] [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]	显示代理度量 `-m` 选项指定要显示的度量的类型： `ttl` 流入和流出代理的消息和包 `rts` 消息和包流入流出代理的速率 （以一秒为衡量单位） `cxn` 连接、虚拟内存堆和线程 默认值：`ttl`。 `-int` 选项指定显示度量的时间间隔 （以秒为单位）。默认值：`5`。 `-msp` 选项指定要显示的样例的数量。默认值：无限制（无穷多）。

1. 此选项仅适用于代理群集。

连接服务管理

表 13-3 列出了用于管理连接服务的 `imqcmd` 子命令。

表 13-3 用于连接服务管理的命令实用程序子命令

语法	描述
`pause svc -n` *serviceName* [`-b` *hostName*`:`*portNumber*]	暂停连接服务 无法暂停 `admin` 连接服务。

表 13-3 用于连接服务管理的命令实用程序子命令 （续）

语法	描述
`resume svc -n` *serviceName* [`-b` *hostName*`:`*portNumber*]	恢复连接服务
`update svc -n` *serviceName* [`-b` *hostName*`:`*portNumber*] `-o` *property1=value1* [[`-o` *property2=value2*] ...]	设置连接服务属性 有关连接服务属性的信息，请参见第 259 页上的 “连接属性”。
`list svc` [`-b` *hostName*`:`*portNumber*]	列出代理中可用的连接服务
`query svc -n` *serviceName* [`-b` *hostName*`:`*portNumber*]	列出连接服务属性值
`metrics svc -n` *serviceName* [`-b` *hostName*`:`*portNumber*] [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]	显示连接服务度量 `-m` 选项指定要显示的度量的类型： `ttl` 通过指定的连接服务流入和流出代理的消息和包 `rts` 消息和包通过指定的连接服务流入流出代理的速率 （以一秒为衡量单位） `cxn` 连接、虚拟内存堆和线程 默认值：`ttl`。 `-int` 选项指定显示度量的时间间隔 （以秒为单位）。默认值：`5`。 `-msp` 选项指定要显示的样例的数量。默认值：无限制（无穷多）。

连接管理

表 13-4 列出了用于管理连接的 `imqcmd` 子命令。

表 13-4 用于连接服务管理的命令实用程序子命令

语法	描述
`list cxn` [`-svn` *serviceName*] [`-b` *hostName*`:`*portNumber*]	列出代理中的连接 列出代理到指定连接服务的所有连接。如果未指定连接服务，则列出所有连接。
`query cxn -n` *connectionID* [`-b` *hostName*`:`*portNumber*]	显示连接信息

物理目标管理

表 13-5 列出了用于管理物理目标的 imqcmd 子命令。在任何情况下， -t （目标类型）选项都可以采用以下两个值之一：

q 队列目标
t 主题目标

表 13-5 用于物理目标管理的命令实用程序子命令

语法	描述
create dst -t *destType* -n *destName* [-o *property1=value1*] [[-o *property2=value2*] ...]	创建物理目标[1] 目标名称 *destName* 只能包含字母数字字符 （不包括空格），并且必须以字母字符开头，或者以下划线 (_) 或美元符号 (**$**) 开头。目标名称不能以字符 mq 开头。
destroy dst -t *destType* -n *destName*	销毁物理目标[1] 此操作不能应用于系统创建的目标，如停用消息队列。
pause dst [-t *destType* -n *destName*] [-pst *pauseType*]	暂停物理目标的消息传送 暂停由 -t 和 -n 选项指定的物理目标的消息传送。如果未指定这些选项，将暂停所有目标。 pst 选项指定要暂停的消息传送的类型： CONSUMERS 暂停向消息使用方传送消息 PRODUCERS 暂停向消息生成方传送消息 ALL 暂停所有消息传送 默认值：ALL
resume dst [-t *destType* -n *destName*]	恢复物理目标的消息传送 恢复由 -t 和 -n 选项指定的物理目标的消息传送。如果未指定这些选项，将恢复所有目标。
update dst -t *destType* -n *destName* -o *property1=value1* [[-o *property2=value2*] ...]	设置物理目标属性 有关物理目标属性的信息，请参见第 15 章 “物理目标属性参考”。
purge dst -t *destType* -n *destName*	清除来自物理目标的所有消息
compact dst [-t *destType* -n *destName*]	压缩物理目标 压缩物理目标 （由 -t 和 -n 选项指定）的基于文件的持久性数据存储。如果未指定这些选项，将压缩所有目标。 压缩之前必须先暂停目标。

表 13-5 用于物理目标管理的命令实用程序子命令 （续）

语法	描述
`list dst` [`-t` *destType*] [`-tmp`]	列出物理目标 列出某个类型的所有物理目标，该类型由 `-t` 选项指定。如果未指定任何目标类型，则同时列出队列目标和主题目标。如果指定了 `-tmp` 选项，还会列出临时目标。
`query dst -t` *destType* `-n` *destName*	列出物理目标属性值
`metrics dst -t` *destType* `-n` *destName* [`-m` *metricType*] [`-int` *interval*] [`-msp` *numSamples*]	显示物理目标度量 `-m` 选项指定要显示的度量的类型： `ttl` 流入和流出目标以及驻留在内存中的消息和包 `rts` 消息和包流入和流出代理的速率 （以秒为衡量单位），以及其他速率信息 `con` 与消息使用方相关的度量 `dsk` 磁盘使用情况 默认值：`ttl`。 `-int` 选项指定显示度量的时间间隔 （以秒为单位）。默认值：`5`。 `-msp` 选项指定要显示的样例的数量。默认值：无限制（无穷多）。

1. 不能在主代理暂时不可用的代理群集中执行此操作。

长期订阅管理

表 13-6 列出了用于管理长期订阅的 `imqcmd` 子命令。

表 13-6 用于长期订阅管理的命令实用程序子命令

语法	描述
`destroy dur -c` *clientID* `-n` *subscriberName*	销毁长期订阅[1]
`purge dur -c` *clientID* `-n` *subscriberName*	清除来自长期订阅的所有消息
`list dur -d` *topicName*	列出主题的长期订阅

1. 不能在主代理暂时不可用的代理群集中执行此操作。

事务管理

表 13-7 列出了用于管理事务的 imqcmd 子命令。

表 13-7 用于事务管理的命令实用程序子命令

语法	描述
`commit txn -n` *transactionID*	提交事务
`rollback txn -n` *transactionID*	回滚事务
`list txn`	列出代理正在跟踪的事务
`query txn -n` *transactionID*	显示事务信息

常规命令实用程序选项

表 13-8 中列出的附加选项适用于 imqcmd 命令的所有子命令。

表 13-8 常规命令实用程序选项

选项	描述
`-secure`	通过 ssladmin 连接服务使用安全的代理连接
`-u` *userName*	用于验证的用户名 如果省略此选项，命令实用程序将以交互方式提示用户输入用户名。
`-p` *password*	用于验证的密码[1]
`-passfile` *path*	密码文件的位置 有关更多信息，请参见第 144 页上的 “使用密码文件”。
`-rtm` *timeoutInterval*	初始超时时间间隔 （以秒为单位） 初始超时时间间隔是命令实用程序最初等待代理回复的时间长度，超过此时间后，将重试请求。随后的每次重试都将使用此初始时间间隔的倍数作为超时时间间隔。 默认值：-10。
`-rtr` *numRetries*	代理请求超时后尝试重试的次数 默认值：5。
`-javahome` *path*	可选 Java 运行时的位置 默认值：使用系统上安装的运行时或 Message Queue 附带的运行时。
`-f`	执行操作时无需用户确认
`-s`	无提示模式 （不显示输出）

表 13-8 常规命令实用程序选项 （续）

选项	描述
-v	显示版本信息[2,3]
-h	显示使用帮助[2,3]
-H	显示详细的使用帮助，包括属性列表和示例[2,3]

1. 不建议使用此选项，它最终将被删除。应完全省略密码 （这样，命令将以交互方式提示用户输入密码）或使用 -passfile 选项来指定包含密码的密码文件。
2. 忽略命令行中指定的其他任何选项。
3. 此选项不需要用户名和密码。

对象管理器实用程序

对象管理器实用程序 (imqobjmgr) 用于创建和管理 Message Queue 受管理对象。表 13-9 列出了可用的子命令。

表 13-9 对象管理器子命令

子命令	描述
add	向对象存储中添加受管理对象
delete	从对象存储中删除受管理对象
list	列出对象存储中的受管理对象
query	显示受管理对象的信息
update	修改受管理对象

表 13-10 列出了 imqobjmgr 命令的选项。

表 13-10 对象管理器选项

选项	描述
-l *lookupName*	受管理对象的 JNDI 查找名称
-j *attribute=value*	JNDI 对象存储的属性 （请参见第 147 页上的 “对象存储”）

表 13-10 对象管理器选项 （续）

选项	描述
-t *objectType*	受管理对象的类型： q 队列目标 t 主题目标 cf 连接工厂 qf 队列连接工厂 tf 主题连接工厂 xcf 分布式事务的连接工厂 xqf 分布式事务的队列连接工厂 xtf 分布式事务的主题连接工厂 e SOAP 端点 （请参见 Message Queue Developer's Guide for Java Clients）
-o *attribute=value*	受管理对象的属性 （请参见第 150 页上的 “受管理对象的属性”和第 16 章 “受管理对象属性参考”）
-r *readOnlyState*	受管理对象是否为只读？ 如果为 true，客户端将无法修改对象的属性。默认值：false。
-i *fileName*	包含所有或部分子命令子句的命令文件的名称
-pre	预览结果但不执行命令 此选项对于检查默认属性值很有用。
-javahome *path*	可选 Java 运行时的位置 默认值：使用系统上安装的运行时或 Message Queue 附带的运行时。
-f	执行操作时无需用户确认
-s	无提示模式 （不显示输出）
-v	显示版本信息[1]
-h	显示使用帮助[1]
-H	显示详细的使用帮助，包括属性列表和示例[1]

1. 忽略命令行中指定的任何其他选项。

数据库管理器实用程序

数据库管理器实用程序 (imqdbmgr) 为基于 JDBC 的持久性数据存储建立数据库结构。您还可以使用它删除已损坏的 Message Queue 数据库表或更改数据存储。表 13-11 列出了可用的子命令。

表 13-11 数据库管理器子命令

子命令	描述
`create all`	创建新的数据库和持久性存储结构 用于嵌入式数据库系统。必须指定代理属性 `imq.persist.jdbc.createdburl`。
`create tbl`	为现有数据库创建持久性存储结构 用于外部数据库系统。
`delete tbl`	从当前持久性存储中删除 Message Queue 数据库表
`delete oldtbl`	从较早版本的持久性存储中删除 Message Queue 数据库表 在持久性存储自动迁移到 Message Queue 当前版本后使用。
`recreate tbl`	重新创建持久性存储结构 从当前的持久性存储中删除所有现有的 Message Queue 数据库表，然后重新创建结构。
`reset lck`	重置持久性存储锁 对锁进行重置，以便其他进程可以使用该持久性存储数据库。

表 13-12 列出了 `imqdbmgr` 命令的选项。

表 13-12 数据库管理器选项

选项	描述
`-b` *instanceName*	代理的实例名称
`-D`*property=value*	设置代理配置属性 有关与持久性相关的代理配置属性的信息，请参见第 266 页上的“持久性属性”。 **注意：**请仔细检查使用此选项设置的属性的拼写和格式。不正确的值将被忽略，且系统不会给出任何通知或警告。
`-u` *name*	用于验证的用户名
`-p` *password*	用于验证的密码[1]
`-passfile` *path*	密码文件的位置 有关更多信息，请参见第 144 页上的 “使用密码文件”。
`-v`	显示版本信息[2]
`-h`	显示使用帮助[2]

1. 不建议使用此选项，它最终将被删除。应完全省略密码 （这样，命令将以交互方式提示用户输入密码）或使用 `-passfile` 选项来指定包含密码的密码文件。

2. 忽略命令行中指定的其他任何选项。

用户管理器实用程序

用户管理器实用程序 (imqusermgr) 用于填充或编辑平面文件用户系统信息库。该实用程序必须在安装代理的主机上运行；如果尚不存在特定于代理的用户系统信息库，您必须首先启动相应的代理实例来创建它。此外，您还必须具有写入系统信息库的适当权限：在 Solaris 或 Linux 平台上，这意味着您必须是 root 用户或最初创建代理实例的用户。

表 13-13 列出了可以与 imqusermgr 命令一起使用的子命令。在任何情况下， -i 选项都指定代理的实例名称，该代理的用户系统信息将应用此命令；如果未指定，将采用默认名称 imqbroker。

表 13-13 用户管理器子命令

语法	描述
add [-i *instanceName*] -u *userName* -p *password* [-g *group*]	向系统信息库中添加用户和密码 -g 选项可选，它指定要将此用户指定到的组： admin user anonymous
delete [-i *instanceName*] -u *userName*	从系统信息库中删除用户
update [-i *instanceName*] -u *userName* -p *password* [-a *activeState*]	设置用户的密码和 / 或活动状态 -a 选项采用布尔值，它指定使用户处于活动状态 (true) 还是非活动状态 (false)。默认值：true。
update [-i *instanceName*] -u *userName* -a *activeState* [-p *password*]	
list [-i *instanceName*] [-u *userName*]	显示用户信息。 如果未指定用户名，则列出系统信息库中的所有用户。

此外，表 13-14 中列出的选项适用于 imqusermgr 命令的所有子命令。

表 13-14 常规用户管理器选项

选项	描述
-f	执行操作时无需用户执行操作，也无需用户确认。
-s	无提示模式 （不显示输出）
-v	显示版本信息[1]
-h	显示使用帮助[1]

1. 忽略命令行中指定的其他任何选项。

服务管理器实用程序

服务管理器实用程序 (imqsvcadmin) 将代理作为 Windows 服务安装。表 13-15 列出了可用的子命令。

表 13-15 服务管理器子命令

子命令	描述
install	安装服务
remove	删除服务
query	显示启动选项 启动选项可以包括服务的启动方式 （手动还是自动）、服务的位置、 Java 运行时的位置以及启动时传递给代理的参数值。

表 13-16 列出了 imqsvcadmin 命令的选项。

表 13-16 服务管理器选项

选项	描述
-javahome *path*	可选 Java 运行时的位置 默认值：使用系统上安装的运行时或 Message Queue 附带的运行时。
-jrehome *path*	可选 Java 运行环境 (Java Runtime Environment, JRE) 的位置
-vmargs *arg1* [[*arg2*] ...]	要传递给运行代理服务的 Java 虚拟机的其他参数[1] 示例： `imqsvcadmin install -vmargs "-Xms16m -Xmx128m"`

表 13-16 服务管理器选项 （续）

选项	描述
-args *arg1* [[*arg2*] ...]	要传递给代理服务的其他命令行参数[1] 示例： `imqsvcadmin install -args "-passfile d:\imqpassfile"` 有关代理命令行参数的信息，请参见第 242 页上的 “代理实用程序”。
-h	显示使用帮助[2]

1. 还可以在服务 “属性”窗口 （可以通过 Windows “管理工具”控制面板中的 “服务”工具访问此窗口）“常规”选项卡下的 “启动参数”字段中指定这些参数。
2. 忽略命令行中指定的任何其他选项。

使用 -javahome、 -vmargs 和 -args 选项指定的所有信息都存储在 Window 注册表项 JavaHome、 JVMArgs 和 ServiceArgs 下，这些注册表项的路径为：

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\iMQ_Broker\Parameters
```

密钥工具实用程序

密钥工具实用程序 (imqkeytool) 为代理生成自签名证书，该证书可用于 ssljms、ssladmin 或 cluster 连接服务。语法为：

```
imqkeytool -broker
```

在 UNIX 系统上，您可能需要使用超级用户 (root) 帐户来运行该实用程序。

第 14 章

代理属性参考

本章提供了有关消息代理配置属性的参考信息。本章包含以下各节：

连接属性

表 14-1 中列出了与连接服务相关的代理属性。

表 14-1 代理连接属性

属性	类型	默认值	描述
imq.service.activelist	字符串	jms,admin	代理启动时将激活的连接服务的列表，用逗号分隔。
imq.hostname	字符串	所有可用的 IP 地址	所有连接服务的默认主机名或 IP 地址
imq.portmapper.hostname	字符串	无	端口映射器的主机名或 IP 地址 如果指定，将覆盖 imq.hostname。

表 14-1 代理连接属性 （续）

属性	类型	默认值	描述
imq.portmapper.port[1]	整数	7676	端口映射器的端口号 **注：**如果多个代理实例在同一台主机上运行，则必须为每个实例指定一个唯一的端口映射器端口。
imq.*serviceName*.*protocolType*.hostname	字符串	无	连接服务的主机名或 IP 地址[2] 如果指定，将覆盖指定连接服务的 imq.hostname。
imq.*serviceName*.*protocolType*.port	整数	0	连接服务的端口号[2] 值为 0 表示应该由端口映射器动态分配端口号。
imq.portmapper.backlog	整数	50	操作系统待办事项中处于待处理状态的端口映射器请求的最大数量。
imq.*serviceName*.threadpool_model	字符串	dedicated	线程池管理的线程处理模型： dedicated 每个连接有两个专用线程，一个用于传入消息，另一个用于传出消息 shared[3] 在发送或接收消息时，连接由共享线程处理 专用模型会限制可以支持的连接数，但能提供较高的性能；共享模型可以增加可用连接数，但由于线程管理需要额外的开销，因此会导致性能下降。
imq.*serviceName*.min_threads	整数	jms: 10 ssljms: 10 httpjms: 10 httpsjms: 10 admin: 4 ssladmin: 4	连接服务的线程池中维护的最小线程数 当可用线程数超出此阈值时，将会关闭空闲线程，直到达到最小线程数。 如表中所示，默认值因连接服务而异。
imq.*serviceName*.max_threads	整数	jms: 1000 ssljms: 500 httpjms: 500 httpsjms: 500 admin: 10 ssladmin: 10	线程数量，达到该数量后，新的线程将不会被添加到线程池中供命名的连接服务使用。该数必须大于零，并且必须大于 min_threads 的值。 如表中所示，默认值因连接服务而异。
imq.shared.connectionMonitor_limit	整数	Solaris：512 Linux：512 Windows：64	分配器线程监视的最大连接数[4] 系统会分配足够多的分配器线程来监视所有连接。此属性的值越小，为活动连接指定线程的速度越快。值为 -1 表示不限制每个线程的连接数。 如表中所示，默认值因操作系统平台而异。

表 14-1 代理连接属性 （续）

属性	类型	默认值	描述
imq.ping.interval	整数	120	测试客户端与代理之间的连接的时间间隔 （以秒为单位） 如果值为 0 或 -1，将禁止定期测试连接。

1. 可用于 imqcmd update bkr 命令。
2. 仅限于 jms、 ssljms、 admin 和 ssladmin 服务；有关配置 httpjms 和 httpsjms 服务的信息，请参见附录 C “HTTP/HTTPS 支持”。
3. 仅限于 jms 和 admin 服务
4. 仅适用于共享线程处理模型。

路由属性

表 14-2 中列出了与路由服务相关的代理属性。表 14-3 中列出了用于配置目标自动创建功能的属性。

表 14-2 代理路由属性

属性	类型	默认值	描述
imq.system.max_count[1]	整数	-1	代理保存的最大消息数 值为 -1 表示不限制消息数量。
imq.system.max_size[1]	字符串	-1	代理保存的消息总量的最大值 可以使用以下后缀 （分别表示字节、千字节或兆字节）来表示该值： b 字节 k 千字节 （1024 字节） m 兆字节 （1024 x 1024 = 1,048,576 字节） 不带后缀的值用字节表示；值为 -1 表示不限制消息容量。 示例： 1600 1600 字节 1600b 1600 字节 16k 16 千字节 （= 16,384 字节） 16m 16 兆字节 （= 16,777,216 字节） -1 无限制
imq.message.max_size[1]	字符串	70m	单个消息主体的最大大小 语法与 imq.system.max_size 的语法相同 （请参见上文）。
imq.message.expiration.interval	整数	60	回收过期消息的时间间隔 （以秒为单位）

表 14-2 代理路由属性 （续）

属性	类型	默认值	描述
imq.*resourceState*.threshold	整数	green： 0 yellow： 80 orange： 90 red： 98	触发内存资源状态的占用百分比 （其中 *resourceState* 为 green、 yellow、 orange 或 red）
imq.*resourceState*.count	整数	green：5000 yellow：500 orange： 50 red： 0	在检查是否达到内存资源状态阈值之前，允许在一批中传入的最大消息数 （其中 *resourceState* 为 green、 yellow、 orange 或 red） 当系统内存越来越不足时，此限制可以限制消息生成方的数量。
imq.destination.DMQ.truncateBody[1]	布尔值	false	在存储到停用消息队列之前是否删除消息主体？ 如果为 true，则仅保存消息头和属性数据。
imq.transaction.autorollback	布尔值	false	是否自动回滚在代理启动时处于 PREPARED 状态的分布式事务？ 如果为 false，则必须使用命令实用程序 (imqcmd) 手动提交或回滚事务。

1. 可用于 imqcmd update bkr 命令。

表 14-3 自动创建目标的代理属性

属性	类型	默认值	描述
imq.autocreate.queue[1,2]	布尔值	true	是否允许自动创建队列目标？
imq.autocreate.topic[3]	布尔值	true	是否允许自动创建主题目标？
imq.autocreate.destination.maxNumMsgs	整数	100000	未使用消息的最大数量 值为 -1 表示不限制消息数量。

表 14-3 自动创建目标的代理属性 （续）

属性	类型	默认值	描述
imq.autocreate.destination.maxBytesPerMsg	字符串	10k	任何单个消息的最大大小 （以字节为单位） 可以使用以下后缀 （分别表示字节、千字节或兆字节）来表示该值： b 字节 k 千字节 （1024 字节） m 兆字节 （1024 x 1024 = 1,048,576 字节） 不带后缀的值用字节表示；值为 -1 表示不限制消息大小。 示例： 1600 1600 字节 1600b 1600 字节 16k 16 千字节 （= 16,384 字节） 16m 16 兆字节 （= 16,777,216 字节） -1 无限制
imq.autocreate.destination.maxTotalMsgBytes	字符串	10m	未使用的消息最多可以占用的内存总量 （以字节为单位） 语法与 imq.autocreate.destination.maxBytesPerMsg 的语法相同 （请参见上文）。

表 14-3 自动创建目标的代理属性 （续）

属性	类型	默认值	描述
imq.autocreate.destination.limitBehavior	字符串	REJECT_NEWEST	当达到内存限制的阈值时，代理的行为如下： FLOW_CONTROL 降低生成方速度 REMOVE_OLDEST 丢弃最旧的消息 REMOVE_LOW_PRIORITY 根据消息的存在时间丢弃优先级最低的消息； 不通知 生成方客户端 REJECT_NEWEST 拒绝最新的消息；仅当消息为持久性消息时，才通知生成方客户端，并报告异常 如果值为 REMOVE_OLDEST 或 REMOVE_LOW_PRIORITY，且 imq.autocreate.destination.useDMQ 属性为 true，则会将超出限制的消息移动到停用消息队列中。
imq.autocreate.destination.maxNumProducers	整数	100	目标的消息生成方的最大数量 当达到此限制时，将无法创建新的生成方。值为 -1 表示不限制生成方数量。
imq.autocreate.queue.maxNumActiveConsumers[2]	整数	1	在来自队列目标的负载平衡传送中活动消息使用方的最大数量 值为 -1 表示不限制使用方数量。
imq.autocreate.queue.maxNumBackupConsumers[2]	整数	0	在来自队列目标的负载平衡传送中备份消息使用方的最大数量 值为 -1 表示不限制使用方数量。

表 14-3 自动创建目标的代理属性 （续）

属性	类型	默认值	描述
imq.autocreate.queue.consumerFlowLimit[2]	整数	1000	在一批中传送给队列使用方的最大消息数 在负载平衡队列传送中，此值为负载平衡开始之前路由到活动使用方的队列消息的初始数量。目标使用方可以通过在连接中指定一个较小的值来覆盖此限制。 值为 -1 表示不限制使用方数量。
imq.autocreate.topic.consumerFlowLimit[3]	整数	1000	在一批中传送给主题使用方的最大消息数 值为 -1 表示不限制使用方数量。
imq.autocreate.destination.isLocalOnly	布尔值	false	仅限本地传送？ 此属性仅适用于代理群集中的目标，目标一旦创建后，此属性即无法更改。如果为 true，将不在其他代理上复制目标，并限制目标只将消息传送到本地使用方 （那些连接到创建该目标的代理的使用方）。
imq.autocreate.queue.localDeliveryPreferred[2]	布尔值	false	是否首选本地传送？ 此属性仅适用于代理群集中的负载平衡队列传送。如果为 true，则只有在本地代理中不存在使用方时，才会将消息传送给远程使用方；不得将目标限制为仅本地传送 （imq.autocreate.destination.isLocalOnly 必须为 false）。
imq.autocreate.destination.useDMQ	布尔值	true	是否将停用消息发送到停用消息队列？ 如果为 false，将会丢弃停用消息。

1. 可用于 imqcmd update bkr 命令。
2. 仅限于队列目标。
3. 仅限于主题目标

持久性属性

Message Queue 支持使用基于文件和基于 JDBC 的模型来存储持久性数据。代理属性 imq.persist.store（表 14-4）指定要使用的模型。以下各节介绍了这两种模型的代理配置属性。

表 14-4 全局代理持久性属性

属性	类型	默认值	描述
imq.persist.store	字符串	file	持久性数据存储模型： file 基于文件的持久性 jdbc 基于 JDBC 的持久性

基于文件的持久性

表 14-5 中列出了与基于文件的持久性相关的代理属性。

表 14-5 基于文件持久性的代理属性

属性	类型	默认值	描述
imq.persist.file.message.max_record_size	字符串	1m	要添加到消息存储文件中的消息的最大大小。 超出此大小的所有消息都将存储到其自身的单独文件中。 可以使用以下后缀（分别表示字节、千字节或兆字节）来表示该值： b 字节 k 千字节（1024 字节） m 兆字节（1024 x 1024 = 1,048,576 字节） 不带后缀的值用字节表示。 示例： 1600 1600 字节 1600b 1600 字节 16k 16 千字节（= 16,384 字节） 16m 16 兆字节（= 16,777,216 字节）

表 14-5 基于文件持久性的代理属性 （续）

属性	类型	默认值	描述
imq.persist.file.destination.message.filepool.limit	整数	100	目标文件池中可重复使用的最大空闲文件数。 超出此限制的空闲文件将被删除。代理将根据需要创建文件或删除超出限制的文件。 此限制越高，代理处理持久性数据的速度越快。
imq.persist.file.message.filepool.cleanratio	整数	0	空闲文件池中将保持清除 （清空）状态的文件的百分比。 此值越大，文件池所需的磁盘空间越少，但在操作中清除文件所需的开销越大。
imq.persist.file.message.cleanup	布尔值	false	是否在关闭时清除空闲文件池中的文件？ 将此属性设置为 true 可以节省用于文件存储的磁盘空间，但会降低代理关闭的速度。
imq.persist.file.sync.enabled	布尔值	false	是否将内存中的状态与物理存储设备同步？ 将此属性设置为 true 可以避免因系统崩溃引起的数据丢失，但性能会有所下降。 **注：**如果为 Message Queue 运行 Sun Cluster 和 Sun Cluster Data Service，请将所有群集节点上的代理的此属性设置为 true。

基于 JDBC 的持久性

表 14-6 中列出了与基于 JDBC 的持久性相关的代理属性。给出的示例适用于 DataMirror Mobile Solutions, Inc. 提供的 PointBase® 数据库产品系列。

表 14-6 基于 JDBC 持久性的代理属性

属性	描述	示例
imq.persist.jdbc.brokerid	**（可选）**代理实例标识符 标识符必须是字母数字字符串，其长度不能超过 *n* - 12，其中 *n* 是数据库允许的最大表名长度。 此标识符将附加到数据库表名之后，当多个代理实例使用同一个数据库作为持久性数据存储时，此标识符可以使数据库表名保持唯一。嵌入式数据库只存储一个代理实例的数据，因此通常不需要该标识符。	PointBase 嵌入式版本不需要。
imq.persist.jdbc.driver	用于连接到数据库的 JDBC 驱动程序的 Java 类名	com.pointbase.jdbc.jdbcUniversalDriver
imq.persist.jdbc.opendburl	用于打开现有数据库连接的 URL	jdbc:pointbase:embedded:*dbName*; database.home= .../instances/*instanceName*/dbstore
imq.persist.jdbc.createdburl	**（可选）**用于创建新数据库的 URL 仅在使用 Message Queue 数据库管理器实用程序 (imqdbmgr) 创建数据库时才需要此属性。	jdbc:pointbase:embedded:*dbName*; new,database.home= .../instances/*instanceName*/dbstore
imq.persist.jdbc.closedburl	**（可选）**用于关闭数据库连接的 URL	PointBase 不需要。
imq.persist.jdbc.user	**（可选）**用于打开数据库连接的用户名（如果需要）。 出于安全原因，可以指定该值，而不使用命令行选项 imqbrokerd -dbuser 和 imqdbmgr -u。	
imq.persist.jdbc.needpassword	**（可选）**数据库是否要求为代理访问输入密码？ 如果为 true，则 imqbrokerd 和 imqdbmgr 命令将提示输入密码，除非您使用 -passfile 选项指定包含密码的密码文件。	

表 14-6 基于 JDBC 持久性的代理属性 （续）

属性	描述	示例
imq.persist.jdbc.password	**（可选）**用于打开数据库连接的密码 应仅在密码文件中指定此属性。	
imq.persist.jdbc.table.IMQSV35	用于创建版本表的 SQL 命令。	CREATE TABLE ${*name*} (STOREVERSION INTEGER NOT NULL, BROKERID VARCHAR(100))
imq.persist.jdbc.table.IMQCCREC35	用于创建配置更改记录表的 SQL 命令	CREATE TABLE ${*name*} (RECORDTIME BIGINT NOT NULL, RECORD BLOB(10k))
imq.persist.jdbc.table.IMQDEST35	用于创建目标表的 SQL 命令。	CREATE TABLE ${*name*} (DID VARCHAR(100) NOT NULL, DEST BLOB(10k), *primaryKey*(DID))
imq.persist.jdbc.table.IMQINT35	用于创建 Interest 表的 SQL 命令	CREATE TABLE ${name} (CUID BIGINT NOT NULL, INTEREST BLOB(10k), *primaryKey*(CUID))
imq.persist.jdbc.table.IMQMSG35	用于创建消息表的 SQL 命令 MSG 列的默认最大长度为 1 兆字节 (1m)。如果您希望消息超出此长度，请对长度进行相应设置。如果已经创建了这些表，则必须重新创建它们才能更改最大消息长度。	CREATE TABLE ${*name*} (MID VARCHAR(100) NOT NULL, DID VARCHAR(100), MSGSIZE BIGINT, MSG BLOB(1m), *primaryKey*(MID))
imq.persist.jdbc.table.IMQPROPS35	用于创建属性表的 SQL 命令	CREATE TABLE ${*name*} (PROPNAME VARCHAR(100) NOT NULL, PROPVALUE BLOB(10k), *primaryKey*(PROPNAME))
imq.persist.jdbc.table.IMQILIST35	用于创建 Interest 状态表的 SQL 命令	CREATE TABLE ${*name*} (MID VARCHAR(100) NOT NULL, CUID BIGINT, DID VARCHAR(100), STATE INTEGER, *primaryKey*(MID, CUID))
imq.persist.jdbc.table.IMQTXN35	用于创建事务表的 SQL 命令	CREATE TABLE ${*name*} (TUID BIGINT NOT NULL, STATE INTEGER, TSTATEOBJ BLOB(10K), *primaryKey*(TUID))

表 14-6 基于 JDBC 持久性的代理属性 （续）

属性	描述	示例
imq.persist.jdbc.table.IMQTACK35	用于创建事务确认表的 SQL 命令	CREATE TABLE ${*name*} (TUID BIGINT NOT NULL, TXNACK BLOB(10k))

安全属性

表 14-7 中列出了与安全服务相关的代理属性。

表 14-7 安全属性

属性	类型	默认值	描述
imq.accesscontrol.enabled	布尔值	true	是否使用访问控制？ 如果为 true，系统将检查访问控制属性文件，以验证是否允许经过验证的用户使用连接服务或执行与特定目标有关的特定操作。
imq.*serviceName*.accesscontrol.enabled	布尔值	无	是否为连接服务使用访问控制？ 如果指定，则覆盖指定连接服务的 imq.accesscontrol.enabled。 如果为 true，系统将检查访问控制属性文件，以验证是否允许经过验证的用户使用指定的连接服务或执行与特定目标有关的特定操作。
imq.accesscontrol.file.filename	字符串	accesscontrol.properties	访问控制属性文件的名称 文件名指定了相对于访问控制目录的路径 （请参见附录 A）。
imq.*serviceName*.accesscontrol.file.filename	字符串	无	连接服务的访问控制属性文件的名称 如果指定，则覆盖指定连接服务的 imq.accesscontrol.file.filename。 文件名指定了相对于访问控制目录的路径 （请参见附录 A）。
imq.authentication.type	字符串	digest	密码编码方法： basic Base-64 digest MD5

表 14-7 安全属性 （续）

属性	类型	默认值	描述
imq.*serviceName*.authentication.type	字符串	无	连接服务的密码编码方法： basic Base-64 digest MD5 如果指定，则覆盖指定连接服务的 imq.authentication.type。
imq.authentication.basic.user_repository	字符串	file	base-64 验证的用户系统信息库类型： file 基于文件 ldap LDAP
imq.authentication.client.response.timeout	整数	180	等待客户端响应验证请求的时间间隔 （以秒为单位）。
imq.passfile.enabled	布尔值	false	是否从密码文件获取密码？
imq.passfile.dirpath	字符串	请参见附录 A	包含密码文件的目录的路径
imq.passfile.name	字符串	passfile	密码文件的名称
imq.imqcmd.password	字符串	无	管理用户的密码。 命令实用程序 (imqcmd) 在执行命令前使用此密码验证用户。

表 14-7 安全属性 （续）

属性	类型	默认值	描述
imq.user_repository.ldap.server	字符串	无	LDAP 服务器的主机名和端口号 该值的格式为： *hostName*:*port* 其中 *hostName* 是运行 LDAP 服务器的主机的全限定 DNS 名称，而 *port* 是服务器使用的端口号。 要指定故障转移服务器列表，请使用下面的语法： *host1*:*port1* ldap://*host2*:*port2* ldap://*host3*:*port3* ... 列表中的条目用空格分隔。请注意，每个故障转移服务器地址均带有 ldap:// 前缀。即使您使用 SSL 并将属性 imq.user_repository.ldap.ssl.enabled 设置为 true，也请使用此格式。您无需在地址中指定 ldaps。
imq.user_repository.ldap.principal	字符串	无	用于绑定到 LDAP 用户系统信息库的标识名 如果 LDAP 服务器允许匿名搜索，则无需此标识名。
imq.user_repository.ldap.password	字符串	无	用于绑定到 LDAP 用户系统信息库的密码 如果 LDAP 服务器允许匿名搜索，则无需此密码。 应仅在密码文件中指定此属性。
imq.user_repository.ldap.*propertyName*	待定	待定	待定
imq.user_repository.ldap.base	字符串	无	LDAP 用户条目的目录库
imq.user_repository.ldap.uidattr	字符串	无	LDAP 用户名的特定于提供者的属性标识符
imq.user_repository.ldap.usrfilter	字符串	无	**（可选）**用于 LDAP 用户搜索的 JNDI 过滤器

表 14-7 安全属性（续）

属性	类型	默认值	描述
imq.user_repository.ldap.grpsearch	布尔值	false	是否启用 LDAP 组搜索？ **注：**Message Queue 不支持嵌套组。
imq.user_repository.ldap.grpbase	字符串	无	LDAP 组条目的目录库。
imq.user_repository.ldap.gidattr	字符串	无	LDAP 组名的特定于提供者的属性标识符。
imq.user_repository.ldap.memattr	字符串	无	LDAP 组中用户名的特定于提供者的属性标识符
imq.user_repository.ldap.grpfilter	字符串	无	**（可选）**用于 LDAP 组搜索的 JNDI 过滤器
imq.user_repository.ldap.timeout	整数	280	LDAP 搜索的时间限制 （以秒为单位）
imq.user_repository.ldap.ssl.enabled	布尔值	false	与 LDAP 服务器通信时是否使用 SSL？
imq.keystore.file.dirpath	字符串	请参见附录 A	包含密钥库文件的目录的路径
imq.keystore.file.name	字符串	keystore	密钥库文件的名称
imq.keystore.password	字符串	无	密钥库文件的密码 应仅在密码文件中指定此属性。
imq.audit.enabled	布尔值	false	是否启动代理日志文件的审计日志？ 此选项仅适用于 Message Queue Enterprise Edition。

监视属性

表 14-8 中列出了与监视服务相关的代理属性。

表 14-8 代理监视属性

属性	类型	默认值	描述
imq.log.level[1]	字符串	INFO	日志记录级别 指定可以写入输出通道的日志记录信息的类别。可能的值为 （由高至低）： ERROR WARNING INFO 每个级别都包括高于它的级别 （例如， WARNING 包括 ERROR）。
imq.destination.logDeadMsgs[1]	布尔值	false	是否记录有关停用消息的信息？ 如果为 true，将记录以下事件： • 目标已满，已达到最大大小或最大消息数。 • 代理由于管理命令或传送确认以外的原因丢弃消息。 • 代理将消息移动到停用消息队列。
imq.log.console.stream	字符串	ERR	控制台输出的目标： OUT stdout ERR stderr
imq.log.console.output	字符串	ERROR|WARNING	可以写入控制台的日志记录信息的类别： NONE ERROR WARNING INFO ALL ERROR、 WARNING 和 INFO 类别*不*包括高于它们的类别，因此必须根据需要明确指定每个类别。可以指定用竖线 (|) 分隔的任意类别组合。
imq.log.file.dirpath	字符串	请参见附录 A	包含日志文件的目录的路径
imq.log.file.filename	字符串	log.txt	日志文件的名称

表 14-8 代理监视属性 （续）

属性	类型	默认值	描述
imq.log.file.output	字符串	ALL	可以写入日志文件的日志记录信息的类别： NONE ERROR WARNING INFO ALL ERROR、 WARNING 和 INFO 类别*不*包括高于它们的类别，因此必须根据需要明确指定每个类别。可以指定用竖线 (|) 分隔的任意类别组合。
imq.log.file.rolloverbytes[1]	整数	-1	文件长度 （以字节为单位），达到该值后，输出将转移到新的日志文件。 值为 -1 表示不限制字节数 （不基于文件长度转移输出）。
imq.log.file.rolloversecs[1]	整数	604800 （一周）	文件的生存期 （以秒为单位），达到该值后，输出将转移到新的日志文件。 值为 -1 表示不限制秒数 （不基于文件生存期转移输出）。
imq.log.syslog.output[2]	字符串	ERROR	可以写入 syslogd(1M) 的日志记录信息的类别： NONE ERROR WARNING INFO ALL ERROR、 WARNING 和 INFO 类别*不*包括高于它们的类别，因此必须根据需要明确指定每个类别。可以指定用竖线 (|) 分隔的任意类别组合。

表 14-8 代理监视属性 （续）

属性	类型	默认值	描述
imq.log.syslog.facility[2]	字符串	LOG_DAEMON	用于记录消息的 syslog 工具 可能的值镜像 syslog(3C) 手册页中列出的值。适用于 Message Queue 的值包括： LOG_USER LOG_DAEMON LOG_LOCAL0 LOG_LOCAL1 LOG_LOCAL2 LOG_LOCAL3 LOG_LOCAL4 LOG_LOCAL5 LOG_LOCAL6 LOG_LOCAL7
imq.log.syslog.identity[2]	字符串	imqbrokerd_${imq.*instanceName*}	标识字符串，该字符串将被加到记录到 syslog 中的每条消息的前面。
imq.log.syslog.logpid[2]	布尔值	true	是否将代理进程 ID 与消息一起记录？
imq.log.syslog.logconsole[2]	布尔值	false	如果无法将消息发送到 syslog，是否将其写入系统控制台？
imq.log.timezone	字符串	当地时区	日志时间戳的时区。 可能的值与方法 java.util.TimeZone.getTimeZone 使用的值相同。示例： GMT GMT-8:00 America/LosAngeles Europe/Rome Asia/Tokyo
imq.metrics.enabled	布尔值	true	是否将度量信息写入记录程序？ 不影响度量消息的生成 （由 imq.metrics.topic.enabled 控制）。
imq.metrics.interval	整数	-1	将度量信息写入记录程序的时间间隔 （以秒为单位）。 不影响度量消息生成的时间间隔 （由 imq.metrics.topic.interval 控制）。 值为 -1 表示时间间隔为无限长 （永不将度量信息写入记录程序）。

表 14-8 代理监视属性 （续）

属性	类型	默认值	描述
imq.metrics.topic.enabled	布尔值	true	是否向度量主题目标中生成度量消息？ 如果为 false，则试图订阅度量主题目标时会抛出客户端异常。
imq.metrics.topic.interval	整数	60	向度量主题目标中生成度量消息的时间间隔 （以秒为单位）
imq.metrics.topic.persist	布尔值	false	发送到度量主题目标的度量消息是否为永久性消息？
imq.metrics.topic.timetolive	整数	300	发送到度量主题目标的度量消息的生命周期 （以秒为单位）

1. 可用于 imqcmd update bkr 命令
2. 仅限于 Solaris 平台

群集配置属性

表 14-9 中列出了与代理群集相关的配置属性。

表 14-9 群集配置的代理属性

属性	类型	默认值	描述
imq.cluster.brokerlist[1]	字符串	无	代理地址的列表 该列表中包含一个或多个用逗号分隔的地址。每个地址指定了群集中某个代理的主机名和端口映射器端口号，格式为 *hostName*:*portNumber*。示例： host1:3000,host2:8000,ctrlhost
imq.cluster.hostname[2]	字符串	无	cluster 连接服务的主机名或 IP 地址 如果指定，则覆盖 cluster 连接服务的 imq.hostname （请参见 第 259 页上的表 14-1）。
imq.cluster.port[2]	整数	0	cluster 连接服务的端口号 值为 0 表示应当由端口映射器动态分配端口号。
imq.cluster.transport[1]	字符串	tcp	cluster 连接服务的网络传输协议 要在代理之间实现安全、加密的消息传送，请将此属性设置为 ssl。

表 14-9 群集配置的代理属性 （续）

属性	类型	默认值	描述
imq.cluster.url[1,3]	字符串	无	cluster 配置文件的 URL （如果有） 示例： http://webserver/imq/cluster.properties （对于 Web 服务器上的文件） file:/net/mfsserver/imq/cluster.properties （对于共享驱动器上的文件）
imq.cluster.masterbroker[1]	字符串	无	群集主代理的主机名和端口号 （如果有） 该值的格式为 *hostName*:*portNumber*，其中 *hostName* 是主代理的主机名， *portNumber* 是它的端口映射器端口号。示例： ctrlhost:7676

1. 群集中的所有代理必须具有相同的值。
2. 可以为群集中的每个代理单独指定。
3. 可用于 imqcmd update bkr 命令。

按字母顺序排列的代理属性列表

表 14-10 是按字母顺序排列的代理配置属性列表，其中包含对本章中相关表的交叉引用。

表 14-10 按字母顺序排列的代理属性列表

属性	表
imq.accesscontrol.enabled	第 270 页上的表 14-7
imq.accesscontrol.file.filename	第 270 页上的表 14-7
imq.audit.enabled	第 270 页上的表 14-7
imq.authentication.basic.user_repository	第 270 页上的表 14-7
imq.authentication.client.response.timeout	第 270 页上的表 14-7
imq.authentication.type	第 270 页上的表 14-7
imq.autocreate.destination.isLocalOnly	第 262 页上的表 14-3
imq.autocreate.destination.limitBehavior	第 262 页上的表 14-3
imq.autocreate.destination.maxBytesPerMsg	第 262 页上的表 14-3
imq.autocreate.destination.maxNumMsgs	第 262 页上的表 14-3
imq.autocreate.destination.maxNumProducers	第 262 页上的表 14-3

表 14-10 按字母顺序排列的代理属性列表 （续）

属性	表
imq.autocreate.destination.maxTotalMsgBytes	第 262 页上的表 14-3
imq.autocreate.destination.useDMQ	第 262 页上的表 14-3
imq.autocreate.queue	第 262 页上的表 14-3
imq.autocreate.queue.consumerFlowLimit	第 262 页上的表 14-3
imq.autocreate.queue.localDeliveryPreferred	第 262 页上的表 14-3
imq.autocreate.queue.maxNumActiveConsumers	第 262 页上的表 14-3
imq.autocreate.queue.maxNumBackupConsumers	第 262 页上的表 14-3
imq.autocreate.topic	第 262 页上的表 14-3
imq.autocreate.topic.consumerFlowLimit	第 262 页上的表 14-3
imq.cluster.brokerlist	第 277 页上的表 14-9
imq.cluster.hostname	第 277 页上的表 14-9
imq.cluster.masterbroker	第 277 页上的表 14-9
imq.cluster.port	第 277 页上的表 14-9
imq.cluster.transport	第 277 页上的表 14-9
imq.cluster.url	第 277 页上的表 14-9
imq.destination.DMQ.truncateBody	第 261 页上的表 14-2
imq.destination.logDeadMsgs	第 274 页上的表 14-8
imq.hostname	第 259 页上的表 14-1
imq.imqcmd.password	第 270 页上的表 14-7
imq.keystore.file.dirpath	第 270 页上的表 14-7
imq.keystore.file.name	第 270 页上的表 14-7
imq.keystore.password	第 270 页上的表 14-7
imq.keystore.*propertyName*	第 270 页上的表 14-7
imq.log.console.output	第 274 页上的表 14-8
imq.log.console.stream	第 274 页上的表 14-8
imq.log.file.dirpath	第 274 页上的表 14-8
imq.log.file.filename	第 274 页上的表 14-8
imq.log.file.output	第 274 页上的表 14-8
imq.log.file.rolloverbytes	第 274 页上的表 14-8
imq.log.file.rolloversecs	第 274 页上的表 14-8

表 14-10 按字母顺序排列的代理属性列表 （续）

属性	表
imq.log.level	第 274 页上的表 14-8
imq.log.syslog.facility	第 274 页上的表 14-8
imq.log.syslog.identity	第 274 页上的表 14-8
imq.log.syslog.logconsole	第 274 页上的表 14-8
imq.log.syslog.logpid	第 274 页上的表 14-8
imq.log.syslog.output	第 274 页上的表 14-8
imq.log.timezone	第 274 页上的表 14-8
imq.message.expiration.interval	第 261 页上的表 14-2
imq.message.max_size	第 261 页上的表 14-2
imq.metrics.enabled	第 274 页上的表 14-8
imq.metrics.interval	第 274 页上的表 14-8
imq.metrics.topic.enabled	第 274 页上的表 14-8
imq.metrics.topic.interval	第 274 页上的表 14-8
imq.metrics.topic.persist	第 274 页上的表 14-8
imq.metrics.topic.timetolive	第 274 页上的表 14-8
imq.passfile.dirpath	第 270 页上的表 14-7
imq.passfile.enabled	第 270 页上的表 14-7
imq.passfile.name	第 270 页上的表 14-7
imq.persist.file.destination.message.filepool.limit	第 266 页上的表 14-5
imq.persist.file.message.cleanup	第 266 页上的表 14-5
imq.persist.file.message.filepool.cleanratio	第 266 页上的表 14-5
imq.persist.file.message.max_record_size	第 266 页上的表 14-5
imq.persist.file.sync.enabled	第 266 页上的表 14-5
imq.persist.jdbc.brokerid	第 268 页上的表 14-6
imq.persist.jdbc.closedburl	第 268 页上的表 14-6
imq.persist.jdbc.createdburl	第 268 页上的表 14-6
imq.persist.jdbc.driver	第 268 页上的表 14-6
imq.persist.jdbc.needpassword	第 268 页上的表 14-6
imq.persist.jdbc.opendburl	第 268 页上的表 14-6
imq.persist.jdbc.password	第 268 页上的表 14-6

表 14-10 按字母顺序排列的代理属性列表 （续）

属性	表
imq.persist.jdbc.table.IMQCCREC35	第 268 页上的表 14-6
imq.persist.jdbc.table.IMQDEST35	第 268 页上的表 14-6
imq.persist.jdbc.table.IMQILIST35	第 268 页上的表 14-6
imq.persist.jdbc.table.IMQINT35	第 268 页上的表 14-6
imq.persist.jdbc.table.IMQMSG35	第 268 页上的表 14-6
imq.persist.jdbc.table.IMQPROPS35	第 268 页上的表 14-6
imq.persist.jdbc.table.IMQSV35	第 268 页上的表 14-6
imq.persist.jdbc.table.IMQTACK35	第 268 页上的表 14-6
imq.persist.jdbc.table.IMQTXN35	第 268 页上的表 14-6
imq.persist.jdbc.user	第 268 页上的表 14-6
imq.persist.store	第 266 页上的表 14-4
imq.ping.interval	第 259 页上的表 14-1
imq.portmapper.backlog	第 259 页上的表 14-1
imq.portmapper.hostname	第 259 页上的表 14-1
imq.portmapper.port	第 259 页上的表 14-1
imq.*resourceState*.count	第 261 页上的表 14-2
imq.*resourceState*.threshold	第 261 页上的表 14-2
imq.service.activelist	第 259 页上的表 14-1
imq.*serviceName*.accesscontrol.enabled	第 270 页上的表 14-7
imq.*serviceName*.accesscontrol.file.filename	第 270 页上的表 14-7
imq.*serviceName*.authentication.type	第 270 页上的表 14-7
imq.*serviceName*.max_threads	第 259 页上的表 14-1
imq.*serviceName*.min_threads	第 259 页上的表 14-1
imq.*serviceName*.*protocolType*.hostname	第 259 页上的表 14-1
imq.*serviceName*.*protocolType*.port	第 259 页上的表 14-1
imq.*serviceName*.threadpool_model	第 259 页上的表 14-1
imq.shared.connectionMonitor_limit	第 259 页上的表 14-1
imq.system.max_count	第 261 页上的表 14-2
imq.system.max_size	第 261 页上的表 14-2
imq.transaction.autorollback	第 261 页上的表 14-2

表 14-10 按字母顺序排列的代理属性列表 （续）

属性	表
imq.user_repository.ldap.base	第 270 页上的表 14-7
imq.user_repository.ldap.gidattr	第 270 页上的表 14-7
imq.user_repository.ldap.grpbase	第 270 页上的表 14-7
imq.user_repository.ldap.grpfilter	第 270 页上的表 14-7
imq.user_repository.ldap.grpsearch	第 270 页上的表 14-7
imq.user_repository.ldap.memattr	第 270 页上的表 14-7
imq.user_repository.ldap.password	第 270 页上的表 14-7
imq.user_repository.ldap.principal	第 270 页上的表 14-7
imq.user_repository.ldap.*propertyName*	第 270 页上的表 14-7
imq.user_repository.ldap.server	第 270 页上的表 14-7
imq.user_repository.ldap.ssl.enabled	第 270 页上的表 14-7
imq.user_repository.ldap.timeout	第 270 页上的表 14-7
imq.user_repository.ldap.uidattr	第 270 页上的表 14-7
imq.user_repository.ldap.usrfilter	第 270 页上的表 14-7

第 15 章

物理目标属性参考

本章提供了有关物理目标配置属性的参考信息。创建或更新物理目标时可设置这些属性。对于自动创建的目标，可以在代理的实例配置文件中设置默认值（请参见 第 262 页上的表 14-3）。

表 15-1 物理目标属性

属性	类型	默认值	描述
maxNumMsgs[1]	整数	-1	未使用消息的最大数量 值为 -1 表示不限制消息数量。 对于停用消息队列，默认值为 1000。
maxBytesPerMsg	字符串	-1	任何单条消息的最大大小（以字节为单位） 将持久性消息被拒绝的消息报告给生成方客户端，并报告异常；对于非持久性消息，则不发送任何通知。 可以使用以下后缀（分别表示字节、千字节或兆字节）来表示该值： b 字节 k 千字节（1024 字节） m 兆字节（1024 x 1024 = 1,048,576 字节） 不带后缀的值用字节表示；值为 -1 表示不限制消息大小。 示例： 1600 1600 字节 1600b 1600 字节 16k 16 千字节（= 16,384 字节） 16m 16 兆字节（= 16,777,216 字节） -1 无限制
maxTotalMsgBytes[1]	字符串	-1	未使用的消息最多可占用的总内存（以字节为单位） 语法与 maxBytesPerMsg 的语法相同（请参见上文）。 对于停用消息队列，默认值为 10m。

表 15-1 物理目标属性 （续）

属性	类型	默认值	描述
limitBehavior	字符串	REJECT_NEWEST	当达到内存限制的阈值时，代理的行为如下： FLOW_CONTROL 减慢生成方。 REMOVE_OLDEST 丢弃最旧的消息。 REMOVE_LOW_PRIORITY 根据消息存在的时间丢弃优先级最低的消息；生成方客户端不会收到通知 REJECT_NEWEST 拒绝最新消息；仅当消息是持久性消息时，才通知生成方客户端，并报告异常。 如果值为 REMOVE_OLDEST 或 REMOVE_LOW_PRIORITY，并且 useDMQ 属性为 true，则将超出数量限制的消息将移动到停用消息队列中。对于停用消息队列本身，默认的限制行为是 REMOVE_OLDEST，并且不能设置为 FLOW_CONTROL。
maxNumProducers[2]	整数	-1	目标的消息生成方的最大数量 当达到此限制时，将无法创建新的生成方。值为 -1 表示不限制生成方数量。
maxNumActiveConsumers[3]	整数	1	来自队列目标的负载平衡传送中活动消息使用方的最大数量 值为 -1 表示不限制使用方数量。在 Sun Java System Message Queue Platform Edition 中，该值被限制为 2。
maxNumBackupConsumers[3]	整数	0	来自队列目标的负载平衡传送中的备份消息使用方的最大数量 值为 -1 表示不限制使用方数量。在 Sun Java System Message Queue Platform Edition 中，该值被限制为 1。
consumerFlowLimit	整数	1000	在一批中传送给使用方的消息的最大数量 在负载平衡队列传送中，该值为负载平衡开始之前路由至活动使用方的队列消息的初始数量。目标使用方可以通过在连接中指定一个较小的值来覆盖此限制。 值为 -1 表示不限制使用方数量。
isLocalOnly[2]	布尔值	false	仅限本地传送？ 此属性仅适用于代理群集中的目标，目标一旦创建，此属性即无法更改。如果为 true，将不在其他代理上复制目标，并限制目标只将消息传送到本地使用方 （那些连接到创建该目标的代理的使用方）。
localDeliveryPreferred[2,3]	布尔值	false	是否首选本地传送？ 此属性仅适用于代理群集中的负载平衡队列传送。如果为 true，则仅当本地代理中没有使用方时，才将消息传送给远程使用方；不得将目标限制为仅本地传送 （isLocalOnly 必须为 false）。

表 15-1 物理目标属性（续）

属性	类型	默认值	描述
useDMQ[2]	布尔值	true	是否将停用消息发送到停用消息队列？ 如果为 false，将仅仅丢弃停用消息。

1. 在群集环境中，应用于目标的各个单独实例，而不是统一应用于群集中的所有实例。
2. 不适用于停用消息队列。
3. 仅队列目标

第 16 章

受管理对象属性参考

本章提供了有关受管理对象属性的参考信息。本章包含以下各节：

连接工厂属性

连接工厂对象的属性分为不同类别，如以下各节所述：

连接处理

表 16-1 列出了用于连接处理的连接工厂属性。

表 16-1 用于连接处理的连接工厂属性

属性	类型	默认值	描述
imqAddressList	字符串	现有的 Message Queue 3.0 地址（如果有）；如果没有，则使用第 289 页上的表 16-2 中的第一个条目。	代理地址列表 该列表包含由逗号分隔的一个或多个消息服务器地址。每个地址指定（或表示）客户端可以连接的代理实例的主机名、端口号和连接服务。地址语法因连接服务及端口分配方法而异；有关详细信息，请参见下文。
imqAddressListBehavior	字符串	PRIORITY	尝试连接到服务器地址的顺序： PRIORITY 地址列表中指定的顺序 RANDOM 随机顺序 **注：**如果多个客户端共享同一连接工厂，则指定随机连接顺序可防止所有客户端尝试连接到同一地址。
imqAddressListIterations	整数	5	重复访问地址列表以尝试建立或重新建立连接的次数。 值为 -1 表示不限制重复次数。
imqPingInterval	整数	30	测试客户端与代理之间的连接的时间间隔（以秒为单位） 如果值为 0 或 -1，则禁止定期测试连接。
imqReconnectEnabled	布尔值	false	是否尝试重新建立断开的连接？
imqReconnectAttempts	整数	0	尝试连接（或重新连接）到地址列表中每个地址的次数，超过该次数后，将移动到下一个地址。 值为 -1 表示不限制连接尝试次数；将反复尝试连接到第一个地址，直至成功。
imqReconnectInterval	长整型	3000	两次重新连接尝试的时间间隔（以毫秒为单位）。 此值适用于针对给定地址的连续尝试，也适用于列表中的连续地址。 **注：**此值过小可能会导致代理没有足够的恢复时间；而此值过大则可能会导致无法接受的连接延迟。
imqSSLIsHostTrusted	布尔值	true	是否接受代理的自签名验证证书？ **注：**要使用来自证书颁发机构的签名证书，请将此属性设置为 false。

imqAddressList 属性的值是由逗号分隔的字符串，它指定要连接的一个或多个消息服务器地址。每个地址的一般语法如下：

scheme://*address*

其中 *scheme* 标识表 16-2 第一列中所示的寻址方案之一， *address* 表示服务器地址本身。用于指定地址的确切语法取决于寻址方案，如表中最后一列所示。

表 16-2 消息服务器寻址方案

方案	服务	语法	描述
mq	jms 或 ssljms	[*hostName*][:*portNumber*][/*serviceName*]	为 jms 或 ssljms 连接服务动态指定端口。 地址列表条目指定 Message Queue 端口映射器的主机名和端口号。端口映射器本身动态指定用于连接的端口。 默认值： *hostName* = localhost *portNumber* = 7676 *serviceName* = jms 对于 ssljms 连接服务，必须明确指定所有变量。
mqtcp	jms	*hostName*:*portNumber*/jms	使用 jms 连接服务连接到指定端口。 绕过端口映射器，直接与指定的主机名和端口号建立 TCP 连接。
mqssl	ssljms	*hostName*:*portNumber*/ssljms	使用 ssljms 连接服务连接到指定端口。 绕过端口映射器，直接与指定的主机名和端口号建立安全的 SSL 连接。
http	httpjms	http://*hostName*:*portNumber*/*contextRoot*/tunnel 如果多个代理实例使用同一隧道 Servlet，则使用以下语法将会连接到特定代理实例而非随机选择的实例： http://*hostName*:*portNumber*/*contextRoot*/tunnel?ServerName=*hostName*:*instanceName*	使用 httpjms 连接服务连接到指定端口。 与位于指定 URL 的 Message Queue 隧道 Servlet 建立 HTTP 连接。必须将代理配置为访问 HTTP 隧道 Servlet。

表 16-2 消息服务器寻址方案 （续）

方案	服务	语法	描述
https	httpsjms	https://*hostName*:*portNumber*/*contextRoot*/tunnel 如果多个代理实例使用同一隧道 Servlet，则使用以下语法将会连接到特定代理实例而非随机选择的实例： https://*hostName*:*portNumber*/*contextRoot*/tunnel?ServerName=*hostName*:*instanceName*	使用 httpsjms 连接服务连接到指定端口。 与位于指定 URL 的 Message Queue 隧道 Servlet 建立安全的 HTTPS 连接。必须将代理配置为访问 HTTPS 隧道 Servlet。

表 16-3 显示了各种地址格式的示例。

表 16-3 消息服务器地址示例

服务	代理主机	端口	示例地址
未指定	未指定	未指定	无地址 (mq://localHost:7676/jms)
未指定	指定的主机	未指定	myBkrHost (mq://myBkrHost:7676/jms)
未指定	未指定	指定的端口映射器端口	1012 (mq://localHost:1012/jms)
ssljms	本地主机	标准的端口映射器端口	mq://localHost:7676/ssljms
ssljms	指定的主机	标准的端口映射器端口	mq://myBkrHost:7676/ssljms
ssljms	指定的主机	指定的端口映射器端口	mq://myBkrHost:1012/ssljms
jms	本地主机	指定的服务端口	mqtcp://localhost:1032/jms
ssljms	指定的主机	指定的服务端口	mqssl://myBkrHost:1034/ssljms
httpjms	不适用	不适用	http://websrvr1:8085/imq/tunnel
httpsjms	不适用	不适用	https://websrvr2:8090/imq/tunnel

客户端身份验证

表 16-4 列出了用于客户端身份验证的连接工厂属性。

表 16-4 用于客户端身份验证的连接工厂属性

属性	类型	默认值	描述
imqDefaultUsername	字符串	guest	用于通过代理验证的默认用户名
imqDefaultPassword	字符串	guest	用于通过代理验证的默认密码
imqConfiguredClientID	字符串	null	以管理方式配置的客户端标识符
imqDisableSetClientID	布尔值	false	是否阻止客户端使用 setClientID 方法更改客户端标识符？

可靠性和流控制

表 16-5 列出了用于可靠性和流控制的连接工厂属性。

表 16-5 用于可靠性和流控制的连接工厂属性

属性	类型	默认值	描述
imqAckTimeout	字符串	0	等待代理确认的最长时间 （以毫秒为单位），超过该时间后，将抛出异常 值为 0 表示无超时 （无限期等待）。 **注：**在某些情况下，此值过低可能会导致提前超时：例如，第一次使用安全 (SSL) 连接对照 LDAP 用户系统信息库来验证用户时需要 30 秒以上的时间。
imqConnectionFlowCount	整数	100	计量批中有效负荷消息的数量 达到此消息数后，会暂停向客户端传送有效负荷消息，以便传送堆积的所有控制消息。当收到客户端运行时通知时，将恢复有效负荷消息传送，直到再次达到该消息数为止。 值为 0 将禁用消息传送计量，这可能会因为有效负荷消息流量过大而导致 Message Queue 控制消息被阻塞。
imqConnectionFlowLimitEnabled	布尔值	false	是否在连接级别限制消息流？

表 16-5 用于可靠性和流控制的连接工厂属性 （续）

属性	类型	默认值	描述
imqConnectionFlowLimit	整数	1000	每个连接中可以传送和缓冲 （以便使用）的最大消息数 如果待处理的未使用有效负荷消息数 （受流计量的约束，流计量由 imqConnectionFlowCount 控制）超过此限制，连接中的消息传送将会停止。仅当待处理的消息数低于该限制时，才会恢复传送。这样可防止客户端因待处理消息太多而过载，从而导致内存不足。 如果 imqConnectionFlowLimitEnabled 为 false，则忽略此属性。
imqConsumerFlowLimit	整数	100	每个使用方可以传送和缓冲 （以便使用）的最大消息数 对于给定的使用方，如果待处理的未使用有效负荷消息数超过此限制，将停止向该使用方传送消息。仅当该使用方的待处理消息数低于 imqConsumerFlowThreshold 指定的百分比时，才会恢复传送。使用此限制可以改进多个使用方之间的负载平衡，并防止同一连接上的任一使用方抢占其他使用方的流量。 如果为队列自身的 consumerFlowLimit 属性设置的值较低，则该值可以覆盖此限制 （请参见第 15 章 “物理目标属性参考”）。另请注意，针对连接上所有使用方的消息传送受 imqConnectionFlowLimit 指定的整体限制的约束。
imqConsumerFlowThreshold	整数	50	每个使用方可以在客户端运行时中缓冲的消息数 （以 imqConsumerFlowLimit 百分比的形式指定），低于该百分比时将恢复消息传送。

队列浏览器及服务器会话

表 16-6 中列出了用于队列浏览和服务器会话的连接工厂属性。

表 16-6 用于队列浏览器和服务器会话的连接工厂属性

属性	类型	默认值	描述
imqQueueBrowserMaxMessagesPerRetrieve	整数	1000	在浏览队列目标的内容时一次检索的最大消息数
imqQueueBrowserRetrieveTimeout	长整型	60000	在浏览队列目标的内容时等待检索消息的最长时间 （以毫秒为单位），超过此时间将会抛出异常。
imqLoadMaxToServerSession	布尔值	true	是否向服务器会话中加载最大数量的消息？ 如果为 false，则客户端一次只加载一条消息。 此属性仅适用于 JMS 应用服务器工具。

设置标准消息属性

表 16-7 中列出的连接工厂属性控制 Message Queue 客户端运行时是否设置 **Java 消息服务规范**中定义的某些标准消息属性。

表 16-7 用于设置标准消息属性的连接工厂属性

属性	类型	默认值	描述
imqSetJMSXUserID	布尔值	false	是否为生成的消息设置 JMSXUserID 属性 （发送消息的用户的标识）？
imqSetJMSXAppID	布尔值	false	是否为生成的消息设置 JMSXAppID 属性 （发送消息的应用程序的标识）？
imqSetJMSXProducerTXID	布尔值	false	是否为生成的消息设置 JMSXProducerTXID 属性 （生成消息的事务的事务标识符）？
imqSetJMSXConsumerTXID	布尔值	false	是否为使用的消息设置 JMSXConsumerTXID 属性 （使用消息的事务的事务标识符）？
imqSetJMSXRcvTimestamp	布尔值	false	是否为使用的消息设置 JMSXRcvTimestamp 属性 （消息传送到使用方的时间）？

消息头覆盖

表 16-8 列出了用于覆盖 JMS 消息头字段的连接工厂属性。

表 16-8 用于覆盖消息头的连接工厂属性

属性	类型	默认值	描述
imqOverrideJMSDeliveryMode	布尔值	false	是否允许覆盖客户端设置的传送模式？
imqJMSDeliveryMode	整数	2	传送模式的覆盖值： 1 非持久性 2 持久性
imqOverrideJMSExpiration	布尔值	false	是否允许覆盖客户端设置的到期时间？
imqJMSExpiration	长整型	0	到期时间的覆盖值 （以毫秒为单位） 值为 0 表示不限制到期时间 （消息永不过期）。
imqOverrideJMSPriority	布尔值	false	是否允许覆盖客户端设置的优先级？

表 16-8 用于覆盖消息头的连接工厂属性 （续）

属性	类型	默认值	描述
imqJMSPriority	整数	4 （标准）	优先级的覆盖值（0 到 9）
imqOverrideJMSHeadersToTemporaryDestinations	布尔值	false	是否对临时目标应用覆盖值？

目标属性

表 16-9 中列出了可以为目标受管理对象设置的属性。

表 16-9 目标属性

属性	类型	默认值	描述
imqDestinationName	字符串	Untitled_Destination_Object	物理目标的名称 目标名称只能包含字母数字字符（不包括空格），并且必须以字母字符开头，或者以下划线 (_) 或美元符号 ($) 开头。目标名称不得以字符 mq 开头。
imqDestinationDescription	字符串	无	目标的描述性字符串

SOAP 端点属性

表 16-10 中列出了用于为使用简单对象访问协议 (Simple Object Access Protocol, SOAP) 的应用程序配置端点 URL 的属性；有关更多信息，请参见 Message Queue Developer's Guide for Java Clients。

表 16-10 SOAP 端点属性

属性	类型	默认值	描述
imqSOAPEndpointList	字符串	无	由一个或多个以空格分隔的 URL 组成的列表，这些 URL 表示要向其发送消息的 SOAP 端点。 每个 URL 都应该与一个可以接收和处理 SOAP 消息的 Servlet 关联。 示例： `http://www.serv1/ http://www.serv2/` 如果该列表指定了多个 URL，则消息将广播到列表中的所有 URL。
imqEndpointName	字符串	Untitled_Endpoint_Object	SOAP 端点的名称
imqEndpointDescription	字符串	无	SOAP 端点的描述性字符串。 示例： `My endpoints for broadcast`（用于接收广播消息的端点）

第 17 章

JMS 资源适配器属性参考

利用 Message Queue JMS 资源适配器 (JMS Resource Adapter, JMS RA)，可以通过标准 J2EE 连接器体系结构 (J2EE Connector Architecture, JCA) 将 Sun Java System Message Queue 和任何 J2EE 1.4 应用服务器集成。将 Message Queue JMS 资源适配器插入应用服务器后，在该应用服务器中部署的应用程序即可使用 Message Queue 来发送和接收 JMS 消息。

Message Queue JMS 资源适配器通过以下三个 JavaBean 组件来公开其配置属性：

- `ResourceAdapter` 配置从总体上影响资源适配器的行为。
- `ManagedConnectionFactory` 配置影响资源适配器创建的用于消息驱动 Bean (Message Driven Bean, MDB) 的连接。
- `ActivationSpec` 配置影响消息端点，这些消息端点在它们与消息传送系统的交互中表示消息驱动 Bean (Message Driven Bean, MDB)。

要设置这些组件的属性值，请使用应用服务器提供的用于配置和部署资源适配器及部署 MDB 的工具。

本章列出并描述了 Message Queue JMS 资源适配器的配置属性。本章包含以下各节：

ResourceAdapter JavaBean

`ResourceAdapter` 配置用于配置默认 JMS 资源适配器行为。表 17-1 列出并描述了可用来配置此 JavaBean 的属性。带脚注的属性是必需属性。

表 17-1 资源适配器属性

属性	默认值	描述
addressList[1]	mq://localhost:7676/jms	资源适配器建立的到 Message Queue 服务的连接，它是使用消息服务地址格式指定的。 资源适配器提供默认值。 属性名 addressList 是 Sun Java System Message Queue 特有的，但与标准属性 connectionURL 具有相同的含义。Sun Java System Message Queue 同时提供这两个属性名。只能设置 connectionURL 和 addressList 中的一个；它们是等效的。
addressListBehavior	PRIORITY	一个字符串，指定资源适配器连接到 Message Queue 服务的方式。值为 PRIORITY 或 RANDOM。 PRIORITY 连接通过选择地址列表 (addressList) 中指定的第一个条目来选择 Message Queue 代理。 RANDOM 连接从地址列表中随机选择 Message Queue 代理。 连接中断后的重新连接对于 PRIORITY 和 RANDOM 是相同的。尝试重新连接时，将从连接中断的代理开始。如果该尝试失败，资源适配器将依次尝试活动地址列表中的后续条目。
addressListIterations	1	循环访问地址列表的次数。该值适用于初始连接以及后来的重新连接尝试。
connectionURL	mq://localhost:7676/jms	资源适配器建立的到 Message Queue 服务的连接，它是使用消息服务地址格式指定的。 等效于 addressList 属性；请参见上面的说明以了解更详细的信息。
userName[1]	guest	资源适配器连接到 Message Queue 服务时使用的默认用户名。 资源适配器提供默认值。
password[1]	guest	资源适配器连接到 Message Queue 服务时使用的默认密码。 资源适配器提供默认值。
reconnectAttempts	6	尝试重新连接到地址列表中的单个条目的次数。当 reconnectEnabled 设置为 true 时，将使用此属性。
reconnectEnabled	false	一个布尔值，指定在连接中断后是否尝试重新连接。 重新连接尝试的行为由 reconnectInterval 和 reconnectAttempts 的值控制。
reconnectInterval	30000	两次重新连接尝试的间隔时间 （以毫秒为单位）。当 reconnectEnabled 设置为 true 时，将使用此属性。

1. 此属性是必需的。

ManagedConnectionFactory JavaBean

受管理连接工厂提供并定义资源适配器为消息驱动 Bean 提供的连接。对于所设置的属性，如果 ResourceAdapter JavaBean 具有类似属性，则该设置将取代为 ResourceAdapter Bean 指定的类似值。

表 17-2 列出并描述了 Message Queue 资源适配器提供的受管理连接工厂的可配置属性。

表 17-2 受管理连接工厂属性

属性	默认值	描述
addressList	无	一个派生自此受管理连接工厂的连接列表。 此属性的格式遵循消息服务 addressList，如第 298 页上的表 17-1 中所述。如果未设置该值，连接将使用为 ResourceAdapter JavaBean 指定的 addressList 值 （如该表中所述）。
addressListBehavior	PRIORITY	一个字符串，指定资源适配器连接到 Message Queue 服务的方式。值为 PRIORITY 或 RANDOM。 PRIORITY 连接通过选择地址列表 (addressList) 中指定的第一个条目来选择 Message Queue 代理。 RANDOM 连接从地址列表中随机选择 Message Queue 代理。 连接中断后的重新连接对于 PRIORITY 和 RANDOM 是相同的。尝试重新连接时，将从连接中断的代理开始。如果该尝试失败，资源适配器将依次尝试活动地址列表中的后续条目。
addressListIterations	1	循环访问地址列表的次数。该值适用于初始连接以及后来的重新连接尝试。
clientID	无	用于派生自此受管理连接工厂的连接的客户端标识符。
password	guest	**（可选）**用于连接的密码。 如果未设置该值，连接将使用为 ResourceAdapter JavaBean 指定的密码 （如第 298 页上的表 17-1 中所述）。
reconnectAttempts	6	尝试重新连接到地址列表中的单个条目的次数。
reconnectEnabled	false	布尔值，指定在连接中断后是尝试重新连接，还是尝试建立新的连接。 重新连接尝试是由 reconnectInterval 和 reconnectAttempts 属性控制的。
reconnectInterval	30000	在尝试重新连接到 Message Queue 服务之前等待的最短时间（以毫秒为单位）。

表 17-2 受管理连接工厂属性 （续）

属性	默认值	描述
userName	guest	**（可选）**用于连接的用户名。 如果未设置该值，连接将使用为 ResourceAdapter JavaBean 指定的用户名 （如第 298 页上的表 17-1 中所述）。

ActivationSpec JavaBean

当应用服务器指示资源适配器激活消息端点并将消息端点与消息驱动 Bean 关联时，需要使用 ActivationSpec JavaBean 属性。

表 17-3 列出并描述了消息端点激活规范的可配置属性。该表列出了 Message Queue 资源适配器的特有属性以及企业 JavaBean 2.1 标准或 J2EE 连接器体系结构 (J2EE Connector Architecture, J2EE CA) 1.5 标准的特有属性。

表 17-3 激活规范属性

属性	默认值	描述
acknowledgeMode	Auto-acknowledge	**（可选）**用于使用方的 JMS 会话确认模式。 这是标准的 EJB 2.1 和 J2EE CA 1.5 属性。 该值可以是 Auto-acknowledge 或 Dups-ok-acknowledge。
addressList	从 ResourceAdapter JavaBean 配置中的 addressList 继承。	**（可选）**资源适配器代表消息端点所建立的连接的规范。 此属性是 Message Queue JMS 资源适配器的特有属性。 有效值必须符合消息服务连接地址语法。
clientId	无	为该使用方创建的 JMS 连接所使用的 JMS 客户端 ID。 如果将 subscriptionDurability 属性设置为 Durable，则必须设置此属性。 这是标准的 EJB 2.1 和 J2EE CA 1.5 属性。
customAcknowledgeMode	无	一个字符串，指定 MDB 消息使用模式。 此属性的有效值是 No_acknowledge 或 null。 只能将 No_acknowledge 模式用于非事务、非长期的主题订阅。如果对事务订阅或长期订阅使用此设置，订阅激活将失败。

表 17-3 激活规范属性 （续）

属性	默认值	描述
ƒø±Í	无	目标的名称，此 MDB 使用该目标中的消息。 这是必需的属性。它是标准的 EJB 2.1 和 J2EE CA1.5 属性。 对于 Message Queue 目标受管理对象，该值必须设置为 destinationName 属性的值。
destinationType	无	destination 属性指定的目标类型。有效值是 javax.jms.Queue 或 javax.jms.Topic。 这是必需的属性。它是标准的 EJB 2.1 和 J2EE CA1.5 属性。
endpointExceptionRedelivery Attempts	6	消息传送期间当 MDB 抛出异常时，向 MDB 重新传送消息的次数。
messageSelector	无	**（可选）**用于对传送到使用方的消息进行过滤的 JMS 消息选择器。该值是字符串类型。 这是标准的 EJB 2.1 和 J2EE CA 1.5 属性。
sendUndeliverableMsgsToDMQ	true	一个布尔值，指定当 MDB 抛出运行时异常且重新传送尝试次数超出 endpointExceptionRedeliveryAttempts 值时，是否将消息置于停用消息队列中。 如果为 false， Message Queue 代理将尝试将消息重新传送到任何有效的使用方，包括同一个 MDB。
subscriptionDurability	NonDurable	一个字符串，指定某一主题目标的使用方是长期的还是非长期的。该值可以是 NonDurable 或 Durable。 对于非长期订阅，此属性是可选的；对于长期订阅，此属性是必需的。如果将该值设置为 Durable，则还必须设置属性 clientID 和 subscriptionName。 这是标准的 EJB 2.1 和 J2EE CA1.5 属性，只有当 destinationType 属性设置为 avax.jms.Topic 时才有效。
subscriptionName	无	一个字符串，用于命名长期订阅。 如果 subscriptionDurability 属性设置为 Durable，则必须设置此属性。 这是标准的 EJB 2.1 和 J2EE CA 1.5 属性。

第 18 章

度量参考

本章列出并说明了 Message Queue 产品生成的度量。本章包含以下各节：

JVM 度量

表 18-1 列出并说明了代理为代理进程 JVM 堆生成的度量数据。对于每个度量，该表都显示了提供该度量的度量监视工具。

表 18-1 JVM 度量

度量数量	描述	imqcmd `metrics bkr` (*metricType*)	日志文件	度量消息（度量主题）[2]
JVM 堆：可用内存	可用于 JVM 堆的内存量	是 (cxn)	是	是 (...jvm)
JVM 堆：总内存	当前 JVM 堆大小	是 (cxn)	是	是 (...jvm)
JVM 堆：最大内存	JVM 堆大小可以达到的最大值。	否	是[1]	是 (...jvm)

1. 仅在代理启动时显示。
2. 有关度量主题目标的名称，请参见第 184 页上的表 10-7。

代理范围内的度量

表 18-2 列出并说明了代理所报告的有关代理范围内的度量信息的数据。它还显示了哪些数据可以使用不同的度量监视工具来获得。

表 18-2 代理范围内的度量

度量数量	描述	imqcmd `metrics bkr` (*metricType*)	日志文件	度量消息（度量主题）[1]
连接数据				
连接数	当前对代理打开的连接数	是 (`cxn`)	是	是 (...broker)
线程数	所有连接服务当前使用的线程总数	是 (`cxn`)	是	否
最小线程数	特定的线程数，达到该数量后，线程池中将保持此数量的线程供连接服务使用	是 (`cxn`)	是	否
最大线程数	特定的线程数，达到该数量后，线程池中将不会增加新的线程以供连接服务使用	是 (`cxn`)	是	否
存储消息数据				
消息数	当前存储在代理内存和持久性存储中的有效负荷消息数	否 使用 `query bkr`	否	是 (...broker)
消息字节总数	当前存储在代理内存和持久性存储中的有效负荷消息字节数	否 使用 `query bkr`	否	是 (...broker)
消息流数据				
流入消息数	自代理上次启动以来流入该代理的有效负荷消息数	是 (`ttl`)	是	是 (...broker)
流入消息字节	自代理上次启动以来流入代理的有效负荷消息字节数	是 (`ttl`)	是	是 (...broker)
流入包数	自代理上次启动以来流入该代理的包数 （包括有效负荷消息和控制消息）	是 (`ttl`)	是	是 (...broker)
流入包字节	自代理上次启动以来流入该代理的包字节数（包括有效负荷消息和控制消息）	是 (`ttl`)	是	是 (...broker)
流出消息数	自代理上次启动以来流出该代理的有效负荷消息数	是 (`ttl`)	是	是 (...broker)
流出消息字节	自代理上次启动以来流出该代理的有效负荷消息字节数	是 (`ttl`)	是	是 (...broker)

表 18-2 代理范围内的度量 （续）

度量数量	描述	imqcmd **metrics bkr** (*metricType*)	日志文件	度量消息（度量主题）[1]
流出包数	自代理上次启动以来流出该代理的包数 （包括有效负荷消息和控制消息）	是 (ttl)	是	是 (...broker)
流出包字节	自代理上次启动以来流出该代理的包字节数 （包括有效负荷消息和控制消息）	是 (ttl)	是	是 (...broker)
消息流入速率	有效负荷消息流入代理的当前速率	是 (rts)	是	否
消息字节流入速率	有效负荷消息字节流入代理的当前速率	是 (rts)	是	否
包流入速率	包流入代理的当前速率 （包括有效负荷消息和控制消息）	是 (rts)	是	否
包字节流入速率	包字节流入代理的当前速率 （包括有效负荷消息和控制消息）	是 (rts)	是	否
消息流出速率	有效负荷消息流出代理的当前速率	是 (rts)	是	否
消息字节流出速率	有效负荷消息字节流出代理的当前速率	是 (rts)	是	否
包流出速率	包流出代理的当前速率 （包括有效负荷消息和控制消息）	是 (rts)	是	否
包字节流出速率	包字节流出代理的当前速率 （包括有效负荷消息和控制消息）	是 (rts)	是	否
目标数据				
目标数	代理中的物理目标数	否	否	是 (...broker)

1. 有关度量主题目标的名称，请参见第 184 页上的表 10-7。

连接服务度量

表 18-3 列出并说明了代理为各个连接服务所报告的度量数据。它还显示了哪些数据可以使用不同的度量监视工具来获得。

表 18-3 连接服务度量

度量数量	描述	imqcmd **metrics svc** (*metricType*)	日志文件	度量消息（度量主题）
连接数据				
连接数	当前打开的连接数	是 (cxn) 还有 query svc	否	否
线程数	当前使用的线程数	是 (cxn) 还有 query svc	否	否
最小线程数	供所有连接服务使用的特定总线程数，达到该数量后，线程池中将保持此数量的线程供连接服务使用	是 (cxn)	否	否
最大线程数	供所有连接服务使用的特定总线程数，达到该数量后，线程池中将不会增加新的线程以供连接服务使用	是 (cxn)	否	否
消息流数据				
流入消息数	自代理上次启动以来流入连接服务的有效负荷消息数	是 (ttl)	否	否
流入消息字节	自代理上次启动以来流入连接服务的有效负荷消息字节数	是 (ttl)	否	否
流入包数	自代理上次启动以来流入连接服务的包数（包括有效负荷消息和控制消息）	是 (ttl)	否	否
流入包字节	自代理上次启动以来流入连接服务的包字节数 （包括有效负荷消息和控制消息）	是 (ttl)	否	否
流出消息数	自代理上次启动以来流出连接服务的有效负荷消息数	是 (ttl)	否	否
流出消息字节	自代理上次启动以来流出连接服务的有效负荷消息字节数	是 (ttl)	否	否
流出包数	自代理上次启动以来流出连接服务的包数（包括有效负荷消息和控制消息）	是 (ttl)	否	否
流出包字节	自代理上次启动以来流出连接服务的包字节数 （包括有效负荷消息和控制消息）	是 (ttl)	否	否
消息流入速率	有效负荷消息通过连接服务流入代理的当前速率。	是 (rts)	否	否
消息字节流入速率	有效负荷消息字节流入连接服务的当前速率	是 (rts)	否	否

表 18-3 连接服务度量 （续）

度量数量	描述	imqcmd metrics svc (*metricType*)	日志文件	度量消息（度量主题）
包流入速率	包流入连接服务的当前速率 （包括有效负荷消息和控制消息）	是 (rts)	否	否
包字节流入速率	包字节流入连接服务的当前速率 （包括有效负荷消息和控制消息）	是 (rts)	否	否
消息流出速率	有效负荷消息流出连接服务的当前速率	是 (rts)	否	否
消息字节流出速率	有效负荷消息字节流出连接服务的当前速率	是 (rts)	否	否
包流出速率	包流出连接服务的当前速率 （包括有效负荷消息和控制消息）	是 (rts)	否	否
包字节流出速率	包字节流出连接服务的当前速率 （包括有效负荷消息和控制消息）	是 (rts)	否	否

目标度量

表 18-4 列出并说明了代理为各个目标所报告的度量数据。它还显示了哪些数据可以使用不同的度量监视工具来获得。

表 18-4 目标度量

度量数量	描述	imqcmd metrics dst (*metricType*)	日志文件	度量消息（度量主题）[1]
使用方数据				
使用方数	当前的使用方数 对于主题，该值包括非长期订阅、活动的长期订阅以及非活动的长期订阅。对于队列，该值包括活动使用方和备份使用方。	是 (con)	否	是 (...*destName*)
平均使用方数	自代理上次启动以来的平均使用方数	是 (con)	否	是 (...*destName*)
最大使用方数	自代理上次启动以来的最大使用方数	是 (con)	否	是 (_*destName*)
活动使用方数	当前活动使用方数	是 (con)	否	是 (...*destName*)

表 18-4 目标度量 （续）

度量数量	描述	imqcmd metrics dst (*metricType*)	日志文件	度量消息（度量主题）[1]
平均活动使用方数	自代理上次启动以来的平均活动使用方数	是 (con)	否	是 (*_destName*)
最大活动使用方数	自代理上次启动以来的最大活动使用方数	是 (con)	否	是 (*_destName*)
备份使用方数	当前备份使用方数 （仅适用于队列）	是 (con)	否	是 (*_destName*)
平均备份使用方数	自代理上次启动以来的平均备份使用方数（仅适用于队列）	是 (con)	否	是 (*_destName*)
最大备份使用方数	自代理上次启动以来的最大备份使用方数（仅适用于队列）	是 (con)	否	是 (*_destName*)
存储消息数据				
消息数	当前存储在目标内存和持久性存储中的有效负荷消息数	是 (con) (ttl) (rts) 还有 query dst	否	是 (*_destName*)
平均消息数	自代理上次启动以来存储在目标内存和持久性存储中的平均有效负荷消息数	是 (con) (ttl) (rts)	否	是 (*_destName*)
最大消息数	自代理上次启动以来存储在目标内存和持久性存储中的最大有效负荷消息数	是 (con) (ttl) (rts)	否	是 (*_destName*)
消息字节总数	当前存储在目标内存和持久性存储中的有效负荷消息字节数	是 (ttl) (rts) 还有 query dst	否	是 (*_destName*)
平均消息字节总数	自代理上次启动以来存储在目标内存和持久性存储中的平均有效负荷消息字节数	是 (ttl) (rts)	否	是 (*_destName*)
最大消息字节总数	自代理上次启动以来存储在目标内存和持久性存储中的最大有效负荷消息字节数	是 (ttl) (rts)	否	是 (*_destName*)
最大消息字节	自代理上次启动以来目标接收到的单个消息中有效负荷消息的最大字节数	是 (ttl) (rts)	否	是 (*_destName*)

表 18-4 目标度量 （续）

度量数量	描述	imqcmd `metrics dst` (*metricType*)	日志文件	度量消息（度量主题）[1]
消息流数据				
流入消息数	自代理上次启动以来流入此目标的有效负荷消息数	是 (`ttl`)	否	是 (*_destName*)
流入消息字节	自代理上次启动以来流入此目标的有效负荷消息字节数	是 (`ttl`)	否	是 (*_destName*)
流出消息数	自代理上次启动以来流出此目标的有效负荷消息数	是 (`ttl`)	否	是 (*_destName*)
流出消息字节	自代理上次启动以来流出此目标的有效负荷消息字节数	是 (`ttl`)	否	是 (*_destName*)
消息流入速率	有效负荷消息流入目标的当前速率	是 (`rts`)	否	否
消息流出速率	有效负荷消息流出目标的当前速率	是 (`rts`)	否	否
消息字节流入速率	有效负荷消息字节流入目标的当前速率	是 (`rts`)	否	否
消息字节流出速率	有效负荷消息字节流出目标的当前速率	是 (`rts`)	否	否
磁盘使用情况数据				
保留的磁盘	目标存储 （基于文件）中所有消息记录（活动的和空闲的）使用的磁盘空间 （以字节为单位）	是 (`dsk`)	否	是 (*...destName*)
已用的磁盘	目标存储 （基于文件）中活动消息记录使用的磁盘空间 （以字节为单位）	是 (`dsk`)	否	是 (*_destName*)
磁盘占用率	已用磁盘空间与保留磁盘空间的比率。比率越高，用于容纳活动消息的磁盘空间就越多	是 (`dsk`)	否	是 (*_destName*)

1. 有关度量主题目标的名称，请参见第 184 页上的表 10-7。

第 IV 部分

附录

Message Queue 数据在特定平台上的位置

Sun Java System Message Queue 数据在不同的操作系统平台上的存储位置也不同。以下各表说明了各种类型的 Message Queue 数据在下列平台上的位置：

在这些表中， *instanceName* 表示与数据关联的代理实例的名称。

Solaris

表 A-1 中显示了 Message Queue 数据在 Solaris 操作系统上的位置。如果在装有 Sun Java System Application Server 独立版的 Solaris 上使用 Message Queue，目录结构将与第 316 页上的 "Windows" 中所述的目录结构类似。

表 A-1 Message Queue 数据在 Solaris 平台上的位置

数据类别	位置
代理实例配置属性	/var/imq/instances/*instanceName*/props/config.properties
代理配置文件模板	/usr/share/lib/imq/props/broker/
持久性存储 （消息、目标、长期订阅和事务）	/var/imq/instances/*instanceName*/fs350 或 JDBC 可访问的数据存储
代理实例日志文件目录 （默认位置）	/var/imq/instances/*instanceName*/log/
受管理对象 （对象存储）	您选择的本地目录或 LDAP 服务器
安全性：用户系统信息库	/var/imq/instances/*instanceName*/etc/passwd 或 LDAP 服务器
安全性：访问控制文件 （默认位置）	/var/imq/instances/*instanceName*/etc/accesscontrol.properties
安全性：密码文件目录 （默认位置）	/var/imq/instances/*instanceName*/etc/
安全性：示例密码文件	/etc/imq/passfile.sample
安全性：代理的密钥存储文件位置	/etc/imq/
JavaDoc API 文档	/usr/share/javadoc/imq/index.html
示例应用程序和配置	/usr/demo/imq/
Java 归档 (.jar) 文件、 Web 归档 (.war) 文件和资源适配器归档 (.rar) 文件	/usr/share/lib/

Linux

表 A-2 中显示了 Message Queue 数据在 Linux 操作系统上的位置。

表 A-2 Message Queue 数据在 Linux 平台上的位置

数据类别	位置
代理实例配置属性	/var/opt/sun/mq/instances/*instanceName*/props/config.properties
代理配置文件模板	/opt/sun/mq/private/share/lib/props/
持久性存储 （消息、目标、长期订阅和事务）	/var/opt/sun/mq/instances/*instanceName*/fs350/ 或 JDBC 可访问的数据存储
代理实例日志文件目录 （默认位置）	/var/opt/sun/mq/instances/*instanceName*/log/
受管理对象 （对象存储）	您选择的本地目录或 LDAP 服务器
安全性：用户系统信息库	/var/opt/sun/mq/instances/*instanceName*/etc/passwd 或 LDAP 服务器
安全性：访问控制文件 （默认位置）	/var/opt/sun/mq/instances/*instanceName*/etc/accesscontrol.properties
安全性：密码文件目录 （默认位置）	/var/opt/sun/mq/instances/*instanceName*/etc/
安全性：示例密码文件	/etc/opt/sun/mq/passfile.sample
安全性：代理的密钥存储文件位置	/etc/opt/sun/mq/
JavaDoc API 文档	/opt/sun/mq/javadoc/index.html
示例应用程序和配置	/opt/sun/mq/examples/
Java 归档 (.jar) 文件、 Web 归档 (.war) 文件和资源适配器归档 (.rar) 文件	/opt/sun/mq/share/lib/
共享库 (.so) 文件	/opt/sun/mq/lib/

Windows

表 A-3 中显示了 Message Queue 数据在 Windows 操作系统上的位置。如果 Message Queue 与 Sun Java System Application Server 独立版捆绑在一起，该表也适用于 Solaris 平台。该版本 Application Server 既不与 Solaris 捆绑在一起，也不与 Sun Java Enterprise System 捆绑在一起。可以使用表 A-3 中的路径名，但要将斜杠字符的方向从 Windows 中的反斜杠 (\) 更改为 Solaris 中的正斜杠 (/)。有关 IMQ_HOME 和 IMQ_VARHOME 目录变量的定义，请参见第 22 页上的 “目录变量约定”。

表 A-3 Message Queue 数据在 Windows 平台上的位置

数据类别	位置
代理实例配置属性	IMQ_VARHOME\instances*instanceName*\props\config.properties
代理配置文件模板	IMQ_HOME\lib\props\broker\
持久性存储 （消息、目标、长期订阅和事务）	IMQ_VARHOME\instances*instanceName*\fs350\ 或 JDBC 可访问的数据存储
代理实例日志文件目录 （默认位置）	IMQ_VARHOME\instances*instanceName*\log\
受管理对象 （对象存储）	您选择的本地目录或 LDAP 服务器
安全性：用户系统信息库	IMQ_VARHOME\instances*instanceName*\etc\passwd 或 LDAP 服务器
安全性：访问控制文件 （默认位置）	IMQ_VARHOME\instances*instanceName*\etc\accesscontrol.properties
安全性：密码文件目录 （默认位置）	IMQ_HOME\etc\
安全性：示例密码文件	IMQ_HOME\etc\passfile.sample
安全性：代理的密钥存储文件位置	IMQ_HOME\etc\
JavaDoc API 文档	IMQ_HOME\javadoc\index.html
示例应用程序和配置	IMQ_HOME\demo\
Java 归档 (.jar) 文件、Web 归档 (.war) 文件和资源适配器归档 (.rar) 文件	IMQ_HOME\lib\

Message Queue 接口的稳定性

Sun Java System Message Queue 使用的许多接口可以帮助管理员自动完成任务。本附录根据这些接口的稳定性对其进行分类。接口越稳定，在本产品的后续版本中对其进行更改的可能性就越小。

本附录中未列出的接口都是专用的，而非供用户使用的。

表 B-1 说明了稳定性分类方案。

表 B-1 接口稳定性分类方案

分类	描述
专用	用户无法直接使用。在任何发行版中都可能对其进行改进或将其删除。
开发中	可供用户使用。可能会在主要发行版 （如 3.0 和 4.0）或次要发行版 （如 3.1 和 3.2）中做不兼容更改。所有更改都将慎重缓慢地进行。虽然我们会尽量保证所有更改都是兼容的，但不能保证做到这一点。
稳定	可供用户使用。只在主要发行版 （如 3.0 和 4.0）中做不兼容更改。
标准	可供用户使用。这些接口根据官方认可的标准进行定义，由标准组织控制。很少对这些接口做不兼容更改。
不稳定	可供用户使用。可能会在主要发行版 （如 3.0 和 4.0）或次要发行版 （如 3.1 和 3.2）中做不兼容更改。用户需注意，在以后的发行版中可能会删除这些接口，也可能以一种不兼容的方式大幅度更改这些接口。建议用户不要在不稳定的接口上创建显式相关性。

表 B-2 列出了这些接口及其分类。

表 B-2 Message Queue 接口的稳定性

接口	分类
命令行接口	
imqbrokerd 命令行接口	开发中
imqadmin 命令行接口	不稳定
imqcmd 命令行接口	开发中
imqdbmgr 命令行接口	不稳定
imqkeytool 命令行接口	开发中
imqobjmgr 命令行接口	开发中
imqusermgr 命令行接口	不稳定
imqbrokerd、imqadmin、imqcmd、imqdbmgr、imqkeytool、imqobjmgr 和 imqusermgr 的输出	不稳定
命令	
imqobjmgr 命令文件	开发中
imqbrokerd 命令	稳定
imqadmin 命令	不稳定
imqcmd 命令	稳定
imqdbmgr 命令	不稳定
imqkeytool 命令	稳定
imqobjmgr 命令	稳定
imqusermgr 命令	不稳定
API	
JMS API (javax.jms)	标准
JAXM API (javax.xml)	标准
C-API	开发中
C-API 环境变量	不稳定
基于消息的监视 API	开发中
受管理对象 API (com.sun.messaging)	开发中
.jar 和 .war 文件	
imq.jar 位置和名称	稳定

表 B-2 Message Queue 接口的稳定性 （续）

接口	分类
jms.jar 位置和名称	开发中
imqbroker.jar 位置和名称	专用
imqutil.jar 位置和名称	专用
imqadmin.jar 位置和名称	专用
imqservlet.jar 位置和名称	开发中
imqhttp.war 位置和名称	开发中
imqhttps.war 位置和名称	开发中
imqjmsra.rar 位置和名称	开发中
imqxm.jar 位置和名称	开发中
jaxm-api.jar 位置和名称	开发中
saaj-api.jar 位置和名称	开发中
saaj-impl.jar 位置和名称	开发中
activation.jar 位置和名称	开发中
mail.jar 位置和名称	开发中
dom4j.jar 位置和名称	专用
fscontext.jar 位置和名称	不稳定
文件	
代理日志文件的位置和内容格式	不稳定
密码文件	不稳定
accesscontrol.properties 文件	不稳定
系统目标	
mq.sys.dmq 目标	稳定
mq.metrics.* 目标	开发中
配置属性	
Message Queue JMS 资源适配器配置属性	开发中
Message Queue JMS 资源适配器 JavaBean 和 ActivationSpec 配置属性	开发中

表 B-2 Message Queue 接口的稳定性 （续）

接口	分类
消息属性和格式	
停用消息队列消息属性 JMSXDeliveryCount	标准
停用消息队列消息属性 JMS_SUN_*	开发中
Message Queue 客户端消息属性：JMS_SUN_*	开发中
度量或监视消息的 JMS 消息格式	开发中
杂项	
Message Queue JMS 资源适配器软件包 com.sun.messaging.jms.ra	开发中
用于存储持久性消息的 JDBC 模式	开发中

HTTP/HTTPS 支持

Message Queue Enterprise Edition 支持 Java 客户端通过 HTTP 或安全 HTTP (Secure HTTP, HTTPS) 传输与代理通信，而不是通过直接的 TCP 连接与代理通信。HTTP/HTTPS 支持不适用于 C 客户端。

本附录介绍了启用此支持所使用的体系结构，并说明了如何设置客户端，使之能够使用基于 HTTP 的连接进行 Message Queue 消息传送。本附录包含以下各节：

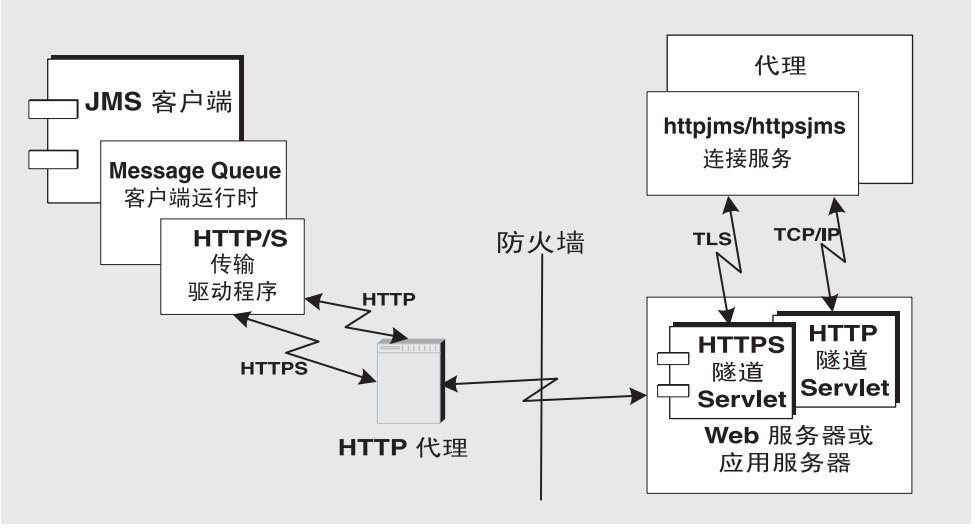
HTTP/HTTPS 支持体系结构

Message Queue 消息传送可以运行于 HTTP/HTTPS 连接之上。因为 HTTP/HTTPS 连接通常可以穿过防火墙，因此可以通过防火墙将客户端应用程序与代理隔开。

第 322 页上的图 C-1 列出了提供 HTTP/HTTPS 支持所需的主要组件。

- 在客户端， HTTP 或 HTTPS 传输驱动程序将 Message Queue 消息封装到 HTTP 请求中，并确保将这些请求以正确的顺序发送给 Web 服务器 / 应用服务器。
- 必要时，客户端可以使用 HTTP 代理服务器与代理进行通信。可以在启动客户端时使用命令行选项指定代理的地址。有关更多信息，请参见第 329 页上的“使用 HTTP 代理”。
- HTTP 或 HTTPS 隧道 Servlet（均与 Message Queue 捆绑在一起）被装入 Web 服务器 / 应用服务器中，用于在将有效负荷消息转发给代理之前，从客户端 HTTP 请求中提取这些消息。 HTTP/HTTPS 隧道 Servlet 还将代理消息发送回客户端，以响应客户端发出的 HTTP 请求。可以使用一个 HTTP/HTTPS 隧道 Servlet 访问多个代理。

图 C-1 HTTP/HTTPS 支持体系结构



- 在代理端， httpjms 或 httpsjms 连接服务展开并分离来自相应隧道 Servlet 的传入消息。

- 如果 Web 服务器 / 应用服务器出现故障并重新启动，则所有连接都将恢复且不会对客户端产生影响。如果代理出现故障并重新启动，则会抛出异常，客户端必须重新建立连接。如果 Web 服务器 / 应用服务器和代理均出现故障 （这种情况不太可能发生），并且代理没有重新启动，则 Web 服务器 / 应用服务器将恢复客户端连接并等待代理恢复连接，而不会通知客户端。为了避免出现这种情况，应始终重新启动代理。

从图 C-1 可以看出，HTTP 和 HTTPS 支持的体系结构非常相似。主要区别在于，对于 HTTPS （httpsjms 连接服务）而言，隧道 Servlet 与客户端应用程序和代理之间的连接都是安全的。

与代理之间的安全连接是通过支持 SSL 的隧道 Servlet （Message Queue 的 HTTPS 隧道 Servlet）提供的，该 Servlet 向请求连接的任何代理发送自签名证书。代理使用该证书建立与 HTTPS 隧道 Servlet 的加密连接。建立此连接之后，客户端应用程序和 Web 服务器 / 应用服务器可以协商建立客户端应用程序与隧道 Servlet 之间的安全连接。

启用 HTTP 支持

以下各节介绍了启用 HTTP 支持所需的步骤。

➤ **启用 HTTP 支持**

1. 部署 HTTP 隧道 Servlet。可以在以下对象上部署 HTTP 隧道 Servlet：
   - Sun Java System Web Server
   - Sun Java System Application Server
2. 配置代理的 httpjms 连接服务并启动代理。
3. 配置 HTTP 连接。

步骤 1：部署 HTTP 隧道 Servlet

在 Sun Java System Web Server 或 Sun Java System Application Server 上，可以将 HTTP 隧道 Servlet 作为 Web 归档 (`.war`) 文件进行部署。

要将 HTTP 隧道 Servlet 作为 `.war` 文件进行部署，需要使用 Web 服务器 / 应用服务器提供的部署机制。HTTP 隧道 Servlet `.war` 文件 (`imqhttp.war`) 位于包含 `.jar`、`.war` 和 `.rar` 文件的目录中，具体目录因操作系统而异 （请参见附录 A “Message Queue 数据在特定平台上的位置”）。

.war 文件包含一个部署描述符，该描述符包含 Web 服务器 / 应用服务器装入和运行 Servlet 时所需的基本配置信息。根据 Web 服务器 / 应用服务器的不同，您可能还需要指定该 Servlet URL 的上下文根部分。

作为 Web 归档文件部署

有关在 Sun Java System Web Server 上部署的信息，请参见第 324 页上的 “在 Sun Java System Web Server 上部署 HTTP 隧道 Servlet”。

有关在 Sun Java System Application Server 上部署的信息，请参见第 325 页上的 “在 Sun Java System Application Server 上部署 HTTP 隧道 Servlet”。

在 Sun Java System Web Server 上部署 HTTP 隧道 Servlet

下面的说明与 Sun Java System Web Server 上的部署有关。通过使用 Web 浏览器访问 Servlet URL，可以验证 HTTP 隧道 Servlet 的部署是否成功。它应该显示状态信息。

➤ **将 HTTP 隧道 Servlet 作为 .war 文件部署**

1. 在基于浏览器的管理 GUI 中，选择 “虚拟 Server 类”选项卡并选择 “管理类”。
2. 选择相应的虚拟服务器类名 （例如， `defaultClass`）并单击 “管理”按钮。
3. 选择 “管理虚拟服务器”。
4. 选择相应的虚拟服务器名称并单击 “管理”按钮。
5. 选择 “Web 应用程序”选项卡。
6. 单击 “部署 Web 应用程序”。
7. 为 “WAR 文件打开”和 “WAR 文件路径”字段选择相应的值，以指向 `imqhttp.war` 文件，该文件所在的目录因操作系统而异 （请参见附录 A “Message Queue 数据在特定平台上的位置”）。
8. 在 “应用程序 URI”字段中输入路径。

   “应用程序 URI”字段值为隧道 Servlet URL 的 */contextRoot* 部分：

   `http://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

   例如，如果将 *contextRoot* 设置为 `imq`，则 “应用程序 URI”字段为：

   `/imq`
9. 输入要在其中部署 Servlet 的安装目录路径 （通常位于 Sun Java System Web Server 的安装根目录下）。

10. 单击 “确定”。
11. 重新启动 Web 服务器实例。

Servlet 现在即被部署到以下位置：

`http://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

客户端现在即可使用此 URL 通过 HTTP 连接来连接到消息服务。

禁用服务器访问日志

您不必禁用服务器访问日志，但禁用服务器访问日志可以获取更佳性能。

➤ **禁用服务器访问日志**

1. 选择 “状态”选项卡。
2. 选择 “日志参考页”。

请使用日志客户端访问控制来禁用日志记录。

在 Sun Java System Application Server 上部署 HTTP 隧道 Servlet

本节说明了如何在 Sun Java System Application Server 上将 HTTP 隧道 Servlet 作为 `.war` 文件进行部署，然后对该隧道 Servlet 进行配置，使其接受来自 Message Queue 代理的连接。

需要执行两个步骤：

- 使用应用服务器部署工具部署 HTTP 隧道 Servlet。
- 修改应用服务器实例的 `server.policy` 文件。

使用部署工具

➤ **在应用服务器环境中部署 HTTP 隧道 Servlet**

1. 在基于 Web 的管理 GUI 中，选择

   “应用服务器” > “实例” > "server1" > “应用程序” > “Web 应用程序”。
2. 单击 “部署”按钮。
3. 在 “文件路径：”文本字段中，输入 HTTP 隧道 Servlet `.war` 文件 (`imqhttp.war`) 的位置，然后单击 “确定”。

   `imqhttp.war` 文件的位置因操作系统而异 （请参见附录 A “Message Queue 数据在特定平台上的位置”）。

4. 设置 “上下文根”文本字段的值，然后单击 “确定”。

   “上下文根”字段的值是隧道 Servlet URL 的 /*contextRoot* 部分：

   http://*hostName*:*portNumber*/*contextRoot*/tunnel

   例如，可以将 “上下文根”字段设置为 /imq。

   显示的确认屏幕确认隧道 Servlet 已成功部署，在默认情况下处于启用状态，并且在本例中该 Servlet 位于以下位置：

   ```
   /var/opt/SUNWappserver8/domains/domain1/server1/applications/
   j2ee-modules/imqhttp_1
   ```

Servlet 现在即被部署到以下 URL：

http://*hostName*:*portNumber*/*contextRoot*/tunnel

客户端现在可以使用此 URL 通过 HTTP 连接来连接到消息服务。

修改 server.policy 文件

应用服务器实施了一组默认安全策略，除非经过修改，否则它们将阻止 HTTP 隧道 Servlet 接受来自 Message Queue 代理的连接。

每个应用服务器实例都有一个包含安全策略或规则的文件。例如， server1 实例的此文件在 Solaris 上的位置为：

```
/var/opt/SUNWappserver8/domains/domain1/server1/config/
server.policy
```

要配置隧道 Servlet 以接受来自 Message Queue 代理的连接，此文件中还必须包含另一个条目。

➤ 修改应用服务器的 server.policy 文件

1. 打开 server.policy 文件。
2. 添加以下条目：

```
grant codeBase
"file:/var/opt/SUNWappserver8/domains/domain1/server1/
            applications/j2ee-modules/imqhttp_1/-î
{
    permission java.net.SocketPermission "*",
            "connect,accept,resolve";
};
```

步骤 2：配置 httpjms 连接服务

默认情况下不为代理激活 HTTP 支持，因此您需要重新配置代理才能激活 httpjms 连接服务。重新配置后，可以按照第 64 页上的 “启动代理”中介绍的步骤启动代理。

➤ 激活 httpjms 连接服务

1. 打开代理的实例配置文件。

   实例配置文件存储在一个目录中，该目录使用与此配置文件关联的代理实例的名称 (*instanceName*) 进行标识（请参见附录 A “Message Queue 数据在特定平台上的位置”）：

   …/instances/*instanceName*/props/config.properties

2. 将 `httpjms` 值添加到 `imq.service.activelist` 属性中：

   ```
   imq.service.activelist=jms,admin,httpjms
   ```

启动时，代理将查找 Web 服务器 / 应用服务器以及在其主机上运行的 HTTP 隧道 Servlet。但是，要访问远程隧道 Servlet，可以重新配置 `servletHost` 和 `servletPort` 连接服务属性。

此外，还可以重新配置 `pullPeriod` 属性来改善性能。表 C-1 详细介绍了 httpjms 连接服务配置属性。

表 C-1 `httpjms` 连接服务属性

属性	描述
`imq.httpjms.http.servletHost`	必要时可以更改此值，以指定运行 HTTP 隧道 Servlet 的主机的名称（主机名或 IP 地址）。（可以是远程主机或本地主机上的特定主机名。）默认值：`localhost`。
`imq.httpjms.http.servletPort`	更改此值可以指定代理用于访问 HTTP 隧道 Servlet 的端口号。（如果在 Web 服务器上更改了默认端口号，则必须对此属性做相应的更改。）默认值：`7675`。
`imq.httpjms.http.pullPeriod`	指定客户端运行时向代理发出提取消息的 HTTP 请求的时间间隔（以秒为单位）。（请注意，此属性在代理上设置并传播到客户端运行时。）如果值为零或负数，客户端将始终使一个 HTTP 请求处于待处理状态，这样可以随时尽快地提取消息。如果客户端数量过多，这样做会消耗大量的 Web/ 应用服务器资源，从而导致服务器停止响应。在这种情况下，应将 `pullPeriod` 属性设置为正的秒数值。此属性设置客户端 HTTP 传输驱动程序在发出下一个提取请求之前等待的时间。将该值设置为正数可以节省 Web/ 应用服务器资源，但却延长了客户端等待响应的时间。默认值：`-1`。

表 C-1 httpjms 连接服务属性 （续）

属性	描述
imq.httpjms.http.connectionTimeout	指定客户端运行时等待 HTTP 隧道 Servlet 响应的时间 （以秒为单位），超过此时间后将抛出异常。（请注意，此属性在代理上设置并传播到客户端运行时。）此属性还指定代理与 HTTP 隧道 Servlet 通信后等待连接断开的时间。在这种情况下设置超时时间是必要的，因为代理和隧道 Servlet 无法知道访问 HTTP Servlet 的客户端是否已异常终止。默认值：60。

步骤 3：配置 HTTP 连接

客户端应用程序必须使用正确配置的连接工厂受管理对象来建立与代理的 HTTP 连接。本节介绍了 HTTP 连接配置问题。

配置连接工厂

要启用 HTTP 支持，必须将连接工厂的 imqAddressList 属性设置为 HTTP 隧道 Servlet URL。 HTTP 隧道 Servlet URL 的一般语法如下：

```
http://hostName:portNumber/contextRoot/tunnel
```

其中 *hostName*:*portNumber* 是作为 HTTP 隧道 Servlet 宿主的 Web 服务器 / 应用服务器的名称和端口，而 *contextRoot* 是在该 Web 服务器 / 应用服务器上部署隧道 Servlet 时设置的路径。

有关连接工厂属性，特别是有关 imqAddressList 属性的更多信息，请参见 Message Queue Developer's Guide for Java Clients。

可以使用以下方法之一设置连接工厂属性：

- 在创建连接工厂受管理对象的 imqobjmgr 命令中使用 -o 选项（请参见第 158 页上的 “添加连接工厂”），或者在使用管理控制台 (imqadmin) 创建连接工厂受管理对象时设置属性。
- 在启动客户端的命令中使用 -D 选项 （请参见 Message Queue Developer's Guide for Java Clients）。
- 以编程方式在客户端代码中创建连接工厂之后，使用 API 调用来设置其属性（请参见 Message Queue Developer's Guide for Java Clients）。

使用一个 Servlet 访问多个代理

如果正在运行多个代理，您不必配置多个 Web 服务器 / 应用服务器和多个 Servlet 实例。可以在并行运行的多个代理之间共享一个 Web 服务器 / 应用服务器和一个 HTTP 隧道 Servlet 实例。如果多个代理实例共享一个隧道 Servlet，则必须如下所示配置 imqAddressList 连接工厂属性：

http://*hostName*:*portNumber*/*contextRoot*/tunnel?ServerName=*bkrHostName*:*instanceName*

其中 *bkrHostName* 是代理实例主机名， *instanceName* 是您希望客户端访问的特定代理实例的名称。

要查看是否为 *bkrHostName* 和 *bkrHostName* 输入了正确的字符串，可以通过从浏览器访问 Servlet URL 来生成 HTTP 隧道 Servlet 的状态报告。状态报告将列出 Servlet 正在访问的所有代理：

```
HTTP 隧道 Servlet 就绪。
Servlet 开始时间：Thu May 30 01:08:18 PDT 2005
接受端口上来自代理的 TCP 连接： 7675
可用代理总数 = 2
代理列表：
   jpgserv:broker2
   cochin:broker1
```

使用 HTTP 代理

如果使用 HTTP 代理访问 HTTP 隧道 Servlet：

- 将 http.proxyHost 系统属性设置为代理服务器主机名。
- 将 http.proxyPort 系统属性设置为代理服务器端口号。

可以通过在启动客户端应用程序的命令中使用 -D 选项来设置这些属性。

启用 HTTPS 支持

以下各节说明了启用 HTTPS 支持的步骤。这些步骤与第 323 页上的 “启用 HTTP 支持”中的步骤类似，只是增加了生成和访问 SSL 证书的步骤。

➤ 启用 HTTPS 支持

1. 为 HTTPS 隧道 Servlet 生成自签名证书。

2. 修改 HTTP 隧道 Servlet .war 文件的部署描述符，以便：
   - 指向放置证书密钥库的位置
   - 指定证书密钥库密码
3. 部署 HTTP 隧道 Servlet。可以在以下服务器上部署 HTTP 隧道 Servlet：
   - Sun Java System Web Server
   - Sun Java System Application Server
4. 配置代理的 httpsjms 连接服务并启动代理。
5. 配置 HTTPS 连接。

以下各节详细介绍了其中每个步骤。

步骤 1：为 HTTPS 隧道 Servlet 生成自签名证书

Message Queue 的 SSL 支持用于保护所传输数据的安全性，它假定客户端与之通信的服务器已知且可信任。因此，仅使用自签名的服务器证书来实现 SSL。在 httpsjms 连接服务体系结构中， HTTPS 隧道 Servlet 充当代理和应用程序客户端的服务器。

运行 keytool 实用程序，以便为隧道 Servlet 生成自签名证书。在命令提示符下输入以下内容：

```
JRE_HOME/bin/keytool -servlet keyStoreLocation
```

命令行实用程序会提示您提供所需的信息。（在 Unix 系统上，您可能需要以超级用户 (root) 身份运行 keytool，才能获取创建密钥库的权限。）

首先， keytool 提示您输入密钥库密码，并提示您输入一些组织信息，然后提示您进行确认。收到确认后，它将在生成密钥对时暂停。然后它要求您输入密码以锁定特定的密钥对 （密钥密码），此时应按回车键来响应此提示：这样做会使密钥密码与密钥库密码相同。

注 请记住输入的密码，稍后您必须将此密码提供给隧道 Servlet，以便它可以打开密钥库。

JDK keytool 实用程序生成一个自签名证书，并将其放入 Message Queue 的密钥库文件中，该文件的位置在 *keyStoreLocation* 参数中指定。

注 HTTPS 隧道 Servlet 必须能够访问该密钥库。请确保将生成的密钥库（位于 *keyStoreLocation*）移动 / 复制到 HTTPS 隧道 Servlet 可以访问的位置 （请参见第 332 页上的 “步骤 3：部署 HTTPS 隧道 Servlet”）。

步骤 2：修改 HTTP 隧道 Servlet .war 文件的描述符文件

HTTP 隧道 Servlet 的 .war 文件包含一个部署描述符，该描述符包含 Web 服务器 / 应用服务器装入和运行该 Servlet 时所需的基本配置信息。

imqhttps.war 文件的部署描述符无法知道该隧道 Servlet 所需的密钥库文件的放置位置。因此在部署 imqhttps.war 文件之前，必须编辑隧道 Servlet 的部署描述符（一个 XML 文件），以指定密钥库的位置和密码。

➤ **修改 HTTPS 隧道 Servlet .war 文件**

1. 将 .war 文件复制到临时目录。

   cp /usr/share/lib/imq/imqhttps.war /tmp (Solaris)

   cp /opt/sun/mq/share/lib/imqhttps.war /tmp (Linux)

   cp IMQ_HOME/lib/imqhttps.war /tmp (Windows)

2. 使临时目录成为当前目录。

   ```
   $ cd /tmp
   ```

3. 提取 .war 文件的内容。

   ```
   $ jar xvf imqhttps.war
   ```

4. 列出 .war 文件的部署描述符。

   ```
   $ ls -l WEB-INF/web.xml
   ```

5. 编辑 web.xml 文件，为 keystoreLocation 和 keystorePassword 参数 （如有必要，还包括 servletPort 和 servletHost 参数）提供正确的值。

6. 重新装入 .war 文件的内容。

   $ jar uvf imqhttps.war WEB-INF/web.xml

现在即可使用修改后的 imqhttps.war 文件来部署 HTTPS 隧道 Servlet。（如果您担心泄漏密钥库密码，可以使用文件系统权限限制对 imqhttps.war 文件的访问。）

步骤 3：部署 HTTPS 隧道 Servlet

在 Sun Java System Web Server 或 Sun Java System Application Server 上，可以将 HTTP 隧道 Servlet 作为 Web 归档 (WAR) 文件进行部署。

将 HTTPS 隧道 Servlet 作为 .war 文件部署时，需要使用 Web 服务器 / 应用服务器提供的部署机制。HTTPS 隧道 Servlet .war 文件 (imqhttps.war) 所在的目录因操作系统而异 （请参见附录 A “Message Queue 数据在特定平台上的位置”）。

您应该确保激活 Web 服务器的加密功能，以使客户端与代理之间的端到端通信是安全的。

作为 Web 归档文件部署

有关在 Sun Java System Web Server 上部署的信息，请参见第 332 页上的 “在 Sun Java System Web Server 上部署 HTTPS 隧道 Servlet”。

有关在 Sun Java System Application Server 上部署的信息，请参见第 333 页上的 “在 Sun Java System Application Server 上部署 HTTPS 隧道 Servlet”。

在 Sun Java System Web Server 上部署 HTTPS 隧道 Servlet

本节说明了如何在 Sun Java System Web Server 上将 HTTPS 隧道 Servlet 作为 .war 文件部署。通过使用 Web 浏览器访问 Servlet URL，可以验证 HTTPS 隧道 Servlet 的部署是否成功。它应该显示状态信息。

在部署 HTTPS 隧道 Servlet 之前，请确保 JSSE .jar 文件包含在 Web 服务器的类路径中。要达到此目的，最简单的方法是将 jsse.jar、 jnet.jar 和 jcert.jar 文件复制到 WebServer_TOPDIR/bin/https/jre/lib/ext。

➤ **将 HTTPS 隧道 Servlet 作为 .war 文件部署**

1. 在基于浏览器的管理 GUI 中，选择 “虚拟服务器类”选项卡。单击 “管理类”。
2. 选择相应的虚拟服务器类名 （例如， defaultClass）并单击 “管理”按钮。
3. 选择 “管理虚拟服务器”。
4. 选择相应的虚拟服务器名称并单击 “管理”按钮。
5. 选择 “Web 应用程序”选项卡。

6. 单击 “部署 Web 应用程序”。
7. 为 “WAR 文件打开”和 “WAR 文件路径”字段选择相应的值，以指向修改后的 `imqhttps.war` 文件 （请参见第 331 页上的 “修改 HTTPS 隧道 Servlet .war 文件”）。
8. 在 “应用程序 URI”字段中输入路径。

   “应用程序 URI”字段值为隧道 Servlet URL 的 */contextRoot* 部分：

   `https://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

   例如，如果将 *contextRoot* 设置为 `imq`，则 “应用程序 URI”字段为：

   `/imq`
9. 输入要在其中部署 Servlet 的安装目录路径 （通常位于 Sun Java System Web Server 的安装根目录下）。
10. 单击 “确定”。
11. 重新启动 Web 服务器实例。

Servlet 现在即被部署到以下 URL：

`https://`*hostName*`:`*portNumber*`/imq/tunnel`

客户端现在即可使用此 URL 通过安全的 HTTPS 连接来连接到消息服务。

禁用服务器访问日志

您不必禁用服务器访问日志，但禁用服务器访问日志可以获取更佳性能。

➤ **禁用服务器访问日志**

1. 选择 “状态”选项卡。
2. 选择 “日志参考页”。

请使用日志客户端访问控制来禁用日志记录。

在 Sun Java System Application Server 上部署 HTTPS 隧道 Servlet

本节说明了如何在 Sun Java System Application Server 上将 HTTPS 隧道 Servlet 作为 `.war` 文件部署。

需要执行两个步骤：

- 使用应用服务器部署工具部署 HTTPS 隧道 Servlet。
- 修改应用服务器实例的 `server.policy` 文件。

使用部署工具

➤ **在应用服务器环境中部署 HTTPS 隧道 Servlet**

1. 在基于 Web 的管理 GUI 中，选择

   “应用服务器” > “实例” > "server1" > “应用程序” > “Web 应用程序”。

2. 单击 “部署”按钮。

3. 在 “文件路径：”文本字段中，输入 HTTPS 隧道 Servlet `.war` 文件 (`imqhttps.war`) 的位置，然后单击 “确定”。

   `imqhttps.war` 文件的位置因操作系统而异 （请参见附录 A “Message Queue 数据在特定平台上的位置”）。

4. 设置 “上下文根”文本字段的值，然后单击 “确定”。

   “上下文根”字段值为隧道 Servlet URL 的 */contextRoot* 部分：

   `https://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

   例如，可以将 “上下文根”字段设置为：

   ```
   /imq
   ```

   下一个屏幕显示隧道 Servlet 已成功部署，在默认情况下处于启用状态，并且在本例中该 Servlet 位于以下位置：

   ```
   /var/opt/SUNWappserver8/domains/domain1/server1/applications/
   j2ee-modules/imqhttps_1
   ```

Servlet 现在即被部署到以下 URL：

`https://`*hostName*`:`*portNumber*`/`*contextRoot*`/tunnel`

客户端现在即可使用此 URL 通过 HTTPS 连接来连接到消息服务。

修改 server.policy 文件

应用服务器实施了一组默认安全策略，除非经过修改，否则它们将阻止 HTTPS 隧道 Servlet 接受来自 Message Queue 代理的连接。

每个应用服务器实例都有一个包含安全策略或规则的文件。例如， server1 实例的此文件在 Solaris 上的位置为：

```
/var/opt/SUNWappserver8/domains/domain1/server1/config/
server.policy
```

要使隧道 Servlet 接受来自 Message Queue 代理的连接，此文件中还必须包含另一个条目。

➤ **修改应用服务器的 server.policy 文件**

1. 打开 server.policy 文件。
2. 添加以下条目：

```
grant codeBase
"file:/var/opt/SUNWappserver8/domains/domain1/server1/
            applications/j2ee-modules/imqhttps_1/-"
{
    permission java.net.SocketPermission "*",
            "connect,accept,resolve";
};
```

步骤 4：配置 httpsjms 连接服务

默认情况下不为代理激活 HTTPS 支持，因此您需要重新配置代理才能激活 httpsjms 连接服务。重新配置后，可以按照第 64 页上的 “启动代理”中介绍的步骤启动代理。

➤ **激活 httpsjms 连接服务**

1. 打开代理的实例配置文件。

   实例配置文件存储在一个目录中，该目录使用与此配置文件关联的代理实例的名称 (*instanceName*) 进行标识（请参见附录 A “Message Queue 数据在特定平台上的位置”）：

   …/instances/*instanceName*/props/config.properties

2. 将 `httpsjms` 值添加到 `imq.service.activelist` 属性中：

   ```
   imq.service.activelist=jms,admin,httpsjms
   ```

启动时，代理将查找 Web 服务器以及在其主机上运行的 HTTP 隧道 Servlet。但是，要访问远程隧道 Servlet，可以重新配置 `servletHost` 和 `servletPort` 连接服务属性。

此外，还可以重新配置 `pullPeriod` 属性来改善性能。表 C-2 详细介绍了 httpsjms 连接服务配置属性。

表 C-2 httpsjms 连接服务属性

属性	描述
imq.httpsjms.https.servletHost	必要时可以更改此值，以指定运行 HTTPS 隧道 Servlet 的主机的名称 （主机名或 IP 地址）。（可以是远程主机或本地主机上的特定主机名。）默认值：localhost。
imq.httpsjms.https.servletPort	更改此值可以指定代理用于访问 HTTPS 隧道 Servlet 的端口号。（如果在 Web 服务器上更改了默认端口号，则必须对此属性做相应的更改。）默认值：7674。
imq.httpsjms.https.pullPeriod	指定每个客户端发出从代理提取消息的 HTTP 请求的时间间隔（以秒为单位）。（请注意，此属性在代理上设置并传播到客户端运行时。）如果值为零或负数，客户端将始终使一个 HTTP 请求处于待处理状态，这样可以随时尽快地提取消息。如果客户端数量过多，这样做会消耗大量的 Web 服务器资源，从而导致服务器停止响应。在这种情况下，应该将 pullPeriod 属性设置为正的秒数值。此属性设置客户端 HTTP 传输驱动程序在发出下一个提取请求之前等待的时间。将该值设置为正数可以节省 Web 服务器资源，但却延长了客户端等待响应的时间。默认值：-1。
imq.httpsjms.https.connectionTimeout	指定客户端运行时等待 HTTPS 隧道 Servlet 响应的时间 （以秒为单位），超过此时间后将抛出异常。（请注意，此属性在代理上设置并传播到客户端运行时。）此属性还指定代理与 HTTPS 隧道 Servlet 通信后等待连接断开的时间。在这种情况下设置超时时间是必要的，因为代理和隧道 Servlet 无法知道访问 HTTPS Servlet 的客户端是否已异常终止。默认值：60。

步骤 5：配置 HTTPS 连接

客户端应用程序必须使用正确配置的连接工厂受管理对象才能建立与代理的 HTTPS 连接。

但是，客户端还必须能够访问 Java 安全套接扩展 (Java Secure Socket Extension, JSSE) 提供的 SSL 库，并且还必须有一个根证书。 SSL 库是随 JDK 1.4 一起提供的。如果您使用的是早期版本的 JDK，请参见 “配置 JSSE”；否则，请参见 “导入根证书”。

解决上述问题后，即可开始配置 HTTPS 连接。

配置 JSSE

➤ 配置 JSSE

1. 将 JSSE .jar 文件复制到 JRE_HOME/lib/ext 目录。

jsse.jar、jnet.jar 和 jcert.jar

2. 静态添加 JSSE 安全服务提供者，方法是将

   security.provider.*n*=com.sun.net.ssl.internal.ssl.Provider

   添加到 JRE_HOME/lib/security/java.security 文件 （其中 *n* 是安全服务提供者软件包的下一个可用优先级编号）。

3. 如果您使用的不是 JDK1.4，则需要在启动客户端应用程序的命令中使用 -D 选项来设置下面的 JSSE 属性：

   java.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol

导入根证书

如果签署 Web 服务器证书的 CA 的根证书默认情况下不在信任数据库中，或者您使用的是专用的 Web 服务器 / 应用服务器证书，则必须将该证书添加到信任数据库中。如果存在上述情况，请按照下面的说明执行操作；否则，请转到 “配置连接工厂”。

假定证书保存在 *certFile* 中，且 *trustStoreFile* 是您的密钥库，请运行以下命令：

```
JRE_HOME/bin/keytool -import -trustcacerts
-alias aliasForCertificate -file certFile
-keystore trustStoreFile
```

对显示的如下问题回答 YES：Trust this certificate?

此外，还需要在启动客户端应用程序的命令中使用 -D 选项来指定以下 JSSE 属性：

javax.net.ssl.trustStore=*trustStoreFile*

javax.net.ssl.trustStorePassword=*trustStorePasswd*

配置连接工厂

要启用 HTTPS 支持，必须将连接工厂的 imqAddressList 属性设置为 HTTPS 隧道 Servlet URL。 HTTPS 隧道 Servlet URL 的一般语法如下：

https://*hostName*:*portNumber*/*contextRoot*/tunnel

其中 *hostName*:*portNumber* 是作为 HTTPS 隧道 Servlet 宿主的 Web 服务器的名称和端口，而 *contextRoot* 是在该 Web 服务器上部署隧道 Servlet 时设置的路径。

有关连接工厂属性，特别是有关 imqAddressList 属性的更多信息，请参见 Message Queue Developer's Guide for Java Clients。

可以使用以下方法之一设置连接工厂属性：

- 在创建连接工厂受管理对象的 imqobjmgr 命令中使用 -o 选项（请参见第 158 页上的 “添加连接工厂”），或者在使用管理控制台 (imqadmin) 创建连接工厂受管理对象时设置属性。
- 在启动客户端应用程序的命令中使用 -D 选项 （请参见 Message Queue Developer’s Guide for Java Clients）。
- 以编程方式在客户端应用程序代码中创建连接工厂之后，使用 API 调用来设置其属性 （请参见 Message Queue Developer’s Guide for Java Clients）。

使用一个 Servlet 访问多个代理

如果正在运行多个代理，您不必配置多个 Web 服务器和多个 Servlet 实例。可以在并行运行的多个代理之间共享一个 Web 服务器和一个 HTTPS 隧道 Servlet 实例。如果多个代理实例共享一个隧道 Servlet，则必须如下所示配置 imqAddressList 连接工厂属性：

https://*hostName*:*portNumber*/*contextRoot*/tunnel?ServerName=*bkrHostName*:*instanceName*

其中 *bkrHostName* 是代理实例主机名， *instanceName* 是您希望客户端访问的特定代理实例的名称。

要查看是否为 *bkrhostName* 和 *instanceName* 输入了正确的字符串，可以通过从浏览器访问 Servlet URL 来生成 HTTPS 隧道 Servlet 的状态报告。状态报告将列出 Servlet 正在访问的所有代理：

```
HTTPS 隧道 servlet 就绪。
Servlet 开始时间：Thu May 30 01:08:18 PDT 2002
接受端口上来自代理的安全连接： 7674
可用代理总数 = 2
代理列表：
   jpgserv:broker2
   cochin:broker1
```

使用 HTTP 代理

如果使用 HTTP 代理访问 HTTPS 隧道 Servlet：

- 将 http.proxyHost 系统属性设置为代理服务器主机名。
- 将 http.proxyPort 系统属性设置为代理服务器端口号。

可以通过在启动客户端应用程序的命令中使用 -D 选项来设置这些属性。

疑难解答

本节介绍了 HTTP 或 HTTPS 连接可能出现的问题，并提供了有关如何解决这些问题的指导。

服务器或代理故障

如果 Web 服务器出现故障并重新启动，则所有连接都将恢复且不会对客户端产生影响。但是，如果代理出现故障并重新启动，则会抛出异常，客户端必须重新建立连接。

如果 Web 服务器和代理均出现故障，并且代理没有重新启动，则 Web 服务器将恢复客户端连接并等待代理恢复连接，而不会通知客户端。为了避免出现这种情况，应始终确保重新启动代理。

客户端无法通过隧道 Servlet 进行连接

如果 HTTPS 客户端无法通过隧道 Servlet 连接到代理，请执行以下操作：

1. 启动 Servlet 和代理。
2. 使用浏览器通过 HTTPS 隧道 Servlet URL 手动访问 Servlet。
3. 使用以下管理命令暂停和恢复连接：

```
imqcmd pause svc -n httpsjms -u admin
imqcmd resume svc -n httpsjms -u admin
```

服务恢复后， HTTPS 客户端应该能够通过隧道 Servlet 连接到代理。

常用命令实用程序命令

本附录列出了一些常用的 Message Queue 命令实用程序 (imqcmd) 命令。有关可以从命令行使用的命令选项和属性的完整列表，请参阅第 13 章 “命令行参考”中第 246 页上的 “命令实用程序”。

语法

```
imqcmd subcommand argument [options]
imqcmd -h|H
imqcmd -v
```

-H 或 -h 提供全面的帮助信息。 -v 子命令提供版本信息。

使用 imqcmd 时，命令实用程序会提示您输入密码。要避免出现此提示 （以提高安全性），可以使用 -passfile *pathToPassfile* 选项将该实用程序指向包含管理员用户名和密码的密码文件。

示例：imqcmd query bkr -u *adminUserName* -passfile *pathToPassfile* -b *myServer*:7676

代理和群集管理

```
imqcmd query bkr

imqcmd pause bkr

imqcmd restart bkr

imqcmd resume bkr

imqcmd shutdown bkr -b myBroker:7676

imqcmd update bkr -o "imq.system.max_count=1000"

imqcmd reload cls
```

代理配置属性 （-o 选项）

表 D-1 列出了常用的代理配置属性。有关代理配置属性及其描述的完整列表，请参见第 14 章 “代理属性参考”。

表 D-1 代理配置属性 （-o 选项）

属性	注释
imq.autocreate.queue	
imq.autocreate.queue.maxNumActiveConsumers	指定 -1 表示无限制
imq.autocreate.queue.maxNumBackupConsumers	指定 -1 表示无限制
imq.autocreate.topic	
imq.cluster.url	
imq.destination.DMQ.truncateBody	
imq.destination.logDeadMessages	
imq.log.file.rolloverbytes	指定 -1 表示无限制
imq.log.file.rolloversecs	指定 -1 表示无限制
imq.log.level	NONE ERROR WARNING INFO
imq.message.max_size	指定 -1 表示无限制
imq.portmapper.port	
imq.system.max_count	指定 -1 表示无限制

表 D-1 代理配置属性 （-o 选项）（续）

属性	注释
imq.system.max_size	指定 -1 表示无限制

服务和连接管理

```
imqcmd list svc

imqcmd query svc

imqcmd update svc -n jms -o "minThreads=200" -o "maxThreads=400" -o
"port=8995"

imqcmd pause svc -n jms

imqcmd resume svc -n jms

imqcmd list cxn -svn jms

imqcmd query cxn -n 1234567890
```

长期订户管理

```
imqcmd list dur -d MyTopic

imqcmd destroy dur -n myDurSub -c "clientID-111.222.333.444"

imqcmd purge dur -n myDurSub -c "clientID-111.222.333.444"
```

事务管理

```
imqcmd list txn

imqcmd commit txn -n 1234567890

imqcmd query txn -n 1234567890

imqcmd rollback txn -n 1234567890
```

目标管理

```
imqcmd create dst -n MyQueue -t q -o "maxNumMsgs=1000" -o
"maxNumProducers=5"

imqcmd update dst -n MyTopic -t t -o "limitBehavior=FLOW_CONTROL|
REMOVE_OLDEST|REJECT_NEWEST|REMOVE_LOW_PRIORITY"

imqcmd compact dst -n MyQueue -t q

imqcmd purge dst -n MyQueue -t q

imqcmd pause dst -n MyQueue -t q -pst PRODUCERS|CONSUMERS|ALL

imqcmd resume dst -n MyQueue -t q

imqcmd destroy dst -n MyQueue -t q

imqcmd query dst -n MyQueue -t q

imqcmd list dst -tmp
```

目标配置属性 （-o 选项）

表 D-2 列出常用的目标配置属性。有关目标配置属性及其描述的完整列表，请参见第 15 章 “物理目标属性参考”。

表 D-2 目标配置属性 （-o 选项）

属性	注释
consumerFlowLimit	指定 -1 表示无限制
isLocalOnly （仅创建）	
limitBehavior	FLOW_CONTROL REMOVE_OLDEST REJECT_NEWEST REMOVE_LOW_PRIORITY
localDeliveryPreferred （仅队列）	
maxNumActiveConsumers （仅队列）	指定 -1 表示无限制
maxNumBackupConsumers （仅队列）	指定 -1 表示无限制
maxBytesPerMsg	指定 -1 表示无限制
maxNumMsgs	指定 -1 表示无限制
maxNumProducers	指定 -1 表示无限制
maxTotalMsgBytes	指定 -1 表示无限制

表 D-2 目标配置属性 （-o 选项）（续）

属性	注释
useDMQ	

度量

```
imqcmd metrics bkr -m cxn|rts|ttl -int 5 -msp 20
imqcmd metrics svc -m cxn|rts|ttl
imqcmd metrics dst -m con|dsk|rts|ttl
```

词汇表

有关 Message Queue 术语的信息，请参见 Message Queue 技术概述中的词汇表。有关 Sun Java System 产品套件中使用的术语的完整列表，请参见 Java Enterprise System 词汇表 (http://docs.sun.com/doc/819-4631)。

索引

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

X

Y

Z

z