管理员指南

Sun™ ONE Message Queue

版本 3.0.1

817-5020-10
2002 年 10 月

版权所有 © 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A.。保留所有权利。

Sun Microsystems, Inc. 拥有与本产品包含的技术相关的知识产权。需特别指出的是 （但不仅限于此），这些知识产权可能包括一项或多项在 http://www.sun.com/patents 上列出的美国专利，以及一项或多项在美国和其它国家 （地区）的其它专利或待批的专利申请。

本产品依据限制其使用、复制、分发和反编译的许可证进行分发。未经 Sun 及其许可方 （如果存在）事先书面授权，不得以任何形式、任何手段复制本产品的任何部分。

第三方软件 （包括字体技术）的版权由 Sun 供应商拥有并由他们授权。

Sun、Sun Microsystems、Sun 徽标、Java、Solaris、iPlanet、JDK、Java Naming and Directory Interface 以及 Java 咖啡杯徽标是 Sun Microsystems, Inc. 在美国和其它国家 / 地区的商标或注册商标。

所有的 SPARC 商标均需获得许可才能使用，它们是 SPARC International, Inc. 在美国和其它国家 （地区）的商标或注册商标。标有 SPARC 商标的产品均基于由 Sun Microsystems, Inc. 开发的体系结构。

UNIX 是在美国和其它国家 （地区）的注册商标，由 X/Open Company, Ltd. 独家授权。

政府采购：商业软件 — 政府用户受标准许可证条款和条件的约束。

目录

图 **11**

表 **13**

过程列表 **15**

前言 **17**

本指南的读者 **17**
本指南的结构 **18**
惯例 **19**
文本惯例 19
目录变量惯例 20
其它文档资源 **21**
MQ 文档集 21
联机帮助 22
JavaDoc 22
客户机应用程序示例 23
Java 消息服务 (JMS) 规范 23

第 1 章 概述 **25**
什么是 Sun ONE Message Queue? **25**
产品版本 **26**
平台版 26
企业版 27
企业消息传送系统 **27**
企业消息传送系统的要求 28
集中式消息传送与点对点消息传送 29
消息传送系统概念 30
消息 30
消息服务体系结构 30
消息传送模型 31

JMS 规范 31
JMS 消息结构 31
JMS 编程模型 32
被管理对象 33
JMS/J2EE 编程：消息驱动 Bean 34
消息驱动 Bean 34
应用服务器支持 36
JMS 消息传送相关问题 36
JMS 提供者无关 36
编程域 37
客户机标识符 38
可靠消息传送 38
确认 / 事务 39
持久性存储器 40
性能折衷 40
消息选择 41
消息顺序和优先级 41

第 2 章 MQ 消息传送系统 43
MQ 消息服务器 44
代理 44
连接服务 46
消息路由器 49
持久性管理器 52
安全性管理器 55
日志记录器 59
物理目标 62
队列目标 62
主题目标 63
自动创建的 （与管理员创建的相对）目标 63
临时目标 64
多代理群集 （企业版） 64
多代理体系结构 65
在开发环境中使用群集 67
群集配置属性 67
MQ 客户机运行时 69
消息生成 69
消息使用 70
MQ 被管理对象 71
“连接工厂”被管理对象 72
Destination 被管理对象 72
在客户机启动时覆盖属性值 73

第 3 章 MQ 管理 75
MQ 管理任务 75
开发环境 75
生产环境 75
设置操作 76
维护操作 76
MQ 管理工具 77
管理控制台 77
命令行实用程序概述 78
命令行语法 79
通用命令行选项 80

第 4 章 管理控制台教程 81
准备工作 82
启动管理控制台 82
获得帮助 84
使用代理 85
启动代理 86
添加代理 87
更改管理员密码 88
连接到代理 89
查看连接服务 89
将物理目标添加到代理 91
使用物理目标 93
获得有关主题目标的信息 95
使用对象存储 95
添加对象存储 96
检查对象存储属性 98
连接到对象存储 99
添加连接工厂被管理对象 99
添加目标被管理对象 101
被管理对象的属性 103
更新控制台信息 104
运行范例应用程序 104

第 5 章 启动与配置代理 107
配置文件 108
合并属性值 108
属性命名语法 109
编辑实例配置文件 110
启动代理 114
使用群集 （企业版） 118
群集配置属性 118

连接代理 120
方法 1：无群集配置文件 120
方法 2：使用群集配置文件 120
将代理添加到群集 120
重新启动群集中的代理 121
从群集中删除代理 121
备份主管代理的配置更改记录 122
恢复主管代理的配置更改记录 122
日志记录 122
默认日志记录配置 123
日志消息格式 123
更改日志记录器配置 124
更改输出通道 125
更改转移标准 125
记录代理性能测量信息 126

第 6 章 代理和应用程序管理 129
命令行实用程序 130
命令语法 130
imqcmd 子命令 130
imqcmd 选项概述 132
使用 imqcmd 的前提条件 133
实例 134
控制代理的状态 134
查询和更新代理属性 136
查询代理 137
更新代理 138
管理连接服务 139
列出连接服务 140
查询和更新服务属性 141
暂停和恢复服务 143
管理目标 143
创建目标 144
获得目标的相关信息 145
更新目标 145
清理目标 145
销毁目标 146
管理长期订阅 146
管理事务 147

第 7 章 管理被管理对象 151
关于对象存储 152

被管理对象 152
对象管理器实用程序 (imqobjmgr) 153
命令语法 153
imqobjmgr 子命令 153
imqobjmgr 命令选项概述 154
所需的信息 155
被管理对象的属性 156
连接工厂被管理对象 156
目标被管理对象 158
对象存储属性 159
初始上下文和地址信息 159
安全信息 （仅用于 LDAP） 159
使用输入文件 160
添加和删除被管理对象 163
添加连接工厂 163
添加主题或队列 164
删除被管理对象 165
获得信息 166
被管理对象列表 166
单个对象的相关信息 167
更新被管理对象 168

第 8 章 安全性管理 169
验证用户 170
使用文本文件用户系统信息库 170
用户管理器实用程序 (imqusermgr) 171
组 173
状态 173
用户名和密码的格式 174
填充和管理用户系统信息库 174
更改默认的管理员密码 175
使用 LDAP 服务器管理用户系统信息库 176
授权用户：访问控制属性文件 178
访问规则语法 179
权限计算 180
连接访问控制 181
目标访问控制 181
目标自动创建访问控制 182
加密：使用 SSL 服务 （企业版） 183
设置通过 TCP/IP 的 SSL 服务 184
步骤 1：生成自签名证书 184
步骤 2：在代理上启用基于 SSL 的服务 186
步骤 3：启动代理 186

步骤 4：配置并运行基于 SSL 的客户机 187
设置通过 HTTP 的 SSL 服务 188
使用密码文件 189

附录 A 设置插入的持久性 191
简介 191
插入支持 JDBC 的数据存储 192
JDBC 相关的代理配置属性 193
数据库管理器实用程序 (imqdbmgr) 195
命令语法 195
imqdbmgr 子命令 196
imqdbmgr 命令选项概述 196

附录 B HTTP/HTTPS 支持 （企业版） 197
HTTP/HTTPS 支持体系结构 197
实现 HTTP 支持 199
步骤 1：在 Web 服务器上部署 HTTP 隧道 Servlet 199
部署为 Jar 文件 199
部署为 Web 归档文件 200
步骤 2：配置 httpjms 连接服务 200
步骤 3：配置 HTTP 连接 201
设置连接工厂属性 201
使用一个 Servlet 访问多个代理 202
使用 HTTP 代理 202
实例：部署 HTTP 隧道 Servlet 203
部署为 Jar 文件 203
部署为 WAR 文件 205
实现 HTTPS 支持 206
步骤 1：为 HTTPS 隧道 Servlet 生成自签名证书 206
步骤 2：在 Web 服务器上部署 HTTPS 隧道 Servlet 207
部署为 Jar 文件 207
部署为 Web 归档文件 208
步骤 3：配置 httpsjms 连接服务 208
步骤 4：配置 HTTPS 连接 210
配置 JSSE 210
输入根证书 210
设置连接工厂属性 211
使用一个 Servlet 访问多个代理 211
使用 HTTP 代理 212
实例：部署 HTTPS 隧道 Servlet 212
部署为 Jar 文件 212
部署为 WAR 文件 214

附录 C 使用代理作为 Windows 服务 217
将代理作为 Windows 服务运行 217
服务管理器实用程序 (imqsvcadmin) 218
命令语法 218
imqsvcadmin 子命令 218
imqsvcadmin 选项概述 219
删除代理服务 219
重新配置代理服务 220
使用替代 Java 运行时 220
查询代理服务 220
疑难解答 220

附录 D MQ 数据的位置 221

附录 E MQ 接口的稳定性 223

词汇表 227

索引 231

图

图 1-1 集中式消息传送与点对点消息传送 29
图 1-2 消息服务体系结构 30
图 1-3 JMS 编程对象 32
图 1-4 与 MDB 进行消息传送 35
图 2-1 MQ 系统的体系结构 43
图 2-2 代理组件 45
图 2-3 连接服务支持 47
图 2-4 持久性管理器支持 53
图 2-5 安全性管理器支持 56
图 2-6 日志记录图解 59
图 2-7 多代理（群集）体系结构 65
图 2-8 消息传送操作 69
图 2-9 将消息传送到 MQ 客户机运行时 70
图 3-1 本地和远程管理实用程序 78
图 5-1 代理配置文件 109
图 B-1 HTTP/HTTPS 支持体系结构 198

表

表 1 本书内容 18
表 2 文档惯例 19
表 3 MQ 目录变量 20
表 4 MQ 文档集 22
表 1-1 JMS 编程对象 37
表 2-1 主要代理组件及功能 45
表 2-2 代理支持的连接服务 46
表 2-3 连接服务属性 48
表 2-4 消息路由器属性 52
表 2-5 持久性属性 54
表 2-6 安全性属性 57
表 2-7 日志种类 59
表 2-8 日志记录器属性 60
表 2-9 自动创建配置属性 64
表 2-10 群集配置属性 68
表 2-11 目标属性 73
表 3-1 通用 MQ 命令行选项 80
表 5-1 代理实例配置属性 110
表 5-2 `imqbrokerd` 选项 115
表 5-3 群集配置属性 118
表 5-4 `imqbrokerd` 日志记录器选项及相应的属性 124
表 5-5 对连接服务收集的测量信息 127
表 5-6 对每个代理所收集的测量信息 127
表 6-1 `imqcmd` 子命令 130
表 6-2 `imqcmd` 选项 132
表 6-3 用于控制代理的 `imqcmd` 子命令 135
表 6-4 用于获取代理信息和更新代理的 `imqcmd` 子命令 136

表 6-5 代理属性 138
表 6-6 用于管理连接服务的 imqcmd 子命令 139
表 6-7 代理所支持的连接服务 140
表 6-8 连接服务属性 142
表 6-9 用于管理目标的 imqcmd 子命令 143
表 6-10 目标属性 144
表 6-11 用于管理长期订阅的 imqcmd 子命令 146
表 6-12 用于管理事务的 imqcmd 子命令 148
表 7-1 imqobjmgr 子命令 153
表 7-2 imqobjmgr 选项 154
表 7-3 连接工厂属性 157
表 7-4 目标属性 158
表 7-5 对象存储的安全属性 160
表 8-1 用户系统信息库中的初始条目 170
表 8-2 imqusermgr 子命令 172
表 8-3 imqusermgr 选项 172
表 8-4 用户名和密码中不能包含的字符 174
表 8-5 LDAP 相关属性 176
表 8-6 访问规则的语法元素 179
表 8-7 目标访问控制规则的元素 182
表 8-8 键存储属性 185
表 8-9 密码文件中的密码 189
表 A-1 JDBC 相关属性 193
表 A-2 imqdbmgr 子命令 196
表 A-3 imqdbmgr 选项 196
表 B-1 httpjms 连接服务属性 201
表 B-2 用于部署 HTTP 隧道 Servlet Jar 文件的 Servlet 参数 204
表 B-3 httpsjms 连接服务属性 209
表 B-4 用于部署 HTTPS 隧道 Servlet Jar 文件的 Servlet 参数 213
表 C-1 imqsvcadmin 子命令 218
表 C-2 imqsvcadmin 选项 219
表 D-1 MQ 3.0 数据的位置 221
表 E-1 MQ 3.0.1 接口的稳定性 223
表 E-2 接口稳定性分类方案 225

过程列表

启动管理控制台的步骤 82
显示管理控制台帮助信息的步骤 84
启动代理的步骤 86
将代理添加到管理控制台的步骤 87
更改管理员密码的步骤 88
连接到代理的步骤 89
查看可用连接服务的步骤 90
将队列目标添加到代理的步骤 92
查看物理目标属性的步骤 93
清理目标中的消息的步骤 94
删除目标的步骤 95
添加文件系统对象存储的步骤 96
显示对象存储属性的步骤 98
连接到对象存储的步骤 99
将连接工厂添加到对象存储的步骤 99
将目标添加到对象存储的步骤 102
查看或更新目标对象属性的步骤 103
运行 `SimpleJNDIClient` 应用程序的步骤 104
将代理连接到群集的步骤 120
将代理添加到群集中的步骤 （如果是使用群集配置文件） 121
万一发生故障时恢复主管代理的步骤 122
更改代理的日志记录器配置的步骤 124
编辑配置文件以使用 LDAP 服务器的步骤 176
设置 ssljms 连接服务的步骤 184
重新生成密钥对的步骤 186
插入支持 JDBC 的数据存储的步骤 192
实现 HTTP 支持的步骤 199

启用 httpjms 连接服务的步骤 200
添加隧道 Servlet 的步骤 203
配置隧道 Servlet 虚拟路径 (Servlet URL) 的步骤 204
在 Web 服务器启动时装入隧道 Servlet 的步骤 204
禁用服务器访问日志的步骤 205
将 HTTP 隧道 Servlet 部署为 WAR 文件的步骤 205
实现 HTTPS 支持的步骤 206
启用 httpsjms 连接服务的步骤 209
配置 JSSE 的步骤 210
添加隧道 Servlet 的步骤 212
配置隧道 Servlet 虚拟路径 (Servlet URL) 的步骤 213
在 Web 服务器启动时装入隧道 Servlet 的步骤 214
禁用服务器访问日志的步骤 214
修改 HTTPS 隧道 Servlet WAR 文件的步骤 214
将 HTTPS 隧道 Servlet 部署为 WAR 文件的步骤 215
查看记录的服务错误事件的步骤 220

前言

《Sun™ ONE Message Queue (MQ) 3.0.1 **管理员指南**》这本书提供了执行 MQ 消息传送系统的管理任务所需的背景和信息。

本前言包含以下章节：

- 本指南的读者
- 本指南的结构
- 惯例
- 其它文档资源

本指南的读者

本指南供需要执行 MQ 管理任务的系统管理员以及应用程序开发者使用。

MQ 管理员负责设置和管理 MQ 消息传送系统，特别是该系统的核心 MQ — 消息服务器。本书并不要求读者掌握任何消息传送系统的知识或对其有一定了解。

本指南也供应用程序开发者使用，使他们更好地了解如何优化应用程序，从而充分利用 MQ 消息传送系统的功能和灵活性。

本指南的结构

请从头到尾按部就班地阅读本指南。下表简要介绍了各章的内容：

表 1 本书内容

章	说明
第 1 章 “概述”	对 MQ 消息传送系统及其术语进行了高级概念综述。
第 2 章 “MQ 消息传送系统”	说明 MQ 消息传送系统，重点介绍了协同工作以提供消息传送服务的 MQ 代理和 MQ 客户机运行时。
第 3 章 “MQ 管理”	介绍 MQ 管理任务和管理工具，并介绍了用于管理的命令行公用程序及其通用功能。
第 4 章 “管理控制台教程”	提供一个实用教程，借助此教程，您将熟练掌握管理控制台（MQ 消息服务器的图形界面）的操作方法。
第 5 章 “启动与配置代理”	介绍如何启动和配置 MQ 代理和代理群集。
第 6 章 “代理和应用程序管理”	介绍如何执行与管理 MQ 代理相关的 （应用程序无关）任务，以及如何执行用于管理消息传送应用程序的任务。
第 7 章 “管理被管理对象”	介绍如何执行与创建和管理 MQ 被管理对象相关的任务。
第 8 章 “安全性管理”	介绍如何执行与应用程序相关的安全任务，例如管理验证、授权和加密。
附录 A “设置插入的持久性”	介绍如何设置 MQ 以使用 JDBC 兼容数据库执行持久性功能。
附录 B “HTTP/HTTPS 支持 （企业版）”	介绍如何在消息传送客户机和 MQ 消息服务器之间建立 HTTP 连接服务。
附录 C “使用代理作为 Windows 服务”	介绍如何使用 MQ 服务管理公用程序 (`imqsvcadmin`) 安装、查询和删除代理 （作为 Windows 服务运行）。
附录 D “MQ 数据的位置”	说明不同类别的 MQ 数据的位置。
附录 E “MQ 接口的稳定性”	说明不同 MQ 接口的稳定性。
“词汇表”	定义在 MQ 文档中使用的术语。

惯例

本节介绍本文档中使用的有关惯例。

文本惯例

表 2 文档惯例

格式	说明
斜体文本	*italics* 表示占位符。可用相应的子句或值替换斜体文本。斜体文本还用于表示文档标题、强调、引入的词或短语。
等宽文本	`monospace` 表示实例代码、在命令行上输入的命令、目录、文件、路径名、错误信息文本、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 （MQ、 JSP）。
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。

目录变量惯例

MQ 使用三种目录变量，不同平台的变量设置有所不同。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 MQ 目录变量

变量	说明
IMQ_HOME	MQ 文档中通常使用它来引用 MQ 根安装目录： • 在 Solaris 平台上，不存在 MQ 根安装目录。因此， MQ 文档不使用 IMQ_HOME 来引用在 Solaris 上的文件位置。 • 在 Solaris 平台上，对于 Sun ONE Application Server 测试版， MQ 根安装目录为：*root Application Server installation directory*/imq。 • 在 Windows 平台上， 根安装目录是由 MQ 安装程序 MQ 设置 （默认情况下为 C:\Program Files\Sun Microsystems\Message Queue 3.0.1）。 • 在 Windows 平台上，对于 Sun ONE Application Server，MQ 根安装目录为：*root Application Server installation directory*/imq。 • 在 Linux 平台上，默认的 MQ 根安装目录为：/opt/imq。
IMQ_VARHOME	这是 /var 目录，其中存储了 MQ 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 平台上，默认的 IMQ_VARHOME 为 /var/imq 目录。 • 在 Solaris 平台上，对于 Sun ONE Application Server 测试版，默认的 IMQ_VARHOME 为 IMQ_HOME/var。 • 在 Windows 平台上，默认的 IMQ_VARHOME 为 IMQ_HOME/var。 • 在 Windows 平台上，对于 Sun ONE Application Server，默认的 IMQ_VARHOME 为 IMQ_HOME/var。 • 在 Linux 平台上，默认的 IMQ_VARHOME 为 IMQ_HOME/var。

表 3 MQ 目录变量 （续）

变量	说明
IMQ_JAVAHOME	这是一个环境变量，指向 MQ 可执行文件所需的 Java 运行时 (JRE 1.4) 的位置： • 在 Solaris 平台上，默认的 IMQ_JAVAHOME 为 /usr/j2se/jre 目录，但是用户可以选择将其值设置为存储 JRE 1.4 的位置。 • 在 Windows 平台上，默认的 IMQ_JAVAHOME 为 IMQ_HOME/jre，但是用户可以选择将其值设置为存储 JRE 1.4 的位置。 • 在 Linux 平台上，默认的 IMQ_JAVAHOME 为 /usr/java/j2sdk1.0/jre 目录，但是用户可以选择将其值设置为存储 JRE 1.4 的位置。

在本指南中，所显示的 IMQ_HOME、IMQ_VARHOME 和 IMQ_JAVAHOME **没有**采用各个平台特有的环境变量表示法或语法 （例如，在 UNIX 上为 $IMQ_HOME）。所有的路径名采用 UNIX 文件分隔符表示法 (/)。

其它文档资源

除了本指南以外， MQ 还提供了其它文档资源。

MQ 文档集

表 4 按照通常的使用顺序列出了 MQ 文档集中的文档。

表 4 MQ 文档集

文档	读者	说明
MQ **安装指南**	开发者和管理员	介绍如何在 Solaris、 Linux 和 Windows 平台上安装 MQ 软件。
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。
MQ *Developer's Guide*	开发者	提供了与 JMS 的 MQ 实现方案相关的快速入门教程和编程信息。
MQ **管理员指南**	管理员，也建议开发者阅读	提供使用 MQ 管理工具执行管理任务时所需的背景和信息。

联机帮助

MQ 3.0.1 包含用于执行 MQ 消息服务的管理任务的命令行公用程序。要访问这些公用程序的联机帮助，请参见第 80 页上的 “通用命令行选项”。

MQ 3.0.1 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

在以下位置提供了 JMS 和 MQ API 文档 （JavaDoc 格式）：

```
IMQ_HOME/javadoc/index.html
```

（在 Solaris 上为：`/usr/share/javadoc/imq/index.html`）

此文档可以在任何 HTML 浏览器中浏览，例如 Netscape 或 Internet Explorer。它包括标准 JMS API 文档以及 MQ 被管理对象的 MQ 特有 API （请参见 《MQ *Developer's Guide*》的第 3 章），这些对消息传送应用程序的开发者很有帮助。

客户机应用程序示例

以下位置含有大量应用程序示例，提供了范例客户机应用程序代码：

IMQ_HOME/demo （在 Solaris 上为 /usr/demo/imq）

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

请在以下位置查找 JMS 规范：

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

规范包含范例客户机代码。

第 1 章

概述

本章提供了 Sun™ ONE Message Queue (MQ) 的入门知识，适合于管理员和程序员阅读。

什么是 Sun ONE Message Queue？

MQ 产品为应用程序间进行通信和可靠的消息传送提供了基于标准的解决方案。MQ 也是实现 Java 消息服务 (JMS) 开放标准的企业消息传送系统，也就是说它是 JMS 提供者。

JMS 规范用于说明一套编程接口 （请参见第 32 页上的 “JMS 编程模型”），这些规范为 Java 应用程序提供了在分布式环境中创建、发送、接收和读取消息的通用方法。

使用 Sun ONE Message Queue 软件，在不同平台和操作系统上运行的进程可以连接到一个通用的 MQ 消息服务 （请参见第 30 页上的 “消息服务体系结构”），以发送和接收信息。这样，应用程序开发者就可以将精力集中在应用程序的业务逻辑上，而不是集中在 “应用程序如何在网络中进行通信”这样的细枝末节上。

MQ 具有超出 JMS 规范最低要求的功能，包括：

集中式管理 提供命令行和 GUI 两种工具来管理 MQ 消息服务及管理消息传送中的应用程序特有项目，例如目标和安全。

可伸缩的消息服务 您可以在以串连模式 （多代理群集）工作的大量 MQ 消息服务组件 （**代理**）之间平衡负荷，从而为不断增加的 JMS 客户机 （组件或应用程序）提供服务。

可调节的性能 在可以降低传送可靠性的情况下，提高 MQ 消息服务的性能。

多种传输协议 支持 JMS 客户机使用多种不同传输协议 （例如 TCP 和 HTTP）进行相互通信的功能，并支持使用安全 (SSL) 连接。

JNDI 支持 支持将 Java 命名和目录接口 (JNDI) 的基于文件的实现方案和 LDAP 实现方案用作对象存储和用户系统信息库。

SOAP 消息传送支持 支持通过 JMS 消息传送来创建和传送 SOAP 消息 （即符合简单对象访问协议 [SOAP] 规范的消息）。SOAP 允许在分布式环境中的点之间交换结构化 XML 数据。有关详细信息，请参见 《MQ *Developer's Guide*》。

有关 JMS 兼容问题的文档，请参见 《MQ 3.0.1 *Release Notes* （**发行说明**）》。

产品版本

Sun ONE Message Queue 产品有两个版本：平台版和企业版，每个版本都与不同的许可功能相对应，如下所述。（要将 MQ 从一个版本升级到另一个版本，请参见《MQ **安装指南**》中的说明。）

平台版

此版本可以从 Sun 的 Web 站点免费下载，并且还捆绑有最新的 Sun ONE Application Server 平台。平台版没有限制每个 MQ 消息服务所支持的 JMS 客户机连接的数量。它附带以下两个许可证，如下所述：

- **基本许可证** 此许可证提供基本的 JMS 支持 （它是 JMS 的全权提供者），但是不包括某些企业功能，例如负荷平衡 （多代理消息服务）、 HTTP/HTTP 连接、安全连接服务、可伸缩连接功能以及多队列传送策略。此许可证没有期限限制，因此可用于要求不太严格的生产环境。

- **90 天试用期的企业许可证** 此许可证包含基本许可证未包含的全部企业功能，例如支持多代理消息服务、 HTTP/HTTP 连接、安全连接服务、可伸缩连接功能以及多队列传送策略。但是，此许可证的期限限制为 90 天，因此，此许可证适用于评估本产品的企业版所提供的企业功能 （请参见第 27 页上的 “企业版”）。

注 使用 `-license` 命令行选项启动 MQ 消息服务，即一个 MQ 代理实例 （请参见第 115 页上的表 5-2 ），并将 “`try`”作为要使用的许可证进行传送，即可启用 90 天试用许可证。

```
imqbrokerd -license try
```

每次启动代理实例时都必须使用此选项，否则默认情况下将回退到基本的平台版许可证。

企业版

此版本用于在生产环境中部署和运行消息传送应用程序。它支持多代理消息服务、HTTP/HTTPS 连接、安全连接服务、可伸缩连接功能以及多队列传送策略。您还可以使用企业版进行开发、调试，并可以装入测试消息传送应用程序和组件。企业版对许可证没有期限限制，对多代理消息服务中的代理数量也没有限制，但对所支持的 CPU 数量有限制。

企业消息传送系统

企业消息传送系统使独立的分布式组件或应用程序可以通过消息进行交互。这些组件 （无论是在相同的系统、相同的网络上运行，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及协调各自的功能。

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，一个企业消息传送系统一般必须满足以下要求：

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成功地传送消息。

异步传送　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使用方是否可以立即接收消息。如果某个使用方忙或脱机，系统必须允许进行这样的操作：先将消息发送出去，然后该使用方在准备就绪时再接收消息。这就是所说的“异步消息传送”，也常称为“存储并转发”消息传送。

安全性　消息传送系统必须支持基本的安全性功能：用户验证、消息及资源的访问授权和摘要式加密。

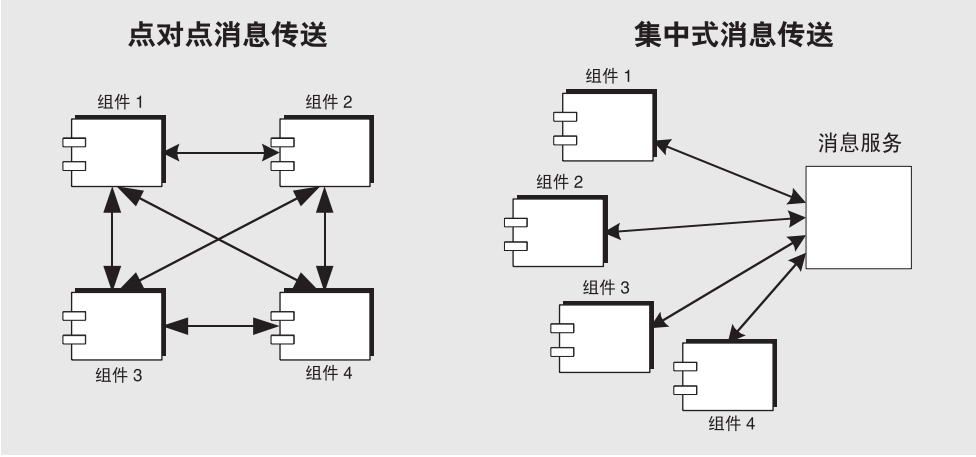
可伸缩性　消息传送系统必须能够在不降低系统性能或消息吞吐量的前提下容纳不断增长的负荷，即用户数量和消息数量的增加。当业务和应用程序扩展时，这将成为一个十分重要的要求。

易于管理性　消息传送系统必须提供用于监视和管理消息传送以及优化系统资源的工具。这些工具有助于衡量和维护系统的可靠性、安全性和性能。

集中式消息传送与点对点消息传送

传统的点对点消息传送系统很难满足企业消息传送系统的要求，如图 1-1 所示。

图 1-1 集中式消息传送与点对点消息传送



在点对点系统中，每一个消息传送组件与全部其它组件都保持连接。这些连接可以实现快速、安全和可靠的传送，但是支持可靠性和安全性的代码必须驻留在每一个组件中。随着组件不断添加到系统中，连接的数量将成指数级增加。这使得异步消息传送和可伸缩性很难实现。集中式管理也变得问题重重。

企业消息传送首选的解决方案就是采用集中式消息传送系统，如图 1-1 中所示。在该解决方案中，每个消息传送组件只需与一个中央消息服务保持连接。消息服务提供组件之间的消息路由和传送，并负责可靠传送及安全性。组件通过定义完善的程序接口与消息服务进行交互。当系统中添加组件时，连接的数量只会线性增加，这样就可以轻松地通过调整消息服务来调整系统。另外，中央消息服务为系统提供集中式管理。

消息传送系统概念

以下几个基本概念构成了企业消息传送系统的基础，包括：消息、消息服务体系结构和消息传送模型。

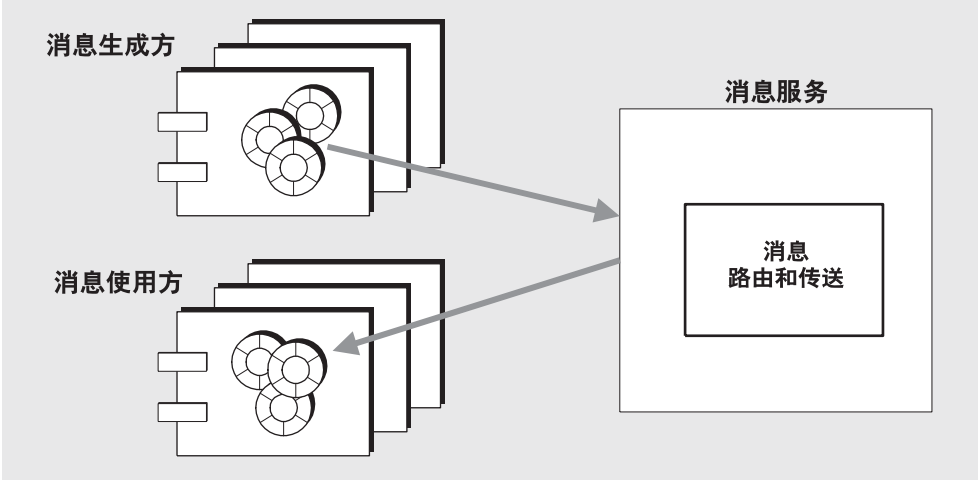
消息

消息由采用某种格式的数据 （消息主体）和说明消息属性 （例如目标、有效期或其他由消息传送系统确定的属性）的元数据 （消息标题）构成。

消息服务体系结构

第 30 页上的图 1-2 所示为消息传送系统的基本体系结构。它包括消息生成方和消息使用方，它们通过通用消息服务来交换消息。同一消息传送组件 （或应用程序）中可以驻留任意数量的消息生成方和消息使用方。消息生成方将消息发送至消息服务。接下来，消息服务使用消息路由和传送组件将消息传送至一个或多个已注册对此消息的请求的消息使用方。消息路由和传送组件负责保证将消息传送至所有相应的使用方。

图 1-2 消息服务体系结构



消息传送模型

生成方和使用方之间存在多种关系：一对一、一对多和多对多。例如，您可能会有以下类型的消息：

- 一个生成方至一个使用方
- 一个生成方至多个使用方
- 多个生成方至一个使用方
- 多个生成方至多个使用方

这些关系一般可简化为两个消息传送模型：**点对点**消息传送和**发布/订阅**消息传送。点对点传送模型主要处理由一个特定生成方创建并由一个特定使用方接收的消息。发布 / 订阅传送模型主要处理由任何数量的生成方创建并由任何数量的使用方接收的消息。这两种消息传送模型可以组合起来使用。

由于历史原因，消息系统支持这两种消息传送模型的各种组合方式。 Java 消息服务 (JMS) API 适用于为 Java 消息传送创建通用的编程方式。它既支持点对点消息传送模型，也支持发布 / 订阅消息传送模型 （请参见第 37 页上的 “编程域”）。

JMS 规范

JMS 指定了消息结构、编程模型、和一套控制消息传送操作的规则和语义。因为 MQ 提供 JMS 的实现方案，所以 JMS 概念是理解 MQ 消息传送系统如何工作的基础。本介绍说明了理解本书其余各章所需的概念和术语。

JMS 消息结构

按照 JMS 规范，一个消息包括三部分：标题、属性和主体。

标题　标题指定了消息的 JMS 属性：目标、持久与否、有效期及优先级。这些属性控制了消息传送系统如何传送消息。

属性　属性 （可以看作标题的扩充）为可选项，应用程序可以使用属性提供的值根据各种选择标准来过滤消息。属性为可选项。

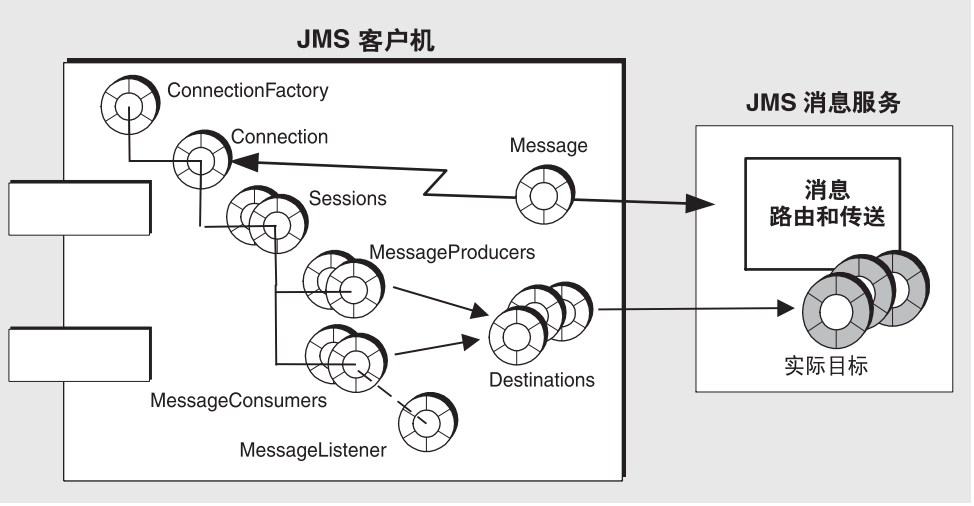
消息主体　消息主体包括要交换的实际数据。 JMS 支持六种主体类型。

JMS 编程模型

在 JMS 编程模型中， JMS 客户机 （组件或应用程序）通过 JMS 消息服务交换消息。消息生成方将消息发送至消息服务，消息使用方则从消息服务接收这些消息。这些消息传送操作是使用一组实现 JMS 应用编程接口 (API) 的对象 （由 JMS 提供者提供）来执行的。

本节将介绍实现 JMS API 的对象和用于配置要进行消息传送的 JMS 客户机的对象。有关详细信息，请参见 《MQ *Developer's Guide*》。第 32 页上的图 1-3 所示为用于编写消息传送的 JMS 对象。

图 1-3 JMS 编程对象



在 JMS 编程模型中， JMS 客户机使用 `ConnectionFactory` 对象创建一个连接，向 JMS 消息服务发送消息以及从 JMS 消息服务接收消息均是通过此连接来进行。`Connection` 是 JMS 客户机与消息服务的活动连接。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。

连接用于创建会话。 Session 是一个用于生成和接收消息的单线程上下文。它用于创建发送和接收消息的生成方和使用方，并为所发送的消息定义发送顺序。通过大量的确认选项或事务 （可由分布式事务管理器来管理），会话支持可靠的传送。

JMS 客户机使用 MessageProducer 向指定的物理目标 （在 API 中表示为目标对象）发送消息。消息生成方可指定一个默认传送模式 （持久性消息或非持久性消息）、优先级和有效期值，以控制生成方向物理目标发送的所有消息。

同样， JMS 客户机使用 MessageConsumer 对象从指定的物理目标 （在 API 中表示为目标对象）接收消息。消息使用方可使用消息选择器，借助它，消息服务可以仅向消息使用方发送与选择标准匹配的那些消息。

消息使用方可以支持同步或异步消息接收 （请参见 《MQ *Developer's Guide*》）。异步接收可通过向使用方注册 MessageListener 来实现。当会话线程调用 MessageListener 对象的 onMessage() 方法时，客户机将接收到消息。

被管理对象

在第 32 页上的 “JMS 编程模型”介绍的对象中，有两个对象取决于 JMS 提供者如何实现 JMS 消息服务。连接工厂对象取决于提供者传送消息时使用的基本协议和机制，而目标对象则取决于提供者使用的特有命名惯例和物理目标容量。

通常，这些 “提供者特有” 属性使得 JMS 客户机代码取决于特定的 JMS 实现方案。然而，要使 JMS 客户机代码与提供者无关， JMS 规范要求提供者特有的实现方案和配置信息应封装在称为**被管理对象的对象中**。这些对象可以使用非提供者特有的标准方式来访问。

被管理对象由管理员创建和配置，并存储在命名服务中，由 JMS 客户机应用程序通过标准 “Java 命名和目录服务” (JNDI) 查找代码来访问。以这种方式使用被管理对象可使 JMS 客户机代码与提供者无关。

JMS 提供了两种通用类型的被管理对象。连接工厂对象和目标对象。虽然两者都用于封存提供者特有的信息，但在 JMS 客户机中，它们的用途却有很大的差异。连接工厂用于创建至消息服务器的连接，而目标对象用于标识 JMS 消息服务使用的物理目标。

有关被管理对象的详细信息，请参见第 71 页上的 “MQ 被管理对象”。

JMS/J2EE 编程：消息驱动 Bean

除在第 32 页上的 “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 Java 2 企业版 (J2EE) 应用程序上下文中的 JMS。这些专用的 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`) 中指定的企业 JavaBean (EJB) 系列组件之一。

之所以需要诸如消息驱动 Bean 的 EJB 组件，是由于其他 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。

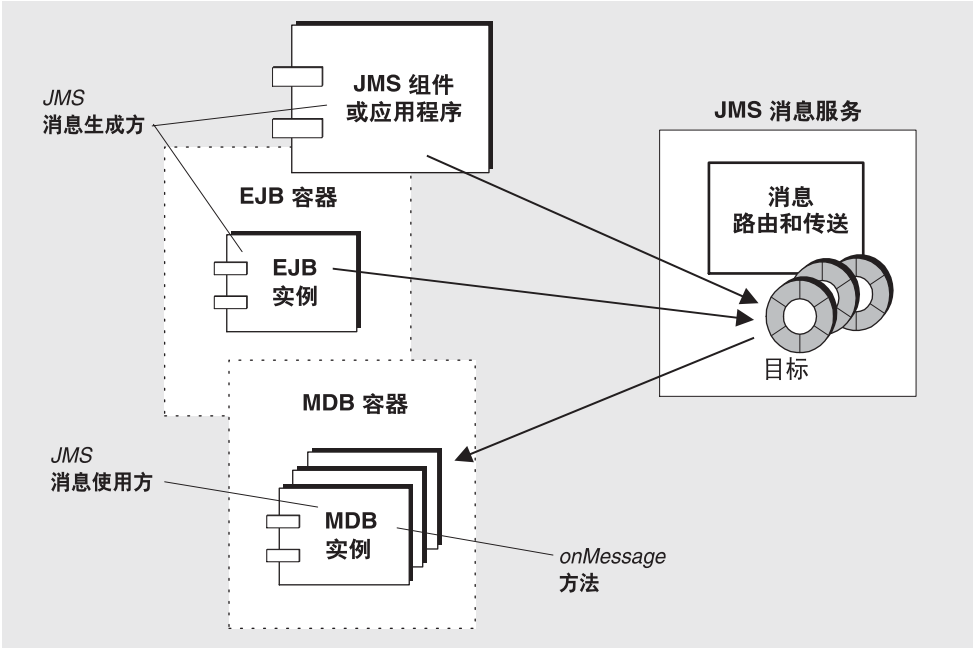
但是，异步消息传送是许多企业应用程序的一项必不可少的要求。这些应用程序大都需要服务器端组件能够互相通信和响应，而不占用服务器资源。因此就需要有这样一种 EJB 组件，它不需要紧密耦合到消息生成方即可接收消息。对于服务器端组件必须响应应用程序事件的任何应用程序，均必须具有上述功能。在企业应用程序中时，此功能还必须在负荷增加时具有可伸缩性。

消息驱动 Bean

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 消息使用方。在 MDB 容器接收消息时， `onMessage` 方法 （由 MDB 开发者编写）将被调用。`onMessage()` 方法接收消息的方式与标准 `MessageListener` 对象的 `onMessage()` 方法接收消息的方式一样。不能以调用其他 EJB 组件的方法来远程调用 MDB 的方法，因此不存在与之关联的主接口或远程接口。 MDB 可接收来自单一目标的消息。如第 35 页上的图 1-4 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 1-4 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，以进行消息的异步接收。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 实例之间分发接收到的消息。由于容器控制了 MDB 实例的有效期，因此它通过管理 MDB 实例池来容纳外来的消息负荷。

与 MDB 关联的是一个部署描述符，它指定了容器设置消息接收时使用的被管理对象的 JNDI 查找名称：连接工厂和目标。部署描述符还可能包括可由部署工具用于配置容器的其它信息。每个这样的容器只支持一个 MDB 的实例。

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 容器驻留在应用服务器中。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 ONE Application Server 中，消息传送资源由 Sun ONE Message Queue 提供。也就是说， MQ 消息传送系统 （请参见第 2 章 “MQ 消息传送系统”）已集成到 Sun ONE Application Server 中，为向应用服务器环境中运行的 MDB 和其他 JMS 消息传送组件发送 JMS 消息提供所需的支持。

JMS 消息传送相关问题

本章介绍了许多影响 MQ 消息服务管理的 JMS 编程相关问题，着重介绍 MQ 管理员需要的概念和术语。

JMS 提供者无关

JMS 指定被管理对象的用途 （请参见第 33 页上的 “被管理对象”）以支持可移植到其他 JMS 提供者的客户机应用程序的开发。被管理对象使 JMS 客户机可以使用逻辑名称查找和引用提供者特有的对象。这样，客户机代码无需知道提供者使用的特有命名语法、寻址语法或可配置属性。从而使代码与提供者无关。

被管理对象是 MQ 系统对象，由 MQ 管理员创建并配置。这些对象存放在 JNDI 目录服务中， JMS 客户机使用 JNDI 查找来访问这些对象。

MQ 被管理对象也可以由客户机对其进行实例化，而不是在 JNDI 目录服务中查找。但这样做的缺点是要求应用程序开发者使用提供者特有的 API，并降低了 MQ 管理员成功控制和管理 MQ 消息服务器的能力。

有关被管理对象的详细信息，请参见第 71 页上的 “MQ 被管理对象”。

编程域

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） 消息从一个生成方传送至一个使用方。在此传送模型中，目标是一个**队列**。消息首先被传送至队列目标，然后根据队列传送策略从此队列 （请参见第 62 页上的 “队列目标”）将消息传送至向此队列进行注册的某一个使用方，一次只传送一条消息。可以向队列目标发送消息的生成方的数量没有限制，但每条消息只能发送至、并由**一个**使用方成功接收。如果没有已向队列目标注册的使用方，队列将保留它收到的消息，并在某个使用方向该队列进行注册时将消息传送给该使用方。

发布/订阅 （主题目标） 消息从一个生成方传送至任意数量的使用方。在此传送模型中，目标是一个**主题**。消息首先被传送至主题目标，然后传送至**所有已订阅**此主题的活动使用方。可以向主题目标发送消息的生成方的数量没有限制，并且每个消息可以发送至任意数量的订阅使用方。主题目标也支持**长期订阅**的概念。长期订阅表示使用方已向主题目标进行注册，但在消息传送时此使用方可以处于非活动状态。当此使用方再次处于活动状态时，它将接收此信息。如果没有已向主题目标注册的使用方，主题不保留其接收到的消息，除非有非活动使用方注册了长期订阅。

这两种消息传送模型使用表示不同编程域的 API 对象 （其语义稍有不同）进行处理，如表 1-1 所示。

表 1-1 JMS 编程对象

基本类型 （统一域）	点对点域	发布/订阅域
Destination （Queue 或 Topic）[1]	Queue	Topic
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Connection	QueueConnection	TopicConnection
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver	TopicSubscriber

1 取决于编程方法，您可以指定特定的目标类型。

可以使用表 1-1 第一列中列出的统一域对象编写点对点和发布 / 订阅消息传送。这是首选方法。然而，为了符合早期的 JMS 1.02b 规范，可以使用点对点域对象编写点对点消息传送，使用发布 / 订阅域对象编制发布 / 订阅消息传送。

客户机标识符

JMS 提供者必须支持**客户机标识符**的概念，客户机标识符将 JMS 客户机至消息服务的连接与消息服务代表客户机维护的状态信息关联起来。按照定义，客户标识符是唯一的，并且每次只应用到一个用户。客户机标识符与长期订阅名称（请参见第 37 页上的 “发布 / 订阅 （主题目标）”）结合使用，确保每个长期订阅只对应一个用户。

JMS 规范允许客户机通过 API 方法调用来设置客户机标识符，但最好使用连接工厂被管理对象 （请参见第 33 页上的 “被管理对象”）从管理级别来设置它。但如果硬链接至连接工厂，那么每个用户均需要单独的连接工厂以拥有一个唯一的标识。

使用可在 `ConnectionFactory` 对象中配置的特殊变量替代语法，MQ 提供了一个途径，使客户机标识符特定于 ConnectionFactory 和用户。使用此方式时，创建长期订阅的多个用户可以使用一个 `ConnectionFactory` 对象，而不必担心命名冲突或安全性得不到保障。用户的长期订阅将因此得到保护，不会因另一用户设置了错误的客户机标识符而导致意外删除或不可用。

有关如何使用此 MQ 功能的详细信息，请参见 《MQ *Developer's Guide*》中连接工厂属性的介绍。

无论何时，如果要创建长期订阅，客户标识符必须由客户机使用 JMS API 在编写程序时设置，或者在客户机使用的 `ConnectionFactory` 对象中进行管理级别的配置。

可靠消息传送

JMS 定义了两种**传送模式**：

持久性消息　保证这些消息只被传送一次和成功接收一次。对于这些消息，可靠性是优先考虑的因素。

非持久性消息　保证这些消息最多被传送一次。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。一个是确保至目标和出自目标的持久性消息传送成功。另一个就是确保在将持久性消息传送至使用方前，消息服务没有丢失持久性消息。

确认/事务

可靠的消息传送的关键在于确保至目标和出自目标的持久性消息传送成功。可通过 MQ 会话支持的两个通用机制：确认或事务来实现此目的。事务 （可以是本地事务或分布式事务）是由分布式事务管理器控制。

确认

可以将会话配置为使用确认来确保可靠传送。

对于生成方，这意味着在生成方的 `send()` 方法返回前，消息服务确认已向其目标发送持久性消息。对于使用方，这意味着在消息服务从该目标删除持久性消息前，客户机确认从此目标发出的持久性消息已传送并已接收。

本地事务

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

在事务中生成或接收消息时，代理跟踪各个发送和接收过程，并在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，将完成所有成功的操作。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成方或使用方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接收的端对端事务。（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

MQ 也支持**分布式**事务。也就是说，消息的生成和接收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 （例如数据库系统）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多个资源管理器 （例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，MQ 代理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。

MQ 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 （请参见第 32 页上的 “JMS 编程模型”）。另外要支持分布式事务，还需要第三方 JTS 或提供 JTS 的 J2EE 兼容应用服务器。

持久性存储器

可靠性的另一个重要方面是确保持久性消息传送至目标后，消息服务在向使用方传送它们之前不会丢失这些消息。这意味着在持久性消息传送至目标时，消息服务必须将其放入持久性数据存储 （请参见第 52 页上的 “持久性管理器”）。如果消息服务由于某种原因导致失败，它可以恢复此消息并将此消息传送至相应的使用方。虽然这样增加了消息传送的开销，但却增加了可靠性。

消息服务还必须存储长期订阅。这是因为要确保主题目标模式下的可靠传送，仅仅恢复持久性消息是不够的。消息服务还必须恢复关于主题的长期订阅的信息，否则它就不能把消息传送至在消息到达时处于非活动状态、随后又恢复活动状态的订户。

需要可靠消息传送的消息传送应用程序必须将消息指定为持久性消息，并使用队列目标或主题目标的长期订阅。

性能折衷

消息传送的可靠性越高，需要的开销和带宽就越多。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。您可以选择生成和接收非持久性消息来获得最佳**性能**。另一方面，您可以通过生成和接收持久性消息并使用事务会话来获得最佳**可靠性**。在这两种极端之间有许多选择，这要取决于应用程序的要求，包括使用 MQ 特有的连接和确认属性，请参见 《MQ *Developer's Guide*》。

消息选择

JMS 提供了一种机制，使用它，消息服务可根据消息选择器中的标准来执行消息过滤和路由。生成方客户机可在消息中放入应用程序特有的属性，而使用方客户机可使用基于这些属性的选择标准表明对消息是否感兴趣。这就简化了客户机的工作，并避免了向不需要这些消息的使用方传送消息的开销。然而，它也使得处理选择标准的消息服务增加了一些额外开销。在 JMS 规范中对消息选择器语法和语义进行了简要说明。

消息顺序和优先级

通常，可以确保将单个会话向目标发送的所有消息按其发送顺序传送至使用方。然而，如果为这些消息分配了不同的优先级，消息传送系统将首先尝试传送优先级较高的消息。

除此之外，客户机应用程序接收消息的顺序只与消息的生成顺序存在一个粗略的关系。这是因为向目标传送消息和从目标传送消息可以取决于多种影响时间的因素，例如消息的发送顺序、发送消息的会话 （连接）、是否是持久性消息、消息的有效期、消息的优先级、队列目标的消息传送规则 （请参见第 62 页上的 “队列目标”）和消息服务可用性。

对于使用多个互连代理 （请参见第 64 页上的 “多代理群集 （企业版）”）的 MQ 消息服务来说，客户机接收消息的顺序更为复杂，这是因为从不同代理的目标传送消息的顺序是不确定的。因此，一个代理息可能会先于另一个代理传送消息，即使是后者先接收到消息。

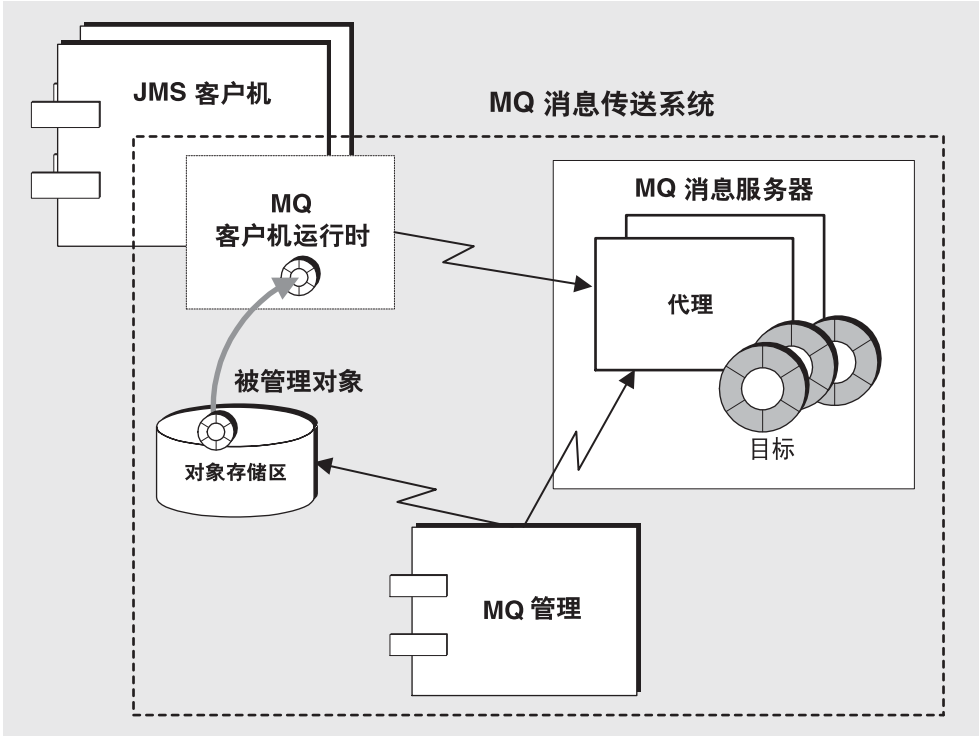
无论何时，对于一个给定的使用方，优先级较高的消息总是先于优先级较低的消息。

第 2 章

MQ 消息传送系统

本章介绍 Sun™ ONE Message Queue (MQ) 消息传送系统，其中重点说明的是系统的主要组成部分 （如图 2-1 所示），并阐述了这些组成部分如何配合工作来提供可靠的消息传送。

图 2-1 MQ 系统的体系结构



MQ 消息传送系统的主要组成部分 （如图 2-1 所示）包括：

- MQ 消息服务器
- MQ 客户机运行时
- MQ 被管理对象
- MQ 管理

前三个组成部分将在下面的小节中进行详细说明。最后一个部分将在第 3 章 “MQ 管理”中介绍。

MQ 消息服务器

本节介绍 MQ 消息服务器 （如第 43 页上的图 2-1 所示）的各个组成部分，包括：

代理 MQ 代理为 MQ 消息传送系统提供传送服务。消息传送依赖于大量支持组件，这些组件负责处理连接服务、消息的路由和传送、持久性、安全性以及日志记录 （有关详细信息，请参见第 44 页上的 “代理”）。消息服务器可以使用一个或多个代理实例 （请参见第 64 页上的 “多代理群集 （企业版）”）。

物理目标 消息的传送过程分为两个阶段：首先从生成消息的客户机 （生成方客户机）将消息发送到由代理维护的物理目标，然后再从目标发送到一个或多个使用消息的客户机 （使用方客户机）。物理目标是指代理的物理内存和 / 或持久性存储器中的位置 （有关详细信息，请参见第 62 页上的 “物理目标”）。

代理

在 MQ 消息传送系统中，消息传送 （从生成方客户机到目标，然后再从目标到一个或多个使用方客户机）由代理 （或以串连模式工作的代理实例群集）来执行。要执行消息传送，代理必须建立与客户机之间的通信通道、执行验证和授权、适当的路由消息、确保可靠传送并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组件，每个组件都在传送过程中担当着各自不同的角色。可以根据负荷条件、应用程序的复杂性等因素配置这些内部组件，以优化代理的性能。图 2-2 所示为主要的代理组件，表 2-1 则对这些组件进行了简要说明。

图 2-2 代理组件

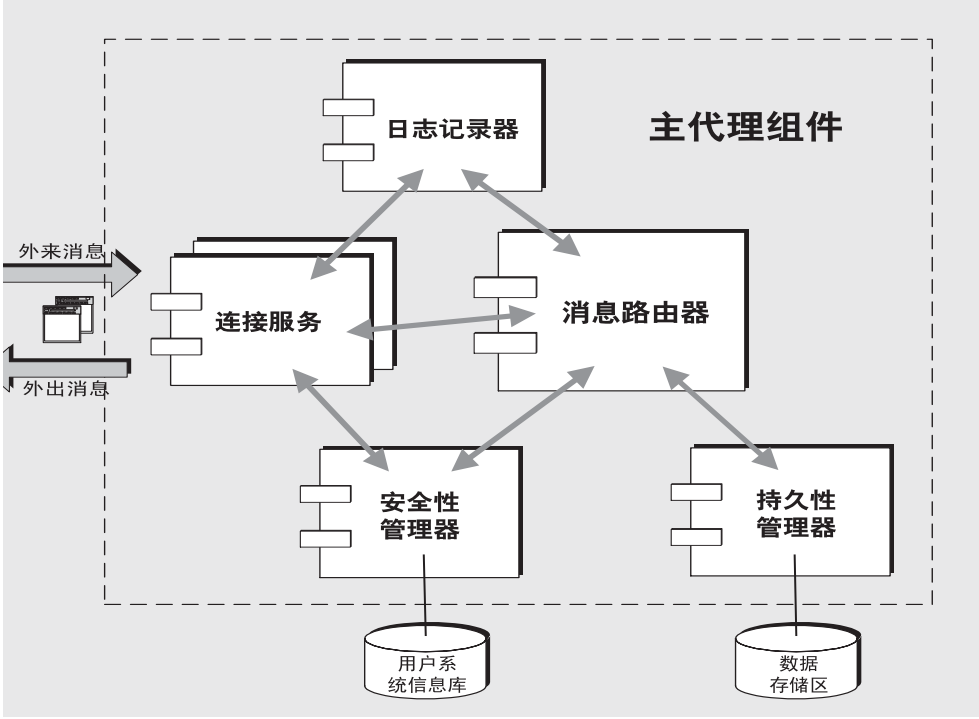


表 2-1 主要代理组件及功能

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，用于传输外来和外出消息。
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 MQ 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）。
日志记录器	将监视和诊断信息写入日志文件或控制台，这样管理员便可以监视和管理代理。

下面的几个小节将详细介绍各代理组件执行的功能及其属性 （可以配置这些属性来影响代理的行为）。

连接服务

MQ 代理既支持与 JMS 客户机的通信，也支持与 MQ 管理客户机的通信 （请参见第 77 页上的 “MQ 管理工具”）。各个服务取决于其服务类型和协议类型。

服务类型　指定服务提供的是 JMS 消息传送 (NORMAL) 服务，还是 MQ 管理 (ADMIN) 服务。

协议类型　指定支持服务的基本传输协议层。

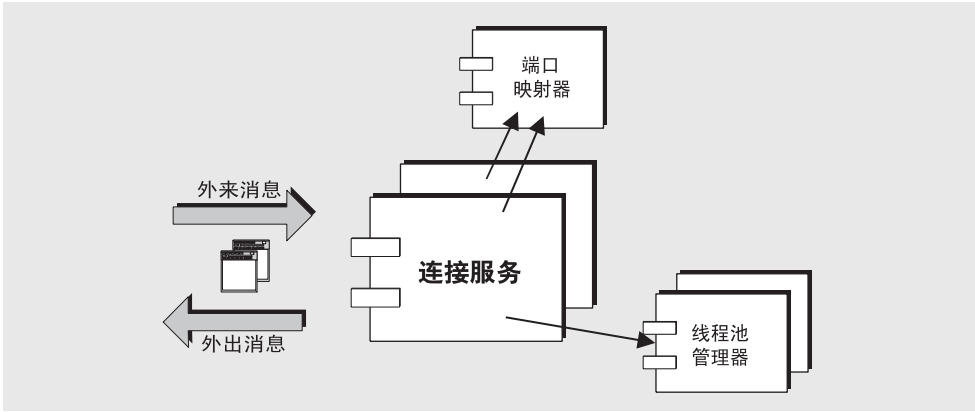
表 2-2 所示为 MQ 代理当前可提供的连接服务：

表 2-2　代理支持的连接服务

服务名称	服务类型	协议类型
jms	NORMAL （JMS 消息传送）	tcp
ssljms （企业版）	NORMAL （JMS 消息传送）	tls （其安全性基于 SSL）
httpjms （企业版）	NORMAL （JMS 消息传送）	http
httpsjms （企业版）	NORMAL （JMS 消息传送）	https （其安全性基于 SSL）
admin	ADMIN	tcp
ssladmin （企业版）	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。每个服务都具有一个线程池管理器，并向通用端口映射器服务注册它本身，如图 2-3 所示。

图 2-3 连接服务支持



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。可以静态或动态地分配端口。 MQ 提供了一个**端口映射器**，用于将动态分配的端口映射到不同的连接服务。端口映射器本身位于标准端口号 7676。当客户机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。

在配置 jms、 ssljms、 admin 和 ssladmin 等连接服务时，也可以为它们分配**静态**端口号，但最好不要这样做。分别使用第 201 页上的表 B-1 和第 209 页上的表 B-3（这两个表格位于附录 B “HTTP/HTTPS 支持 （企业版）”。）中说明的属性来配置 httpjms 和 httpsjms 服务。

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。您希望此值足够大，这样就不必不断创建新线程。如果连接负荷较重，线程数量可能会增加，直到达到线程池的最大数量。此后，连接不得不一直等待，直到某个线程可用。

线程池中的线程可专用于单个连接 （**专用**模型），也可根据需要分配到多个连接（**共享**模型）。

专用模型 在专用模型中，每个与代理的连接都需要两个线程：一个专门用来处理外来消息，另一个用来处理外出消息。这样，连接的数量就被限制为线程池中最大线程数量的一半，但却提供了较高的性能。

共享模型 （企业版） 在共享线程模型中，只有当发送或接收消息时，连接才会被分配到某个线程。此模型 （其中多个连接共享一个线程）增加了连接服务可支持的连接的数量，从而增加了代理可支持的连接的数量，但这同时带来了某些性能开销。线程池管理器使用一组分配器线程来监视连接活动并根据需要将连接分配到线程。可以限制每个这样的分配器线程所监视的连接的数量来提高性能。

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 55 页上的“安全性管理器”）。

表 2-3 所示为与连接服务相关的可配置属性。（有关配置这些属性的说明，请参见第 5 章 “启动与配置代理”。）

表 2-3 连接服务属性

属性名称	说明
`imq.service.activelist`	按名称列出启动代理时将变为活动状态的连接服务，服务之间用逗号分隔。支持的服务包括：`jms`、`ssljms`、`httpjms`、`httpsjms`、`admin` 和 `ssladmin`。默认值：`jms` 和 `admin`
`imq.`*service_name*`.min_threads`	指定线程数量，达到该数量后，线程池中将维护此数量的线程供命名的连接服务使用。默认值：取决于连接服务 （请参见第 110 页上的表 5-1）。
`imq.`*service_name*`.max_threads`	指定线程数量，达到该数量后，新的线程将不会被添加到线程池中供命名的连接服务使用。该数量必须大于零，并且必须大于 `min_threads` 的值。默认值：取决于连接服务 （请参见第 110 页上的表 5-1）。
`imq.`*service_name*`.threadpool_model`	为命名的连接服务指定线程是专用于特定的连接 (`dedicated`)，还是根据需要由多个连接共享 (`shared`)。共享模型 （线程池管理）将增加代理支持的连接数量，但只能用于 `jms` 和 `admin` 连接服务。默认值：取决于连接服务 （请参见第 110 页上的表 5-1）。
`imq.shared.connectionMonitor_limit`	仅适用于共享线程池模型，用于指定分配器线程可监视的连接的最大数量。（系统会分配足够多的分配器线程来监视所有连接。）此值越小，系统向线程分配活动连接的速度就越快。值 `0` 表示没有限制。默认值：取决于操作系统 （请参见第 110 页上的表 5-1）。
`imq.portmapper.port`	代理的主端口，即端口映射器所在的端口。如果主机上运行多个代理实例，则必须为每个实例分配唯一的端口映射器端口。默认值：`7676`

表 2-3 连接服务属性 （续）

属性名称	说明
`imq.`*service_name*`.`*protocol_type*[1]`.port`	仅适用于 jms、 ssljms、 admin 和 ssladmin 服务，指定命名的连接服务的端口号。默认值：`0` （端口由端口映射器动态分配） 要配置 httpjms 和 httpsjms 连接服务，请参见附录 B “HTTP/HTTPS 支持 （企业版）”。
`imq.`*service_name*`.`*protocol_type*[1]`.hostname`	仅适用于 jms、 ssljms、 admin 和 ssladmin 服务，如果有多个主机可用 （例如，一台计算机中安装了多个网络接口卡），命名的连接服务将绑定到此属性指定的主机（主机名或 IP 地址）。 默认值：`null` （任意主机）

1 protocol_type 为表 2-2 中指定的协议类型。

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。

基本传送机制

广义来讲，代理处理的消息可分为两类：由生成方客户机发往使用方客户机的 JMS 消息 （即有效负荷消息）以及从客户机发出或发往客户机的大量用于支持 JMS 消息传送的控制消息。

如果外来消息为 JMS 消息，代理会根据其目标的类型 （队列或主题）将其发送到使用方客户机：

- 如果目标为主题， JMS 消息将立即被路由到此主题的所有活动订户。对于未处于活动状态的长期订户，消息路由器将保留消息，直到订户变为活动状态后再将消息传送给该订户。
- 如果目标为队列， JMS 消息将被放置在相应的队列中，并在到达队列前端时被传送到相应的使用方。消息到达队列前端的顺序取决于它们进入队列的顺序以及它们的优先级。

消息路由器将消息传送到其所有目标使用方后，将从内存中删除此消息。如果该消息为持久性消息 （请参见第 38 页上的 “可靠消息传送”），还会将其从代理的持久性数据存储中删除。

可靠传送：确认和事务

如果增加**可靠**传送 （请参见第 38 页上的 “可靠消息传送”）的要求，则刚才讨论的传送机制就会复杂得多。可靠传送涉及两个方面：一是确保消息成功传送到代理以及从代理成功传送出去，二是确保在实际传送消息前，代理没有丢失消息或传送信息。。

为了确保至代理和出自代理的消息传送能够成功， MQ 使用了大量称为 “确认”的控制消息。

例如，当生成方向目标发送 JMS 消息 （有效负荷消息而非控制消息）时，代理会发回一条说明它已接收到 JMS 消息的控制消息，这条控制消息就是代理确认。（实际上，只有当生成方将 JMS 消息指定为持久性消息时， MQ 才会进行上述操作。）生成方客户机使用代理确认来确保至目标的传送成功 （请参见第 69 页上的 “消息生成”）。

同样，当代理将 JMS 消息传送到使用方时，使用方客户机会发回一个确认，说明它已接收并处理了此消息。客户机可以在创建会话对象时指定发送这些确认的自动程度和频率，但原则是消息路由器在收到每个消息使用方 （即接收它所传送的消息的客户机，例如某个主题的每个订户）的确认之前，不会从内存中删除 JMS 消息。

对于某个主题的长期订户，消息路由器会将每条 JMS 消息保留在该目标中，以便在每个长期订户变为活动使用方时将消息传送出去。消息路由器会在接收客户机确认时进行记录，并且只有在接收到所有确认后才会删除 JMS 消息（除非在这之前 JMS 消息已过期）。

另外，消息路由器会将代理确认发送回客户机，确认已接收到客户机确认。使用方客户机使用代理确认来确保代理不会多次传送 JMS 消息 （请参见第 70 页上的 “消息使用”）。如果由于某些原因使代理未能接收到客户机确认，则可能发生多次传送的情况。

如果代理没有接收到客户机确认并重新传送了一次 JMS 消息，此消息将被标注 “重新传送”标志。通常，如果在代理接收到客户机确认之前客户机连接被关闭，并且随后又打开了新的连接，那么代理会重新传送 JMS 消息。例如，如果队列的消息使用方在确认消息之前脱机，随后另一位使用方向队列进行了注册，则代理会向该新使用方重新传送未确认的消息。

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。对于后者，客户机和代理确认在事务级别以及各个 JMS 消息发送或接收级别运行。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 39 页上的“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。

可靠传送：持久性

可靠传送的另一个方面是确保在实际传送消息前，代理没有丢失消息或要传送的信息。通常，消息将始终保留在内存中，直到被传送或过期。但如果代理出现故障，这些消息将会丢失。

生成方客户机可以将消息指定为持久性消息，在这种情况下，消息路由器会将此消息传送到**持久性管理器**，该管理器将把消息存储在数据库或文件系统中（请参见第 52 页上的“持久性管理器”），这样即使代理出现故障，消息也可以得到恢复。

管理系统资源

代理的性能取决于可用的系统资源以及资源（例如内存）的使用效率。例如，消息路由器具有用于监视系统内存的内存管理机制。当内存资源不足时，将激活释放内存并减慢外来消息流的机制。

内存管理机制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）。随着内存资源状态由 green 变为 yellow 再变为 orange 最后变为 red，代理所采取的措施也会越来越严格：首先是释放内存，然后是限制消息生成方，最后甚至会停止进入代理的消息流。

使用用于对内存中的消息总数以及总大小设置限制的属性以及用于调整占用阈值（达到该阈值后，内存资源将变为一个新状态）的属性，您可以配置代理的内存管理功能。

表 2-4 中详细介绍了这些属性。（有关设置这些属性的说明，请参见第 5 章 “启动与配置代理”。）

表 2-4 消息路由器属性

属性名称	说明
imq.message.expiration.interval	指定过期消息的收回频率 （以秒为单位）。默认值：60
imq.system.max_count	指定内存和磁盘中的消息最大数量 （由于交换）。超出限制的消息将被拒绝。值 0 表示没有限制。默认值：0
imq.system.max_size	指定内存和磁盘中的消息最大总大小 （由于交换），以字节、千字节或兆字节为单位。超出限制的消息将被拒绝。值 0 表示没有限制。默认值：0
imq.message.max_size	指定消息主体允许的最大大小 （以字节、千字节或兆字节为单位）。任何大于此值的消息将被拒绝。值 0 表示没有限制。默认值：70m （兆字节）
imq.*resource_state*.threshold	指定将触发各个内存资源状态的内存占用百分比。资源状态的值可以是 green、 yellow、 orange 和 red。默认值：分别为 0、 60、 75 和 90
imq.redelivered.optimization	指定 (true/false) 消息路由器是在重新传送消息时通过设置重新传送标志来优化性能 (true)，还是仅当逻辑上需要时这样做 (false)。默认值： true
imq.transaction.autorollback	指定 (true/false) 当启动代理时，处于 PREPARED 状态的分布式事务是否自动回滚。如果为 false，则必须使用 imqcmd 手动提交或回滚事务 （请参见第 147 页上的 “管理事务”）。默认值： false

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中。**持久性管理器**组件用于管理此信息的写入和检索。

为了恢复出现故障的代理，不仅需要恢复未传送的消息，代理还必须完成以下操作：

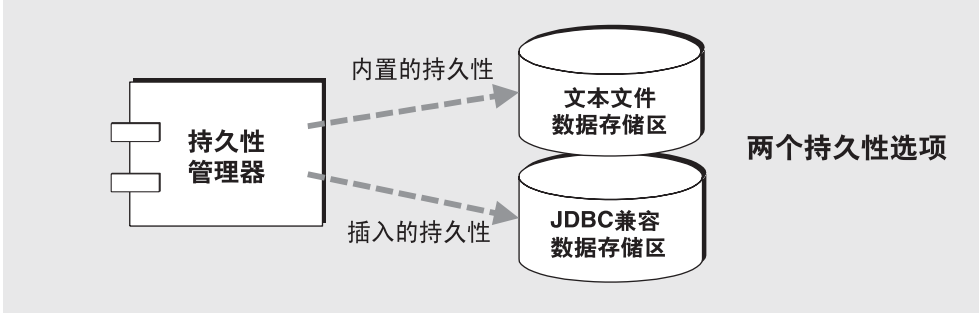
- 重新创建目标
- 恢复每个主题的长期订阅列表
- 恢复每条消息的确认列表
- 重新生成所有已提交事务的状态

持久性管理器用于管理所有这些状态信息的存储和检索。

当代理重新启动时，它将重新创建目标和长期订阅，恢复持久性消息和所有事务的状态，并重新创建未传送消息的路由表。然后代理才可以恢复消息传送。

MQ 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 2-4）。内置的持久性基于文本文件数据存储。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。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使用 JDBC 兼容的数据库系统的冗余和管理功能。

图 2-4 持久性管理器支持



内置的持久性

MQ 持久性存储器的默认解决方案是文本文件存储。该解决方案使用单个文件来存储持久性数据，例如消息、目标、长期订阅和事务等。

文本文件数据存储位于：

`IMQ_VARHOME/instances/`*`brokerName`*`/filestore/`
（在 Solaris 中为 `/var/imq/instances/`*`brokerName`*`/filestore/`）

其中 *brokerName* 为标识代理实例的名称。

基于文件的数据存储的结构是每条持久性消息都存储在各自的文件中，每个文件存储一条消息。不过，目标、长期订阅和事务都是每一类存储到一个单独的文件中。例如，所有目标存储到一个文件，所有长期订阅存储到另一个文件，依此类推。

创建或删除文件时，因为要将消息添加到数据存储或从数据存储中删除消息，所以涉及大量的文件系统操作。因此 MQ 实现方案会重复使用这些消息文件：当不再需要某个文件时，将把它添加到可供重复使用的空闲文件池中，而不是将其删除。可以配置此文件池的大小。也可以指定文件池中空闲文件所占的百分比，达到此百分比后将清除文件池 （清空），而不是仅标记这些空闲文件供重复使用 （不清空）。此清除文件的百分比越高，文件池所需的磁盘空间越小，但维护文件池所需的开销就越大。另外，也可以指定关闭时是否清除标记的文件。如果清除这些文件，它们所占用的磁盘空间就会减少，但代理将需要更长的时间来进行关闭操作。

在文本文件存储中，存储消息的速度受数据存储能够使用的文件描述符数量的影响，描述符数量越多，系统处理大量持久性消息的速度就越快。有关增加文件描述符数量的信息，请参见 《MQ **发行说明**》中的 “技术说明”一节。

另外，对于目标文件存储，更高效的做法是将目标添加到固定大小的文件中，而不是随着目标不断添加而增加文件的大小。因此，可以根据最终要存储的目标数量来设置目标文件的原始大小，从而提高性能 （每个目标占用大约 500 字节）。

由于数据存储可能包含具有专利信息的消息，所以建议您对 *brokerName*`/filestore/` 目录进行保护，以防止未获授权的访问。有关说明请参见 《*MQ* **发行说明**》中的 “技术说明”一节。

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。这将设计到设置多个与 JDBC 相关的代理配置属性以及使用数据库管理器实用程序 (`imqdbmgr`) 来创建具有相应架构的数据存储。附录 A “设置插入的持久性”中详细说明了上述过程和相关的配置属性。

第 54 页上的表 2-5 详细说明了与持久性相关的配置属性。（有关设置这些属性的说明，请参见第 5 章 “启动与配置代理”。）

表 2-5 持久性属性

属性名称	说明
`imq.persist.store`	指定代理是使用内置的、基于文件的 (`file`) 持久性，还是使用插入的、JDBC 兼容的 (`jdbc`) 持久性。默认值：`file`
`imq.persist.file.destination.file.size`	适用于内置的、基于文件的持久性，指定用于存储目标的文件的原始大小。默认值：`1m` （兆字节）

表 2-5 持久性属性 （续）

属性名称	说明
imq.persist.file.message.filepool.limit	适用于内置的、基于文件的持久性，指定文件池中可供重复使用的空闲文件的最大数量。该数量越大，代理处理持久性数据的速度越快。超过该值的空闲文件将被删除。代理将根据需要创建文件或删除超过此限制的文件。默认值：10000
imq.persist.file.message.filepool.cleanratio	适用于内置的、基于文件的持久性，指定保持**清除**状态 （清空）的文件池中空闲文件的百分比。该值越大，在操作中清除文件所需的开销越大，但文件池所需的磁盘空间越小。默认值：0
imq.persist.file.message.cleanup	适用于内置的、基于文件的持久性，指定代理在关闭时是否清除文件存储中的空闲文件。如果值为 false，则关闭的速度会加快，但文件存储需要的磁盘空间会更多。默认值：false
imq.persist.file.message.fdpool.limit	适用于内置的、基于文件的持久性，指定要保持打开状态的数据文件的最大数量 （即文件描述符池的大小）。该值越大，持久性操作的性能越良好，但会降低其它需要文件描述符的代理操作 （例如创建客户机连接）的性能。默认值：25 （Solaris 和 Linux）、1024 (Windows)
imq.persist.file.sync.enabled	指定持久性操作是否使内存中状态与物理存储设备同步。如果为 true，则可以避免由于系统崩溃导致的数据丢失，但持久性操作的性能会有所降低。默认值：false

安全性管理器

MQ 提供验证和授权 （访问控制）功能，同时也支持加密功能。

验证和授权功能取决于用户系统信息库 （请参见第 56 页上的图 2-5）：包含消息传送系统的用户信息 （用户名、口令和组成员资格）的文件、目录或数据库。用户名和口令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。

MQ 管理员可填充 MQ 提供的用户系统信息库 （请参见第 170 页上的 “使用文本文件用户系统信息库”），也可以将原有的 LDAP 用户系统信息库插入到安全性管理器组件中。文本文件用户系统信息库使用起来较简单，但在安全性上也易遭攻击，因此应**只**将其用于测试和开发目的，而 LDAP 用户系统信息库相对较安全，因此更适合用于生产目的。

验证

MQ 安全性支持基于口令的验证。当客户机请求连接到代理时，该客户机必须提交用户名和口令。安全性管理器将把客户机提交的用户名和口令与用户系统信息库中存储的用户名和口令进行比较。口令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 57 页上的“加密 （企业版）”。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。

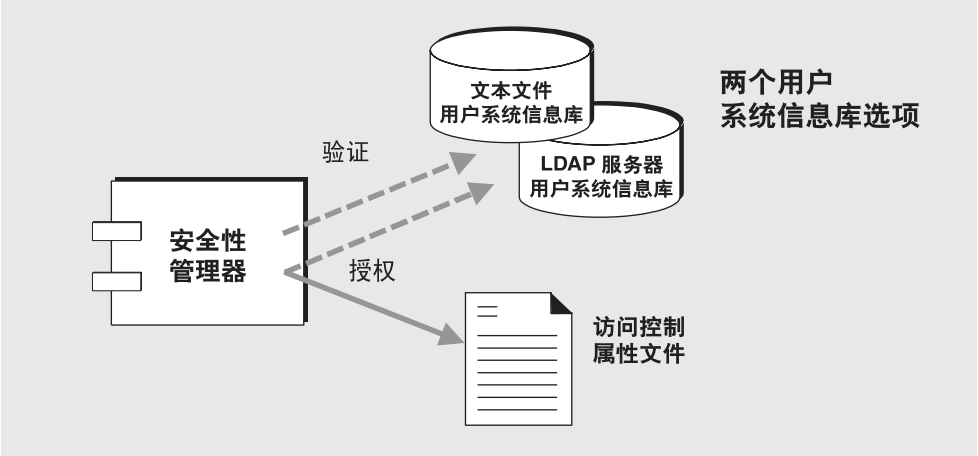
授权

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 MQ 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 MQ 操作的权限也不同。这些访问控制在访问控制属性文件中指定 （请参见图 2-5）。

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（通过用户系统信息库）是否属于被指定允许访问该操作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：

- 建立与代理的连接
- 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

图 2-5 安全性管理器支持



对于 MQ 3.0.1，默认的访问控制属性文件只明确引用一个组：**管理员** （参见第 173 页上的 “组”）。**管理员**组中的用户拥有管理服务连接权限。管理服务使用户可以执行管理功能，例如创建目标以及监视和控制代理。默认情况下，您定义的其它任何组中的用户都无法获得管理服务连接。

作为 MQ 管理员，可以在用户系统信息库中定义组并使用户与这些组关联 （虽然在文本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，可以按照用户和组的不同目的 （生成和使用消息，或浏览队列目标中的消息）指定对目标的权限。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 63 页上的 “自动创建的（与管理员创建的相对）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 （企业版）

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。

为了使用 MQ 基于 SSL 的连接服务，需要使用密钥工具实用程序 (`imqkeytool`) 生成专用密钥/公用密钥对。此实用程序将公用密钥嵌入自签名的证书中，并将其放入 MQ 密钥存储中。 MQ 密钥存储本身设有口令保护，要解除锁定，需要在启动时提供密钥存储的口令。请参见第 183 页上的 “加密：使用 SSL 服务 （企业版）”。

解除密钥存储的锁定后，代理即可将证书传送给所有请求连接的客户机。然后，客户机使用此证书建立与代理的加密连接。

表 2-6 所示为验证、授权、加密以及其它安全性通信的可配置属性。（有关配置这些属性的说明，请参见第 5 章 “启动与配置代理”。）

表 2-6 安全性属性

属性名称	说明
`imq.authentication.type`	指定传送口令时应使用 Base 64 编码 (`basic`) 还是 MD5 摘要 (`digest`)。设置代理支持的所有连接服务的编码。默认值：`digest`
`imq.`*service_name*`.authentication.type`	指定传送口令时应使用 Base 64 编码 (`basic`) 还是 MD5 摘要 (`digest`)。设置命名的连接服务的编码，此设置将覆盖任意代理范围的设置。 默认值：从 `imq.authentication.type` 的值继承。

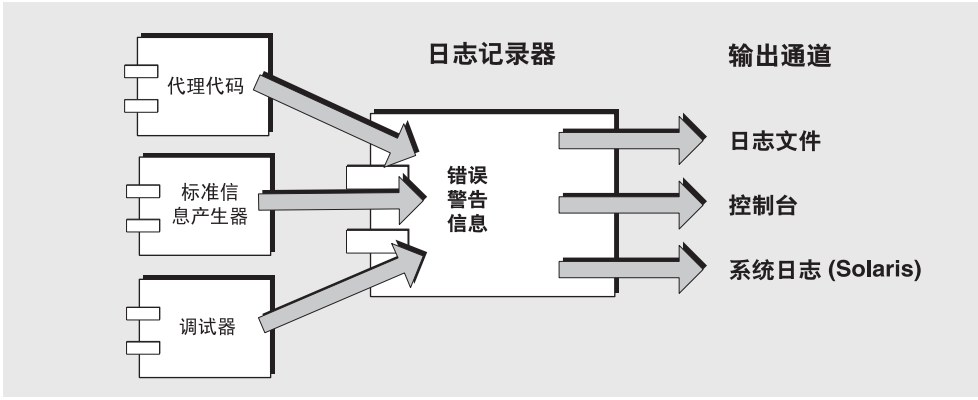
表 2-6 安全性属性 （续）

属性名称	说明
`imq.authentication.basic.user_repository`	对于 Base 64 编码，指定用于验证的用户系统信息库的类型：基于文件 (`file`) 或 LDAP (`ldap`)。有关其它 LDAP 属性，请参见第 176 页上的表 8-5。默认值：`file`
`imq.authentication.client.response.timeout`	指定系统等待客户机响应来自代理的验证请求的时间（以秒为单位）。默认值：`180` （秒）
`imq.accesscontrol.enabled`	为代理支持的所有连接服务设置访问控制 (`true/false`)。表明系统是否要检查已验证的用户是否如访问控制属性文件所指定，拥有使用连接服务或执行与特定目标有关的特定 MQ 操作的权限。 默认值： `true`
`imq.`*service_name*`.accesscontrol.enabled`	为命名的连接服务设置访问控制 (`true/false`)，此设置将覆盖代理范围的设置。表明系统是否要检查已验证的用户是否如访问控制属性文件所指定，拥有使用连接服务或执行与特定目标有关的特定 MQ 操作的权限。 默认值：继承 `imq.accesscontrol.enabled` 属性的设置
`imq.accesscontrol.file.filename`	为代理支持的所有连接服务指定访问控制属性文件的名称。文件名会指定相对于 `IMQ_HOME/etc` 目录的文件路径 （在 Solaris 中为 `/etc/imq`）。 默认值： `accesscontrol.properties`
`imq.`*service_name*`.accesscontrol.file.filename`	为命名的连接服务指定访问控制属性文件的名称。文件名会指定相对于 `IMQ_HOME/etc` 目录的文件路径 （在 Solaris 中为 `/etc/imq`）。默认值： 继承由 `imq.accesscontrol.file.filename` 指定的设置。
`imq.passfile.enabled`	指定 (true/false) 是否在密码文件中指定用于安全性通信的用户口令 （适用于 SSL、 LDAP 和 JDBC）。 默认值：`false`
`imq.passfile.dirpath`	指定包含密码文件的目录的路径。默认值：`IMQ_HOME/etc` （在 Solaris 中为 `/etc/imq`）
`imq.passfile.name`	指定密码文件的名称。默认值：`passfile`
`imq.keystore.`*property_name*	适用于基于 SSL 的服务：指定与 SSL 密钥存储相关的安全性属性。请参见第 185 页上的表 8-8。

日志记录器

代理包含大量用于监视和诊断其操作的组件。这些组件包括用于生成数据 （代理代码、测量生成器和调试器）的组件以及用于通过多种输出通道 （日志文件、控制台和 Solaris 系统日志）记录输出信息的日志记录器组件。图 2-6 是日志记录器的图解。

图 2-6 日志记录图解



可以打开或关闭测量数据的生成，并指定生成测量报告的频率。

还可以指定日志记录器级别范围：从最严重和最重要的信息 （错误）到次要信息（测量数据）。表 2-7 所示为信息的种类 （按重要性的降序排列）。

表 2-7 日志种类

种类	说明
ERROR	指出可能导致系统故障的问题的消息
WARNING	需要注意但不会导致系统故障的警报
INFO	测量及其它信息性消息的报告

要设置日志记录器级别，需要指定其中的某个种类。日志记录器将记录指定种类及所有更高级别种类的数据。例如，如果指定记录 WARNING 级别的信息，则日志记录器将记录警告信息和错误信息。

日志记录器可以将数据写入到多个输出通道：标准输出 （控制台）、日志文件以及 Solaris 平台上的系统日志守护程序进程。

可以分别指定写入到每个输出通道的日志记录器的种类集。例如，如果将日志记录器级别设置为 ERROR，则可以指定只将错误和警告写入控制台，只将信息 （测量数据）写入日志文件。有关配置和使用 Solaris 系统日志的信息，请参见 syslog(1M)、syslog.conf(4) 和 syslog(3C) 手册页。

对于日志文件，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。日志文件保存在以下位置：

IMQ_VARHOME/instances/*brokerName*/log/
（在 Solaris 上为 /var/imq/instances/*brokerName*/log /）

创建新的转移日志文件时，将保留最新的 9 个日志文件的归档文件。日志文件为文本文件，依次命名为：

```
log.txt
log_1.txt
log_2.txt
...
log_9.txt
```

log.txt 是最新的文件，编号最大的文件是最早的文件。

表 2-8 所示为用于通过代理设置信息的生成和记录的可配置属性。（有关配置这些属性的说明，请参见第 5 章 “启动与配置代理”。）

表 2-8 日志记录器属性

属性名称	说明
imq.metrics.enabled	指定 (true/false) 是否要收集测量信息。默认值：true
imq.metrics.interval	指定报告测量信息的时间间隔 （以秒为单位）。值 0 表示不报告。默认值：0
imq.log.level	指定日志记录器级别：可以写入到输出通道的输出的种类。包括指定的种类以及所有更高级别的种类。值分别为（由高至低）：ERROR、WARNING 和 INFO。默认值：INFO

表 2-8 日志记录器属性 （续）

属性名称	说明
imq.log.file.output	指定写入到日志文件的日志记录信息的种类。允许的值为：由竖线 (|) 分隔的任意日志记录种类集，或 ALL 和 NONE。默认值：ALL
imq.log.file.dirpath	指定包含日志文件的目录的路径。默认值：IMQ_VARHOME/instances/*brokerName*/log/（在 Solaris 上为 /var/imq/instances/*brokerName*/log/）
imq.log.file.filename	指定日志文件的名称。默认值： log.txt
imq.log.file.rolloverbytes	指定日志文件的大小 （以字节为单位），达到该大小后输出将转移到新的日志文件。值 0 表示不进行基于文件大小的转移。默认值： 0
imq.log.file.rolloversecs	指定日志文件的生存期 （以秒为单位），达到该值后输出将转移到新的日志文件。值 0 表示不进行基于文件生存期的转移。默认值： 604800 （一周）
imq.log.console.output	指定将哪些种类的日志记录信息写入到控制台。允许的值为：由竖线 (|) 分隔的任意日志记录种类集，或 ALL 和 NONE。默认值：ERROR、 WARNING 和 INFO。
imq.log.console.stream	指定是否将控制台输出写入到标准输出 (OUT) 或标准错误 (ERR)。默认值：ERR
imq.log.syslog.facility	（仅对于 Solaris）指定 MQ 代理应将日志记录为哪种系统日志工具。属性值镜像 syslog(3C) 手册页中列出的值。适用于 MQ 的值包括：LOG_USER、LOG_DAEMON 和 LOG_LOCAL0 到 LOG_LOCAL7。默认值：LOG_DAEMON
imq.log.syslog.logpid	（仅对于 Solaris）指定 (true/false) 是否将代理进程 ID 与消息一起记录。默认值：true
imq.log.syslog.console	（仅对于 Solaris）指定 (true/false) 如果无法将消息发送到系统日志，是否将其写入到系统控制台。默认值：false
imq.log.syslog.identity	（仅对于 Solaris）此属性指定的标识字符串应添加到每个记录到系统日志的消息的前面。默认值：imqbrokerd_ 代理实例名称。
imq.log.syslog.output	（仅对于 Solaris）指定将哪些种类的日志记录信息写入到 syslogd(1M)。允许的值为：由竖线 (|) 分隔的任意日志记录种类，或 ALL 和 NONE。默认值：ERROR

物理目标

MQ 消息传送基于两个传送阶段：首先，消息从生成方客户机传送到代理上的目标，然后再从代理上的目标传送到一个或多个使用方客户机。目标分两种类型（参见第 37 页上的“编程域”）：队列（点对点传送模型）和主题（发布/订阅传送模型）。这些目标表示代理的物理内存中的位置（外来消息在被路由到使用方客户机前将保留在其中）。

可以使用 MQ 管理工具创建物理目标（请参见第 143 页上的“管理目标”）。也可以按照第 63 页上的“自动创建的（与管理员创建的相对）目标”中的介绍来自动创建目标。

本节介绍这两种类型的物理目标（队列和主题）的属性和行为。

队列目标

队列目标用于点对点消息传送。在这种消息传送中，会有多个使用方在目标中注册请求，但消息最终仅传送到其中的一个使用方。当来自生成方客户机的消息到达时，它们将排成队列的形式，然后发送到使用方客户机。

队列消息的路由取决于队列的传送策略。 MQ 执行三种队列传送策略：

- **单一** 此队列只能将消息路由到一个消息使用方。如果第二个消息使用方尝试向队列进行注册，它将被拒绝。如果注册的消息使用方断开了连接，则不再路由消息，并且会保存这些未传送的消息，直到注册了新的使用方。
- **故障切换（企业版）** 此队列可以将消息路由到多个消息使用方，但只有其主消息使用方（第一个向代理注册的使用方）断开连接时，消息才会被发往其它使用方。在这种情况下，消息将被转到下一个注册的消息使用方，即消息将继续被路由到下一个注册的消息使用方，直到该使用方也出现故障，依此类推。如果没有已注册的消息使用方，消息将被保存，直到某个使用方进行注册。
- **循环（共享）（企业版）** 此队列可以将消息路由到多个消息使用方。假设有多个使用方向队列进行了注册，第一个进入队列的消息将路由到第一个注册的消息使用方，第二个进入队列的消息将发往第二个注册的使用方，依此类推。其它消息按此顺序分别路由到同一使用方的集合。在使用方向队列进行注册前，如果已有大量消息排队等候，则会按批发送消息以避免一齐涌向某个使用方。如果有任意消息使用方断开连接，则本应路由到该使用方的消息将在其余的活动使用方之间重新进行分配。由于进了这样的重新分配，就不能保证消息传送到使用方的顺序与它们在队列中接收消息的顺序相同。

由于消息可以在队列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。为队列分配内存时，您一定希望分配得恰到好处，向一个队列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。出于灵活性的考虑，可以基于每个队列的负荷要求在创建队列时设置物理属性：队列消息的最大数量、分配给队列消息的最大内存和队列消息的最大大小 （请参见第 144 页上的表 6-10）。

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，消息最终将被传送到所有在目标中注册请求的使用方。当来自生成方的消息到达时，它们将被路由到所有订阅该主题的使用方。如果使用方已注册为长期订阅该主题，则当消息被传送到主题时并不需要这些使用方必须处于活动状态。代理将存储此消息，直到使用方再次处于活动状态，然后将消息传送给它。

通常，消息不会在主题目标中保留过长的时间，因此，内存资源通常也不是大问题。不过，您可以为目标收到的任意消息配置允许的最大大小 （请参见第 144 页上的表 6-10）。

自动创建的 （与管理员创建的相对）目标

JMS 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。因为目标可能消耗大量的资源 （取决于它们处理的消息的数量和大小以及注册的消息使用方的数量和长期性），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 MQ 管理员的标准做法。

不过，有些时候可能需要动态创建目标。例如，在开发和测试阶段，可能希望代理根据需要自动创建目标，而不需要管理员进行干预。

MQ 支持这种**自动创建**功能。启用自动创建后，每当 MessageConsumer 或 MessageProducer 尝试访问不存在的目标时，代理会自动创建一个目标。（客户机应用程序的用户必须拥有自动创建权限，请参见第 182 页上的 “目标自动创建访问控制”。）

不过，当自动 （而非明确）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 （由于支持目标需要消耗资源）。因此，当不再使用 MQ 自动创建的目标时，也就是当目标不再与消息使用方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。

可以使用表 2-9 中所示的属性来配置 MQ 消息服务器，以便启用或禁用自动创建功能。（有关配置这些属性的说明，请参见第 5 章 “启动与配置代理”。）

表 2-9 自动创建配置属性

属性名称	说明
`imq.autocreate.topic`	指定 (`true/false`) 是否允许代理自动创建主题目标。默认值：`true`
`imq.autocreate.queue`	指定 (`true/false`) 是否允许代理自动创建队列目标。默认值：`true`
`imq.queue.deliverypolicy`	指定自动创建的队列的默认传送策略。允许的值为：`single`（单一）、`round-robin`（循环 [共享]）或 `failover`（故障切换）。默认值：`single`

临时目标

某些客户机应用程序需要一个目标来接收对发送至其它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使用方），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。此外，也无法通过 MQ 管理工具来对它们进行管理。

多代理群集 （企业版）

MQ 企业版使用多个互连的代理实例 （代理群集）来支持消息服务器的实现方案。群集支持为消息服务器提供了可伸缩性。

随着连接到代理的客户机数量以及所传送的消息数量的不断增加，代理最终将超出资源限制 （例如文件描述符和内存限制）。适应不断增加的负荷的方法之一是将更多的代理 （即更多代理实例）添加到 MQ 消息服务器，从而将客户机连接和消息传送分布到多个代理。

也可以使用多代理优化网络带宽。例如，您可能希望在一组远程代理之间使用低速的长途网络链接，而在客户机与其各自的代理之间使用高速链接进行连接。

虽然使用代理群集还有其它原因 （例如，适应具有不同用户系统信息库的工作组或处理防火墙限制），但故障切换并不是其中的一个原因。群集中的某个代理不能用作其它发生故障的代理的自动备份。 MQ 版本 3.0.1 不支持代理的自动故障切换保护。（不过，可以将应用程序设计为使用多代理来实现自定义的故障切换机制。）

有关配置和管理代理群集的信息，请参见第 118 页上的 “使用群集 （企业版）”。

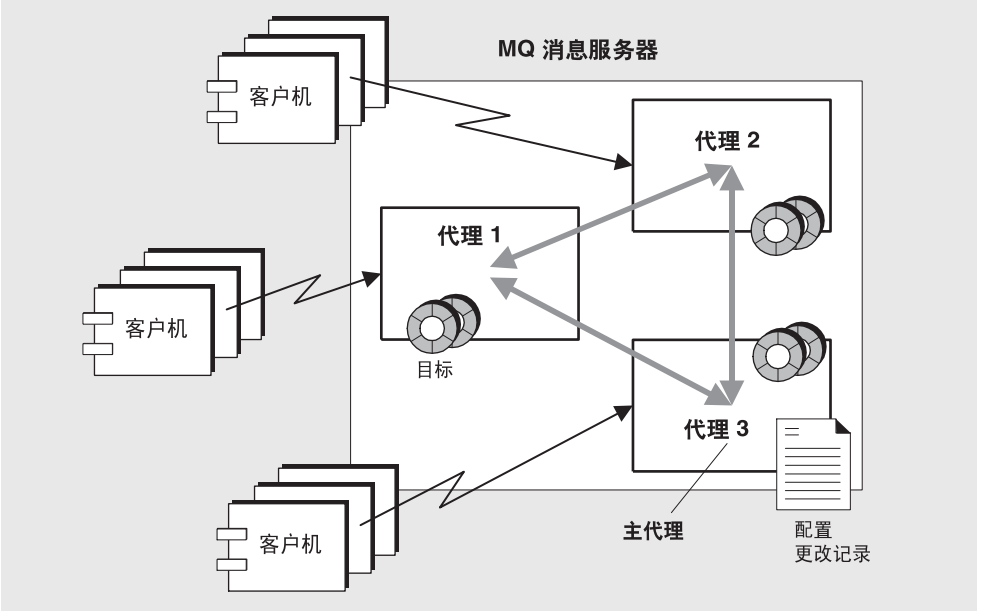
以下几个小节介绍了 MQ 代理群集的体系结构和内部功能。

多代理体系结构

多代理消息服务器允许将客户机连接分布在多个代理实例上，如图 2-7 中所示。从客户机的角度来看，每个客户机都连接到各自的代理 （其主代理），发送和接收消息时就好像主代理是群集中唯一的代理。但从消息服务器的角度来看，主代理与群集中的其它代理串联起来进行工作，为主代理直接连接的消息生成方和使用方提供传送服务。

通常，群集中的代理可以以任意拓扑结构连接。但 MQ 版本 3.0.1 仅支持完全连接的群集，即在该拓扑结构中，每个代理都与群集中的其它所有代理直接连接，如第 65 页上的图 2-7 所示。

图 2-7 多代理 （群集）体系结构



在多代理配置中，每个目标的实例位于群集中的所有代理上。另外，每个代理都了解向所有其它代理进行注册的消息使用方。因此，每个代理既可以将消息从其直接连接的消息生成方路由到远程消息使用方，也可以从远程生成方将消息发送到其直接连接的使用方。

在群集配置中，与每个消息生成方直接连接的代理将对该直接相连的生成方发送给它的消息执行路由。因此，持久性消息是通过消息的**主**代理进行存储和路由的。

当管理员创建或销毁代理上的目标时，此信息会自动传播到群集中其它所有代理。同样，当消息使用方向其主代理进行注册，或当使用方与其主代理断开连接 （无论是明确断开，还是因为客户机、网络故障或其主代理故障而断开），该使用方的相关信息都会被传播到整个群集。与此类似，**长期**订阅的相关信息也会被传播到群集中的所有代理。

注 网络繁忙和/或消息较大可能会堵塞内部群集连接。该不断增加的延迟有时还会引起锁定协议超时错误。这样，客户机在尝试创建长期订户或将消息使用方排队时可能会引发异常。通常，使用高速连接可以避免这些问题。

将目标和消息使用方的相关信息传播至特定代理时，通常要求在共享资源中进行更改时该代理处于联机状态。如果进行这样的更改时代理处于脱机状态，那么将会发生怎样的情况呢 （例如，代理崩溃、随后又重新启动，或者新代理被动态添加到群集）？

为了解决脱机的代理 （或添加的新代理）带来的问题， MQ 会提供一份记录，其中包括对群集中所有持久性实体的更改，即所有已创建或已销毁的目标和长期订阅的记录。代理被动态添加到群集后，首先从此**配置更改记录**中读取目标和长期订户信息，当它进入联机状态后，将与其它代理交换当前活动使用方的相关信息。通过这些信息，新的代理将完全集成到群集中。

配置更改记录由群集中的一个代理 （即指定为**主管代理**的代理）进行管理。因为主管代理是向群集动态添加代理的关键，所以应始终先启动此代理。如果主管代理未联机，则群集中的其它代理将无法完成初始化。

如果主管代理脱机，而其它代理又无法访问配置更改记录，则 MQ 将不允许在群集中传播目标和长期订阅。此时，如果尝试创建或销毁目标和长期订阅 （或尝试大量相关操作，例如重新激活长期订阅），将引发异常。

在关键任务的应用程序环境中，最好定期备份配置更改记录，以避免因记录意外损坏或主管代理故障带来的损失。可以使用 `imqbrokerd` 命令的 `-backup` 选项 （此选项用于创建包含配置更改记录的备份文件，请参见第 115 页上的表 5-2）来完成上述备份。然后可以使用 `-restore` 选项来恢复配置更改记录。

如果需要，可以更改用作主管代理的代理，更改方法为：备份配置更改记录，修改相应的群集配置属性 （请参见第 68 页上的表 2-10）以指定新的主管代理，然后使用 `-restore` 选项重新启动新的主管代理。

在开发环境中使用群集

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，因此基本不需要主管代理。在配置为**没有**主代理的环境中， MQ 不再要求必须运行主管代理才能启动其它代理，并允许对目标和长期订阅进行更改，并且可以在群集的所有运行代理中传播更改。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。

在测试环境下，通常自动创建目标 （请参见第 63 页上的 “自动创建的 （与管理员创建的相对）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。目标和长期订阅中的这些更改将在整个群集中传播。但是，如果将环境重新配置为使用主管代理，则 MQ 将重新强制要求必须运行主管代理才能更改目标和长期订阅并在整个群集中传播这些更改。

群集配置属性

启动时，群集中的每个代理都必须接收到有关群集中其它代理的信息 （主机名和端口号）。此信息用于在群集中的代理之间建立连接。另外，每个代理也必须知道主管代理 （如果使用）的主机名和端口号。

群集中的所有代理应使用相同的群集配置属性。为此，可以将它们放入一个中心**群集配置文件**，启动时每个代理均会引用该文件。

（也可以复制这些配置属性，然后分别为每个代理提供这些属性。但最好不要采用这种做法，因为这会导致群集配置中出现不一致。只保留一份群集配置属性可以确保所有代理获得相同的信息。）

表 2-10 所示为 MQ 群集配置属性。（有关设置这些属性的说明，请参见第 118 页上的 “使用群集 （企业版）”。）

表 2-10 群集配置属性

属性名称	说明
`imq.cluster.brokerlist`	指定群集中的所有代理。由 *host:port* 项的列表 （用逗号分隔各项）组成，其中 *host* 是每个代理的主机名，*port* 是其端口映射器的端口号。
`imq.cluster.masterbroker`	指定群集中哪个代理为记录状态更改的主管代理 （如果存在）。该属性由 *host:port* 组成，其中 *host* 是主管代理的主机名， *port* 是其端口映射器的端口号。
`imq.cluster.url`	指定代理配置文件的位置。此属性的适用情况：多个代理引用一个中心配置文件，而不是分别为每个代理提供群集属性值。由 URL 字符串组成：如果是在 Web 服务器上，可以使用标准的 `http:`*URL* 来访问。如果是在共享的驱动器上，可以使用 `file:`*URL* 来访问。
`imq.cluster.port`	对于群集中的**每个**代理，该属性可以用来指定群集连接服务的端口号。群集连接服务用于群集中代理之间的内部通信。 默认值：`0` （动态分配端口）
`imq.cluster.hostname`	对于群集中的**每个**代理，如果有多个主机可用 （例如，一台计算机中安装了多个网络接口卡），群集连接服务将绑定到该属性指定的主机 （主机名或 IP 地址）。群集连接服务用于群集中代理之间的内部通信。 默认值：`null` （所有可用主机）

群集配置文件可以用于存储一组代理通用的所有代理配置属性。虽然，它原本是用来配置群集的，但也可以用来存储群集中所有代理通用的其它代理属性。

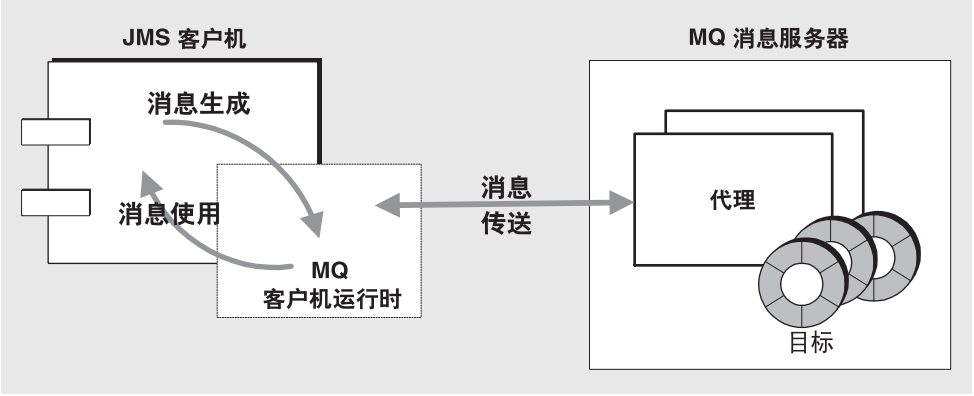
MQ 客户机运行时

MQ 客户机运行时向客户机应用程序提供至 MQ 消息服务器的接口，即向客户机应用程序提供第 32 页上的 “JMS 编程模型”中介绍的所有 JMS 编程对象。它支持客户机向目标发送消息以及接收来自这些目标的消息所需的所有操作。

本节提供 MQ 客户机运行时的工作方式的高级说明。《MQ *Developer's Guide*》中介绍了影响其性能的因素，因为它们也是影响客户机应用程序的设计和性能的因素。

第 69 页上的图 2-8 说明了在消息的生成和使用中客户机应用程序如何与 MQ 客户机运行时之间进行交互以及在消息传送中 MQ 客户机运行时如何与 MQ 消息服务器之间进行交互。

图 2-8 消息传送操作



消息生成

在消息生成中，由客户机创建消息，然后通过连接将消息发送至代理中的目标。如果将 MessageProducer 对象的消息传送模式设置为持久性 （有保证的传送，传送一次且仅传送一次），客户机线程将阻塞，直到代理确认消息已传送到其目标并存储在代理的持久性数据存储中为止。如果消息不是持久性的，则代理不会返回代理确认消息 （在属性名称中称为 “Ack”），并且客户机线程不会阻塞。

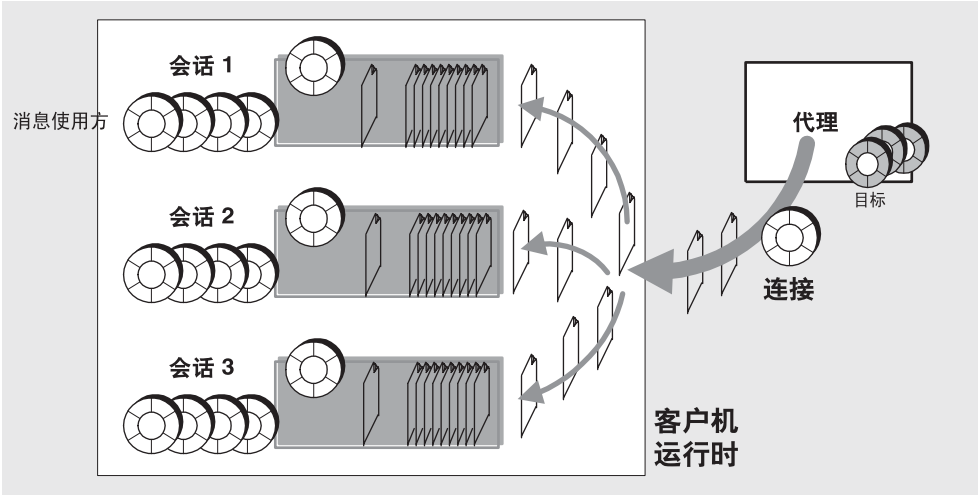
消息使用

消息的使用要比生成复杂得多。在以下条件下，将把到达代理中的目标的消息通过连接传送到 MQ 客户机运行时：

- 客户机已经设置了给定目标的使用方
- 使用方的选择条件 （如果存在）与到达给定目标的消息的选择条件匹配
- 已告知连接开始传送消息。

通过连接传送的消息将分发到相应的 MQ 会话，在这些会话中，它们排队等待供相应的 MessageConsumer 对象使用，如图 2-9 所示。每次从每个会话队列中取出一条消息 （会话是单线程的），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 MessageListener 对象的 `onMessage` 方法的客户机线程）消息。

图 2-9 将消息传送到 MQ 客户机运行时



代理在向客户机运行时传送消息时，会对消息进行相应地标记，但无法实际得知这些消息是否已被接收或使用。因此，只有客户机确认接收到消息后，代理才会将消息从其目标中删除。

MQ 被管理对象

被管理对象可以使客户机应用程序代码与提供者无关。为此，它们需要将提供者特有的实现方案和配置信息封装在对象中，而客户机应用程序可以通过与提供者无关的方式使用这些对象。被管理对象由管理员创建和配置，并存储在命名服务中，由客户机应用程序通过标准 JNDI 查找代码来访问。

MQ 提供两种类型的被管理对象：`ConnectionFactory`（连接工厂）和 `Destination`（目标）。虽然两者都用于封存提供者特有的信息，但在客户机应用程序中，它们的用途却有很大的差异。`ConnectionFactory` 对象用于创建至消息服务器的连接，而 `Destination` 对象用于标识物理目标。

通过被管理对象，可以非常容易地控制和管理 MQ 消息服务器：

- 您可以通过要求客户机应用程序访问预配置的 `ConnectionFactory` 对象来控制连接的行为（请参见第 156 页上的 “被管理对象的属性”）。
- 您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 `Destination` 对象来控制物理目标的创建。（您还需要禁用代理的自动创建功能，请参见第 63 页上的 “自动创建的（与管理员创建的相对）目标”。）
- 您可以通过覆盖由客户机应用程序设置的消息标题值来控制 MQ 消息服务器资源（请参见第 156 页上的 “被管理对象的属性”）。

因此，被管理对象使您（MQ 管理员）可以控制消息服务器配置的详细信息，同时又可以使客户机应用程序与提供者无关：它们不需要了解提供者特有的语法和对象命名惯例（请参见第 36 页上的 “JMS 提供者无关”）或提供者特有的配置属性。

请使用 MQ 管理工具来创建被管理对象，如第 7 章 “管理被管理对象”中所述。创建被管理对象时，可以将对象指定为只读，也就是说，不允许客户机应用程序更改 MQ 特有的配置值（即在创建对象时设置的值）。换句话说，客户机代码无法设置只读被管理对象的属性值，而且也无法使用客户机应用程序的启动选项覆盖这些值，如第 73 页上的 “在客户机启动时覆盖属性值”中所述。

虽然客户机应用程序可以独立地实例化 `ConnectionFactory` 和 `Destination` 这两种被管理对象，但这样会削弱被管理对象的基本用途（即使 MQ 管理员 可以控制应用程序所需的代理资源以及调节其性能的用途）。另外，直接实例化被管理对象将使客户机应用程序成为提供者特有的应用程序，而无法实现与提供者无关。

“连接工厂”被管理对象

ConnectionFactory 对象用于建立客户机应用程序与 MQ 消息服务器之间的物理连接，还可用于指定连接的行为以及使用此连接访问代理的客户机运行时的行为。

如果要支持分布式事务 （请参见第 39 页上的 “本地事务”），需要使用支持分布式事务的特殊的 XAConnectionFactory 对象。

要创建 ConnectionFactory 被管理对象，请参见第 163 页上的 “添加连接工厂”。

通过配置 ConnectionFactory 被管理对象，可以指定该对象生成的所有连接通用的属性值 （属性）。 ConnectionFactory 和 XAConnectionFactory 对象共享同一属性集。按照其影响的行为，这些属性可分为多个种类：

- 连接规范
- 自动重新连接行为
- 客户机标识
- 消息标题覆盖
- 可靠性和流控制
- 队列浏览器行为
- 应用程序服务器支持
- JMS 定义的属性支持

每个种类及其相应属性将在《MQ *Developer's Guide*》中进一步讨论。虽然作为 MQ 管理员，可能会要求您调整这些属性的值，但通常是由应用程序开发者来决定需要调整哪些属性以调节客户机应用程序的性能。第 157 页上的表 7-3 按字母顺序概述了这些属性。

Destination 被管理对象

Destination 被管理对象表示代理中与公开命名的 Destination 对象对应的物理目标 （队列或主题）。表 2-11 中列出了它的两个属性。通过创建 Destination 对象，可以使客户机应用程序的 MessageConsumer 和 / 或 MessageProducer 对象能够访问相应的物理目标。

要创建 Destination 被管理对象，请参见第 164 页上的 “添加主题或队列”。

表 2-11 目标属性

属性/属性名称	说明
imqDestinationName	指定物理目标的供应商特有的名称。在创建物理目标时指定此名称。目标名称必须只包含字母数字字符（不包括空格），可以以字母字符或 “_” 和 “$” 字符开头。 默认值：Untitled_Destination_Object
imqDestinationDescription	指定用于管理对象的信息。默认值：目标对象的说明

在客户机启动时覆盖属性值

像使用任何 Java 应用程序一样，您可以使用命令行来启动消息传送应用程序以指定系统属性。此机制也可以用来覆盖客户机应用程序代码中使用的被管理对象的属性值。例如，可以覆盖在客户机应用程序代码中通过 JNDI 查找来访问的被管理对象的配置。

要在客户机应用程序启动时覆盖被管理对象的设置，请使用以下命令行语法：

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

其中，*attribute* 与第 156 页上的 “连接工厂被管理对象”中记录的任意 ConnectionFactory 被管理对象的属性相对应。

例如，如果希望客户机应用程序连接到其它代理，而不是连接到在客户机代码中访问的 ConnectionFactory 被管理对象中指定的代理，可以使用命令行覆盖来启动客户机应用程序以设置另一个代理的 imqBrokerHostName 和 imqBrokerHostPort。

如果已将被管理对象设置为只读，则无法使用命令行覆盖来更改其属性的值。所有此类覆盖将被忽略。

第 3 章

MQ 管理

Sun™ ONE Message Queue (MQ) 管理由若干任务以及用于执行这些任务的若干工具组成。

本章首先概述了管理任务，然后介绍了管理工具，其中着重介绍命令行管理实用程序的通用特性。

MQ 管理任务

您需要执行哪些特定任务取决于您处于开发环境还是处于生产环境。

开发环境

在开发环境中，工作的重点是 MQ 客户机应用程序编程。 MQ 消息服务器主要用于测试。在开发环境中，强调的是灵活性，管理任务很少，大都是启动开发者要用于测试的代理。对于开发测试来说，数据存储、用户系统信息库、访问控制属性文件和对象存储的默认实现方案通常就已经足够了。如果是执行多代理测试，很可能不会使用主管代理。另外，被测试的应用程序一般可以使用自动创建的目标，并且您可能不希望使用集中管理的被管理对象。

生产环境

在生产环境中，由于必须在其中可靠地部署和运行应用程序，所以管理就重要得多了。需要执行的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。不过，通常这些任务可分为设置操作和维护操作。

设置操作

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：

- 安全性 （请参见第 8 章 “安全性管理”）：
  - 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。）
  - 修改访问控制属性文件中的访问设置
  - 设置基于 SSL 的连接服务
- 被管理对象 （请参见第 7 章 “管理被管理对象”）：
  - 配置或设置 LDAP 对象存储
  - 创建连接工厂和目标被管理对象
- 代理群集 （请参见第 118 页上的 “使用群集 （企业版）”）：
  - 创建中心配置文件
  - 使用主管代理
- 持久性：将代理配置为使用插入的持久性，而不是内置的持久性 （请参见附录 A “设置插入的持久性”）

维护操作

另外，在生产环境中，需要对 MQ 消息服务器资源进行严格的监视和控制。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 MQ 管理工具执行如下所述的若干任务：

- 应用程序管理：
  - 禁用代理的自动创建功能 （请参见第 64 页上的表 2-9）
  - 代替应用程序创建物理目标 （请参见第 144 页上的 “创建目标”）
  - 设置对目标的用户访问权限 （请参见第 178 页上的 “授权用户：访问控制属性文件”）
  - 监视和管理目标 （请参见第 143 页上的 “管理目标”）
  - 监视和管理长期订阅 （请参见第 146 页上的 “管理长期订阅”）
  - 监视和管理事务 （请参见第 147 页上的 “管理事务”）

- 代理管理和调节：
  - 使用代理测量信息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
- 管理被管理对象
  - 按照需要创建其它连接工厂和目标被管理对象
  - 调节连接工厂的属性值以提高性能和吞吐量 （请参见第 72 页上的 “ “连接工厂” 被管理对象”）

MQ 管理工具

MQ 管理工具分为两类：命令行实用程序和图形用户界面 (GUI) 管理控制台 (`imqadmin`)。控制台结合了两个命令行实用程序的功能：命令实用程序 (`imqcmd`) 和对象管理器实用程序 (`imqobjmgr`)。可以使用控制台 （和这两个命令行实用程序）来远程管理代理以及管理 MQ 被管理对象。其它命令行实用程序 （`imqbrokerd`、`imqusermgr`、`imqdbmgr` 和 `imqkeytool`）必须在其关联代理的同一主机上运行，如图 3-1 所示。

可以从联机帮助中获得有关管理控制台的信息。“命令行实用程序概述”介绍了通常用于执行特殊任务的命令行实用程序。

管理控制台

可以使用管理控制台执行以下任务：

- 连接到代理并对其进行管理。
- 在代理上创建物理目标
- 连接到对象存储
- 向对象存储添加被管理对象。

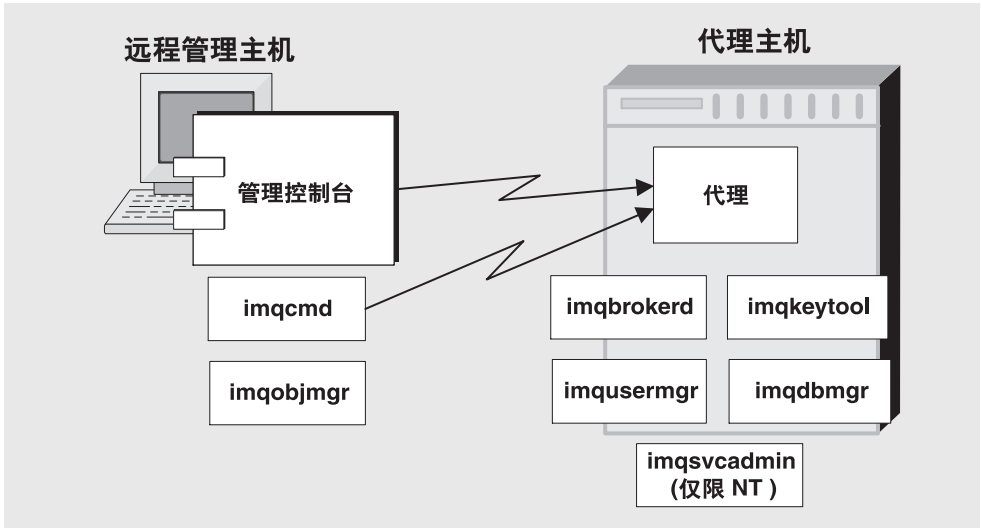
某些任务不能使用管理控制台来执行，主要有：启动代理、创建代理群集、配置代理更加具体的属性和管理用户数据库。

第 4 章 “管理控制台教程”提供了一个简明扼要的实用教程，帮助您熟悉控制台，并说明了如何使用它完成基本任务。

命令行实用程序概述

本节介绍用于执行 MQ 管理任务的命令行实用程序。使用 MQ 实用程序启动和管理代理并执行其它更加具体的管理任务。

图 3-1 本地和远程管理实用程序



所有 MQ 实用程序均可通过命令行界面 (CLI) 来访问。实用程序命令共用通用格式、语法惯例和选项，本章下一小节将进行介绍。可以在后面的章节中找到有关使用命令行实用程序更为详细的信息。

代理 (`imqbrokerd`) 使用代理实用程序启动代理。使用 `imqbrokerd` 命令的选项来指定是否将代理连接到代理群集中，并指定其它配置信息。在第 5 章 “启动与配置代理”中介绍了此实用程序。

命令 (imqcmd) 启动代理后，使用命令实用程序来创建、更新和删除物理目标、控制代理及其连接服务，并管理代理的资源。可以使用 `imqcmd` 命令来运行此实用程序。在第 6 章 “代理和应用程序管理”中介绍了此实用程序。

对象管理器 (imqobjmgr) 使用对象管理器实用程序在可通过 JNDI 访问的对象存储中添加、列举、更新和删除被管理对象。通过将 JMS 客户机与 JMS 提供者特有的命名和配置格式分离，被管理对象使 JMS 客户机可以与提供者无关。可以使用 `imqobjmgr` 命令来运行此实用程序。在第 7 章 “管理被管理对象”中介绍了此实用程序。

用户管理器 (imqusermgr) 使用用户管理器实用程序来填充用于验证和授权用户的基于文件的用户系统信息库。可以使用 `imqusermgr` 命令来运行此实用程序。在第 8 章 “安全性管理”中介绍了此实用程序。

密钥工具 (imqkeytool) 使用密钥工具实用程序来生成用于 SSL 验证的自签名证书。可以使用 `imqkeytool` 命令来运行此实用程序，在第 8 章 “安全性管理”和附录 B “HTTP/HTTPS 支持 （企业版）”中介绍了此实用程序。

数据库管理器 (imqdbmgr) 使用数据库管理器实用程序来创建和管理用于持久性存储器的 JDBC 兼容数据库。可以使用 `imqdbmgr` 命令来运行此实用程序。有关详细信息，请参见附录 A “设置插入的持久性”。

服务管理器 (imqsvcadmin) 使用服务管理器实用程序可以将代理作为 Windows 服务来进行安装、查询和删除。有关详细信息，请参见附录 C “使用代理作为 Windows 服务”。

命令行语法

MQ 命令行界面实用程序是简单的 shell 命令。也就是说，从 Windows、 Linux 或 Solaris 命令 shell 的角度来看，命令行界面实用程序的名称本身是一个命令，而它的子命令或选项只是要传送到该命令的变量而已。因此，没有、也不需要用于启动或退出实用程序的命令。

所有的命令行实用程序均共享以下命令语法：

Utility_Name [*subcommand*] [*argument*] [[*-option_name* [*-option_argument*]]...]

Utility_Name 指定 MQ 实用程序的名称，例如：`imqcmd`、`imqobjmgr`、`imqusermgr` 等等。

以下四点需要注意：

- 在子命令**之后**指定选项 （如果此实用程序还接受变量类型的操作数，则还包括变量）。
- 如果变量包含空格，请将整个变量置于引号中。将属性 - 值对置于引号中通常是最安全的做法。
- 如果在命令行中指定了 -v （版本）或 -h/-H （帮助）选项，所有此命令行中的其它选项均不执行。关于通用选项的说明，请参见第 80 页上的表 3-1。
- 用空格分隔子命令、变量、选项和选项变量。

以下是不包含子命令子句的一个命令行示例。此命令启动默认代理。

```
imqbrokerd
```

以下命令要稍微复杂一些：它将为用户名为 `admin` 且相应密码为 `admin` 的管理员（用户）销毁一个名为 `myQueue` 的 `queue` 类型的目标，此销毁操作不需要确认，并且控制台上不显示输出。

```
imqcmd destroy dst -t q -n myQueue -u admin -p admin -f -s
```

通用命令行选项

表 3-1 介绍了通用于所有 MQ 管理实用程序的选项。除了要在命令行中指定的子命令**之后**指定下述选项的要求以外，对这些选项 （或任何传送至实用程序的其它选项）的输入顺序无任何特殊要求。

表 3-1 通用 MQ 命令行选项

选项	说明
`-h`	显示指定实用程序的使用帮助
`-H`	显示详细的用法帮助，包括属性列表和示例 （只支持 `imqcmd` 和 `imqobjmgr`）。
`-s`	打开静默模式：不显示输出。对于 `imqbrokerd`，指定为 `-silent`。
`-v`	显示版本信息。
`-f`	执行给定操作，而不提示用户进行确认。
`-pre`	（仅用于 `imqobjmgr`）打开预览模式，允许用户查看命令行中其它部分的执行效果而无需实际执行此命令。此项功能有助于检查默认属性值。
`-javahome` *path*	指定要使用的替代 Java 2 运行时的位置。

第 4 章

管理控制台教程

本教程重点介绍如何使用管理控制台 — 管理 Sun™ ONE Message Queue (MQ) 消息服务器的图形界面。跟随本教程，您将学习如何进行以下操作：

- 启动代理，使用控制台连接到代理并对其进行管理。
- 在代理上创建物理目标。
- 创建对象存储，并使用控制台连接到对象存储
- 将被管理对象添加到对象存储

在本教程中，将以设置运行简单 JMS 兼容应用程序 `SimpleJNDIClient` 所需的目标和被管理对象为范例。在本教程的最后一部分，您将运行此应用程序。

本教程主要用来指导您通过使用管理控制台来执行基本管理任务。即使完成了本教程，您还是需要仔细阅读 《MQ *Developer's Guide*》或本**管理员指南**的其它章节。

某些 MQ 管理任务无法使用图形工具来完成，您将需要使用命令行实用程序执行如下所示的这些任务：

- 配置某些代理属性

  某些代理属性无法使用管理控制台来配置。可以按照第 107 页上的第 5 章 “启动与配置代理”或第 138 页上的 “更新代理”中的说明配置这些属性。

- 创建代理群集

  有关详细信息，请参见第 118 页上的 “使用群集 （企业版）”。

- 管理用户数据库

  有关详细信息，请参见第 170 页上的 “验证用户”。

准备工作

必须安装 MQ 产品才能启动本教程。有关详细信息，请参见 《MQ **安装指南**》。请注意，本教程以 Windows 为中心，并为 Unix 用户添加了说明。

在本教程中，选择 “Item1” > “Item2” > “Item3” 表示应下拉名为 “Item1” 的菜单，从该菜单选择 “Item2”，然后从 “Item2” 提供的选择中选择 “Item3”。

启动管理控制台

管理控制台 (Administration Console) 是您用于完成以下操作的图形工具：

- 创建对代理的引用并连接到代理
- 管理代理
- 在代理上创建物理目标，代理将使用它来进行消息传送
- 连接到存储 MQ 被管理对象的对象存储

  被管理对象使您可以管理 JMS 兼容应用程序的消息传送需要。有关详细信息，请参见第 71 页上的 “MQ 被管理对象”。

➤ **启动管理控制台的步骤**

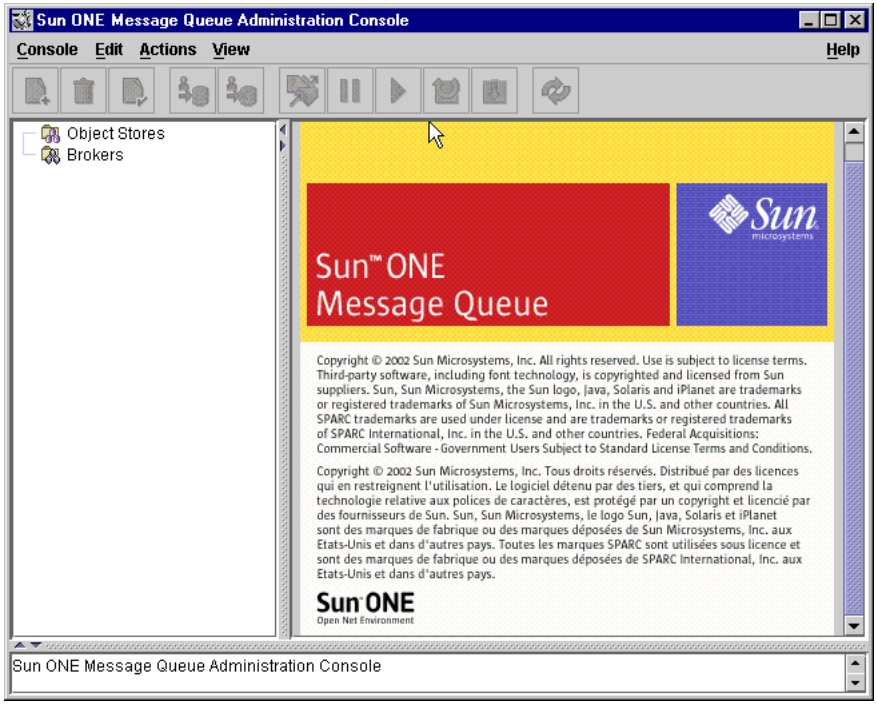
1. 选择 “开始” > “程序” > “Sun ONE Message Queue 3.0 > Administration”。

   显示控制台窗口之前，您可能需要等待几秒钟。

   非 Windows 用户：在命令提示符下输入以下命令：

   `$IMQ_HOME/bin/imqadmin`（在 `Solaris` 操作系统上为 `/usr/bin/imqadmin`）

2. 请花几秒钟仔细观察一下控制台窗口。



控制台包括位于顶部的菜单、菜单下面的工具栏、左侧的浏览窗格、右侧的较大窗格（现在显示标识 Sun ONE Message Queue 产品的图形）和底部的状态窗格。

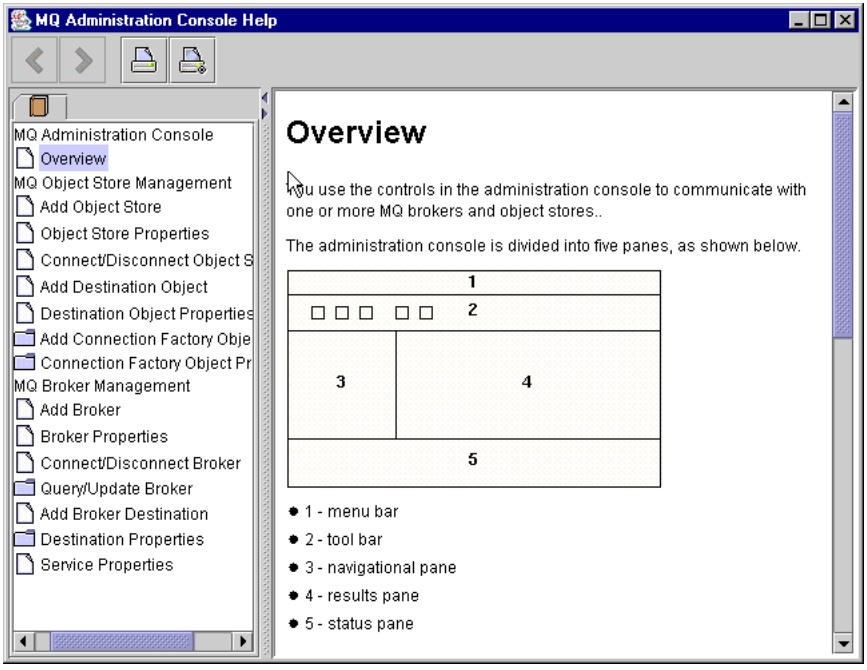
任何教程都不可能提供全部的信息，因此让我们首先找出如何获得管理控制台的帮助信息。

获得帮助

在菜单条的最右侧找到 “Help”菜单。

➤ 显示管理控制台帮助信息的步骤

1. 下拉 “Help”菜单并选择 “Overview”。将显示一个帮助窗口。



请注意帮助信息是如何组织的。左侧窗格显示内容表，右侧窗格显示在左侧选择的任意项的内容。

观察 “Help”窗口的右侧窗格。它显示管理控制台的概貌，并标识了每个控制台窗格的用途。

2. 观察 “Help”窗口的内容窗格。它使用以下三个区域来组织主题：概述、对象存储管理和代理管理。其中每个区域都包含文件和文件夹。每个文件夹提供的是包含多个选项卡的对话框的帮助，每个文件则提供简单对话框或选项卡的帮助。

   您的第一个控制台管理任务 （请参见第 87 页上的 “添加代理”）将是通过控制台创建对管理的代理的引用。但是，启动之前请查阅联机帮助以获得有关信息。

3. 单击 “Help”窗口的内容窗格中的 “Add Broker”项。

   您会发现内容窗格发生了变化。它现在包含一些文本，说明添加代理的含义并介绍 “Add Broker”对话框中每个字段的使用。字段名称以粗体文本显示。

4. 仔细阅读帮助文本。

5. 关闭 “Help”窗口。

使用代理

代理为 MQ 消息传送系统提供传送服务。消息传送的过程可分为两个阶段：首先将消息传送到代理上的物理目标，然后将其传送给一个或多个使用方客户机。

使用代理涉及以下任务：

- 启动和配置代理

  可以从 Windows 上的 “开始” > “程序”菜单或使用 `imqbrokerd` 命令启动代理。如果使用 `imqbrokerd` 命令，可以使用命令行选项指定代理配置信息。如果使用 “程序”菜单，可以使用控制台和第 5 章 “启动与配置代理”中说明的其它方法来指定配置信息。

  注 使用管理控制台无法启动代理实例。

- 使用管理控制台或使用 Command 命令行实用程序 (`imqcmd`) 管理代理及其服务。
- 创建客户机应用程序所需的物理目标
- 监视资源使用以提高吞吐量和可靠性

代理支持与应用程序客户机和管理客户机之间的通信。它利用不同的连接服务实现此功能，并且您可以将代理配置为运行这些服务之一或全部运行。有关连接服务的详细信息，请参见第 46 页上的 “连接服务”。

启动代理

使用管理控制台无法启动代理。按照以下 （请参见第 5 章 “启动与配置代理”）说明启动代理。

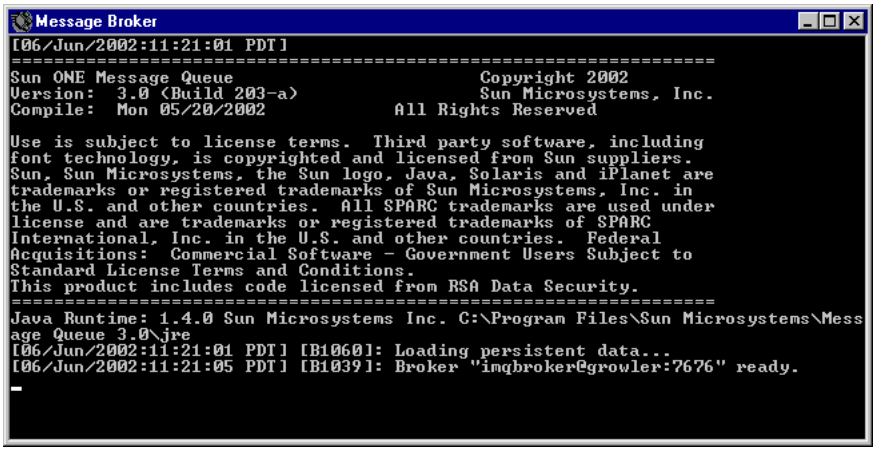
➤ **启动代理的步骤**

1. 选择 “开始” > “程序” > “Sun ONE Message Queue 3.0 > Broker”。

   非 Windows 用户：输入以下命令启动代理。

   `%$IMQ_HOME/bin/imqbrokerd` （在 Solaris 操作系统上为 `/usr/bin/imqbrokerd`）

   将显示一个代理进程窗口。在代理准备就绪时指定代理的名称。



2. 现在让我们把注意力转回到 “Administration Console”窗口。您现在可以将代理添加到控制台并连接到代理。

在管理控制台中添加对代理的引用之前不必启动代理，但是必须启动代理才能连接到它。

添加代理

添加代理在管理控制台中创建了一个对该代理的引用。添加代理之后，您就可以连接到它。

➤ **将代理添加到管理控制台的步骤**

1. 在浏览窗格的 “Brokers”上单击右键并选择 “Add Broker”。
2. 在 “Broker Label”字段中输入 `MyBroker`。

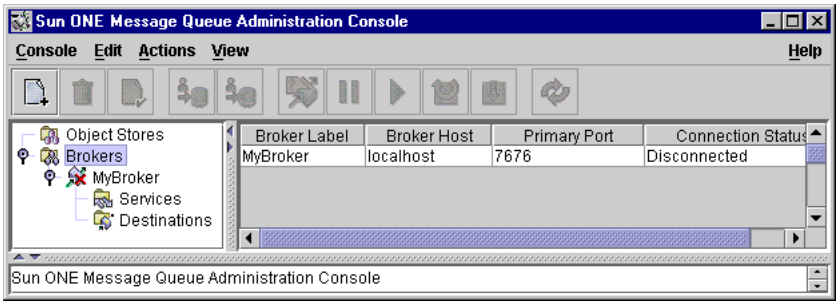
   这样便提供了一个在管理控制台中标识代理的标签。

   注意：在对话框中已指定默认主机名 (`localhost`) 和默认主端口 (`7676`)。这些值是以后需要指定的值，即在您配置客户机用来建立与此代理的连接的连接工厂时需要指定这些值。

   将 “Password”字段保留为空。如果在连接时指定密码，则密码将会更安全。

3. 单击 “OK”以添加代理。

观察浏览窗格。“Brokers”下方应已列出刚才添加的代理。代理图标上的红色 X 表示该代理当前未连接到控制台。



4. 在 “MyBroker”上单击右键，并从弹出式菜单中选择 “Properties”。

   将显示代理属性对话框。可以使用此对话框更新添加代理时指定的任意属性。

更改管理员密码

当连接到代理时，如果添加代理时未指定密码，系统将提示您指定密码。为了提高安全性，建议您在连接之前更改默认管理员密码 (admin)。

➤ 更改管理员密码的步骤

1. 打开命令提示符窗口。如果窗口已打开，则请继续下一步操作。
2. 输入如下命令，并用自己的密码代替 abracadabra。然后，您指定的密码将代替 admin 的默认密码。

   ```
   imqusermgr update -u admin -p abracadabra
   ```

   （在 Solaris 和 Linux 操作系统上，必须是根用户才能执行此操作。）

更改将立即生效。每当使用某个 MQ 命令行实用程序或管理控制台时，必须指定新密码。

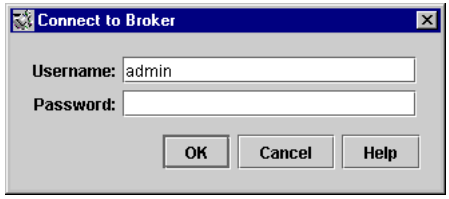
尽管客户机使用不同于管理员的连接服务，但也为其分配了默认用户名和密码，这样您不必进行大量的管理设置便可以测试 MQ。默认情况下，客户机可以作为用户 guest 并使用密码 guest 连接到代理。但是，应尽快建立客户机的安全用户名和密码。有关详细信息，请参见第 170 页上的 “验证用户”。

连接到代理

➤ **连接到代理的步骤**

1. 在 “MyBroker” 上单击右键并选择 “Connect to Broker”。

   将显示一个对话框，您可以在其中指定用户名和密码。



2. 在 “Password” 字段中输入 admin 或在第 88 页上的 “更改管理员密码”中指定的任何密码值。

   指定用户名 admin 并提供正确密码将使您以管理员权限连接到代理。

3. 单击 “OK” 连接到代理。

连接到代理后，可以从 “Actions” 菜单中选择各个菜单项，以获得有关代理的信息、暂停和恢复代理、关闭和重新启动代理以及从代理断开连接。

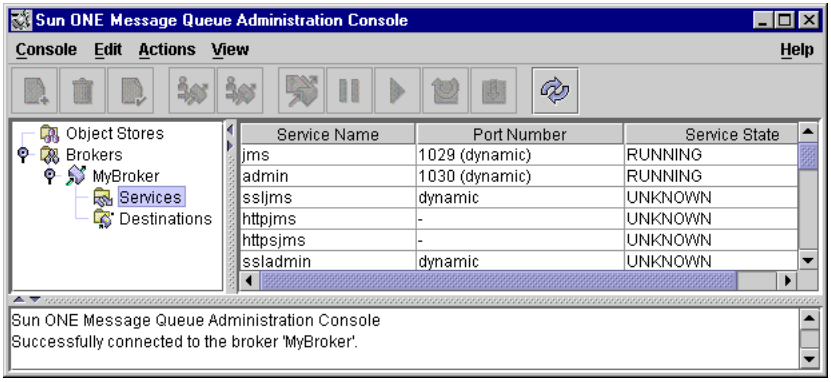
查看连接服务

从代理提供的连接服务及其支持的物理目标可以区分代理。

➤ 查看可用连接服务的步骤

1. 在浏览窗格中选择 “Services”。

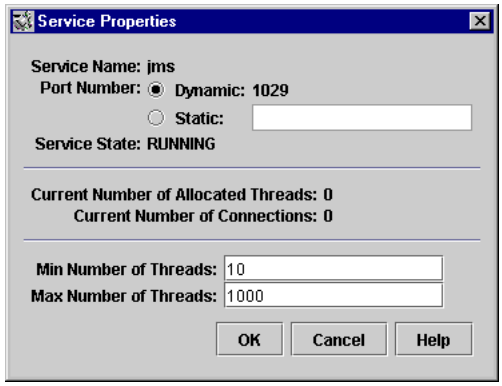
   在结果窗格中列出了可用服务。对于每项服务，均提供了其名称、端口号和状态。



2. 单击结果窗格中的 “jms”服务，以选中它。
3. 下拉 “Actions”菜单并注意高亮显示的项。

   您可以选择暂停 jms 服务或查看和更新其属性。
4. 从 “Actions”菜单中选择 “Properties”。

请注意，在 “Service Properties”对话框中，您可以分配服务的静态端口号并可以更改分配给此服务的最小和最大线程数。



5. 单击 “OK”或单击 “Cancel”关闭属性对话框。
6. 在结果窗格中选择 “admin”服务。
7. 下拉 “Actions”菜单。

   您会注意到无法暂停此服务，暂停项已被禁用。 admin 服务是至代理的管理员链接。如果您暂停该服务，将再也无法访问代理。
8. 选择 “Actions” > “Properties”查看 admin 服务的属性。
9. 完成时请单击 “OK”或 “Cancel”。

将物理目标添加到代理

必须在代理上明确创建物理目标，这样 JMS 兼容应用程序才能正常运行。如果代理已启用自动创建目标功能 （该功能使代理能够动态创建物理目标），则不必进行此操作。

在开发环境中，自动创建目标是可以接受的。但是，在生产设置中，建议关闭这个功能，并将代理设置为使用已明确创建的物理目标。这使您 （管理员）可以完全掌握代理上使用的目标。

您可以控制代理是否可以通过设置 `imq.autocreate.topic` 或 `imq.autocreate.queue` 属性来添加自动创建的目标。有关详细信息，请参见第 63 页上的 “自动创建的 （与管理员创建的相对）目标”。

在本教程的这个小节中，您将把一个物理目标添加到代理。应注意分配给目标的名称，您以后会在创建对应此物理目标的被管理对象时需要此名称。

➤ 将队列目标添加到代理的步骤

1. 在 “MyBroker” 的 “Destinations” 节点上单击右键并选择 “Add Broker Destination”。

   将显示以下对话框：

2. 在 “Destination Name”字段中输入 `MyQueueDest`。
3. 选择 “Queue”单选按钮 （如果尚未选中此按钮）。
4. 确保将 “Queue Delivery Policy”选择为 “Single”。
5. 单击 “OK”添加物理目标。

   现在，目标显示在结果窗格中。

使用物理目标

在代理上添加一个物理目标后，您可以按照稍后的步骤说明来执行以下任意一项任务：

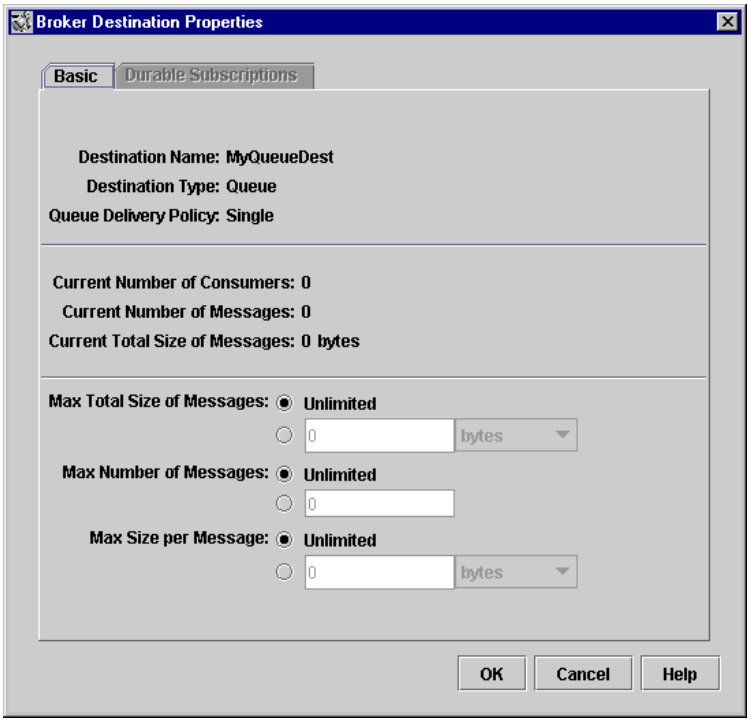
- 查看和更新物理目标的属性
- 清理目标上的消息
- 删除目标

➤ 查看物理目标属性的步骤

1. 选择 “MyBroker”的 “Destinations”节点。
2. 在结果窗格中选择 “MyQueueDest”。

3. 选择 “Actions” > “Properties”。

   将显示以下对话框：



   请注意，您仅能更改与传送到该队列的消息的大小和数目有关的队列属性。

4. 单击 “Cancel” 关闭对话框。

➤ 清理目标中的消息的步骤

1. 在结果窗格中选择物理目标。
2. 选择 “Actions” > “Purge Messages”。

   将显示一个确认对话框。

清理消息将删除消息，留下一个空目标。

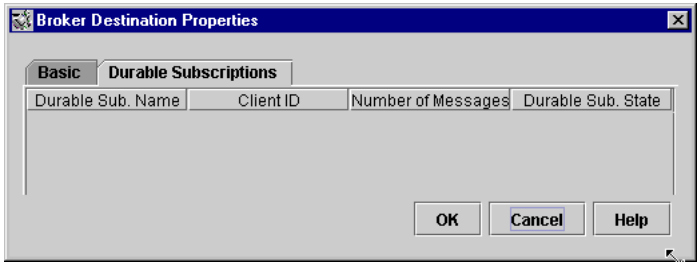
➤ 删除目标的步骤

1. 在结果窗格中选择物理目标。
2. 选择 “Edit” > “Delete”。

删除目标将清除该目标上的消息并删除目标。

获得有关主题目标的信息

有关主题目标的对话框包含一个列出长期订阅的相关信息的附加选项卡。



使用此对话框可以执行以下操作：

- 清理长期订阅，从而删除与长期订阅关联的所有消息
- 删除长期订阅，从而清理与长期订阅关联的所有消息并删除该长期订阅

使用对象存储

对象存储可以是 LDAP 目录服务器或文件系统存储 （文件系统中的目录），用于存储 MQ 被管理对象。这些被管理对象将封装有关客户机应用程序使用的对象的 MQ 特有实现方案和配置信息的。

尽管可以在客户机代码中实例化和配置被管理对象，但建议您 （管理员）最好创建和配置这些对象，并将其存储在对象存储中，客户机应用程序将通过标准 JNDI 查找代码来访问该对象存储。这将使客户机代码与提供者无关。

有关被管理对象的详细信息，请参见第 71 页上的 “MQ 被管理对象”。

您不能使用管理控制台来**创建**对象存储。必须按照以下小节中的说明提前创建。

添加对象存储

添加对象存储将在管理控制台中创建一个对现有对象存储的引用。即使退出并重新启动控制台，此引用也将保留。

➤ 添加文件系统对象存储的步骤

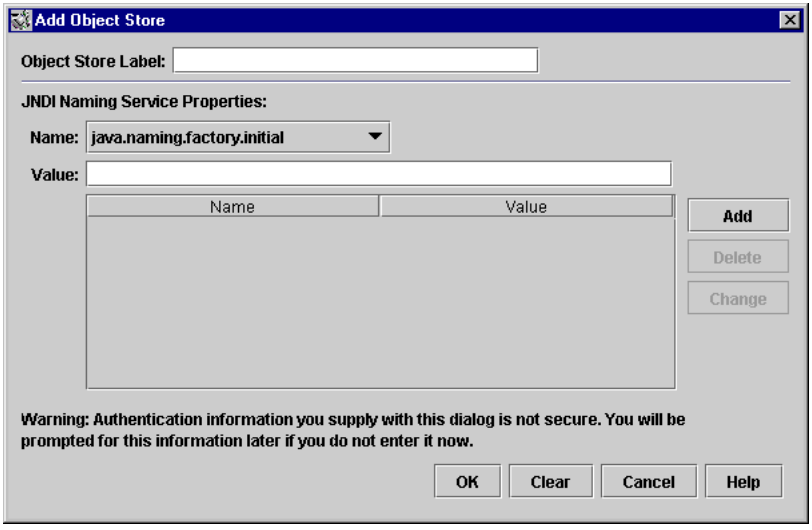
1. 如果在您的 C 驱动器上没有名为 `Temp` 的文件夹，那么现在就请创建这个文件夹。

   本教程中使用的范例应用程序假定对象存储是 C 驱动器上一个名为 `Temp` 的文件夹。文件系统对象存储通常可以是任何驱动器上的任何目录。

   非 Windows 用户：可以使用现有的 `/tmp` 目录。

2. 在 “Object Stores” 上单击右键并选择 “Add Object Store”。

   将显示以下对话框：



3. 在名为 “ObjectStoreLabel”的字段中输入 `MyObjectStore`。

   这仅是在管理控制台中提供一个对象存储显示的标签。

   在以下步骤中，您将需要输入 JNDI 名称/值对。 JMS 兼容应用程序使用这些名称/值对查找被管理对象。

4. 从 “Name”下拉菜单中选择 “`java.naming.factory.initial`”。

   此属性使您可以指定希望使用哪个 JNDI 服务提供者。例如，文件系统服务提供者或 LDAP 服务提供者。

5. 在 “Value”字段中输入以下内容：

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

   这表示您将使用文件系统存储。（对于 LDAP 存储，将指定 `com.sun.jndi.ldap.LdapCtxFactory`。）

   在生产环境中，您可能希望将 LDAP 目录服务器用作对象存储。有关设置服务器和进行 JNDI 查看的信息，请参见第 159 页上的 “对象存储属性”。

6. 单击 “Add”按钮。

   您会注意到现在属性概要窗格中列出了属性及其值。

7. 从 “Name”下拉菜单中选择 “`java.naming.provider.url`”。

   此属性使您可以指定对象存储的确切位置。对于文件系统类型的对象存储，这将是现有目录的名称。

8. 在 “Value”字段中，输入以下内容

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

   （在 Solaris 和 Linux 操作系统上为 `file:///tmp`）

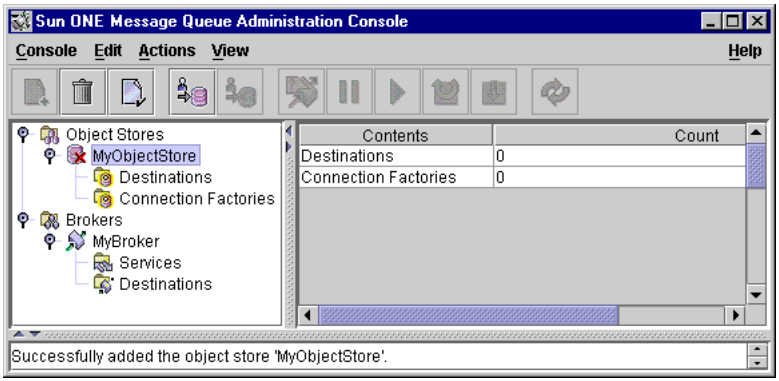
9. 单击 “Add”按钮。

   您会注意到现在属性概要窗格中列出了属性及其值。如果是在使用 LDAP 服务器，还可能必须要指定验证信息。这对文件系统存储不是必需的。

10. 单击 “OK”添加对象存储。

11. 如果未在浏览窗格中选中节点 “MyObjectStore”，那么现在请选中它。

    现在，管理控制台的显示如下所示：



    浏览窗格中列出了该对象存储，结果窗格中则列出了对象存储的内容 “Destinations”和 “Connection Factories”。我们尚未将任何被管理对象添加到对象存储，如结果窗格的 “Count”列中所示。

    浏览窗格中的对象存储图标上绘制了一个红色的 X。这表示它处于断开连接的状态。需要先连接到对象存储，然后才能使用它。

检查对象存储属性

当管理控制台从对象存储断开连接时，可以查看和更改对象存储的某些属性。

➤ **显示对象存储属性的步骤**

1. 在浏览窗格的 “MyObjectStore”上单击右键。
2. 从弹出式菜单中选择 “Properties”。

   将显示一个对话框，其中显示所有在添加对象存储时指定的属性。您可以更改这些属性中的任意属性，并单击 “OK”更新旧信息。
3. 单击 “OK”，或单击 “Cancel”关闭对话框。

连接到对象存储

必须先连接到对象存储，然后才能将对象添加其中。

➤ **连接到对象存储的步骤**

1. 在浏览窗格的 “MyObjectStore”上单击右键。
2. 从弹出式菜单中选择 “Connect to Object Store”。

   您会注意到对象存储图标上的红色叉号不见了。现在可以将对象、连接工厂和目标添加到对象存储。

添加连接工厂被管理对象

可以使用管理控制台来创建和配置连接工厂。客户机代码使用连接工厂连接到代理。通过配置连接工厂，可以控制使用连接工厂创建的连接的行为。

有关配置连接工厂的信息，请参见联机帮助和 《MQ *Developer's Guide*》。

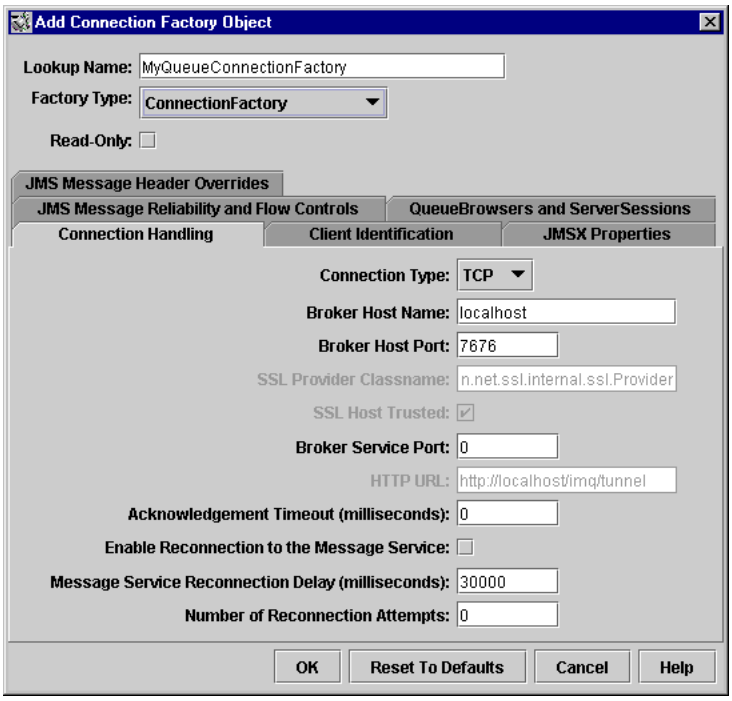
注 管理控制台仅列出和显示了 MQ 被管理对象。如果对象存储中包含与希望添加的被管理对象具有相同查找名称的非 MQ 对象，则您将在尝试添加操作时会接收到一个错误信息。

➤ **将连接工厂添加到对象存储的步骤**

1. 如果仍未连接，请连接到 MyObjectStore （请参见第 99 页上的 “连接到对象存储”）。

2. 在 “Connection Factories”节点上单击右键，并选择 “Add Connection Factory Object”。

   将显示 “Add Connection Factory Object”对话框。



3. 在 “LookupName”字段中输入名称 “`MyQueueConnectionFactory`”。

   这是客户机代码查找连接工厂时使用的名称，如 `SimpleAdmin.java` 中的以下代码行所示：

```
qcf=(javax.jms.QueueConnectionFactory)
                    ctx.lookup("MyQueueConnectionFactory")
```

4. 从下拉菜单中选择 “QueueConnectionFactory”以指定连接工厂的类型。

5. 在 “Broker Host Name”和 “Broker Host Port”字段中，输入客户机计划连接的代理主机名和端口。

   在本教程中，客户机连接到默认代理，即位于 `7676` 端口的 `localhost` 上的代理，因此不必更改这些字段。

6. 逐个单击此对话框的各个选项卡，查看您可以配置的连接工厂的信息类型。使用 “Add Connection Factory Object”对话框右下角的 “Help”按钮来获得有关各个选项卡的信息。目前请勿更改任何默认值。

7. 单击 “OK”创建队列连接工厂。

8. 观察结果窗格。列出了新创建的连接工厂的查找名称和类型。

添加目标被管理对象

目标被管理对象与代理上的物理目标相关联。或者说，它们指向这些目标，使客户机可以查找并找到物理目标，而与命名和配置这些目标的提供者特有方法无关。

当 JMS 客户机发送消息时，它将查找（或实例化）目标被管理对象，并在 JMS API 的 `send()` 方法中引用它。然后代理负责将消息传送给与该被管理对象相关联的物理目标。

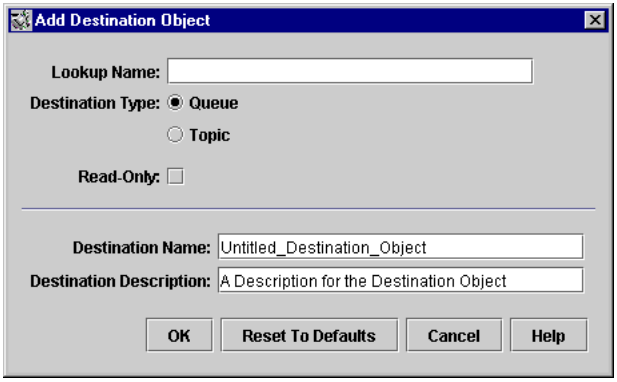
- 如果已创建与该被管理对象相关联的物理目标，则代理会将消息传送给该物理目标。
- 如果尚未创建物理目标，并且物理目标的自动创建功能已启用，代理将自动创建物理目标并将消息传送给该目标。
- 如果尚未创建物理目标，并且物理目标的自动创建功能已被**禁用**，代理将无法创建物理目标，也就无法传送消息。

在本教程的下一部分中，您将添加一个与前面添加的物理目标相对应的被管理对象。

➤ 将目标添加到对象存储的步骤

1. 在浏览窗格中的 “Destinations”节点上单击右键。
2. 选择 “Add Destination Object”。

   管理控制台将显示一个 “Add Destination Object”对话框，用于指定有关对象的信息。



3. 在 “Lookup Name”字段中输入 “`MyQueue`”。

   JNDI 查找调用使用该查找名称来查找对象。在范例应用程序中，此调用如下所示：

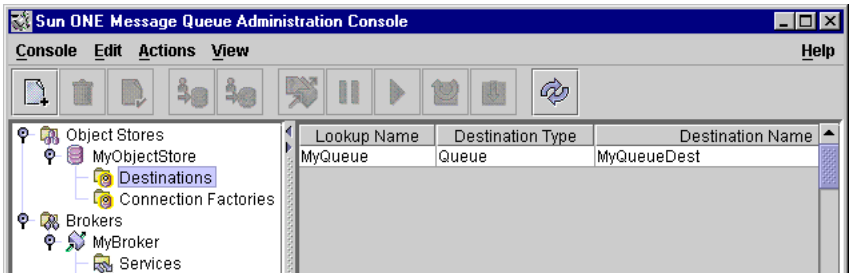
   ```
   queue=(javax.jms.Queue)ctx.lookup("MyQueue");
   ```

4. 对于 “Destination Type”，请选择 “Queue”单选按钮。
5. 在 “Destination Name”字段中输入 `MyQueueDest`。

   这是您在代理上添加物理目标时指定的名称。

6. 单击 “OK”。

7. 在浏览窗格中选择 “Destinations”，并注意在结果窗格中如何显示有关刚才添加的队列目标被管理对象的信息。



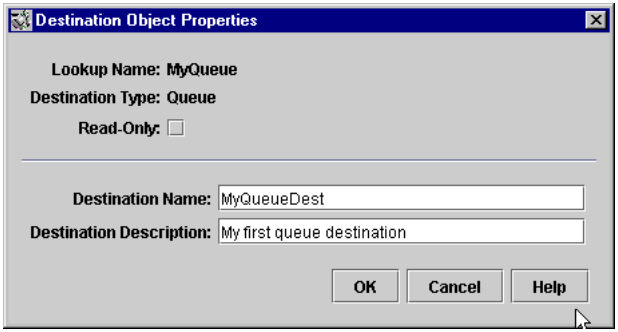
被管理对象的属性

要查看或更新被管理对象的属性，需要在浏览窗格中选择 “Destinations”或 “Connection Factories”，然后在结果窗格中选择特定对象，并选择 “Actions” > “Properties”。

➤ **查看或更新目标对象属性的步骤**

1. 在浏览窗格中选择 “Destinations”。
2. 在结果窗格中选择 “MyQueue”。
3. 选择 “Actions” > “Properties”以查看 “Destination Object Properties”对话框。

   请注意，您仅能更改目标名称和说明的值。要更改查找名称，就必须先删除对象，然后添加一个命名为所需查找名称的新队列被管理对象。



更新控制台信息

不管使用对象存储还是使用代理，都可以通过选择 “View” > “Refresh”来更新任何元素或元素组的显示。

运行范例应用程序

范例应用程序 SimpleJNDIClient 是为与本教程配套使用而提供的。它使用在前面的教程中创建的被管理对象和目标：名为 MyQueueDest 的队列目标，名为 MyQueueConnectionFactory 的队列连接工厂被管理对象和名为 MyQueue 的队列被管理对象。

代码创建简单的队列发件人和收件人，并发送和接收一个 “Hello World”消息。

➤ 运行 SimpleJNDIClient 应用程序的步骤

1. 确保 JAVA_HOME 环境变量指向 JDK 的安装目录。
2. 使包含 SimpleJNDIClient 应用程序的目录成为当前目录，例如：

   cd IMQ_HOME/demo/jms （在 Solaris 操作系统上为 /usr/demo/imq/jms）

   您会找到 SimpleJNDIClient.class 文件。（如果要更改该应用程序，需要使用 《MQ *Developer's Guide*》的快速入门教程中用于编译客户机应用程序的指令对其重新编译。
3. 设置 CLASSPATH 变量以包含含有 SimpleJNDIClient.java 和以下 jar 文件的当前目录：jms.jar、 imq.jar 和 fscontext.jar。
4. 在运行应用程序之前，请打开源文件 SimpleJNDIClient.java，并仔细阅读。

   虽然这个源文件很简短，但是其注释非常详细，您可以很清楚地了解该应用程序是如何使用您已使用教程创建的被管理对象和目标。

5. 运行 `SimpleJNDIClient` 应用程序。

`C:> java SimpleJNDIClient` (Windows)

`% java SimpleJNDIClient file:///tmp`（在 Solaris 和 Linux 上）

如果应用程序成功运行，应会看到以下输出：

```
==================================================
$JAVA SimpleJDNIClient file:///tmp
Usage:java SimpleJDNIClinet [Context.PROVIDER_URL]

On Unix:
        java SimpleJDNIClient file:///tmp
On Windows:
        java SimpleJDNIClient file:///C:Temp

Using Windows default file:///C/Temp for Context.PROVIDER_URL

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

第 5 章

启动与配置代理

安装 Sun™ ONE Message Queue (MQ) 以后，可以使用 `imqbrokerd` 命令来启动代理。代理实例的配置由一组配置文件和使用 `imqbrokerd` 命令传送的选项共同控制，这些选项会覆盖配置文件中相应的属性。

本章说明了 `imqbrokerd` 命令的语法，以及如何使用命令行选项和配置文件来配置代理实例。此外，本章还介绍了如何执行以下操作：

- 编辑代理实例配置文件
- 使用代理群集
- 控制代理的日志记录

关于如何启动代理并将其用作 Windows 服务的说明，请参见第 217 页上的 “使用代理作为 Windows 服务”。

配置文件

用于配置代理的配置文件将安装在以下目录中。

IMQ_HOME/lib/props/broker
（在 Solaris 操作系统中为 /usr/share/lib/imq/props/broker）

该目录存储以下文件：

- 启动时装入的默认配置文件。文件名是 default.properties。该文件不可编辑。您可能需要读取该文件来确定默认设置以及查找要更改的属性的确切名称。
- 安装配置文件，其中包含安装 MQ 时指定的所有属性。文件名是 install.properties。安装完成后该文件不可编辑。

此外，首次运行一个代理时，系统将创建一个实例配置文件，用于指定该代理的实例的配置属性。该文件由代理实例根据管理命令来进行维护，直接编辑该文件时请务必小心。实例配置文件存储在以下位置：

IMQ_VARHOME/instances/*brokerName*/props/config.properties
（在 Solaris 中为
/var/imq/instances/*brokerName*/props/config.properties）

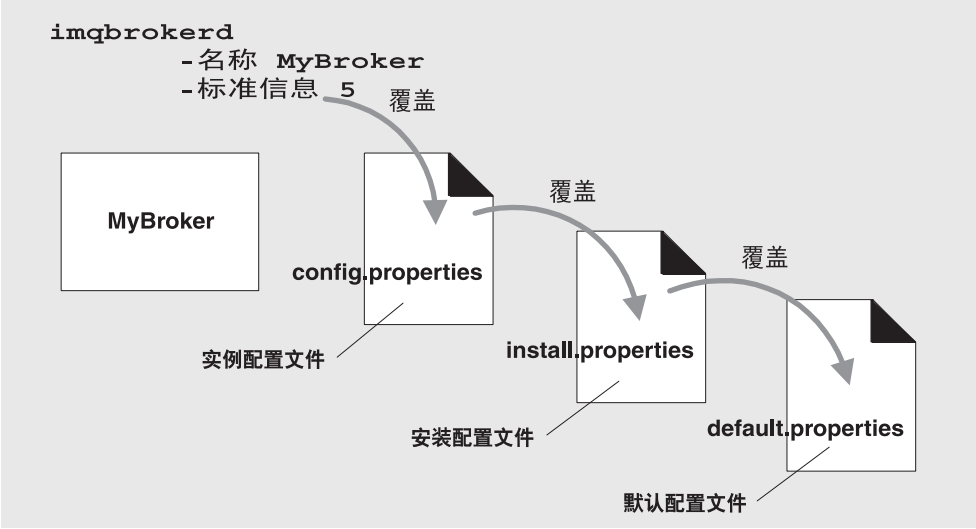
其中 *brokerName* 是代理实例的名称 （默认值是 imqbroker）。您可以编辑实例配置文件来更改配置 （请参见第 110 页上的 “编辑实例配置文件”）。

如果连接群集中的代理实例 （请参见第 64 页上的 “多代理群集 （企业版）”），那么可能还需要使用**群集配置文件**来指定群集配置信息。更多信息，请参见第 118 页上的 “群集配置属性”。

合并属性值

启动时，系统会合并不同配置文件中的属性值。它将使用在安装配置文件和实例配置文件中设置的值来覆盖默认配置文件中指定的值。您可以用 imqbrokerd 命令选项来覆盖得到的值。第 109 页上的图 5-1 所示为以上说明的图解。

图 5-1 代理配置文件



属性命名语法

配置文件中的所有 MQ 属性定义均使用以下命名语法：

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

例如，以下项目定义了自动创建队列的队列类型：

```
imq.queue.default=single
```

以下项目定义了消息失效期的超时值：

```
imq.message.expiration.timeout=90
```

第 110 页上的表 5-1 按字母顺序列出了代理配置属性及其默认值。

编辑实例配置文件

首次运行代理实例时，系统将自动创建一个 config.properties 文件。您可以编辑该实例配置文件，以自定义相应代理实例的行为和资源使用。

代理实例仅在启动时读取 config.properties 文件。要持久性更改 config.properties 文件，您可以：

- 使用管理工具。关于可以使用 imqcmd 设置的属性的信息，请参见第 138 页上的表 6-5。
- 在代理实例关闭时编辑 config.properties 文件，然后重新启动实例。（在 Solaris 和 Linux 平台中，只有第一个启动代理实例的用户具有编辑 config.properties 文件的权限。）

表 5-1 按字母顺序列出了代理配置属性及其默认值。关于各个属性的含义和用法的更多信息，请参考指定的交叉参考的小节。

表 5-1 代理实例配置属性

属性名称	类型	默认值	参考
imq.accesscontrol.enabled	布尔值	true	第 57 页上的表 2-6
imq.accesscontrol.file.filename	字符串	accesscontrol.properties	第 57 页上的表 2-6
imq.authentication.basic.user_repository	字符串	file	第 57 页上的表 2-6
imq.authentication.client.response.timeout	整数 （秒）	180	第 57 页上的表 2-6
imq.authentication.type	字符串	digest	第 57 页上的表 2-6
imq.autocreate.queue	布尔值	true	第 64 页上的表 2-9
imq.autocreate.topic	布尔值	true	第 64 页上的表 2-9
imq.cluster.url	字符串	null	第 68 页上的表 2-10
imq.keystore.*property_name*			第 185 页上的表 8-8
imq.log.console.output	字符串	ERROR|WARNING	第 60 页上的表 2-8
imq.log.console.stream	字符串	ERR	第 60 页上的表 2-8

1 键入为**字节字符串**的值，可以使用字节、千字节和兆字节的形式来表示：例如：1000 表示 1000 字节； 7500b 表示 7500 字节； 77k 表示 77 千字节 （77 x 1024 = 78848 字节）； 17m 表示 17 兆字节 （17 x 1024 x 1024 = 17825792 字节）

表 5-1 代理实例配置属性 （续）

属性名称	类型	默认值	参考
imq.log.file.dirpath	字符串	IMQ_VARHOME/instances/*brokerName*/log （在 Solaris 上为 /var/imq/...）	第 60 页上的表 2-8
imq.log.file.name	字符串	log.txt	第 60 页上的表 2-8
imq.log.file.output	字符串	ALL	第 60 页上的表 2-8
imq.log.file.rolloverbytes	整数 （字节）	0	第 60 页上的表 2-8
imq.log.file.rolloversecs	整数 （秒）	604800	第 60 页上的表 2-8
imq.log.level	字符串	INFO	第 60 页上的表 2-8
imq.log.syslog.facility	字符串	LOG_DAEMON	第 60 页上的表 2-8
imq.log.syslog.logpid	布尔值	true	第 60 页上的表 2-8
imq.log.syslog.logconsole	布尔值	false	第 60 页上的表 2-8
imq.log.syslog.identity	字符串	imqbrokerd_${imq.*brokerName*}	第 60 页上的表 2-8
imq.log.syslog.output	字符串	ERROR	第 60 页上的表 2-8
imq.message.expiration.interval	整数 （秒）	60	第 52 页上的表 2-4
imq.message.max_size	字节字符串 [1] 0 （没有限制）	70m	第 52 页上的表 2-4
imq.metrics.enabled	布尔值	true	第 60 页上的表 2-8
imq.metrics.interval	整数 （秒）	0	第 60 页上的表 2-8
imq.passfile.enabled	布尔值	false	第 57 页上的表 2-6
imq.passfile.dirpath	字符串	IMQ_HOME/etc （在 Solaris 上为 /etc/imq）	第 57 页上的表 2-6
imq.passfile.name	字符串	passfile	第 57 页上的表 2-6
imq.persist.file.destination.file.size	字节字符串 [1]	1m	第 54 页上的表 2-5
imq.persist.file.message.cleanup	布尔值	false	第 54 页上的表 2-5

1 键入为**字节字符串**的值，可以使用字节、千字节和兆字节的形式来表示：例如：1000 表示 1000 字节； 7500b 表示 7500 字节； 77k 表示 77 千字节 （77 x 1024 = 78848 字节）； 17m 表示 17 兆字节 （17 x 1024 x 1024 = 17825792 字节）

表 5-1 代理实例配置属性 （续）

属性名称	类型	默认值	参考
imq.persist.file.message.fdpool.limit	整数	25 （Solaris 和 Linux） 1024 (Windows)	第 54 页上的表 2-5
imq.persist.file.message.filepool.cleanratio	整数	0	第 54 页上的表 2-5
imq.persist.file.message.filepool.limit	整数	10000	第 54 页上的表 2-5
imq.persist.file.sync.enabled	布尔值	false	第 54 页上的表 2-5
imq.persist.jdbc.*property_name*			第 193 页上的表 A-1
imq.persist.store	字符串	file	第 54 页上的表 2-5
imq.portmapper.port	整数	7676	第 48 页上的表 2-3
imq.queue.deliverypolicy	字符串	single	第 64 页上的表 2-9
imq.redelivered.optimization	布尔值	true	第 52 页上的表 2-4
imq.*resource_state*.threshold	整数 （百分比）	0 （green） 60 （yellow） 75 （orange） 90 （red）	第 52 页上的表 2-4
imq.service.activelist	列表	jms、 admin	第 48 页上的表 2-3
imq.*service_name*.accesscontrol.enabled	布尔值	从系统范围属性继承值	第 57 页上的表 2-6
imq.*service_name*.accesscontrol.file.filename	字符串	从系统范围属性继承值	第 57 页上的表 2-6
imq.*service_name*.authentication.type	字符串	从系统范围属性继承值	第 57 页上的表 2-6
imq.*service_name*.max_threads	整数	1000 (jms) 500 (ssljms) 500 (httpjms) 500 (httpsjms) 50 (admin)	第 48 页上的表 2-3

1 键入为**字节字符串**的值，可以使用字节、千字节和兆字节的形式来表示：例如：1000 表示 1000 字节； 7500b 表示 7500 字节； 77k 表示 77 千字节 （77 x 1024 = 78848 字节）； 17m 表示 17 兆字节 （17 x 1024 x 1024 = 17825792 字节）

表 5-1 代理实例配置属性 （续）

属性名称	类型	默认值	参考
`imq.`*service_name*`.min_threads`	整数	`10` (jms) `10` (ssljms) `10` (httpjms) `10` (httpsjms) `4` (admin)	第 48 页上的表 2-3
`imq.`*service_name*`.`*protocol type*`.hostname`	字符串	`null`	第 48 页上的表 2-3
`imq.`*service_name*`.`*protocol type*`.port`	整数	`0`	第 48 页上的表 2-3
`imq.`*service_name*`.threadpool_model`	字符串	`dedicated` (jms) `dedicated` (ssljms) `dedicated` (httpjms) `dedicated` (httpsjms) `dedicated` (admin)	第 48 页上的表 2-3
`imq.shared.connectionMonitor_limit`	整数	`512` （Solaris 和 Linux） 64 (Windows)	第 48 页上的表 2-3
`imq.system.max_count`	整数、0（没有限制）	`0`	第 52 页上的表 2-4
`imq.system.max_size`	字节字符串 [1]、0（没有限制）	`0`	第 52 页上的表 2-4
`imq.transaction.autorollback`	布尔值	`false`	第 52 页上的表 2-4
`imq.user_repository.ldap.`*property_name*			第 176 页上的表 8-5

1 键入为**字节字符串**的值，可以使用字节、千字节和兆字节的形式来表示：例如：1000 表示 1000 字节； 7500b 表示 7500 字节； 77k 表示 77 千字节 （77 x 1024 = 78848 字节）； 17m 表示 17 兆字节 （17 x 1024 x 1024 = 17825792 字节）

启动代理

要启动代理实例，请使用 imqbrokerd 命令。

注 使用管理控制台 (imqadmin) 或命令实用程序 (imqcmd) 无法启动代理实例。这两个 MQ 管理工具只有代理实例已在运行时才能使用。

要覆盖一个或多个属性值，请指定有效的 imqbrokerd 命令行选项。命令行选项覆盖代理配置文件中的值，但仅对当前代理会话有效：命令行选项不会被写入到实例配置文件中。

imqbrokerd 命令的语法如下所示 （选项与变量之间以一个空格分隔）：

```
imqbrokerd [[ -Dproperty=value]...]
     [ -backup fileName]
     [ -cluster "[broker] [[,broker]...]"
     [ -dbuser userName] [ -dbpassword password]
     [ -force]
     [ -h]
     [ -javahome path | -jrehome path]
     [ -ldappassword password]
     [ -license name]
     [ -loglevel level]
     [ -metrics number]
     [ -name brokerName ] [ -port number]
     [ -shared]
     [ -password keypassword] [ -passfile fileName]
     [ -remove instance]
     [ -reset data]
     [ -restore fileName]
     [ -shared]
     [ -silent] [ -tty]
     [ -version] [ -vmargs arg [[arg]...]
```

例如，要启动使用默认代理名称和配置的代理，请使用以下命令：

```
imqbrokerd
```

该命令启动一个位于本地计算机上的代理 （名为 imqbroker）的默认实例，且其端口映射器的端口号是 7676。

注 在 Solaris 中，可以将代理配置为异常退出后自动重新启动，方法是把 /etc/imq/imqborkerd.conf 配置文件中的 RESTART 属性设置为 YES。

注 在 Solaris 和 Linux 平台中，对包含配置信息和持久性数据的目录的权限取决于第一次启动代理实例的用户的 umask。因此，为了让代理实例正常运行，就只能由原用户启动该实例。

表 5-2 说明了 imqbrokerd 命令的选项以及受到各个选项影响的配置属性 （如果存在）。

表 5-2 imqbrokerd 选项

选项	所影响的属性	说明
-backup *fileName*	不影响任何属性。	仅适用于代理群集。将主管代理的配置更改记录备份到指定的文件。请参见第 122 页上的“备份主管代理的配置更改记录”。
-cluster"[*broker*][[,*broker*]...]" *broker* 可以是 • *host*[:*port*] • [*host*]:*port*	将 imq.cluster.brokerlist 设置为要连接的代理的列表。	仅适用于代理群集。连接至指定的主机和端口上的所有代理。该列表将与 imq.cluster.brokerlist 属性中的列表合并。如果没有指定 host 的值，将使用 localhost。如果没有指定 *port* 的值，将使用值 7676。关于如何使用该选项连接多个代理的更多信息，请参见第 118 页上的“使用群集（企业版）”。
-dbpassword *password*	将 imq.persist.jdbc.password 设置为指定密码	指定插入的 JDBC 兼容数据存储的密码。请参见附录 A“设置插入的持久性”。
-dbuser *userName*	将 imq.persist.jdbc.user 设置为指定用户名	指定插入的 JDBC 兼容数据存储的用户名。请参见附录 A“设置插入的持久性”。
-D*property*=*value*	设置系统属性。覆盖实例配置文件中的相关属性值。	将指定属性设置为指定值。关于代理配置属性，请参见第 110 页上的表 5-1。 **注意**：请仔细检查使用 D 选项设置的属性的拼写和格式。如果传送了不正确的值，系统不会向您发出警告， MQ 将无法对这些属性进行设置。

表 5-2 imqbrokerd 选项 （续）

选项	所影响的属性	说明
-force	不影响任何属性。	执行操作，而无需用户确认。该选项仅适用于 -remove instance 选项，后者通常需要确认。
-h	不影响任何属性。	显示帮助。不执行命令行上的其它选项。
-javahome *path*	不影响任何属性。	指定替代 Java 2 兼容 JDK 的路径。默认使用捆绑的运行时。
-jrehome *path*	不影响任何属性。	指定 Java 2 JRE 的路径。
-ldappassword *password*	将 imq.user_repository.ldap.password 设置为指定密码	指定访问 LDAP 用户系统信息库的密码。请参见第 176 页上的 “使用 LDAP 服务器管理用户系统信息库”。
-license [*name*]	不影响任何属性。	指定装入一个与 MQ 产品版本的默认许可证不同的许可证。如果不指定许可证名称，该选项将列出系统中安装的所有许可证。根据所安装的 MQ 版本的不同， *name* 的取值也不同：pe（平台版，只具有基本功能）、try（平台版的 90 天试用企业功能）和 unl （企业版）。请参见第 26 页上的 “产品版本”。
-loglevel *level*	将 imq.broker.log.level 设置为指定级别。	将日志记录级别指定为下列四项之一：NONE、ERROR、WARNING 或 INFO。默认值是 INFO。更多信息，请参见第 59 页上的 “日志记录器”。
-metrics *int*	将 imq.metrics.report.interval 设置为指定秒数。	指定报告测量信息的时间间隔 （以秒为单位）。
-name *brokerName*	将 imq.instancename 设置为指定名称。	指定该代理的实例名称，并使用相应的实例配置文件。如果不指定代理名称，文件名将设置为 imqbroker。 **注**：如果在同一台主机上运行了多个代理实例，每个实例必须具有唯一的名称。
-passfile *fileName*	将 imq.passfile.enabled 设置为 true。将 jmq.passfile.dirpath 设置为包含文件的路径。将 imq.passfile.name 设置为该文件名。	指定文件名，从该文件中可以读取 SSL 密钥存储、 LDAP 用户系统信息库或 JDBC 兼容数据库的密码。更多信息，请参见第 189 页上的 “使用密码文件”。
-password *keypassword*	将 imq.keystore.password 设置为指定密码。	指定 SSL 证书密钥存储的密码。更多信息，请参见第 55 页上的 “安全性管理器”。

表 5-2 imqbrokerd 选项 （续）

选项	所影响的属性	说明
-port *number*	将 imq.portmapper.port 设置为指定端口号。	指定代理的端口映射器的端口号。默认情况下，该端口号设置为 7676。要在同一台服务器上运行两个代理实例，每个代理的端口映射器必须拥有不同的端口号。 JMS 客户机使用该端口号连接到代理实例。
-remove instance	不影响任何属性。	导致代理实例被删除：删除与实例相关联的实例配置文件、日志文件、持久性存储器与其它文件和目录。除非指定 -force 选项，否则需要用户确认。
-reset store| messages| durables| props	不影响任何属性。	根据给定变量，重置代理实例的数据存储 （或数据存储的子集）或配置属性。 重置数据存储将清除所有的持久性数据，包括持久性消息、长期订阅和事务信息。这使您可以启动一个完全清空的代理实例。也可以仅清除所有的持久性消息或仅清除所有的长期订阅。（如果不希望下次启动代理实例时重置持久性存储器，请勿使用 -reset 选项重新启动代理实例。）更多信息，请参见第 52 页上的“持久性管理器”。 重置代理的属性，用一个空文件代替现有的实例配置文件 (config.properties)：所有的属性将使用默认值。
-restore *fileName*	不影响任何属性。	仅适用于代理群集。用指定的备份文件替换主管代理的配置更改记录。该文件必须是原来用 -backup 选项创建的那个备份文件。请参见第 122 页上的 “恢复主管代理的配置更改记录”。
-shared	将 imq.jms.threadpool_model 设置为 shared。	指定使用共享线程池模型来实现 jms 连接服务。在该模型中，连接之间共享线程，以增加代理实例支持的连接数量。更多信息，请参见第 46 页上的 “连接服务”。
-silent	将 imq.log.console.output 设置为 NONE。	停止向控制台记录日志信息。
-tty	将 imq.log.console.output 设置为 ALL	指定在控制台上显示所有消息。默认情况下，只显示 WARNING 和 ERROR 级别的消息。

表 5-2 imqbrokerd 选项 （续）

选项	所影响的属性	说明
-version	不影响任何属性。	显示所安装产品的版本号。
-vmargs *arg* [[*arg*] . . .]	不影响任何属性。	指定传送到 Java VM 的变量。变量之间用空格分隔。如要传送多个变量，或是变量内包含空格，请使用闭合的引号。例如： imqbrokerd -tty -vmargs "-Xmx128m -Xincgc"

使用群集 （企业版）

本节介绍用于配置多代理群集的属性，并介绍了连接代理的两种方法，并说明如何管理群集。关于群集的介绍，请参见第 64 页上的 “多代理群集 （企业版）”。

使用群集时，请确保同一个群集中所有代理的主机时钟处于同步状态。

群集配置属性

将代理连接到群集时，所有连接的代理必须指定相同的群集配置属性值。这些属性说明了群集中代理的合作。表 5-3 概述了与群集相关的配置属性。

表 5-3 群集配置属性

属性	说明
imq.cluster.brokerlist	以列表方式指定群集中的所有代理，代理之间用逗号分隔；每一项指定一个代理的主机和端口。 例如：host1:3000, host2:8000, ctrhost
imq.cluster.masterbroker	指定主管代理的主机和端口。对生产环境设置该值。 例如， ctrhost:7676
imq.cluster.url	指定群集配置文件的位置。例如： http://webserver/imq/cluster.properties file:/net/mfsserver/imq/cluster.properies
imq.cluster.port	用于为群集中的**每个**代理指定群集连接服务的端口号。群集连接服务用于群集中代理之间的内部通信。 默认值：0 （动态分配端口）

表 5-3 群集配置属性 （续）

属性	说明
imq.cluster.hostname	对于群集中的**每个**代理，如果有多个主机可用 （例如，一台计算机中安装了多个网络接口卡），群集连接服务将绑定到该属性指定的主机 （主机名或 IP 地址）。群集连接服务用于群集中代理之间的内部通信。 默认值：null （所有可用主机）

可以使用以下两种方法之一来设置群集属性：

- 可以在每个代理的实例配置文件中 （或启动各个代理的命令行中）设置与群集相关的配置属性。例如，要连接代理 A （位于 host1、 7676 端口）、代理 B （位于 host2、 5000 端口）和代理 C （位于 ctrlhost、 7676 端口），则代理 A、 B 和 C 的实例配置文件需要设置如下属性。

  ```
  imq.cluster.brokerlist=host1, host2:5000, ctrlhost
  ```

  如果决定更改群集配置，此方法要求更新所有代理中与群集相关的属性。

- 在一个中心群集配置文件中设置群集配置属性。这些属性可能包括要连接的代理列表 (imq.cluster.brokerlist) 和主管代理的地址 (imq.cluster.masterbroker)，后一项为可选项。

  如果使用该方法，还必须将群集中的每个代理的 imq.cluster.url 属性设置为指向群集配置文件的位置。从易于维护的角度出发，建议使用这种方法来配置群集。

以下代码范例显示了群集配置文件的内容。 host1 和 ctrlhost 都是在默认端口上运行。这些属性指定 host1 和 ctrlhost 连接到一个群集中，并指定 ctrlhost 作为主管代理。

```
imq.cluster.brokerlist=host1,host2:5000,ctrlhost
imq.cluster.masterbroker=ctrlhost
```

连接到此群集中的每个代理的实例配置文件都必须包含群集配置文件的 URL，例如：

```
imq.cluster.url=file:/home/cluster.properties
```

连接代理

本节介绍将代理连接到群集中的两种方法。无论使用哪种方法，启动的每个代理将每隔五秒钟就尝试一次连接到其它代理。群集中的其它代理启动后，该尝试就会成功。

如果将代理连接到群集，并不一定要首先启动主管代理。如果群集中的某个代理在主管代理之前启动，该代理将保持暂停状态，拒绝客户机连接。主管代理启动后，暂停的代理将自动进入正常运行状态。

方法 1：无群集配置文件

➤ **将代理连接到群集的步骤**

1. 使用带有 `-cluster` 选项的 `imqbrokerd` 命令来启动代理，并将 （要连接的）代理的完整列表指定为 `-cluster` 选项的变量。
2. 请在启动要连接到群集的每个代理时执行该操作。

   例如，以下命令启动了一个新代理，并将其连接到运行于 host1 的默认端口上的代理、运行于 host2 的 7677 端口上的代理和运行于 localhost 的 7678 端口上的代理。

   ```
   imqbrokerd -cluster host1,host2:7677,:7678
   ```

方法 2：使用群集配置文件

也可以创建一个群集配置文件，来指定要连接的代理列表 （可选：指定主管代理的地址）。这种定义群集的方法更适合生产系统。请注意，必须将群集中的每个代理的 `imq.cluster.url` 属性值设置为指向群集配置文件。

将代理添加到群集

设置代理群集后，可能需要添加新代理或重新启动群集中原有的代理。

要将新代理添加到现有群集，可以执行以下操作之一：

如果没有使用群集配置文件，那么启动新代理时，在命令行中使用 `-D` 选项指定 `imq.cluster.brokerlist` 和 （如果必要） `imq.cluster.masterbroker` 属性。

➤ **将代理添加到群集中的步骤（如果是使用群集配置文件）**

1. 将新代理添加到群集配置文件中的 `imq.cluster.brokerlist` 属性。
2. 对群集中的任何代理执行以下命令。

   ```
   imqcmd reload cls
   ```

   该命令强制所有的代理重新装入 `imq.cluster.brokerlist` 属性，以保证群集中代理的所有持久性信息都是最新的。

重新启动群集中的代理

要重新启动群集中的现有代理，可以执行以下操作之一：

- 如果使用群集配置文件定义群集，请在用于启动代理的命令行中使用 -D 选项来指定 `imq.cluster.url` 属性。
- 如果不是使用群集配置文件定义群集，那么启动新代理时，请在命令行中使用 -D 选项指定 `imq.cluster.brokerlist` 属性（如有必要，请一并指定 `imq.cluster.masterbroker` 属性）。如果群集中没有主管代理，可以在启动新代理仅使用 `-cluster` 选项来指定群集中的代理列表。

从群集中删除代理

从群集中删除代理时，请注意以下几点：

- 如果代理 A、B 和 C 均是使用以下命令行启动，那么仅仅重新启动 A 将无法将其从群集中删除。

  ```
  imqbrokerd -cluster A,B,C
  ```

  而是需要使用以下命令行重新启动所有的其它代理：

  ```
  imqbrokerd -cluster B,C
  ```

  然后，需要启动代理 A，且不指定 `-cluster` 选项。
- 如果代理列表是使用群集配置文件来指定，那么需要执行以下操作：
  - 从配置文件中删除该代理的相关内容。
  - 更改或删除要删除的代理的 `imq.cluster.url` 属性，这样代理就不会再使用该通用属性。
  - 使用 `imqcmd reload cls` 命令强制所有的代理重新装入它们的群集配置，从而重新配置群集。

备份主管代理的配置更改记录

每个群集都可以拥有一个主管代理，它将记录群集的持久性状态的所有更改，包括对长期订阅和管理员创建的物理目标的更改。所有的代理在启动时都会咨询主管代理，以同步关于这些持久性对象的信息。因此，主管代理发生故障会影响整个群集。有鉴于此，使用 `imqbrokerd` 命令的 `-backup` 选项来定期备份主管代理的更改记录是很重要的。例如，

```
imqbrokerd -backup mybackuplog
```

请及时进行如上的备份操作，这很重要。恢复一个很旧的备份可能会导致丢失信息：该备份最后一次完成之后创建的任何持久性对象都将丢失。

恢复主管代理的配置更改记录

➤ **万一发生故障时恢复主管代理的步骤**

1. 关闭群集中的所有代理。
2. 用以下命令恢复主管代理的配置更改记录：

   ```
   imqbrokerd  -restore mybackuplog
   ```

3. 如果要给主管代理分配新名称或端口号，必须更新群集配置文件，指定主管代理是群集中的代理之一，并指定它的新名称 （使用 `imq.cluster.masterbroker` 属性）。
4. 重新启动所有代理。

恢复代理将不可避免地导致某些过时的数据重新装入到代理的配置更改记录中。不过，只要经常进行前一小节所介绍的定期备份，可以将这个问题的影响减到最小。

因为主管代理记录对持久性对象的所有更改的历史记录，所以每过一段时间其数据库就会有显著的扩充。备份和恢复操作有利于压缩和优化该数据库。

日志记录

本节介绍了代理的默认日志记录配置，并说明了如何更改该配置，从而将日志信息重定向到替代输出通道、更改转移标准以及报告代理的测量信息。关于日志记录的介绍，请参见第 59 页上的 “日志记录器”。

默认日志记录配置

启动代理时，系统自动将其配置为将日志输出保存到以下位置的一组滚动生成的日志文件中：

IMQ_VARHOME/instances/*brokerName*/log/
（在 Solaris 中为 /var/imq/instances/*brokerName*/log/）

日志文件是纯文本文件。它们的名称如下 （从最早生成到最近生成）：

```
log.txt
log_1.txt
log_2.txt
...
log_9.txt
```

默认情况下，日志文件每星期转移一次；系统维护九个备份文件。

- 要更改保存日志文件的目录，请将 imq.log.file.dirpath 属性设置为所需的路径。
- 要将日志文件的根名称由 log 改为其它名称，请设置 imq.log.file.filename 属性。

代理支持三种日志种类：ERROR、 WARNING 和 INFO （请参见第 59 页上的表 2-7）。设置日志记录级别后，将收集所有高于和等于该级别的消息。默认日志级别是 INFO。这表示将记录 ERROR、 WARNING 和 INFO 消息。

日志消息格式

记录的消息由时间戳、消息代码和消息本身构成。信息的大小取决于所设置的日志级别。以下是一个 INFO 消息的实例。

```
[13/Sep/2000:16:13:36 PDT] B1004 Starting the broker service
using tcp [ 25374,100] with min threads 50 and max threads of 500
```

更改日志记录器配置

第 60 页上的表 2-8 说明了所有的日志记录器属性。

➤ 更改代理的日志记录器配置的步骤

1. 设置日志级别。
2. 设置一个和多个日志记录种类的输出通道 （文件、控制台或两者）。
3. 如果将输出记录到文件中，请为该文件配置转移标准。

可以通过设置日志记录器属性完成上述步骤。可以使用以下两种方法之一来进行设置：

- 启动代理之前，在该代理的 `config.properties` 文件中更改或添加日志记录器属性。
- 在启动代理的 `imqbrokerd` 命令中指定日志记录器命令行选项。也可以使用代理选项 `-D` 来更改日志记录器属性 （或**任何**代理属性）。

在命令行上传送的选项将覆盖代理实例配置文件中指定的属性。表 5-4 列出了影响日志记录的 `imqbrokerd` 选项。

表 5-4 `imqbrokerd` 日志记录器选项及相应的属性

imqbrokerd 选项	说明
`-metrics` *number*	指定收集测量信息的时间间隔 （以秒为单位）。
`-loglevel` *level*	设置日志级别为以下三项之一：`ERROR`、 `WARNING` 或 `INFO`。
`-silent`	停止向控制台记录日志信息。
`-tty`	将所有消息发送到控制台。默认情况下，只显示 `WARNING` 和 `ERROR` 级别的消息。

以下小节介绍了如何更改默认配置，以执行以下操作：

- 更改输出通道 （日志消息的目标）
- 更改转移标准
- 记录代理测量信息

更改输出通道

默认情况下，错误和警告消息除了记录到日志文件中以外，还会显示在终端上。（在 Solaris 中，错误消息还会写入到系统的系统日志守护程序中。）

可以用以下方式更改日志消息的输出通道：

- 要在屏幕上显示**所有**的日志种类（对于给定的级别）输出，请在 `imqbrokerd` 命令中使用 `-tty` 选项。
- 要阻止在屏幕上显示日志输出，请在 `imqbrokerd` 命令中使用 `-silent` 选项。
- 使用 `imq.log.file.output` 属性指定将哪些种类的日志记录信息写入到日志文件。例如，

  ```
  imq.log.file.output=ERROR
  ```

- 使用 `imq.log.console.output` 属性指定将哪些种类的日志记录信息写入到控制台。例如，

  ```
  imq.log.console.output=INFO
  ```

- 在 Solaris 中，使用 `imq.log.syslog.output` 属性指定将哪些种类的日志记录信息写入到 Solaris 系统日志中。例如，

  ```
  imq.log.syslog.output=NONE
  ```

注 更改日志消息的目标之前，必须确保所设置的日志记录级别与映射到输出通道的日志种类相对应。例如，如果将日志级别设置为 `ERROR`，然后将 `imq.log.console.output` 属性设置为 `WARNING`，那么将不会记录任何消息，因为没有启用对这些级别的消息的记录。

更改转移标准

转移日志文件有两个标准：时间和大小。默认是使用时间标准，每七天转移一次文件。

- 要更改时间间隔，需要更改 `imq.log.file.rolloversecs` 属性。例如，以下属性定义将时间间隔更改为十天：

  ```
  imq.log.file.rolloversecs=864000
  ```

- 要将转移标准更改为取决于文件大小，需要设置 `imq.log.file.rolloverbytes` 属性。例如，以下定义将在文件达到 500,000 字节的限制时将代理定向到转移文件。

  ```
  imq.log.file.rolloverbytes=500000
  ```

如果同时设置与时间相关和与大小相关的转移属性，首先达到的限制将触发转移。如前所述，代理最多维护九个转移文件。

记录代理性能测量信息

代理的默认配置，包括以下设置：

- `imq.metrics.enabled=true`
- `imq.metrics.interval=0`
- `imq.log.level=INFO`

这些设置的结果是，代理收集它本身和活动连接服务的性能测量信息，但并不生成测量信息报告。

可以使用以下两种方法之一来使代理生成测量信息报告：

- 在 `imqbrokerd` 命令中使用 `-metrics` 选项，并指定代理生成报告的时间间隔（以秒为单位）。
- 将 `imq.metrics.interval` 属性设置为希望代理生成报告的时间间隔 （以秒为单位）。

因为测量信息报告包含在 `INFO` 种类中，所以默认情况下，测量信息报告会被写入到日志文件输出通道。

以下所示为范例测量信息：

```
[31/Jan/2001:15:00:50 PST]
Connections:0 JVM Heap:6291456 bytes (5186320 free)
      In:0 mesgs (0bytes) 0 pkts (0 bytes)
     Out:0 mesgs (0bytes) 0 pkts (0 bytes)
 Rate In:0 msgs/sec (0 bytes/sec) 0 pkts/sec (0 bytes/sec)
Rate Out:0 msgs/sec (0 bytes/sec) 0 pkts/sec (0 bytes/sec)
```

表 5-5 说明了对每个连接服务所生成的测量信息的含义。

表 5-5 对连接服务收集的测量信息

测量信息	说明
Pkts in (total)	上次重置以后，代理已读取的包的总数。其中包括 MQ 协议包，而不仅仅是 JMS 消息。
Pkts out (total)	上次重置以后，代理已写入的包的总数。
JMS Messages in (total)	上次重置以后，代理已读取的 JMS 消息的总数。
JMS Messages out (total)	上次重置以后，代理已写入的 JMS 消息的总数。
Message Bytes in (total)	上次重置以后，代理已读取的消息的总字节数。
Message Bytes out (total)	已写入的消息的总字节数。
Current # connections	当前打开的连接的数目。

表 5-6 介绍了对每个代理所收集和报告的测量信息。

表 5-6 对每个代理所收集的测量信息

测量信息	说明
VM heap size (bytes)	Java VM 堆的最大大小。
VM heap free space (bytes)	Java VM 堆中所剩可用空间的容量。

注 使用 `imqcmd metrics` 命令也可以获得该信息。

第 6 章

代理和应用程序管理

本章介绍如何执行与管理代理及其提供的服务相关的任务。某些任务并不是针对特定的客户机应用程序。这些任务包括：

- 控制代理的状态：可以暂停、恢复、关闭和重新启动代理。
- 查询和更新代理属性
- 查询和更新连接服务
- 分配和管理资源
- 管理连接服务

其它代理任务是在特定应用程序上执行的，包括管理物理目标、长期订阅和事务等任务：

- MQ 消息通过代理目标路由到接收者或订户。您负责在代理上创建这些目标。
- MQ 负责为长期订户分配并维护资源，即使这些客户机的长期订阅处于非活动状态。使用 MQ 命令工具可以获得长期订阅的相关信息，也可以销毁长期订阅或清理其中的消息以节约 MQ 资源。
- MQ 事务和分布式事务由代理来跟踪。如果出现错误，可能需要手动提交或回滚事务。

本章介绍如何使用命令行实用程序 (`imqcmd`) 来执行这些任务。许多任务还可以使用管理控制台 （MQ 消息服务器的图形界面）来完成。有关详细信息，请参阅第 4 章 “管理控制台教程”。

命令行实用程序

命令行实用程序可用于管理代理及其提供的服务。本节介绍了基本的 imqcmd 命令语法，提供了一个子命令列表，并概述了 imqcmd 的选项。后续各节将介绍如何使用这些命令完成特定的任务。

命令语法

imqcmd 命令的一般语法如下：

```
imqcmd subcommand argument [options]
imqcmd -h|H
imqcmd -v
```

请注意，如果指定 -v、 -h 或 -H 选项，将不会执行命令行中指定的子命令。例如，输入以下命令将显示版本信息，而不是执行 restart 子命令。

```
imqcmd restart bkr -v
```

imqcmd 子命令

表 6-1 列出了命令行实用程序 (imqcmd) 包含的子命令：

表 6-1 imqcmd 子命令

子命令和变量	说明
create dst	创建目标。
commit txn	提交事务。
destroy dst	销毁目标。
destroy dur	销毁长期订阅。
list dst	列出代理中的目标。
list dur	列出主题下的长期订阅。
list svc	列出代理中的服务。
list txn	列出代理中的事务。
metrics bkr	显示代理标准。
metrics svc	显示服务标准。
pause bkr	暂停代理中的所有服务。

表 6-1 imqcmd 子命令 （续）

子命令和变量	说明
pause svc	暂停某项服务。
purge dst	清理目标中的所有消息，但不销毁该目标。
purge dur	清理长期订阅中的所有消息，但不销毁该长期订阅。
query bkr	查询并显示代理中的信息。
query dst	查询并显示目标中的信息。
query svc	查询并显示服务中的信息。
query txn	查询并显示事务中的信息。
reload cls	重新装入代理群集配置。
restart bkr	重新启动当前正在运行的代理实例。不能用于启动新的代理实例。
resume bkr	恢复代理中的所有服务。
resume svc	恢复某项服务。
rollback txn	回滚某项事务。
shutdown bkr	关闭代理实例。关闭代理实例后可以使用 imqbrokerd 命令重新启动，但不能使用 imqcmd 的 restart bkr 子命令。
update bkr	更新某个代理的属性。
update dst	更新某个目标的属性。
update svc	更新某项服务的属性。

imqcmd 选项概述

表 6-2 列出了 imqcmd 命令的选项。有关如何使用它们的讨论，请参阅后面针对具体任务的各节。

表 6-2 imqcmd 选项

选项	说明
-b *hostName:port*	指定代理的主机名和端口号。默认值为 localhost:7676。 要仅指定端口，请使用：-b :7878 要仅指定主机名，请使用： -b 某台主机
-c *clientID*	指定订阅某个主题的长期订户的 ID。有关详细信息，请参阅第 146 页上的 “管理长期订阅”。
-d *topicName*	指定主题的名称。与 list dur 和 destroy dur 子命令一起使用。请参阅第 146 页上的 “管理长期订阅”。
-f	执行操作，无需用户确认。
-h	显示使用帮助。不执行命令行上的命令。
-H	显示使用帮助、属性列表和实例。不执行命令行上的命令。
-int *interval*	指定 imqcmd 显示代理标准的时间间隔，以秒为单位。（与 metrics 子命令一起使用。）
-javahome	指定可使用的替代 Java 2 兼容运行时 （默认使用产品附带的运行时）。
-m *metricType*	指定要显示的标准类型。可以是下列类型之一： ttl 流入和流出代理的所有消息 （默认）。 rts 与 ttl 提供相同的信息，但指定了每秒钟的消息数量。 cxn 连接、虚拟内存堆、线程 与 metrics bkr 或 metrics svc 子命令一起使用。以下命令每隔五秒钟显示一次默认代理的 cxn 类型的标准。 imqcmd metrics bkr -m cxn -int 5
-n *argumentName*	指定子命令参数的名称。根据子命令的不同，它可能是服务名称、物理目标、长期订阅或事务 ID。
-o *attribute=value*	指定属性的值。根据子命令参数的不同，它可能是代理属性（请参阅第 136 页上的 “查询和更新代理属性”）、服务属性（请参阅第 139 页上的 “管理连接服务”）或目标属性 （请参阅第 143 页上的 “管理目标”）。

表 6-2 imqcmd 选项 （续）

选项	说明
-secure	使用 ssladmin 连接服务指定代理的安全管理连接 （请参阅“步骤 4：配置并运行基于 SSL 的客户机”，第 188 页上的“命令行实用程序 (imqcmd)”）。
-p *password*	指定管理员的密码。如果省略此值，系统会提示您输入密码。
-s	无提示模式。不显示任何输出。
-t *destinationType*	指定目标的类型：t （主题）或 q （队列）。
-tmp	显示临时目标。
-u *name*	指定管理员的用户名。如果省略此值，系统会提示您输入用户名。
-v	显示版本信息。不执行命令行上的命令。

每次发出一个 imqcmd 子命令时，都必须指定以下选项：主机名和端口号 (-b)、用户名 (-u)、密码 (-p) 和安全连接 (-secure)。如果不指定主机名和端口号，系统将使用默认值。如果不指定用户名和密码，系统会提示您输入这些信息。如果不指定 -secure，连接将是不安全的。

使用 imqcmd 的前提条件

要使用 imqcmd 命令管理代理，必须执行以下操作：

- 使用 imqbrokerd 命令启动代理。

  请参阅第 114 页上的 “启动代理”。命令行实用程序只能用于管理正在运行的代理，不能用于启动代理。

- 如果代理没有在本地主机的 7676 端口上运行，必须使用 -b 选项指定目标代理。
- 指定适当的管理员用户名和密码。如果不指定，系统会提示您输入这些信息。总之请记住，使用 imqcmd 执行的每一个操作都要在用户系统信息库中进行验证。

安装 MQ 时，系统会安装默认的用户系统信息库，这是一个纯文本文件。文件名为 IMQ_HOME/etc/passwd（在 Solaris 操作系统中为 /etc/imq/passwd）。系统信息库出厂时包括两个条目：一个供管理员用户使用，另一个供临时用户使用。使用这两个条目即可建立代理连接，无需执行额外的操作。例如，如果只是要测试 MQ，可以使用默认的用户名和密码 (admin/admin) 来运行实用程序。

如果要设置生产系统，则需要完成一些额外的工作，对用户进行验证和授权。也可以将现有的 LDAP 目录服务器作为用户系统信息库。有关详细信息，请参阅第 170 页上的 “验证用户”。

实例

以下命令将列出运行在 localhost 7676 端口上的代理的属性：

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

以下命令将列出运行在 myserver 7676 端口上的代理的属性，用户名为 alladin，密码为 abracadabra。

```
imqcmd query bkr -b myserver:1564 -u alladin -p abracadabra
```

如果指定用户名 alladin 属于 admin 组，将以管理员客户机的身份连接到指定代理。

控制代理的状态

启动代理后，可以使用下列 imqcmd 子命令控制其状态。

- 暂停代理

  暂停代理将暂停代理服务线程，从而使代理停止监听端口。然后您可以执行控制代理消息流所需的管理任务。例如，当某个目标受到消息攻击时，您可以暂停该代理，然后执行下列操作之一，以期解决该问题：跟踪消息来源、限制目标的大小或销毁目标。

  以下命令将暂停运行在 myhost 1588 端口上的代理。

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

- 恢复代理

  恢复代理将重新激活代理服务线程，从而使代理恢复监听端口。以下命令将恢复运行在 `localhost` 7676 端口上的代理。

  ```
  imqcmd resume bkr
  ```

- 关闭代理

  关闭代理将终止代理进程。这属于正常终止：代理停止接收新的连接和消息，完成现有消息的传送后将终止代理进程。以下命令将关闭运行在 `ctrlsrv` 1572 端口上的代理。

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

- 重新启动代理

  关闭并重新启动代理。以下命令将重新启动运行在 `localhost` 7676 端口上的代理。

  ```
  imqcmd restart bkr
  ```

表 6-3 概述了用于控制代理的 `imqcmd` 子命令。请记住，除非您希望代理在 `localhost` 7676 端口上运行，否则必须指定代理主机名和端口号。

表 6-3 用于控制代理的 `imqcmd` 子命令

子命令	说明
`pause bkr [-b` *hostName:port*`]`	暂停默认代理或指定主机和端口上的代理。
`resume bkr [-b` *hostName:port*`]`	恢复默认代理或指定主机和端口上的代理。
`shutdown bkr [-b` *hostName:port*`]`	关闭默认代理或指定主机和端口上的代理。
`restart bkr [-b` *hostName:port*`]`	关闭并重新启动默认代理或指定主机和端口上的代理。 请注意，此命令将使用第一次启动该代理时指定的选项重新启动该代理。如果要执行不同的操作，必须先关闭代理，然后再使用所需的选项重新启动它。

查询和更新代理属性

命令行实用程序包含的子命令可用于获取代理信息、更新代理属性。表 6-4 列出了这些子命令。

表 6-4 用于获取代理信息和更新代理的 imqcmd 子命令

子命令语法	说明
`query bkr -b` *hostName:port*	列出默认代理或指定主机和端口上的代理的当前属性设置。还将列出与指定代理连接且正在运行的代理（在多代理群集中）。
`reload cls`	仅适用于代理群集。强制群集中的所有代理重新装入 `imq.cluster.brokerlist` 属性并更新群集信息。有关详细信息，请参阅第 120 页上的“将代理添加到群集”。

`update bkr [-b` *hostName:port*`]`
`-o` *attribute=value*
`[-o` *attribute=value1*`] . . .`
	更改默认代理或指定主机和端口上的代理的指定属性。

`metrics bkr [-b` *hostName:port*`]`
`[-m` *metricType*`]`
`[-int` *interval*`]`
	显示默认代理或指定主机和端口上的代理的代理标准。
	使用 -m 选项指定要显示的标准类型：
	`ttl` 流入和流出代理的所有消息（默认）。
	`rts` 与 ttl 提供相同的信息，但指定了每秒钟的消息数量。
	`cxn` 连接、虚拟内存堆、线程
	使用 -int 选项指定显示标准的时间间隔（以秒为单位）。默认值为 5 秒。

请记住，使用表 6-4 中列出的子命令时，除非您希望代理在 `localhost` 7676 端口上运行，否则必须指定代理主机名和端口号。

查询代理

要查询并显示某个代理的信息，请使用 query 子命令。例如：

```
imqcmd query bkr
```

此语句的输出如下：

```
%imqcmd query bkr

Querying the broker specified by:

-------------------------
Host         Primary Port
-------------------------
localhost    7676

Auto Create Queues                      true
Auto Create Topics                      true
Auto Created Queue Delivery Policy      Round Robin
Cluster Broker List (active)            myhost/192.18.116.221:7676
Cluster Broker List (configured)
Cluster Master Broker
Cluster URL
Current Number of Messages in System    0
Current Size of Messages in System      0
Instance Name                           imqbroker
Log Level                               INFO
Log Rollover Interval (seconds)         604800
Log Rollover Size (bytes)               0 (unlimited)
Max Message Size                        70m
Max Number of Messages in System        0 (unlimited)
Max Size of Messages in System          0 (unlimited)
Primary Port                            7676
Version                                 3.0

Successfully queried the broker.
```

更新代理

可以使用 update 子命令更新表 6-5 中列出的任何代理属性。请注意，对代理的更新会自动写入代理的实例配置文件。

表 6-5 代理属性

属性	说明
imq.autocreate.queue	指定是否允许代理自动创建队列目标。默认值为 True。
imq.autocreate.topic	指定是否允许代理自动创建主题目标。默认值为 True。
imq.queue.deliverypolicy	为自动创建的队列指定默认传送策略。有效值包括：s（单向）、r（循环）或 f（故障切换）。默认值为 s。
imq.cluster.url	指定群集配置文件的位置。有关详细信息，请参阅第 118 页上的 “群集配置属性”。
imq.log.level	指定下列日志级别之一：NONE、ERROR、WARNING 或 INFO。默认值为 INFO。
imq.log.file.rolloversecs	日志文件转移前的生存期（以秒为单位）。0 表示不发生基于文件生存期的转移。默认值为 604800 秒（7 天）。
imq.log.file.rolloverbytes	指定日志文件转移前的最大文件大小。0 表示不发生基于文件大小的转移。默认值为 0。
imq.message.max_size	指定消息的最大大小（以字节为单位）。默认值为 70m。
imq.system.max_count	指定内存和磁盘中的最大消息数量。0 表示没有限制。默认值为 0。
imq.system.max_size	指定内存和磁盘中所有消息总大小的最大值。0 表示没有限制。默认值为 0。
imq.portmapper.port	指定端口映射器的端口号。默认值为 7676。

例如，以下命令会将队列的默认传送策略由单向改为循环。

```
imqcmd update bkr -o "imq.queue.deliverypolicy=r"
```

管理连接服务

命令行实用程序包括许多子命令，您可以使用这些子命令执行以下操作：

- 列出可用的连接服务
- 显示某项特定服务的信息
- 更新某项服务的属性
- 暂停和恢复服务

有关 MQ 连接服务的概述信息，请参阅第 46 页上的 “连接服务”。

表 6-6 列出了控制连接服务的 `imqcmd` 子命令。如果没有指定主机名和端口，则假定为 `localhost` 和 `7676`。

表 6-6 用于管理连接服务的 `imqcmd` 子命令

子命令语法	说明
`list svc [-b` *hostName:port*`]`	列出默认代理或指定主机和端口上的代理中的所有连接服务。

`metrics svc -n` *serviceName*
`[-b` *hostName:port*`]`
`[-m` *metricType*`] [-int` *interval*`]`
	列出默认代理或指定主机和端口上的代理中指定服务的标准。
	使用 -m 选项指定要显示的标准类型：
	`ttl` 流入和流出代理的所有消息 （默认）。
	`rts` 与 ttl 提供相同的信息，但指定了每秒钟的消息数量。
	`cxn` 连接、虚拟内存堆、线程
	使用 -int 选项指定显示标准的时间间隔 （以秒为单位）。默认值为 5 秒。

`query svc -n` *serviceName* `[-b` *hostName:port*`]`	显示运行在默认代理或指定主机和端口上的代理中的指定服务的信息。
`pause svc -n` *serviceName* `[-b` *hostName:port*`]`	暂停运行在默认代理或指定主机和端口上的代理中的指定服务。不能暂停管理服务。
`resume svc -n` *serviceName* `[-b` *hostName:port*`]`	恢复运行在默认代理或指定主机和端口上的代理中的指定服务。

`update svc -n` *serviceName*
`[-b` *hostName:port*`]`
`-o` *attribute=value*
`[-o` *attribute=value1*`] . . .`
	更新运行在默认代理或指定主机和端口上的代理中的指定服务的指定属性。有关服务属性的说明，请参阅第 142 页上的表 6-8。

代理支持与应用程序客户机和管理客户机的通信。第 140 页上的表 6-7 显示了 MQ 代理当前支持的连接服务。“服务名称”一栏中的值用于为 -n 选项指定服务名称。（正如该表所示，每项服务都是通过它使用的服务类型 — NORMAL [JMS] 或 ADMIN — 和底层传输层来指定的。）

表 6-7 代理所支持的连接服务

服务名称	服务类型	协议类型
jms	NORMAL （JMS 消息传送）	tcp
ssljms （企业版）	NORMAL （JMS 消息传送）	tls （基于 SSL 的安全性）
httpjms （企业版）	NORMAL （JMS 消息传送）	http
httpsjms （企业版）	NORMAL （JMS 消息传送）	https （基于 SSL 的安全性）
admin	ADMIN	tcp
ssladmin （企业版）	ADMIN	tls （基于 SSL 的安全性）

列出连接服务

要列出某个代理支持的连接服务，请使用如下命令：

```
imqcmd list svc [-b 主机名 : 端口号 ]
```

例如，以下命令将列出运行在 MyServer 主机 6565 端口上的代理所支持的服务。

```
imqcmd list svc -b MyServer:6565
```

以下命令将列出运行在 localhost 7676 端口上的代理支持的所有服务：

```
imqcmd list svc
```

该命令的输出信息如下：

```
Listing all the services on the broker specified by:

Host                  Primary Port
localhost             7676

Service Name      Port Number        Service State
admin             33984 (dynamic)    RUNNING
httpjms                              UNKNOWN
httpsjms          -                  UNKNOWN
jms               33983 (dynamic)    RUNNING
ssladmin          dynamic            UNKNOWN
ssljms            dynamic            UNKNOWN

Successfully listed services.
```

查询和更新服务属性

要查询并显示某项服务的信息，请使用 query 子命令。例如：

```
imqcmd query svc -n jms
```

此语句的输出如下：

```
Querying the service where:

Service Name
jms

On the broker specified by:

Host                          Primary Port
localhost                     7676


Current Number of Allocated Threads    120
Current Number of Connections          20
Max Number of Threads                  1000
Min Number of Threads                  50
Port Number                            42019 (dynamic)
Service Name                           jms
Service State                          RUNNING

Successfully queried the service.
```

可以使用 update 子命令更改表 6-8 中列出的一个或多个服务属性的值。

表 6-8 连接服务属性

属性	说明
port	为要更新的服务指定的端口（不适用于 httpjms 和 httpsjms）。
minThreads	为服务指定的最小线程数。
maxThreads	为服务指定的最大线程数。

以下命令将指定给 jms 服务的最小线程数更改为 20。

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

暂停和恢复服务

要暂停除管理服务之外的其它服务 （不能暂停管理服务），请使用如下命令：

```
imqcmd pause svc -n serviceName
```

要恢复某项服务，请使用如下命令：

```
imqcmd resume svc -n serviceName
```

管理目标

所有 MQ 消息都是通过在特定代理上创建的目标、队列和主题路由到其使用方客户机上的。您负责在代理上管理这些目标。管理任务包括使用命令行实用程序创建和销毁目标、列出目标、显示目标的相关信息以及清理消息。有关目标的介绍，请参阅第 62 页上的 “物理目标”。

表 6-9 概述了 `imqcmd` 目标子命令。如果不使用默认代理 (`localhost:7676`)，则必须指定主机名和端口。

表 6-9 用于管理目标的 `imqcmd` 子命令

子命令	说明
`list dst [-tmp]`	列出所有目标，后面的选项表示同时列出临时目标（请参阅第 64 页上的 “临时目标”）。

`create dst -t` *type*
`-n` *destName*
`[-o` *attribute=value*`]`
`[-o` *attribute=value1*`] . . .`
	创建指定类型、名称和属性的目标。目标名称只能包含字母数字字符 （不能包含空格），可以以字母字符或者字符 “_” 开头。

`destroy dst -t` *type* `-n` *destName*	销毁指定类型和名称的目标。
`purge dst -t` *type* `-n` *destName*	清理指定类型和名称的目标中的消息。
`query dst -t` *type* `-n` *destName*	列出指定类型和名称的目标的相关信息。
`update dst -t` *type* `-n` *destName* `-o` *attribute=value* `[-o` *attribute=value1*`] . . .`	更新指定目标上指定属性的值。 属性名称可以是表 6-10 中列出的任意属性。

创建目标

创建目标时，必须指定目标类型 （主题或队列），如果需要，还要指定目标的属性值。这些属性的默认值是在代理的配置文件中设置的 （请参阅第 108 页上的 “配置文件”）。

销毁目标将清理该目标中的所有消息并将该目标从代理上删除，此操作是不可恢复的。

表 6-10 列出了创建目标时，可以为每种目标类型设置的属性。

表 6-10 目标属性

目标类型	属性	默认值	说明
队列	queueDeliveryPolicy	单向	表示用于路由消息的算法。 有效值包括 f = 故障切换 r = 循环 s = 单向
队列	maxTotalMsgBytes	0 （无限制）	队列中允许的消息总大小的最大值 （以字节为单位）。
队列	maxNumMsgs	0 （无限制）	队列中允许的最大消息数量。
队列	maxBytesPerMsg	0 （无限制）	队列中允许的一条消息的最大大小。
主题	maxBytesPerMsg	0 （无限制）	发布到主题中的一条消息的最大大小。

- 要创建队列目标，请输入如下命令：

  ```
  imqcmd create dst -n myQueue -t q -o "queueDeliveryPolicy=f"
  ```

  请注意，目标名称必须是有效的 Java 标识符。

- 要创建主题目标，请输入如下命令：

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

获得目标的相关信息

要获得某个目标属性当前值的信息，请使用如下命令：

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

然后可以使用 `update imqcmd` 子命令更改一个或多个属性的值。

要列出特定代理 （例如，运行在 `myHost` 4545 端口上的代理）中的所有目标，请使用如下命令：

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

使用 `list` 命令时可以选择是否列出临时目标 （使用 `-tmp` 选项）。临时目标是由客户机应用程序创建的，该应用程序需要一个目标来接收发送到其它客户机上的消息的回复 （请参阅第 64 页上的 “临时目标”）。您不能销毁这些目标；只有当不再存在活动的消息使用方时，才能通过客户机应用程序的 API 调用来销毁。

更新目标

可以使用 `update dst` 子命令更改目标的属性，而使用 `-o` 选项指定要更新的属性。如果要更新多个属性，可以多次使用 `-o` 选项。例如，以下命令将 `maxBytesPerMsg` 属性更改为 `1000`，同时将 `MaxNumMsgs` 属性更改为 `2000`：

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

要获得可以更新的属性列表，请参阅第 144 页上的表 6-10。

不能使用 `update dst` 子命令更新目标的 *type* 属性或队列的队列传送策略。

清理目标

可以清理某个目标上当前排队的所有消息。清理目标意味着物理目标上排队的所有消息都将被删除。当目标中堆积的消息占用了大量系统资源时，您可能需要清理掉这些消息。这可能发生在某个队列没有注册使用方客户机，但仍在接收大量消息的情况下。也可能发生在某个主题的长期订户始终处于非活动状态的情况下。在上述两种情况下，都没有必要保留消息。

要清理目标上的消息，请输入如下命令：

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

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

如果关闭代理后不希望重新启动时传送旧消息，可以使用 imqbrokerd 命令的 reset 子命令将这些旧消息清理掉，如下所示：

```
imqbrokerd -reset messages
```

这样可以避免重新启动代理后清理目标的麻烦。

销毁目标

要销毁目标，请输入如下命令：

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

管理长期订阅

要管理代理的长期订阅，可以使用 imqcmd 子命令。**长期订阅**是指某台客户机长期注册订阅某个主题；长期订阅有一个唯一标识，要求代理保留该订阅的消息，即使消息使用方处于非活动状态。通常情况下，代理只能在消息已过期的情况下删除为长期订户保留的消息。

表 6-12 概述了 imqcmd 长期订阅子命令。如果不使用默认代理 (localhost:7676)，则必须指定主机名和端口。

表 6-11 用于管理长期订阅的 imqcmd 子命令

子命令	说明
`list dur -d` *destination*	列出指定目标的所有长期订阅。
`destroy dur -n` *subscrName* `-c` *client_id*	销毁指定客户机 ID 的指定长期订阅（请参阅第 38 页上的 “客户机标识符”）。
`purge dur -n` *subscrName* `-c` *client_id*	清理指定客户机 ID 的所有消息（请参阅第 38 页上的 “客户机标识符”）。

例如，以下命令将列出 SPQuotes 主题下的所有长期订阅：

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

对于某个主题下的每个长期订阅，`list` 子命令将返回长期订阅的名称、用户的客户机 ID、该主题下排队的消息数量以及该长期订阅的状态（活动/非活动）。例如：

```
Listing all the durable subscriptions on the topic myTopic
on the broker specified by:
-------------------------
Host          Primary Port
-------------------------
localhost     7676

Name         Client ID       Number of    Durable Sub
                             Messages       State
------------------------------------------------------------------
myDurable    myClientID       1            INACTIVE

Successfully listed durable subscriptions.
```

可以使用 `list` 命令返回的信息标识希望销毁或清理其消息的长期订阅。使用订阅名称和客户机 ID 可以标识订阅。例如：

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

管理事务

客户机应用程序启动的所有事务都由代理进行跟踪。这些事务可以是 XA 资源管理器管理的简单 MQ 事务或分布式事务（请参阅第 39 页上的 “本地事务”）。所有事务都有一个 MQ 事务 ID，这是一个 64 位的数字，唯一标识代理上的每个事务。分布式事务也有一个分布式事务 ID (XID)，长度为 128 字节，由分布式事务管理器指定。 MQ 负责维护 MQ 事务 ID 与 XID 之间的关联。

对于分布式事务来说，失败的事务可能会保持 `PREPARED` 状态，而不会提交。因此，管理员需要监视并在需要时回滚或提交那些处于 PREPARED 状态的事务。

表 6-12 概述了 imqcmd 事务子命令。如果不使用默认代理 (localhost:7676)，则必须指定主机名和端口。

表 6-12 用于管理事务的 imqcmd 子命令

子命令	说明
`list txn`	列出代理跟踪的所有事务。
`query txn -n` *transaction_id*	列出指定事务的相关信息。
`commit txn -n` *transaction_id*	提交指定的事务。
`rollback txn -n` *transaction_id*	回滚指定的事务。

例如，以下命令将列出某个代理中的所有事务。

```
imqcmd list txn
```

对于每个事务， list 子命令将返回事务 ID、状态、用户名、消息或确认的数量以及创建时间。例如：

```
Listing all the transactions on the broker specified by:
-------------------------
Host          Primary Port
-------------------------
localhost     7676

------------------------------------------------------------------
Transaction ID  State     User name    # Msgs/    Creation time
                                       # Acks
------------------------------------------------------------------

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

Successfully listed transactions.
```

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

例如，如果代理的 auto-rollback 属性被设置为 false （请参阅第 52 页上的表 2-4），那么必须在启动代理时手动提交或回滚发现处于 PREPARED 状态的事务。

list 子命令还显示事务中生成和确认的消息数量 (#Msgs/#Acks)。提交事务之前不会传送消息，也不会处理确认。

query 子命令也可以显示这些信息，除此之外，它还显示以下信息：客户机 ID、连接标识和分布式事务 ID (XID)。例如：

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

此语句的输出如下：

```
Querying the transaction where:
-------------------------
Transaction ID
-------------------------
64248349708800

On the broker specified by:

-------------------------
Host         Primary Port
-------------------------
localhost    7676


Client ID
Connection                 guest@192.18.116.219:62209->jms:62195
Creation time              1/30/02 10:08:31 AM
Number of acknowledgements 0
Number of messages         4
State                      PREPARED
Transaction ID             64248349708800
User name                  guest
XID
6469706F6C7369646577696E6465723130313234313431313030373230

Successfully queried the transaction.
```

commit 和 rollback 子命令可用于提交和回滚分布式事务。正如前文所述，只能提交或回滚处于 PREPARED 状态的事务。例如：

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

您也可以配置代理，使它在启动时自动回滚处于 PREPARED 状态的事务。有关详细信息，请参阅第 52 页上的表 2-4 中的 imq.transaction.autorollback 属性。

第 7 章

管理被管理对象

使用被管理对象可以开发供其它 JMS 提供者使用的客户机应用程序。**被管理对象**是指包含提供者特定的配置信息和命名信息的对象。这些对象通常由 Sun™ ONE Message Queue (MQ) 管理员创建，由客户机应用程序用来与代理建立连接，然后用于向物理目标发送消息并从物理目标接收消息。

有关被管理对象的概述信息，请参阅第 71 页上的 “MQ 被管理对象”。

MQ 为创建和管理被管理对象提供了两个管理工具：命令行对象管理器实用程序 (`imqobjmgr`) 和 GUI 管理控制台。您可以使用这两个工具执行以下操作：

- 在对象存储中添加或删除被管理对象。
- 列出现有的被管理对象。
- 查询和显示被管理对象的相关信息。
- 修改对象存储中现有的被管理对象。

本章说明如何使用对象管理器实用程序 (`imqobjmgr`) 执行上述任务。有关管理控制台的详细信息，请参阅第 4 章 “管理控制台教程”。

关于对象存储

被管理对象放置在便于访问的对象存储中，客户机应用程序可以通过 JNDI 查找访问这些对象。可以使用两种类型的对象存储：标准 LDAP 目录服务器和文件系统对象存储。

LDAP 服务器　对于生产型消息传送系统，建议使用 LDAP 服务器对象存储。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。许多 MQ 管理工具都可以与 LDAP 服务器一起使用。

文件系统存储　MQ 还支持文件系统对象存储实现。文件系统对象存储尚未进行完整测试，因此不建议用于生产系统。它的优势就是在开发环境中非常易于使用。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。任何可以访问这个目录的用户都可以使用 MQ 管理工具创建和管理被管理对象。

被管理对象

有关被管理对象的概述信息，请参阅第 71 页上的 “MQ 被管理对象”。

MQ 被管理对象有两种基本类型：连接工厂和目标。**连接工厂**被管理对象由客户机应用程序用来建立与代理之间的连接。**目标**被管理对象由客户机应用程序用来标识生成方向其发送消息或使用方从其检索消息的目标。（SOAP 消息传送使用特殊的 *SOAP* **端点**被管理对象，有关详细信息，请参阅 《MQ *Developer's Guide*》。）

可以根据消息传送模型 （点对点或者发布 / 订阅）使用特定类型的连接工厂和目标。例如，在点对点编程中，可以使用队列连接工厂和队列目标。而在发布和订阅编程中，可以使用主题连接工厂和主题目标。也可以使用非特定的连接工厂和目标被管理对象类型，只要它们属于支持分布式事务的连接工厂类型 （要查看所有支持的类型，请参阅第 37 页上的表 1-1）。

对象管理器实用程序 (imqobjmgr)

对象管理器实用程序允许您创建和管理 MQ 被管理对象。本节介绍了基本的 imqobjmgr 命令语法，提供了一个子命令列表，并概述了 imqobjmgr 命令选项。以下各节说明如何使用 imqobjmgr 子命令完成特定任务。

命令语法

imqcmd 命令的一般语法如下所示：

```
imqobjmgr subcommand [options]
imqobjmgr -h|H
imqobjmgr -v
```

请注意，如果指定 -v、-h 或 -H 选项，将不会执行命令行中指定的子命令。例如，输入以下命令将显示版本信息，而不是执行 list 子命令。

```
imqobjmgr list -v
```

imqobjmgr 子命令

表 7-1 列出了对象管理器实用程序 (imqobjmgr) 包含的子命令：

表 7-1 imqobjmgr 子命令

子命令	说明
add	在对象存储中添加被管理对象。
delete	从对象存储中删除被管理对象。
list	列出对象存储中的被管理对象。
query	显示特定被管理对象的相关信息。
update	修改对象存储中的现有被管理对象。

imqobjmgr 命令选项概述

表 7-2 列出了 `imqobjmgr` 命令的选项。有关如何使用这些选项的讨论，请参阅基于任务的后续各节。

表 7-2 `imqobjmgr` 选项

选项	说明
`-f`	执行操作，无需用户确认。
`-h`	显示使用帮助。不执行命令行上的命令。
`-H`	显示用法帮助、属性列表和实例。不执行命令行上的命令。
`-i` *fileName*	指定输入文件的名称，该文件包含所有或部分子命令子句，指定了对象类型、查找名称、对象属性、对象存储属性或其它选项。通常用于重复性信息，例如对象存储属性。
`-j` *attribute=value*	指定标识和访问 JNDI 对象存储所需的属性。
`-javahome`	指定可使用的替代 Java 2 兼容运行时 （默认使用产品附带的运行时）。
`-l` *lookupName*	指定被管理对象的 JNDI 查找名称。此名称在对象存储的上下文中必须唯一。
`-o` *attribute=value*	指定被管理对象的属性。
`-pre`	预览模式。指明不执行命令的情况下执行什么操作。
`-r` *read-only_state*	指定某个被管理对象是否为只读对象。 `true` 值表示被管理对象为只读对象。 JMS 客户不能修改只读被管理对象的属性。 read-only_state 的默认值为 `false`。
`-s`	无提示模式。不显示任何输出。

表 7-2 imqobjmgr 选项 （续）

选项	说明
-t *type*	指定 MQ 被管理对象的类型： q = 队列 t = 主题 cf = 连接工厂 qf = 队列连接工厂 tf = 主题连接工厂 xcf = XA 连接工厂 （分布式事务） xqf = XA 队列连接工厂 （分布式事务） xtf = XA 主题连接工厂 （分布式事务） e = SOAP 端点[1]
-v	显示版本信息。不执行命令行上的命令。

1 此被管理对象类型用于支持 SOAP 消息 （请参阅 《MQ *Developer's Guide*》）。

下一节说明使用 imqobjmgr 子命令时需要提供的信息。

所需的信息

执行大多数与被管理对象相关的任务时，管理员必须指定以下信息作为 imqobjmgr 子命令的选项：

- 被管理对象的类型：

  表 7-2 列出了允许的类型。

- 被管理对象的 JNDI 查找名称：

  这是在客户机代码中表示对象存储中的被管理对象 （使用 JNDI）时使用的逻辑名。

- 被管理对象的属性：
  - 队列和主题：代理上的物理目标的名称。这是使用 `imqcmd create` 子命令的 `-n` 选项指定的名称。如果不指定名称，将使用默认名称 `Untitled_Destination_Object`。
  - 连接工厂：客户机将连接的代理的主机名和端口号。如果不指定这些信息，将使用本地主机和默认端口号 (7676)。第 156 页上的 “被管理对象的属性”一节说明了如何指定对象属性。

  有关其它属性的详细信息，请参阅第 156 页上的 “被管理对象的属性”。
- JNDI 对象存储的属性：

  此信息取决于您使用的是文件系统存储还是 LDAP 服务器，但必须包括以下属性：
  - JNDI 实现的类型 （初始上下文属性）。例如，文件系统或 LDAP。
  - 被管理对象在对象存储中的位置 （提供者 URL 属性），即它所在的 “文件夹”。
  - 访问对象存储所需的用户名、密码和授权类型 （如果有）。

  有关对象存储属性的详细信息，请参阅第 159 页上的 “对象存储属性”。

被管理对象的属性

被管理对象的属性是使用属性 - 值对指定的。下面将介绍这些属性。

连接工厂被管理对象

表 7-3 列出了连接工厂 （和 XA 连接工厂）被管理对象的属性。最重要的两个属性为 `imqBrokerHostPort` 和 `imqBrokerHostName`，它们用于指定客户机应用程序将要与之建立连接的代理。第 163 页上的 “添加连接工厂”一节说明了如何在向对象存储中添加连接工厂被管理对象时指定这些属性。

有关连接工厂属性的说明以及如何使用它们的详细信息，请参阅 《MQ *Developer's Guide*》和适用于 MQ 类 `com.sun.messaging.ConnectionConfiguration` 的 JavaDoc API 文档。

表 7-3 连接工厂属性

属性/属性名	类型	默认值
imqAckOnAcknowledge	字符串	未指定
imqAckOnProduce	字符串	未指定
imqAckTimeout	字符串	0 毫秒
imqBrokerHostName	字符串	本地主机
imqBrokerHostPort	字符串	7676
imqBrokerServicePort	字符串	0
imqConfiguredClientID	字符串	未指定
imqConnectionType	字符串	TCP
imqConnectionURL	字符串	http://localhost/imq/tunnel
imqDefaultPassword	字符串	临时用户
imqDefaultUsername	字符串	临时用户
imqDisableSetClientID	字符串	false
imqFlowControlCount	字符串	100
imqFlowControlIsLimited	字符串	false
imqFlowControlLimit	字符串	1000
imqLoadMaxToServerSession	字符串	false
imqQueueBrowserMaxMessagesPerRetrieve	字符串	1000
imqQueueBrowserRetrieveTimeout	字符串	60,000 毫秒
imqReconnect	布尔值	false
imqReconnectDelay	字符串	30,000 毫秒
imqReconnectRetries	字符串	0
imqSetJMSXAppID	字符串	false
imqSetJMSXConsumerTXID	字符串	false
imqSetJMSXProducerTXID	字符串	false
imqSetJMSXRcvTimestamp	字符串	false
imqSetJMSXUserID	字符串	false

表 7-3 连接工厂属性 （续）

属性/属性名	类型	默认值
imqSSLIsHostTrusted	字符串	true
imqJMSDeliveryMode	整数	2 （永久）
imqJMSExpiration	整数	0 （不过期）
imqJMSPriority	整数	4 （一般优先级）
imqOverrideJMSDeliveryMode	布尔值	false
imqOverrideJMSExpiration	布尔值	false
imqOverrideJMSPriority	布尔值	false
imqOverrideJMSHeadersTo TemporaryDestinations	布尔值	false

目标被管理对象

表 7-4 列出了标识物理主题或队列目标的目标被管理对象的属性。第 164 页上的“添加主题或队列”一节说明了如何在向对象存储中添加目标被管理对象时指定这些属性。

最重要的属性是 imqDestinationName。这是您为与主题或队列被管理对象相对应的物理目标指定的名称。您也可以为目标提供说明，帮助您将它与您创建以支持更多应用程序的其它目标区分开。

有关详细信息，请参阅适用于 MQ 类 com.sun.messaging.DestinationConfiguration 的 JavaDoc API 文档。

表 7-4 目标属性

属性/属性名	类型	默认值
imqDestinationDescription	字符串	目标对象的说明
imqDestinationName	字符串[1]	Untitled_Destination_Object

1 目标名称只能包含字母数字字符 （不能包含空格），且必须以字母数字字符或者字符 “_” 和/或 “$” 开头。

对象存储属性

对象存储的属性是使用 -j 选项指定的，由属性 - 值对组成。通常情况下，必须指定以下属性：

初始上下文和地址信息

这些条目的格式不尽相同，取决于您使用的是文件系统存储还是 LDAP 服务器。

文件系统存储 作为使用文件系统存储的实例，我们在 C 盘驱动器上创建一个名为 *MyObjstore* 的文件夹，并分别为初始上下文和地址属性指定以下值：

```
-j "java.naming.factory.initial=
          com.sun.jndi.fscontext.RefFSContextFactory"

-j "java.naming.provider.url=file:///C:/MyObjStore"
```

LDAP 服务器 作为使用 LDAP 服务器的实例，我们分别为初始上下文和地址属性指定以下值：

```
-j "java.naming.factory.initial=
          com.sun.jndi.ldap.LdapCtxFactory"

-j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
```

安全信息 （仅用于 LDAP）

这些条目的格式因 LDAP 提供者而异。同时还请参阅 LDAP 实现附带的文档，以决定是所有的操作都需要安全信息，还是只有更改存储数据的操作需要安全信息。

安全属性如下所示：

```
-j "java.naming.security.principal=
      uid=fooUser, ou=People, o=imq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

表 7-5 说明了这些条目：

表 7-5 对象存储的安全属性

属性	说明
...principal	负责验证服务呼叫者的人员的身份。此条目的格式取决于验证方案。如果不指定此属性，性能将由服务提供者决定。
...credentials	负责验证服务呼叫者的人员的凭证。此属性的值取决于验证方案。例如，它可能是散列密码、纯文本密码、密钥、证书等。如果不指定此属性，性能将由服务提供者决定。
...authentication	使用的安全级别。其值是以下关键字之一：none、 simple 或 strong。如果不指定此属性，性能将由服务提供者决定。 如果指定 simple， imqobjmgr 将提示缺少 principal 值或 credential 值。这使您可以一种更安全的方式提供身份信息。

使用输入文件

imqobjmgr 命令允许您指定输入文件的名称，该文件使用 Java 属性文件语法表示全部或部分 imqobjmgr 子命令字句。

在对象管理器实用程序 (imqobjmgr) 中使用输入文件对于指定对象存储属性特别有用，因为对象存储属性可能在 imqobjmgr 的多个调用中都相同，这通常需要多次键入。使用输入文件还可以避免一种情况，即避免超出命令行所允许的最大字符数量。

下面显示了 imqobjmgr 输入文件的一般语法 （version 属性只适用于输入文件，它不是命令行选项，而且它的值必须设置为 2.0）：

```
version=2.0
cmdtype=[ add | delete | list | query | update ]
obj.type=[ q | t | qf | tf | cf | xqf | xtf | xcf | e ]
obj.lookupName=lookup name
obj.attrs.objAttrName1=value1
obj.attrs.objAttrName2=value2
obj.attrs.objAttrNameN=valueN
...
objstore.attrs.objStoreAttrName1=value1
objstore.attrs.objStoreAttrName2=value2
objstore.attrs.objStoreAttrNameN=valueN
...
```

作为使用输入文件的一个实例，请考虑以下 imqobjmgr 命令：

```
imqobjmgr add
       -t qf
       -l "cn=myQCF"
       -o "imqBrokerHostName=foo"
       -o "imqBrokerHostPort=777"
       -j "java.naming.factory.initial=
                com.sun.jndi.ldap.LdapCtxFactory"
       -j "java.naming.provider.url=
                ldap://mydomain.com:389/o=imq"
       -j "java.naming.security.principal=
                uid=fooUser, ou=People, o=imq"
       -j "java.naming.security.credentials=fooPasswd"
       -j "java.naming.security.authentication=simple"
```

此命令可以包含在一个文件中 （例如 MyCmdFile 文件），其内容如下：

```
    version=2.0
    cmdtype=add
    obj.type=qf
    obj.lookupName=cn=myQCF
    obj.attrs.imqBrokerHostName=foo
    obj.attrs.imqBrokerHostPort=777
    objstore.attrs.java.naming.factory.initial=\
              com.sun.jndi.ldap.LdapCtxFactory
    objstore.attrs.java.naming.provider.url=\
              ldap://mydomain.com:389/o=imq
    objstore.attrs.java.naming.security.principal=\
              uid=fooUser, ou=People, o=imq
    objstore.attrs.java.naming.security.credentials=fooPasswd
    objstore.attrs.java.naming.security.authentication=simple
```

然后可以使用 -i 选项将该文件传递给对象管理器实用程序 (imqobjmgr)：

```
imqobjmgr -i MyCmdFile
```

您也可以使用输入文件指定一些选项，而使用命令行指定其它选项。这样即可使用输入文件指定在实用程序的多次调用中都相同的那部分子命令子句。例如，以下命令可以指定添加连接工厂被管理对象所需的所有选项，但不包括指定被管理对象存储位置的选项。

```
imqobjmgr add
        -t qf
        -l "cn=myQCF"
        -o "imqBrokerHostName=foo"
        -o "imqBrokerHostPort=777"
        -i MyCmdFile
```

在本例中，文件 MyCmdFile 将包含以下定义：

```
    version=2.0
    objstore.attrs.java.naming.factory.initial=\
              com.sun.jndi.ldap.LdapCtxFactory
    objstore.attrs.java.naming.provider.url=\
              ldap://mydomain.com:389/o=imq
    objstore.attrs.java.naming.security.principal=\
              uid=fooUser, ou=People, o=imq
    objstore.attrs.java.naming.security.credentials=fooPasswd
    objstore.attrs.java.naming.security.authentication=simple
```

可以从以下位置找到其它输入文件的例子：

```
IMQ_HOME/demo/imqobjmgr
```

添加和删除被管理对象

本节说明如何在对象存储中添加连接工厂和主题或队列目标被管理对象。

注 对象管理器实用程序 (imqobjmgr) 只列出和显示了 MQ 被管理对象。如果对象存储中包含的一个非 MQ 对象与您要添加的被管理对象具有相同的查找名称，试图执行添加操作时将收到错误消息。

添加连接工厂

要使客户机应用程序获得与代理的连接，需要添加一个用来表示客户机应用程序所需连接类型的被管理对象：主题连接工厂或队列连接工厂

要添加队列连接工厂，请使用如下所示的命令：

```
imqobjmgr add
  -t qf
  -l "cn=myQCF"
  -o "imqBrokerHostName=myHost"
  -o "imqBrokerHostPort=7272"
  -j "java.naming.factoryinitial=
              com.sun.jndi.ldap.LdapCtxFactory"
  -j "java.naming.provider.url=ldap://mydomain.com:389/o=imq"
  -j "java.naming.security.principal=
        uid=fooUser, ou=People, o=imq"
  -j "java.naming.security.credentials=fooPasswd"
  -j "java.naming.security.authentication=simple"
```

上述命令创建一个查找名称为 cn=myQCF 的被管理对象，它连接到在 myHost 上运行的代理并在端口 7272 上监听。被管理对象存储在 LDAP 服务器上。

注 如果使用 LDAP 服务器存储被管理对象，必须像上例那样指定一个前缀为 “cn=” 的查找名称。可以通过 -l 选项指定查找名称。如果使用文件系统对象存储，则无需使用该前缀。

通过指定一个输入文件作为 imqobjmgr 命令的参数，也可以实现此目的。有关详细信息，请参阅第 160 页上的 “使用输入文件”。

添加主题或队列

要使客户机应用程序能够访问代理上的物理目标，可以将标识这些目标的被管理对象添加到对象存储中。

最好先创建物理目标，然后再将对应的被管理对象添加到对象存储中。使用命令行实用程序 (imqcmd) 在代理上创建物理目标，这个代理由对象存储中的目标被管理对象标识。有关创建物理目标的详细信息，请参阅第 143 页上的 “管理目标”。

以下命令添加了一个标识主题目标的被管理对象，这个主题目标的查找名称为 myTopic，物理目标名称为 TestTopic。被管理对象存储在 LDAP 服务器上。

```
imqobjmgr add
-t t
-l "cn=myTopic"
-o "imqDestinationName=TestTopic"
-j "java.naming.factory.initial=
        com.sun.jndi.ldap.LdapCtxFactory"
-j "java.naming.provider.url=
        ldap://mydomain.com:389/o=imq"
-j "java.naming.security.principal=
        uid=fooUser, ou=People, o=imq"
-j "java.naming.security.credentials=fooPasswd"
-j "java.naming.security.authentication=simple"
```

这是同一个命令，只不过被管理对象存储在 Solaris 文件系统中：

```
imqobjmgr add
      -t t
      -l "cn=myTopic"
      -o "imqDestinationName=TestTopic"
      -j "java.naming.factory.initial=
              com.sun.jndi.fscontext.RefFSContextFactory"
      -j "java.naming.provider.url=
              file:///home/foo/imq_admin_objects"
```

以 LDAP 服务器为例，您可以使用输入文件 MyCmdFile 指定子命令子句。该文件将包含以下内容：

```
version=2.0
cmdtype=add
obj.type=t
obj.lookupName=cn=myTopic
obj.attrs.imqDestinationName=TestTopic
objstore.attrs.java.naming.factory.initial=
        com.sun.jndi.fscontext.RefFSContextFactory
objstore.attrs.java.naming.provider.url=
        file:///home/foo/imq_admin_objects
objstore.attrs.java.naming.security.principal=
        uid=fooUser, ou=People, o=imq
objstore.attrs.java.naming.security.credentials=fooPasswd
objstore.attrs.java.naming.security.authentication=simple
```

使用 -i 选项将该文件传递给 imqobjmgr 命令：

```
imqobjmgr -i MyCmdFile
```

注 如果使用 LDAP 服务器存储被管理对象，必须像上例那样指定一个前缀为 “cn=”的查找名称。可以通过 -l 选项指定查找名称。如果使用文件系统对象存储，则无需使用该前缀。

添加队列对象与此基本相同，不同之处在于要为 -t 选项指定 q。

删除被管理对象

可以使用 delete 子命令删除被管理对象。必须指定对象的查找名称、类型和地址。

以下命令删除了查找名称为 cn=myTopic 且存储在 LDAP 服务器上的主题的被管理对象。

```
imqobjmgr delete
     -t t
     -l "cn=myTopic"
     -j "java.naming.factory.initial=
            com.sun.jndi.ldap.LdapCtxFactory"
     -j "java.naming.provider.url=
            ldap://mydomain.com:389/o=imq"
     -j "java.naming.security.principal=
            uid=fooUser, ou=People, o=imq"
     -j "java.naming.security.credentials=fooPasswd"
     -j "java.naming.security.authentication=simple"
```

获得信息

可以使用 list 和 query 子命令列出对象存储中的被管理对象，显示单个对象的相关信息。

被管理对象列表

使用 list 子命令可以获得所有被管理对象的列表，或者某个特定类型的被管理对象的列表。以下样例代码假定被管理对象存储在 LDAP 服务器上。

以下命令将列出所有对象。

```
imqobjmgr list
    -j "java.naming.factory.initial=
            com.sun.jndi.ldap.LdapCtxFactory"
    -j "java.naming.provider.url=
            ldap://mydomain.com:389/o=imq"
    -j "java.naming.security.principal=
            uid=fooUser, ou=People, o=imq"
    -j "java.naming.security.credentials=fooPasswd"
    -j "java.naming.security.authentication=simple"
```

以下命令将列出类型为 queue 的所有对象。

```
imqobjmgr list
  -t q
  -j "java.naming.factory.initial=
            com.sun.jndi.ldap.LdapCtxFactory"
  -j "java.naming.provider.url=
            ldap://mydomain.com:389/o=imq"
  -j "java.naming.security.principal=
            uid=fooUser, ou=People, o=imq"
  -j "java.naming.security.credentials=fooPasswd"
  -j "java.naming.security.authentication=simple"
```

单个对象的相关信息

使用 query 子命令可以获得单个被管理对象的相关信息。必须指定对象的查找名称，以及包含被管理对象的对象存储的属性 （例如，初始上下文和地址）。

在以下实例中，使用 query 子命令显示查找名称为 myTopic 的对象的相关信息。

```
imqobjmgr query
  -l "cn=myTopic"
  -j "java.naming.factory.initial=
          com.sun.jndi.ldap.LdapCtxFactory"
  -j "java.naming.provider.url=
          ldap://mydomain.com:389/o=imq"
  -j "java.naming.security.principal=
          uid=fooUser, ou=People, o=imq"
  -j "java.naming.security.credentials=fooPasswd"
  -j "java.naming.security.authentication=simple"
```

更新被管理对象

可以使用 update 命令修改被管理对象的属性。必须指定对象的查找名称和地址。可以使用 -o 选项修改属性值。

此命令将改变表示主题连接工厂的被管理对象的属性：

```
imqobjmgr update
 -t tf
 -l "cn=MyTCF"
 -o imqReconnectRetries=3
 -j "java.naming.factory.initial=
        com.sun.jndi.ldap.LdapCtxFactory"
 -j "java.naming.provider.url=
        ldap://mydomain.com:389/o=imq"
 -j "java.naming.security.principal=
        uid=fooUser, ou=People, o=imq"
 -j "java.naming.security.credentials=fooPasswd"
 -j "java.naming.security.authentication=simple"
```

第 8 章

安全性管理

本章介绍如何执行与安全性相关的任务，包含验证、授权和加密。

验证用户　您负责维护用户列表、用户组以及用户系统信息库中的密码。本章第一部分介绍如何创建、填充和管理系统信息库。有关 Sun™ ONE Message Queue (MQ) 安全性的介绍，请参阅第 55 页上的 “安全性管理器”。

授权用户　您负责编辑属性文件，此属性文件用于将用户对代理操作的访问权限映射到用户名或用户组。本章第二部分介绍如何自定义此属性文件。

加密：设置 SSL 服务　使用基于安全套接字层 (SSL) 标准的连接服务可以对客户机与代理之间传送的消息进行加密。有关 MQ 如何处理加密的介绍，请参阅第 57 页上的 “加密 （企业版）”。本章最后一部分介绍如何设置基于 SSL 的连接服务，并提供了与 SSL 有关的其它信息。

在代理需要提供密码才能安全访问 SSL 键存储、 LDAP 用户系统信息库或 JDBC 兼容持久性存储器的情况下，提供密码的方式有三种：

- 让系统在代理启动时提示您输入密码
- 启动代理时通过命令行选项提供密码 （请参阅第 114 页上的 “启动代理”和第 115 页上的表 5-2）
- 将密码存储在代理启动时可以访问的密码文件中 （请参阅第 189 页上的 “使用密码文件”）

验证用户

当某个用户试图与代理建立连接时，代理将通过检查该用户提供的用户名和密码对其进行验证。如果用户提供的用户名和密码与配置代理时指定要参照的用户系统信息库中的用户名和密码相匹配，则允许该用户连接到该代理。系统信息库有两种类型：

- MQ 出厂时附带的文本文件系统信息库

  这种用户系统信息库易于使用，但容易受到安全性攻击，因此仅用于测试和开发目的。您可以使用用户管理器实用程序 (`imqusermgr`) 填充和管理系统信息库。要启用验证，您需要用每个用户名、密码和用户组名称填充用户系统信息库。

  有关设置和管理用户系统信息库的详细信息，请参阅第 170 页上的 “使用文本文件用户系统信息库”。

- LDAP 服务器

  这可以是现有的或新的 LDAP 目录服务器，这种服务器上的用户系统信息库使用 LDAP v2 或 v3 协议。它并不象文本文件系统信息库那样易于使用，但它更安全，因此更适用于生产环境。

  如果您目前使用的是 LDAP 用户系统信息库，您需要使用 LDAP 供应商提供的工具来填充和管理该用户系统信息库。有关详细信息，请参阅第 176 页上的 “使用 LDAP 服务器管理用户系统信息库”。

使用文本文件用户系统信息库

MQ 提供了一个文本文件用户系统信息库，还提供了一个命令行工具 MQ 用户管理器 (`imqusermgr`)，您可以使用它填充和管理文本文件用户系统信息库。以下各节介绍文本文件系统信息库、信息库的初始条目以及如何填充和管理该信息库。

默认的文本文件系统信息库位于以下位置：

`IMQ_HOME/etc/passwd` （在 Solaris 操作系统上为 `/etc/imq/passwd`）

系统信息库出厂时附带了两个已定义的条目 （行），如下表所示。

表 8-1 用户系统信息库中的初始条目

用户名	密码	组	状态
admin	admin	admin	active
guest	guest	anonymous	active

这些初始条目使 MQ 代理安装后即可使用，无需任何管理员的介入。换句话说，使用 MQ 代理之前无需设置初始用户 / 密码。

初始 guest 用户条目使 JMS 客户机可以使用默认的 guest 用户名和密码连接到代理（例如，用于测试目的）。

初始 admin 用户条目使您可以通过 imqcmd 命令，使用默认的 admin 用户名和密码管理代理。建议您更新此初始条目以更改密码。

您可以使用用户管理器实用程序编辑或填充文本文件用户系统信息库，而不需要首先配置或启动代理。使用用户管理器实用程序的唯一要求是要在安装代理的主机上运行它，此外，如果您需要对系统信息库执行写入操作，则必须具有相应的权限：

- 在 Solaris 操作系统上，可以以 root 用户身份或者以其它通过基于 Solaris 角色访问控制获得访问权限的非 root 用户身份运行用户管理器实用程序。要获得此类访问权限，root 用户需要按照以下格式在 /etc/user_attr 中添加一个条目：

  username::::type=normal;profiles=Message Queue Management

  此操作将指定用户添加到 MQ 权限简要表中。要使用此设备，首先需要运行简要表 shell （pfsh、 pfksh 或 pfcsh），例如：

  ```
  % /usr/bin/pfsh
  ```

  然后执行所需的用户管理器 (imqusermgr) 命令。

  有关基于 Solaris 角色的访问控制的详细信息，请访问：
  http://docs.sun.com/
  ?q=Rights+Profile&p=/doc/806-4078/6jd6cjrvl&a=view。

- 在 Windows 操作系统上，安装后，任何用户都可以写入用户系统信息库文件，因为操作系统没有使用基于用户名权限属性控制对文件的访问。

以下各节说明如何填充和管理文本文件用户系统信息库。

用户管理器实用程序 (imqusermgr)

可以使用用户管理器实用程序管理基于文件的用户系统信息库。本节介绍了基本的 imqusermgr 命令语法，提供了一个子命令列表，并概述了 imqusermgr 命令选项。下文说明如何使用 imqobjmgr 子命令完成特定任务。

命令语法

imqusermgr 命令的一般语法如下：

```
imqusermgr subcommand [options]
imqusermgr -h
imqusermgr -v
```

请注意，如果指定 -v 或 -h 选项，将不会执行命令行中指定的子命令。例如，输入以下命令将显示版本信息，而不是执行 list 子命令。

```
imqusermgr list -v
```

imqusermgr 子命令

表 8-2 列出了 imqusermgr 子命令。

表 8-2 imqusermgr 子命令

子命令	说明
`add -u` *name* `-p` *passwd* `[-g` *group*`] [-s]`	将用户和关联的密码添加到系统信息库中，并有选择地指定用户所属的组。
`delete -u` *name* `[-s] [-f]`	从系统信息库中删除指定用户。
`list [-u` *name*`]`	显示指定用户或所有用户的相关信息。
`update -u` *name* `-p` *passwd* `[-a` *state*`] [-s] [-f]` `update -u` *name* `-a` *state* `[-p` *passwd*`] [-s] [-f]`	更新指定用户的密码和/或状态。

imqusermgr 命令选项概述

表 8-3 列出了 imqusermgr 命令的选项。

表 8-3 imqusermgr 选项

选项	说明
`-a` *active_state*	指定用户是否处于活动状态 (true/false)。true 表示处于活动状态。这是默认值。
`-f`	执行操作，无需用户确认。
`-h`	显示使用帮助。不执行命令行上的命令。
`-p` *passwd*	指定用户密码。
`-g` *group*	指定用户组。有效值包括 admin、 user 和 anonymous。
`-s`	设置无提示模式。
`-u` *name*	指定用户名。
`-v`	显示版本信息。不执行命令行上的命令。

组

在系统信息库中添加用户条目时，管理员可以为该用户指定三个预定义组中的一个组，这三个组为：admin、 user 或 anonymous。如果不指定任何组，则默认属于 user 组。

- *admin* 组用于代理管理员。默认情况下，指定到该组中的用户可以配置和管理代理。管理员可以为 admin 组指定多个用户。
- *user* 组用于一般 （非管理） JMS 客户机应用程序。多数 MQ 客户机应用程序将访问在 user 组中获得授权的代理。同样，默认情况下，客户机应用程序可以生成和使用所有主题和队列的消息，也可以浏览任意队列中的消息。
- *anonymous* 组用于那些不想使用代理知道的用户名 （也许应用程序不知道可以使用的实际用户名）的 JMS 客户机应用程序。这类似于多数 FTP 服务器中的匿名帐户。管理员一次只能为 anonymous 组指定一个用户。与 user 组相比，您需要通过访问控制限制此组的访问权限，或者在部署时从此组中删除这个用户。

要更改某个用户所属的组，管理员必须删除该用户的条目，然后再添加一个用户条目并为其指定一个新组。

您可以指定访问规则，定义组成员可以执行哪些操作。有关详细信息，请参阅第 178 页上的 “授权用户：访问控制属性文件”。

状态

当管理员将用户添加到系统信息库时，默认情况下用户处于活动状态。要使用户处于非活动状态，管理员必须使用 update 命令。例如，以下命令将使用户 JoeD 处于非活动状态：

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

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

用户名和密码的格式

用户名和密码必须遵循以下原则：

- 用户名和密码不能包含表 8-4 中列出的字符。

表 8-4 用户名和密码中不能包含的字符

字符	说明
*	星号
,	逗号
:	冒号

- 用户名和密码不能包含新行符或回车符。
- 如果用户名或密码包含空格，必须将整个用户名或密码放在引号中。
- 用户名或密码必须至少包含一个字符。
- 除了命令 shell 规定的命令行中最多可输入的字符数之外，密码或用户名的长度没有限制。

填充和管理用户系统信息库

要在系统信息库中添加用户，可以使用 `add` 子命令。例如，以下命令将添加用户名为 `Katharine`，密码为 `sesame` 的用户。

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

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

```
imqusermgr delete -u Bob
```

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

```
imqusermgr update -u Katharine -p alladin
```

要列出一个或多个用户的相关信息，可以使用 `list` 命令。以下命令将显示用户名为 `isa` 的相关信息：

```
imqusermgr list -u isa
```

```
-----------------------------------
User Name    Group     Active State
-----------------------------------
isa          admin     true
```

以下命令将列出所有用户的相关信息：

```
imqusermgr list
```

```
---------------------------------------
User Name    Group        Active State
---------------------------------------
testuser3    user         true
testuser2    user         true
testuser1    user         true
isa          admin        true
admin        admin        true
guest        anonymous    true
testuser5    user         false
testuser4    user         false
```

更改默认的管理员密码

为安全起见，必须将 `admin` 的默认密码更改为只有您自己知道的密码。为此，需要使用 `imqusermgr` 工具。

以下命令将默认密码更改为 `grandpoobah`：

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

要快速确认此更改是否已生效，可以在代理运行时运行任何命令行工具。例如，可以使用以下命令：

```
imqcmd list svc -u admin -p grandpoobah
```

使用旧密码时将失败。

更改密码后，使用任何管理工具 （包含管理控制台）时都应该提供新密码。

使用 LDAP 服务器管理用户系统信息库

如果要使用 LDAP 服务器管理用户系统信息库，必须在实例配置文件中设置某些代理属性。设置这些属性后，当某个用户试图连接到代理或执行某些操作时，代理将在 LDAP 服务器中查询用户和组的相关信息。实例配置文件位于以下位置：

IMQ_VARHOME/instances/*brokerName*/props/config.properties
（在 Solaris 操作系统上为
/var/imq/instances/*brokerName*/props/config.properties）

➤ 编辑配置文件以使用 LDAP 服务器的步骤

1. 通过设置以下属性，指定您正在使用 LDAP 用户系统信息库：

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

2. 设置 imq.authentication.type 属性，确定是以 Base64 编码的形式 (basic) 还是以 MD5 摘要的形式 (digest) 将密码从客户机传送给代理。使用 LDAP 目录服务器管理用户系统信息库时，必须将验证类型设置为 basic。例如：

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

3. 还必须设置控制 LDAP 访问的代理属性。表 8-5 列出了存储在代理实例配置文件中的这些属性。 MQ 使用 JNDI API 与 LDAP 目录服务器进行通信。有关这些属性的语法及其所引用术语的详细信息，请参阅 JNDI 文档。 MQ 3.0.1 使用 Sun JNDI LDAP 提供者并使用简单验证。

表 8-5 LDAP 相关属性

属性	说明
imq.user_repository.ldap.server	LDAP 服务器的 host:port。 host 指定运行目录服务器的主机的全限定 DNS 名称。 port 指定目录服务器用于通信的端口号。
imq.user_repository.ldap.principal	代理用来绑定到目录服务器以进行搜索时使用的独特名称。如果目录服务器允许匿名搜索，则无需为此属性指定值。

表 8-5 LDAP 相关属性 （续）

属性	说明
imq.user_repository.ldap.password	与代理使用的独特名称关联的密码。只能在密码文件中指定 （请参阅第 189 页上的 “使用密码文件”）。为了增强安全性，最好让代理提示您输入密码，或使用以下命令行选项指定密码：imqbrokerd -ldappassword 如果目录服务器允许匿名搜索，则无需输入密码。
imq.user_repository.ldap.base	用户条目的目录库。
imq.user_repository.ldap.uidattr	提供者特定的属性标识符，其值用于唯一标识一个用户。例如：uid、 cn 等。
imq.user_repository.ldap.usrfilter	JNDI 搜索过滤器 （以逻辑表达式的形式表示搜索查询）。通过为用户指定搜索过滤器，代理可以缩小搜索范围，从而使搜索更有效。有关详细信息，请参阅位于以下位置的 JNDI 教程：http://java.sun.com/products/jndi/tutorial。 此属性并不是必须设置的。
imq.user_repository.ldap.grpsearch	指定是否启用组搜索的布尔值。请参阅 LDAP 提供者提供的文档，决定您是否可以将用户与组相关联。 请注意， MQ 3.0.1 不支持嵌套组。 默认值：false
imq.user_repository.ldap.grpbase	组条目的目录库。
imq.user_repository.ldap.gidattr	提供者特定的属性标识符，其值为组名。
imq.user_repository.ldap.memattr	组条目中的属性标识符，其值是组成员的独特名称。
imq.user_repository.ldap.grpfiltler	JNDI 搜索过滤器 （以逻辑表达式的形式表示搜索查询）。通过为组指定搜索过滤器，代理可以缩小搜索范围，从而使搜索更有效。有关详细信息，请参阅位于以下位置的 JNDI 教程： http://java.sun.com/products/jndi/tutorial 此属性并不是必须设置的。

表 8-5 LDAP 相关属性 （续）

属性	说明
`imq.user_repository.ldap.timeout`	指定搜索的时间限制 （以秒为单位）的一个整数。默认值为 180 秒。
`imq.user_repository.ldap.ssl.enabled`	指定代理在与 LDAP 服务器通信时是否应使用 SSL 协议的布尔值。默认值为 `false`。

请参阅代理的 default.properties 文件，查看实例 （默认） LDAP 用户系统信息库相关属性的设置。

4. 必要时还需要编辑访问控制属性文件中的用户 / 组和规则。有关使用访问控制属性文件的详细信息，请参阅第 178 页上的 “授权用户：访问控制属性文件”。

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

   - 为 LDAP 用户系统信息库属性指定一个安全端口。例如：

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

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

授权用户：访问控制属性文件

连接到代理后，用户可能希望生成消息、在目标上使用消息或在队列目标上浏览消息。当用户试图进行此类操作时，代理将检查**访问控制属性文件** （ACL 文件），确认该用户是否有权执行此操作。 ACL 文件包含指定某个用户 （或一组用户）可以执行哪些操作的规则。默认情况下，所有授权用户都可以在任何目标上生成和使用消息。您可以编辑访问控制属性文件，将这些操作限制到某些用户和组。

不管用户信息是放置在文本文件系统信息库中还是放置在 LDAP 系统信息库中，都使用 ACL 文件。默认的 ACL 属性文件与代理一起安装。它的名称为 `accesscontrol.properties`，安装程序将其放置在以下目录中：

`IMQ_HOME/etc` （在 Solaris 操作系统上为 `/etc/imq`）

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

- 连接访问控制
- 目标访问控制
- 目标自动创建访问控制

`version` 属性定义 ACL 属性文件的版本，不能更改此条目。

```
version=JMQFileAccessControlModel/100
```

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

访问规则语法

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

这些规则的基本语法如下：

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

表 8-6 介绍了语法规则的元素。

表 8-6 访问规则的语法元素

元素	说明
resourceType	下列选项之一：`connection`、 `queue` 或 `topic`。
resourceVariant	*resourceType* 指定的类型的一个实例。例如， myQueue。通配符 (*) 可用于表示所有连接服务类型或所有目标。
operation	其值取决于所设置访问规则的类型。
access	下列选项之一：`allow` 或 `deny`。
principalType	下列选项之一：`user` 或 `group`。有关详细信息，请参阅第 173 页上的“组”。
principals	可能具有规则左侧指定了访问权限的人。如果 `principalType` 是 `user`，则可能是单个用户或以逗号分隔的用户列表；如果 `principalType` 是 `group`，则可能是单个组或以逗号分隔的组列表。通配符 (*) 可用于表示所有用户或所有组。

下面是一些访问规则实例：

- 以下规则表示所有用户都可以向名为 ql 的队列发送消息。

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

- 以下规则表示任何用户都可以向任何队列发送消息。

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

注 要指定非 ASCII 用户、组或目标名称，必须使用 Unicode 换码符 (\uXXXX) 表示法。如果在您编辑并保的 ACL 文件中，这些名称采用了非 ASCII 编码，则可以通过 Java `native2ascii` 工具将此文件转换为 ASCII。有关详细信息，请访问 `http://java.sun.com/j2se/1.3/docs/guide/intl/faq.html`

权限计算

计算一系列规则所代表的权限时需应用以下原则：

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

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

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

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

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

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

- 用户的 * *principal* 规则将覆盖组的对应 * *principal* 规则。例如，以下两条规则允许所有授权用户向 tq1 发送消息。

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

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

- 授予用户的访问权限将覆盖授予用户所属组的访问权限。在以下实例中，如果 Bob 是 “User” 组的成员，他将不能向 tq1 发送消息，但是 “User” 组中的其他成员则可以向 tq1 发送消息。

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

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

- 任何未通过访问规则明确授予的访问权限均默认为被拒绝。例如，如果 ACL 文件不包含任何访问规则，则所有用户的所有操作都将被拒绝。
- 如果同时允许和拒绝同一个用户或组的访问权限，则访问权限将相互抵消。例如，以下两条规则将使 Bob 无法浏览 tl：

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

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

  以下两条规将使 “User”组无法使用 q5 中的消息。

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

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

- 如果多条规则等号左侧的内容都相同，那么只有最后一条规则起作用。

连接访问控制

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

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

为 *resourceVariant* 定义了两个值：NORMAL 和 ADMIN。默认情况下，所有用户都可以访问 NORMAL 类型的连接服务，但只有 admin 组中的用户可以访问 ADMIN 类型的连接服务。

您可以编辑连接访问控制规则，限制某个用户的连接访问权限。例如，以下规则将拒绝 Bob 访问 NORMAL，但允许其他人访问：

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

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

可以使用星号 (*) 指定所有授权用户或组。

不能创建自己的服务类型，只能使用常量 NORMAL 和 ADMIN 指定的预定义类型。

目标访问控制

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

默认情况下，任何用户或组都能够对任何目标进行任意类型的访问。您可以添加更多特定的目标访问规则或编辑默认的规则。本节下面介绍目标访问规则的语法，必须理解该语法才能编写自己的规则。

目标规则的语法如下：

resourceType.resourceVariant.operation.access.principalType = principals

表 8-7 列出了这些元素：

表 8-7 目标访问控制规则的元素

组件	说明
resourceType	必须是 `queue` 或 `topic` 其中之一。
resourceVariant	某个目标名或所有目标 (*)，星号表示所有队列或所有主题。
operation	必须是 `produce`、 `consume` 或 `browse` 其中之一。
access	必须是 `allow` 或 `deny` 其中之一。
principalType	必须是 `user` 或 `group` 其中之一。

可以将访问权限授予一个或多个用户和/或一个或多个组。

以下实例说明了不同类型的目标访问控制规则：

- 允许所有用户向任意 queue 目标发送消息。

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

- 拒绝 `user` 组的任何成员订阅 Admissions 主题。

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

目标自动创建访问控制

ACL 属性文件的最后一部分访问规则指定代理将为哪些用户和组自动创建目标。

当用户在尚不存在的目标上创建生成方或使用方时，如果代理的 auto-create 属性已启用且物理目标尚不存在，代理将会创建该目标。

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

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

您可以编辑 ACL 文件以限制此类访问权限。

目标自动创建访问规则的一般语法如下：

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

其中 *resourceType* 为 queue 或 topic。

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

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

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

加密：使用 SSL 服务 （企业版）

MQ Enterprise Edition 支持基于安全套接字层 (SSL) 标准的连接服务：通过 TCP/IP （ssljms 和 ssladmin）和通过 HTTP (httpsjms)。这些基于 SSL 的连接服务允许对在客户机和代理之间传递的消息进行加密。当前的 MQ 发行版支持基于自签名服务器证书的 SSL 加密。

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

尽管 MQ 基于 SSL 的连接服务在概念上很相似，但它们的设置方法却不尽相同。因此以下各节将分别讨论通过 TCP/IP 和通过 HTTP 的安全连接。

设置通过 TCP/IP 的 SSL 服务

有两种基于 SSL 的连接服务通过 TCP/IP 提供直接、安全的连接。

ssljms 此连接服务用于在客户机和代理之间通过安全、加密的连接传送 JMS 消息。

ssladmin 此连接服务用于在命令行实用程序 (`imqcmd`) （命令行管理工具）和代理之间创建安全、加密的连接。不支持管理控制台 (`imqadmin`) 安全连接。

➤ **设置 ssljms 连接服务的步骤**

1. 生成自签名证书。
2. 在代理中启用 ssljms 连接服务。
3. 启动代理。
4. 配置并运行客户机。

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

下文详细介绍了每个步骤。

步骤 1：生成自签名证书

MQ 3.0.1 中的 SSL 支持用于保护所传输数据的安全，假定客户机正在与已知且可信任的服务器进行通信。因此，在 MQ 3.0.1 中，仅使用自签名证书实现 SSL。

运行 `imqkeytool` 命令，为代理生成自签名证书。可以对 ssljms 和 ssladmin 连接服务使用相同的证书。在命令提示符下输入以下内容：

```
imqkeytool -broker
```

命令行实用程序会提示您提供所需的信息。（在 Unix 操作系统上，可能需要以超级用户 [root 用户] 身份运行 `imqkeytool` 命令，以获得创建键存储所需的权限。）

首先，`imqkeytool` 提示您输入键存储密码，然后提示您输入组织信息，最后提示您进行确认。收到您的确认后，它将在生成键对时暂停。然后它要求您输入密码以锁定特定关键字对 （关键字密码），此时请按回车键响应此提示：这将使关键字密码与键存储密码相同。

注 请记住您提供的密码，稍后 （启动代理时）您需要向代理提供此密码，使代理可以打开键存储。您还可以将键存储密码存储在密码文件中 （请参阅第 189 页上的 “使用密码文件”）。

运行 imqkeytool 以运行 JDK keytool 实用程序，生成自签名证书并将其放置在位于以下位置的 MQ 键存储中：

IMQ_HOME/etc/keystore （在 Solaris 上操作系统上为 /etc/imq/keystore）

键存储格式与 JDK1.2 keytool 实用程序支持的格式相同。

表 8-8 列出了 MQ 键存储的配置属性。（有关配置这些属性的说明，请参阅第 5 章 “启动与配置代理”）。

表 8-8 键存储属性

属性名称	说明
imq.keystore.file.dirpath	适用于基于 SSL 的服务：指定包含键存储文件的目录的路径。 默认值：IMQ_HOME/etc （在 Solaris 操作系统上为 /etc/imq/）
imq.keystore.file.name	适用于基于 SSL 的服务：指定键存储文件的名称。 默认值：keystore
imq.keystore.password	适用于基于 SSL 的服务：指定键存储密码。只能存储在密码文件中 （请参阅第 189 页上的 “使用密码文件”）。为了增强安全性，最好让代理提示您输入密码，或使用以下命令行选项指定密码： imqbrokerd -password

您可能需要重新生成密钥对以解决某些问题，例如：

- 您忘记了键存储密码。
- 启动代理时 SSL 服务初始化失败并出现异常：
  java.security.UnrecoverableKeyException:Cannot recover key.

  出现异常的原因可能是您提供的密码与您在第 184 页上的 “步骤 1：生成自签名证书”中生成自签名证书时的键存储密码不同。

➤ **重新生成密钥对的步骤**

1. 删除代理的键存储，它位于以下位置：

   `IMQ_HOME/etc/keystore` （在 Solaris 上操作系统上为 `/etc/imq/keystore`）

2. 再次运行 `imqkeytool`，生成第 184 页上的 “步骤 1：生成自签名证书”中描述的密钥对。

步骤 2：在代理上启用基于 SSL 的服务

要在代理上启用 SSL 服务，您需要将 `ssljms` (`ssladmin`) 添加到 `imq.service.activelist` 属性中。

1. 打开代理的实例配置文件。您可以从以下位置找到它：

   `IMQ_VARHOME/instances/`*`brokerName`*`/props/config.properties`
   （在 Solaris 操作系统上为
   `/var/imq/instances/`*`brokerName`*`/props/config.properties`）

   其中 *brokerName* 是代理实例的名称。

2. 将 `ssljms` 和/或 `ssladmin` 值 （取决于您需要的服务）添加到 `imq.service.activelist` 属性中：

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

步骤 3：启动代理

启动代理并提供键存储密码。可以通过以下方法之一提供密码：

- 让代理启动时提示您输入密码：

  ```
  imqbrokerd
  Please enter Keystore password:mypassword
  ```

- 使用 `imqbrokerd` 命令的 `-password` 选项：

  ```
  imqbrokerd -password mypassword
  ```

- 将密码放在代理启动时访问的密码文件中 （请参阅第 189 页上的 “使用密码文件”）。您需要首先设置以下代理配置属性 （请参阅第 110 页上的 “编辑实例配置文件”）：

  ```
  imq.passfile.enabled=true
  ```

设置此属性后，您可以通过以下两种方式之一访问密码文件：

- 将密码文件的位置传递给 imqbrokerd 命令：

  ```
  imqbrokerd -passfile /tmp/mypassfile
  ```

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

  ```
  imq.passfile.dirpath=/tmp
  imq.passfile.name=mypassfile
  ```

要查看与密码文件相关的代理属性列表，请参阅第 57 页上的表 2-6。

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

步骤 4：配置并运行基于 SSL 的客户机

最后，您需要配置客户机以使用安全连接服务。根据您使用的连接服务，有两种类型的客户机：使用 ssljms 的 JMS 客户机和使用 ssladmin 的 MQ 管理命令行实用程序 (imqcmd)。下面分别介绍这两种客户机。

JMS 客户机

您需要确保客户机的类路径中有必需的安全套接扩展 (JSSE) Jar 文件，还需要告诉该文件使用 ssljms 连接服务。

1. 如果您的客户机不是使用 J2SDK1.4 （它具有内置的 JSSE 和 JNDI 支持），请确保客户机的类路径中有以下 Jar 文件：

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

2. 请确保客户机的类路径中有以下 MQ Jar 文件：

   ```
   imq.jar 和 jms.jar
   ```

3. 启动客户机并连接到代理的 ssljms 服务。执行上述操作的方法之一是输入如下命令：

   ```
   java -DimqConnectionType=TLS clientAppName
   ```

   设置 imqConnectionType，通知连接使用 SSL。

   有关在客户机应用程序中使用 ssljms 连接服务的详细信息，请参阅 《MQ *Developer's Guide*》中有关使用被管理对象的章节。

命令行实用程序 *(imqcmd)*

您可以通过在使用 `imqcmd` 时包含 `-secure` 选项来建立一个安全管理连接 （请参阅第 132 页上的表 6-2），例如：

```
imqcmd list svc -b hostName:port -u adminName -p adminPassword -secure
```

其中 *adminName* 和 *adminPassword* 是 MQ 用户系统信息库中的有效条目 （如果使用文本文件系统信息库，请参阅第 175 页上的 “更改默认的管理员密码”）。

列出连接服务 （如本实例所示）是查看 `ssladmin` 服务是否正在运行的一种方法，您可以成功建立安全管理连接，如以下输出所示：

```
Listing all the services on the broker specified by:

Host                    Primary Port
localhost               7676

Service Name      Port Number         Service State
admin             33984 (dynamic)     RUNNING
httpjms           -                   UNKNOWN
httpsjms          -                   UNKNOWN
jms               33983 (dynamic)     RUNNING
ssladmin          35988 (dynamic)     RUNNING
ssljms            dynamic             UNKNOWN

Successfully listed services.
```

设置通过 HTTP 的 SSL 服务

在这种基于 SSL 的连接服务 (httpsjms) 中，客户机和代理通过 HTTPS 隧道 Servlet 建立安全连接。有关 HTTPS 支持所需的体系结构以及实现方法，请参阅第 197 页上的附录 B “HTTP/HTTPS 支持 （企业版）”。

使用密码文件

如果您不希望代理启动时提示您输入密码，或者要求您作为 `imqbrokerd` 命令选项提供密码，您可以将所需的密码放在**密码文件**中。

密码文件是包含密码的纯文本文件。该文件没有加密，因此没有手动提供密码安全。但您可以为该文件设置权限，限制哪些用户可以查看它。密码文件的权限需要提供给启动代理权限并需要读取该文件的用户。

表 8-9 列出了密码文件可能包含的密码：

表 8-9 密码文件中的密码

密码	说明
`imq.keystore.password`	为基于 SSL 的服务指定键存储密码。
`imq.user_repository.ldap.password`	指定与代理的独特名称（用于绑定到已配置的 LDAP 用户系统信息库）相关联的密码。
`imq.persist.jdbc.password`	指定必要时用于打开数据库连接的密码。

实例密码文件可以从以下位置找到：

`IMQ_HOME/etc/`passfile.sample（在 Solaris 操作系统上为 `/etc/imq/passfile.sample`）

附录 A

设置插入的持久性

本附录说明如何设置代理，使之使用插入的持久性访问支持 JDBC 的数据存储。

简介

Sun™ ONE Message Queue (MQ) 代理包含一个持久性管理器组件 （请参阅第 52 页上的 “持久性管理器”），用于管理持久性信息的写入和检索。默认情况下，持久性管理器被配置为访问基于文件的内置数据存储，但是您可以重新对其进行配置，以插入任何可通过 JDBC 兼容的驱动程序进行访问的数据存储。

要配置代理以使用插入的持久性，需要在代理实例配置文件中设置一些 JDBC 相关的属性。还需要创建相应的数据库架构以执行 MQ 持久性操作。 MQ 提供的数据库管理器实用程序 (`imqdbmgr`) 可以使用 JDBC 驱动程序和代理配置属性创建和管理插入的数据库。

本附录中列出的步骤以 Java 2 SDK Enterprise Edition 附带的 Cloudscape DBMS 为例进行说明 （J2EE SDK 可以从 java.sun.com 下载）。本实例使用 Cloudscape 的嵌入式版本 （而不是客户机/服务器版本）。这些步骤以 Cloudscape 实例中的路径名和属性名为例进行说明。它们标有 “实例：” 字样。

其它实例可以从以下地址找到：

`IMQ_HOME/demo/jdbc` （在 Solaris 操作系统上为 `/usr/demo/imq/jdbc`）

插入支持 JDBC 的数据存储

插入支持 JDBC 的数据存储只需几个步骤。

➤ 插入支持 JDBC 的数据存储的步骤

1. 在代理的配置文件中设置 JDBC 相关属性。

   请参阅第 193 页上的表 A-1 中列出的属性。

2. 在以下路径中放置一个 JDBC 驱动程序 jar 文件的副本或符号链接：

   `IMQ_VARHOME/lib/ext` （在 Solaris 操作系统上为 `/usr/share/lib/imq/ext/`）

   副本实例 *(Solaris)*：

   ```
   % cp j2sdk_install_directory/lib/cloudscape/cloudscape.jar
   IMQ_VARHOME/lib/ext
   ```

   符号链接实例 *(Solaris)*：

   ```
   % ln -s j2sdk_install_directory/lib/cloudscape/cloudscape.jar
   IMQ_VARHOME/lib/ext
   ```

3. 创建 MQ 持久性所需的数据库架构。

   使用 `imqdbmgr create all` 命令 （对于嵌入式数据库）或者 `imqdbmgr create tbl` 命令 （对于外部数据库）。请参阅第 195 页上的 “数据库管理器实用程序 (imqdbmgr)”。

   实例：

   ```
   % cd IMQ_HOME/bin （在 Solaris 操作系统上为 /usr/bin）
   % imqdbmgr create all
   ```

注 如果使用嵌入式数据库，建议在以下目录中创建数据库架构：

`IMQ_VARHOME/instances/`*brokerName*`/dbstore/`*dabatabseName*.
（在 Solaris 操作系统上为
/var/imq/instances/*brokerName*`/dbstore/`）

如果嵌入式数据库没有设置用户名和密码保护，则可能设置了文件系统权限保护。要确保代理能够对数据库进行读写操作，运行该代理的用户应该是使用 imqdbmgr 命令创建该嵌入式数据库的同一个用户（请参阅第 195 页上的 “数据库管理器实用程序 (imqdbmgr)”）。

JDBC 相关的代理配置属性

代理的实例配置文件位于以下位置：

IMQ_VARHOME/instances/*brokerName*/props/config.properties
（在 Solaris 操作系统上为
/var/imq/instances/*brokerName*/props/config.properties）

如果该文件尚未创建，必须使用 -name 代理名称选项启动 MQ 的代理，以创建该文件。

表 A-1 列出了插入支持 JDBC 的数据存储时需要设置的配置属性。这些属性是在使用插入持久性的每个代理实例的实例配置文件 (config.properties) 中设置的。该表包含了需要为 Cloudscape DBMS 实例指定的值。

表 A-1 JDBC 相关属性

属性名称	说明
imq.persist.store	指定基于文件或基于 JDBC 的数据存储。 **实例：** jdbc
imq.persist.jdbc.brokerid（可选）	指定附加到数据库表名称中的代理实例标识符。当多个代理实例使用同一个数据库作为持久性数据存储时，标识符可使代理实例保持唯一。（嵌入式数据库只存储一个代理实例的数据，因此通常不需要代理实例标识符。）标识符必须是字母数字字符串，其长度不能超过数据库所允许的最大表名长度减去 12 后的长度。 **实例：**对于 Cloudscape 来说不需要
imq.persist.jdbc.driver	指定连接到数据库的 JDBC 驱动程序的 Java 类名。 **实例：** COM.cloudscape.core.JDBCDriver
imq.persist.jdbc.opendburl	指定打开现有数据库连接的数据库 URL。 **实例：** jdbc:cloudscape:IMQ_VARHOME/instances/*brokerName*/dbstore/imqdb（在 Solaris 操作系统上为 jdbc:cloudscape:var/imq...）

表 A-1 JDBC 相关属性 （续）

属性名称	说明
`imq.persist.jdbc.createdburl`（可选）	指定打开创建数据库连接的数据库 URL。（仅当要使用 `imqdbmgr` 创建数据库时指定。） **实例：** `jdbc:cloudscape:IMQ_VARHOME/instances/`*`brokerName`*`/dbstore/imqdb;create=true` （在 Solaris 操作系统上为 `jdbc:cloudscape:var/imq...`）
`imq.persist.jdbc.closedburl`（可选）	指定当代理关闭时关闭当前数据库连接的数据库 URL。 **实例** （对 Cloudscape 来说是必需的）： `jdbc:cloudscape:;shutdown=true`
`imq.persist.jdbc.user`（可选）	指定打开数据库连接的用户名 （如果需要）。出于安全性考虑，可以指定值，而不要使用命令行选项： `imqbrokerd -dbuser` 和 `imqdbmgr -u`
`imq.persist.jdbc.needpassword`（可选）	指定数据库是否要求为代理访问提供密码。如果值为 `true`，则表示需要提供密码。可以使用以下命令行选项指定密码： `imqbrokerd -dbpassword imqdbmgr -p` 如果没有使用命令行选项或密码文件提供密码（请参阅第 189 页上的 “使用密码文件”），代理将提示输入密码。
`simq.persist.jdbc.password`（可选）	指定用于打开数据库连接的密码 （如果需要）。只能在密码文件中指定 （请参阅第 189 页上的 “使用密码文件”）。为了增强安全性，最好让代理提示您输入密码，或使用以下命令行选项指定密码： `imqbrokerd -dbpassword imqdbmgr -p`

所有代理配置属性的值都可以使用 -D 命令行选项进行设置。如果某个数据库要求设置某些特定的数据库属性，也可以在启动代理 (imqbrokerd) 时使用 -D 命令行选项进行设置，或者使用数据库管理器实用程序 (imqdbmgr) 进行设置。

实例：

以 Cloudscape 嵌入式数据库为例，可以使用 -D 命令行选项定义 Cloudscape 系统目录，而不是在数据库连接 URL 中指定数据库的绝对路径 （如表 A-1 中的实例所示）：

```
-Dcloudscape.system.home=IMQ_VARHOME/instances/brokerName/dbstore
```

在这种情况下，可以分别将用于创建和打开数据库的 URL 简单指定为：

```
imq.persist.jdbc.createdburl=jdbc:cloudscape:imqdb;create=true
```

和

```
imq.persist.jdbc.opendburl=jdbc:cloudscape:imqdb。
```

数据库管理器实用程序 (imqdbmgr)

MQ 为设置持久性所需的架构提供了数据库管理器实用程序 (imqdbmgr)。当 MQ 数据库表被损坏或者您希望使用不同的数据库作为数据存储时，也可以使用该实用程序删除数据库表。

注 如果代理实例异常崩溃，未引用的持久性信息可能会保留在数据存储中。启动代理后再将其关闭通常可以清除数据存储。

本节介绍了基本的 imqdbmgr 命令语法，提供了一个子命令列表，并概述了 imqdbmgr 命令选项。

命令语法

imqdbmgr 命令的一般语法如下：

```
imqdbmgr subcommand argument [options]
imqdbmgr -h|-help
imqdbmgr -v|-version
```

请注意，如果指定 -v 或 -h 选项，将不会执行命令行中指定的子命令。例如，输入以下命令将显示版本信息，而不是执行 create 子命令。

```
imqdbmgr create all -v
```

imqdbmgr 子命令

表 A-2 列出了数据库管理器实用程序 (imqdbmgr) 包含的子命令：

表 A-2 imqdbmgr 子命令

子命令和参数	说明
create all	创建一个新的数据库和 MQ 持久性存储器架构。此命令用于嵌入式数据库系统，使用时需指定 imq.persist.jdbc.createdburl 属性。
create tbl	在现有数据库系统中创建 MQ 持久性存储器架构。此命令用于外部数据库系统。
delete tbl	删除当前持久性存储器数据库中的现有 MQ 数据库表。
recreate tbl	删除当前持久性存储器数据库中的现有 MQ 数据库表，然后重新创建 MQ 持久性存储器架构。

imqdbmgr 命令选项概述

表 A-3 列出了 imqdbmgr 命令的选项。

表 A-3 imqdbmgr 选项

选项	说明
-D*property*=*value*	将指定属性设置为指定的值。
-b *brokerName*	指定代理实例名称并使用对应的实例配置文件。
-h	显示使用帮助。不执行命令行上的命令。
-p *password*	指定数据库密码。
-u *name*	指定数据库用户名。
-v	显示版本信息。不执行命令行上的命令。

HTTP/HTTPS 支持 （企业版）

Sun™ ONE Message Queue (MQ) Enterprise Edition （请参阅第 26 页上的 “产品版本”）支持 HTTP 和 HTTPS 连接。（HTTPS 连接使用安全套接字层标准，比 HTTP 连接更安全。）此支持使客户机应用程序可以使用 HTTP 协议 （而不是直接 TCP 连接）与代理进行通信。本附录介绍了实现此支持所使用的体系结构，并说明了如何设置客户机，使之能够使用基于 HTTP 的连接进行 MQ 消息传送。

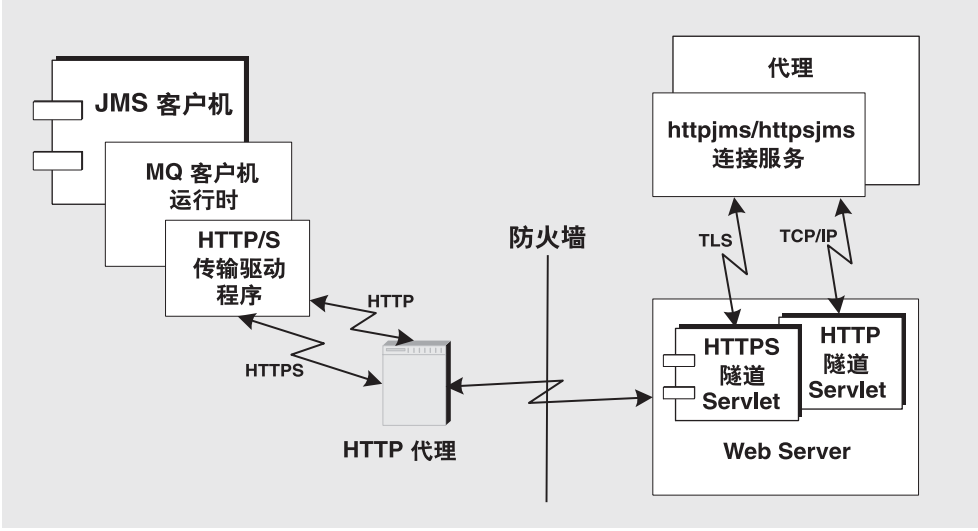
HTTP/HTTPS 支持体系结构

可以通过 HTTP/HTTPS 连接进行 MQ 消息传送。因为 HTTP/HTTPS 连接通常可以穿过防火墙，因此可以通过防火墙将客户机应用程序与代理分隔开。

第 198 页上的图 B-1 列出了提供 HTTP/HTTPS 支持所需的主要组件。

- 在客户端， HTTP 传输驱动程序将 MQ 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， JMS 客户机可以使用 HTTP 代理服务器与代理进行通信。可以在启动客户机时使用命令行选项指定代理的地址。有关详细信息，请参阅第 202 页上的 “使用 HTTP 代理”。
- 将 JMS 消息转发给代理之前，需要将 HTTP 或 HTTPS 隧道 Servlet（均随 MQ 一起提供）装入 Web 服务器，并使用它们从客户机 HTTP 请求中提取 JMS 消息。 HTTP/HTTPS 隧道 Servlet 还将代理消息发送回客户机，以响应客户机发出的 HTTP 请求。可以使用一个 HTTP/HTTPS 隧道 Servlet 访问多个代理。

图 B-1 HTTP/HTTPS 支持体系结构



- 在代理端， httpjms 或 httpsjms 连接服务对来自相应隧道 Servlet 的消息进行展开和分离。
- 如果 Web 服务器失败并重新启动，所有连接都将恢复且不会对客户机产生影响。如果代理失败并重新启动，将抛出异常，而客户机必须重新建立连接。如果 Web 服务器和代理同时失败 （这种情况不太可能发生）而代理没有重新启动， Web 服务器将恢复客户机连接并等待代理恢复连接而不会通知客户机。要避免这种情况，应始终重新启动代理。

从图 B-1 可以看出， HTTP 和 HTTPS 支持所需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 与客户机应用程序和代理之间的连接都是安全的。

与代理之间的安全连接是通过支持 SSL 的隧道 Servlet （MQ 的 HTTPS 隧道 Servlet）提供的，该 Servlet 会向请求连接的任何代理发送自签名证书。代理使用该证书建立与 HTTPS 隧道 Servlet 的加密连接。建立此连接后，客户机应用程序和 Web 服务器可以协商建立客户机应用程序和隧道 Servlet 之间的安全连接。

实现 HTTP 支持

以下各节介绍了实现 HTTP 支持所需执行的步骤。

➤ **实现 HTTP 支持的步骤**

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。

步骤 1：在 Web 服务器上部署 HTTP 隧道 Servlet

在 Web 服务器上部署 HTTP 隧道 Servlet 的常见方法有两种：

- 将其部署为支持 Servlet 2.1 或更早版本的 Web 服务器 Jar 文件
- 将其部署为支持 Servlet 2.2 或更新版本的 Web 服务器 Web 归档 (WAR) 文件

部署为 Jar 文件

部署 MQ 隧道 Servlet 包括两个步骤，首先要使主机 Web 服务器可以访问相应的 Jar 文件，然后要对该 Web 服务器进行配置，使之在启动时装入该 Servlet。

隧道 Servlet Jar 文件 (`imqServlet.jar`) 包含 HTTP 隧道 Servlet 所需的所有类，该文件位于以下目录中：

`IMQ_HOME/lib`（在 Solaris 操作系统上为 `/usr/share/lib/imq`）

可以使用任何支持 Servlet 2.x 的 Web 服务器装入此 Servlet。 Servlet 类名为：

```
com.sun.messaging.jmq.transport.
httptunnel.servlet.HttpTunnelServlet
```

Web 服务器必须能够访问 `imqservlet.jar` 文件。如果计划在不同的主机上运行 Web 服务器和代理，应该在 Web 服务器能够访问的位置放置 `imqservlet.jar` 文件的一个副本。

还需要配置 Web 服务器，使之在启动时装入此 Servlet （请参阅第 203 页上的 “实例：部署 HTTP 隧道 Servlet”）。

同时建议您禁用 Web 服务器的访问日志功能以提高性能。

部署为 Web 归档文件

要将 HTTP 隧道 Servlet 部署为 WAR 文件，需要使用 Web 服务器提供的部署机制。 HTTP 隧道 Servlet WAR 文件 (`imqhttp.war`) 位于以下目录中：

`IMQ_HOME/lib`（在 Solaris 操作系统上为 `/usr/share/lib/imq`）

WAR 文件包含一个部署描述符，该描述符包含 Web 服务器装入和运行 Servlet 所需的基本配置信息。

步骤 2：配置 httpjms 连接服务

默认情况下， MQ 3.0.1 代理没有启用 HTTP 支持，因此需要重新配置代理以启用 httpjms 连接服务。重新配置后，可以按照第 114 页上的 “启动代理”中介绍的步骤启动代理。

➤ **启用 httpjms 连接服务的步骤**

1. 打开位于以下位置的代理实例配置文件：

   `IMQ_VARHOME/instances/`*`brokerName`*`/props/config.properties`
   （在 Solaris 操作系统上为
   `/var/imq/instances/`*`brokerName`*`/props/config.properties`）

   其中 *brokerName* 是代理实例的名称。

2. 将 `httpjms` 值添加到 `imq.service.activelist` 属性中：

   ```
   imq.service.activelist=jms,admin,httpjms
   ```

启动时，代理将在其主机上查找运行的 Web 服务器和 HTTP 隧道 Servlet。但是要访问远程隧道 Servlet，需要重新配置 servletHost 和 servletPort 连接服务属性。

还可以重新配置 pullPeriod 属性以提高性能。第 201 页上的表 B-1 详细介绍了 httpjms 连接服务配置属性。

表 B-1 httpjms 连接服务属性

属性名称	说明
imq.httpjms.http.servletHost	必要时可以更改此值，以指定运行 HTTP 隧道 Servlet 的主机的名称 （主机名或 IP 地址）。（可以是远程主机或本地主机上的特定主机名。）默认值：localhost
imq.httpjms.http.servletPort	需要更改此值，以指定代理用于访问 HTTP 隧道 Servlet 的端口号。（如果更改了 Web 服务器上的默认端口，也必须对此属性做相应的更改。）默认值：7675
imq.httpjms.http.pullPeriod	指定每一台客户机创建从代理提取消息的 HTTP 请求的时间间隔 （以秒为单位）。如果值为零或为负数，客户机将始终使一个 HTTP 请求处于待处理状态，这样可以随时尽快地提取消息。如果客户机的数量过多，可能会对 Web 服务器资源产生较大压力，从而导致服务器停止响应。在这种情况下，应将 pullPeriod 属性设置为正秒数。此属性设置客户机 HTTP 传输驱动程序在发出下一个提取请求之前等待的时间。将此属性设置为正值将以牺牲客户机响应时间为代价来保持 Web 服务器资源的可用性。默认值：-1

步骤 3：配置 HTTP 连接

客户机应用程序必须使用正确配置的连接工厂被管理对象，建立与代理之间的 HTTP 连接。本节介绍 HTTP 连接配置问题。

设置连接工厂属性

要实现 HTTP 支持，需要设置以下连接工厂属性 （请参阅第 156 页上的 “连接工厂被管理对象”）：

- 将 imqConnectionType 属性设置为 HTTP
- 将 imqConnectionURL 设置为 HTTP 隧道 Servlet URL

  http://*hostName*:*port*/imq/tunnel

可以按照以下方法之一设置连接工厂属性：

- 在创建连接工厂被管理对象 （请参阅第 163 页上的 “添加连接工厂”）的 imqobjmgr 命令中使用 -o 选项，或在使用管理控制台 (imqadmin) 创建连接工厂时设置属性。

- 在启动客户机应用程序的命令中使用 -D 选项 （请参阅 《MQ *Developer's Guide*》）。
- 通过编程方式在客户机应用程序代码中创建被管理对象 （请参阅 《MQ *Developer's Guide*》）之后，使用 JMS API 调用设置其属性。

使用一个 Servlet 访问多个代理

即使正在运行多个代理，也无需配置多个 Web 服务器和多个 Servlet 实例。可以在并行运行的多个代理之间共享一个 Web 服务器和一个 HTTP 隧道 Servlet 实例。要实现此目的，必须按照如下所示配置 imqConnectionURL 连接工厂属性：

http://*hostName:port/imq/tunnel*?ServerName=*hostName*:*brokerName*

其中 *hostName* 为代理主机名， *brokerName* 是您希望客户机访问的特定代理实例的名称。

要查看是否输入了正确的 *hostName* 和 *brokerName*，可以通过浏览器访问 Servlet URL，生成 HTTP 隧道 Servlet 的状态报告。状态报告将列出 Servlet 正在访问的所有代理：

```
HTTP tunnel servlet ready.
Servlet Start Time :Thu May 30 01:08:18 PDT 2002
Accepting TCP connections from brokers on port : 7675
Total available brokers = 2
Broker List :
   jpgserv:broker2
   cochin:broker1
```

使用 HTTP 代理

如果使用 HTTP 代理访问 HTTP 隧道 Servlet：

- 将 http.proxyHost 系统属性设置为代理服务器主机名。
- 将 http.proxyPort 系统属性设置为代理服务器端口号。

可以通过在启动客户机应用程序的命令中使用 -D 选项来设置这些属性。

实例：部署 HTTP 隧道 Servlet

本节说明如何在 Sun ONE Web Server 上将 HTTP 隧道 Servlet 部署为 Jar 文件和 WAR 文件。使用的方法取决于 Sun ONE Web Server 的版本：如果 Sun ONE Web Server 不支持 Servlet 2.2 或更新版本，则不能处理 WAR 文件部署。

部署为 Jar 文件

下面的说明适用于使用基于浏览器的管理 GUI 在 Sun ONE Web Server FastTrack Edition 4.1 上部署 HTTP 隧道 Servlet 的情况。此过程包含以下通用步骤：

1. 添加 Servlet
2. 配置 Servlet 虚拟路径
3. 装入 Servlet
4. 禁用 Servlet 访问日志

以下小节对这些步骤进行了说明。通过使用 Web 浏览器访问 Servlet URL 可以验证是否成功部署了 HTTP 隧道 Servlet。它应该显示状态信息。

添加 *Servlet*

➤ **添加隧道 Servlet 的步骤**

1. 选择 “Servlets”选项卡。
2. 选择 “Configure Servlet Attribute”。
3. 在 “Servlet Name”字段中指定隧道 Servlet 的名称。
4. 将 “Servlet Code”（类名）字段设置为以下值：

   ```
   com.sun.messaging.jmq.transport.
   httptunnel.servlet.HttpTunnelServlet
   ```

5. 在 “Servlet Classpath”字段中输入 `imqservlet.jar` 的完整路径。例如：

   ```
   IMQ_HOME/lib/imqservlet.jar
   ```

   （在 Solaris 操作系统上为 `/usr/share/lib/imq/imqservlet.jar`）

6. 在 “Servlet args”字段中输入可选参数，如表 B-2 所示：

表 B-2 用于部署 HTTP 隧道 Servlet Jar 文件的 Servlet 参数

参数	默认值	参阅
servletHost	所有主机	请参阅第 201 页上的表 B-1
servletPort	7675	请参阅第 201 页上的表 B-1

如果使用两个参数，请使用逗号分隔：

`servletPort=`*portnumber*`, servletHost=...`

`serverHost` 和 `serverPort` 参数仅应用于 Web 服务器和代理之间的通信，仅当默认值存在问题时才需要进行设置。但是，在这种情况下，还必须相应地设置代理配置属性 （请参阅第 201 页上的表 B-1），例如：

```
imq.httpjms.http.servletPort
```

配置 *Servlet* 虚拟路径 *(Servlet URL)*

➤ **配置隧道 Servlet 虚拟路径 (Servlet URL) 的步骤**

1. 选择 “Servlets”选项卡。
2. 选择 “Configure Servlet Virtual Path Translation”。
3. 设置 “Virtual Path”字段。

   例如，要使 URL 类似 `http://`*hostName*`:`*port*`/imq/tunnel`，请在 “Virtual Path”字段中输入以下字符串：

   ```
   /imq/tunnel
   ```

4. 设置 “Servlet Name”字段的值，使之与第 203 页上的 “添加 Servlet”中的步骤 3 中设置的值相同。

装入 *Servlet*

➤ **在 Web 服务器启动时装入隧道 Servlet 的步骤**

1. 选择 “Servlets”选项卡。
2. 选择 “Configure Global Attributes”。
3. 在 “Startup Servlets”字段中输入与第 203 页上的 “添加 Servlet”中的步骤 3 中的值相同的 Servlet 名称值。

禁用服务器访问日志

不是必须禁用服务器访问日志，但禁用服务器访问日志可以获得更好的性能。

➤ **禁用服务器访问日志的步骤**

1. 选择 “Status”选项卡。
2. 选择 “Log Preferences Page”。
3. 使用 “Log”客户机访问控制禁用日志。

部署为 WAR 文件

下面的说明适用于在 Sun ONE Web Server 6.0 SP2 上的部署。通过使用 Web 浏览器访问 Servlet URL 可以验证是否成功部署了 HTTP 隧道 Servlet。它应该显示状态信息。

➤ **将 HTTP 隧道 Servlet 部署为 WAR 文件的步骤**

1. 在基于浏览器的管理 GUI 中，选择 “Virtual Server Class”选项卡并选择 “Manage Classes”。
2. 选择相应的虚拟服务器类名 （例如 defaultclass）并单击 “Manage”按钮。
3. 选择 “Manage Virtual Servers”。
4. 选择相应的虚拟服务器名称并单击 “Manage”按钮。
5. 选择 “Web Applications”选项卡。
6. 在 “Deploy Web Application”上单击。
7. 为 “WAR File On”和 “WAR File Path”字段选择相应的值，以指向 `imqhttp.war` 文件。该文件位于以下目录中：

   `IMQ_HOME/lib` （在 Solaris 操作系统上为 `/usr/share/lib/imq`）
8. 在 “Application URI”字段中输入 `/imq`。
9. 输入要在其中部署 Servlet 的安装目录路径 （通常位于 Sun ONE Web Server 的安装根目录下）。
10. 单击 “`OK`”。
11. 重新启动 Web 服务器实例。

Servlet 现在即被部署到以下位置：

`http://`*`hostName`*`:`*`port`*`/imq/tunnel`

客户机现在即可使用此 URL 连接至使用 HTTP 连接的消息服务。

实现 HTTPS 支持

以下各节说明实现 HTTPS 支持所需执行的操作。所需操作与第 199 页上的 “实现 HTTP 支持”中的操作类似，只是增加了生成和访问 SSL 证书的步骤。

➤ **实现 HTTPS 支持的步骤**

1. 为 HTTPS 隧道 Servlet 生成自签名证书。
2. 在 Web 服务器上部署 HTTPS 隧道 Servlet。
3. 配置代理的 httpjms 连接服务并启动代理。
4. 配置 HTTPS 连接。

下文详细介绍了每个步骤。

步骤 1：为 HTTPS 隧道 Servlet 生成自签名证书

MQ 3.0.1 中的 SSL 支持用于保护所传输数据的安全，假定客户机正在与已知且可信任的服务器进行通信。因此，在 MQ 3.0.1 中，仅使用自签名的服务器证书实现 SSL。在 httpsjms 连接服务体系结构中， HTTPS 隧道 Servlet 充当代理和应用程序客户机的服务器。

运行 `imqkeytool` 实用程序，为隧道 Servlet 生成自签名证书。在命令提示符下输入以下内容：

```
imqkeytool -servlet keystore_location
```

命令行实用程序会提示您提供所需的信息。（在 Unix 操作系统上，可能需要以超级用户 [root 用户] 身份运行 `imqkeytool` 命令，以获得创建键存储所需的权限。）

首先， `imqkeytool` 会提示您输入键存储密码，然后提示您输入组织信息，最后提示您进行确认。收到您的确认后，它将在生成键对时暂停。然后它要求您输入密码以锁定特定密钥字对 （密钥密码），此时请按回车键响应此提示：这将使密钥密码与密钥存储密码相同。

注 请记住您输入的密码，稍后您需要将此密码提供给隧道 Servlet 使其能够打开键存储。

运行 imqkeytool 命令以运行 JDK keytool 实用程序，生成自签名证书并将其放置在 MQ 的键存储文件中，该文件的位置由 *keystore_location* 参数指定。（键存储格式与 JDK1.2 keytool 支持的键存储格式相同。）

注 HTTPS 隧道 Servlet 必须可以访问该键存储。确保将 *keystore_location* 中已生成的键存储移动 / 复制到 HTTPS 隧道 Servlet 可以访问的位置（请参阅第 207 页上的 “步骤 2：在 Web 服务器上部署 HTTPS 隧道 Servlet”）。

步骤 2：在 Web 服务器上部署 HTTPS 隧道 Servlet

在 Web 服务器上部署 HTTPS 隧道 Servlet 的常见方法有两种：

- 将其部署为支持 Servlet 2.1 或更早版本的 Web 服务器 Jar 文件
- 将其部署为支持 Servlet 2.2 或更新版本的 Web 服务器 Web 归档 (WAR) 文件

在任一种情况下，都应确保激活 Web 服务器的加密功能，确保客户机与代理之间的端到端通信是安全的。

部署为 Jar 文件

部署 MQ 隧道 Servlet 包括两个步骤，首先要使主机 Web 服务器可以访问相应的 Jar 文件，然后要对该 Web 服务器进行配置，使之在启动时装入该 Servlet。

隧道 Servlet Jar 文件 (imqServlet.jar) 包含 HTTPS 隧道 Servlet 所需的所有类，该文件位于以下目录中：

IMQ_HOME/lib （在 Solaris 操作系统上为 /usr/share/lib/imq）

可以使用任何支持 Servlet 2.x 的 Web 服务器装入此 Servlet。 Servlet 类名为：

```
com.sun.messaging.jmq.transport.
httptunnel.servlet.HttpsTunnelServlet。
```

Web 服务器必须能够访问 imqservlet.jar 文件。如果计划在不同的主机上运行 Web 服务器和代理，应该在 Web 服务器能够访问的位置放置 imqservlet.jar 文件的一个副本。

还需要配置 Web 服务器，使之在启动时装入此 Servlet （请参阅第 212 页上的 “实例：部署 HTTPS 隧道 Servlet”）。

确保 JSSE Jar 文件位于 Web 服务器上运行 Servlets 的类路径中。请参阅 Web 服务器文档，了解如何执行此操作。

配置 Web 服务器有一个重要的方面，即指定 HTTPS 隧道 Servlet 用来与代理建立安全连接所需的自签名证书的位置和密码。必须将您在第 206 页上的 “步骤 1：为 HTTPS 隧道 Servlet 生成自签名证书”中创建的键存储放置在 HTTPS 隧道 Servlet 可以访问的位置。

同时建议您禁用 Web 服务器的访问日志功能以提高性能。

部署为 Web 归档文件

要将 HTTPS 隧道 Servlet 部署为 WAR 文件，需要使用 Web 服务器提供的部署机制。 HTTPS 隧道 Servlet WAR 文件 (`imqhttps.war`) 位于以下目录中：

`IMQ_HOME/lib` （在 Solaris 操作系统上为 `/usr/share/lib/imq`）

WAR 文件包含一个部署描述符，该描述符包含 Web 服务器装入和运行 Servlet 所需的基本配置信息。

但是， `imqhttps.war` 文件的部署描述符无法知道您放置隧道 Servlet 所需的键存储的位置 （请参阅第 206 页上的 “步骤 1：为 HTTPS 隧道 Servlet 生成自签名证书”）。因此您需要在部署 `imqhttps.war` 文件之前编辑隧道 Servlet 部署描述符（一个 XML 文件），以指定键存储的位置。

步骤 3：配置 httpsjms 连接服务

默认情况下， MQ 3.0.1 代理没有启用 HTTPS 支持，因此需要重新配置代理以启用 httpsjms 连接服务。重新配置后，可以按照第 114 页上的 “启动代理”中介绍的步骤启动代理。

➤ 启用 httpsjms 连接服务的步骤

1. 打开位于以下位置的代理实例配置文件：

   `IMQ_VARHOME/instances/`*brokerName*`/props/config.properties`
   （在 Solaris 操作系统上为
   `/var/imq/instances/`*brokerName*`/props/config.properties`）

   其中 *brokerName* 是代理实例的名称。

2. 将 `httpsjms` 值添加到 `imq.service.activelist` 属性中：

   ```
   imq.service.activelist=jms,admin,httpsjms
   ```

启动时，代理将在其主机上查找运行的 Web 服务器和 HTTPS 隧道 Servlet。但是要访问远程隧道 Servlet，需要重新配置 servletHost 和 servletPort 连接服务属性。

还可以重新配置 pullPeriod 属性以提高性能。表 B-3 详细介绍了 httpsjms 连接服务配置属性。

表 B-3 httpsjms 连接服务属性

属性名称	说明
`imq.httpsjms.https.servletHost`	必要时可以更改此值，以指定运行 HTTPS 隧道 Servlet 的主机的名称 （主机名或 IP 地址）。（可以是远程主机或本地主机上的特定主机名。）默认值：`localhost`
`imq.httpsjms.https.servletPort`	需要更改此值，以指定代理用于访问 HTTPS 隧道 Servlet 的端口号。（如果更改了 Web 服务器上的默认端口，也必须对此属性做相应的更改。）默认值：`7674`
`imq.httpsjms.https.pullPeriod`	指定每一台客户机创建从代理提取消息的 HTTPS 请求的时间间隔 （以秒为单位）。如果值为零或为负数，客户机将始终使一个 HTTP 请求处于待处理状态，这样可以随时尽快地提取消息。如果客户机的数量过多，可能会对 Web 服务器资源产生较大压力，导致服务器停止响应。在这种情况下，应将 `pullPeriod` 属性设置为正秒数。此属性设置客户机 HTTP 传输驱动程序在发出下一个提取请求之前等待的时间。将此属性设置为正值将以牺牲客户机响应时间为代价来保持 Web 服务器资源的可用性。默认值：`-1`

步骤 4：配置 HTTPS 连接

客户机应用程序必须使用正确配置的连接工厂被管理对象，建立与代理之间的 HTTPS 连接。

但是，客户机还必须能够访问 Java 安全套接扩展 (JSSE) 提供的 SSL 库，并且必须具有一个根证书。SSL 库是随 JDK 1.4 一起提供的。如果您使用的是早期版本的 JDK，请参阅 “配置 JSSE”；否则，请转至 “输入根证书”。

解决上述问题后，可以继续配置 HTTPS 连接。

配置 JSSE

➤ **配置 JSSE 的步骤**

1. 将 JSSE Jar 文件复制到 `JRE_HOME/lib/ext` 目录中。

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

2. 静态添加 JSSE 安全服务供应商，

   `security.provider.`*n*`=com.sun.net.ssl.internal.ssl.Provider`

   即将以下内容添加到 `JRE_HOME/lib/security/java.security` 文件中：（其中 *n* 是安全服务供应商软件包的下一个可用优先编号）。

3. 如果您使用的不是 JDK1.4，则需要在启动客户机应用程序的命令中使用 `-D` 选项，以设置下列 JSSE 属性：

   ```
   java.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol
   ```

输入根证书

如果默认情况下签署您的 Web 服务器证书的 CA 的根用户证书不在信任数据库中，或者您使用的是专用 Web 服务器证书，则必须将该证书添加到信任数据库中。如果出现上述情况，请执行以下操作；否则，请转至 “设置连接工厂属性”。

如果证书保存在 *cert_file* 中，而您的键存储为 *trust_store_file*，请运行以下命令：

```
JRE_HOME/bin/keytool -import -trustcacerts
-alias alias_for_certificate -file cert_file
-keystore trust_store_file
```

显示以下问题时请选择 `YES`：Trust this certificate?

还需要在启动客户机应用程序的命令中使用 `-D` 选项，以指定下列 JSSE 属性：

```
javax.net.ssl.trustStore=trust_store_file
```

```
javax.net.ssl.trustStorePassword=trust_store_passwd
```

设置连接工厂属性

要实现 HTTPS 支持，需要设置以下连接工厂属性 （请参阅第 156 页上的 “连接工厂被管理对象”）：

- 将 `imqConnectionType` 属性设置为 `HTTP`

  通过 HTTPS 隧道 Servlet 而不是 HTTP 隧道 Servlet 进行部署和连接，可以获得安全的代理连接。但这两种方式都使用相同的连接类型。

- 将 `imqConnectionURL` 设置为 HTTPS 隧道 Servlet URL

  `https://`*hostName*`:`*port*`/imq/tunnel`

可以按照以下方法之一设置连接工厂属性：

- 在创建连接工厂被管理对象 （请参阅第 163 页上的 “添加连接工厂”）的 `imqobjmgr` 命令中使用 `-o` 选项，或在使用管理控制台 (`imqadmin`) 创建连接工厂时设置属性。
- 在启动客户机应用程序的命令中使用 `-D` 选项 （请参阅 《MQ *Developer's Guide*》）。
- 通过编程方式在客户机应用程序代码中创建被管理对象 （请参阅 《MQ *Developer's Guide*》）之后，使用 JMS API 调用设置其属性。

使用一个 Servlet 访问多个代理

即使正在运行多个代理，也无需配置多个 Web 服务器和多个 Servlet 实例。可以在并行运行的多个代理之间共享一个 Web 服务器和一个 HTTPS 隧道 Servlet 实例。要实现此目的，必须按照如下所示配置 `imqConnectionURL` 连接工厂属性：

`https://`*hostName:port/imq/tunnel*`?ServerName=`*hostName*`:`*brokerName*

其中 *hostName* 为代理主机名， *brokerName* 是您希望客户机访问的特定代理实例的名称。

要查看是否输入了正确的 *hostName* 和 *brokerName*，可以通过浏览器访问 Servlet URL，生成 HTTPS 隧道 Servlet 的状态报告。状态报告将列出 Servlet 正在访问的所有代理：

```
HTTPS tunnel servlet ready.
Servlet Start Time :Thu May 30 01:08:18 PDT 2002
Accepting TCP connections from brokers on port : 7674
Total available brokers = 2
Broker List :
   jpgserv:broker2
   cochin:broker1
```

使用 HTTP 代理

如果使用 HTTP 代理访问 HTTPS 隧道 Servlet：

- 将 `http.proxyHost` 系统属性设置为代理服务器主机名。
- 将 `http.proxyPort` 系统属性设置为代理服务器端口号。

可以通过在启动客户机应用程序的命令中使用 `-D` 选项来设置这些属性。

实例：部署 HTTPS 隧道 Servlet

本节说明如何在 Sun ONE Web Server 上将 HTTPS 隧道 Servlet 部署为 Jar 文件和 WAR 文件。使用的方法取决于 Sun ONE Web Server 的版本：如果 Sun ONE Web Server 不支持 Servlet 2.2 或更新版本，则不能处理 WAR 文件部署。

部署为 Jar 文件

下面的说明适用于使用基于浏览器的管理 GUI 在 Sun ONE Web Server FastTrack Edition 4.1 上部署 HTTP 隧道 Servlet 的情况。此过程包含以下通用步骤：

1. 添加 Servlet
2. 配置 Servlet 虚拟路径
3. 装入 Servlet
4. 禁用 Servlet 访问日志

以下小节对这些步骤进行了说明。通过使用 Web 浏览器访问 Servlet URL 可以验证是否成功部署了 HTTP 隧道 Servlet。它应该显示状态信息。

添加 *Servlet*

➤ **添加隧道 Servlet 的步骤**

1. 选择 “Servlets”选项卡。
2. 选择 “Configure Servlet Attribute”。
3. 在 “Servlet Name”字段中指定隧道 Servlet 的名称。
4. 将 “Servlet Code”（类名）字段设置为以下值：

```
com.sun.messaging.jmq.transport.
httptunnel.servlet.HttpsTunnelServlet
```

5. 在 “Servlet Classpath” 字段中输入 `imqservlet.jar` 的完整路径。例如：

   `IMQ_HOME/lib/imqservlet.jar`
   （在 Solaris 操作系统上为 `/usr/share/lib/imq/imqservlet.jar`）

6. 在 “Servlet args” 字段中输入必需参数和可选参数，如表 B-4 所示：

表 B-4 用于部署 HTTPS 隧道 Servlet Jar 文件的 Servlet 参数

参数	默认值	是否必需?	参阅
`keystoreLocation`	无	是	第 185 页上的表 8-8
`keystorePassword`	无	是	第 185 页上的表 8-8
`serverHost`	所有主机	无	第 209 页上的表 B-3
`serverPort`	`7674`	无	第 209 页上的表 B-3

参数之间用逗号分隔，例如：

`keystoreLocation=`*keystore_location*`,keystorePassword=`*keystore_password*`, servletPort=`*portnumber*

`serverHost` 和 `serverPort` 参数仅应用于 Web 服务器和代理之间的通信，仅当默认值存在问题时才需要进行设置。但是，在这种情况下，还必须相应地设置代理配置属性 （请参阅第 209 页上的表 B-3），例如：

```
imq.httpsjms.https.servletPort
```

配置 *Servlet* 虚拟路径 *(Servlet URL)*

➤ **配置隧道 Servlet 虚拟路径 (Servlet URL) 的步骤**

1. 选择 “Servlets” 选项卡。
2. 选择 “Configure Servlet Virtual Path Translation”。
3. 设置 “Virtual Path” 字段。

   例如，要使 URL 类似 `http://`*hostName*`:`*port*`/imq/tunnel`，请在 “Virtual Path” 字段中输入以下字符串：

   `/imq/tunnel`

4. 设置 “Servlet Name” 字段的值，使之与第 212 页上的 “添加 Servlet” 中的步骤 3 中设置的值相同。

装入 *Servlet*

➤ **在 Web 服务器启动时装入隧道 Servlet 的步骤**

1. 选择 “Servlets”选项卡。
2. 选择 “Configure Global Attributes”。
3. 在 “Startup Servlets”字段中输入与第 212 页上的 “添加 Servlet”中的步骤 3中的值相同的 Servlet 名称值。

禁用服务器访问日志

不是必须禁用服务器访问日志，但禁用服务器访问日志可以获得更好的性能。

➤ **禁用服务器访问日志的步骤**

1. 选择 “Status”选项卡。
2. 选择 “Log Preferences Page”。
3. 使用 “Log”客户机访问控制禁用日志。

部署为 WAR 文件

下面的说明适用于在 Sun ONE Web Server 6.0 SP2 上的部署。通过使用 Web 浏览器访问 Servlet URL 可以验证是否成功部署了 HTTPS 隧道 Servlet。它应该显示状态信息。

部署 HTTPS 隧道 Servlet 之前，请确保 JSSE Jar 文件位于 Web 服务器的类路径中。要达到此目的，最简单的方法是将 `jsse.jar`、`jnet.jar` 和 `jcert.jar` 文件复制到 `IWS60_TOPDIR/bin/https/jre/lib/ext` 中。

另外，部署 HTTPS 隧道 Servlet 之前，必须指定键存储密码并修改部署描述符，使其指向放置键存储文件的位置。

➤ **修改 HTTPS 隧道 Servlet WAR 文件的步骤**

1. 将 WAR 文件复制到临时目录中。

```
$ cp IMQ_HOME/lib/imqhttps.war /tmp
（在 Solaris 操作系统上为
$ cp /usr/share/lib/imq/imqhttps.war/tmp）
```

2. 使临时目录成为当前目录。

```
$ cd /tmp
```

3. 提取 WAR 文件的内容。

```
$ jar xvf imqhttps.war
```

4. 列出 WAR 文件的部署描述符。

```
$ ls -l WEB-INF/web.xml
```

5. 编辑 `web.xml` 文件，以提供正确的 `keystoreLocation` 和 `keystorePassword` 参数值（如果有必要，还包括 `serverPort` 和 `serverHost` 参数）。

6. 重新装入 WAR 文件的内容。

```
$ jar uvf imqhttps.war WEB-INF/web.xml
```

现在即可使用这个修改后的 `imqhttps.war` 文件部署 HTTPS 隧道 Servlet 了。（如果您担心会泄漏键存储密码，可以使用文件系统权限限制对 `imqhttps.war` 文件的访问。）

➤ 将 HTTPS 隧道 Servlet 部署为 WAR 文件的步骤

1. 在基于浏览器的管理 GUI 中，选择 “Virtual Server Class”选项卡。单击 “Manage Classes”。
2. 选择相应的虚拟服务器类名（例如 defaultclass）并单击 “Manage”按钮。
3. 选择 “Manage Virtual Servers”。
4. 选择相应的虚拟服务器名称并单击 “Manage”按钮。
5. 选择 “Web Applications”选项卡。
6. 在 “Deploy Web Application”上单击。
7. 为 “WAR File On”和 “WAR File Path”字段选择相应的值，以指向修改后的 `imqhttps.war` 文件（请参阅第 214 页上的 “修改 HTTPS 隧道 Servlet WAR 文件的步骤”）。
8. 在 “Application URI”字段中输入 `/imq`。
9. 输入要在其中部署 Servlet 的安装目录路径（通常位于 Sun ONE Web Server 的安装根目录下）。
10. 单击 “`OK`”。
11. 重新启动 Web 服务器实例。

Servlet 现在即被部署到以下位置：

`https://`*hostName*`:`*port*`/imq/tunnel`

客户机现在即可使用此 URL 连接至使用安全 HTTPS 连接的消息服务。

附录 C

使用代理作为 Windows 服务

本附录说明如何使用服务管理器 (imqsvcadmin) 实用程序安装、查询和删除作为 Windows 服务运行的代理。

将代理作为 Windows 服务运行

可以在安装 Sun™ ONE Message Queue (MQ) 3.0.1 时将代理安装为 Windows 服务，也可以在安装 MQ 3.0.1 之后，使用 imqsvcadmin 将代理安装为 Windows 服务。

将代理安装为 Windows 服务意味着它将在系统启动时启动，并在后台运行直到系统关闭。因此，无需使用 imqbrokerd 命令启动代理，除非您希望启动其它实例。要将启动选项传递给代理，可以使用 imqsvcadmin 命令的 -args 参数 （请参阅第 219 页上的表 C-2）并准确指定您希望 imqbrokerd 命令使用的选项（请参阅第 114 页上的 “启动代理”）。用来控制代理操作的命令仍然是 imqcmd。

当作为 Windows 服务运行时，任务管理器将代理列为两个可执行的进程。第一个是本地 Windows 服务包装程序 imqbrokersvc.exe。第二个是实际运行代理的 Java 运行时。

一次只能安装一个代理并作为 Windows 服务运行。

服务管理器实用程序 (imqsvcadmin)

可以使用服务管理器实用程序 (imqsvcadmin) 安装、查询和删除作为 Windows 服务运行的代理。本节介绍了 imqsvcadmin 命令的基本语法，提供了一个子命令列表，概述了 imqsvcadmin 命令选项，并说明了如何使用这些命令执行特定的任务。

命令语法

imqsvcadmin 命令的一般语法如下：

```
imqsvcadmin subcommand [options]
imqsvcadmin -h
```

请注意，如果指定 -v、 -h 或 -H 选项，将不会执行命令行中指定的其它子命令。例如，输入以下命令将显示帮助信息，而不是执行 query 子命令。

```
imqsvcadmin query -h
```

imqsvcadmin 子命令

表 C-1 列出了 MQ 服务管理器实用程序 (imqsvcadmin) 包含的子命令：

表 C-1 imqsvcadmin 子命令

子命令	说明
install	安装服务和特定的启动选项。
query	显示 imqsvcadmin 命令的启动选项。启动选项包括服务的启动方式（手动或自动）、服务的位置、 Java 运行时的位置以及启动时传递给代理的参数值。
remove	删除服务。

imqsvcadmin 选项概述

表 C-2 列出了 imqsvcadmin 命令的选项。有关如何使用它们的讨论，请参阅后面针对具体任务的各节。

表 C-2 imqsvcadmin 选项

选项	说明
-h	显示使用帮助。不执行命令行上的命令。
-javahome *path*	指定替代 Java 2 兼容 JDK 的路径。默认使用附带的运行时。 实例：imqsvcadmin -install -javahome d:\jdk1.4
-jrehome *path*	指定替代 Java 2 兼容 JRE 的路径。 实例：imqsvcadmin -install -jrehome d:\jre\1.4
-vmargs *arg* [[*arg*] . . .]	指定传递给正在运行代理服务的 Java VM 的其它参数。（也可以在 Windows 服务控制面板的 “启动参数” 字段中指定这些参数。） 实例：-vmargs "-Xms16m -Xmx128m"
-args *arg* [[*arg*] . . .]	指定传递给代理服务的其它命令行参数。有关 imqbrokerd 选项的说明，请参阅第 114 页上的 “启动代理”。 （也可以在 Windows 服务控制面板的 “启动参数” 字段中指定这些参数。）实例： imqsvcadmin -install -args "-passfile d:\imqpassfile"

使用 -javahome、 -vmargs 和 -args 选项指定的信息存储在 Windows 注册表关键字 JavaHome、 JVMArgs 和 ServiceArgs 下，路径为

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet
     \Services\iMQ_Broker\Parameters
```

删除代理服务

删除代理服务之前，应使用 imqcmd shutdown bkr 命令关闭代理。然后使用 imqsvcadmin remove 命令删除服务并重新启动计算机。

重新配置代理服务

要重新配置服务，首先要删除服务，然后重新安装并使用 -args 参数指定不同的启动选项。

使用替代 Java 运行时

可以使用 -javahome 或 -jrehome 选项指定替代 Java 运行时的位置。（也可以在 Windows 服务控制面板的 “启动参数”字段中指定这些选项。）请注意，“启动参数”字段将反斜杠 (\) 作为转义符进行处理，因此如果要使用它作为路径分隔符，必须键入两次，例如 -javahome d:\\jdk1.3。

查询代理服务

要确定代理服务的启动选项，请使用 imqsvcadmin 命令的 -q 选项。

```
imqsvcadmin -query

Service iMQ_Broker is installed.
Display Name:iMQ_Broker
Start Type:Manual
Binary location:c:\Program Files\Sun Microsystems\
                  Message Queue 3.0\bin\imqbrokersvc
JavaHome:c:\j2sdk1.4.0
Broker Args:-passfile d:\imqpassfile
```

疑难解答

如果试图启动服务时收到错误消息，可以通过执行以下操作查看记录的错误事件。

➤ **查看记录的服务错误事件的步骤**

1. 启动事件查看器。
2. 找到 “日志” > “应用程序”。
3. 选择 “查看” > “刷新”，以查看错误事件。

MQ 数据的位置

Sun™ ONE Message Queue (MQ) 使用了多种数据，每种数据都存储在不同的位置，如表 D-1 所示。

表 D-1 MQ 3.0 数据的位置

数据类型	数据的位置
代理实例配置属性	IMQ_VARHOME/instances/*brokerName*/props/config.properties （在 Solaris 操作系统中为 /var/imq/instances/*brokerName*/props/config.properties）
持久性存储器 （消息、目标、长期订阅、事务）	IMQ_VARHOME/instances/*brokerName*/filestore/ （在 Solaris 操作系统中为 /var/imq/instances/*brokerName*/filestore/） 或支持 JDBC 的数据存储
代理实例日志文件	IMQ_VARHOME/instances/*brokerName*/log/ （在 Solaris 操作系统中为 /var/imq/instances/*brokerName*/log/）
被管理对象 （对象存储）	您选择的本地目录 或 LDAP 服务器
安全性：用户系统信息库	IMQ_HOME/etc/passwd （在 Solaris 操作系统中为 /etc/imq/passwd） 或 LDAP 服务器
安全性：访问控制文件	IMQ_HOME/etc/accesscontrol.properties （在 Solaris 操作系统中为 /etc/imq/accesscontrol.properties）

表 D-1 MQ 3.0 数据的位置 （续）

数据类型	数据的位置
安全性：密码文件	IMQ_HOME/etc/*passfileName*
	（在 Solaris 操作系统中为 /etc/imq/*passfileName*）
安全性：代理的键存储	IMQ_HOME/etc/keystore
	（在 Solaris 操作系统中为 /etc/imq/keystore）

MQ 接口的稳定性

Sun™ ONE Message Queue (MQ) 使用的许多接口可能对管理员自动化管理任务很有帮助。表 E-1 按照稳定性 （即本产品的后续版本中对其进行更改的可能性）列出了这些接口。第 225 页上的表 E-2 介绍了分类方案。

表 E-1 MQ 3.0.1 接口的稳定性

接口	分类
`imqbrokerd` 命令行接口	开发中
`imqadmin` 命令行接口	不稳定
`imqcmd` 命令行接口	开发中
`imqdbmgr` 命令行接口	不稳定
`imqkeytool` 命令行接口	开发中
`imqobjmgr` 命令行接口	开发中
`imqusermgr` 命令行接口	不稳定
`imqobjmgr` 命令文件	开发中
`imqbrokerd` 命令	稳定
`imqadmin` 命令	不稳定
`imqcmd` 命令	稳定
`imqdbmgr` 命令	不稳定
`imqkeytool` 命令	稳定
`imqobjmgr` 命令	稳定
`imqusermgr` 命令	不稳定
JMS API (`javax.jms`)	标准
JAXM API (`javax.xml`)	标准

表 E-1 MQ 3.0.1 接口的稳定性 （续）

接口	分类
被管理对象 API (com.sun.messaging)	开发中
imq.jar 位置和名称	稳定
jms.jar 位置和名称	开发中
imqbroker.jar 位置和名称	专用
imqutil.jar 位置和名称	专用
imqadmin.jar 位置和名称	专用
imqservlet.jar 位置和名称	开发中
imqhttp.war 位置和名称	开发中
imqhttps.war 位置和名称	开发中
imqxm.jar 位置和名称	开发中
jaxm-api.jar 位置和名称	开发中
saaj-api.jar 位置和名称	开发中
saaj-impl.jar 位置和名称	开发中
activation.jar 位置和名称	开发中
mail.jar 位置和名称	开发中
dom4j.jar 位置和名称	专用
fscontext.jar 位置和名称	不稳定
imqbrokerd、imqadmin、imqcmd、imqdbmgr、imqkeytool、imqobjmgr 和 imqusermgr 的输出	不稳定
代理日志文件的位置和内容格式	不稳定
密码文件	不稳定
accesscontrol.properties	不稳定

表 E-2 接口稳定性分类方案

分类	说明
专用	用户无法直接使用。在任何发行版中都可能对其进行改进或将其删除。
开发中	可供用户使用。可能会在主版本 （例如 3.0 和 4.0）或次版本（例如 3.1 和 3.2）中做不兼容更改。所有更改都将慎重逐步地进行。虽然我们会尽量保证所有更改都是兼容的，但不能保证做到这一点。
稳定	可供用户使用。只在主版本 （例如 3.0 和 4.0）中做不兼容更改。
标准	可供用户使用。这些接口根据官方认可的标准进行定义，由标准组织控制。很少对这些接口做不兼容更改。
不稳定	可供用户使用。可能会在主版本 （例如 3.0 和 4.0）或次要版本 （例如 3.1 和 3.2）中做不兼容更改。用户需注意，在以后的发行版中可能会删除这些接口，也可能以一种不兼容的方式更改这些接口。建议用户不要在不稳定的接口上创建显式相关性。

词汇表

本词汇表解释了使用 Sun™ ONE Message Queue (MQ) 时可能会遇到的术语和概念。

JMS 客户机 与其它 JMS 客户机进行交互的应用程序（或软件组件），使用消息服务与其它客户机交换消息。

JMS 提供者 实现消息传送系统的 JMS 接口并添加完整产品所需的管理和控制功能的产品。

JMS（Java 消息服务） 定义 JMS 客户机如何访问消息服务设备的接口和语义的标准集。这些接口为 Java 程序创建、发送、接收和阅读消息提供了一个标准方法。

MQ 客户机运行时 在 JMS 客户机与 MQ 消息服务器之间提供通信接口的软件。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。

MQ 消息服务器 为 MQ 消息传送系统提供传送服务的软件，传送服务包括 JMS 客户机的连接、消息的路由和传送、持久性、安全性以及日志。消息服务器负责维护从 JMS 客户机接收消息以及向使用消息的客户机发送消息的物理目标。

被管理对象 由管理员创建并由一个或多个 JMS 客户机使用的预配置的 MQ 对象，例如连接工厂或目标。

被管理对象的使用使得 JMS 客户机能够独立于提供者，也就是说，它使 JMS 客户机不再专属于提供者。管理员将这些对象置于 JNDI 名称空间， JMS 客户机可以使用 JNDI 查找对它们进行访问。

传送策略 队列如何在注册了多个消息用户时路由消息的规范。策略包括：单向、故障切换和循环。

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。

代理　管理消息路由、传送、持久性、安全性和日志的 MQ 实体，它为管理员监视和调节性能和资源使用提供了一个接口。

点对点传送模型　生成方将消息发送给特定队列，用户从指定用来接收消息的队列中提取消息。一个消息只能传送给一个消息用户。

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。队列可用作生成方和用户之间的中转站。

发布/订阅传送模型　发布人和订户通常是匿名的，可以动态发布或订阅某个主题。系统将某个主题的多个发布人发布的消息分发给多个订户。

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。

客户机标识符　用于将 JMS 客户机与其对象相关联的标识符，其状态由 MQ 消息服务器代表 JMS 客户机来维护。

客户机运行时　请参阅 MQ 客户机运行时。

连接　1) 到 MQ 消息服务器的活动连接。可以是队列连接或主题连接。2) 使用连接底层 MQ 消息服务器来生成和使用消息的会话工厂。

连接工厂　客户机创建 MQ 消息服务器连接时使用的被管理对象。可以是 QueueConnectionFactory 对象或 TopicConnectionFactory 对象。

目标　MQ 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用户。此物理目标由被管理对象标识和封装， JMS 客户机使用被管理对象指定生成和 / 或使用消息的目标。

配置文件　一个或多个文本文件，包含配置代理所需的 MQ 设置。配置文件中的属性是特定于某个实例或与某个群集相关的。

群集　两个或多个互连的代理，以串联模式提供消息传送服务。

生成　将消息传送给客户机运行时，然后传送给目标。

生成方　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，生成方为发件人 (QueueSender) ；在发布 / 订阅传送模型中，生成方为发布人 (TopicPublisher)。

使用　消息用户从目标接收消息的过程。

事务　不可细分的工作单位，要么整个完成，要么整个回滚。

授权　消息服务确定某个用户是否可以访问消息服务资源 （例如连接服务或目标）的过程。

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

消息　JMS 客户机使用的异步请求、报告或事件。消息包括标头 （可以在其中添加其它字段）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。

消息服务　请参阅 MQ 消息服务器。

消息选定器　用户基于 JMS 消息标头中的属性值 （选定器）选择消息的方法。消息服务基于消息选定器中的标准对消息进行过滤和路由。

异步通信　一种通信模式，在这种模式下，消息的发送方不需要等到发送方法返回即可继续其它工作。

用户　由用于从目标接收消息的会话创建的对象 (MessageConsumer)。在点对点传送模型下，用户为收件人或浏览器 （QueueReceiver 或 QueueBrowser）；在发布 / 订阅传送模型下，用户即为订户 (TopicSubscriber)。

用户组　JMS 客户机用户所属的组，用于授权对 MQ 消息服务器资源 （例如连接或目标）的访问。

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。

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。

索引

英文

admin 连接服务 46
config.properties 文件 108
HTTP 46， 140
HTTP 传输驱动程序 197
HTTP 代理 197
HTTP 连接
  多个代理 202
  请求时间间隔 201
  隧道 Servlet，**请参阅** HTTP 隧道 Servlet
  支持 197
HTTP 隧道 Servlet 197， 203
HTTP 支持体系结构 197
httpjms 连接服务 46， 140
HTTPS 连接
  多个代理 211
  请求时间间隔 209
  隧道 Servlet，**请参阅** HTTPS 隧道 Servlet
  支持 197
HTTPS 隧道 Servlet 188， 197
HTTPS 支持体系结构 197
httpsjms 连接服务 46， 140
imq.accesscontrol.enabled 属性 58， 110
imq.accesscontrol.file.filename 属性 58， 110
imq.authentication.basic.user_repository
  属性 58， 110
imq.authentication.client.response.timeout
  属性 58， 110
imq.authentication.type 属性 57， 110
imq.autocreate.queue 属性 64， 110， 138
imq.autocreate.topic 属性 64， 110， 138
imq.cluster.brokerlist 属性 68， 118
imq.cluster.hostname 属性 68， 119
imq.cluster.masterbroker 属性 68， 118
imq.cluster.port 属性 68， 118
imq.cluster.url 属性 68， 110， 118， 138
imq.httpjms.http.pullPeriod 属性 201
imq.httpjms.http.servletHost 属性 201
imq.httpjms.http.servletPort 属性 201
imq.httpsjms.https.pullPeriod 属性 209
imq.httpsjms.https.servletHost 属性 209
imq.httpsjms.https.servletPort 属性 209
imq.keystore.file.dirpath 属性 185
imq.keystore.file.name 属性 185
imq.keystore.password 属性 185， 189
imq.log.console.output 属性 61， 110
imq.log.console.output.stream 属性 61， 110
imq.log.console.syslog.facility 属性 61， 111
imq.log.console.syslog.identity 属性 61， 111
imq.log.console.syslog.logconsole 属性 61， 111
imq.log.console.syslog.logpid 属性 61， 111
imq.log.console.syslog.output 属性 61， 111
imq.log.file.dirpath 属性 61， 111
imq.log.file.filename 属性 61， 111
imq.log.file.output 属性 61， 111

imq.log.file.rolloverbytes 属性 61， 111， 138
imq.log.file.rolloversecs 属性 61， 111， 138
imq.log.level 属性 60， 111， 138
imq.message.expiration.interval 属性 52， 111
imq.message.max_size 属性 52， 111， 138
imq.metrics.enabled 属性 60， 111
imq.metrics.interval 属性 60， 111
imq.passfile.dirpath 属性 58， 111
imq.passfile.enabled 属性 58， 111
imq.passfile.name 属性 58， 111
imq.persist.file.destination.file.size 属性 54， 111
imq.persist.file.message.cleanup 属性 55， 111
imq.persist.file.message.fdpool.limit 属性 55， 112
imq.persist.file.message.filepool.cleanratio 属性 55， 112
imq.persist.file.message.filepool.limit 属性 55， 112
imq.persist.file.sync.enabled 属性 55， 112
imq.persist.jdbc.brokerid 属性 193
imq.persist.jdbc.closedburl 属性 194
imq.persist.jdbc.createdburl 属性 194
imq.persist.jdbc.driver 属性 193
imq.persist.jdbc.needpassword 属性 194
imq.persist.jdbc.opendburl 属性 193
imq.persist.jdbc.password 属性 189， 194
imq.persist.jdbc.user 属性 194
imq.persist.store 属性 54， 112， 193
imq.portmapper.port 属性 48， 112， 138
imq.queue.deliverypolicy 属性 64， 112， 138
imq.redelivered.optimization 属性 52， 112
imq.*resource_state*.threshold 属性 52， 112
imq.service.activelist 属性 48， 112
imq.*service_name*.accesscontrol.enabled 属性 58， 112
imq.*service_name*.accesscontrol.file.filename 属性 58， 112
imq.*service_name*.authentication.type 属性 57， 112
imq.*service_name*.max_threads 属性 48， 112
imq.*service_name*.min_threads 属性 48， 113
imq.*service_name.protocol_type*.hostname 属性 49， 113
imq.*service_name.protocol_type*.port 属性 49， 113
imq.*service_name*.threadpool_model 属性 48， 113
imq.shared.connectionMonitor_limit 属性 48， 113
imq.system.max_count 属性 52， 113， 138
imq.system.max_size 属性 52， 113， 138
imq.transaction.autorollback 属性 52， 113， 150
imq.user_repository.ldap.base 属性 177
imq.user_repository.ldap.gidattr 属性 177
imq.user_repository.ldap.grpbase 属性 177
imq.user_repository.ldap.grpfiltler 属性 177
imq.user_repository.ldap.grpsearch 属性 177
imq.user_repository.ldap.memattr 属性 177
imq.user_repository.ldap.password 属性 177， 189
imq.user_repository.ldap.principal 属性 176
imq.user_repository.ldap.server 属性 176
imq.user_repository.ldap.ssl.enabled 属性 178
imq.user_repository.ldap.timeout 属性 178
imq.user_repository.ldap.uidattr 属性 177
imq.user_repository.ldap.usrfilter 属性 177
IMQ_HOME 目录变量 20
IMQ_JAVAHOME 目录变量 21
IMQ_VARHOME 目录变量 20
imqAckOnAcknowledge 属性 157
imqAckOnProduce 属性 157
imqAckTimeout 属性 157
imqbrokerd 命令
  关于 78
  命令语法 114
  使用 114
  选项 115
imqBrokerHostName 属性 157
imqBrokerHostPort 属性 157
imqBrokerServicePort 属性 157
imqcmd 命令
  安全连接到代理 188
  代理的安全连接 133
  关于 79

连接到代理 133
命令语法 130
目标管理 143
使用 130，153，171
事务管理 147
选项 132
子命令 130
imqConfiguredClientID 属性 157
imqConnectionType 属性 157
imqConnectionURL 属性 157
imqdbmgr 命令
关于 79
命令语法 195
选项 196
子命令 196
imqDefaultPassword 属性 157
imqDefaultUsername 属性 157
imqDestinationDescription 属性 73，158
imqDestinationName 属性 73，158
imqDisableSetClientID 属性 157
imqFlowControlCount 属性 157
imqFlowControlIsLimited 属性 157
imqFlowControlLimit 属性 157
imqJMSDeliveryMode 属性 158
imqJMSExpiration 属性 158
imqJMSPriority 属性 158
imqkeytool 命令
关于 79
命令语法 184，206
使用 184，206
imqLoadMaxToServerSession 属性 157
imqobjmgr 命令
关于 79
命令语法 153
选项 154
子命令 153
imqOverrideJMSDeliveryMode 属性 158
imqOverrideJMSExpiration 属性 158
imqOverrideJMSHeadersToTemporaryDestinations
属性 158
imqOverrideJMSPriority 属性 158
imqQueueBrowserMaxMessagesPerRetrieve
属性 157
imqQueueBrowserRetrieveTimeout 属性 157
imqReconnect 属性 157
imqReconnectDelay 属性 157
imqReconnectRetries 属性 157
imqSetJMSXAppID 属性 157
imqSetJMSXConsumerTXID 属性 157
imqSetJMSXProducerTXID 属性 157
imqSetJMSXRcvTimestamp 属性 157
imqSetJMSXUserID 属性 157
imqSSLIsHostTrusted 属性 158
imqsvcadmin 命令
关于 79
命令语法 218
使用 218
选项 219
子命令 218
imqusermgr 命令
关于 79
密码 174
命令语法 171
选项 172
用户名 174
子命令 172
J2EE 应用程序
EJB 规范 34
JMS，和 34
消息驱动 Bean，请参见消息驱动 Bean
JDBC 支持
关于 54
驱动程序 191，193
设置 191
JDK
指定路径 116
指定路径的选项 132，154，219
JMS
编程模型 32
规范 31
消息结构 31

JMS 规范 23， 25
jms 连接服务 46， 140
JNDI
  MQ 支持 26
  被管理对象，和 33， 36
  查找 71， 73， 95， 155
  初始上下文 159
  地址 （提供者 URL） 159
  对象存储 79， 152
  对象存储属性 156， 159
  消息驱动 Bean，和 35
JRE，指定路径 116
LDAP 服务器
  对象存储属性 159
  用户系统信息库访问 176
MDB 请参见消息驱动 Bean
SOAP 26
SSL
  服务，设置 169
  关于 57
  加密，和 183
  连接服务，和 46， 140
  通过 HTTP 188
  通过 TCP/IP 184
ssladmin 连接服务 46， 140
  配置 186
  设置 184
ssljms 连接服务 46， 140
  配置 186
  设置 184
TCP 46， 140
TLS 46， 140
Windows 服务，运行代理 217
XA 连接工厂
  关于 40
  请参见连接工厂被管理对象
XA 资源管理器，请参见分布式事务

A

安全
  对象存储 159
安全套接字层标准，请参见 SSL
安全性
  管理器，请参见安全性管理器
  加密，参阅加密
  授权，参阅授权
  验证，请参阅验证
安全性管理器
  关于 55
  属性 57
  作为代理组件 45

B

版本，产品
  关于 26
  平台 26
  企业 27
被管理对象
  XA 连接工厂，请参见连接工厂被管理对象
  查询 167
  查找名称 154
  队列，参阅队列
  对象存储，参阅对象存储
  更新 168
  关于 33， 71
  类型 33， 71， 152
  连接工厂，请参见连接工厂被管理对象
  列表 166
  目标，请参见目标被管理对象
  删除 165
  属性 156
  所需的信息 155
  提供者无关 71
  主题，参阅主题
编程域 37

C

测量
  关于 59
测量信息
  报告时间间隔 116
  概述 127
插入的持久性
  关于 54
  设置 191
长期订户，**请参见**长期订阅
长期订阅
  ClientID，和 38
  关于 37
  列出 146
  清理消息 146
  销毁 146
持久性
  JDBC，**参阅** JDBC 持久性
  插入的，**请参见**插入的持久性
  持久性管理器，**请参见**持久性管理器
  传送模式，**请参见**传送模式
  内置的 53
  数据存储，**请参见**数据存储
持久性管理器
  JDBC 数据存储 193
  插入的持久性，和 191
  关于 52
  属性 54
  数据存储，**请参见**数据存储
  作为代理组件 45
持久性消息 38
重新传送标志 50
传输协议
  HTTP 46，140
  TCP 46，140
  TLS 46，140
传送，可靠 38
传送模式
  持久性 38
  非持久性 38

D

代理
  HTTP 支持 199
  httpjms 连接服务属性 201，209
  HTTPS，支持 206
  JDBC 支持，**参阅** JDBC 支持
  Windows 服务，运行 217
  安全性管理器，**请参见**安全性管理器
  测量，**请参见**测量
  查询 136，137
  持久性管理器，**请参见**持久性管理器
  重新启动 53，135
  从故障中恢复 53
  多代理群集，**请参见**代理群集
  访问控制，**请参见**授权
  更新 136
  关闭 135
  关于 44
  互连，**请参见**代理群集
  恢复 135
  控制 135
  连接到 133
  连接到一起 120
  连接服务，**请参见**连接服务
  列出服务 140
  配置文件，**请参见**配置文件
  启动 114
  启动基于 SSL 的服务 186
  确认 (Ack) 50
  群集，**请参见**代理群集
  任务 44
  日志记录，**请参见**日志记录器
  实例名称 116
  实例配置文件 110
  属性 138
  系统资源 51
  消息路由，**请参见**消息路由器
  消息容量 52
  暂停 135
  主管代理 66
  自动创建目标属性 64
代理群集
  重新启动代理 121

连接代理 120
配置更改记录 66
配置属性 67，68，118
群集配置文件 67，118
设置属性 119
使用原因 64
体系结构 65
添加代理到 120
信息传播 66
选项以指定 115
在仅用于开发的环境中 67
主管代理 66，67
代理实例，**请参见**代理
点对点传送 37
订阅
长期订阅 ID 132
管理长期订阅 146
销毁长期订阅 147
端口，动态分配 47
端口映射器
端口分配 48，117
关于 47
队列 62
传送策略，**请参见**队列传送策略
属性 144
添加被管理对象 164
自动创建 64，110
队列传送策略
单一 62
故障切换 62
关于 62
属性 144
循环（共享） 62
队列目标，**请参见**队列
对象存储
LDAP 服务器 152
LDAP 服务器属性 159
关于 152
位置 221
文件系统存储 152
文件系统存储属性 159

F

发布 / 订阅传送 37
防火墙 197
访问规则 180
访问控制属性文件，**参阅**访问控制文件
访问控制文件
版本 179
访问规则 180
格式 179
使用 178
位置 178，221
分布式事务
XA 资源管理器 39，147
关于 39
请参见 XA 连接工厂
服务类型
ADMIN 46
NORMAL 46

G

工具，管理，**请参见**管理工具。
管理工具
关于 77
管理控制台 77
命令行实用程序 78
管理连接服务 140
管理任务
开发环境 75
生产环境 75

H

环境变量，**请参见**目录变量
会话
介绍 33
确认选项 39
事务 39

J

基于 SSL 的服务
  启动 186
基于 SSL 的连接服务
  设置 184
加密
  关于 57
  基于 SSL 的服务，和 183
  密钥工具，和 57
监听器 33，34
简单对象访问协议，**请参见** SOAP
键存储
  属性 185
  文件 185，207

K

可靠传送 38
可移植性，**请参见**提供者无关
客户机
  编程模型 32
  标识符 (ClientID) 38
  应用程序，**请参见**客户机应用程序
  运行时 69
客户机应用程序
  提供者无关 36
  系统属性，和 73
控制消息 50
口令
  编码 57

L

连接
  重新连接 157
  重新连接尝试 157
  重新连接延迟 157
  介绍 32
连接服务
  admin 46
  HTTP，**请参阅** HTTP 连接
  httpjms 46，140
  HTTPS，**请参阅** HTTPS 连接
  httpsjms 46，140
  jms 46，140
  ssladmin 46，140，184
  ssljms 46，140，184
  查询 139，141
  端口映射器，**请参见**端口映射器
  访问控制 58
  服务类型 46
  更新 139，141
  关于 45
  管理 140
  恢复 139，143
  基于 SSL 185
  静态端口 48
  连接类型 46
  启动时激活 48
  属性 48
  线程池管理器 47
  线程分配 142
  相关命令 139
  暂停 139，143
连接工厂被管理对象
  ClientID，和 38
  JNDI 查找 33
  覆盖 73
  关于 72
  介绍 32
  属性 72，156，157
  添加 163
临时目标 64，145
路由，**参见**消息路由器

M

密码
  JDBC 189

LDAP 189
SSL 键存储 185，189
SSL 密钥存储 116
密码文件，**参阅**密码文件
命名惯例 174
默认值 157
密码文件
代理配置属性 58
命令行选项 116
使用 189
位置 189，222
密码文件，**参阅**密码文件
密钥对
重新生成 185
生成 184
密钥工具 57
命令行实用程序
imqcmd 79，130
imqdbmgr 79，196
imqkeytool 79，184，206
imqobjmgr 79，153
imqsvcadmin 79，218
imqusermgr 79，172
关于 78
基本语法 79
通用选项 80
命令行语法 79
命令选项 80
目标
创建 144
队列，**请参见**队列
访问控制 181
更新属性 143，145
管理 143
介绍 44
列出 143
临时 64，145
清理消息 143，146
实际 62
属性 144
相关信息 143，145
销毁 143，144
主题，**请参见**主题
自动创建 63，182
目标被管理对象 72
介绍 33
属性 158
目录变量
IMQ_HOME 20
IMQ_JAVAHOME 21
IMQ_VARHOME 20

N

内存管理 51
内置的持久性 53

P

配置更改记录 66
配置文件
config.properties 108
安装 108
编辑 110
覆盖已设置的值 108
默认 108
实例 108，119，138，221
位置 108，221
平台版 26

Q

企业版 27
权限
MQ 操作 56
访问控制属性文件 56，179
管理服务 57
计算 180
键存储 206

密码文件 189
嵌入式数据库 192
数据存储 54
用户系统信息库 171
确认
传送，对象 50
代理 50，157
等待周期 157
关于 39，50
客户机 50
事务，和 51
群集，**请参见**代理群集
群集配置文件 67

R

日志记录，**请参见**日志记录器
日志记录器
测量信息 60，126
重定向日志消息 125
更改配置 124
关于 59
归档文件 60
级别 59，60，116
默认配置 123
输出通道 60，125
属性 60
消息格式 123
写入到控制台 61，117
种类 59
转移标准 125
作为代理组件 45
日志文件
默认位置 60，221
转移条件 61
容器
EJB 35
MDB 35

S

生成方 33
实例配置文件，**请参见**配置文件
使用方 33
事务
分布式，**请参见**分布式事务
关于 39
管理 147
回滚 148
确认，和 51
提交 148
信息 148
授权
关于 56
管理 178
请参见访问控制文件
用户组 57
输入文件 160
属性
httpjms 连接服务 201
httpsjms 连接服务 209
安全性 57
持久性 54
代理，更新 138
代理实例配置 110
键存储 185
连接服务 48
群集配置 68
日志记录器 60
消息路由器 52
与 JDBC 相关的 193
与 LDAP 相关 176
自动创建 64
数据，MQ，位置 221
数据存储
JDBC 可访问 54
重置 117
关于 52
位置 221
文本文件 53

T

提供者无关
被管理对象 71
关于 36
图表显示
消息生成方将消息发送至消息服务，然后消息服务将这些消息转发至使用方。 30

X

系统日志 60，125
系统属性，设置 73
线程池管理器
共享线程 48
关于 47
专用线程 48
消息
SOAP 26
持久性 38，51，52
重新传送 50
传送模式，**请参见**传送模式
传送模型 31，37
代理限制 52
点对点传送 37
发布/订阅传送 37
过滤 41
过期收回 52
监听器 33，70
结构 31
介绍 31
可靠传送 38
控制 50
路由和传送 49
排列优先级 41
排序 41
清理目标上的 143
确认 50，157
生成 69
使用 70
限制 144
消息传送模型 31，37
消息传送系统
MQ 的体系结构 44
体系结构 30
消息服务 30
消息服务 30
消息服务器
多代理，**请参见**代理群集 64
关于 44
消息监听器，**请参见**监听器
消息路由器
关于 49
属性 52
作为代理组件 45
消息驱动 Bean
MDB 容器 35
部署描述符 35
关于 34
应用服务器支持 36
消息生成方，**请参见**生成方
消息使用方，**请参见**使用方
协议，**参见**传输协议
协议，**请参见**传输协议
协议类型
HTTP 46，140
TCP 46，140
TLS 46，140
性能，可靠性，和 40
许可证
MQ 版本 26
装入 116

Y

验证
关于 56
管理 170
应用服务器 36
用户名
格式 174

默认 170
默认值 157
用户系统信息库
LDAP 服务器 176
关于 55
管理 174
类型 58
平台相关性 171
填充 174
位置 170，221
文本文件 170
用户状态 173
用户组 173
用户组
关于 56
默认值 57
删除指定 173
预定义 173
域 37

Z

证书 184，206
主管代理 66，67
主题
关于 37
属性 144
添加被管理对象 164
自动创建 64，110
作为物理目标 63
主题目标，**请参见**主题
自动创建目标
关于 63
属性 64
自签名证书 184，206
组件
EJB 34
MDB 34

Z