Sun Java™ System

Message Queue 3
技术概述

2005Q1

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

文件号码 819-2223

目录

图目录

表目录

前言

本书（即《Sun Java™ System Message Queue 3 2005Q1 技术概述》）对 Message Queue 消息传送服务的技术、概念、体系结构、功能性进行了介绍。

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。

本前言包含以下章节：

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、功能的人员编写的。

Message Queue 管理员负责设置和管理 Message Queue 消息传送系统，特别是该系统的核心 Message Queue 消息服务器。本书并不要求读者掌握所有消息传送系统的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 服务与其他客户机应用程序交换消息的 Message Queue 客户机应用程序。本书并未假定读者了解有关 Java 消息服务 (JMS) 规范 （通过 Message Queue 服务来实现）的任何知识。

阅读本书之前

阅读本书没有任何前提条件。应先阅读本书，对 Message Queue 的基本概念有一定了解后，再阅读 Message Queue Developer and Administration Guides。

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	说明
第 1 章 “基础概念”	提供 Message Queue 的概念背景；其中描述了企业消息传送系统并介绍了 Java 消息服务概念和术语
第 2 章 “Message Queue 简介”	介绍 Message Queue 服务；其中阐述了其体系结构并介绍其企业强度功能
第 3 章 “可靠消息传送”	介绍 Message Queue 服务如何为消息传送应用程序提供可靠的消息传送
第 4 章 “消息服务器”	阐述代理的内部结构，并对各种代理组件及其功能进行了描述。介绍在开发和生产环境中使用 Message Queue 的各种方法。
第 5 章 “代理群集”	阐述 Message Queue 代理群集的体系结构和内部功能
第 6 章 “Message Queue 和 J2EE”	探讨在 J2EE 平台环境中实现 JMS 支持的其他信息

表 1 本书的内容和结构 （续）

章	说明
附录 A“可选 JMS 功能 Message Queue 实现”	介绍 Message Queue 产品如何处理 JMS 可选项
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

本书所使用的约定

本节介绍本文档中使用的有关约定。

文本约定

表 2 文档约定

格式	说明
AaBbCc123 *（斜体文本）*	斜体文本代表占位符。斜体文本可用相应的子句或值替换。斜体文本还用来表示文档标题。
`AaBbCc123` （等宽文本）	等宽文本表示实例代码、在命令行输入的命令、目录、文件、路径名、错误消息文本、类名称、方法名称 （包括签名中的所有元素） 、软件包名称、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	全部大写的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 （Message Queue、 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，然后释放它，再按 S 键。

目录变量约定

Message Queue 使用三种目录变量；其设置因平台而异。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	说明
IMQ_HOME	Message Queue 文档中通常使用此变量，表示 Message Queue 基目录（根安装目录）： • 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 表示 Solaris 平台上的文件位置。 • 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。 • Windows 平台上， 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）。 • 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。 • 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 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。 • Solaris 平台上， IMQ_VARHOME 默认为 /var/imq 目录。 • Solaris 平台上，对于 Sun Java System Application Server 测试版， IMQ_VARHOME 默认为 IMQ_HOME/var 目录。 • Windows 平台上， IMQ_VARHOME 默认为 IMQ_HOME\var 目录。 • Windows 平台上，对于 Sun Java System Application Server，IMQ_VARHOME 默认为 IMQ_HOME\var 目录。 • 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	说明
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 按下列顺序查找 Java 运行时，但用户可以选择将该值设置为所需的 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 先按下列顺序查找 Java 运行时，但用户可以选择将 IMQ_JAVAHOME 值设置为所需的 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* • Windows 平台上，IMQ_JAVAHOME 默认为 IMQ_HOME\jre，但是用户可以选择将该值设置为所需的 JRE 所在的位置。

在本指南中，显示 IMQ_HOME、IMQ_VARHOME 和 IMQ_JAVAHOME 时，不使用特定平台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	说明
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	包含对新功能、限制、已知错误以及技术说明的介绍。
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时所需的背景和信息。
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 客户机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 客户机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 (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 客户机 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。

示例客户机应用程序

许多示例应用程序提供了样例客户机应用程序代码，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 README 文件。

Java 消息服务 (JMS) 规范

可以在以下位置查找 JMS 规范：

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

规范包含样例客户机代码。

第三方 Web 站点

本文档中所引用的第三方 URL 提供了附加的相关信息。

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，并十分乐意接收到您的意见和建议。

要分享您的意见，请转到 http://docs.sun.com，然后单击“发送意见”。在联机表单中，请提供文档的标题和文件号。文件号码可以在书的标题页或文档顶部找到，是一个 7 位或 9 位的数字。例如，本书的标题是《Sun Java System Message Queue 3 2005Q1 技术概述》，文件号码是 819-2223。

当您提供意见和建议时，可能需要在表单中提供文档英文版本的标题和文件号码。本文档英文版本的文件号码和标题是：819-0069，《Sun Java System Message Queue 3 2005Q1 Technical Overview》。

第 1 章

基础概念

Sun Java™ System Message Queue (Message Queue) 可以提供可靠的**异步消息传送**，能够将企业中的分布式应用程序和组件集成在一起。在不同平台和操作系统上运行的进程可以通过连接到该服务来彼此进行交互。

Message Queue 是一种基于标准的**消息传送**解决方案，它实现了 Java™ 消息服务 (JMS) 开放标准。此外，Message Queue 还提供了大规模企业部署所需的互操作性、安全性、可伸缩性、可用性、易管理性以及其他功能。

本章介绍 Message Queue 的基础概念。涵盖下列主题：

如果您已熟知 JMS 概念和术语，可以跳过本章，直接阅读第 2 章 “Message Queue 简介”。

企业消息传送系统

企业消息传送系统使独立的分布式应用程序或应用程序组件可以通过**消息**进行交互。这些组件无论是在同一主机、同一网络上运行，还是通过 Internet 松散地连接在一起，均使用消息传送来传递数据以协调各自的功能。

为了使大量组件能够同时交换消息，并支持高密度吞吐量，消息的发送就不能取决于消费者是否已做好接收消息的准备。如果某个消费者正忙或处于脱机状态，系统必须允许进行这样的操作：当该消费者准备就绪时再接收消息。这种去耦合的消息发送与接收称为“异步消息传送”。

异步消息传送模型非常适于完成集成复杂系统的任务；对于此类系统，在执行操作的过程中让一个组件为另一个组件提供支持既不切实际，也不值得这样做。尽管异步消息传送放弃了同步系统所具有的某些控制功能，但大大提高了组件间相互作用的灵活性。它还增强了系统的稳定性，因为一个组件的故障并不会导致整个系统瘫痪。

企业消息传送系统的要求

企业应用程序系统一般都包括大量的分布式组件，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。要支持这样的系统，除了支持异步消息传送外，企业消息传送系统还必须满足以下要求：

可靠的传送。 从一个组件向另一个组件传送的消息不能由于网络或系统故障而丢失。这就意味着，系统必须能够保证消息的传送。

安全性。 消息传送系统必须支持基本的安全功能：用户验证、消息及资源的访问授权和线上加密。

可伸缩性。 消息传送系统必须能够在不降低系统性能或消息吞吐量的前提下容纳不断增长的负荷，即用户数量和消息数量的增加。随着业务和应用程序的扩展，这将成为一个很重要的要求。

可用性。 消息传送系统的停机时间必须非常短。也就是说在发生故障时，系统具有足够的冗余以继续提供消息传送服务。

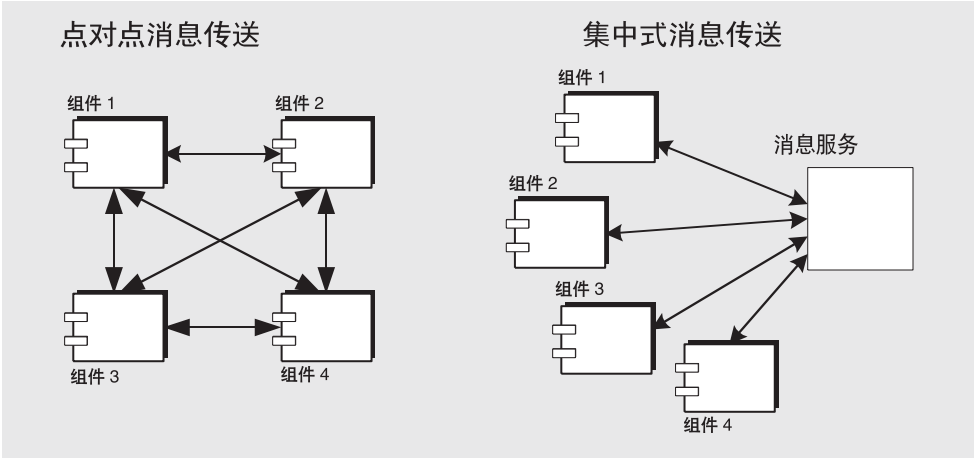
易管理性。 消息传送系统必须提供用于监视和管理消息传送的工具。管理员必须能够优化系统资源以及调节系统性能。

集中式 (MOM) 消息传送

Message Queue 采用集中式消息传送系统，如图 1-1 中所示。在此类系统中，每个消息传送组件只需与一个中央消息服务保持连接。组件通过定义完善的接口与消息服务进行交互。

图中左侧所示为另一种系统，即点对点系统，在该系统中，每个消息传送组件均与所有其他组件保持连接。点对点系统可以实现快速、安全和可靠的传送，但是支持可靠性和安全性的代码必须保存在每一个组件中。消息的发送和接收紧密耦合，因而难以实现异步传送。随着组件被不断添加到系统中，连接的数量将成指数级增加，因此，系统的可伸缩性很差。在点对点系统中也很难实施集中式管理。

图 1-1 集中式消息传送与点对点消息传送



在集中式系统 （企业首选消息传送方案）中，消息服务能够在组件间进行消息路由和传送，并负责可靠且安全地进行传送。因为这种系统中的组件间是松散耦合的，所以更容易实现异步消息传送。

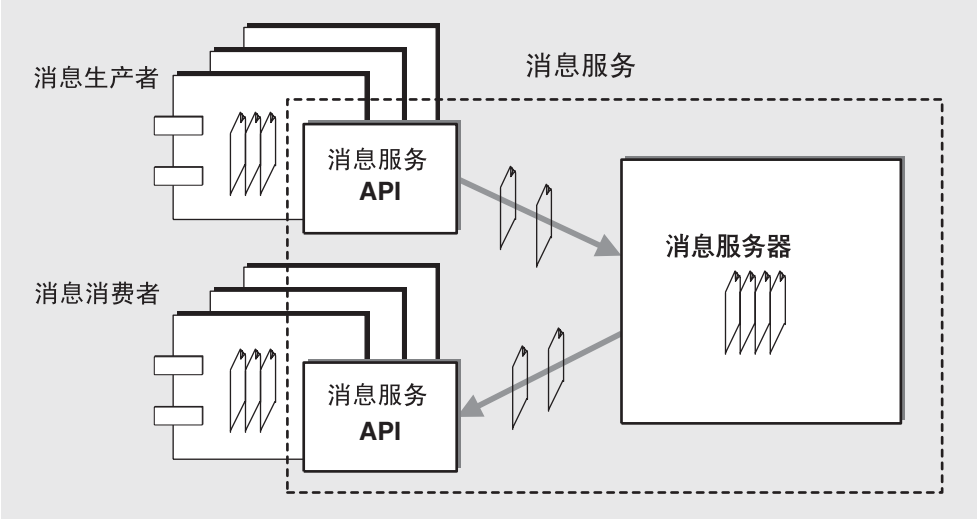
随着消息传送组件被不断添加到系统中，连接的数量成线性增加，这样就可以轻松地通过调整消息服务来调整系统。除连接消息传送客户机外，集中式消息服务还提供了一种管理界面，可以使用该界面来配置行为、监视性能和调节服务，以满足每个消息传送客户机的需要。

基本消息服务体系结构

图 1-2 所示为集中式消息传送系统的基本体系结构。它包括消息**生产者**和消息**消费者**，它们通过公共的**消息服务**来交换消息。同一消息传送组件 （或应用程序）中可以保存任意数量的消息生产者和消息消费者。

消息生产者使用消息服务编程 API 将消息发送至**消息服务器**。消息服务器进行消息路由并将消息传送至一个或多个已注册申请此消息的消息消费者。消费者使用消息服务编程 API 来接收消息。消息服务负责保证将消息传送至所有相应的消费者。

图 1-2 消息服务体系结构



也许将这一过程比喻为交换邮件最恰当不过：尽管一封邮件上标注的是其最终收件人的地址，实际上却要通过邮局进行路由，并经由多个中转站后，才会到达收件人的邮箱。

Java 消息服务 (JMS) 基础

Message Queue 是一种实现了 Java 消息服务 (JMS) 开放标准的企业消息传送系统：它是 **JMS 提供者**。因此， JMS 概念是理解 Message Queue 服务工作方式的基础。

JMS 规范规定了一套管理可靠异步消息传送的规则和语义。该规范定义了消息结构、编程模型和 API。

本节说明了理解本书其余各章所需的 JMS 概念和术语。涵盖下列主题：

JMS 消息结构

在 Message Queue 中，数据是使用 JMS 消息进行交换的。根据 JMS 规范，生产者**客户机**创建的消息包括三部分：标题、属性和主体。

标题

每条 JMS 消息都必须具有标题。标题字段包含用于路由和识别消息的值。

可以通过多种方式来设置标题的值：

- 由 JMS 提供者在生成或传送消息的过程中自动设置
- 由生产者客户机通过在创建消息生产者时指定的设置进行设置
- 由生产者客户机逐一对各条消息进行设置

有关 JMS 定义的标题字段的信息，请参见 Message Queue Developer's Guide for Java Clients 或 Message Queue Developer's Guide for C Clients。可以通过这些标题字段定义消息的目标、到期时间及其优先级等。

属性

消息可以包含称作**属性**的可选标题字段。它们是以属性名和属性值对的形式指定的。可以将属性视为消息标题的扩展，其中可以包括以下信息：创建数据的进程、数据的创建时间以及每条数据的结构。JMS 提供者也可以添加影响消息处理的属性，如是否应压缩消息或如何在消息生命周期结束时废弃消息。

JMS 提供者可以将消息属性用作**选择器**，以对消息进行排序和路由。生产者客户机可以在消息中放置特定于应用程序的属性；而消费者客户机可以选择只接收具有特定属性值的消息。例如，消费者客户机可能只请求获得有关新泽西州兼职雇员工资单的消息。不符合指定的选择标准的消息将不会传送给该客户机。

选择器简化了消费者客户机的工作，并消除了向不需要这些消息的客户机传送消息的开销。不过，由于要处理选择标准，它们也会增加消息服务的一些开销。在 JMS 规范中对消息选择器语法和语义进行了简要说明。

消息主体类型

JMS 消息的类型决定其主体的内容，如表 1-1 中所示。

表 1-1 消息主体类型

类型	说明
StreamMessage	一种主体中包含 Java 基元值流的消息。其填充和读取均按顺序进行。
MapMessage	一种主体中包含一组名 - 值对的消息。没有定义条目顺序。
TextMessage	一种主体中包含 Java 字符串的消息（例如， XML 消息）。
ObjectMessage	一种主体中包含序列化 Java 对象的消息。
BytesMessage	一种主体中包含连续字节流的消息。

JMS 编程模型

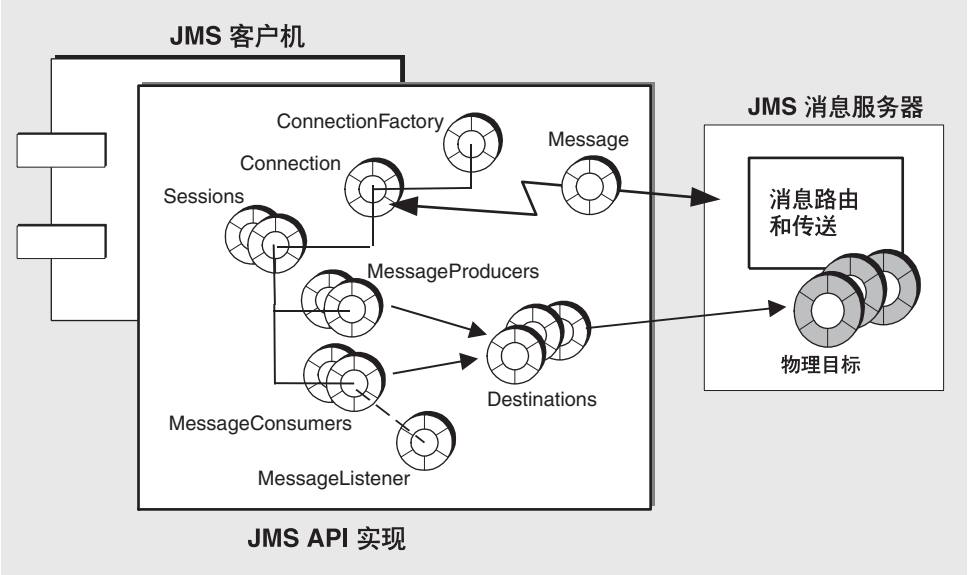
JMS 编程模型支持异步消息传送服务体系结构：JMS 客户机通过 JMS 消息服务交换消息。JMS 提供者提供执行 JMS 消息传送所需的对象；这些对象用于实现 JMS 应用程序编程接口 (API)。

本节说明 JMS 消息传送所需的编程对象，并介绍用于发送和接收消息的传送模型（点对点及发布 / 订阅）。

编程对象

图 1-3 中显示了用于设置 JMS 客户机以进行消息传送的对象。

图 1-3 JMS 编程对象



在 JMS 编程模型中， JMS 客户机使用**连接工厂**对象 (ConnectionFactory) 创建一个**连接**，向 JMS 消息服务器发送消息以及从 JMS 消息服务器接收消息均通过此连接来进行。连接对象 (Connection) 表示客户机与消息服务器之间的活动连接。

创建连接时，将分配通信资源以及验证客户机。这是一个相当重要的对象，大多数客户机均使用一个连接来进行所有的消息传送。

该连接用于创建**会话**对象 (Session)。会话是一个用于生成和使用消息的单线程上下文。它用于创建消息以及发送和接收消息的生产者和消费者，并为所传送的消息定义发送顺序。会话通过大量**确认**选项或通过事务来支持可靠传送。

客户机使用消息生产者对象 (MessageProducer) 向指定的物理**目标**（在 API 中由目标对象表示）发送消息。消息生产者可指定一个默认的消息标题值，例如，传送模式（持久性与非持久性） 、优先级和有效期，以控制生产者向物理目标发送的所有消息。

同样，客户机使用消息消费者对象 (MessageConsumer) 从指定的物理目标 （在 API 中表示为目标对象）接收消息。共有两种类型的目标，分别是**队列**和**主题**，具体类型取决于消息传送模型。

消息消费者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。

消息消费者可以支持同步或异步消息使用。

- 同步使用是指，消费者明确请求传送消息并随后使用该消息。
- 异步使用是指，自动将消息传送给为消费者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，客户机将使用该消息。

编程域：消息传送模型

JMS 支持两种截然不同的消息**传送模型**：点对点模型和发布 / 订阅模型。

点对点 （队列目标） 消息从一个生产者传送至单个消费者。在此传送模型中，目标类型是**队列**。消息首先被传送至队列目标，然后从该队列将消息传送至对此队列进行注册的某个消费者，一次只传送一条消息。可以向队列目标发送消息的生产者的数量没有限制，但每条消息只能发送至、并由一个消费者成功使用。如果没有已经向队列目标注册的消费者，队列将保留它收到的消息，并在某个消费者对该队列进行注册时将消息传送给该消费者。

发布 / 订阅 （主题目标） 消息从一个生产者传送至任意数量的消费者。在此传送模型中，目标类型是**主题**。消息首先被传送至主题目标，然后传送至**所有**已**订阅**此主题的活动消费者。可以向主题目标发送消息的生产者的数量没有限制，并且每个消息可以发送至任意数量的订阅消费者。

主题目标也支持**长期订阅**。长期订阅表示消费者已注册了主题目标，但在消息到达目标时该消费者可以处于非活动状态。当消费者再次处于活动状态时，将会接收该消息。如果消费者均没有注册某个主题目标，该主题只保留注册了长期订阅的非活动消费者的消息。

这两种消息传送模型使用三组表示不同编程**域**的 API 对象 （其语义略有不同）进行处理，如表 1-2 所示。

表 1-2 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

* 取决于编程方法，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 还具有编程接口为全新接口的优点，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，则应选择统一的域 API。

Message Queue 产品包含的示例应用程序以及 Message Queue 文档中提供的很多代码示例均使用单独的编程域。

可靠消息传送

可以将消息的**传送模式**设置为持久性或非持久性；此模式控制消息传送的可靠性。

- 保证只将**持久性消息**传送并成功使用一次。在消息服务发生故障时，持久性消息不会丢失。对于这些消息，可靠性是优先考虑的因素。
- 保证最多将**非持久性消息**传送一次。在消息服务发生故障时，非持久性消息将会丢失。对于这些消息，可靠性并非主要的考虑因素。

对于持久性消息，确保可靠性有两个方面。一方面，通过使用确认和事务来确保成功生成和使用消息。另一方面，通过将消息放在持久性存储库中，确保在将持久性消息传送至消费者之前消息服务不会丢失持久性消息。

以下各节介绍这两个方面的可靠性保证措施。

确认 / 事务

可靠的消息传送取决于保证成功地将持久性消息从消息生产者传送到消息服务器上的物理目标，然后再成功地从该物理目标传送到消息消费者。这种可靠性可以通过 JMS 会话支持的两个通用机制实现：**确认**或**事务**。事务可以是本地事务或分布式事务（由分布式事务管理器控制）。

确认

确认是在客户机与消息服务间发送的消息，用于确保可靠地进行传送。

生成消息时，消息服务确认它已收到传送的消息，将该消息置于其目标中并进行持久性存储。生产者的 `send()` 方法会阻塞，直至确认返回为止。

使用消息时，客户机确认已收到从某个目标传送来的消息并已使用，然后消息服务从该目标中删除消息。JMS 规定了不同的确认模式，它们分别代表不同的可靠性级别。在其中的某些模式下，客户机会阻塞，等待消息服务器确认已删除某个消息，并因此无法重新传送该消息。

本地事务

可以将会话配置为**已处理**，这样，可以将一个或多个消息的生成和 / 或使用组成原子单元，也就是**事务**。 JMS API 提供了启动、提交或回滚事务的方法。

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，并在 JMS 客户机发出提交事务的调用时完成这些操作。如果事务中特定的发送或接收操作失败，则出现异常。客户机代码通过忽略异常、重试操作或回滚整个事务来处理异常。在事务提交时，将完成其所有操作。在事务进行回滚时，将取消所有成功的操作。

本地事务的范围始终为一个会话。也就是说，可以将单个会话的上下文中执行的一个或多个生产者或消费者操作组成一个本地事务。

由于事务的范围只能为单个的会话，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。（换句话说，至目标的消息传送和随后进行的至客户机的消息传送不能放在同一个事务中。）

分布式事务

JMS 规范还支持**分布式**事务。也就是说，消息的生成和使用可作为较大的分布式事务的一部分，该分布式事务中包括涉及其他资源管理器 （如数据库系统）的操作。在分布式事务中，分布式事务管理器使用在 Java 事务 API (JTA)、XA 资源 API 规范中定义的两阶段提交协议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， JTA 规范说明了资源管理器和分布式事务管理器之间的交互。

支持分布式事务是指消息传送客户机可通过 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。当客户机端进行 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 (JTS) 提供）协作来跟踪分布式事务中的各种发送和接收操作、事务状态并完成消息传送操作。

处理本地事务时，客户机通过忽略异常、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消费者之前，消息服务不会将它们丢失。这意味着，当持久性消息到达其物理目标时，消息服务器必须将其置于持久性**数据存储库**中。如果消息服务器由于某种原因发生故障，它可以恢复此消息并将其传送至相应的消费者。

消息服务器还必须持久性地存储长期订阅。否则，当消息服务器发生故障时，就无法向长期订户传送消息；消息到达主题目标后，长期订户会恢复活动状态。

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。

JMS 受管理对象

JMS 编程模型中使用的两个对象 （连接工厂和目标）可能会因提供者的 JMS 规范实现而有所不同。

- *连接工厂对象*用于创建以下连接：其行为取决于提供者传送消息时所采用的协议和机制。
- *目标对象*用于指定代理上的物理目标的名称，它取决于具体的命名约定及消息服务器上的物理目标的功能。

为使提供者在定义这些对象时具有最大限度的灵活性，同时使客户机具有移植性，JMS 规范定义了**受管理对象**（针对连接工厂和目标），其中封装了特定于提供者的信息。这些对象由管理员创建和配置，并存储在 JNDI 名称空间（对象存储库）中，由客户机通过标准 JNDI 查找代码来访问。

受管理对象允许 JMS 客户机使用逻辑名称查找和引用特定于提供者的对象。这样，客户机代码无需知道提供者使用的特定命名语法、寻址语法或可配置属性。从而使代码与提供者无关。

第 37 页 “受管理对象”一节提供了有关 Message Queue 中使用的受管理对象的其他信息。

注 JMS 规范并不要求您必须使用 JNDI 查找来访问受管理对象。客户机代码可以实例化连接工厂和目标对象，并设置其属性值。不过，这意味着客户机代码无法移植到其他提供者上。

第 2 章

Message Queue 简介

Message Queue 是一种符合 JMS 1.1 规范的可靠、异步的消息传送服务。此外，为满足大规模企业部署的需要， Message Queue 还提供了 JMS 规范要求之外的许多功能。

本章说明 Message Queue 服务体系结构，并介绍其企业功能。本章涵盖以下主题：

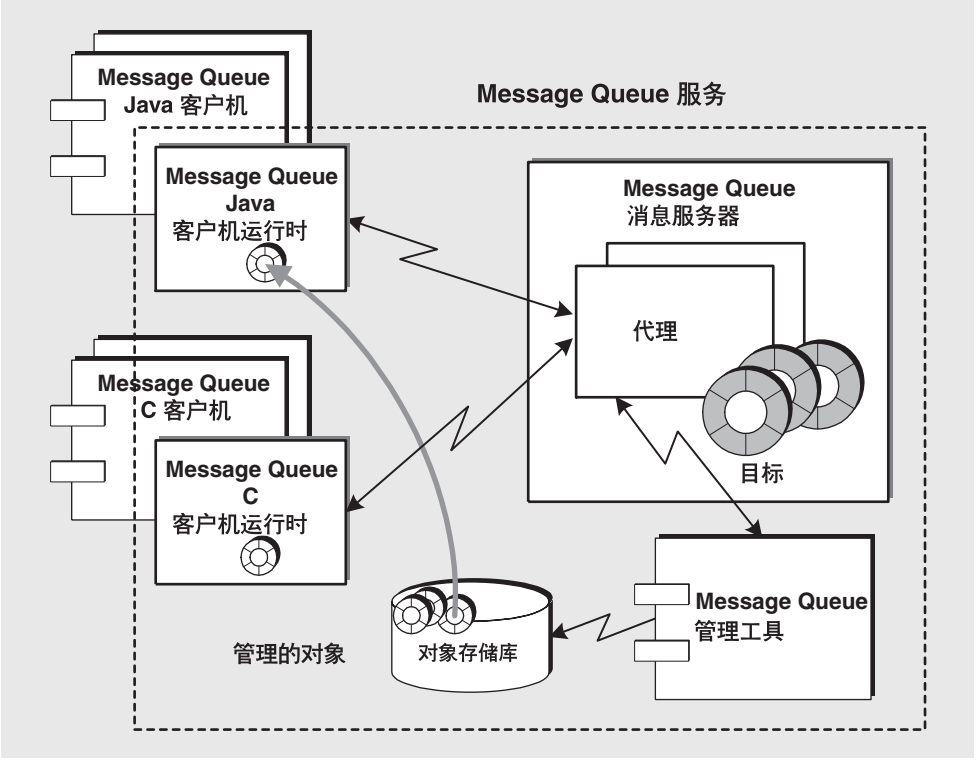
消息服务体系结构

Message Queue 服务由下列元素构成：

- 第 33 页 “消息服务器”
- 第 33 页 “客户机运行时”
- 第 37 页 “受管理对象”
- 第 39 页 “管理工具”

图 2-1 显示了这些元素如何协同工作。

图 2-1 Message Queue 服务体系结构



如图中所示， Message Queue 客户机使用 Java 或 C API 来发送或接收消息。这些 API 是在 Java 或 C 客户机运行时库中实现的，该库的实际作用是创建到代理的连接并为请求的连接服务适当地封装位。如果应用程序使用受管理对象，则客户机运行时会在对象存储库中找到这些对象，并使用它们来配置连接和定位物理目标。代理会路由和传送消息。管理员可以使用 Message Queue 管理工具来管理代理，并向对象存储库中添加受管理对象。

以下各节简要说明这些元素。

消息服务器

消息服务器由一个或多个代理组成，它执行消息的路由和传送。它是 Message Queue 服务的核心。

消息服务器由单个**代理**或一组协同工作的代理（代理**群集**）组成，用于执行消息路由和传送服务。代理是执行下列任务的一个进程：

- **验证**用户及其想要执行的**授权**操作
- 建立与客户机的通信通道
- 从生产者客户机接收消息，然后将消息置于其各自的物理目标中
- 将消息路由并传送到一个或多个消费者客户机
- 确保传送可靠
- 提供用来监视系统性能的数据。

有关消息服务器、其内部组件以及这些组件所执行的功能的详细说明，请参见第 59 页第 4 章 “消息服务器”。

Message Queue Enterprise Edition 支持使用代理群集，该群集由多个相互连接的代理实例组成，使消息服务器能够根据消息通信流量进行调整。有关体系结构和群集配置问题的说明，请参见第 5 章 “代理群集”。

客户机运行时

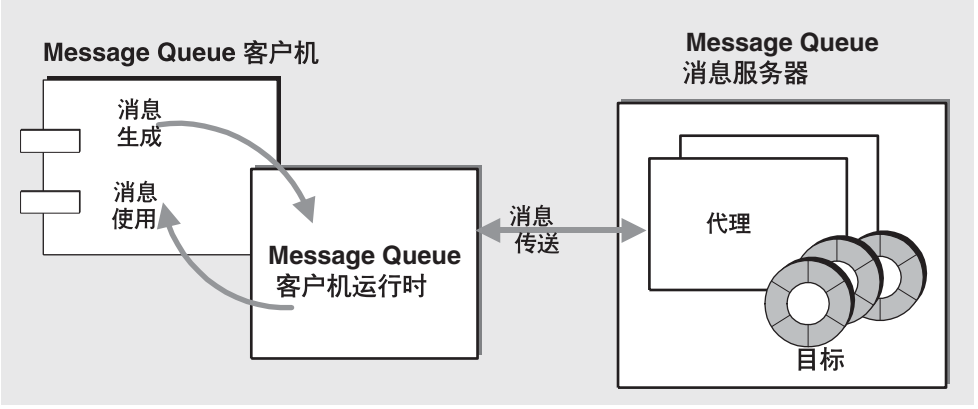
Message Queue **客户机运行时**为客户机应用程序提供了它与 Message Queue 服务的接口。客户机运行时支持 Message Queue 客户机生成消息（向目标发送消息）和使用消息 （从目标中检索消息）所需的所有操作。

Message Queue 客户机运行时有两种语言实现，如第 32 页图 2-1 中所示：

- **Java 客户机运行时。**为 Java 客户机应用程序和组件提供实现 JMS API 及其与 Message Queue 消息服务器进行交互所需的所有对象。这些接口对象包括连接、会话、消息、消息生产者和消息消费者。
- **C 客户机运行时。**为 C 客户机应用程序和组件提供与 Message Queue 服务器进行交互所需的 C 编程接口。（C 客户机运行时支持 JMS API 消息传送模型的程式化版本。）

图 2-2 说明了客户机运行时在 Message Queue 客户机与消息服务器间所起到的核心作用。消息生成和使用是客户机与客户机运行时之间的交互；而消息传送是客户机运行时与消息服务器之间的交互。

图 2-2 客户机运行时和消息传送操作



客户机运行时执行下列功能：

- 管理向消息服务器传送消息
- 建立连接
- 创建客户机的标识
- 实现客户机确认
- 控制连接间的消息流
- 可以覆盖由生产者客户机设置的消息标题值

以下小节简要说明客户机运行时功能。可以通过配置连接工厂对象的属性来自定义客户机运行时某些方面的功能。

连接处理

要配置连接处理功能，必须指定客户机要连接的代理的主机名和端口以及所需的连接服务类型。如果连接到的代理是某个群集的一部分，则必须指定要连接到的地址列表。如果某个代理没有联机，客户机运行时可以让您连接到群集中的其他代理。

在企业版中，如果连接失败，则客户机运行时可以自动重新连接到代理。如果客户机连接的代理是某个群集的一部分，则重新连接的代理可以是原来连接的同一代理，也可以是不同的代理。

如果代理实例不使用共享的、高可用性的持久性存储（可以通过将 Message Queue 与 Sun Cluster 相集成来实现），则当代理发生故障（或连接中断）时，它所保留的持久性消息和其他状态信息可能会因重新连接到其他代理实例而丢失。也就是说，重新连接能够提供连接故障转移，但不能保证数据可用性。

客户机标识

如果应用程序认为有用，可以在任何连接上设置客户机 ID；要标识长期订户，就必须设置客户机 ID。

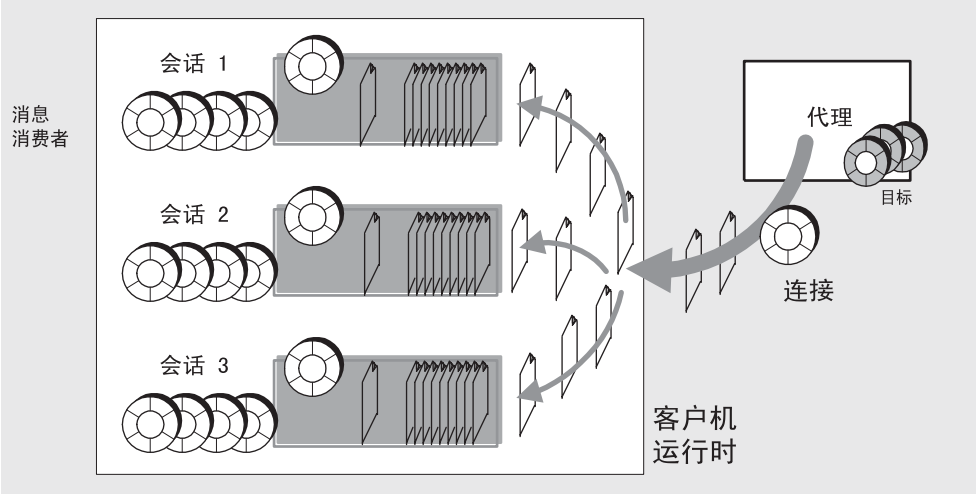
为跟踪长期订阅，代理使用唯一的客户机标识。客户机 ID 用于标识在将消息传送到主题目标时处于非活动状态的长期订户。代理会保留传送给这些订户的消息，在订户恢复活动状态时再将消息提供给它们。

因此，每当在已部署的应用程序中使用长期订阅时，都必须设置客户机标识符。Message Queue 功能，允许您在指定客户机 ID 时使用一种特殊的变量名语法。这样，就可以为从连接工厂对象获取的每个连接获取不同的客户机 ID，无论该对象是由管理员创建的，还是通过编程方式创建的。有关详细信息，请参见 Message Queue 管理指南。

向消费者分发消息

客户机运行时接收代理通过连接传送的消息，并分发到适当的 Message Queue 会话，在这些会话中，将消息排队以等待供各自的消息消费者使用，如第 36 页图 2-3中所示。

图 2-3 将消息传送到 Message Queue 客户机运行时



每次从每个会话队列中取出一条消息，然后同步使用（通过调用 `receive()` 方法的客户机线程）或异步使用（通过调用消息侦听器对象的 `onMessage()` 方法的会话线程）消息。（会话是单线程的。）

传送到客户机运行时的消息流是在每个消费者级别测量的。可以通过适当调整连接工厂属性来平衡消息流，从而使传送到一个会话的消息不会对使用同一连接传送到其他会话的消息产生不利影响。

确保可靠的消息传送

客户机运行时在确保可靠传送消息方面具有非常重要的作用。它支持 JMS 规范的客户机确认和事务模式，并控制各种用于保证可靠传送的代理确认功能。

JMS 规范描述了多种客户机确认模式，这些模式可以提供不同级别的可靠性。这些确认模式及由 Message Queue 实现的其他模式是在消息使用上下文中描述的（请参见第 54 页 “客户机确认”）。

在持久性消息和可靠传送的情况下，代理通常会在完成用来确保只对消息使用且只使用一次的操作后对客户机运行时进行确认。可以使用连接工厂属性来抑制这些代理确认，从而节省网络带宽和处理过程。当然，这种抑制代理确认的方式也会使传送的可靠性得不到保证。

消息流控制

客户机运行时是连接间消息流的监控者。除通过连接传送的一般 JMS 有效负荷消息外， Message Queue 还会发送各种控制消息，这些消息用于保证传送的可靠性、管理连接间的消息流以及执行其他控制功能。

由于有效负荷消息和控制消息争用同一连接，因此可能会发生冲突，从而导致拥堵。客户机运行时会强制实施各种可配置的流量限制和测量模式，来最大限度减少有效负荷消息与控制消息间的冲突，进而最大限度地提高消息吞吐量。

覆盖消息标题值

客户机运行时可以覆盖 JMS 消息标题字段，此字段指定了消息的持久性、生命周期以及优先级。

Message Queue 允许在连接级别覆盖消息标题：覆盖应用于在给定连接上下文中生成的所有消息。

由于客户机运行时能够覆盖消息标题值，因此， Message Queue 管理员对消息服务器资源的控制能力得到提升。不过，覆盖这些字段的风险是会干扰特定于应用程序的要求（例如，消息持久性）。因此，应事先咨询适当的应用程序用户或设计人员，再使用此功能。

其他功能

客户机运行时还执行其他几种不同类型的功能：

- **队列浏览特性。**浏览队列目标内容时，可以配置客户机运行时一次检索的消息数以及等待消息的时间。
- **消息压缩。**Java 客户机运行时可以在生成消息时对其进行压缩，并在使用消息时对其进行解压缩。是否进行这种压缩或解压缩取决于客户机创建消息时在消息标题中设置的特定于 Message Queue 的消息属性。

受管理对象

受管理对象封装特定于提供者的有关连接和目标的实现和配置信息。可以通过编程方式来创建受管理对象，也可以使用管理员工具来创建和配置它们，并将其存储在对象存储库中，供客户机应用程序通过标准 JNDI 查找代码进行访问。

Message Queue 提供了下表中所示的受管理对象类型。

表 2-1 Message Queue 受管理对象类型

类型	说明
目标	表示代理中的一种物理目标。包含代理中物理目标的特定于提供者的名称。消息消费者和 / 或消息生产者使用目标受管理对象来访问相应的物理目标。
连接工厂	在客户机应用程序与 Message Queue 消息服务器之间建立物理连接。还对 Message Queue 客户机运行时（控制物理连接的功能）进行配置。设置连接工厂受管理对象的属性值时，请指定应用于它所建立的所有连接的属性。
XA 连接工厂	用于建立支持分布式事务的物理连接（请参见第 28 页“分布式事务”）。XA 连接工厂对象与普通连接工厂对象共享同一组属性，但它可以启用支持分布式事务所需的其他机制。
SOAP 端点	标识 SOAP 消息的最终目标：这是可以接收 SOAP 消息的 Servlet 的 URL。可以配置 SOAP 端点受管理对象来指定多个 URL。还可以指定与对象关联的查找名以及对象存储库属性。

通过 JNDI 使用受管理对象

尽管 JMS 规范并不要求 JMS 客户机在 JNDI 名称空间中查找受管理对象，但这样做的优点是显而易见的：可以实现单一的控制源，不必重新编码即可对连接 （客户机运行时功能）进行配置和重新配置，并可以将客户机移植到其他 JMS 提供者上。

通过使用受管理对象，可以更容易地控制和管理 Message Queue 服务：

- 管理员可以通过要求客户机应用程序访问预配置的连接工厂对象来指定客户机运行时功能。
- 管理员可以通过要求客户机应用程序访问与现有物理目标对应的预配置目标受管理对象来控制物理目标的扩散。

换言之， Message Queue 管理员可以使用受管理对象来控制消息服务配置细节，同时使客户机应用程序成为与提供者无关的应用程序。

使用受管理对象意味着，客户机程序员不需要了解特定于提供者的语法和对象命名惯例或提供者特有的配置属性。实际上，通过将受管理对象指定为只读，管理员可以确保客户机应用程序无法更改最初创建受管理对象时为其设置的属性值。

虽然客户机应用程序可以独立地实例化连接工厂和目标受管理对象，但这样会削弱受管理对象的基本用途。 Message Queue 管理员需要控制应用程序所需的代理资源以及调节消息传送性能。此外，通过直接实例化受管理对象，将使客户机应用程序成为与提供者有关的应用程序。

尽管有这些参数，应用程序通常仍会在没有管理控制问题的开发环境中实例化受管理对象。

对象存储库

Message Queue 受管理对象位于对象存储库中 （请参见第 32 页图 2-1），客户机应用程序可以通过 JNDI 查找访问它们。 Message Queue 支持以下两种类型的对象存储库：标准 LDAP 目录服务器和文件系统对象存储库。

LDAP 服务器对象存储库　对于生产型消息传送系统，建议使用 LDAP 服务器对象存储库。很多供应商都提供 LDAP 实现，而且 LDAP 实现可以在分布式系统中使用。 LDAP 服务器还提供对生产环境特别有用的安全功能。

文件系统对象存储库　Message Queue 支持文件系统对象存储库 （不建议将其用于生产系统），但它的优势是在开发环境中非常容易使用。与其设置一台 LDAP 服务器，还不如在本地文件系统中创建一个目录。不过，文件系统对象存储库不能用作在多个计算机节点上部署的客户机的集中式对象存储库，除非这些客户机可以访问该对象存储库所在的目录。

管理工具

Message Queue 管理工具包含一组命令行实用程序和一个图形用户界面 (GUI) 管理控制台。

命令行实用程序　Message Queue 提供了一套命令行实用程序来执行所有 Message Queue 管理任务，如启动和管理代理、创建和管理物理目标、管理受管理对象以及执行其他更专门化的管理任务。所有的命令行实用程序均共用相同的格式、语法约定和选项。有关使用命令行实用程序的更多详细信息，请参见 Message Queue 管理指南。

管理控制台　控制台提供了 Message Queue 命令行实用程序的部分功能。可以使用管理控制台来管理代理、创建和管理物理目标以及管理受管理对象。不过，无法执行部分命令行实用程序能够执行的更专门化的任务。例如，无法使用管理控制台来启动代理、创建代理群集或管理用户系统信息库。必须使用 Message Queue 命令行实用程序来执行这些任务。

Message Queue 管理指南提供了一个简明扼要的实用教程，帮助您熟悉管理控制台，并说明了如何使用控制台完成基本任务。

管理控制台和某些命令行实用程序允许对代理和物理目标进行远程管理。

产品功能

Message Queue 服务和上一节中介绍的体系结构完全实现了 JMS 1.1 规范对于消息传送的可靠性、异步性及灵活性的要求。有关 JMS 兼容性相关问题的文档，请参见附录 A“可选 JMS 功能 Message Queue 实现”。

不过， 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 可以集成包含大量分布式组件的系统，而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。

Message Queue 的企业功能 （下文中阐述）分为以下几个类别：

- 第 40 页“集成支持功能”
- 第 42 页“安全性功能”
- 第 43 页“可伸缩性功能”
- 第 44 页“可用性功能”
- 第 44 页“易管理性功能”
- 第 45 页“灵活的服务器配置功能”

集成支持功能

通过包含对多种传输协议的支持、 Message Queue 服务的 C 客户机接口、对 SOAP (XML) 消息的支持以及可插入的 J2EE 资源适配器，Message Queue 使您可以集成企业中各种完全不同的应用程序和组件。

多传输支持

Message Queue 支持客户机使用多种不同的传输协议（包括 TCP 和 HTTP）并通过安全连接与 Message Queue 消息服务器进行交互的功能。

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。客户机生成的消息通过 HTTP 经过防火墙传送到隧道 Servlet。隧道 Servlet 从 HTTP 请求提取消息，然后将消息通过 TCP/IP 传送到代理。类似地， Message Queue 还支持使用 HTTPS 隧道 Servlet 的安全 HTTP 连接。有关 HTTP 连接体系结构的详细信息，请参见第 63 页“HTTP/HTTPS 支持”。有关建立和配置 HTTP/HTTPS 连接的信息，请参见 Message Queue 管理指南。

安全连接 Message Queue 提供 TCP/IP 和 HTTP 传输上基于安全套接字层 (SSL) 标准的安全消息传输。这些基于 SSL 的连接服务允许对在客户机和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连接到代理的客户机，客户机使用该证书来建立加密连接。有关创建自签名证书来启用基于 SSL 的连接服务的信息，请参见 Message Queue 管理指南。

C 客户机接口

除了支持 Java 语言消息传送客户机外， Message Queue 还提供了 Message Queue 服务的 C 语言接口。C API 使传统 C 应用程序及 C++ 应用程序可以参与基于 JMS 的消息传送。不过，使用 Message Queue C API 的客户机无法移植到其他 JMS 提供者上。

支持 Message Queue C API 的 C 客户机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。 C 客户机运行时也不支持大多数 Message Queue 企业功能。

有关 C API 功能及其如何使用 C 数据类型和函数实现 JMS 编程模型的详细信息，请参见 Message Queue Developer's Guide for C Clients。

SOAP (XML) 消息传送支持

Message Queue 支持创建和传送符合简单对象访问协议 (SOAP) 规范的消息。SOAP 允许在分散的分布式环境中的点之间交换结构化 XML 数据或 SOAP 消息。SOAP 消息是一种也可以包含附件 （不必为 XML 格式）的 XML 文档。

由于 SOAP 消息以 XML 格式编码，因此与平台无关。可以使用它们来访问传统系统中的数据以及在企业间共享数据。XML 提供的数据集成还使这种技术成为基于 Web 计算（如 Web 服务）的自然选择。防火墙可以识别 SOAP 数据包，并可以根据 SOAP 消息标题中公开的信息对消息进行过滤。

Message Queue 实现了 SOAP with Attachments API for Java (SAAJ) 规范。 SAAJ 是一种应用程序编程接口，可以通过实现该接口来支持用于 SOAP 消息传送的编程模型及提供可用于构建、发送、接收和检查 SOAP 消息的 Java 对象。 SAAJ 定义了两个软件包：

- `javax.xml.soap`：可以使用该软件包中的对象来定义 SOAP 消息的各个部分以及集合和分解 SOAP 消息。还可以在没有提供者支持的情况下使用该软件包发送 SOAP 消息。

- `javax.xml.messaging`：可以使用该软件包中的对象，通过提供者来发送 SOAP 消息以及接收 SOAP 消息。

Message Queue 提供了用于将 SOAP 消息与 JMS 消息进行双向转换的实用程序。使用它们可依次实现：Servlet 接收 SOAP 消息，然后将其转换为 JMS 消息； Message Queue 服务将 JMS 消息传送给 JMS 消费者，并将其转换回 SOAP 消息，然后再传送到 SOAP 端点。换言之， Message Queue 支持在 SOAP 端点间可靠、异步地交换 SOAP 消息的能力。更简单地讲，就是将 SOAP 消息发布给 Message Queue 订户。

有关其他信息，请参见 Message Queue Developer's Guide for Java Clients。

J2EE 资源适配器

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进行交互。简言之， J2EE 平台需要 JMS 支持。

这一支持是通过在 J2EE 编程模型中使用消息驱动 Bean (MDB) 提供的，MDB 是一种可以使用 JMS 消息的专用类型的 Enterprise Java Bean (EJB) 组件。符合 J2EE 规范的应用服务器必须提供支持 JMS 消息传送的 MDB 容器。这可以通过在应用服务器中插入 JMS 资源适配器来实现。 Message Queue 提供了这样的资源适配器。

通过将 Message Queue 资源适配器插入应用服务器，即可在应用服务器环境中部署和运行的 J2EE 组件（包括 MDB）之间以及这些组件与外部 JMS 组件之间交换 JMS 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 6 章 “Message Queue 和 J2EE”。

安全性功能

对于大多数企业应用而言，对存储和传送的消息数据进行保护至关重要。 Message Queue 提供了多级安全性，其中包括用户验证、受控的资源访问以及消息加密。

验证 Message Queue 支持基于密码的用户验证。系统将根据存储在平面文件 (flat file) 或 LDAP 用户系统信息库中的密码决定是否授予用户连接到消息服务器的权限。将记录所有连接尝试（用户和主机）的相关信息，并可以对其进行跟踪。

授权 访问控制表 (ACL) 对代理连接和物理目标访问提供了可配置的精细控制。同时支持用户和组访问。授权是针对代理逐一执行的；每个代理可以具有不同的访问控制文件。

加密 SSL 支持允许使用全强度 SSL 实现对消息服务器与其客户机间的所有消息流量（无论是采用 TCP/IP 还是 HTTP 连接）进行加密。

有关填充用户系统信息库、管理访问控制表和设置 SSL 支持的信息，请参见 Message Queue 管理指南。

可伸缩性功能

Message Queue 允许您随用户、客户机连接和消息负荷的增长扩展应用程序。

可伸缩的连接功能

Message Queue 代理可以处理成千上万个并发连接。默认情况下，每个连接由专用的代理线程进行处理。因为这样会使线程即使在连接处于空闲状态时仍被占用，所以可以对连接服务进行配置，使多个连接可以共享同一线程。这种共享线程池模型可以显著增加代理可支持的连接的数量。有关详细信息，请参见第 63 页 “线程池管理器”。

代理群集

随着连接数和通过代理传送的消息数的增加，可以通过为 Message Queue 服务器添加其他代理实例来管理附加的负荷。代理群集在多个代理实例间平衡客户机连接和消息传送负荷，从而使消息服务器具有较高的可伸缩性。这些代理实例可以位于同一主机上，也可以分布在网络中。随着业务需求的增长，群集将是一种提高消息吞吐量和扩展消息传送带宽的理想方法。第 77 页第 5 章 “代理群集”介绍了代理群集，Message Queue 管理指南 中对代理群集进行了更全面的阐述。

多个消费者的队列传送

按照 JMS 规范，只能将队列目标中的消息传送到一个消费者。Message Queue 允许多个消费者向队列注册。然后，代理可将消息分发至其他注册消费者，在它们之间平衡负荷，从而允许系统实现扩展。

多个消费者的队列传送是使用一种可配置的负荷平衡方法实现的。使用这种方法，可以指定最大活动消费者数量及最大备份消费者数量。备份消费者会在活动消费者发生故障时接替它的工作。此外，负荷平衡机制还将消费者的当前容量和消息处理速率考虑在内。

有关负荷平衡的队列传送的详细信息，请参见第 53 页 “多个消费者的队列传送”。

可用性功能

Message Queue 提供了很多用于最大限度缩短服务停机时间的功能。这些功能包括故障预防机制以及通过与 Sun Cluster 集成来提供高可用性的功能。

消息服务稳定性

确保消息服务可用性的一种最有效的方法是，提供一种既能实现高性能又能最大限度减少故障的服务。 Message Queue 提供了转移内存过载或性能拥堵的机制。这些机制将在消息服务器和客户机运行时上运行。

消息服务器资源管理 由于消息服务器的内存和 CPU 资源有限，因此可能会因过载情况严重而停止响应或出现不稳定的情况。如果消息生成速率远远超过消息使用速率，则通常会发生这种情况。要避免出现以上情况，可以在物理目标级别和系统范围级别对代理进行配置以防止内存超限。有关详细信息，请参见第 66 页 “内存资源管理”。

客户机运行时消息流控制 此外， Message Queue 还提供了用于控制向客户机运行时传送消息的机制。可以使用流控制机制来优化向客户机运行时传送消息，同时防止客户机出现内存不足的现象。有关详细信息，请参见第 37 页 “消息流控制”。

自动重新连接到消息服务器

Message Queue 提供了自动重新连接功能：如果消息服务器与客户机间的连接发生故障， Message Queue 会保持客户机的状态，同时尝试重新建立连接。在大多数情况下，重新建立连接后， Message Queue 将以透明方式恢复消息的生成和使用。有关详细信息，请参见 Message Queue 管理指南。

通过 Sun Cluster 实现高可用性

尽管 Message Queue 的代理群集功能使消息服务器具有较高的可伸缩性，但目前尚不支持从群集内的一个代理实例到另一个代理实例的故障转移。不过，可以将 Message Queue 与 Sun Cluster 软件集成，以提供高可用性的消息服务器。通过使用为 Message Queue 开发的 Sun Cluster 代理，Sun Cluster 可以确保代理发生故障时状态数据不会丢失，从而使得消息服务器能够以透明方式得到立即恢复，而不会停机。

易管理性功能

Message Queue 提供了多种可用于监视和管理消息服务以及调节消息服务性能的功能。

强大的管理工具

Message Queue 同时提供了命令行工具和 GUI 工具来管理 Message Queue 消息服务器及管理目标、事务、长期订阅和安全性 （请参见第 39 页 “管理工具”）。

Message Queue 还支持对消息服务器进行远程监视和管理，同时提供用于管理 JMS 受管理对象、用户系统信息库、符合 JDBC 规范的插入式数据存储库以及自签名服务器证书的工具。有关使用这些管理工具的信息，请参见 Message Queue 管理指南。

基于消息的监视 API

Message Queue 提供了基于 JMS 的简单监视 API，可以使用它来创建自定义的监视应用程序。这些监视应用程序是从特殊主题目标检索度量消息的消费者。度量消息包含 Message Queue 代理提供的监视数据 （请参见第 72 页 “度量消息生产者 （企业版）”）。

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，其中介绍了如何开发使用度量消息的 Message Queue 客户机。有关如何配置度量消息生成的信息，请参见 Message Queue 管理指南。

可调节的性能

为获得最佳性能， Message Queue 提供了许多用以调节消息服务器和客户机运行时的方法。可以对主要资源进行监视及对内存使用情况、线程资源、消息流、连接服务、可靠性参数和其他影响消息吞吐量与系统性能的因素进行调整。有关如何调节消息服务性能的详细信息，请参见 Message Queue 管理指南。

灵活的服务器配置功能

Message Queue 允许您选择持久性对象、用户信息及受管理对象的存储方式。

可配置的持久性

为保证消息传送， Message Queue 会存储消息和其他持久性对象，直到消息被使用为止。除提供基于文件的高性能持久性存储外， Message Queue 还支持可配置的持久性。利用这一功能，可以将持久性消息存储在符合 JDBC 规范的嵌入式或外部数据库 （如 Oracle 8i）中。有关详细信息，请参见第 67 页 “持久性管理器”。

LDAP 服务器支持

Message Queue 为基于文件的存储同时提供了验证和授权所需的受管理对象和用户信息。不过， Message Queue 还支持使用 LDAP 服务器作为受管理对象存储库和用户系统信息库。LDAP 服务器提供了更安全、标准的方法来存储和检索此类信息，建议在生产系统中使用。有关使用 LDAP 服务器作为受管理对象存储库和用户系统信息库的信息，请参见 Message Queue 管理指南。

产品版本

可在两个版本中找到 Message Queue 企业版和平台版。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。下表对这些功能集进行了比较。有关这些功能的说明，请参见第 40 页 “产品功能”。

表 2-2 功能比较：企业版和平台版

企业版	平台版
高级集成支持功能	
HTTP 支持、 TCP 支持	TCP 支持
基于安全套接字层 （SSL） 标准的安全连接	基于安全套接字层 （SSL） 标准的安全连接
C 语言 API、 Java API	Java API （只能以试用许可证使用 C API。）
SOAP （XML） 消息传送支持	SOAP （XML） 消息传送支持
J2EE 资源适配器	J2EE 资源适配器
安全性功能	
通过平面文件或 LDAP 用户系统信息库进行验证；使用访问控制文件和 SSL 加密进行授权。	同企业版

表 2-2 功能比较：企业版和平台版 （续）

企业版	平台版
可伸缩性功能	
可伸缩的连接功能	固定连接功能
消息服务器可以作为代理群集来实现	单个代理消息服务器
可以向无限多个消息消费者进行队列传送 （每个队列）	最多可以向三个消息消费者进行队列传送 （每个队列）
可用性功能	
通过内存资源管理和消息流控制来实现消息服务的稳定性	同企业版
客户机连接故障转移到群集内的另一个代理或自动重新连接到同一代理。	无客户机连接故障转移。允许自动重新连接到同一代理。
通过 Sun Cluster 实现高可用性	通过 Sun Cluster 实现高可用性
易管理性功能	
强大的管理工具	强大的管理工具
除管理工具和日志功能外，还提供基于消息的监视 API	管理工具和日志功能，但不提供基于消息的监视 API。
可调节的性能	可调节的性能
灵活的服务器配置功能	
可插入的持久性	可插入的持久性
LDAP 服务器支持	LDAP 服务器支持

下面介绍平台版与企业版的许可证功能。

企业版

通过使用 Message Queue 企业版，您可以在企业生产环境中部署和运行消息传送应用程序。您还可以使用它来开发和调试消息传送应用程序和组件，并对其进行负荷测试。基于所使用的 CPU 数量，企业版的许可证有效期是无限的。其许可证对多代理消息服务中的代理数量没有限制。

平台版

Message Queue 平台版对消息服务器所支持的客户机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**的有效期是无限的。在要求不太苛刻的生产环境中，可以将具有基本许可证的平台版用作 JMS 提供者。该许可证不包含企业版功能。
- **90 天试用企业版许可证**包括基本许可证中不包括的所有企业版功能。但是，软件将此许可证的有效期强制限制为 90 天，因此，此许可证适用于评估企业版中所提供的企业功能。有关使用 90 天试用企业许可证的说明，请参见 Message Queue 管理指南中阐述的启动选项。

可以从 Sun 的 Web 站点免费下载平台版，它还附带有 Sun Java System Application Server 平台。 Message Queue Installation Guide 中提供了有关从平台版 Message Queue 升级到企业版的说明。

注 对于所有 Message Queue 版本，可以将该产品的一部分 （Message Queue 客户机运行时）作为商业用途而免费再分发。不能再分发产品中的所有其他文件。被许可方可以利用能够免费再分发的部分来开发 Message Queue 客户机应用程序 （可以连接到 Message Queue 服务的应用程序），并可将其出售给第三方而不必支付任何许可费用。第三方需要购买 Message Queue 来访问 Message Queue 消息服务器，或者连接到已安装并正在运行 Message Queue 消息服务器的另一方。

Sun 产品环境下的 Message Queue

除作为应用程序直接使用的中间件外，其他中间件以及 Sun 提供的其他服务器和应用程序也可以使用 Message Queue。为方便对它的使用，在 Solaris 和 Java Enterprise System 以及 Sun Java System Application Server 中均提供了 Message Queue。

在 Application Server 中， Message Queue 满足了 JMS 要求 （J2EE 平台必须提供 JMS 提供者）。由 Application Server 托管的应用程序可以直接使用它。有关详细信息，请参见第 83 页第 6 章 “Message Queue 和 J2EE”。

第 3 章

可靠消息传送

本章介绍 Message Queue 服务如何提供可靠的消息传送。它描述了消息在系统内的传送路径，并介绍用来将消息路由和传送到相应消费者及确保消息得到传送的各种机制。

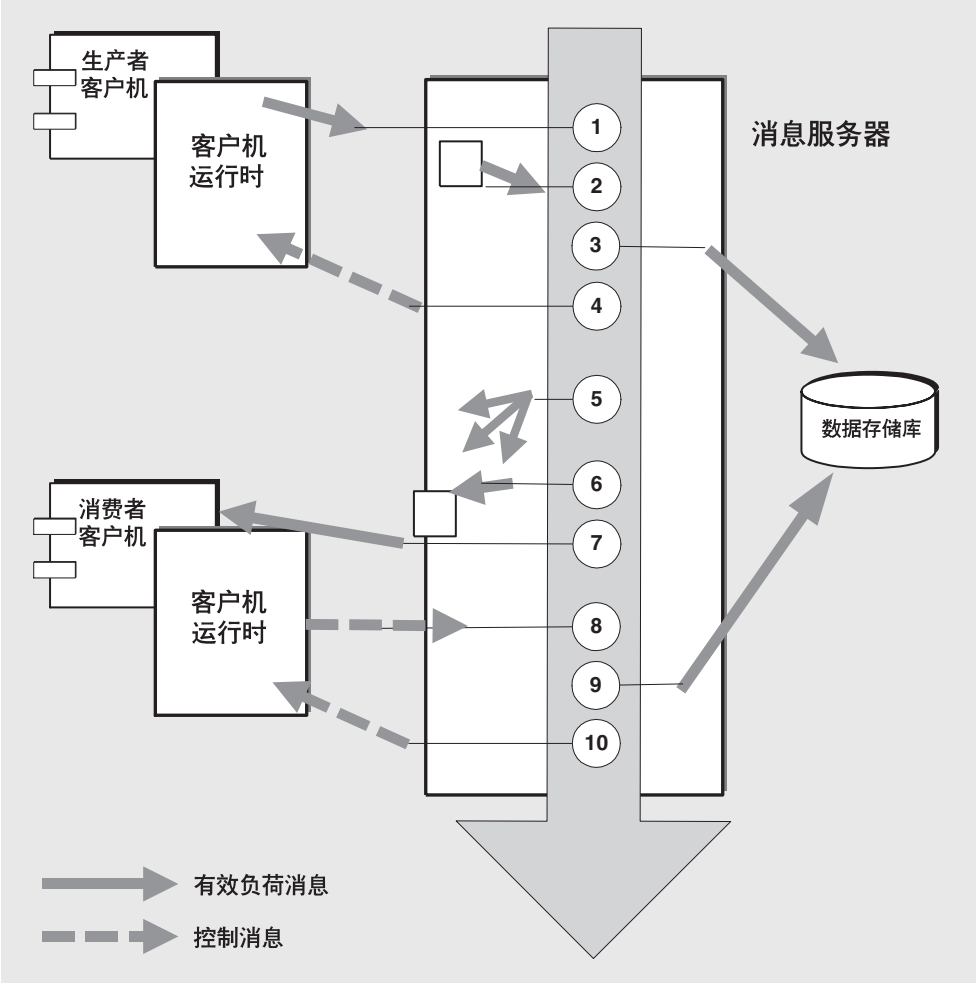
本章涵盖以下主题：

开发者和管理员都会对本章的内容感兴趣，它是对第 2 章 “Message Queue 简介”中信息的补充。

消息在系统中的传送路线

图 3-1 中简要说明了 Message Queue 消息服务将消息从消息生产者传送到消息消费者的过程。以下各小节对传送过程的每个阶段进行了更详细的说明。

图 3-1 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. 客户机运行时通过连接将消息从消息生产者传送到消息服务器。

消息处理和路由

2. 消息服务器从连接读入消息，然后将其置于相应的目标中。
3. 消息服务器将 （持久性）消息置于数据存储库中。
4. 消息服务器向消息生产者的客户机运行时确认已收到消息。
5. 消息服务器确定消息的路由。
6. 消息服务器将消息从其目标写出到适当的连接。

消息使用

7. 消息消费者的客户机运行时将消息从连接传送到消息消费者。
8. 消息消费者的客户机运行时向消息服务器确认消息的使用。

消息生命周期结束

9. 消息服务器处理客户机确认，将 （持久性）消息从其目标和数据存储库中删除。
10. 消息服务器向消费者的客户机运行时确认，告知客户机确认已得到处理，不能再次传送此消息了。

在这些传送步骤中，系统处理的消息分为以下两类：

- **有效负荷消息**。生产者客户机发送给消费者客户机的 JMS 消息。
- **控制消息**。消息服务器与客户机运行时之间传递的确认及其他非有效负荷消息，用于确保成功传送有效负荷消息并使连接中的消息流得到控制。

消息传送处理

在将消息从生产者传送到消费者的过程中， Message Queue 服务对消息的处理分几个阶段进行，如图 3-1 后面的步骤说明所示。

这些阶段如下：

- 第 52 页 “消息生成”
- 第 52 页 “消息处理和路由”
- 第 54 页 “消息使用”

- 第 56 页 “消息生命周期结束”

以下小节中将逐一介绍这些阶段。

消息生成

在消息生成过程中，由客户机创建消息，然后由客户机运行时通过连接将消息发送至代理中的目标。

如果已将消息的传送模式设置为持久性 （有保证的传送，传送且只传送一次，即便代理发生故障），则默认情况下，代理将控制消息 （代理确认）发送回客户机运行时。该代理确认指示代理已将消息传送到其目标，并将其存储在代理的数据存储库中。客户机线程将阻塞，直到它收到代理确认为止。

如果将消息的传送模式设置为非持久性，则默认情况下，代理不会将代理确认发送回客户机运行时，客户机线程不会阻塞。不过，如果了解代理是否收到非持久性消息很重要，则可以启用代理确认。实际上，必须启用代理确认才能使代理在达到目标内存限制时降低消息生成速度 （请参见第 66 页 “目标消息限制”）。

消息处理和路由

代理收到外来的 JMS 有效负荷消息时，会将其置于它的目标中，然后将其路由到相应的一个或多个消费者。

通常，所有消息始终保留在它们的物理目标 （内存中）中，直到被传送或过期。如果代理出现故障，这些消息将会丢失。如果消息是持久性的，代理会将其存储在数据库或文件系统中，并在发生故障后对其进行恢复。

消息的处理方式取决于其目标类型，即队列目标或主题目标，以下各节对二者进行了说明。此外，还取决于管理员创建物理目标时为目标设置的属性。

队列目标

队列目标用于点对点消息传送；在这种消息传送方式中，消息只传送给一个消费者且只供该消费者使用。

虽然队列中的任何消息仅传送到单个消费者，但 Message Queue 允许在队列中注册多个消费者。然后，代理可将消息分布至其他注册消费者，由这些消费者平衡负荷。

基本路由机制

当来自生产者的消息到达时，即被加入队列。当每条消息到达队列前端时，就会将其路由到已在队列中注册的某个消费者。消息到达队列前端的顺序取决于它们进入队列的顺序及其优先级。

如果在消息中设置了选择器属性值，则代理会将该值与已注册的消费者所指定的任何选择器值进行比较，确保两者匹配，然后再将消息路由到消费者。

多个消费者的队列传送

对多个消费者的队列传送的实现是根据多个队列目标属性使用可配置的负荷平衡方法：

- 可以设置负荷平衡队列传送中处于活动状态的消费者的最大数量。
- 可以设置在任何活动消费者失败时替代它们的备份消费者的最大数量。

如果消费者数量超过这两个属性的和，将拒绝新的消费者。（Message Queue Platform Edition 支持每个队列多达 3 个消费者（2 个活动和 1 个备份），而 Message Queue Enterprise Edition 支持无限个消费者。）

负荷平衡机制考虑了不同消费者的消息使用率。队列目标中的消息按可配置大小的批次（队列目标的消费者流限制属性）路由至新的可用活动消费者（以便在队列中注册）。在传送这些消息后，到达队列的其他消息将在消费者可用时按批次路由至消费者。当消费者所使用的消息数已达到先前传送给它的消息的某个百分比（可以对其进行配置）时，该消费者即变为可用。换句话说，每个消费者的发送率取决于消费者的当前容量和消息处理速率。

如果某个活动消费者失败，则第一个备份消费者变为活动消费者，并接管失败消费者的工作。由于这些机制，当队列目标中有多于一个的活动消费者时，消息使用的顺序是不一定的。

当消息生成速率较低时，代理可在活动消费者之间不平衡地发送消息。如果活动消费者多于所需个数，其中的一些可能永远收不到消息。

在代理群集环境中，对多个消费者的传送可设置为优先考虑本地消费者。可以使用队列目标属性，指定仅当生产者的主代理（即，生产者将消息发送到的代理，本地代理）中无消费者时，才能将消息传送到远程消费者。这可在路由到远程消费者（通过**其**主代理）可能导致流通量降低时提升性能。

主题目标

主题目标用于发布 / 订阅消息传送，在该消息传送中，消息将被传送到所有已在目标中注册请求的消费者。

基本路由机制

当来自生产者的消息到达时，它将被路由到所有订阅该主题的消费者。如果消费者已注册了该主题的长期订阅，则当消息到达时，它们不必处于活动状态以接收消息：代理将存储该消息，直到消费者再次变为活动状态为止，然后传送该消息。

如果在消息中设置了选择器属性值，代理会将该值与注册的消费者所指定的任何选择器值进行比较，确保两者匹配，然后再将消息路由到消费者。

长期订阅和客户机标识符

只有一位用户可以建立对某主题的长期订阅。在该用户打开和关闭与消息服务器的连接时，其标识必须保持不变。**客户机标识符**用于确保每个长期订阅只对应一位用户。

客户机标识符在客户机的连接与消息服务器间建立关联，由消息服务器代表客户机来维护状态信息。按照定义，客户机标识符是唯一的。

要创建长期订阅，客户机标识符必须由客户机使用 JMS API 以编写方式进行设置，或者在客户机使用的连接工厂对象中以管理方式进行配置。

消息使用

路由消息后，即会将其传送给相应的消费者。当消费者收到有效负荷消息时，消费者客户机运行时向代理发送一个确认，说明客户机已收到消息并对消息进行了处理。代理等待该客户机确认，确认后才会将消息从其目标中删除。客户机确认可以应用于个别消息、消息组或事务。

客户机确认

按照 JMS 规范，客户机可以在创建会话时指定三种基本确认模式中的一种。选择哪一种模式取决于所需的消息传送可靠性：

Message Queue 通过增加 `NO_ACKNOWLDEGE` 模式，扩展了客户机确认模式集。以下小节对这些基本及扩展模式进行了说明。

AUTO_ACKNOWLEDGE 模式

在 `AUTO_ACKNOWLEDGE` 模式下，会话自动确认客户机使用的每条消息。此外，会话线程会阻塞，等待代理确认它已处理了每个已使用消息的客户机确认。而该确认称为“代理确认”。

- 对于由消息侦听器执行的异步消息使用，消息在返回 `onmessage()` 方法后被确认。

- 对于同步消息使用，消息恰好在 receive() 方法返回前被确认。在这种情况下，可能只对消息进行部分处理；如果在使用消息前系统发生了故障，则消息就会丢失。为提高可靠性，可以使用 CLIENT_ACKNOWLEDGE 模式或事务会话来确保系统发生故障时消息不会丢失。

CLIENT_ACKNOWLEDGE 模式

CLIENT_ACKNOWLEDGE 模式授予客户机最高的控制权限。在此模式下，客户机在使用一条或多条消息后显式对其进行确认。确认发生在客户机调用消息对象的 acknowledge() 方法时，会导致会话确认自上次调用该方法以来它使用的所有消息。（其中可能包括会话中许多不同的消息侦听器以异步方式使用的消息，与使用这些消息的**顺序**无关。）

此外，会话线程会阻塞，等待为已使用的一批消息返回代理确认，它确定代理已对客户机确认进行了处理。

由于通常按批次发送客户机确认和代理确认（而非逐个发送），因此，与 AUTO_ACKNOWLEDGE 模式相比，CLIENT_ACKNOWLEDGE 模式通常节省连接带宽，并减少代理确认的开销。当然，如果在此模式下客户机对每条消息逐一进行确认，则不会按批次发送确认，而是逐一地发送确认。

注 Message Queue 还提供了一种可以在 CLIENT_ACKNOWLEDGE 模式下使用的特定方法，可以利用该方法，只确认调用了该方法（而非标准行为）的**个别**消息。这是使用 Message Queue Developer's Guide for Java Clients 中介绍的编程技术实现的。

DUPS_OK_ACKNOWLEDGE 模式

在 DUPS_OK_ACKNOWLEDGE 模式下，会话在使用了十条消息后进行确认。目前无法对该值进行配置。与 AUTO_ACKNOWLEDGE 或 CLIENT_ACKNOWLEDGE 模式不同的是，会话线程不会阻塞等待代理确认，因为在 DUPS_OK_ACKNOWLEDGE 模式下不请求任何代理确认。

这意味着，无法保证只对消息传送和使用一次。一般来说，不会非常频繁地重新传送消息；这种情况只会在发生故障（代理未收到它所传送的消息的客户机确认）时出现。如果客户机对重复传送并不在意，则应使用 DUPS_OK_ACKNOWLEDGE 模式。

由于客户机确认按批次发送且客户机线程不会阻塞，因此消息吞吐量通常远高于其他模式。

NO_ACKNOWLEDGE 模式

在 NO_ACKNOWLEDGE 模式下，代理会代表客户机进行客户机确认，因此不能保证消费者客户机已成功地对消息进行了处理。

如果消息吞吐量很重要，而传送的可靠性并不是很重要，则请使用此模式。例如，可能会出现以下情况：定期发送消息的时间间隔很短，因而消息负荷很高，此时丢失一些消息不会有太大的影响。

此模式扩展了 JMS 规范，只应由不需要与其他 JMS 提供者配合使用的客户机使用。

事务

上述客户机和代理的确认过程同样适用于编组为事务的 JMS 消息传送。在此类情况下，客户机和代理确认在事务 （包括事务中包含的所有消息）级别进行。当提交事务时，将自动发送代理确认。

代理会跟踪事务，使它们可以在出现故障时进行提交或回滚。该事务管理也支持本地事务 （较大的分布式事务的一部分，请参见第 28 页 “分布式事务”）。代理会一直跟踪这些事务的状态，直到它们被提交。当代理启动时，它会检查所有未提交的事务，在默认情况下，回滚未处于 PREPARED 状态的所有事务 （必须手动解决）。

Message Queue 通过 XA 连接工厂支持分布式事务。XA 连接工厂使您可以创建 XA 连接，XA 连接又可以使您创建 XA 会话。另外，要支持分布式事务还需第三方 Java 事务服务 (JTS) 或提供 JTS 的 J2EE 兼容应用服务器。

消息生命周期结束

在将消息成功传送后，代理会将其从目标内存中删除。不过，有时在未成功传送消息的情况下，可能会将其放弃。以下小节介绍在哪些条件下会将消息放弃。

正常的消息删除

正常情况下，在成功传送消息 （通过客户机确认进行确认）后，代理就会将消息从目标内存中删除。

当代理将消息传送给消费者时，它就会将消息标记为已传送，但它实际上并不知道是否已收到和使用消息。因此，代理等待客户机确认后，再将消息从其物理目标和持久性存储中删除。

如果将消息发送到某个主题，代理从它将消息传送到的每个消息消费者收到客户机确认后，才会将消息删除。对于某个主题的长期订户，代理会将每条消息保留在该目标中，以便在每个长期订户变为活动消费者时将消息传送出去。代理会在收到客户机确认时进行记录，并且只有在收到所有确认后才会删除消息 （除非在这之前消息已过期）。根据客户机确认模式的不同，代理可能会通过将代理确认发送回客户机来确认已收到客户机确认。

如果代理或连接发生故障，代理可能会收不到客户机确认，并重新传送所有先前已传送但未得到确认的消息，并为其标记已重新传送标志。例如，如果队列消费者在确认收到消息之前脱机，随后另一位消费者 （或者甚至是同一消费者）在队列中进行了注册，则代理会向该新消费者重新传送未确认的消息，并为其标记已重新传送标志。关注消息在此类情况下重新传送的客户机应用程序应检查消息是否有该标志。

注 客户机可以通过一种 JMS API (恢复会话)，明确请求重新传送客户机已收到但尚未确认的消息。重新传送此类消息时，代理会为它们标记已重新传送标志。

异常的消息删除

消息无法传送时，系统会将其放弃或置于停用消息队列中，具体情况取决于阻止其传送的因素。

在下列情况下，消息在尚未成功传送和使用的情况下将被代理放弃：

- 使用管理工具清除目标中的一条或多条消息
- 删除或重新定义了长期订阅，因而使主题目标中的消息无法传送。

不过，在以下情况下，将消息视为停用，并将其放弃或置于停用消息队列中，具体情况取决于所配置的行为：

- 消息过期，超出了在其消息标题中设置的 `JMSExpiration` 值。
- 因目标超出了内存限制阈值而调用了目标的 “删除”限制行为。
- 消息传送因消息无法使用 （客户机引发异常）而失败。

可以选择保留此类消息，并将其置于停用消息队列中。将消息置于停用消息队列中时，代理会向消息中写入特定于 Message Queue 的属性值，指定放置的时间和原因。

以后，可以从停用消息队列中检索消息以便进行诊断。有关详细信息，请参见第 65 页 “停用消息队列”。

性能问题

消息传送的可靠性越高，需要的开销和带宽就越多。因此，性能和可靠性之间的折衷是设计时要重点考虑的一个方面。您可以选择生成和使用非持久性消息来获得最佳**性能**。另一方面，您可以通过生成和使用持久性消息并使用事务会话来获得最佳可靠性。在以上两种极端情况之间存在若干种选择，具体情况取决于各应用程序的需要。

例如，消息处理速率受许多因素的影响，其中包括消息传送应用程序设计、消息服务器配置和客户机运行时配置。虽然这些因素差异很大，但它们之间存在的交互可能会增加最大限度提高性能这一任务的复杂性。

本节简要介绍其中的几个因素，它们与可靠性和性能折衷有关。

传送模式 传送模式指定是将消息至多传送一次（非持久性消息）还是传送且只传送一次（持久性消息）。要管理持久性消息，您需要使用通过连接传送的代理确认消息，还需要使用客户机确认模式，它会阻塞以等待接收代理确认。要增加吞吐量，可以将客户机运行时设置为抑制代理确认，但这样做将无法保证持久性消息被传送且只传送一次。

客户机确认模式 四种客户机确认模式需要不同级别的处理和带宽开销。AUTO_ACKNOWLEDGE 模式的开销最大，可以保证消息逐条传送的可靠性；CLIENT_ACKNOWLEDGE 模式按批次发送确认，因此需要的带宽开销较小；而 DUPS_OK_ACKNOWLEDGE 模式的开销最小，但允许重复传送消息。NO_ACKNOWLEDGE 模式可以提供最佳性能，代价是可能会丢失消息。

客户机应用程序设计 在会话中排队等候的消息数是使用该会话的消息消费者数量以及每个消费者的消息负荷的函数。如果客户机生成或使用消息的速度很慢，您通常可以通过下列操作提高性能：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产者和消费者，或者在更大数量的连接之间分布会话。Message Queue Developer's Guide for Java Clients 和 Message Queue Developer's Guide for C Clients 中介绍了影响性能的设计问题。

消息流测量 客户机运行时可以管理控制消息和有效负荷消息争用连接带宽的问题。通过对客户机运行时进行适当配置，可以帮助提高代理确认的传送速度，从而释放阻塞的会话线程并提高消息的使用速度。有关详细信息，请参见 Message Queue 管理指南。

消息流限制 接近客户机运行时资源极限时，消息使用速度可能会下降。通过限制客户机运行时中保留的消息（等待被一个或多个消费者使用）的数量，可以避免达到这些资源极限。有关详细信息，请参见 Message Queue 管理指南。

第 4 章

消息服务器

第 33 页 “消息服务器”中介绍了 Message Queue 消息服务器，它由单个**代理**或一组协同工作的代理 （代理群集）组成，用于执行消息路由和传送。

本章介绍代理的内部结构及其各种组件，并简要说明在开发及生产环境中配置和管理代理需要执行的步骤。本章包含以下小节：

- 第 60 页 “代理体系结构”
- 第 61 页 “代理组件”
- 第 72 页 “开发和生产环境”

了解代理的功能组成部分有助于您配置所需的代理功能，调整操作以及优化性能。因此，本章内容可能更合乎管理员而非应用程序开发者的需要。

代理体系结构

为进行消息传送，代理会建立与客户机之间的通信通道、进行验证和授权、正确路由消息、确保可靠传送以及提供用于监视系统性能的数据。

为了实现这么多复杂的功能，代理使用了大量的各种内部组件，每个组件都在传送过程中担当着各自不同的角色。这些代理组件如图 4-1 所示，表 4-1 则对这些组件进行了简要说明。其中消息路由器组件执行关键的消息路由和传送服务，而其他组件则提供消息路由器所依赖的重要的支持服务。

图 4-1 代理组件

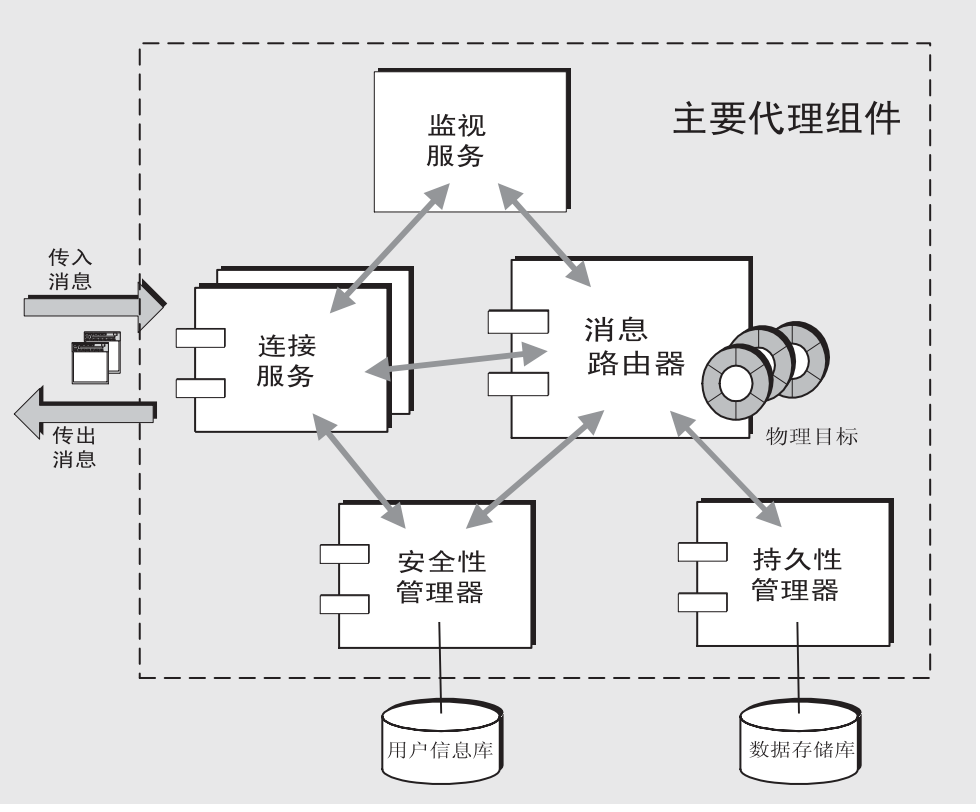


表 4-1 主要代理服务组件及功能

组件	说明 / 功能
连接服务	管理代理和客户机之间的物理连接，用于传输传入和传出消息。
消息路由器	管理消息的路由和传送，这里的消息包括：JMS 消息以及在 Message Queue 消息传送系统中用于支持 JMS 消息传送的控制消息。
持久性管理器	管理向持久性存储器写入数据，这样即使系统发生故障也不会导致无法传送 JMS 消息。
安全性管理器	为请求连接到代理的用户提供验证服务，并为通过验证的用户提供授权服务（访问控制）。
监视服务	生成度量和诊断信息，这些信息可被写入许多用来监视和管理代理的输出通道中。

在配置代理时，实际上是配置这些服务，以便根据负荷条件、应用程序复杂性等因素来优化代理的性能。

代理组件

以下几节介绍图 4-1 中显示的各个代理组件及其功能和行为。有关这些组件的相应属性和配置过程，请参见 Message Queue 管理指南。

连接服务

Message Queue 代理支持与应用程序客户机和管理客户机的通信。每个连接服务由其服务类型和协议类型指定。

服务类型 指定服务提供的是 JMS 消息传送服务 (NORMAL)，还是支持管理工具的 Message Queue 管理服务 (ADMIN)。

协议类型 指定支持服务的下层传输协议层。

表 4-2 显示了 Message Queue 代理当前支持的连接服务。

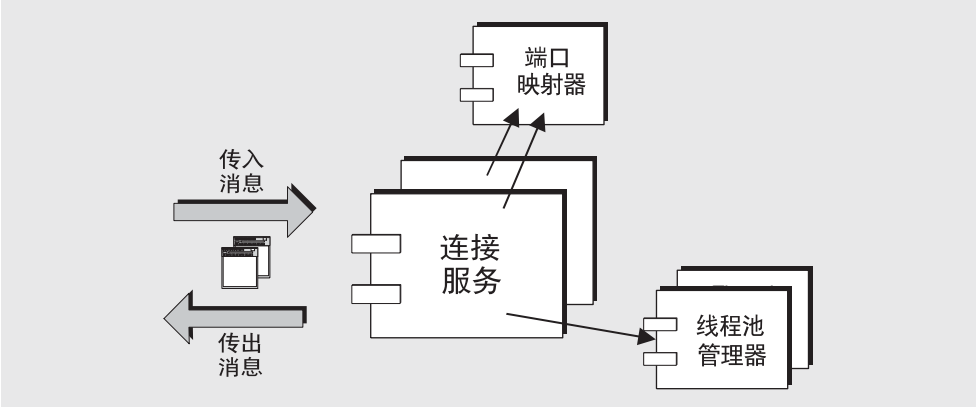
表 4-2 代理支持的连接服务

服务名称	服务类型	协议类型
jms	NORMAL	TCP
ssljms	NORMAL	TLS（其安全性基于 SSL）
httpjms（企业版）	NORMAL	HTTP
httpsjms（企业版）	NORMAL	HTTPS（其安全性基于 SSL）
admin	ADMIN	TCP
ssladmin	ADMIN	TLS（其安全性基于 SSL）

可以将代理配置为运行以上任一或全部连接服务。每个连接服务均支持特定的验证和授权（访问控制）功能（请参见第 69 页 “安全性管理器”）。每个连接服务都是多线程的，从而支持多个连接。

每个连接服务仅在由代理的主机名和端口号指定的特定端口上可用。端口可动态分配，或者可指定连接服务可用的端口。图 4-2 中显示了常规模式。

图 4-2 连接服务支持



端口映射器

连接服务由公共的**端口映射器**来分配端口。端口映射器本身位于标准端口号 7676。当 Message Queue 客户机运行时建立与代理的连接时，首先与端口映射器进行通信，请求其所需的连接服务端口号。

可以通过在配置 jms、ssljms、admin 和 ssladmin 连接服务时为其分配**静态**端口号的方式忽略端口映射器的分配。不过，通常仅在特殊情况下使用静态端口号，例如，通过防火墙建立连接（请参见第 63 页 “HTTP/HTTPS 支持”），一般不建议使用静态端口号。

线程池管理器

每个连接服务都是多线程的，从而支持多个连接。这些连接所需的线程在线程池中进行维护，而线程池又由**线程池管理器**组件来管理。通过配置线程池管理器，可以设置线程池中维护的线程的最大和最小数量。当连接需要线程时，这些线程将被添加到线程池中。当线程数量超过最小数量时，系统将关闭变为空闲状态的线程，直到达到最小阈值，以此来节省内存资源。线程池中的线程可专用于单个连接，也可根据需要分配到多个连接。

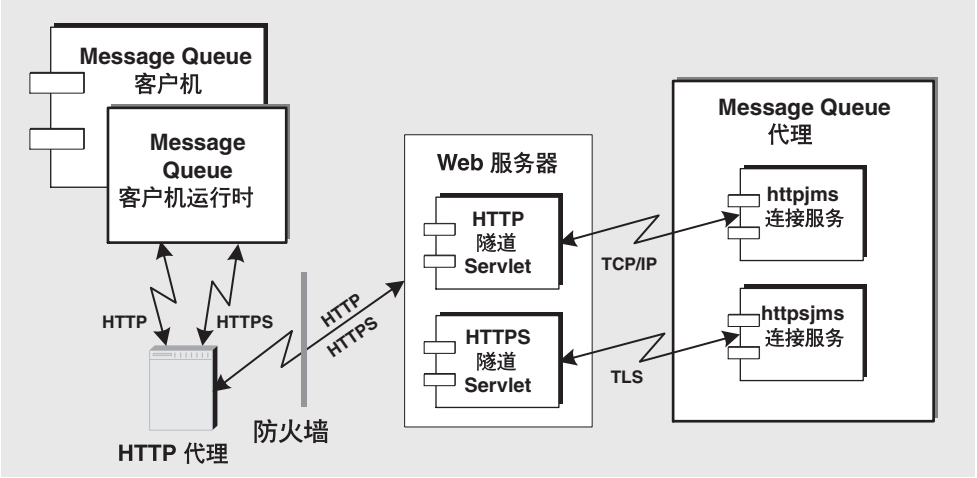
HTTP/HTTPS 支持

HTTP/HTTPS 支持允许 Message Queue 客户机使用 HTTP 协议（而非通过直接 TCP 连接）与代理进行交互。如果需要使用防火墙将客户机与代理分隔，则可以使用 HTTP/HTTPS 服务，因为它允许通过防火墙进行通信。

注 HTTP/HTTPS 支持可用于 Java 客户机，但不能用于 C 客户机。

图 4-3 显示了提供 HTTP/HTTPS 支持所需的主要组件。

图 4-3 HTTP/HTTPS 支持体系结构



- 在 Message Queue 客户端， HTTP 或 HTTPS 传输驱动程序将 JMS 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。
- 必要时，客户机可以选择使用 HTTP 代理服务器与 Web 服务器进行交互。
- HTTP 或 HTTPS 隧道 Servlet （均随 Message Queue 一起提供）被装入 Web 服务器，并在将 JMS 消息转发给代理之前，从客户机 HTTP 请求中提取出 JMS 消息。 HTTP/HTTPS 隧道 Servlet 还将代理响应发送回客户机。可以使用一个 HTTP/HTTPS 隧道 Servlet 访问多个代理。
- 在代理端，代理启动时将建立其与隧道 Servlet 的连接。在从 HTTP 或 HTTPS 隧道 Servlet 发送消息时，`httpjms` 或 `httpsjms` 连接服务相应地解开消息，并将其提交到代理的消息路由器组件。

图 4-3 中所示的 HTTP 与 HTTPS 的体系结构非常相似。主要区别在于， HTTPS （`httpsjms` 连接服务）中的隧道 Servlet 与客户机和代理之间的连接都是安全的。

Message Queue 的 HTTPS 隧道 Servlet 将自签名证书传递给请求连接的任何代理。代理使用该证书建立与 HTTPS 隧道 Servlet 的加密连接。建立此连接后，就可以协商建立 Message Queue 客户机和隧道 Servlet 之间的安全连接。

可以使用 Message Queue 管理指南中介绍的属性来配置 `httpjms` 和 `httpsjms` 服务。

消息路由器

使用支持的连接服务在客户机与代理之间建立起连接后，即可进行消息的路由和传送。

Message Queue 消息传送基于两个传送阶段：首先，从生产者客户机传送到代理上的物理目标，然后再从代理上的目标传送到一个或多个消费者客户机。消息路由器管理这一过程：将到达的消息置于相应的目标上，然后再将消息路由并传送到相应消费者。

本节介绍不同种类的目标以及按单独及统一方式对这些目标的内存资源进行管理。第 3 章 “可靠消息传送”中介绍了消息的路由和传送机制。

物理目标

物理目标表示代理的物理内存中的位置 （传入消息在被路由到消费者客户机前将存储在其中）。

可按创建方式和创建目的将目标划分为以下几个类别：管理员创建、自动创建、临时及停用消息队列。

管理员创建的目标

管理员创建的目标是管理员使用 Message Queue 管理工具创建的。这些目标对应于以编程方式创建的逻辑目标，或者对应于客户机查找的目标受管理对象。

Message Queue 消息服务器是消息传送系统的核心，因此其性能和可靠性对企业应用程序的正常运行至关重要。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消息消费者的数量和长期性），所以需要对它们严格地进行管理，以确保消息服务器的性能和可靠性。因此为应用程序创建目标、监视目标以及根据需要重新配置它们的资源要求已成为管理员的标准做法。

自动创建的目标

自动创建的目标是在需要时由代理自动创建的，不需要管理员进行干预。特别是，每当消息消费者或消息生产者尝试访问不存在的目标时，就会创建一个自动创建的目标。在需要动态创建目标时，就会使用这些目标：通常是在开发和测试阶段。可以配置代理以启用或禁用**自动创建**功能。

当自动创建目标时，可能会导致不同客户机应用程序 （使用相同的目标名称）之间发生冲突，或者导致系统性能降低（由于支持目标需要使用资源）。因此，当不再使用自动创建的目标时，也就是当目标不再与消息消费者客户机保持连接且不再包含任何消息时，代理会自动将其销毁。重新启动代理后，只有当自动创建的目标包含持久性消息时，才会重新创建这些目标。

临时目标

某些客户机需要一个目标来接收对发送至其他客户机的消息的回复。这些客户机应用程序使用 JMS API 明确创建和销毁的目标称为临时目标。这些目标是为连接而创建的，并由代理维护，而且仅在连接期间存在。临时目标无法由管理员销毁，而且只要此目标在使用中（即连接有活动的消息消费者），那么它也无法由客户机应用程序销毁。临时目标与管理员创建或自动创建的目标不同（后两者包含持久性消息），它们不会被持久保存，并且在重新启动代理时也不会重新创建临时目标；但是，可通过 Message Queue 管理工具看到这些内容。

停用消息队列

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，以便进行诊断。**停用消息**是指由于非正常处理或显式管理操作原因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、因超出内存限制而被从目标中删除或传送尝试失败。

可以使用两种方法，将消息置于停用消息队列中：

- 可以将目标配置为将消息置于停用消息队列中，而不是将其废弃。
- 客户机开发者也可以在创建消息时设置一个属性值，以确定停用该消息时，是否应将其置于停用消息队列中。

消息被置于停用消息队列时，其他属性信息也将写入到消息中，从而为您提供有关停用原因的信息。

内存资源管理

消息服务器在资源方面很有限：内存、 CPU 周期等。因此，根据代理所支持的消息传送应用程序的使用模式，消息服务器可能会在无响应或不稳定的位置发生崩溃。特别是，当目标的消息生成速度远远高于消息使用速度时，就可能会出现该问题。

消息路由器具有管理内存资源并防止代理用尽内存的机制。它使用三级内存保护，使系统在资源不足时仍可正常运行：目标消息限制、系统范围限制以及系统内存阈值。

目标消息限制

由于消息可以在目标中保留相当长的一段时间，因此内存资源可能会成为一个问题。为目标分配内存时，您一定希望分配得恰到好处，向一个目标分配过多内存会导致内存不足，而太少的话消息就会被拒绝。

为了根据各个目标的负荷要求实现灵活性，可以设置一些属性来管理它们各自的内存资源和消息流。例如，可指定目标允许的最多生产者数量、目标允许的最多消息数（或最大消息量）以及任何一条消息大小的最大值。

还可以指定在达到任何此类限制时，消息路由器作出以下四种响应中的哪一种。四种限制行为是：

- 减慢消息生产者
- 丢弃最旧的消息
- 根据消息的存在时间丢弃优先级最低的消息
- 拒绝最新的消息

系统范围消息限制

系统范围消息限制构成了第二道防线。可指定应用于代理中所有目标的系统范围限制：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，消息路由器将拒绝新消息。

系统内存阈值

系统内存阈值是第三道防线。可指定可用系统内存的阈值，代理可利用阈值采取更为严肃的操作以防内存过载。采取的操作取决于内存资源的状态：绿色 （可用内存充足）、黄色（代理内存不足）、橙色（代理内存严重不足）、红色（代理无可用内存）。随着代理的内存状态由绿色变为黄色，再变为橙色，最后变为红色，代理所采取的措施也会越来越严格，操作类型如下：

- 丢弃数据存储库中持久性消息在内存中的副本。

- 限制非持久性消息的生产者，最终甚至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：每条消息必需由代理确认。

这些措施都会降低性能。

如果达到系统内存的阈值，则说明没有适当地设置目标到目标的消息限制以及系统范围消息限制。在某些情况下，阈值不可能及时地捕捉到潜在的内存过载。因此，您不应依赖此功能，而应该分别配置目标，共同优化内存资源。

通过仔细地监视和调整，这些基于目标的限制和行为可以用于平衡消息的内流和外流，以免发生系统过载。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成性。

持久性管理器

对于发生故障后待恢复的代理，需要重新创建其消息传送操作的状态。这需要它将所有持久性消息以及基本的路由和传送信息保存到数据存储库中。要进行恢复，代理还必须完成以下操作：

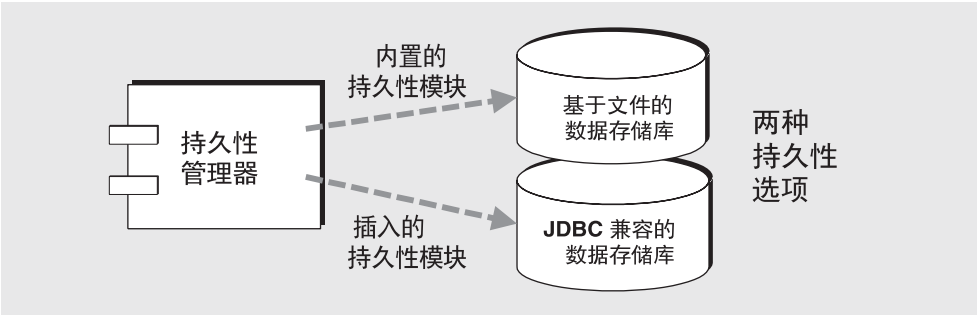
- 重新创建目标
- 恢复每个主题的长期订阅列表
- 恢复每条消息的确认列表
- 重新生成所有已提交事务的状态

持久性管理器用于管理所有这些状态信息的存储和检索。

代理重新启动时，会使用由持久性管理器管理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，并为未传送的消息重新创建其路由表。然后代理才可以恢复消息传送。

Message Queue 既支持内置的持久性模块，也支持插入的持久性模块（请参见图 4-4）。内置的持久性是基于文件的数据存储库。插入的持久性使用 Java 数据库连接 (JDBC™) 接口，并需要 JDBC 兼容的数据存储库。通常，内置的持久性比插入的持久性速度更快，但某些用户更希望获得与 JDBC 兼容的数据库系统提供的冗余和管理控制。

图 4-4 持久性管理器支持



内置的持久性 默认的 Message Queue 持久性存储器解决方案是基于文件的数据存储库，它使用单个文件来存储持久性数据。为减少添加和删除消息带来的碎片，可压缩基于文件的数据存储库。要使可靠性最大化，可指定持久性操作使用物理存储设备与内存中状态同步。这有助于消除因系统崩溃而导致的数据丢失，但会使性能下降。由于数据存储库中可能包含敏感或专用消息，因此应对数据存储库文件进行保护以防止未授权的访问。

插入的持久性 您可以代理进行设置，以访问可通过 JDBC 驱动程序访问的任何数据存储库。这将涉及到设置多个与 JDBC 相关的代理配置属性以及使用 Message Queue 数据库管理器实用程序来创建具有相应模式的数据存储库。Message Queue 管理指南中详细说明了上述过程和相关的配置属性。

安全性管理器

Message Queue 提供验证和授权 （访问控制）功能，同时也支持加密功能：

- **验证**可以确保只有通过验证的用户才能与消息服务器建立连接。
- **验证**指定哪些用户有权访问连接服务或目标等资源以执行消息服务所支持的特定操作。
- **加密**可以防止消息在通过连接传送时被篡改。

验证和授权功能取决于用户系统信息库（请参见第 70 页图 4-5）：包含消息传送系统的用户信息 （用户名、密码和**组**成员资格）的文件、目录或数据库。用户名和密码用于在请求连接至代理时对用户进行验证。用户名和组成员资格 （与访问控制文件配套使用）用于授权操作，例如为目标生成消息或使用目标的消息。

可以填充 Message Queue 提供的用户系统信息库，也可以将原有的 LDAP 用户系统信息库插入到代理中。平面文件用户系统信息库易于使用，但也容易受到安全性攻击，因此仅用于测试和开发目的。LDAP 用户系统信息库比较安全，因此最适合用于生产目的。

以下各小节介绍了验证、授权和加密。有关更详细信息，请参见 Message Queue 管理指南。

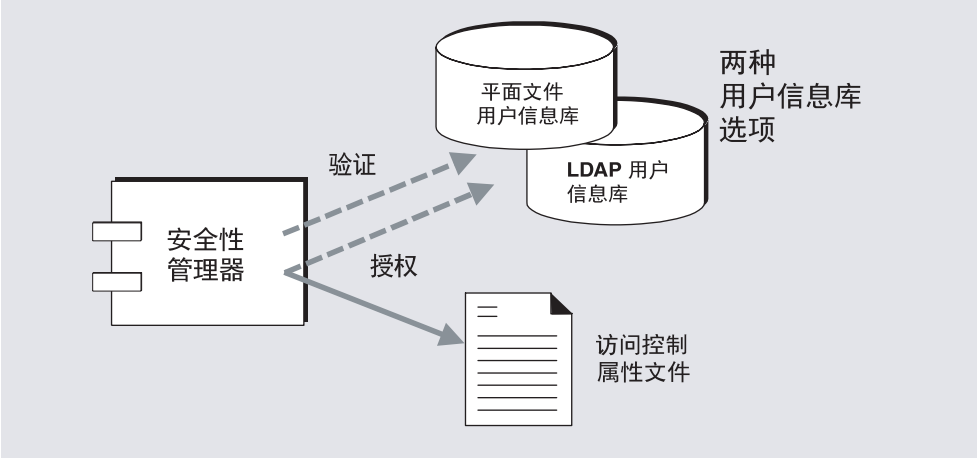
验证

Message Queue 安全性支持基于密码的验证。当客户机请求连接到代理时，该客户机必须提供用户名和密码。安全性管理器将把客户机提交的用户名和密码与用户系统信息库中存储的用户名和密码进行比较。密码在从客户机传送到代理的过程中，将使用 Base -64 编码或消息摘要 (MD5) 进行编码。要想获得更安全的传送，请参见第 70 页 “加密”。可以分别配置每个连接服务使用的编码的类型，也可以基于代理范围设置编码方式。

授权

客户机应用程序的用户通过验证后，即可以获得授权来执行各种 Message Queue 相关的活动。安全性管理器既支持基于用户的访问控制，也支持基于组的访问控制：根据用户在用户系统信息库中分配到的用户名或组的不同，该用户所拥有的执行特定 Message Queue 操作的权限也不同。可以在访问控制属性文件中指定访问控制 （请参见图 4-5）。

图 4-5 安全性管理器支持



当用户尝试执行某个操作时，安全性管理器将检查该用户的用户名和组成员资格（在用户系统信息库中）是否属于被授予访问该操作权限的用户名或组成员资格 （在访问控制属性文件中）。访问控制属性文件指定了执行以下操作的权限：

- 连接到代理
- 访问目标：创建给定目标或所有目标的消费者、生产者或队列浏览器
- 自动创建目标

您可以在用户系统信息库中定义组并使用户与这些组关联 （虽然在平面文件用户系统信息库中不完全支持组）。然后，通过编辑访问控制属性文件，可以根据用户或组的目的 （生成、使用或浏览消息）指定它们可以访问的目标。可以分别指定各个目标或指定所有目标仅能够被特定用户或组访问。

另外，如果将代理配置为允许自动创建目标（请参见第 65 页“自动创建的目标”），则可以通过编辑访问控制属性文件来控制代理可为谁自动创建目标。

加密

要对客户机和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 (SSL) 标准的连接服务。 SSL 通过在启用 SSL 的代理与启用 SSL 的客户机之间建立加密连接来提供连接级别的安全性。

监视服务

代理包含大量用于监视和诊断其操作的组件。包括：

- 生成数据的组件 （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
- 将包含度量信息的 JMS 消息发送到主题目标供 JMS 监视客户机使用的消息生产者

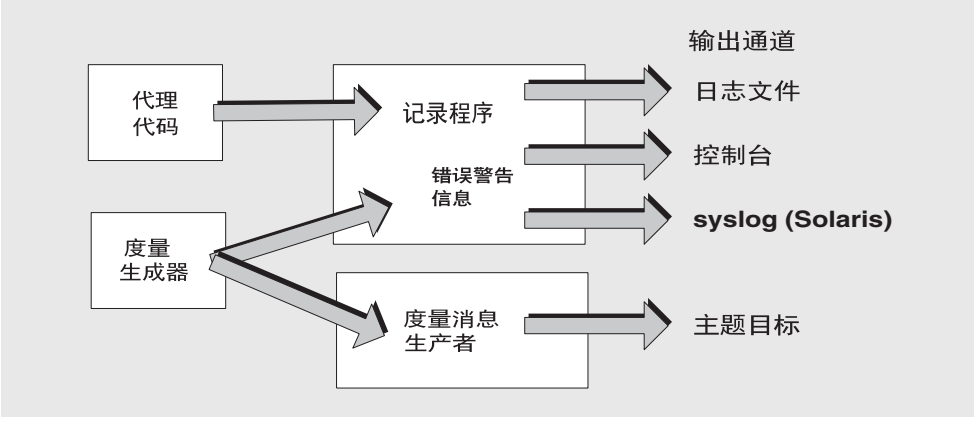
第 71 页图 4-6 中是常规模式的图解。

度量生成器

图 4-6 所示的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量及其使用的内存量、打开的连接数量和正在使用的线程数量。

可以打开或关闭度量数据的生成，并指定生成度量报告的频率。

图 4-6 监视服务支持



记录程序

图 4-6 所示的 Message Queue 记录程序记录由代理代码和度量生成器生成的信息，并将该信息写入多个输出通道中：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。

您可以指定记录程序所收集的信息类型以及写到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入转移到新文件。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。

度量消息生产者 （企业版）

图 4-6所示的度量消息生产者组件按一定的时间间隔接收度量生成器的信息，并将信息写入消息，然后根据消息中包含的度量信息类型，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 客户机可以使用目标中的消息，并处理消息中包含的度量信息。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报告的度量数量的详细信息，请参见 Message Queue Developer's Guide for Java Clients，其中介绍了如何开发使用度量消息的 Message Queue 客户机。有关如何配置度量消息生成的信息，请参见 Message Queue 管理指南。

开发和生产环境

要开发和测试消息传送应用程序以及在生产环境中部署和管理这些应用程序，您需要使用 Message Queue 服务提供的消息传送基础结构。

本节介绍在开发和生产环境中使用 Message Queue 的各种方法。涵盖下列主题：

- 第 72 页 “开发环境和任务”
- 第 73 页 “生产环境和任务”

尽管管理员负责设置和管理生产环境和任务，但开发者了解这些环境也是非常重要的，因为只有这样他们才能为管理员提供设置和配置此环境的某些部分所需的信息以及确定客户机是否按预期方式工作。

开发环境和任务

在开发环境中，工作的重点是 Message Queue 客户机应用程序编程。 Message Queue 服务主要用于测试。

即装即用式配置

Message Queue 产品被设计为即装即用式。可以使用默认值启动代理，并为您提供默认的数据存储库、用户系统信息库以及访问控制属性文件。

默认用户系统信息库是使用默认条目创建的，这些条目允许在安装 Message Queue 代理后即可使用它，而无需管理员进行任何干预。换句话说，使用代理之前无需设置初始用户 / 密码。可以使用默认用户名 (guest) 和密码 (guest) 来验证客户机用户。

此外，还提供了多个样例应用程序来指导您开发新的应用程序。

开发惯例

在开发环境中，强调的是灵活性，您通常采用以下惯例：

- 您想做最少的管理工作，主要是为开发者启动代理以便用于测试。
- 您使用数据存储库（基于内置文件持久性）、用户系统信息库（基于文件的用户系统信息库）和访问控制属性文件的默认实现方案。利用这些默认的实现方案进行开发测试通常已经足够了。
- 可以使用简单文件系统对象存储（通过创建用于该用途的目录）来存储受管理对象；也可以在客户机代码中实例化受管理对象，而根本不使用对象存储。
- 如果是执行多代理测试，很可能不会使用主代理（请参见第 79 页“群集同步”）。
- 您通常使用自动创建的目标，而不使用明确创建目标。

生产环境和任务

在生产环境中，必须对应用程序进行部署、管理和调节以获得最佳的性能。在此类环境中，对消息传送基础结构进行管理和调节是一项很重要的要求 （如果不是至关重要的）。

此外，部署必须在规模和可用性方面都能满足企业的要求。这就需要对消息服务进行自定义配置和设置，调节性能和扩展系统以应对日益增加的负荷，并对消息服务和特定于应用程序的资源进行日常监视和管理。因此，管理任务取决于消息系统的复杂性和它必须支持的应用程序的复杂性。以下各节将管理任务划分为设置操作和维护操作两类。有关执行这些任务所需的过程，请参见 Message Queue 管理指南。

设置操作

生产环境要求设置安全访问，配置连接工厂和目标对象，设置群集，配置持久性存储以及管理内存。

➤ 设置生产环境

通常需要执行以下全部设置操作或至少执行某些设置操作：

- **安全管理访问** （保护管理工具的使用）：
  - 确保 admin 连接服务处于激活状态。

- 授权：允许特定个人或 admin 组访问 admin 连接服务。
- 如果针对某个组授权，请确保管理员属于该 admin 组。
  - 基于文件的用户系统信息库：有默认的 admin 组。确保管理员属于 admin 组，或者如果使用默认的 admin 用户，请更改 admin 密码。
  - LDAP 用户系统信息库：确保管理员属于 admin 组

- **安全客户机访问：**
  - 验证：创建进入基于文件的用户系统信息库的入口或将代理配置为使用现有 LDAP 用户系统信息库。

    （至少需要设置保护管理功能的密码。）
  - 授权：修改访问控制属性文件中的访问设置。
  - 加密：设置基于 SSL 的连接服务。
- **创建物理目标**
- **创建受管理对象：**
  - 配置或设置 LDAP 对象存储。
  - 创建连接工厂和目标受管理对象。
- **必要时创建代理群集：**
  - 创建中央配置文件。
  - 指定主代理。
- **将代理配置为使用插入的持久性。**
- **配置内存管理**

  设置目标属性，以便消息数和为消息分配的内存容量与可用的代理内存资源相适合。

维护操作

在生产环境中，需要对 Message Queue 消息服务器资源进行监视和控制。应用程序性能、可靠性和安全性是优先考虑的因素，并且需要使用 Message Queue 管理工具执行如下所述的若干任务：

➤ **维护生产环境**

- **管理应用程序行为**
  - 禁用代理的自动创建功能。

- 代表应用程序创建物理目标。
- 设置用户对目标的访问权限。
- 监视和管理目标。
- 监视和管理长期订阅。
- 监视和管理事务。

- **监视和调节代理**
  - 使用代理度量来调节和重新配置代理。
  - 管理代理内存资源。
  - 将代理添加到群集中以平衡负荷。
  - 恢复出现故障的代理。
- **管理受管理对象：**
  - 根据需要创建其他连接工厂和目标被管理对象。
  - 调整连接工厂的属性值以提高性能和吞吐量。

第 5 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机提供消息传送服务的代理。使用群集，消息服务器可将客户机连接分布在多个代理上，从而调整对消息通信流量的操作。

本章阐述这些代理群集的体系结构和内部功能。涵盖下列主题：

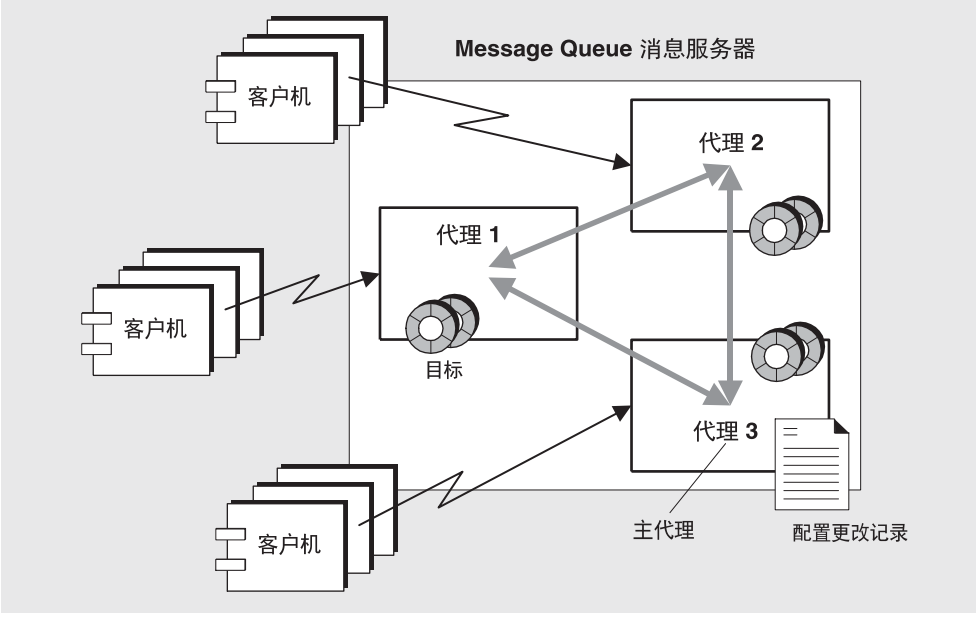
如果您是负责配置和管理代理群集的管理员，或者是需要使用群集来测试消息传送应用程序的开发者，则需要阅读本章的内容。

群集体系结构

图 5-1 显示了 Message Queue 的代理群集体系结构。群集内的每个代理直接与所有其他代理相连。每个客户机（消息生产者或消费者）都有一个**本机代理**，借助它该客户机可直接进行通信，发送和接收消息，就好像该代理是服务器上的唯一代理。实际上，本机代理与群集中的其他代理协同工作，分担为所有连接的客户机提供传送服务的负载。

可以将群集内的一个代理指定为**主代理**。主代理维护**配置更改记录**，其中记录了对群集持久性实体（目标和长期订阅）所做的更改。该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；有关详细信息，请参见下面的“群集同步”。

图 5-1 群集体系结构



以下各节阐述如何在群集内进行消息传送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，将在群集的所有代理中复制各个目标。每个代理都了解向所有其他代理的目标进行注册的消息消费者。因此，每个代理既可以将消息从其直接连接的消息生产者路由到远程消息消费者，也可以从远程生产者将消息发送到其直接连接的消费者。

对于消息生产者发出的消息，其所有存储和路由、处理所有客户机确认等操作均由该生产者自身的本机代理进行处理。为最大限度降低群集内的消息通信流量，只有在将消息传送给连接到目标代理的消费者时，才会将消息从一个代理发送到另一个代理。在某些情况下（例如，对多个消费者的队列传送），还可以通过指定向本地消费者的传送优先于向远程消费者的传送，来进一步降低通信流量。如果需要在客户机和消息服务器之间安全地传送加密消息，也可以将群集配置为在代理间提供安全的消息传送。

注 也有某些例外，群集中的目标属性整体应用于群集，而不是应用于单个目标实例。有关特定目标属性的信息，请参见 Message Queue 管理指南。

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所有其他代理 （包括主代理，如果有的话）的主机名和端口号。该信息是由一组**群集配置属性**指定的，对于群集中的所有代理，这些属性应该是统一的。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放在一个中央**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中所有代理共享的其他属性。

群集同步

每当更改群集的配置时，都会将有关更改的信息自动传播到群集中的所有代理。此类配置更改包括以下情况：

- 在群集的某个代理上创建或销毁目标。

- 消息消费者向其本机代理进行注册。
- 消息消费者与其本机代理断开连接（无论是明确断开，还是由于客户机、代理或网络故障而断开）。
- 消息消费者建立了对某主题的长期订阅。

此类配置更改信息会立即传播到群集中发生更改时处于联机状态的所有代理。不过，发生更改时，处于脱机状态的代理 （例如，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销毁的所有持久性实体（目标和长期订阅）。当脱机代理恢复联机状态时（或向群集添加新代理时），它就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消息消费者的信息。

群集中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，所以应始终先启动群集中的主代理。如果主代理处于脱机状态，则将无法在整个群集中传播配置信息，因为其他代理无法访问配置更改记录。在这些情况下，如果尝试创建或销毁目标或长期订阅，或者尝试执行某个相关操作 （例如，重新激活长期订阅），将发生异常。（不过，非管理消息传送仍然可以正常进行。）

部署环境

代理群集的使用取决于是将它们部署在开发环境中，还是部署在生产环境中。

开发环境

在开发环境中，群集用于测试，而且可伸缩性和代理恢复不是很重要，因此基本不需要主代理。在测试条件下，通常自动创建目标 （请参见第 65 页 “自动创建的目标”），并且由正在测试的应用程序创建和销毁这些目标的长期订阅。如果没有主代理， Message Queue 不再要求必须运行主代理才能启动其他代理，并允许对目标和长期订阅进行更改，同时可以将更改传播到所有正在运行的代理。（然而，如果代理脱机、随后又恢复，它将不会与脱机期间所作的更改同步。）如果将环境重新配置为使用主代理， Message Queue 将重新强制执行常规要求。

生产环境

在生产环境中，可伸缩性和代理恢复**很**重要，因此必须使用主代理并维护配置更改记录。这样，可以确保如果代理脱机、随后又恢复，它将与脱机期间所作的更改同步。

实际上，最好定期备份配置更改记录，以预防记录意外损坏及主代理故障而造成的损失。Message Queue 提供了用于备份和恢复配置更改记录的命令行选项。如果需要，也可以更改作为主代理的代理。有关详细信息，请参见 Message Queue 管理指南。

第 6 章

Message Queue 和 J2EE

Java 2 Platform, Enterprise Edition（J2EE 平台）是一种标准服务器平台用于托管多层和瘦客户机企业应用程序的规范。 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。

本章探讨在 J2EE 平台环境中实现 JMS 支持的其他信息。本章涵盖以下主题：

由于本章涉及 J2EE 组件的编程和部署两方面的内容，因此应用程序开发者和管理员都会对本章内容感兴趣。

JMS/J2EE 编程：消息驱动 Bean

除在第 24 页 “JMS 编程模型”中介绍的通用 JMS 客户机编程模型外，还有专用于 J2EE 平台应用程序上下文中的 JMS。这种专用的 JMS 客户机称为**消息驱动 Bean**，是 EJB 2.0 规范 (http://java.sun.com/products/ejb/docs.html) 中描述的企业 JavaBean (EJB) 系列组件之一。

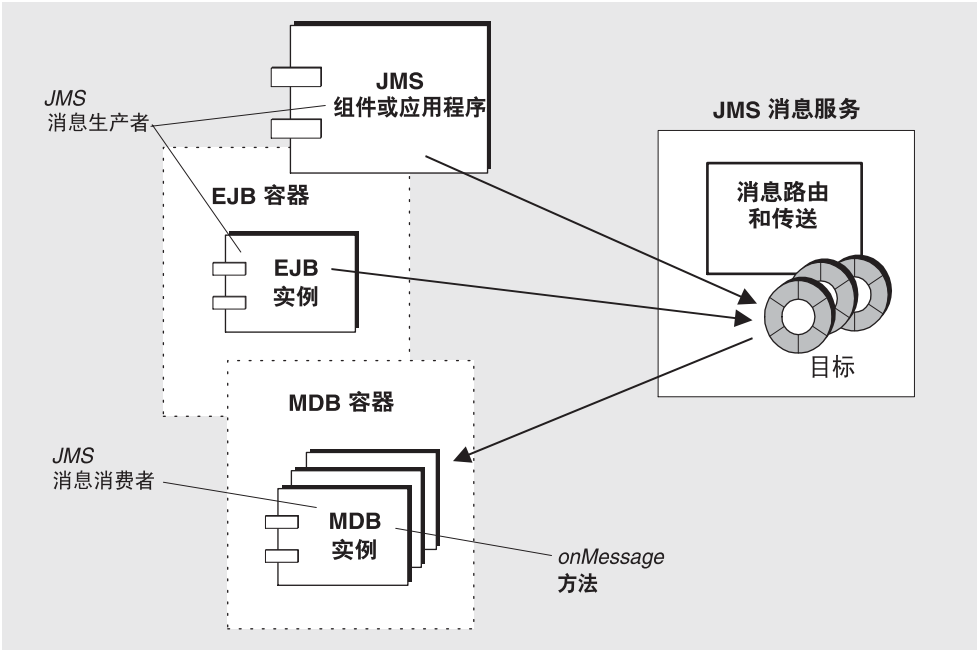
之所以需要消息驱动 Bean，是因为其他 EJB 组件（会话 Bean 和实体 Bean）只能被同步调用。这些 EJB 组件只能通过标准 EJB 接口来访问，因此不具备异步接收消息的机制。

但是，异步消息传送是许多企业应用程序的一项必不可少的要求。这些应用程序大多需要服务器端组件能够互相通信和响应，而不占用服务器资源。因此就需要有这样一种 EJB 组件：不需要紧密耦合到消息生产者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，此能力还必须在负荷增加时具有可伸缩性。

消息驱动 Bean (MDB) 是由专用的 EJB 容器 （为所支持的组件提供分布式服务的软件环境）支持的专用 EJB 组件。

消息驱动 Bean　MDB 是实现 JMS `MessageListener` 接口的 JMS 消息消费者。当 MDB 容器接收消息时，`onMessage` 方法（由 MDB 开发者编写）将被调用。`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 上的方法，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 6-1 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。

图 6-1 与 MDB 进行消息传送



MDB 容器 MDB 由专用 EJB 容器支持，负责创建并设置 MDB 实例，以进行消息的异步使用。这包括建立与消息服务的连接（包括验证）、创建与给定目标关联的会话池、管理在会话池和关联 MDB 实例之间接收到的消息的分发。由于容器控制了 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荷。

与 MDB 关联的是一个部署描述符，它指定了容器设置消息使用时使用的受管理对象的 JNDI 查找名称：连接工厂和目标。部署描述符还可能包括部署工具配置容器所需的其他信息。每个这样的容器只支持一个 MDB 的实例。

J2EE 应用服务器支持

在 J2EE 体系结构中 （请参见位于 http://java.sun.com/j2ee/download.html#platformspec 的 J2EE 平台规范）， EJB 容器由 J2EE 应用服务器托管。应用服务器为各种容器提供所需的资源：事务管理器、持久性管理器、命名服务以及 JMS 提供者 （对于消息传送和 MDB）。

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 消息传送系统被作为它的本地 JMS 提供者集成到应用服务器中。
- 对于 Sun J2EE 1.4 Application Server，Message Queue 被作为嵌入式 JMS 资源适配器插入到应用服务器中。
- 对于应用服务器的后续发行版本，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。

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 (J2EECA) 1.5 规范定义了该标准。）该体系结构允许任何应用服务器（符合 J2EE 1.4 规范）以标准方式与外部系统进行交互。外部系统可能包括各种企业信息系统 (EIS) 以及各种消息传送系统：例如， JMS 提供者。 Message Queue 包括 JMS 资源适配器，它允许应用服务器将 Message Queue 作为 JMS 提供者使用。

J2EECA 1.5 提供的标准交互包括连接池、线程池、事务与安全上下文传播，以及各种消息驱动 Bean 容器的支持。该规范还包括创建连接工厂和其他受管理对象的标准方式。

通过将 JMS 资源适配器插入应用服务器，即可在应用服务器中部署和运行的 J2EE 组件之间交换 JMS 消息。这些组件所需的 JMS 连接工厂和目标受管理对象可使用 J2EE 应用服务器管理工具进行创建和配置。

但是其他管理操作（如管理消息服务器和物理目标）则不包括在 J2EECA 规范之中，且仅能通过提供者特定工具执行。

Message Queue 资源适配器集成在 Sun J2EE 1.4 应用服务器中。不过，它尚未经过任何其他 J2EE 1.4 应用服务器的认证。

Message Queue 资源适配器是单个文件 (`imqjmsra.rar`)，所在目录视操作系统而定（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 J2EE 1.4 兼容应用服务器上使用 Message Queue 资源适配器。随着商业 J2EE 1.4 应用服务器的上市以及 Message Queue 资源适配器经过了那些应用服务器的认证，Message Queue 文档将提供相关部署和配置过程的特定信息。

可选 JMS 功能 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。

此材料最适合应用程序开发者阅读。

对每个 JMS 可选项的处理方式如表 A-1 所示：

表 A-1 可选的 JMS 功能

JMS 规范中的章节	说明和 Message Queue 处理
3.4.3 `JMSMessageID`	“Since message IDs take some effort to create and increase a message’s size, some JMS providers may be able to optimize message overhead if they are given a hint that message ID is not used by an application. JMS Message Producer provides a hint to disable message ID.”
	Message Queue 实现： 产品不会禁用消息 ID 生成 （在 `MessageProducer` 中的所有 `setDisableMessageID()` 调用均被忽略）。所有消息都将包含有效的 `MessageID` 值。
3.4.12 Overriding Message Header Fields	“JMS does not define specifically how an administrator overrides these header field values. A JMS provider is not required to support this administrative option.”
	Message Queue 实现： Message Queue 产品支持通过配置客户机运行时以管理方式覆盖消息标题字段的值 （请参见第 37 页“覆盖消息标题值”）。
3.5.9 JMS Defined Properties	“JMS Reserves the ’`JMSX`’ Property name prefix for JMS defined properties.” “Unless noted otherwise, support for these properties is optional.”
	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）。

表 A-1 可选的 JMS 功能 （续）

JMS 规范中的章节	说明和 Message Queue 处理
3.5.10 Provider-specific Properties	“JMS reserves the ’`JMS_<vendor_name>`’ property name prefix for provider-specific properties.” **Message Queue 实现：** 使用特定于提供者的属性，其目的在于提供需要的特殊功能以支持提供者本地客户机使用 JMS。不应将它们用于 JMS 至 JMS 的消息传送。Message Queue 3 不使用特定于提供者的属性。
4.4.8 Distributed Transactions	“JMS does not require that a provider support distributed transactions.” **Message Queue 实现：** 此版本的 Message Queue 产品支持分布式事务 （请参见第 56 页 “事务”）。
4.4.9 Multiple Sessions	“For PTP <point-to-point distribution model>, JMS does not specify the semantics of concurrent `QueueReceivers` for the same queue; however, JMS does not prohibit a provider from supporting this.” See section 5.8 of the JMS specification for more information. **Message Queue 实现：** Message Queue 实现支持对多个消费者的队列传送。有关详细信息，请参见第 53 页 “多个消费者的队列传送”。

词汇表

本词汇表提供了使用 Message Queue 时可能遇到的术语和概念的相关信息。

确认　系统为确保可靠的消息传送在客户机与消息服务器间交换的控制消息。确认一般分为两种类型：客户机确认与代理确认。

受管理对象　由管理员创建并供一个或多个 JMS 客户机使用的预配置对象 （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 客户机与提供者无关。受管理对象被置于 JNDI 名称空间中， JMS 客户机可以使用 JNDI 查找对它们进行访问。

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消费者做好接收准备。换言之，消息的发送方不需要等到发送方法返回即可继续进行其他工作。如果某个消息消费者正忙或处于脱机状态，系统先将消息发送出去，然后当该消费者准备好时再接收消息。

验证　只允许通过验证的用户与消息服务器建立连接的过程。

授权　消息服务确定用户是否可以访问消息服务资源 （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。

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，它为监视和调节性能以及资源使用提供了一个接口。

客户机　与其他使用消息服务的客户机进行交互的应用程序（或软件组件），以交换消息。客户机可以是生产者客户机、消费者客户机，或两者都是。

客户机标识符　将连接及其对象与代表客户机的 Message Queue 消息服务器所维护的状态关联的标识符。

客户机运行时　在消息传送客户机与 Message Queue 消息服务器之间提供接口的 Message Queue 软件。客户机运行时支持客户机向目标发送消息以及从目标接收消息所需的所有操作。客户机运行时是通过设置 ConnectionFactory 属性进行配置的。

群集　两个或多个互连的代理，协同工作以提供可伸缩的消息传送服务。

连接　客户机与消息服务器之间用于传递有效负荷消息和控制消息的通信通道。

连接工厂　客户机创建消息服务器连接时使用的受管理对象。它可以是 ConnectionFactory 对象、 QueueConnectionFactory 对象或 TopicConnectionFactory 对象。

消费者　由用于接收从目标发送的消息的会话创建的对象 (MessageConsumer)。在点对点传送模型下，消费者为收件人或浏览器（QueueReceiver 或 QueueBrowser）；在发布 / 订阅传送模型下，消费者为订户 (TopicSubscriber)。

数据存储库　用于永久存储代理所需的信息 （长期订阅、目标数据、持久性消息、审计数据）的数据库。

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、因超出内存限制而被从目标中删除或传送尝试失败。可以选择将停用消息存储在停用消息队列中。

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，以便进行诊断。

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，确保消息至少传送一次。

传送模型　消息传送的方式：点对点或发布 / 订阅。 JMS 中有两个独立的编程域与这两种模型相对应 （使用特定的客户机运行时对象、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。

目标　Message Queue 消息服务器上的物理目标，生成的消息先被传送到此处，然后再路由并传送给消费者。此物理目标由受管理对象标识和封装，客户机使用受管理对象指定生成和 / 或使用消息的目标。

域　JMS 客户机用于对 JMS 消息传送操作进行编程的一组对象。有两个编程域：一个用于点对点传送模型，另一个用于发布 / 订阅传送模型。

加密　一种防止消息在通过连接传送时被篡改的机制。

组 Message Queue 客户机用户所属的组，用于授权对连接、目标及特定操作的访问。

JMS 提供者 一种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的产品。

消息服务器 一个或多个为 Message Queue 服务提供集中式传送服务（包括客户机连接、消息处理和路由、持久性、安全性及监视）的代理。消息服务器负责维护从生产者客户机接收消息以及向消费者客户机发送消息的物理目标。

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。它包括消息服务器、客户机运行时及消息服务器执行其功能所需的多个数据存储库。

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

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。

生产者 由用于向某个目标发送消息的会话创建的对象 (MessageProducer)。在点对点传送模型中，生产者为发件人 (`QueueSender`)；在发布 / 订阅传送模型中，生产者为发布人 (`TopicPublisher`)。

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，即使使用其消息的客户机处于非活动状态。队列可用作生产者和消费者之间的中转站。

选择器 一种用于对消息进行排序和路由的消息标头属性。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。

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。

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，可以使消息发布人与消息订户相分离。

事务 不可细分的工作单位，要么整个完成，要么整个回滚。

索引

D

F

G

H

I

J

K

L

M

N

P

Q

R

S

T

W

X

Y

Z