Sun Java™ System

Message Queue 3.6 SP3
技术概述

2005Q4

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

文件号码：819-3565

目录

图

表

前言

本书（即《Sun Java™ System Message Queue 3.6 SP3 2005Q4 技术概述》）对 Message Queue 消息传送服务的技术、概念、体系结构和功能进行了介绍。

因此， Message Queue 技术概述为 Message Queue 文档集中的其他书籍提供了基础。应先阅读本书，然后再阅读 Message Queue 文档集中的其他书籍。

本前言包含以下各节：

目标读者

本指南是专为管理员、应用程序开发者及其他计划使用 Message Queue 产品或有意了解该产品的技术、概念、体系结构和功能的人员编写的。

管理员负责设置和管理 Message Queue 消息传送服务。本书并不要求读者掌握消息传送系统的任何知识或对其有一定了解。

应用程序开发者负责编写使用 Message Queue 服务与其他客户端应用程序交换消息的 Message Queue 客户端应用程序。本书并未假定读者了解有关通过 Message Queue 服务实现的 Java 消息服务 (Java Message Service, JMS) 规范的任何知识。

阅读本书之前

阅读本书没有任何前提条件。应当先阅读本书，对 Message Queue 的基本概念有一定了解后，再阅读 Message Queue 开发者指南以及管理指南。

本书的结构

请从头到尾阅读本指南；因为每一章的内容都是以前面章节中所包含的信息为基础。下表简要介绍了各章的内容：

表 1 本书的内容和结构

章	描述
第 1 章 “消息传送系统：简介”	介绍消息传送中间件技术，讨论 JMS 标准并介绍该标准的 Message Queue 服务实现方案。
第 2 章 “客户端编程模型”	介绍 JMS 编程模型以及如何使用 Message Queue 客户端运行时创建 JMS 客户端。介绍 C++ 客户端和 SOAP 消息传输的运行时支持。
第 3 章“Message Queue 服务”	讨论管理任务和工具，并介绍用于配置连接、路由、持久性、安全性及监视的代理服务。
第 4 章 “代理群集”	讨论 Message Queue 代理群集的体系结构和使用。
第 5 章 “Message Queue 和 J2EE”	探讨有关在 J2EE 平台环境中实现 JMS 支持的其他信息。
附录 A “可选 JMS 功能的 Message Queue 实现”	介绍 Message Queue 产品如何处理 JMS 可选项。
附录 B “Message Queue 功能”	列出 Message Queue 功能，概述实现这些功能所需的步骤并提供有关进一步信息的参考。
词汇表	提供有关使用 Message Queue 时可能遇到的术语和概念的信息。

本手册中使用的约定

本节介绍本文档中使用的约定。

文本约定

表 2　文档约定

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。
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	全部大写文本表示文件系统类型（GIF、TXT、HTML 等）、环境变量 (IMQ_HOME) 或首字母缩略词 (JMS, JSP)。
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 使用三种目录变量；它们的设置因平台而异。表 3 介绍了这些变量并概述了如何在 Solaris™、 Windows 和 Linux 平台上使用这些变量。

表 3 Message Queue 目录变量

变量	描述
IMQ_HOME	Message Queue 文档中通常使用此变量来表示 Message Queue 基目录 （根安装目录）： • 在 Solaris 平台上，不存在 Message Queue 根安装目录。因此，Message Queue 文档中不使用 IMQ_HOME 来表示 Solaris 平台上的文件位置。 • 在 Solaris 平台上，对于 Sun Java System Application Server，Message Queue 根安装目录为 Application Server 基目录下的 /imq。 • 在 Windows 上， Message Queue 根安装目录由 Message Queue 安装程序设置 （默认情况下为 C:\Program Files\Sun\MessageQueue3）。 • 在 Windows 平台上，对于 Sun Java System Application Server，Message Queue 根安装目录为 Application Server 基目录下的 /imq。 • 在 Linux 平台上，不存在 Message Queue 根安装目录。因此，Message Queue 文档中不使用 IMQ_HOME 来表示 Linux 平台上的文件位置。
IMQ_VARHOME	这是 /var 目录，其中存储了 Message Queue 临时或动态创建的配置和数据文件。可设置为指向任何目录的环境变量。 • 在 Solaris 上， IMQ_VARHOME 默认为 /var/imq 目录。 • 在 Solaris 上，对于 Sun Java System Application Server 测试版， IMQ_VARHOME 默认为 IMQ_HOME/var 目录。 • 在 Windows 上， IMQ_VARHOME 默认为 IMQ_HOME\var 目录。 • 在 Windows 上，对于 Sun Java System Application Server，IMQ_VARHOME 默认为 IMQ_HOME\var 目录。 • 在 Linux 平台上， IMQ_VARHOME 默认为 /var/opt/imq 目录。

表 3 Message Queue 目录变量 （续）

变量	描述
IMQ_JAVAHOME	这是一个环境变量，指向 Message Queue 可执行文件所需的 Java™ 运行时 (JRE) 的位置： • 在 Solaris 上， IMQ_JAVAHOME 按以下顺序查找 Java 运行时，但用户可以选择性地将该值设置为所需 JRE 所在的位置。 Solaris 8 或 9： /usr/jdk/entsys-j2se /usr/jdk/jdk1.5.* /usr/jdk/j2sdk1.5.* /usr/j2se Solaris 10： /usr/jdk/entsys-j2se /usr/java /usr/j2se • 在 Linux 上，Message Queue 先按以下顺序查找 Java 运行时，但用户可以选择性地将 IMQ_JAVAHOME 值设置为所需 JRE 所在的位置。 /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_JAVAHOME 默认为 IMQ_HOME\jre，但用户可以选择性地将它的值设置为所需 JRE 所在的位置。

在本指南中，显示 IMQ_HOME、 IMQ_VARHOME 和 IMQ_JAVAHOME 时，**不使用**平台特定的环境变量表示法或语法 （例如，在 UNIX® 平台上为 $IMQ_HOME）。路径名通常采用 UNIX 目录分隔符表示法 (/)。

相关文档

除本指南外， Message Queue 还提供了其他文档资源。

Message Queue 文档集

表 4 按照通常的使用顺序列出了 Message Queue 文档集中的文档。

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、限制、已知错误以及技术说明的介绍。
Message Queue 技术概述	开发者、管理员、设计者	介绍 Message Queue 产品的技术、概念、体系结构及功能。
Message Queue 管理指南	管理员，也推荐开发者阅读	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	为使用 JMS 和 SOAP/JAXM 规范 Message Queue 实现方案开发 Java 客户端程序的人员提供快速入门教程和编程信息。
Message Queue Developer's Guide for C Clients	开发者	为使用 Message Queue 消息服务的 C 接口 (C-API) 开发 C 客户端程序的人员提供编程和参考文档。

联机帮助

Message Queue 包含用于执行 Message Queue 消息服务的管理任务的命令行实用程序。要访问这些实用程序的联机帮助，请参见 Message Queue 管理指南。

Message Queue 还包含图形用户界面 (Graphical User Interface, GUI) 管理工具，即管理控制台 (`imqadmin`)。管理控制台包含上下文有关联机帮助。

JavaDoc

以下位置提供了 JavaDoc 格式的 Message Queue Java 客户端 API（包括 JMS API）文档：

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

此文档可以在任何 HTML 浏览器中浏览，如 Netscape 或 Internet Explorer。它包括标准 JMS API 文档以及 Message Queue 受管理对象的 Message Queue 特定 API（请参见 Message Queue Developer's Guide for Java Clients 的第 3 章），这些对消息传送应用程序的开发者很有帮助。

示例客户端应用程序

许多示例应用程序提供了样例客户端应用程序代码，这些示例应用程序所在的目录取决于操作系统 （请参见 Message Queue 管理指南）。

请参见位于该目录及其每个子目录中的自述文件。

Java 消息服务 (Java Message Service, JMS) 规范

可以在以下位置找到 JMS 规范：

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

规范包括样例客户端代码。

相关的第三方 Web 站点引用

本文档引用第三方 URL，并提供其他相关信息。

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 819-2574，文件标题为 Sun Java System Message Queue 3.6 SP3 Technical Overview。

有关 Message Queue 的特定问题和疑问，另请参见
http://swforum.sun.com/jive/forum.jspa?forumID=24

第 1 章

消息传送系统：简介

Sun Java™ System Message Queue 是一种消息传送中间件产品，它实现并扩展 Java 消息服务 (Java Message Service, JMS) 标准。如果您已熟知这些概念，则请从第 30 页上的 “Message Queue：元素和功能”一节开始阅读。否则，请您从头开始阅读本章。

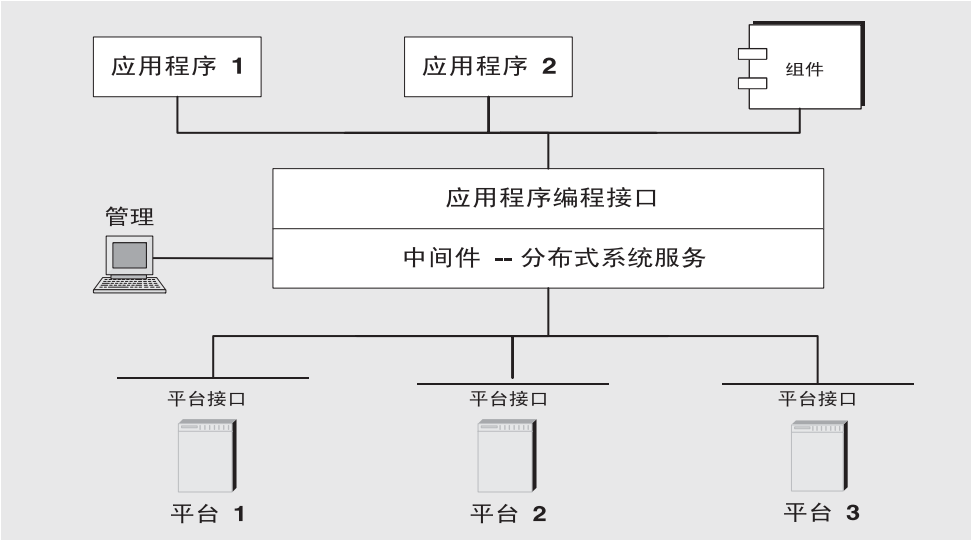
本章介绍 Message Queue 等产品底层的消息传送技术，并说明 Message Queue 如何实现和扩展对该技术进行标准化的 JMS 规范。本章涵盖以下主题：

面向消息的中间件 (Message-Oriented Middleware, MOM)

由于公司、机构和技术在不断变化，因此为它们提供服务的软件系统必须能够适应这些变化。在合并、添加服务或扩展可用服务之后，公司可能无力负担重新创建信息系统所需的成本。正是在这个关键时刻，才需要集成新组件或者尽可能高效地扩展现有组件。要集成异类组件，最方便的方法不是将它们重新创建为同类元素，而是提供一个允许它们进行通信 （不考虑它们之间的差异）的层。该层被称作**中间件**，它允许独立开发且运行于不同网络平台上的软件组件 （应用程序、 Enterprise Java Bean、Servlet 和其他组件）彼此交互。当能够进行这样的交互时，网络才成为计算机。

如图 1-1 所示，在概念上，中间件位于应用程序层与平台层（操作系统和底层网络服务）之间。

图 1-1 中间件



分布于不同网络节点上的应用程序使用应用程序接口进行通信，而不必关心托管其他应用程序的操作环境的细节，也不必关心将它们连接到这些应用程序的服务。此外，通过提供管理接口，可以使这个新的互连应用程序虚拟系统安全可靠。可以对性能进行度量和调整，也可以在不丢失任何功能的情况下进行扩展。

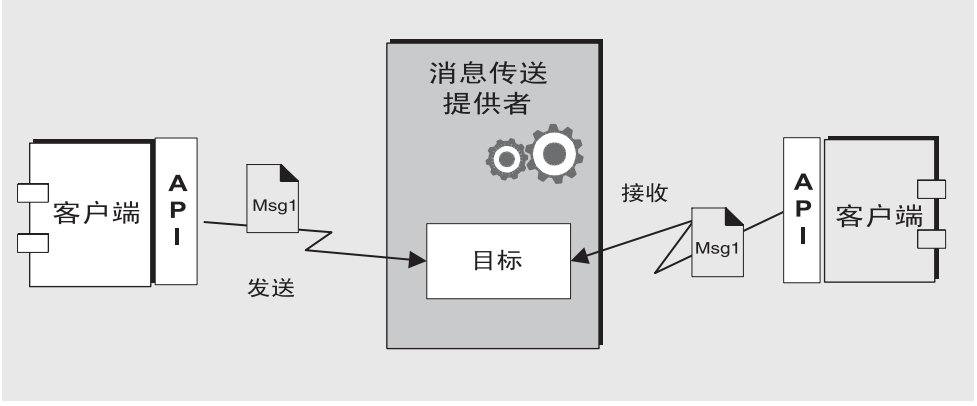
中间件可分为以下几类：

- 基于远程过程调用 (Remote Procedure Call, RPC) 的中间件，允许一个应用程序中的过程调用远程应用程序中的过程，就好像它们是本地调用一样。该中间件实现一个查找远程过程的链接机制并使调用方能够以透明方式使用这些过程。以前，这种类型的中间件处理基于过程的程序；现在，它还包括基于对象的组件。
- 基于对象请求代理 (Object Request Broker, ORB) 的中间件，使应用程序的对象能够在异类网络之间分布和共享。
- 基于面向消息的中间件 (Message Oriented Middleware, MOM) 的中间件使分布式应用程序可以通过发送和接收消息来进行通信和交换数据。

所有这些模型都使一个软件组件可以通过网络影响另一个组件的行为。它们的区别在于基于 RPC 和 ORB 的中间件会创建紧密耦合组件系统，而基于 MOM 的系统允许组件进行更松散的耦合。在基于 RPC 或 ORB 的系统中，当一个过程调用另一个过程时，它必须等待被调用的过程返回才能执行其他操作。正如前面所提到的，在这些模型中，中间件在一定程度上充当超级链接程序，在网络上查找被调用过程，并使用网络服务将函数或方法参数传递到被调用过程，然后返回查找结果。

基于 MOM 的系统允许通过异步交换消息来进行通信，如图 1-2 所示。

图 1-2 基于 MOM 的系统



面向消息的中间件使用消息传送提供者来协调消息传送操作。 MOM 系统的基本元素有客户端、消息和 MOM 提供者，后者包括 API 和管理工具。 MOM 提供者可以使用不同的体系结构来路由和传送消息：它可以使用集中式消息服务器，也可以将路由和传送功能分布在每台客户机上。有些 MOM 产品结合了这两种方法。

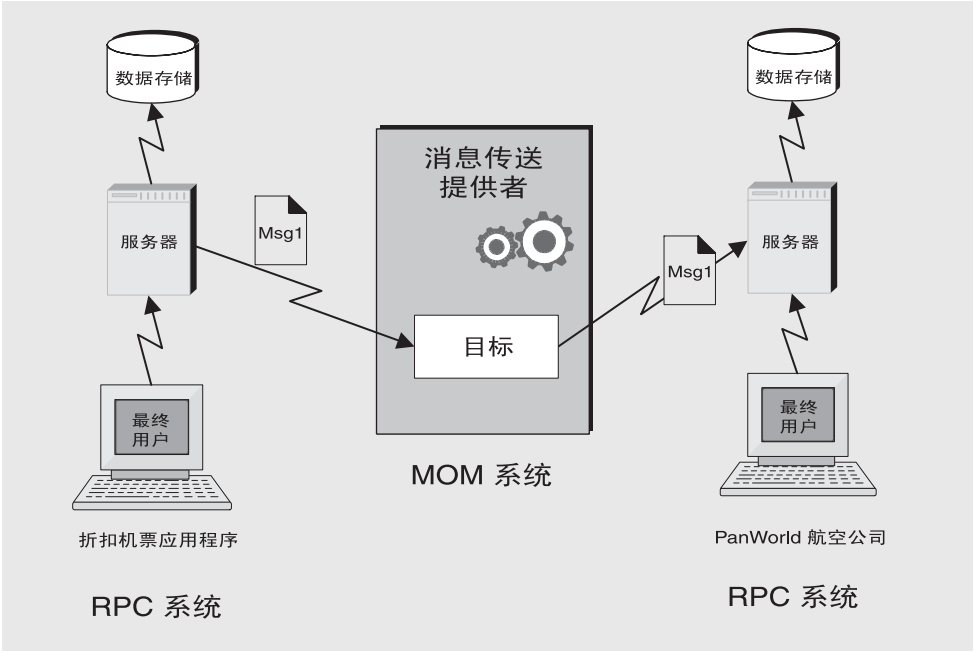
使用 MOM 系统，客户端可以进行 API 调用，以便将消息发送到由提供者管理的目标。该调用会调用提供者服务以路由和传送消息。在发送消息之后，客户端会继续执行其他工作，并确信在接收方客户端检索该消息之前，提供者一直保留该消息。基于消息的模型与提供者的协调耦合在一起，使得创建松散耦合的组件系统成为可能。这样的系统可以继续可靠地工作，即使在有个别组件或连接失败时也不会停机。

让消息传送提供者协调客户端之间的消息传送还有一个优点，那就是可以通过添加管理接口来监视和调整性能。这样，客户端应用程序便不必关心发送、接收和处理消息之外的任何问题。对于互操作性、可靠性、安全性、可扩展性和性能之类的问题，应当由管理员通过编码实现 MOM 系统来解决。

至此，我们已经介绍了使用面向消息的中间件连接分布式组件的优点，下面将介绍其缺点。缺点之一源自松散耦合本身。在 RPC 系统中，只有在被调用函数完成任务之后，才能返回调用函数。在异步系统中，调用方客户端会在收到消息时继续加载工作，直到处理加载工作所需的资源耗尽且被调用组件失败。当然，可以通过监视性能和调整消息流来尽量减少或避免这些情况，但对于 RPC 系统却不必这样做。有一点很重要，那就是了解每种系统的优缺点。每种系统所适合执行的任务都不同。有时，您需要结合两种系统才能完全获得所需的行为。

图 1-3 显示 MOM 系统如何使两个基于 RPC 的系统进行通信。该图的左侧显示在不同的网络节点上分布客户端、服务器和数据存储组件以提高性能的应用程序。这是一个折扣机票预定系统：最终用户为使用此服务支付一定的费用，使用该服务可以找到特定目的地和时间的最低费用。数据存储保存有关注册用户和参与此折扣计划的航空公司的信息。服务器上的逻辑功能根据用户的请求在所参与的航空公司中查询价格、对信息进行排序并向用户提供三个最低报价。该图的右侧显示一个基于 RPC 的系统，该系统表示所参与的任一航空公司的机票 / 预定系统。该图的右侧将为折扣系统所连接到的任意多个航空公司进行复制。对于每个这样的航空公司，数据存储都将保存有关可用航班的信息（座位、飞行时间和价格）。服务器组件将更新这些信息以响应最终用户输入的数据。航空公司的服务器还订阅 MOM 服务，接收折扣预定系统的信息请求，并返回座位和价格信息。如果用户决定购买 PanWorld 航空公司的折扣机票，则该系统的服务器组件将更新数据存储中的信息，然后为请求者生成机票或者向折扣服务发送一条消息以生成机票。

图 1-3 结合 RPC 和 MOM 系统



此示例说明 RPC 系统与 MOM 系统之间的一些区别。我们已经提到了它们在分布式组件耦合方式上的区别。另一个区别在于， RPC 系统通常用于分布和连接客户端和服务器组件 （在这些组件中，客户端通常是最终用户），而对于 MOM 系统，客户端通常是只能通过消息传送来互操作的异类软件组件。

MOM 是作为专用产品实现的，这为 MOM 系统带来了一个更为严重的问题。如果您的公司依赖于 SuperMOM-X，但最近收购了一家使用 SuperMOM-Y 的公司，会出现什么情况？要解决此问题，需要一个标准的消息传送接口。如果 SuperMOM-X 和 SuperMOM-Y 均实现了此接口，则针对一个系统开发的应用程序也可以运行在另一个系统上。这样的接口应该易于学习，同时提供足够的功能来支持复杂的消息传送应用程序。1998 年推出的 Java 消息服务 (Java Message Service, JMS) 规范就是为了实现这样的目的。下一节将介绍 JMS 的基本功能，并说明包含现有专用 MOM 产品的通用元素的标准是如何制订的。这些标准既允许差异存在，又使得进一步发展成为可能。

作为 MOM 标准的 JMS

Java 消息传送服务规范最初是为了允许 Java 应用程序访问现有的 MOM 系统而开发的。引入之后，它已被许多现有的 MOM 供应商采用并且已经凭借自身的功能实现为异步消息传送系统。

在创建 JMS 规范时，设计者希望捕获现有消息传送系统的精髓。这包括：

- 路由和传送消息的消息传送提供者的概念。
- 不同的消息传送模式或域 （如点对点消息传送和发布 / 订阅消息传送）。
- 用于接收同步和异步消息的工具。
- 对可靠消息传送的支持。
- 常见的消息格式 （如流、文本和字节）。

供应商通过提供一个 **JMS 提供者**来实现 JMS 规范，该提供者由实现 JMS 接口的库、消息的路由和传送功能以及用来管理、监视和调整消息传送服务的管理工具组成。路由和传送功能可以由集中式消息服务器或代理来执行，也可以通过每个客户端的运行时都具备的功能来实现。

同样，JMS 提供者可以扮演多种角色：它可以作为独立产品创建，也可以作为大型分布式运行时系统中的嵌入式组件创建。作为独立产品时，它可以用于定义企业应用程序集成系统的主干；在嵌入到应用服务器中时，它可以支持组件间消息传送。例如，J2EE 使用 JMS 提供者来实现消息驱动 Bean 并允许 EJB 组件发送和接收消息。

如果所创建的是一个包括现有系统所有功能的标准，则会导致系统既难以学习，又难以实现。而 JMS 定义了消息传送概念和功能的一个共同特点。这将导致所创建的标准易于学习，并最大限度地提高了 JMS 应用程序在 JMS 提供者之间的可移植性。一定要注意， JMS 是 API 标准，而不是协议标准。可以很容易地将 JMS 客户端从一个供应商移到另一个供应商。但是，不同的 JMS 供应商之间通常不能直接通信。

下一节将介绍 JMS 规范定义的基本对象和消息传送模式。

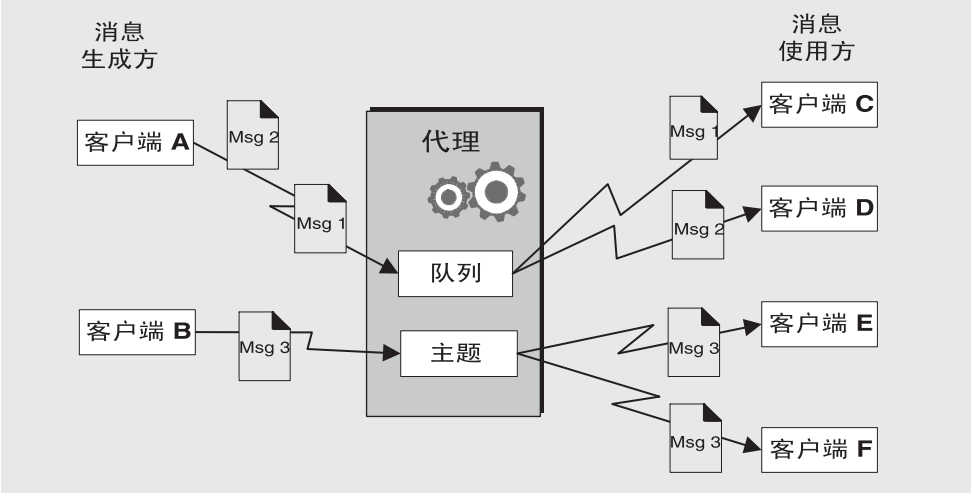
JMS 消息传送对象和模式

为了发送或接收消息， JMS 客户端必须首先连接到通常作为消息代理实现的 JMS 提供者：此连接在客户端与代理之间打开一个通信通道。接下来，客户端必须设置一个用来创建、生成和使用消息的会话。可以将该会话视为定义客户端与代理之间的特定对话的消息流。客户端本身是**消息生成方** 和 / 或**消息使用方**。消息生成方向代理所管

理的**目标**发送一条**消息**。消息使用方访问该目标以使用此消息。该消息包括头、属性（可选）和主体。消息主体用来保存数据；消息头中包含代理路由和管理消息所需的信息；属性可以由客户端应用程序或提供者定义，以满足处理消息的需要。连接、会话、目标、消息、生成方和使用方是构成 JMS 应用程序的基本对象。

有了这些基本对象，客户端应用程序就可以使用两种消息传送模式（或域）来发送和接收消息。图 1-4 对此进行了说明。

图 1-4 JMS 消息传送模式



客户端 A 和 B 是消息生成方，它们通过两种不同类型的目标向客户端 C、D 和 E 发送消息。

- 客户端 A、C 和 D 之间的消息传送说明了点对点模式。客户端使用此模式向队列目标发送一条消息，只有一个接收者能够从该目标获得该消息。访问该目标的其他任何接收者都不能获得该消息。
- 客户端 B、E 和 F 之间的消息传送说明了发布 / 订阅模式。客户端使用此广播模式向主题目标发送一条消息，任意数量的使用方订户都可以从该目标检索此消息。每个订户都获得此消息的一个副本。

任何域中的消息使用方都可以选择同步或异步获取消息。同步使用方通过进行显式调用来检索消息；异步使用方则指定一个回调方法，将调用该回调方法来传递待处理的消息。使用方还可以通过为传入消息指定选择标准来过滤消息。

受管理对象

JMS 规范创建了一个标准，该标准结合了现有 MOM 系统的许多元素，但并不试图穷举所有可能的元素。相反，它试图设置一个可扩展的方案来兼顾不同元素之间的区别并适应将来的发展。JMS 将许多消息传送元素留给各个提供者来定义和实现。其中包括负载平衡、标准错误消息、管理 API、安全性、底层线路协议以及消息存储。下一节 （“Message Queue：元素和功能”）将介绍 Message Queue 如何实现其中的许多元素以及如何扩展 JMS 规范。

连接工厂和目标是 JMS 未完全定义的两个消息传送元素。尽管这些元素是 JMS 编程模型中的基础元素，但在提供者定义和管理这些对象的方式上，存在许多现有的和预期的区别，以致于不可能也不值得创建一个公共的定义。因此，这两个对象通常使用管理工具来创建和配置，而不是以编程方式来创建。它们随后将存储在对象存储中，JMS 客户端可以通过标准的 JNDI 查找功能来访问它们。

- **连接工厂受管理对象**用于生成客户端与代理的连接。它们封装特定于提供者的信息，这些信息控制消息传送行为的某些方面：连接处理、客户端标识、消息头覆盖、可靠性和流控制等。源自给定连接工厂的每个连接都表现出为该工厂配置的行为。
- **目标受管理对象**用于引用代理中的物理目标。它们封装特定于提供者的命名（地址 - 语法）约定，并指定使用该目标的消息传送域：队列或主题。

JMS 客户端不是必须查找受管理对象；它们可以通过编程方式创建这些对象 （这些对象随后将存储在代理的内存中）。要快速建立原型，最方便的方法可能是以编程方式创建这些对象。但是，如果要在生产环境中部署，则在中心系统信息库中查找受管理对象会更便于控制和管理消息传送行为：

- 通过对连接工厂对象使用管理对象，管理员可以通过重新配置这些对象来调整消息传送性能。不必重新编码即可改善性能。
- 通过对物理目标使用管理对象，管理员可以通过要求客户端访问这些预配置的对象来控制这些目标在代理上的扩散。
- 受管理对象使开发者无需了解特定于提供者的实现详细信息，而允许将他们针对某个提供者开发的代码在无需更改或只需稍作更改的情况下移植到其他提供者。

如图 1-5 中所示，受管理对象的使用是基本 JMS 应用程序图中的最后一项关键技术。

图 1-5 JMS 应用程序的基本元素

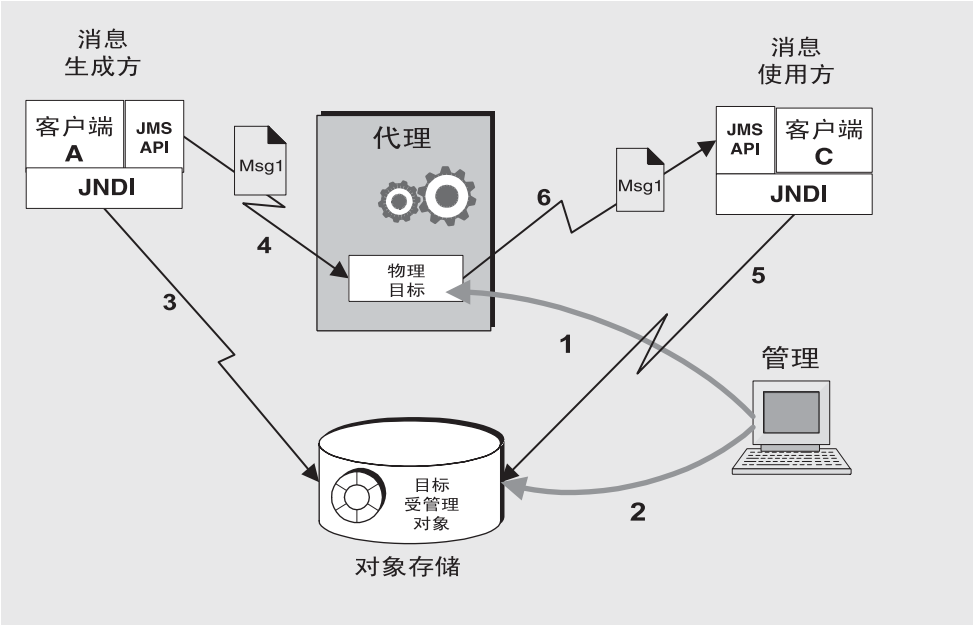


图 1-5 显示消息生成方和消息使用方如何使用目标受管理对象来访问对应于该对象的物理目标。带编号的步骤是管理员和客户端应用程序在使用此机制发送和接收消息时需要执行的操作。

1. 管理员在代理上创建一个物理目标。
2. 管理员创建一个目标受管理对象，并通过指定对应于该对象的物理目标的名称和该对象的类型（队列或主题）来对其进行配置。
3. 消息生成方使用 JNDI 查找调用来查找目标受管理对象。
4. 消息生成方向目标发送消息。
5. 消息使用方查找应当从中获取消息的目标受管理对象。
6. 消息使用方从该目标获取消息。

连接工厂受管理对象的使用过程与此类似。管理员使用管理工具来创建和配置连接工厂受管理对象。客户端查找连接工厂对象并使用它来建立连接。

尽管使用受管理对象会为消息传送过程添加几个步骤，但它也提高了消息传送应用程序的稳定性和可移植性。

Message Queue：元素和功能

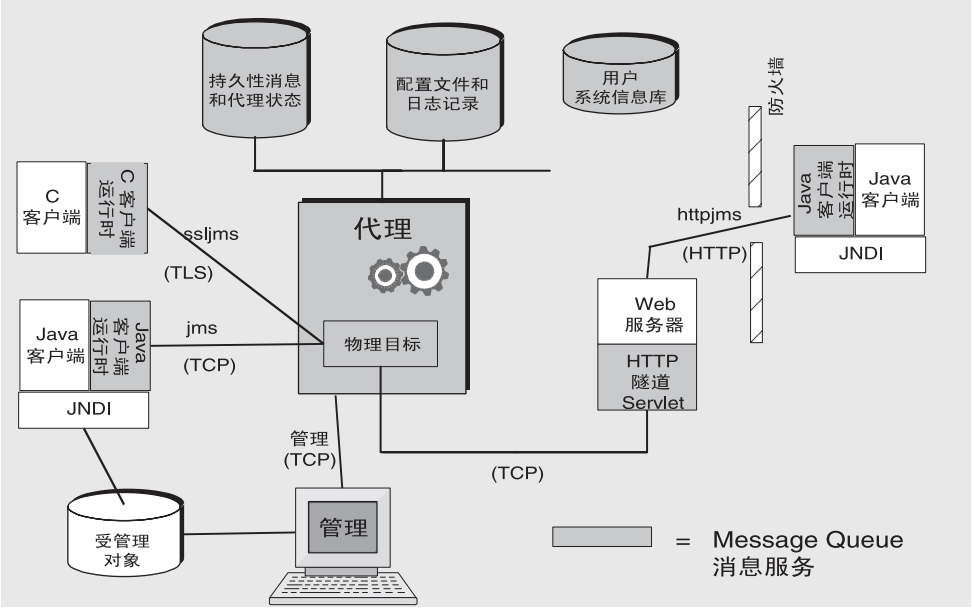
至此，我们已经介绍了面向消息的中间件的元素，并指出使用 JMS 可提高 MOM 应用程序的可移植性。接下来介绍 Message Queue 如何实现 JMS 规范，并介绍它用来提供可靠、安全且可扩展的消息传送服务的功能和工具。

首先，与许多 JMS 提供者一样， Message Queue 既可以用作独立产品，也可以作为启用技术嵌入到 J2EE 应用服务器中以提供异步消息传送。第 5 章“Message Queue 和 J2EE”对 Message Queue 在 J2EE 中扮演的角色进行了更详细的介绍。与其他 JMS 提供者不同的是， Message Queue 已被指定为 JMS 参考实现。这一指定证明了这样一个事实：Message Queue 是正确而又完整的 JMS 实现。它还保证了 Message Queue 产品将与以后推出的任何 JMS 修订和扩展保持同步。

Message Queue 服务

作为一个 JMS 提供者，Message Queue 提供一个实现 JMS 接口并提供管理服务和控制的**消息传送服务**。至此，我们已在说明 JMS 提供者时重点介绍了代理在消息转发过程中扮演的角色。但实际上，除了代理外，JMS 提供者中还必须包括许多元素才能提供可靠、安全且可扩展的消息传送。图 1-6 显示了构成 Message Queue 消息服务的元素。这些元素包括各种连接服务（支持不同的协议）、管理工具以及用于消息传送功能、监视功能和用户信息的数据存储。 Message Queue 服务包括该图中用灰色标记的所有元素。

图 1-6 Message Queue 服务



正如您所看到的那样，功能全面的 JMS 提供者比基本 JMS 模型更为复杂。这很容易引起我们的怀疑。以下各节将介绍上面显示的 Message Queue 服务元素。这些元素可以分成三类：代理、客户端运行时支持和管理。

连接到代理

如图 1-6 所示，应用程序客户端和管理客户端都可以连接到代理。JMS 规范未规定提供者实现任何特定的线路协议。Message Queue 服务当前位于 TCP、TLS、HTTP 或 HTTPS 协议的上一层，供应用程序客户端和管理客户端用来连接到代理。（位于 HTTP 上一层的服务使消息可以穿过防火墙。）

- 提供 JMS 支持并使客户端可以连接到代理的服务（`jms`、`ssljms`、`http` 或 `https`）的服务类型为 `NORMAL`，它们位于 `TCP`、 `TLS`、 `HTTP` 或 `HTTPS` 协议的上一层。
- 使管理员可以连接到代理的服务（`admin` 和 `ssladmin`）的服务类型为 `ADMIN`，它们位于 `TCP` 或 `TLS` 协议的上一层。

默认情况下，当启动代理时，会启动并运行 `jms` 和 `admin` 服务。此外，还可以将代理配置为运行上述任一或全部连接服务。每个服务都支持特定的验证和授权 （访问控制）功能，每个服务都是多线程的并且支持多个连接。

当连接失败时， Message Queue 服务能够自动重新尝试将客户端连接到同一代理或另一代理 （如果启用了此功能） 。有关详细信息，请参见附录 B “Message Queue 功能”中有关自动重新连接功能的描述。

当客户端创建连接工厂 （它们将从中获取连接）时，可以配置连接运行时支持。可以通过选项来指定要连接到的代理、重新连接的处理方式、消息流控制等。有关如何配置连接的其他信息，请参见第 45 页上的 “连接工厂和连接”。

代理

代理是消息服务的核心，它以可靠的方式路由和传送消息，对用户进行验证并收集用来监视性能的数据。

- 为路由和传送消息，代理将传入的消息放入各自的目标并管理传入和传出这些目标的消息流。
- 为了提供可靠的传送，代理使用持久性存储来保存状态信息和持久性消息，直到它们被接收。如果代理或连接失败，所保存的信息使代理可以恢复代理的状态并重试操作。
- 要为所交换的数据提供安全性，代理需要使用经过验证的连接。可以选择通过在安全协议（如 SSL）上运行来对数据进行加密。代理还使用并管理一个系统信息库，该系统信息库保存有关用户以及他们可以访问的数据或操作的信息。代理对请求服务的用户进行验证，并通过在该系统信息库中查找信息来授予他们执行相应操作的权限。
- 要监视系统，代理需要生成度量和诊断信息，以便管理员可以通过访问这些信息来度量性能并调整代理。度量信息还可以通过编程方式来使用，这使应用程序可以调整消息流和模式，从而提高性能。

Message Queue 服务提供各种管理工具，管理员可以使用这些工具来配置代理支持。有关详细信息，请参见第 33 页上的 “管理”。

客户端运行时支持

客户端运行时支持在构建 Message Queue 客户端时所链接到的库中提供。可以将客户端运行时视为已成为客户端一部分的 Message Queue 服务的代码。例如，当客户端代码进行 API 调用以发送消息时，将调用这些库中的代码，以便根据用来将消息转发到代理上的物理目标的协议来相应地包装消息位。

Java 和 C 客户端支持

只有当需要支持 Java 客户端时，才需要 JMS 提供者；但是，如图 1-6 所示，Message Queue 客户端可以使用 Java 或特定于提供者的 C API 来发送或接收消息。这些接口是在 Java 或 C 运行时库中实现的，这些库的实际作用是建立与代理的连接并根据所请求的连接服务来相应地包装位。

- Java 客户端运行时向 Java 客户端提供与代理进行交互所需的对象。这些对象包括连接、会话、消息、消息生成方和消息使用方。
- C 客户端运行时向 C 客户端提供与代理进行交互所需的功能和结构。它支持 JMS 编程模型的程式化版本。 C 客户端不能使用 JNDI 来访问受管理对象，但是可以通过编程方式来创建连接工厂和目标。

Message Queue 服务提供一个 C API，使传统 C 和 C++ 应用程序能够参与基于 JMS 的消息传送。这两个 API 所提供的功能有许多不同；第 57 页上的“Java 客户端与 C 客户端”对此进行了说明。

请记住，JMS 规范是仅适用于 Java 客户端的标准。C 支持特定于 Message Queue 提供者，因此在计划移植到其他提供者的客户端应用程序中不应该使用该支持。

对 Java 客户端的 SOAP 支持

Message Queue Java 客户端还能够发送和接收包装为 JMS 消息的 SOAP 消息。使用 SOAP （Simple Object Access Protocol，简单对象访问协议）可以实现在分布式环境中的两个对等方之间交换结构化数据。所交换的数据由 XML 方案指定。

Sun SOAP 处理当前仅限于使用点对点模型并且不能保证可靠性。通过将 SOAP 消息包装到 JMS 消息中并使用代理来路由它，可以利用全功能的 Message Queue 消息传送，从而保证可靠的传送并允许您使用主题和点对点域。 Message Queue 提供实用程序例程，使用这些例程，消息生成方可以将 SOAP 消息包装到 JMS 消息中，消息使用方可以从 JMS 消息中提取 SOAP 消息。

第 56 页上的 “使用 SOAP 消息”更详细地介绍了 SOAP 消息处理。

管理

Message Queue 服务提供可用来执行以下操作的命令行工具：

- 启动和配置代理。
- 创建和管理目标、管理代理连接以及管理代理资源。
- 添加、列出、更新和删除 JNDI 对象存储中的受管理对象。

- 填充和管理基于文件的用户系统信息库。
- 为持久性存储器创建和管理符合 JDBC 的数据库。

还可以使用基于 GUI 的管理控制台来执行以下命令行功能：

- 连接到代理并对它进行管理。
- 创建和管理物理目标。
- 连接到对象存储、向其中添加对象并管理这些对象。

扩展 Message Queue 服务

随着客户端或连接数量的增加，您可能需要扩展消息服务以消除瓶颈或改善性能。Message Queue 消息服务根据您的需要提供许多扩展选项。可以很容易地将这些选项归为以下几类：

- **垂直扩展**是通过添加处理能力和扩展可用资源来实现的。这可以通过添加更多处理器或内存，切换到共享线程模型或者在 64 位模式下运行 Java VM 来完成。

  如果您使用的是点对点域，则可以通过允许多个使用方访问一个队列来扩展使用方。使用此方法，可以指定活动使用方和备份使用方的最大数量。负载平衡机制还将使用方的当前容量和消息处理速率考虑在内。这是 Message Queue 的一个功能。（JMS 规范定义当只有一个使用方访问队列时的消息传送行为；允许多个使用方访问的队列的行为是特定于提供者的。 Message Queue 开发者指南提供了有关此扩展选项的更多信息。）

- **无状态水平扩展**是通过使用其他代理并将现有的客户端重新分配给这些代理来实现的。此方法易于实现，但是只有当消息传送操作可以分成多个独立的工作组时，此方法才适用。

- **有状态水平扩展**是通过将代理连接成**群集**来实现的。在代理群集中，每个代理不仅连接到群集内的其他每个代理，而且还连接到本地应用程序客户端。这些代理可以位于同一个主机上，也可以分布在网络中。有关目标和使用方的信息会在群集内的所有代理中复制。目标和订户的更新也会进行传播。因此，每个代理都可以将消息从与它直接相连的生成方路由到与群集内的其他代理连接的使用方。当使用备份使用方时，如果某个代理或连接失败，则发送到不可访问的使用方的消息可以转发到另一个代理上的备份使用方。

  如果代理或连接失败，则有关持久性实体（目标和长期订阅）的状态信息可能会失去同步。例如，如果当某个群集代理停机时，在群集内的另一个代理上创建了一个目标，则当前者重新启动时，将不会知道有这个新目标。要防止出现此问题，可以将群集内的某个代理指定为**主代理**。此代理负责跟踪**主配置文件**中目标和长期订阅的所有更改，还负责更新群集内临时脱机的代理。有关其他信息，请参见第 4 章 “代理群集”。

即使在使用主代理时，当代理或连接失败时，Message Queue 也只是提供服务可用性，而不提供数据可用性。例如，如果群集代理变得不可用，则在该代理恢复之前，由该代理保存的任何持久性消息都将不可用。目前，只能通过使用 SunCluster Message Queue 代理来确保数据可用性。在这种情况下，持久性存储保留在共享文件系统上。如果某个代理失败，则第二个节点上的 Message Queue 代理会启动一个接管共享存储的代理。客户端会重新连接到该代理，从而既可以获得不间断的服务，又可以访问持久性数据。

Message Queue 作为启用技术

Java 2 Platform, Enterprise Edition （J2EE 平台）是一种面向 Java 编程环境中分布式组件模型的规范。 J2EE 平台的一项要求是，分布式组件之间必须能够通过可靠的异步消息交换来进行交互。此功能是通过可扮演以下两种角色的 JMS 提供者来提供的：它可用于提供服务并且可以支持消息驱动 Bean (message-driven bean, MDB)，MDB 是一种可以使用 JMS 消息的专用 Enterprise Java Bean (EJB) 组件。

符合 J2EE 的应用服务器必须使用由给定的 JMS 提供者提供的资源适配器才能使用该提供者的功能。Message Queue 提供这样的资源适配器。使用插入到 JMS 提供者的支持，在应用服务器环境中部署和运行的 J2EE 组件 （包括 MDB）既可以彼此交换 JMS 消息，也可以与外部 JMS 组件交换消息。这为分布式组件提供了强大的集成功能。

有关 Message Queue 资源适配器的信息，请参见第 5 章 “Message Queue 和 J2EE”。

产品版本

Message Queue 有两个可用的版本：Enterprise Edition 和 Platform Edition。两个版本都完全实现了 JMS 规范，只是它们所对应的功能集和许可功能不同。

Enterprise Edition 在 Platform Edition 的基础上添加了以下功能：

- HTTP 和 HTTPS 支持
- C 客户端支持
- 可扩展的连接功能
- 代理群集

- 每个队列都可以向任意多个消息使用方传送消息。（Platform Edition 将传送限制在每个队列三个使用方。）
- 可以将客户端连接故障转移到群集内的其他代理。
- 基于消息的监视 API

Message Queue Enterprise Edition 许可证有效期理论上是无期限的，但实际受使用的 CPU 数量的限制。许可证对多代理消息服务中的代理数量没有限制。

Message Queue Platform Edition 对代理所支持的客户端连接的数量没有限制。它提供基本许可证或 90 天试用许可证：

- **基本许可证**没有期限限制，但是不包括 Enterprise Edition 的功能。
- **90 天试用企业许可证**允许您使用和评估 Enterprise Edition 的所有功能，但是该软件将此许可证的有效期限强制限制为 90 天。有关使用此许可证的说明，请参见 Message Queue 管理指南中讨论的启动选项。

有关许可功能、重新分发权限以及将 Platform Edition 升级到 Enterprise Edition 的进一步信息，请参见 Message Queue Installation Guide。

Message Queue 功能概述

Message Queue 拥有的功能远远超出了 JMS 规范的要求。这些功能使 Message Queue 可以集成由大量分布式组件组成的系统，这些组件在全天候的关键任务操作中交换数以万计的消息。有关这些功能的概述，请参见附录 B “Message Queue 功能”。

第 2 章

客户端编程模型

本章介绍 Message Queue 客户端编程的基本知识，涵盖了以下主题：

本章着重介绍 Java 客户端的设计与实现。C 客户端设计与 Java 客户端设计大体上是平行的。本章的最后一节概述 Java 客户端与 C 客户端之间的差异。有关 Message Queue 客户端编程的详细论述，请参见 Message Queue Developer's Guide for Java Clients 和 Message Queue Developer's Guide for C Clients。

第 3 章 “Message Queue 服务” 说明如何使用 Message Queue 服务来支持、管理和调整消息传送性能。

设计与性能

Message Queue 应用程序的行为取决于多个因素：客户端设计、连接配置、代理配置、代理调整及资源管理。在这些因素中，一部分是开发者的责任，另外一部分则是管理员的责任。不过，最理想的情况是开发者了解 Message Queue 服务如何支持和调整应用程序设计，而管理员在调整应用程序时了解设计目标。通过重新设计以及仔细的监视和调整，可以优化消息传送性能。所以，开发出高性能 Message Queue 应用程序的关键是开发者和管理员了解应用程序生命周期各阶段可实现的目标，并交流有关预期性能及观测性能的信息。

消息传送域

消息传送中间件使各组件和应用程序可通过生成并使用消息来进行通信。 JMS API 定义管理该通信的两种模式或**消息传送域：点对点消息传送**和**发布 / 订阅消息传送**。JMS API 被组织为支持以上模式。基本 JMS 对象：用于指定两个域中的消息传送行为的连接、会话、生成方、使用方、目标和消息。

点对点消息传送

在点对点域中，消息生成方被称为**发送者**，而使用方则被称为**接收者**。它们通过被称为**队列**的目标来交换消息：发送者**生成**队列中的消息；而接收者则**使用**队列中的消息。

图 2-1 显示了点对点域中最简单的消息传送操作。`MyQueueSender` 向队列目标 `MyQueue1` 发送 `Msg1`。然后， `MyQueueReceiver` 从 `MyQueue1` 获得该消息。

图 2-1 简单点对点消息传送

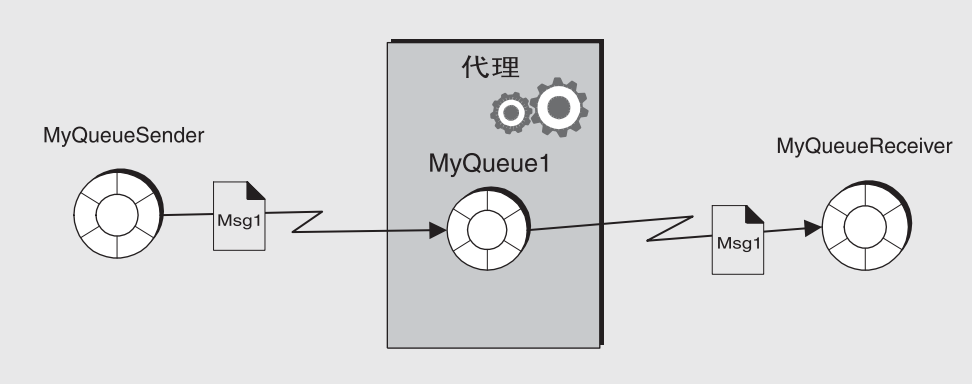
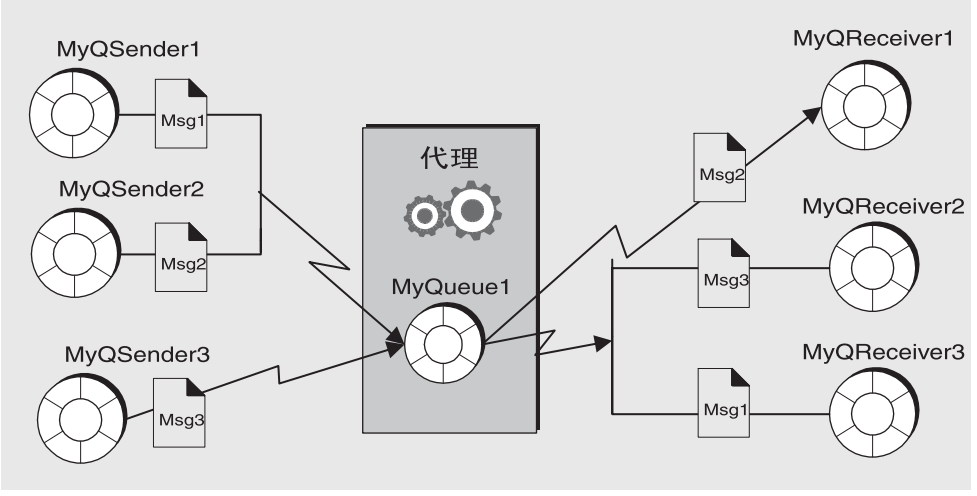


图 2-2 显示了一个更为复杂的点对点消息传送图，以说明该域中的可能情况。两个发送者 MyQSender1 和 MyQSender2 使用同一连接向 MyQueue1 发送消息。 MyQSender3 使用另一连接向 MyQueue1 发送消息。在接收端， MyQReceiver1 使用 MyQueue1、MyQReceiver2 和 MyQReceiver3 中的消息，共享一个连接以使用 MyQueue1 中的消息。

图 2-2 复杂点对点消息传送



该图略为复杂，说明了有关点对点消息传送的其他几点。

- 多个生成方可向一个队列发送消息。生成方可共享连接或使用不同连接，但它们均可访问同一队列。
- 多个接收者可使用一个队列中的消息，但每条消息只能由一个接收者使用。所以，`Msg1`、`Msg2` 和 `Msg3` 由不同接收者使用。（这是对 Message Queue 的扩展。）
- 接收者可共享连接或使用不同连接，但它们均可访问同一队列。（这是对 Message Queue 的扩展。）
- 发送者和接收者之间不存在时间上的相关性：客户端发送一条消息后，无论接收者是否正在运行，都能取出该消息。
- 可在运行时动态添加和删除发送者和接收者，这样，即可根据需要扩展或收缩消息传送系统。
- 消息在队列中的放置顺序与发送顺序相同，但它们的使用顺序则取决于消息失效期、消息优先级以及使用消息时是否使用选择器等因素。

点对点模型具有许多优势：

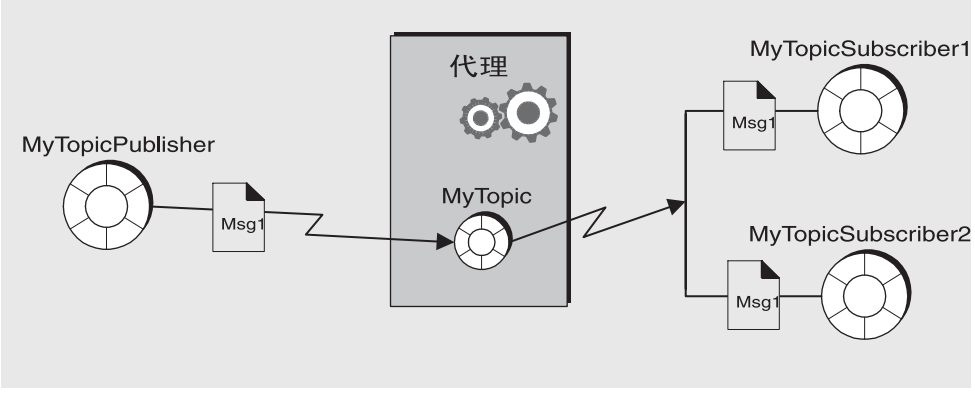
- 由于多个接收者可使用同一队列中的消息，因此如果接收消息的顺序无关紧要，那么您可以平衡消息使用负载。（这是对 Message Queue 的扩展。）
- 要发送到队列的消息始终保留，即使没有接收者也是如此。
- Java 客户端可使用队列浏览器对象检查队列内容。然后，它们可以根据通过该检查所获得的信息来使用消息。也就是说，虽然使用模型一般为 FIFO （first in, first out，先进先出），但如果使用者知道要使用哪些消息，即可使用消息选择器来使用未处于队列最前方的消息。管理客户端也可以使用队列浏览器来监视队列的内容。

发布 / 订阅消息传送

在发布 / 订阅域中，消息生成方被称为**发布者**，而消息使用方则被称为**订户**。它们通过称为**主题**的目标来交换消息：发布者生成主题中的消息；订户则**订阅**主题并**使用**主题中的消息。

图 2-3 显示了发布 / 订阅域中的简单消息传送操作。 MyTopicPublisher 向目标 MyTopic 中发布 Msg1。然后， MyTopicSubscriber1 和 MyTopicSubscriber2 均从 MyTopic 接收 Msg1 的副本。

图 2-3 简单发布 / 订阅消息传送

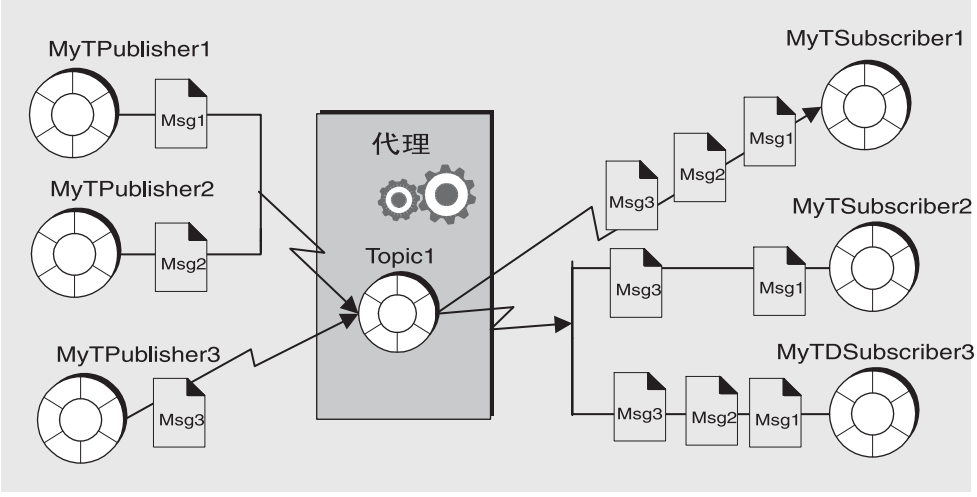


虽然发布 / 订阅模型不需要存在多个订户，但图中显示了两个订户，这是为了强调在该域中可以广播消息。一个主题的所有订户均可获得发布到该主题的任何消息的副本。

订户可以是非长期的，也可以是长期的。代理会为所有活动订户保留消息，但对于非活动订户，则只为那些长期订户保留消息。

图 2-4 显示了更为复杂的发布 / 订阅消息传送图，以说明该模式提供的可能情况。多个生成方向 Topic1 目标发布消息。多个订户使用来自 Topic1 目标的消息。除非订户使用选择器来过滤消息，否则每个订户均可获得发布到所选主题的所有消息。在图 2-4 中， MyTSubscriber2 已过滤掉 Msg2。

图 2-4 复杂发布 / 订阅消息传送



该图略为复杂，说明了有关发布 / 订阅消息传送的其他几点。

- 多个生成方可向一个主题发布消息。生成方可共享连接或使用不同连接，但它们均可访问同一主题。
- 多个订户可使用一个主题中的消息。订户可检索发布到一个主题中的所有消息，除非它们使用选择器过滤掉消息或消息在使用之前已过期。
- 订户可共享连接或使用不同连接，但它们均可访问同一主题。
- 长期订户可能处于活动状态，也可能处于非活动状态。在它们处于非活动状态时，代理会为它们保留消息。
- 可在运行时动态添加和删除发布者和订户，这样，即可根据需要扩展或收缩消息传送系统。
- 消息发布到主题的顺序与发送顺序相同，但它们的使用顺序则取决于消息失效期、消息优先级以及使用消息时是否使用选择器等因素。
- 发布者与订户之间存在时间上的相关性：主题订户只能使用在它创建订阅后发布的消息。

发布 / 订阅模型的主要优势在于它允许向订户广播消息。

特定于域的 API 及统一域 API

JMS API 定义可用于实现点对点域或发布 / 订阅域的接口和类。这些是表 2-1 的第 2 列和第 3 列中显示的**特定于域的** API。JMS API 还定义另外一个**统一**域，用于通过编程实现常规的消息传送客户端。这类客户端的行为由目标类型决定，客户端向目标中生成消息并使用目标中的消息。如果该目标是一个队列，则消息传送行为将为点对点模式；如果该目标是一个主题，则消息传送行为将为发布 / 订阅模式。

表 2-1 JMS 编程域和对象

基本类型 （统一域）	点对点域	发布 / 订阅域
Destination （Queue 或 Topic）*	Queue	Topic
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Connection	QueueConnection	TopicConnection
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver	TopicSubscriber

* 根据编程方法，您必须指定特定的目标类型。

JMS 版本 1.1 中引入了统一域。如果需要遵循早期的 1.02b 规范，可以使用特定于域的 API。使用特定于域的 API 还能够提供全新编程接口，可以防止出现某些类型的编程错误：例如，为队列目标创建长期订户。不过，特定于域的 API 也有缺点，即无法合并同一事务或会话中的点对点操作及发布 / 订阅操作。如果需要执行该操作，则应选择统一域 API。有关组合两种域的示例，请参见第 50 页上的 “请求 - 回复模式”。

编程对象

用于实现 JMS 消息传送的对象在编程域中基本保持不变：连接工厂、连接、会话、生成方、使用方、消息和目标。这些对象显示在图 2-5 中。图中从连接工厂对象开始，自上而下地显示对象是如何派生出来的。

显示的连接工厂和目标这两类对象驻留在对象存储中。这是为了强调通常将这些对象作为受管理对象来创建、配置和管理。我们假定本章中的连接工厂和目标都是以管理方式 （而不是以编程方式）创建的。

图 2-5 JMS 编程对象

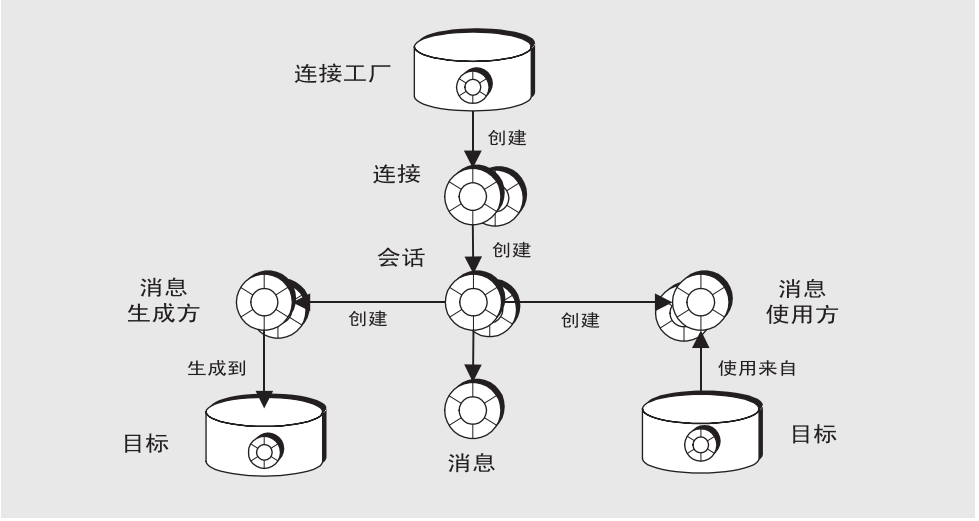


表 2-2 概述了发送和接收消息所需的步骤。请注意，步骤 1 以及步骤 3 到 6 对于发送者和接收者来说是相同的。

表 2-2 生成和使用消息

生成消息	使用消息
1. 管理员创建连接工厂受管理对象。	
2. 管理员创建物理目标以及引用该目标的受管理对象。	
3. 客户端通过 JNDI 查找获得连接工厂对象。	
4. 客户端通过 JNDI 查找获得目标对象。	
5. 客户端创建连接并设置特定于该连接的所有属性。	
6. 客户端创建会话并设置决定消息传送可靠性的属性。	
7. 客户端创建消息生成方。	客户端创建消息使用方。
8. 客户端创建消息。	客户端建立连接。
9. 客户端发送消息。	客户端接收消息。

以下几节将介绍生成方和使用方使用的对象：连接、会话、消息和目标。之后，我们将通过说明消息的生成和使用来完成对 JMS 对象的介绍。

连接工厂和连接

客户端使用 connection factory （连接工厂）对象 (ConnectionFactory) 来创建 connection （连接）。连接对象 (Connection) 表示客户端与代理之间的活动连接。它使用在默认情况下启动或者由管理员为该客户端明确启动的底层连接服务。

创建连接时，将分配通信资源并验证客户端。这是一个相当重要的对象，大多数客户端均使用一个连接来完成所有的消息传送。连接支持并发使用：一个连接可由任意数量的生成方和使用方共享。

创建连接工厂时，可通过设置它的属性来配置从它派生的所有连接的行为。对于 Message Queue，这些设置可指定以下信息：

- 代理所在主机的名称、需要的连接服务以及客户端用于访问该服务的端口。
- 连接失败时应如何处理与代理的自动重新连接。（如果连接断开，该功能会将客户端重新连接到相同（或不同）的代理。但不保证进行数据故障转移：如果重新连接到不同代理，持久性消息及其他状态信息可能会丢失。）
- 需要代理跟踪长期订阅的客户端 ID。
- 尝试进行连接的用户的默认名称和密码。如果连接时未指定密码，则该信息用于验证用户和授权操作。
- 对于已确认可靠性的那些客户端是否应抑制代理确认。
- 如何管理代理与客户端运行时之间的控制流和有效负荷消息。
- 应如何处理队列浏览。（仅限于 Java 客户端。）
- 是否应覆盖某些消息头字段。

可以在用于启动客户端应用程序的命令行上覆盖连接工厂属性。还可以通过设置任意给定连接的属性来覆盖该连接的属性。

可使用连接对象来创建 session（会话）对象，从而设置异常侦听器或者获得 JMS 版本及提供者信息。

会话

如果连接代表客户端与代理之间的通信渠道，则会话标记客户端与代理之间的单次对话。会话对象主要用于创建消息、消息生成方和消息使用方。创建会话时，通过多个 acknowledgement （确认）选项或通过事务来配置可靠传送。有关详细信息，请参见第 52 页上的 “可靠消息传送”。

从 JMS 规范可看出，会话是生成和使用消息的单线程上下文。可以为一个会话创建多个消息生成方和使用方，但只能顺次使用它们。 Java 客户端与 C 客户端在线程实现上的差异非常小。有关线程实现与限制的其他信息，请参考相应的开发者指南。

还可以使用会话对象来完成以下任务：

- 为不使用受管理对象定义目标的那些客户端创建和配置目标。
- 创建并配置临时主题和队列；它们用作请求-回复模式的一部分。请参见第 50 页上的 “请求 - 回复模式”。
- 支持事务处理。
- 定义生成或使用消息的顺序。
- 为异步使用方序列化消息侦听器的执行。
- 创建队列浏览器。（仅限于 Java 客户端。）

消息

消息由三部分组成：消息头、属性和主体。您必须了解此结构，才能正确编写消息和配置某些消息传送行为。

消息头

每条 JMS 消息都必须有消息头。消息头包含十个预定义字段，表 2-3 中列出并介绍了这些字段。

表 2-3 JMS 定义的消息头

头字段	描述
JMSDestination	指定消息要发送到的目标对象的名称。（由提供者设置。）
JMSDeliveryMode	指定消息是否为持久性消息。（默认情况下由提供者设置，也可以由客户端为生成方或为单独的消息显式设置。）
JMSExpiration	指定消息的到期时间。（默认情况下由提供者设置，也可以由客户端为生成方或为单独的消息设置。）
JMSPriority	指定消息的优先级，范围为 0（低）到 9（高）。（默认情况下由提供者设置，也可以由客户端为生成方或为单独的消息显式设置。）
JMSMessageID	指定消息在提供者安装的上下文中的唯一 ID。（由提供者设置。）
JMSTimestamp	指定提供者接收消息的时间。（由提供者设置。）
JMSCorrelationID	客户端用于定义两条消息之间的对应性的值。（由客户端在需要时设置。）

表 2-3 JMS 定义的消息头 （续）

头字段	描述
JMSReplyTo	指定使用方应发送回复的目标。（由客户端在需要时设置。）
JMSType	可由消息选择器评估的值。（由客户端在需要时设置。）
JMSRedelivered	指定消息是否已传送但尚未得到确认。（由提供者设置。）

通读上表可知，消息头字段可用于多种用途：标识消息，配置消息路由，提供有关消息处理的信息等等。

JMSDeliveryMode 是最重要的字段之一，它决定了消息传送的可靠性。该字段指示一条消息是否为持久性消息。

- 保证只将**持久性消息**传送并成功使用一次。如果消息服务失败，持久性消息不会丢失。
- 保证最多将**非持久性消息**传送一次。如果消息服务失败，非持久性消息将丢失。

部分消息头字段由提供者 （代理或客户端运行时）设置，其他头字段则由客户端设置。消息生成方可能需要配置头字段值，才能实现某些消息传送行为；消息使用方可能需要读取头字段值，才能了解消息的路由方式以及可能需要对它执行哪些进一步处理。

可以在三个不同级别设置头字段（JMSDeliveryMode、JMSExpiration 和 JMSPriority）：

- 对于通过从连接工厂派生的每个连接发出的消息。
- 对于生成的每条消息。
- 对于特定消息生成方发出的所有消息。

如果这些字段是在多个级别设置的，则为连接工厂设置的值会覆盖为单独的消息设置的那些值；而为给定消息设置的值会覆盖为消息生成方设置的那些值。

消息头字段的固定名称因语言实现而异。有关详细信息，请参见 Message Queue Developer's Guide for Java Clients 或 Message Queue Developer's Guide for C Clients。

消息属性

消息还可以包含称为**属性**的可选头字段，这类字段以属性名 / 属性值对的形式来指定。客户端和提供者可以使用属性来扩展消息头，并可以在其中包含有助于客户端或提供者标识和处理消息的任何信息。利用消息属性，接收客户端可以要求仅传送符合给定标准的那些消息。例如，使用方客户端可能请求获得有关新泽西州兼职雇员工资单的消息。提供者将不会传送不符合指定标准的消息。

JMS 规范定义九个标准属性。其中部分由客户端设置，部分由提供者设置。这些属性的名称以保留字符 "JMSX" 开头。客户端或提供者可以使用这些属性来确定消息发送者、消息的状态以及传送的频率和时间。这些属性有助于提供者提供路由消息和诊断信息。

Message Queue 也定义消息属性，它们用于标识压缩消息以及在无法传送消息时应如何处理消息。有关详细信息，请参见 Message Queue Developer's Guide for Java Clients。

消息主体

消息主体包含客户端需要交换的数据。

JMS 消息类型决定了主体可以包含哪些内容以及使用方应如何处理主体，如表 2-4 中所指定的那样。 Session 对象包含各类消息主体的创建方法。

表 2-4 消息主体类型

类型	描述
StreamMessage	主体中包含 Java 基元值流的消息。它的填充和读取均按顺序进行。
MapMessage	主体中包含一组名 / 值对的消息。没有定义条目顺序。
TextMessage	主体中包含 Java 字符串的消息，例如 XML 消息。
ObjectMessage	主体中包含序列化 Java 对象的消息。
BytesMessage	主体中包含连续字节流的消息。

Java 客户端可设置相应的属性，以使客户端运行时压缩发送的消息的主体。使用方一端的 Message Queue 运行时会在传送消息前先解压缩消息。

生成消息

消息由消息生成方在连接和会话的上下文中发送或发布。生成消息相当简单：客户端使用消息生成方对象 (`MessageProducer`) 向物理 destination（目标）（在 API 中由目标对象表示）发送消息。

创建生成方时，可指定生成方的所有消息将发送到的默认目标。您也可以指定决定持久性、优先级和有效期的消息头字段的默认值。这样，从该生成方发出的所有消息都使用这些默认值，除非您通过在发送消息时指定替代目标，或者为给定消息的头字段设置替代值来覆盖它们。

消息生成方也可以通过设置 JMSReplyTo 消息头字段来实现请求 - 回复模式。有关详细信息，请参见第 50 页上的 “请求 - 回复模式”。

使用消息

消息由消息使用方在连接和会话的上下文中接收。客户端使用消息使用方对象 (MessageConsumer) 从指定的物理目标 （在 API 中由目标对象表示）接收消息。

三个因素影响代理向使用方传送消息的方式：

- 使用是同步的还是异步的
- 是否使用选择器来过滤传入消息
- 如果使用主题目标中的消息，则订户是否为长期订户也会有影响

影响消息传送和客户端设计的另一主要因素是使用方需要的可靠度。请参见第 52 页上的 “可靠消息传送”。

同步和异步使用方

消息使用方可以支持同步或异步消息使用。

- 同步使用是指，使用方明确请求传送消息并随后使用该消息。

  根据请求消息时使用的方法，同步使用方可选择等待（无限期）消息到达，等待指定的时间或者在不存在可供使用的消息时立即返回。（“可供使用”是指对象立即可供客户端使用。已成功发送但代理尚未处理完毕的消息不可用。）

- 异步使用是指，自动将消息传送给为使用方注册的消息侦听器对象 (MessageListener)。当会话线程调用消息侦听器对象的 onMessage() 方法时，客户端将使用该消息。

使用选择器过滤消息

消息使用方可以利用消息选择器，使消息服务只传送那些属性匹配特定选择标准的消息。该标准在创建使用方时指定。

选择器使用类似 SQL 的语法来匹配消息属性。例如，

```
color = 'red'
```

```
size > 10
```

Java 客户端也可以在浏览队列时指定选择器；这样您可以查看在所选消息中，哪些消息正在等待被使用。

使用长期订户

可以使用会话对象创建主题的长期订户。即使订户变为非活动状态，代理也会为这些类型的订户保留消息。

由于代理必须维护订户的状态并在订户被重新激活后恢复消息传送，因此，代理必须能够在给定订户的订阅期内识别该订户。订户标识是根据创建该订户的连接的 `ClientID` 属性以及创建订户时指定的订户名构造的。

请求 - 回复模式

可以在一个连接中同时包含生成方和使用方（在使用统一 API 时甚至可包含会话）。此外， JMS API 允许使用临时目标来实现消息传送操作的请求 - 回复模式。

消息生成方必须执行以下操作才能设置请求 - 回复模式：

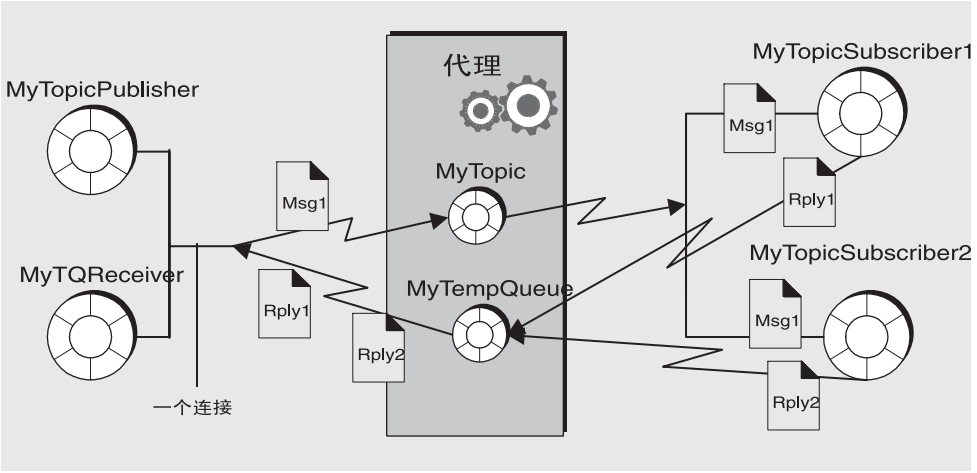
1. 创建使用方可发送回复的临时目标。
2. 在要发送的消息中，将消息头的 `JMSReplyTo` 字段设置为该临时目标。

消息使用方在处理消息时，会检查消息的 `JMSReplyTo` 字段，以确定是否需要回复并向指定目标发送回复。

如果采用请求 - 回复机制，生成方将无需为回复目标设置受管理对象，并且使用方可以轻松地响应请求。当生成方只有在确保请求已处理后才能继续时，此模式非常有用。

图 2-6 说明了向主题中发送消息并接收临时队列中的回复的请求 / 回复模式。

图 2-6 请求 / 回复模式



如图所示， MyTopicPublisher 生成 Msg1 并将它发送到目标 MyTopic。MyTopicSubsriber1 和 MyTopicSubscriber2 接收该消息并向 MyTempQueue 发送回复， MyTQReceiver 从 MyTempQueue 中检索该回复。该模式可能适用于向大量客户端发布定价信息，并将客户端的回复订单排队以便按顺序进行处理的应用程序。

临时目标的存在时间只能与创建它们的连接的存在时间一样长。任何生成方都可以向临时目标发送消息，但只有创建目标的连接所创建的那些使用方能够访问临时目标。

由于请求 / 回复模式依赖于创建临时目标，因此在以下情况下不应使用该模式：

- 如果预计创建临时目标的连接可能在回复发送前终止。
- 如果您需要向临时目标发送持久性消息。

可靠消息传送

消息传送在两个跃点上进行：第一个跃点从代理上的物理目标的生成方获得消息；第二个跃点从使用方的目标获得消息。因此，在下面的三个阶段，消息可能丢失：在至代理的跃点上，当代理发生故障时在代理内存中以及在代理至使用方的跃点上。可靠传送可保证传送过程在上述任一阶段都不会失败。由于当代理失败时非持久性消息总是会丢失，因此可靠传送仅适用于持久性消息。

使用了两种机制来确保可靠传送：

- 客户端可使用 acknowledgement（确认）或 transaction（事务）来确保成功地生成和使用消息。
- 代理可以在持久存储中存储消息，这样，如果代理在消息被使用前失败，它可以检索存储的消息副本并重试操作。

以下各节将介绍这两个方面的可靠性保证措施。

确认

确认是指客户端与消息服务之间为确保可靠消息传送而发送的消息。对于生成方和使用方，确认的用途是不同的。

生成消息时，代理确认它已收到消息，将该消息放入它的目标中并永久存储它。生成方的 `send()` 方法会被阻止，直至它收到此确认为止。这些确认对于持久性消息要发送到的客户端是透明的。

使用消息时，客户端确认已收到从某个目标传送来的消息并已使用它，然后代理从该目标中删除该消息。 JMS 规定了不同的确认模式，它们分别代表不同的可靠度。

- 在 `AUTO_ACKNOWLEDGE` 模式下，会话自动确认客户端使用的每条消息。会话线程会被阻止，以等待代理确认它已处理了每个已使用消息的客户端确认。
- 在 `CLIENT_ACKNOWLEDGE` 模式下，在一条或多条消息被使用后，客户端通过调用消息对象的 `acknowledge()` 方法来显式确认。这样该会话就确认了自上次调用该方法后使用的所有消息。会话线程会被阻止，以等待代理确认它已处理了客户端确认。

  Message Queue 提供使客户端可仅仅确认收到一条消息的方法，从而扩展了该模式。
- 在 `DUPS_OK_ACKNOWLEDGE` 模式下，会话在使用了十条消息后进行确认。会话线程不会因等待代理确认而被阻止，因为在该模式下代理确认不是必需的。虽然该模式可保证不会丢失消息，但并不能保证不会收到重复的消息，因此名称为：`DUPS_OK`。

对于更关心性能而不是可靠性的客户端， Message Queue 服务通过提供 NO_ACKNOWLEDGE 模式来扩展 JMS API。在该模式下，代理不跟踪客户端确认，所以不保证使用方客户端已成功处理了消息。对于发送至非长期订户的非持久性消息，选择该模式可提高性能。

事务

事务是将一条或多条消息的生成和 / 或使用组合到一个工作单位的方法。上述客户端和代理确认过程同样适用于事务。在这种情况下，客户端运行时和代理确认默认在事务级别进行。当事务提交时，将自动发送代理确认。

可以将会话配置为**事务**，并且 JMS API 提供了启动、提交和回滚事务的方法。

在事务中生成或使用消息时，消息服务跟踪各个发送和接收过程，并只有在 JMS 客户端发出提交事务的调用时才完成这些操作。如果事务中特定的发送或接收操作失败，将引发异常。客户端代码可以通过忽略异常、重试操作或回滚整个事务来处理异常。事务提交时，所有操作都已完成。事务回滚时，所有成功的操作都取消。

事务的作用范围始终为一个会话。也就是说，可以将在单个会话的上下文中执行的一个或多个生成方或使用方操作组成一个事务。由于事务只能跨越单个会话，因此不存在既包括消息生成又包括消息使用的端对端事务。

JMS 规范还支持**分布式**事务。也就是说，消息的生成和使用可以是大型分布式事务的一部分，该事务中包括涉及其他资源管理器 （如数据库系统）的操作。必须有事务管理器（例如， Java Systems Application Server 提供的事务管理器）才能支持分布式事务。

在分布式事务中，分布式事务管理器使用在 Java 事务 API (Java Transaction API, JTA) XA 资源 API 规范中定义的两阶段提交协议，来跟踪和管理由多个资源管理器（如消息服务和数据库管理器）执行的操作。在 Java 中，资源管理器与分布式事务管理器之间的交互在 JTA 规范中描述。

支持分布式事务是指消息传送客户端可通过 JTA 定义的 XAResource 接口参与分布式事务。此接口定义了实现两阶段提交的许多方法。当客户端进行 API 调用时，JMS 消息服务只与分布式事务管理器 （由 Java 事务服务 (Java Transaction Service, JTS) 提供）协作来跟踪分布式事务中的各种发送和接收操作、跟踪事务状态并完成消息传送操作。

与本地事务一样，客户端可以通过忽略异常、重试操作或回滚整个分布式事务来处理异常。

持久性存储器

另一方面的可靠性就是确保在将持久性消息传送至使用方之前，代理不会将它们丢失。这意味着，当消息到达物理目标时，代理必须将消息放入持久性 data store（数据存储）中。如果代理由于某种原因发生故障，它可以在以后恢复此消息并将此消息传送至相应的使用方。

此外，代理必须永久存储长期订阅。否则，当代理发生故障时，就无法向长期订户传送消息；当有消息到达主题目标后，长期订户就会变为活动状态。

要保证成功传送消息，消息传送应用程序必须将消息指定为持久性消息，并将它们传送给具有长期订阅的主题目标或传送给队列目标。

第 3 章“Message Queue 服务”介绍了 Message Queue 服务提供的默认消息存储，以及管理员如何设置和配置替代的存储。

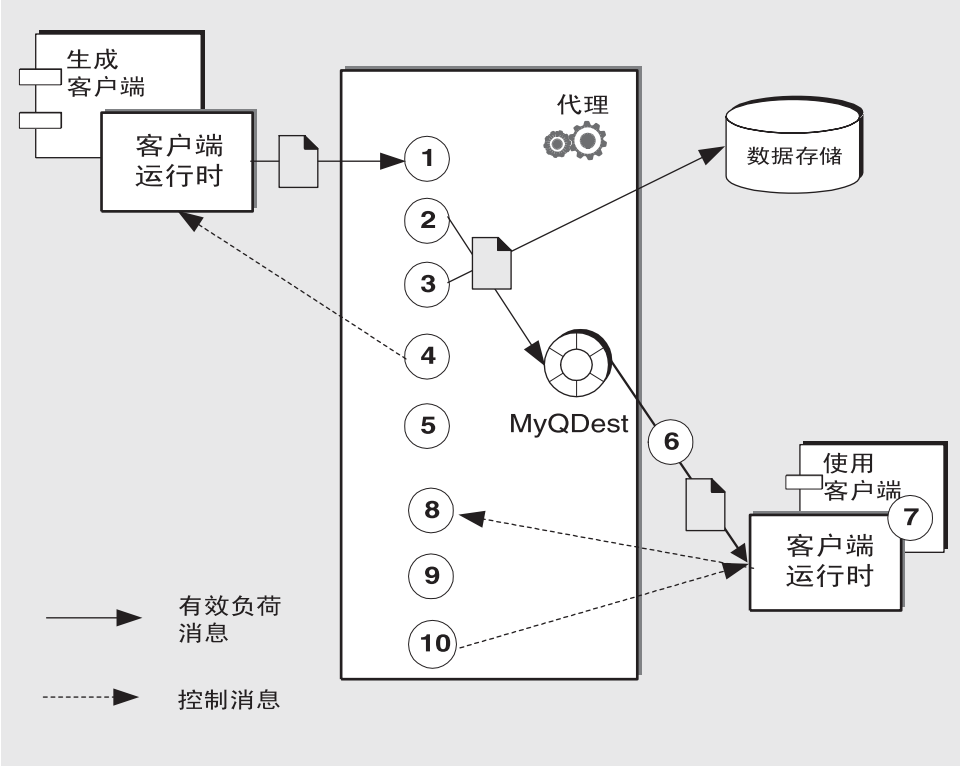
消息在系统中的传送路线

作为对上述内容的总结，本节介绍如何使用 Message Queue 服务从生成方向使用方传送消息。为了描绘一个完整的画面，我们需要补充另外一个细节：在传送过程中，系统处理的消息分为以下两类：

- **有效负荷消息**，即生成方向使用方发送的消息。
- **控制消息**，即在代理与客户端运行时之间传递的专用消息，用于确保已成功传送有效负荷消息并控制连接中的消息流。

图 2-7 说明了消息传送过程。

图 2-7 消息传送步骤



以持久、可靠的方式传送消息的步骤如下：

消息生成

1. 客户端运行时通过连接将消息从消息生成方传送到代理。

消息处理和路由

2. 代理从连接中读取消息并将此消息放入相应的目标中。
3. 代理将 （持久性）消息放入数据存储中。
4. 代理向消息生成方的客户端运行时确认已收到消息。
5. 代理确定消息的路由。
6. 代理将消息从目标写入适当的连接，并使用使用方的唯一标识符标记该消息。

消息使用

7. 消息使用方的客户端运行时将消息从连接传送到消息使用方。
8. 消息使用方的客户端运行时向代理确认消息已使用。

消息生命周期结束

9. 代理处理客户端确认，并在收到所有确认后删除 （持久性）消息。
10. 代理向使用方的客户端运行时确认，告知客户端确认已得到处理。

如果管理员删除目标中的消息，或者管理员删除或重新定义长期订阅，导致主题目标中的消息未被传送即被删除，则代理可以在消息被使用前将它丢弃。在其他情况下，您可能希望代理将消息存储在称为**停用消息队列**的特殊目标中，而不是将它们丢弃。在以下情况下，消息会被放入停用消息队列中：消息过期时、消息因内存限制而被删除时以及因客户端引发异常而导致传送失败时。通过将消息存储在停用消息队列中，您可以解决系统问题并在某些情况下恢复消息。

使用 SOAP 消息

使用 SOAP （请参见第 33 页上的 “对 Java 客户端的 SOAP 支持”）可以在分布式环境中的两个对等方之间交换结构化数据（由 XML 方案指定）。SOAP 的 Sun 实现当前不支持可靠 SOAP 消息传送，也不支持发布 SOAP 消息。不过，您可以使用 Message Queue 服务获得可靠的 SOAP 消息传送，并在需要时发布 SOAP 消息。Message Queue 服务并不直接传送 SOAP 消息，但它允许您将 SOAP 消息包装为 JMS 消息并像生成和使用正常的 JMS 消息一样生成和使用这些消息，然后从 JMS 消息中提取 SOAP 消息。

Message Queue 通过两个软件包来提供 SOAP 支持：javax.xml.messaging 和 com.sun.messaging.xml。您可以使用在这些库中实现的类来接收 SOAP 消息，将 SOAP 消息包装为 JMS 消息，然后从 JMS 消息中提取 SOAP 消息。J2EE 平台提供软件包 java.xml.soap，您可以使用它来组合和分解 SOAP 消息。

要实现可靠的 SOAP 消息传送，需要执行以下操作：

1. 使用 java.xml.soap 软件包中定义的对象构造 SOAP 消息，或使用 javax.xml.messaging 软件包中定义的 servlet 接收 SOAP 消息，也可以使用 JAX-RPC 等 Web 服务来接收 SOAP 消息。
2. 使用 MessageTransformer 实用程序将 SOAP 消息转换为 JMS 消息。
3. 将 JMS 消息发送到所需目标。
4. 以异步方式或同步方式使用 JMS 消息。
5. 在 JMS 消息已使用后，使用 MessageTransformer 实用程序将它转换为 SOAP 消息。
6. 使用 SAAJ API （在 java.xml.soap 软件包中定义）分解 SOAP 消息。

有关 SOAP 消息及其处理的详细信息，请参见 Message Queue Developer's Guide for Java Clients。

Java 客户端与 C 客户端

Message Queue 向消息传送服务提供 C API，使传统 C 应用程序和 C++ 应用程序可参与基于 JMS 的消息传送。

JMS 编程模型是设计 Message Queue C 客户端的基础。Message Queue Developer's Guide for C Clients 说明了 C 数据类型及函数如何实现该模型。

与 Java 接口一样， C 接口也支持以下功能：

- 发布 / 订阅及点对点连接
- 同步及异步接收
- CLIENT、 AUTO 和 DUPS_OK 确认模式
- 本地事务
- 会话恢复
- 临时主题和队列

- 消息选择器

不过，必须了解 Java Message Service 规范是仅适用于 **Java** 客户端的标准；所以，C Message Queue API 特定于 Message Queue 提供者，不能用于其他 JMS 提供者。包含 C 客户端的消息传送应用程序不能由另一 JMS 提供者处理。

C 接口不支持以下功能：

- 使用受管理对象
- 映射、流或对象消息类型
- 基于使用方的流控制
- 队列浏览器
- JMS 应用服务器工具 （ConnectionConsumer、分布式事务）
- 接收或发送 SOAP 消息
- 接收或发送压缩的 JMS 消息
- 自动重新连接或故障转移，使客户端运行时可在连接失败的情况下自动与代理重新连接
- `NO_ACKNOWLEDGE` 模式

第 3 章

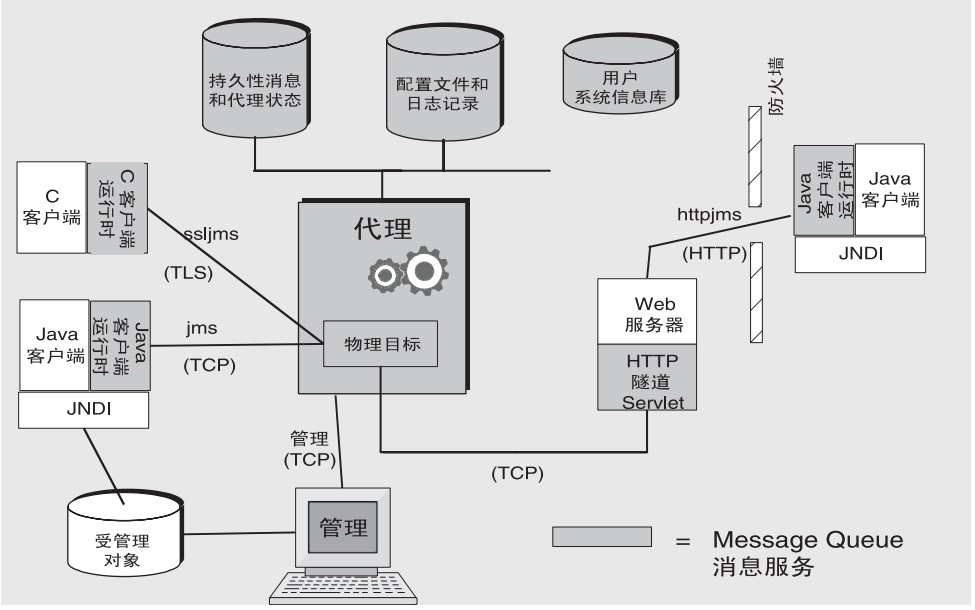
Message Queue 服务

Message Queue 客户端性能取决于客户端设计以及 Message Queue 服务的配置和管理方式。本章更详细地讲述第 1 章中介绍的 Message Queue 服务。同时本章还列出该服务的组件，介绍配置这些组件所用的工具，并概述在不同环境中管理消息服务所需的任务。本章包含以下各节：

组件服务

图 3-1 显示 Message Queue 服务。第 2 章介绍编程模型以及客户端如何使用 Java 和 C API 与客户端运行时进行交互。客户端运行时是客户端应用程序可以访问的消息服务的一部分。本章重点介绍管理员可以访问的消息服务的组件和服务。

图 3-1 Message Queue 服务



可通过设置代理属性来控制 Message Queue 服务。这些属性可分为多个类别，具体取决于受特定属性影响的服务或代理组件。代理服务包括：

- **连接服务**，管理代理与客户端之间的物理连接，提供传入和传出消息的传输。
- **路由服务**，路由和传送 JMS 消息以及消息服务为支持可靠传送而使用的控制消息。
- **持久性服务**，管理持久性存储器的数据写入和数据检索。
- **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**，生成度量和诊断信息并将这些信息写入到指定的输出通道中。

以下各节将介绍上述每个服务，并概述根据您的特定需要自定义该服务时所使用的属性。

代理属性在不同的配置文件中定义，还可以在用来启动代理的命令行上定义。Message Queue 管理指南介绍这些配置文件，并解释优先级顺序。通过该顺序，可以确定是否可以用一个文件中的属性值覆盖另一个文件中设置的值。使用启动命令设置的属性可以覆盖其他所有设置。

连接服务

使用与连接有关的属性，可以配置和管理代理与客户端之间的物理连接。可供 Message Queue 客户端使用的连接服务将在第 31 页上的 “连接到代理”中进行介绍，该节介绍了可用的连接服务：它们的名称、类型和底层协议。连接服务是多线程的，并可以通过专用端口使用。专用端口可以由代理的端口映射器动态分配，也可以由管理员静态分配。默认情况下，当启动代理时，会启动并运行 `jms` 和 `admin` 服务。

因为每个连接都存在两方，所以连接配置会发生在每一方并需要协调：

- 客户端必须配置连接工厂对象的某些属性才能执行以下操作：请求非默认的连接服务、主机和端口；指定在需要重新连接到另一个代理时可连接到的代理的列表；配置重新连接行为。客户端还可以指定测试失败连接的 ping 间隔。
- 而管理员可以使用代理属性来执行以下操作：激活非默认的连接服务；根据需要分配静态端口；配置线程处理；在使用多个网卡时指定要连接到的主机。管理员还可以指定测试客户端是否可以访问的 ping 间隔；这对于管理资源非常有用。

客户端可以通过防火墙连接到 Message Queue 服务。这可以通过以下方法来完成：让防火墙管理员打开特定的端口，然后连接到该（静态）端口，或者如第 92 页上的 “HTTP 连接”中概述的那样使用 HTTP 或 HTTPS 服务。

每个连接服务还支持特定的验证和授权功能。有关详细信息，请参见第 66 页上的 “安全服务”。

端口映射器

驻留在代理主端口 7676 上的通用**端口映射器**会动态地为连接服务分配端口。当 Message Queue 客户端运行时建立与代理的连接时，会首先与该端口映射器联系，为它已选定的连接服务请求端口号。

可以通过在配置 `jms`、`ssljms`、`admin` 和 `ssladmin` 连接服务时为这些服务分配**静态**端口号来覆盖端口映射器的分配。但是，通常仅在特殊情况下 （如，通过防火墙建立连接时）才使用静态端口，一般不建议使用静态端口。

线程池管理

每个连接服务都是多线程的，可以支持多个连接。这些连接所需的线程由线程池中的代理来维护。它们的分配方式取决于您指定的最小和最大线程值以及您选择的线程处理模型。

可以设置用来指定最小和最大线程数的代理属性。因为线程是连接所必需的，所以它们将添加到支持该连接的服务的线程池中。最小值指定可供分配的线程的数量。当可用线程超过这个最小阈值时，系统将在线程变为空闲状态时关闭这些线程，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，线程数量可能会增加，直到达到线程池的最大数量。此后，新连接将被拒绝，直到某个线程变得可用。

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，一个用于传出消息。这限制了可能的连接数量，但提高了性能。
- 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，该模型增加了可能的连接数量，但同时也增加了线程管理开销，从而会影响性能。

目标和路由服务

客户端连接到代理之后，就可以开始路由和传送消息。在该阶段，代理负责创建和管理不同类型的物理目标、确保消息顺利流动以及高效使用资源。代理使用与路由和目标有关的代理属性，按照符合应用程序需要的方式来管理这些任务。

我们已经介绍了代理中的**物理目标**的概念，物理目标是在将消息传送到消息使用方之前，用来存储该消息的内存位置。物理目标分为四种：

- **管理员创建的目标**，由管理员使用 GUI (imqadmin) 或 imqcmd 实用程序创建。这些目标对应于以编程方式创建的逻辑目标，或者对应于由管理员创建、由客户端查找的目标受管理对象。可以使用 imqcmd 实用程序来为管理员创建的每个目标设置或更新属性。
- **自动创建的目标**，当消息使用方或生成方尝试访问不存在的目标时，由代理自动创建的目标。这些目标通常在开发过程中使用。可以设置用来禁止创建此类目标的代理属性。可以设置用来配置特定代理中所有自动创建的目标的代理属性。

  当自动创建的目标不再使用时，也就是当它不再与任何消息使用方客户端连接且不再包含任何消息时，代理会自动将它销毁。如果代理重新启动，则只有当它包含持久性消息时，才会重新创建这种目标。
- **临时目标**，由需要一个用来接收消息回复的目标的客户端以编程方式显式创建和销毁。顾名思义，这些目标是为连接临时创建的，并由代理维护，而且仅在连接期间存在。

  临时目标不永久存储，并且绝不会在代理重新启动时重新创建，但是它们对于管理工具是可见的。
- **停用消息队列**，是一种在代理启动时自动创建的专用目标，用于存储停用消息，以便于进行诊断。可以使用 imqcmd 实用程序为停用消息队列设置属性。

管理目标

可以使用 imqcmd 实用程序来管理目标。要管理目标，需要完成下面的一个或多个任务：

- 创建、暂停、继续或销毁目标。
- 列出代理中的所有目标。
- 显示有关目标的状态和属性的信息。
- 显示目标的度量信息。
- 压缩用于保持目标的消息的硬盘空间。
- 更新物理目标的属性。

管理任务因所管理的目标的种类 （管理员创建、自动创建、临时或停用消息队列）而异。例如，临时目标不需要显式销毁；自动创建的属性可通过使用某些代理配置属性来配置，这些属性应用于该代理中自动创建的所有目标。

配置物理目标

为获得最佳性能，可以在创建或更新物理目标时设置属性。下面是可以设置的属性：

- 目标的类型和名称。
- 目标的单个和合计限制（消息的最大数量、最大总字节数、每条消息的最大字节数以及生成方的最大数量）。
- 在超过单个或合计限制时，代理应当执行的操作。
- 要在一批中传送的消息的最大数量。
- 是否应将目标的停用消息发送到停用消息队列。
- 对于群集代理，是否应当将目标复制到群集内的其他代理。

对于队列目标，还可以配置备份使用方的最大数量，并为群集代理指定是否优先传送到本地队列。

还可以配置停用消息队列的限制和行为。但是，请注意，停用消息队列的默认属性不同于标准队列的属性。

管理内存

因为目标可能使用大量的资源 （取决于它们处理的消息的数量和大小以及注册的使用方数量和长期性），所以需要对它们进行严格的管理，以保证消息传送服务具有良好的性能和可靠性。

可以设置一些属性，以防止向代理传入过多的消息并防止代理内存不足。代理使用以下三级内存保护，使消息服务在资源不足时仍可正常运行：目标限制、系统范围限制以及系统内存阈值。理想情况下，如果目标限制和系统范围限制设置得当，则应当不会达到紧急系统内存阈值。

目标消息限制

可以设置用来管理每个目标的内存和消息流的目标属性。例如，可指定目标允许的生成方的最大数量、目标允许的消息的最大数量或最大大小以及任何一条消息的最大大小。

还可以指定在达到上述任何限制时代理的响应方式：降低生成方的速度、丢弃最旧的消息、丢弃优先级最低的消息或者拒绝最新的消息。

系统范围消息限制

还可以使用属性来设置应用于代理中所有目标的限制：可以指定消息总数和所有消息占用的内存。如果达到了任何系统范围消息限制，代理将拒绝新消息。

系统内存阈值

最后，可以使用属性来设置阈值。当达到阈值时，代理会采取越来越严格的措施来防止内存过载。采取的操作取决于内存资源的状态：green（可用内存充足）、yellow（代理内存不足）、orange（代理内存严重不足）、red（代理无可用内存）。随着代理的内存状态从 green 变为 red，代理所采取的措施也会越来越严格：

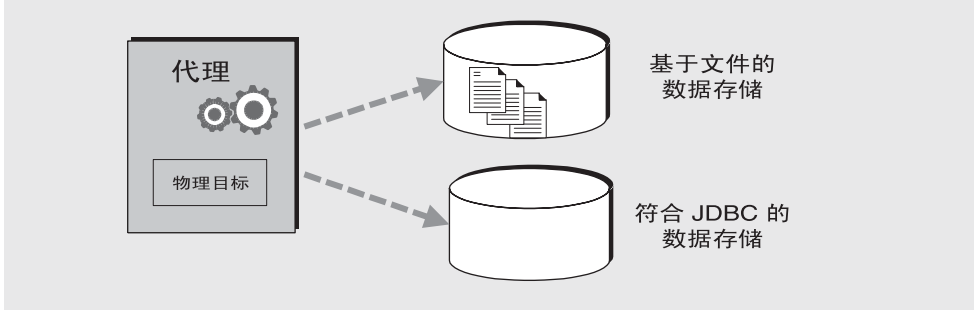
- 它丢弃持久性消息在数据存储中的内存副本。
- 它限制非持久性消息的生成方，最终会停止进入代理的消息流。持久性消息流自动受以下要求的限制：每条消息必须由代理确认。

持久性服务

对于发生故障后待恢复的代理，需要重新创建它的消息传送操作的状态。为此，它必须将状态信息保存到数据存储。代理重新启动时，会使用所保存的数据来重新创建目标和长期订阅、恢复持久性消息、回滚打开的事务以及为未传送的消息重新创建路由表。然后代理才能恢复消息传送。

Message Queue 服务既支持基于文件的持久性模块，又支持符合 JDBC 的持久性模块（请参见图 3-2）。默认情况下它使用基于文件的持久性。

图 3-2 持久性支持



基于文件的持久性

基于文件的持久性是一种使用单个的文件来存储持久性数据的机制。如果您使用基于文件的持久性，则可以设置用来执行以下操作的代理属性：

- 压缩数据存储以减少添加和删除消息所产生的碎片。
- 在**每次**写入时与物理存储器同步内存中的状态。这有助于避免因系统崩溃而导致的数据丢失。
- 管理数据存储文件的消息分配以及进行文件管理和存储所需的资源。

通常，基于文件的持久性比基于 JDBC 的持久性速度快；但是，某些用户更希望获得符合 JDBC 的存储所提供的冗余和管理控制。

基于 JDBC 的持久性

基于 JDBC 的持久性使用 Java 数据库连接 (Java Database Connectivity, JDBC™) 接口来将代理连接到符合 JDBC 的数据存储。为了让代理通过 JDBC 驱动程序来访问数据存储，必须执行以下操作：

- 设置与 JDBC 有关的代理配置属性。使用这些属性可以指定使用的 JDBC 驱动程序、将代理作为 JDBC 用户来验证、创建需要的表格等。
- 使用 `imqdbmgr` 实用程序可以创建具有正确模式的数据存储。

Message Queue 管理指南中详细说明了完成这些任务及相关配置属性的完整步骤。

安全服务

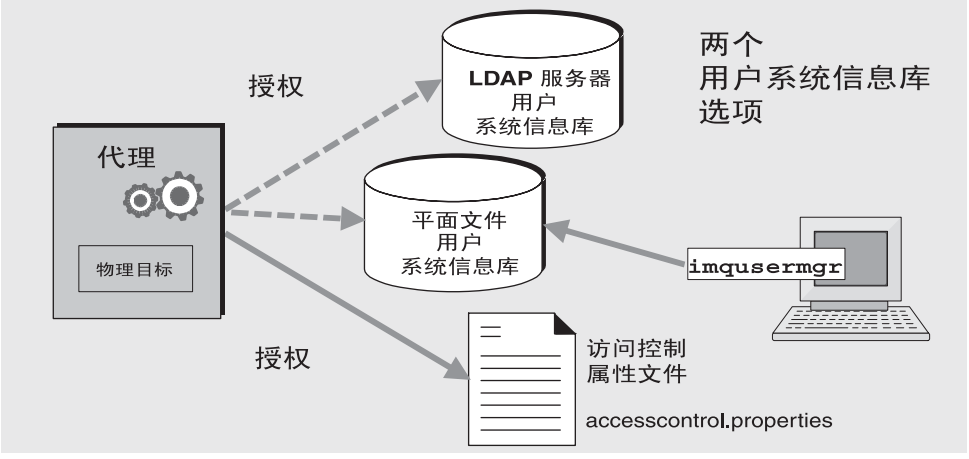
Message Queue 服务对每个代理实例均支持验证和授权 （访问控制）功能，同时还支持加密功能：

- **验证**功能确保只有通过验证的用户才能与代理建立连接。
- **授权**功能指定有权访问资源和执行特定操作的用户或组。
- **加密**功能防止消息在通过连接传送时被篡改。

验证和授权功能依赖于包含消息传送系统的用户信息（用户名、密码和 group（组）成员资格）的系统信息库。此外，要授予用户或组执行特定操作的权限，代理必须检查**访问控制属性文件**，该文件指定用户或组可以执行的操作。您需要设置某些信息，代理需要这些信息来验证用户并授予执行相应操作的权限。

图 3-3 显示代理为提供验证和授权功能所需的组件。

图 3-3 安全性管理器支持



如图 3-3 所示，您可以将用户数据存储在随 Message Queue 服务提供的平面文件用户系统信息库中，也可以将它插入已有的 LDAP 系统信息库中。设置一个用来指示您所进行的选择的代理属性。

- 如果您选择平面文件系统信息库，则必须使用 `imqusermgr` 实用程序来管理系统信息库。此选项是内置选项且易于使用。
- 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。还必须在代理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。

如果可扩展性非常重要，或者需要在不同的代理之间共享系统信息库，则最好选择 LDAP 选项。如果您使用的是代理群集，就可能会遇到这种情况。

验证和授权

当客户端请求连接时，必须提供用户名和密码。代理会将指定的名称和密码与存储在用户系统信息库中的名称和密码进行比较。密码在从客户端传送到代理的过程中，将使用 Base 64 编码或消息摘要 (MD5) 散列进行编码。MD5 适用于平面文件系统信息库； Base 64 是 LDAP 系统信息库所必需的。如果您使用的是 LDAP，则可能希望使用安全的 TLS 协议。通过设置一些代理属性，可以分别配置每个连接服务使用的编码类型，也可以设置适用于整个代理的编码方式。

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访问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：创建任意给定目标或所有目标的使用方、生成方或队列浏览器
- 自动创建目标

可以设置用来指定以下信息的代理属性：

- 是否启用了访问控制
- 访问控制文件的名称
- 密码的编码方式
- 系统等待客户端响应来自代理的验证请求的时间。
- 建立安全连接所需的信息

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 的代理与启用 SSL 的客户端之间建立加密连接来提供连接级别的安全性。

可以设置一些代理属性，以指定要使用的 SSL 密钥存储的安全属性以及密码文件的名称和位置。

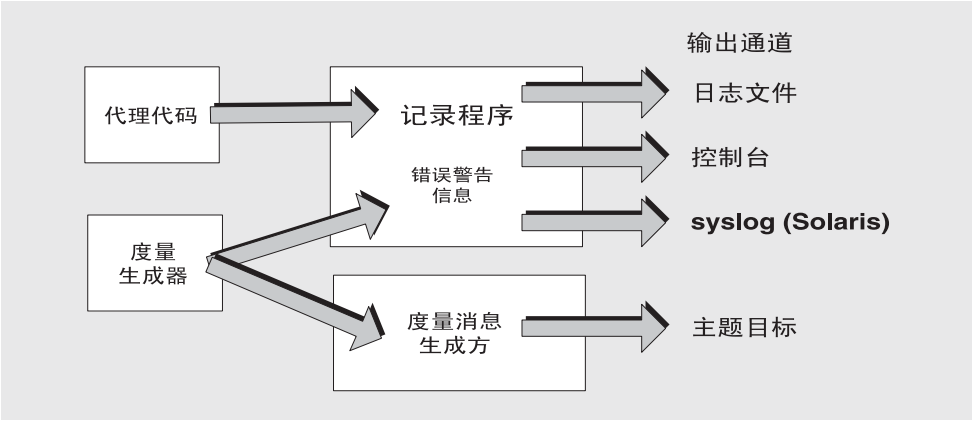
监视服务

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 3-4 中显示的是通用方案。

图 3-4 监视服务支持



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。

通过设置代理属性，可以启用和禁用度量数据的生成，以及指定度量报告的生成频率。

记录程序

出错时， Message Queue 记录程序提取由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（Solaris™ 平台上的）`syslog` 守护程序进程。

您可以设置一些代理属性，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类型。对于日志文件，还可以指定何时关闭日志文件并将输出转移到新文件。在日志文件达到指定的大小或生存期后，将保存该文件并创建一个新的日志文件。

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详细信息，请参见 Message Queue 管理指南。

度量消息生成方 (Enterprise Edition)

图 3-4 所示的度量消息生成方按一定的时间间隔接收来自度量生成器的信息，并将这些信息写入消息，然后根据消息中包含的度量信息的类型，将消息发送至多个度量主题目标之一。

订阅这些度量主题目标的 Message Queue 客户端可以使用消息，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监视工具来支持消息传送应用程序。有关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。有关如何配置度量消息生成的信息，请参见 Message Queue 管理指南。

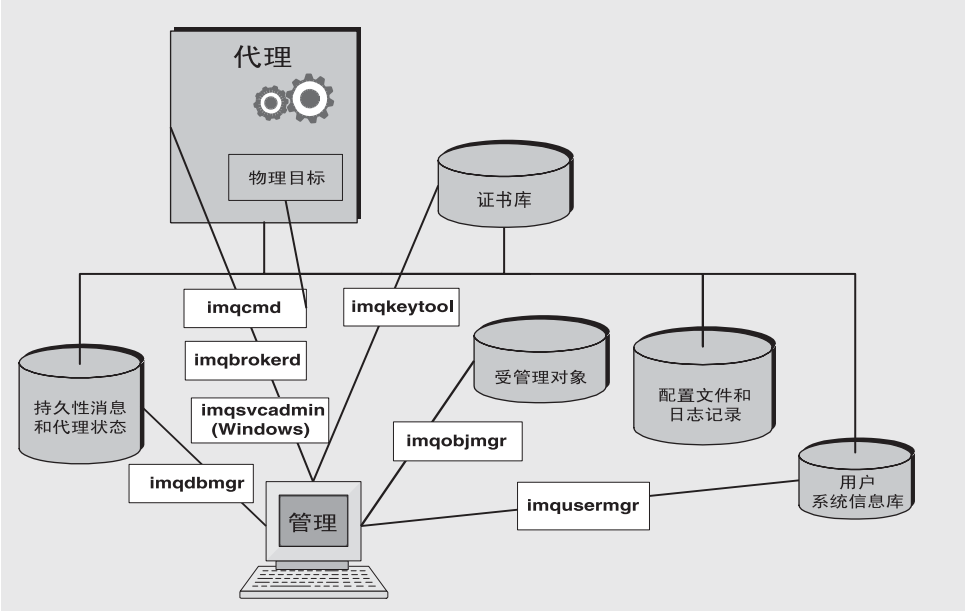
管理工具和任务

本节介绍用来配置 Message Queue 服务的工具以及为支持开发或生产环境而需要完成的任务。

管理工具

图 3-5 显示不包括客户端连接的消息服务的视图，并主要呈现代理组件和用来管理这些组件的工具。

图 3-5 管理工具



可以使用以下命令行工具来配置和管理 Message Queue 服务。

- 可以使用 `imqbrokerd` 实用程序来启动代理。可以使用 `imqbrokerd` 命令的选项来指定是否应该连接群集中的代理以及指定其他启动配置信息。
- 启动代理后，可以使用 `imqcmd` 实用程序来创建、更新和删除物理目标，控制代理和它的连接服务以及管理代理的资源。
- 可以使用 `imqobjmgr` 实用程序来添加、列出、更新和删除 JNDI 对象存储中的受管理对象。
- 可以使用 `imqusermgr` 实用程序来填充基于文件的用户系统信息库，用于用户验证和授权。
- 可以使用 `imqdbmgr` 实用程序来创建和管理用于持久性存储的符合 JDBC 的数据库。（不需要从外部来管理内置的文件存储。）
- 可以使用 `imqkeytool` 实用程序来生成用于 SSL 验证的自签名证书。
- 使用 `imqsvcadmin` 实用程序，可以将代理作为 Windows 服务来安装、查询和删除。

基于 GUI 的管理控制台结合了 `imqcmd` 和 `imqobjmgr` 实用程序的某些功能。可以使用管理控制台来执行以下操作：

- 连接到代理并对它进行管理。
- 创建和管理物理目标。
- 连接到对象存储、向存储添加对象并管理这些对象。

支持开发环境

开发客户端组件时，最好尽量减少管理工作。 Message Queue 产品的设计有助于实现此目标并且可以即装即用。代理只需启动即可使用。以下做法有助于您将重点放在开发上：

- 使用数据存储（内置文件持久性）、用户系统信息库（基于文件）和访问控制属性文件的默认实现。这些对于开发测试已经足够了。默认的用户系统信息库是使用默认条目创建的，这些条目使您在安装代理之后可以立即使用它。可以使用默认用户名 (guest) 和密码 (guest) 来验证客户端。
- 通过创建一个用于该目的的目录来使用简单的文件系统对象存储，并在其中存储受管理对象。如果您不希望创建存储，则还可以直接在代码中实例化受管理对象。
- 使用自动创建的物理目标，而不是在代理中显式创建它们。有关信息，请参见相应的开发者指南。

支持生产环境

在生产环境中，消息服务管理在应用程序性能以及满足企业对可扩展性、可用性和安全性的要求等方面扮演着重要角色。在生产环境中，管理员还需要执行更多任务。这些任务大致可分为设置操作和维护操作。

设置操作

通常，您必须执行以下设置操作：

- 安全的管理访问

  无论您使用的是基于文件的用户系统信息库还是 LDAP 用户系统信息库，都需要确保管理员在 admin 组中并且拥有安全的密码。如有必要，请为管理员创建一个与代理的安全连接。

- 安全的客户端访问

  无论您使用的是基于文件的用户系统信息库还是 LDAP 用户系统信息库，都需要使用可以访问消息服务的用户名来填充用户系统信息库，并编辑访问控制属性文件，以赋予用户相应的权限。如有必要，请设置基于 SSL 的连接服务。为了防止未经验证的连接，请确保更改 "guest" 用户的密码。

- 创建和配置物理目标

  设置目标属性，以确保消息数量和为消息分配的内存量在代理资源支持的范围内。

- 创建和配置受管理对象。

  如果您希望使用 LDAP 对象存储，请配置并设置该存储。创建和配置连接工厂和目标受管理对象。

- 如果需要有状态的水平扩展，请创建一个代理群集。

  创建一个中心配置文件并指定一个主代理。

维护操作

要监视和控制代理资源并调整应用程序性能，必须在部署应用程序之后执行以下操作：

- 支持和管理应用程序客户端
  - 监视和管理目标、长期订阅以及事务。
  - 禁用自动创建功能
  - 监视和管理停用消息队列
- 监视和调整代理
  - 恢复出现故障的代理
  - 监视、调整和重新配置代理。
  - 管理代理内存资源。
  - 根据需要扩展群集。
- 管理受管理对象

  根据需要创建其他受管理对象，并对连接工厂属性进行调整，以提高性能和吞吐量。

扩展 Message Queue 服务

可以通过连接代理并允许它们共享状态信息来水平扩展 Message Queue 服务。这允许任何代理访问远程目标并为更多的客户端提供服务。有关其他信息，请参见第 4 章 “代理群集”。

第 4 章

代理群集

Message Queue Enterprise Edition 支持使用**代理群集**：一组协同工作为客户端提供消息传送服务的代理。使用群集，管理员可以将客户端连接分布在多个代理中，从而使用消息流量来扩展消息传送操作。

本章讨论此类代理群集的体系结构和内部功能，涵盖了以下主题：

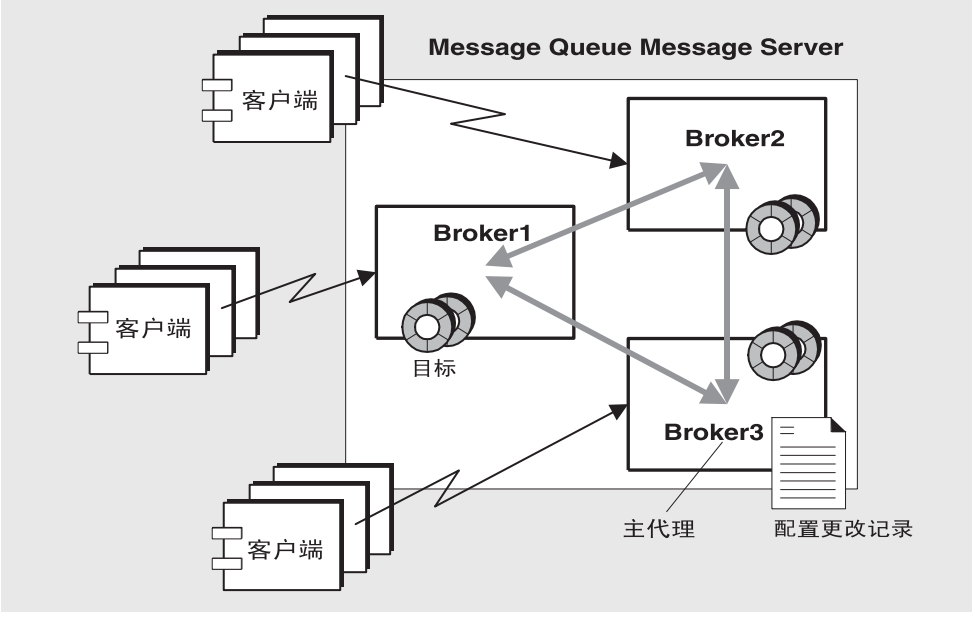
请注意，代理群集提供服务可用性，但不提供数据可用性。如果某个群集内的一个代理失败，则连接到该代理的客户端能够重新连接到该群集内的另一个代理，但是在这些客户端重新连接到备用代理时可能会丢失一些数据。

群集体系结构

图 4-1 显示 Message Queue 代理群集的体系结构。群集内的每个代理直接与其他所有代理相连。每个客户端（消息生成方或使用方）都有一个**本机代理**，借助它，该客户端可直接进行通信以收发消息，就好像该代理是群集内的唯一代理。实际上，本机代理与其他代理协同工作，为连接的所有客户端提供传送服务。

在群集内，服务可用性取决于代理能否共享有关目标和长期订户的信息。如果某个群集代理失败，则此状态信息可能会失去同步。要防止这种可能的情况，可以将群集内的某个代理指定为**主代理**。主代理维护**配置更改记录**，以跟踪群集的持久性实体（目标和长期订阅）的更改。此记录用于将此类更改信息传播到发生更改时处于脱机状态的代理。

图 4-1 群集体系结构



以下各节讨论如何在群集内传送消息以及如何配置和同步代理（即使一个或多个代理处于脱机状态）。

消息传送

在群集配置中，代理共享有关目标和消息使用方的信息。每个代理都知道以下信息：

- 群集内所有物理目标的名称、类型和属性
- 每个消息使用方的名称、位置和兴趣
- 上述信息的更新 （删除、添加或重新配置）

这使每个代理都可以从自己直接相连的消息生成方将消息路由到远程消息使用方。生成方的本机代理与使用方的本机代理具有不同的职责：

- 生成方的本机代理负责保持和路由源自该生成方的消息、进行日志记录、管理事务以及处理来自使用方客户端的确认。
- 使用方的本机代理负责保持有关使用方的信息、将消息转发给使用方、让生成方的代理知道使用方是否仍可用以及消息是否已成功使用。

多个群集代理可以协同工作以最大限度地降低群集内的消息流量；例如，如果某个远程代理对于同一个主题目标有两个完全相同的订阅，则消息将只通过线路发送一次。可以设置一个目标属性以指定向本地使用方的传送优先于向远程使用方的传送，从而进一步减小流量。

如果要求客户端与代理之间的消息传送安全且经过加密，则可以对群集进行配置，以便在代理之间提供安全的消息传送。

目标属性

为一个群集代理中的物理目标设置的属性适用于群集内该目标的所有实例；但是，由这些属性指定的一些限制会应用于整个群集，而其他属性则应用于单独的目标实例。表 4-1 列出了可以为物理目标设置的属性并指定了它们的适用范围。

表 4-1 群集代理中物理目标的属性

属性名称	适用范围
maxNumMsgs	每个代理。因此，通过将生成方分布在一个群集中，可以提高对未使用消息的总数的限制。
maxTotalMsgBytes	每个代理。因此，通过将生成方分布在一个群集中，可以提高为未使用消息保留的总内存的限制。
lmitBehavior	全局。
maxBytesPerMsg	每个代理。
maxNumProducers	每个代理。

表 4-1 群集代理中物理目标的属性 （续）

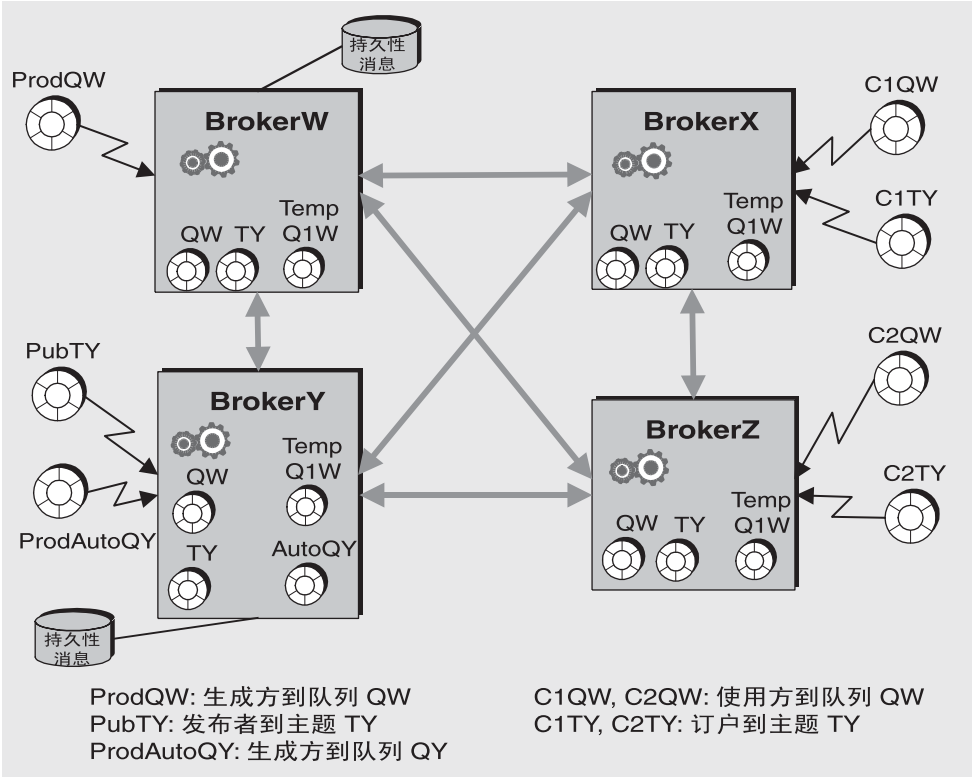
属性名称	适用范围
maxNumActiveConsumers	全局。
maxNumBackupConsumers	全局。
consumerFlowLimit	全局。
localDeliveryPreferred	全局。
isLocalOnly	全局。
useDMQ	每个代理。

群集和目标

无论目标是管理员创建的、自动创建的还是临时的，都会影响该目标在群集内的传播方式以及在连接或代理失败时该目标的处理方式。

图 4-2 显示了四个群集代理。该图显示代理之间的直接（专用）连接，以及客户端与它们所连接到的代理之间的连接。该图说明了以下各节将介绍的多种可能性。

图 4-2 群集示例



使用回复模型生成消息并放入队列

如上图所示：

1. 管理员创建物理目标 `QW`。创建时，队列在整个群集内复制。
2. 生成方 `ProdQW` 向队列 `QW` 发送一条消息，并使用回复模型将回复定向到临时队列 `TempQ1W`。（当应用程序创建临时目标并添加使用方时，系统会创建并复制临时队列。）
3. 本机代理 `BrokerW` 保持发送到 `QW` 的消息并将消息路由到第一个符合该消息选择标准的活动使用方。根据哪个使用方准备好接收该消息，将消息传送到使用方 `C1QW`（位于 `BrokerX` 上）或使用方 `C2QW`（位于 `BrokerY` 上）。接收该消息的使用方会将回复发送到目标 `TempQ1W`。

生成消息并放入自动创建的目标

如上图所示：

1. 生成方 ProdAutoQY 向代理中不存在的目标 AutoQY 发送一条消息。
2. 代理保持该消息并创建目标 AutoQY。

   自动创建的目标不会自动在群集内复制。只有当某个使用方选择从队列 AutoQY 接收消息时，该使用方的本机代理才创建目标 AutoQY 并将这些消息传递到该使用方。当一个使用方创建自动创建的目标时，该目标将在群集内复制。

发布到主题目标

如上图所示：

1. 管理员创建物理主题目标 TY。管理员创建的目标 TY 会在使用之前在代理群集内复制。
2. 发布者 PubTY 向 TY 发送一条消息。
3. 本机代理 BrokerY 保持发布到 TY 的任何消息，并将这些消息路由到所有符合该消息选择标准的主题订户。

在连接或代理失败时处理目标

表 4-2 解释如何在群集内复制和删除不同种类的目标。

表 4-2 在群集内处理目标

目标	传播和删除
管理员创建的	该目标创建出来后，会在群集内传播，而且每个代理都永久存储有关该目标的信息。 当该目标由管理员明确删除时，它会被销毁。 如果有一个主代理，则会在主代理中存储有关创建和删除的记录，以使群集内的其他代理可以同步状态信息。
临时	该目标创建出来后，会在群集内传播。 如果允许与临时目标关联的使用方重新连接，则该目标将永久存储在使用方的本机代理中。否则，该目标将永远也不会存储。 如果该使用方的连接断开，则该目标将从所有代理中删除。 如果该使用方的本机代理崩溃，并且允许该使用方重新连接，则与该使用方关联的临时目标将受到监视。如果使用方客户端在特定的时间段内未重新连接，则会假定该客户端出现故障，该目标将被删除。

表 4-2　在群集内处理目标 （续）

目标	传播和删除
自动创建	创建生成方时，如果目标不存在，则会在生成方的本机代理中创建一个目标。 为不存在的目标创建使用方时，有关该使用方和目标的信息会在群集内传播。 自动创建的目标可以由管理员明确删除，也可以在以下情况下由每个代理自动删除： • 在给定的时间段内没有使用方或消息。 • 当代理重新启动时，该目标没有消息。

群集配置

要在启动时在群集代理之间建立连接，必须为每个代理传送其他所有代理 （包括主代理，如果有）的主机名和端口号。这些信息是通过一组**群集配置属性**来指定的，对于群集内的所有代理，这些属性应该是相同的。虽然可以为每个代理分别指定配置属性，但这种方法容易出错，并且容易导致群集配置出现不一致的情况。因此，建议您将所有配置属性都集中放在一个**群集配置文件**中，供每个代理在启动时引用。这样，就可以确保所有代理共享相同的配置信息。

有关群集配置属性的详细信息，请参见 Message Queue 管理指南。

注　虽然群集配置文件原本是用来配置群集的，但也可以方便地使用它来存储群集内的所有代理共有的其他属性。

群集同步

每次更改群集的配置时，有关更改的信息都会自动传播到群集内的所有代理。发生以下事件之一时，会更改群集配置：

- 在群集的某个代理中创建或销毁目标。
- 目标的属性发生更改。
- 消息使用方向它的本机代理进行注册。
- 消息使用方与它的本机代理断开连接（无论是明确断开，还是由于客户端、代理或网络故障而断开）。
- 消息使用方建立了对某个主题的长期订阅。

有关这些更改的信息会立即传播到群集内发生更改时处于联机状态的所有代理。但是，发生更改时，处于脱机状态的代理（例如，已崩溃的代理）不会收到更改通知。为了给脱机代理提供该信息， Message Queue 维护群集的**配置更改记录**，其中记录了已创建或已销毁的所有持久性实体（目标和长期订阅）。当脱机代理恢复为联机状态时（或向群集添加新代理时），会查阅此记录以获取有关目标和长期订户的信息，然后与其他代理交换有关当前活动的消息使用方的信息。

群集内的某个代理被指定为**主代理**，该主代理负责维护配置更改记录。因为在没有主代理的情况下其他代理无法完成初始化，所以应该始终首先启动群集内的主代理。如果主代理处于脱机状态，则将无法在整个群集内传播配置信息，因为其他代理无法访问配置更改记录。在这些情况下，如果尝试创建、重新配置或销毁目标或长期订阅，或者尝试执行某个相关的操作 （例如，重新激活长期订阅），将发生异常。（但是，非管理消息传送仍然可以正常进行。）主代理和配置更改记录的使用是可选的。只有当群集配置发生更改或代理失败之后群集的同步非常重要时，才有必要使用它们。

第 5 章

Message Queue 和 J2EE

Java 2 Platform, Enterprise Edition（J2EE 平台）是用于托管多层和瘦客户端企业应用程序的标准服务器平台的规范。 J2EE 平台的要求之一是，分布式组件之间必须能够通过可靠的异步消息传送进行交互。这种交互是通过使用 JMS 提供者实现的。事实上， Message Queue 是 J2EE 平台的参考 JMS 实现。

本章探讨有关在 J2EE 平台环境中实现 JMS 支持的其他信息，涵盖了以下主题：

- 第 82 页上的 “JMS/J2EE 编程：消息驱动 Bean”
- 第 83 页上的 “J2EE 应用服务器支持”

有关将 Message Queue 用作符合 J2EE 的应用服务器的 JMS 提供者的其他信息，请参见 Message Queue 管理指南。

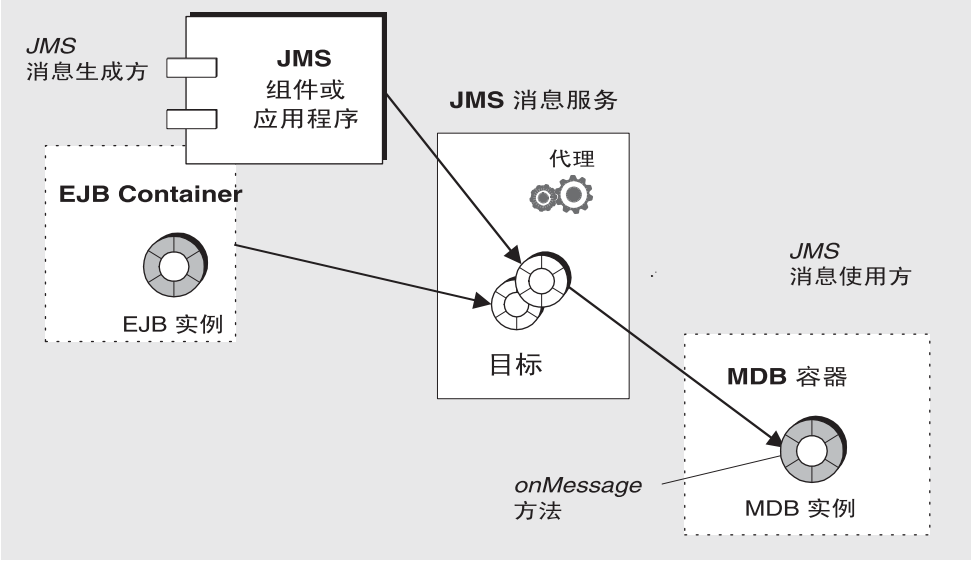
JMS/J2EE 编程：消息驱动 Bean

除了在第 2 章介绍的通用 JMS 客户端编程模型外，还有专用于 J2EE 平台应用程序上下文中的 JMS 客户端。这种专用的客户端称为**消息驱动 Bean**，它是 Enterprise JavaBeans (EJB) 2.0 （和更高版本）规范 (http://java.sun.com/products/ejb/docs.html) 中描述的 EJB 系列组件之一。

消息驱动 Bean 是必需的，因为只能通过标准的 EJB 接口同步调用其他 EJB 组件（会话 Bean 和实体 Bean）。但是，许多企业应用程序都需要异步消息传送。这些应用程序大都要求服务器端组件能够在不占用服务器资源的情况下相互通信。因此需要有这样一种 EJB 组件：不需要紧密耦合到消息生成方即可接收和使用消息。这种能力对于服务器端组件必须响应应用程序事件的任何应用程序都是必需的。在企业应用程序中，这种能力还必须在负载增加时扩展。

消息驱动 Bean (Message-driven Bean, MDB) 是由专用的 EJB 容器 （为所支持的组件提供分布式服务）支持的 EJB 组件。

图 5-1 与 MDB 进行消息传送



- JMS 消息驱动 Bean 是用于实现 JMS `MessageListener` 接口的 EJB。当 MDB 容器接收到消息时，将调用 `onMessage` 方法（由 MDB 开发者编写）。`onMessage()` 方法使用消息的方式与标准 `MessageListener` 对象的 `onMessage()` 方法使用消息的方式一样。不能像对其他 EJB 组件那样远程调用 MDB 的方法，因此，不存在与它们关联的主接口或远程接口。MDB 可使用来自单一目标的消息。如图 5-1 所示，独立 JMS 应用程序、 JMS 组件、 EJB 组件或 Web 组件均可生成消息。
- 专用的 EJB 容器支持 MDB。它将创建 MDB 的实例，并对这些实例进行设置以便异步使用消息。该容器与消息服务（包括验证）建立连接，创建与给定目标关联的会话池，并管理池中会话之间的消息分发。由于该容器控制 MDB 实例的有效期，因此它通过管理 MDB 实例池来容纳传入的消息负载。

  与 MDB 关联的是部署描述符，它指定容器在设置消息使用时所使用的连接工厂和目标的属性。部署描述符还可以包括部署工具配置容器所需的其他信息。每个这样的容器都支持单个 MDB 的实例。

J2EE 应用服务器支持

在 J2EE 体系结构中，EJB 容器由 J2EE 应用服务器托管。应用服务器提供各种容器所需的资源：事务管理器、持久性管理器、命名服务以及（消息传送和 MDB 所需的）JMS 提供者。

在 Sun Java System Application Server 中， JMS 消息传送资源由 Sun Java System Message Queue 提供：

- 对于 Sun Java System Application Server 7.0，Message Queue 消息传送系统作为本地 JMS 提供者集成到应用服务器中。
- 对于 Sun J2EE 1.4 Application Server，Message Queue 作为嵌入式 JMS 资源适配器插入到应用服务器中。

对于 Application Server 的后续发行版本，将利用标准资源适配器部署和配置方法将 Message Queue 插入到应用服务器中。

有关 J2EE 体系结构的信息，请参见位于
`http://java.sun.com/j2ee/download.html#platformspec` 的 J2EE 平台规范。

JMS 资源适配器

资源适配器是一种在符合 J2EE 1.4 规范的应用服务器中插入附加功能的标准方法。J2EE 连接器体系结构 (J2EE Connector Architecture, J2EECA) 1.5 规范定义的标准允许应用服务器以标准方式与外部系统进行交互。外部系统可以包括企业信息系统 (Enterprise Information System, EIS) 和消息传送系统，如 JMS 提供者。 Message Queue 包括 JMS 资源适配器，该适配器允许应用服务器将 Message Queue 用作 JMS 提供者。

将 JMS 资源适配器插入应用服务器后，在应用服务器中部署和运行的 J2EE 组件即可交换 JMS 消息。这些组件所需的 JMS 连接工厂和目标受管理对象可以使用 J2EE 应用服务器管理工具来创建和配置。

但是 J2EECA 规范中不包括其他管理操作（如管理代理和物理目标），只能通过特定于提供者的工具来执行这些操作。

Message Queue 资源适配器集成在 Sun J2EE 1.4 应用服务器中。但它尚未经过其他任何 J2EE 1.4 应用服务器的验证。

Message Queue 资源适配器是单个文件 (`imqjmsra.rar`)，它所在的目录因操作系统而异 （请参见 Message Queue 管理指南）。 `imqjmsra.rar` 文件包含资源适配器部署描述符 (`ra.xml`) 以及应用服务器使用适配器所需的 JAR 文件。

您可按照应用服务器附带的资源适配器部署和配置说明，在任何符合 J2EE 1.4 的应用服务器上使用 Message Queue 资源适配器。随着商业 J2EE 1.4 应用服务器的推出以及 Message Queue 资源适配器通过了这些应用服务器的验证，Message Queue 文档将提供有关相关部署和配置过程的特定信息。

附录 A

可选 JMS 功能的 Message Queue 实现

JMS 规范指出了某些项是可选的：每个 JMS 提供者 （供应商）可以选择是否实现这些项。本附录介绍 Message Queue 产品如何处理 JMS 可选项。

表 A-1 介绍 Message Queue 服务如何处理 JMS 可选项。

表 A-1 可选的 JMS 功能

JMS 规范中的章节	描述和 Message Queue 实现
3.4.3 JMSMessageID	“由于创建消息 ID 比较麻烦，并会使消息体积增大，因此如果提示某些 JMS 提供者，告诉它们应用程序不使用消息 ID，则它们能够优化消息开销。JMS 消息生成方提供了禁用消息 ID 的提示。”
	Message Queue 实现： 产品不会禁用消息 ID 生成（MessageProducer 中的所有 setDisableMessageID() 调用均被忽略）。所有消息都将包含一个有效的 MessageID 值。
3.4.12 覆盖消息头字段	“JMS 未明确指定管理员应如何覆盖这些头字段值。不要求 JMS 提供者支持此管理选项。”
	Message Queue 实现： Message Queue 产品支持通过配置客户端运行时以管理方式覆盖消息头字段的值 （请参见第 46 页上的 “消息头”）。
3.5.9 JMS 定义的属性	“JMS 将 'JMSX' 属性名前缀保留用于 JMS 定义的属性。” “除非另行规定，否则对这些属性的支持是可选的。”
	Message Queue 实现： Message Queue 产品支持由 JMS 1.1 规范定义的 JMSX 属性 （请参见 Message Queue 管理指南）。
3.5.10 特定于提供者的属性	“JMS 将 'JMS_*<vendor_name>*' 属性名前缀保留用于特定于提供者的属性。”
	Message Queue 实现： 特定于提供者的属性的用途是提供支持提供者本地客户端使用 JMS 所需的特殊功能。不应将它们用于 JMS 至 JMS 的消息传送。

表 **A-1** 可选的 JMS 功能 （续）

JMS 规范中的章节	描述和 Message Queue 实现
4.4.8 分布式事务	“JMS 不要求提供者支持分布式事务。” **Message Queue** 实现： 此版本的 Message Queue 产品支持分布式事务 （请参见第 53 页上的 “事务”）。
4.4.9 多个会话	“对于 PTP < 点对点分布模型 >，JMS 不为同一个队列指定并行 `QueueReceivers` 的语义；但 JMS 并不禁止提供者支持此功能。”有关详细信息，请参见 JMS 规范的第 5.8 节。 **Message Queue** 实现： Message Queue 实现支持以多个使用方为目标的队列传送。有关详细信息，请参见第 38 页上的 “点对点消息传送”。

Message Queue 功能

Message Queue 服务完全实现了 JMS 1.1 规范，以实现灵活而可靠的异步消息传送。有关 JMS 符合性相关问题的信息，请参见附录 A“可选 JMS 功能的 Message Queue 实现”。但是， Message Queue 所提供的功能超出了 JMS 要求。使用这些功能，可以集成和监视包含大量分布式组件的系统，这些组件在全天候的关键任务操作中交换数以万计的消息。

本书已经在描述 Message Queue 服务的过程中介绍了这些功能。为方便起见，本附录提供了 Message Queue 功能概述：简要描述每项功能，概述使用该功能需要执行的操作，并提供了一些参考，这些参考指向本书中介绍这些功能的章节以及 Message Queue 文档集中详述这些功能的特定文档。

表 B-1 按字母顺序列出了 Message Queue 的功能，这些功能大致可分为以下类别。

- 集成支持
  - HTTP 连接
  - 安全连接
  - C 客户端支持
  - SOAP 支持
  - J2EE 资源适配器

- 安全性
  - 验证
  - 授权
  - 加密 （请参见安全连接）
- 可扩展性
  - 线程管理
  - 线程管理
  - 线程管理
- 可用性
  - 内存资源管理
  - 客户端的消息流控制
  - 自动重新连接
  - 可靠数据的持久性
  - 连接 ping
- 易管理性
  - 管理工具
  - 基于消息的监视 API
  - 可调整的性能
  - 可配置的物理目标
  - 代理配置
  - 停用消息队列
- 灵活的服务器配置
  - 可配置的持久性
  - LDAP 服务器支持
  - JNDI 服务提供者支持

- 性能
  - 消息压缩
  - 可调整的性能
  - 可配置的物理目标

表 B-1 Message Queue 功能

功能	描述和参考
管理工具	Message Queue 服务包括 GUI 和命令行工具，用于管理目标、事务、长期订阅、受管理对象存储、用户系统信息库、符合 JDBC 的数据存储以及服务器证书。 **参考** 第 69 页上的 “管理工具”。 Message Queue 管理指南中的 “管理任务和工具”。
验证	对试图与代理建立连接的用户进行验证。 Message Queue 服务对照存储在用户系统信息库中的值对用户的名称和密码进行验证，并根据验证结果确定是否允许用户连接到代理。系统信息库可以是随 Message Queue 提供的平面文件系统信息库，也可以是 LDAP 系统信息库（LDAP v2 或 v3 协议）。 **用法** 1. 创建用户系统信息库或使用默认实例。 2. 使用 `imqusermgr` 工具填充系统信息库。 **参考** 第 67 页上的 “验证和授权”。 Message Queue 管理指南中的 “管理安全性”
授权	授予用户执行特定操作的权限。 Message Queue 服务允许您创建访问控制属性文件，以指定用户和用户组可以执行的操作。当客户端试图建立连接、创建生成方、创建使用方或浏览队列时，代理会检查此文件。 **用法** 编辑自动为代理实例创建的访问控制属性文件。 **参考** 第 67 页上的 “验证和授权”。 Message Queue 管理指南中的 “管理安全性”

表 B-1 Message Queue 功能 （续）

功能	描述和参考
自动重新连接	管理员对连接工厂受管理对象设置连接属性，以允许在连接失败时自动重新建立连接。可以重新连接到同一个代理，也可以连接到群集中的另一个代理（如果使用群集）。可以指定尝试重新连接的次数以及重试的时间间隔。对于群集代理，还可以指定循环访问代理列表的频率以及是否按特定顺序循环访问列表。 **参考** 第 61 页上的 “连接服务”。 Message Queue 管理指南中的“管理受管理对象”和“受管理对象的属性参考”。
代理群集	通过将许多代理实例组合到一个代理群集内，管理员可以在这些代理之间平衡客户端连接和消息传送。 **用法** 1. 为群集中的每个代理指定群集配置属性。可以通过使用配置文件或为每个代理设置属性来完成此操作。 2. 如果存在主代理，请启动该代理。 3. 启动群集中的其他代理。 **参考** 第 4 章 “代理群集”。 Message Queue 管理指南中的 “使用群集”。
代理配置	管理员可以通过设置代理属性来调整 Message Queue 服务的性能，其中包括路由服务、持久性服务、安全性、监视以及受管理对象的管理。 **参考** 第 3 章 “Message Queue 服务”。 Message Queue 管理指南中的 “配置代理”和 “代理属性参考”
C 客户端支持	C 客户端可以使用 Message Queue 消息传送服务来发送和接收消息。C API 使传统 C 应用程序和 C++ 应用程序可以参与基于 JMS 的消息传送。 支持 Message Queue C API 的 C 客户端运行时支持大多数标准 JMS 功能，但不支持以下 JMS 功能：使用受管理对象；映射、流或对象消息主体类型；分布式事务；以及队列浏览器。此外，C 客户端运行时也不支持 Message Queue 的大部分企业功能。 **参考** 第 57 页上的 “Java 客户端与 C 客户端”。 Message Queue Developer's Guide for C Clients

表 B-1 Message Queue 功能 （续）

功能	描述和参考
压缩的消息	Java 客户端可以设置消息属性，以便让客户端运行时压缩所发送的消息。使用方运行时先解压缩消息，然后再将其传送给使用方。此外还提供了其他属性，可以使用这些属性确定压缩消息是否确实改善了性能。 **参考** 第 48 页上的 “消息主体”。 Message Queue Developer's Guide for Java Clients 中的 "Message Queue Clients: Design and Features"。
可配置的持久性	管理员可以对代理进行配置，以使用随 Message Queue 提供的基于文件的持久性存储，或者使用符合 JDBC 的数据库，如 Oracle 8i。 **用法** 设置与文件系统持久性存储器或与符合 JDBC 的存储器相关的代理属性。 **参考** 第 64 页上的 “持久性服务”。 Message Queue 管理指南中的 “配置代理”。
可配置的物理目标	创建目标时，管理员可以通过设置物理目标属性来定义某些消息传送行为。可以针对任何目标配置以下行为：未使用消息的最大数量或允许这些消息使用的最大内存（当超出内存限制时，代理应拒绝这些消息）、生成方和使用方的最大数量、消息的最大大小、单批传送的最大消息数、目标是否只能将消息传送给本地使用方以及是否可以将目标上的停用消息移到停用消息队列中。 **参考** 第 62 页上的 “目标和路由服务”。 Message Queue 管理指南中的 “管理物理目标”和 “物理目标属性参考”。
连接 ping	管理员可以设置连接工厂属性，以指定从客户端运行时到代理之间 ping 操作的频率。此功能使得客户端可以尽早检测到失败的连接。 **参考** 第 61 页上的 “连接服务”。 Message Queue 管理指南中的 “连接工厂属性”。
停用消息队列	Message Queue 消息服务通过创建停用消息队列来保存已过期或代理无法处理的消息。可以通过检查队列内容来监视和调整系统性能，或者解决系统性能问题。 **参考** 第 62 页上的 “目标和路由服务”。 Message Queue 管理指南中的 “管理物理目标”。

表 B-1 Message Queue 功能 （续）

功能	描述和参考
HTTP 连接	Java 客户端可以建立与代理之间的 HTTP 连接。 HTTP 传输允许通过防火墙传送消息。Message Queue 使用在 Web 服务器环境中运行的 HTTP 隧道 Servlet 来实现 HTTP 支持。客户端运行时将客户端生成的消息包装为 HTTP 请求，并借助于 HTTP 通过防火墙将其传送到隧道 Servlet。隧道 Servlet 从 HTTP 请求提取 JMS 消息，然后将该消息通过 TCP/IP 传送到代理。 **用法** 1. 在 Web 服务器上部署 HTTP 隧道 Servlet。 2. 配置代理的 `httpjms` 连接服务并启动代理。 3. 配置 HTTP 连接 4. 获取与代理的 HTTP 连接。（仅限 Java 客户端。） **参考** 第 31 页上的 “连接到代理”。 Message Queue 管理指南中的附录 C “启用 HTTP 支持”
交互监视	管理员可以使用 `imqcmd metrics` 命令远程监视代理。监视的数据包括 JVM 度量、代理消息流、连接、连接资源、消息、目标消息流、目标使用方和目标资源的使用情况。 **参考** 第 68 页上的 “监视服务”。 Message Queue 管理指南中的 “监视消息服务器”
J2EE 资源适配器	Message Queue 提供可以插入到符合 J2EE 的应用服务器的资源适配器。应用服务器通过将 Message Queue 用作 JMS 提供者来满足 J2EE 要求，即，在应用服务器中运行的分布式组件能够使用可靠的异步消息进行交互。 **用法** 通过设置适配器属性来配置适配器。 **参考** 第 83 页上的 “J2EE 应用服务器支持”。 Message Queue 管理指南中的 “JMS 资源适配器属性参考”。
JNDI 服务提供者支持	客户端可以使用 JNDI API 查找受管理对象。 管理员可以使用 `imqobjmgr` 实用程序，在可通过 JNDI 访问的对象存储中添加、列出、更新和删除受管理对象。 **参考** 第 69 页上的 “管理工具”。 Message Queue 管理指南中的 “命令参考”

表 B-1 Message Queue 功能 （续）

功能	描述和参考
LDAP 服务器支持	管理员可以将 LDAP 服务器用于受管理对象存储，以及用于存储验证和授权所需的用户信息。默认情况下， Message Queue 为这些数据提供基于文件的存储器。 **用于受管理对象** 1. 使用供应商提供的工具来填充和管理用户系统信息库。 2. 设置与 LDAP 相关的代理属性。 3. 为管理用户设置访问控制。 **参考** Message Queue 管理指南中的 “管理安全性” **用于用户系统信息库** 1. 使用供应商提供的工具设置 LDAP 服务器。 2. 设置与 LDAP 相关的代理属性，以定义初始上下文和存储位置。 3. 设置与保护 LDAP 服务器操作有关的 LDAP 相关代理属性。 **参考** 第 66 页上的 “安全服务”。 Message Queue 管理指南中的 “管理受管理对象”
内存资源管理	管理员可以配置以下行为： 1. 设置目标属性，以指定生成方的最大数量、消息的最大总大小以及任一消息的最大大小。 2. 设置目标属性以控制消息流 3. 设置目标属性以管理每个目标的消息流 4. 设置代理属性以便为该代理的所有目标指定消息限制。 5. 设置代理属性以指定可用系统内存的阈值，当达到该阈值时，代理可以采取逐渐严格的操作来防止内存过载。采取的操作取决于内存资源的状态。 **参考** 第 62 页上的 “目标和路由服务”。 Message Queue 管理指南中的 “配置代理”、“代理属性参考”和 “物理目标属性参考”
消息压缩	开发者可以设置消息头属性，以便让客户端运行时在发送消息之前先进行压缩。客户端运行时先解压缩消息，然后再将其传送给使用方。 **参考** 第 47 页上的 “消息属性”。 Message Queue Developer's Guide for Java Clients 中的 "Message Compression"。

表 B-1 Message Queue 功能 （续）

功能	描述和参考
客户端的消息流控制	管理员或开发者可以通过配置连接来指定各种流限制和度量方案，以便最大限度地减少有效负荷消息与控制消息之间的冲突，进而最大限度地提高消息吞吐量。 **用法** 为连接工厂受管理对象 （管理员）设置流控制属性，或者为连接工厂 （开发者）设置流控制属性。 **参考** 第 45 页上的 “连接工厂和连接”。 Message Queue 管理指南中的“管理受管理对象”和“受管理对象的属性参考”。
基于消息的监视 API	Java 客户端可以使用监视 API 创建自定义的监视应用程序。监视应用程序是从特殊度量主题目标中检索度量消息的使用方。 **用法** 1. 编写度量监视客户端。 2. 设置代理属性以配置代理的度量消息生成方。 3. 在度量主题目标上设置访问控制。 4. 启动监视客户端。 **参考** 第 68 页上的 “监视服务”。 Message Queue Developer's Guide for Java Clients 中的 "Using the Metrics Monitoring API"。 Message Queue 管理指南中的 “监视消息服务器”。
多个使用方的队列传送	客户端可以为一个给定队列注册多个使用方。 管理员可以为该队列指定活动使用方的最大数量以及备份使用方的最大数量。代理将消息分发给注册的使用方，在使用方之间平衡负载以便系统可以进行扩展。 **用法** 设置物理目标属性 `maxNumActiveConsumers` 和 `maxNumBackupConsumers`。 **参考** 第 38 页上的 “点对点消息传送”。 Message Queue 管理指南中的 “物理目标属性”和 “多使用方队列性能”。
可靠数据的持久性	要获得绝对的可靠性，可以将 `imq.persist.file.sync.enabled` 属性设置为 true，以此要求操作系统将数据同步写入持久性存储。这会避免因系统崩溃可能引起的数据丢失，但负面影响是会导致性能下降。请注意，尽管数据未丢失，但它们对于（群集中的）任何其他代理都不可用，因为群集代理当前未共享这些数据。当系统恢复时，代理可以可靠地继续执行操作。 **参考** 第 64 页上的 “持久性服务”。 Message Queue 管理指南中的 “代理属性参考”。

表 B-1 Message Queue 功能 （续）

功能	描述和参考
安全连接	客户端可以在 TCP/IP 和 HTTP 传输中使用安全套接字层 (Secure Socket Layer, SSL) 标准来保护消息传输。这些基于 SSL 的连接服务允许对在客户端与代理之间发送的消息进行加密。 SSL 支持基于自签名服务器证书。Message Queue 提供了一个实用程序，它可以生成专用 / 公共密钥对，并将公共密钥嵌入到自签名证书中。此证书将被传递到任何请求代理连接的客户端，该客户端将使用此证书建立加密连接。 **用法** 1. 生成自签名证书或签名证书。 2. 启用安全的服务。 3. 启动代理 4. 配置客户端安全连接属性并运行客户端。 **参考** 第 31 页上的 “连接到代理”。 第 66 页上的 “安全服务”。 Message Queue 管理指南中的 “管理安全性”。 Message Queue Developer's Guide for Java Clients Message Queue Developer's Guide for C Clients
SOAP 支持	客户端可以接收 SOAP (XML) 消息，将它们包装为 JMS 消息，并使用 Message Queue 像交换 JMS 消息一样来交换这些消息。 客户端可以使用特殊的 Servlet 来接收 SOAP 消息；它们可以使用一个实用程序类将 SOAP 消息包装为 JMS 消息，而使用另一个实用程序类从 JMS 消息中提取 SOAP 消息。客户端可以使用标准的 SAAJ 库来组合和分解 SOAP 消息。 **参考** 第 56 页上的 “使用 SOAP 消息”。 Message Queue Developer's Guide for Java Clients 中的 "Working With SOAP Message"。
线程管理	管理员可以指定分配给任何特定连接服务的最大线程数和最小线程数。管理员还可以确定使用共享线程模型能否提高连接服务的吞吐量，该模型允许非空闲连接使用空闲连接专用的线程。 **用法** 设置与连接服务线程相关的属性。 **参考** 第 61 页上的 “线程池管理”。 Message Queue 管理指南中的 “配置代理”。

表 B-1 Message Queue 功能 （续）

功能	描述和参考
可调整的性能	管理员可以设置代理属性，以便对内存使用情况、线程资源、消息流、连接服务、可靠性参数以及其他影响消息吞吐量和系统性能的因素进行调整。 **参考** 第 68 页上的 “监视服务”。 Message Queue 管理指南中的 “监视消息服务器”和 “分析和调整消息服务”

词汇表

本词汇表提供使用 Message Queue 时可能遇到的术语和概念的相关信息。有关包括 Sun Java System 中使用的所有术语的词汇表，请参见 http://docs.sun.com/doc/819-4631

acknowledgement （确认） 为确保可靠的消息传送而在客户端与代理之间交换的控制消息。确认分为两大类：客户端确认与代理确认。

administered objects （受管理对象） 由管理员创建并供一个或多个 JMS 客户端使用的预配置对象 （如连接工厂或目标），其中封装了特定于提供者的实现详细信息。通过使用受管理对象，可以使 JMS 客户端与提供者无关。受管理对象被置于 JNDI 名称空间中， JMS 客户端可以使用 JNDI 查找来访问它们。

asynchronous messaging （异步消息传送） 一种消息交换模式。在该模式下，发送消息时不要求接收消息的使用方做好接收准备。换言之，消息的发送者不需要等到发送方法返回即可继续其他工作。如果消息使用方正处于忙碌或脱机状态，则系统会先将消息发送出去，然后等该使用方准备好时再接收消息。

authentication （验证） 只允许通过验证的用户与代理建立连接的过程。

authorization （授权） 消息服务确定用户能否访问消息服务资源 （如连接服务或目标）以执行消息服务所支持的特定操作的过程。

broker （代理） 管理消息路由、传送、持久性、安全性和日志记录的 Message Queue 实体，它提供了一个用于监视和调整性能及资源使用的接口。

client （客户端） 与其他使用消息服务的客户端进行交互以交换消息的应用程序 （或软件组件）。客户端可以是生成方客户端或 / 和使用方客户端。

client identifier （客户端标识符） 一个标识符，它将连接及连接的对象与代表客户端的 Message Queue 代理所维护的状态相关联。

client runtime （客户端运行时） 在消息传送客户端与 Message Queue 消息服务之间提供接口的 Message Queue 软件。客户端运行时支持客户端向目标发送消息以及从目标接收消息所需的全部操作。

cluster （群集） 两个或多个互连的代理，它们协同工作以提供可扩展的消息传送服务。

connection （连接） 客户端与代理之间用于传递有效负荷消息和控制消息的通信渠道。

connection factory （连接工厂） 客户端用于创建与代理的连接的受管理对象。它可以是 `ConnectionFactory` 对象、`QueueConnectionFactory` 对象或 `TopicConnectionFactory` 对象。

consumer （使用方） 一个对象 (`MessageConsumer`)，由用于接收从目标发送的消息的会话创建。在点对点传送模型中，使用方为接收者或浏览器 （`QueueReceiver` 或 `QueueBrowser`）；在发布 / 订阅传送模型中，使用方为订户 (`TopicSubscriber`)。

data store （数据存储） 用于永久存储代理所需信息 （长期订阅、有关目标的数据、持久性消息、审计数据）的数据库。

dead message （停用消息） 一种由于非正常处理或显式管理员操作而从系统中删除的消息。消息被视为停用的可能原因有：过期、因超出内存限制而从目标中删除或发送尝试失败。可以选择将停用消息存储在停用消息队列中。

dead message queue （停用消息队列） 一种在代理启动时自动创建的专用目标，用于存储停用消息，以便于进行诊断。

delivery mode （传送模式） 消息传送可靠性的指示符：持久性传送模式，确保消息传送并成功使用一次 （且仅一次）；非持久传送模式，确保消息至多传送一次。

delivery model （传送模型） 传送消息的模型：点对点或发布 / 订阅。 JMS 中有两种独立的编程域，其中每一种域都与一种模型对应，这些域使用特定的客户端运行时对象和特定的目标类型 （队列或主题）。除此之外，还有一个统一的编程域。

destination （目标） Message Queue 代理中的物理目标，生成的消息先传送至此处，然后再路由并传送至使用方。此物理目标由受管理对象标识和封装，客户端使用受管理对象指定生成和 / 或使用消息的目标。

domain （域） JMS 客户端用于对 JMS 消息传送操作进行编程的一组对象。有两种编程域：一种对应于点对点传送模型，一种对应于发布 / 订阅传送模型。

encryption （加密） 一种防止消息在通过连接传送时被篡改的机制。

group （组） Message Queue 客户端用户所属的组，用于授予对连接、目标及特定操作的访问权限。

JMS provider （JMS 提供者） 一种产品，用于实现消息传送系统的 JMS 接口并添加配置和管理该系统所需的管理和控制功能。

message service （消息服务） 一种在分布式组件或应用程序间提供可靠的异步消息交换的中间件服务。它包括一个代理、客户端运行时、该代理执行功能所需的几个数据存储，以及配置和监视该代理及调整性能所需的管理工具。

message （消息） 消息传送客户端使用的异步请求、报告或事件。消息包括头（可以在其中添加其他字段）和主体两部分。消息头指定标准字段和可选属性。消息主体包含要传输的数据。

messaging （消息传送） 企业应用程序使用的异步请求、报告或事件系统，使松散耦合的应用程序可以安全可靠地传送信息。

producer （生成方） 由用于向目标发送消息的会话创建的对象 (MessageProducer)。在点对点传送模型中，生成方为发送者 (`QueueSender`)；在发布 / 订阅传送模型中，生成方为发布者 (`TopicPublisher`)。

queue （队列） 管理员为实现点对点传送模型而创建的对象。队列可以一直接收消息，即使使用该消息的客户端处于非活动状态。队列用作生成方与使用方之间的中转站。

selector （选择器） 一种用于对消息进行排序和路由的消息头属性。消息服务基于消息选择器中的标准对消息进行过滤和路由。

session （会话） 发送和接收消息的单线程环境。可以是队列会话或主题会话。

topic （主题） 管理员为实现发布 / 订阅传送模型而创建的对象。可以将主题看作目录结构中的一个节点，负责收集和分发传送给它的消息。使用主题作为消息传送的中转站，可以将消息发布者与消息订户分开。

transaction （事务） 不可细分的工作单位，要么全部完成，要么全部回滚。

索引

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

W

X

Y

Z