JBoss 企业级应用程序平台 4.3
JBoss Messaging 用户指南

适用于 JBoss 企业级应用程序平台 4.3
版 4.3.0

Tim Fox
Ovidiu Feodorov
Madhu Konda
Tyronne Wickramarathne
Adrian Brock
Alex Fu
Luc Texier
David Boeren
Clebert Suconic
Sergey Koshcheyev
Jay Howell
Aaron Walker
Rajdeep Dua
Juha Lindfors
Scott Stark
Mike Clark
Andy Taylor
Ron Sigal
Tom Elrod
Alexey Loubyansky
Jay Howell

David Boeren Mike Clark
Tyronne Wickramarathne Mark Little
Pete Bennett

JBoss 企业级应用程序平台 4.3 JBoss Messaging 用户指南

适用于 JBoss 企业级应用程序平台 4.3
版 4.3.0

Tim Fox

Clebert Suconic

Andy Taylor

Ovidiu Feodorov

Sergey Koshcheyev

Ron Sigal

Madhu Konda

Jay Howell

Tyronne Wickramarathne

Aaron Walker

Adrian Brock

Rajdeep Dua

Tom Elrod

Alex Fu

Juha Lindfors

Alexey Loubyansky

Luc Texier

Scott Stark

Jay Howell

David Boeren

Mike Clark

Tyronne Wickramarathne

Mark Little

Pete Bennett

摘要

本用户指南介绍了使用 JBoss 企业级应用程序平台 4.3 里 JBoss Messaging 1.4 的相关信息

目录

第 1 章 关于本书

本书的目的是使你对 JBoss Messaging 有大概的了解。它解释了 JBoss Messaging 的重要功能以及为企业级 Java 应用程序提供高性能和稳定消息系统的能力。

JBoss Messaging 是 JBoss 里最新的企业级消息系统，它提供了出众的性能、可靠性和可扩充性以及高吞吐量和低延迟。JBoss Messaging 作为 JBoss 企业级应用程序平台 4.3 里缺省的 JMS 提供者已经替换了 JBossMQ。现在的版本是 JBoss Messaging 1.4。对于演示 JBoss Messaging 的例程，你应该使用这个或更高版本。虽然本书不会教授你企业级消息系统的基础知识，但你可以熟悉 JBoss Messaging 的功能和不同的配置。

第 2 章 简介

JBoss Messaging 提供一个开源的、基于业界标准的消息平台，它是红帽关于企业级消息系统策略的一个集成部分。它完全重写了旧的 JBoss JMS 提供者 － JBossMQ，并为单节点和群集环境提供了更好的性能。它也使用了更好的模块化架构，方便以后添加新的功能。

注意

JBoss Messaging 1.4 是 JBoss 企业级应用程序平台 4.3 里的缺省 JMS 提供者。

2.1. 反馈

如果你在本指南里发现了印刷错误，或者你有改进该手册的建议，我们希望听到你的声音！请提交报告到 JIRA 并指明：产品：JBoss Enterprise Application Platform, 版本：***<version>***，组件：*Doc*。如果是改进本文档的建议，请尽量具体化；如果是其他错误，请指出章节号以及具体的内容，这样我们就可以尽快改正。

2.2. 其他文档

如果你需要详细的产品信息，请访问 http://www.redhat.com/docs/manuals/jboss 里的在线手册。

第 3 章 JBoss Messaging - 快速教程

3.1. JBossMQ 的局限性

JBossMQ 有两个根本的局限性：

- JBossMQ 基于非群集的 SpyderMQ（一个开源项目）。
- 它的线程模型和非群集设计在某些高负载情形下会导致性能问题。

3.2. JBoss Messaging 的特征

JBoss Messaging 实现了一个高性能和稳定的消息核心，它的设计目的是为了最大限度地利用面向服务架构（Service Oriented Architectures，SOA）、企业级服务总线（Enterprise Service Buses，ESB）以及其他从最简单到最高要求的网络的集成需要。

它将允许你在群集系统里顺利地分布应用程序负载，智能地平衡和利用每个节点的 CPU 周期。它没有单个故障点和瓶颈，但它具有复杂和完全可配置的消息过期管理和 XA 事务恢复，因此提供了具有高可扩充性和性能的群集实现。它也包括了以基于标准的格式递送消息且能够支持将来的其他消息协议的 JMS 前端。

注意

JMS compliance：完全兼容和经 Sun 认证的 JMS 1.1 实现，可以在 JBoss 企业级应用程序平台 4.2、4.3 或更新版本里使用。

JBoss Messaging 还包含了其他功能：

- 发布-订阅和点到点消息模型
- 持久的和非持久的消息
- 确保消息在有需要时只到达一次的消息递送保证
- 事务性和可靠性 — 支持 ACID 模式
- 基于 JAAS 的可定制的安全框架
- 和 JBoss Transactions（以前称为 Arjuna JTA）完全集成，具有完全事务恢复能力。
- 扩展的 JMX 管理接口
- 对 Oracle、Sybase、MS SQL Server、PostgreSQL 和 MySQL 等主要数据库的支持
- 通过 HTTP 传输使用只允许 HTTP 通信的防火墙
- SSL 传输
- 可配置的 DLQs (Dead Letter Queues) 和 Expiry Queue
- 消息统计：给你关于递送给不同队列和订阅的不同消息的历史数据视图
- 消息到存储的自动分页。这允许你使用非常大的队列 — 大到不能马上存放在内存里

3.3. 群集功能

完全群集的队列和主题

“逻辑的”队列和主题分布在整个群集里。你可以向任何节点的队列和主题发送消息和接受消息。

群集的持久性订阅

你可以从群集里任何节点访问特定的持久性订阅 — 这允许把订阅的负载分摊到整个群集里。

完全群集的临时队列

如果发送带有 **`replyTo`** 临时队列的消息，它就可以送回群集里的任何节点。

智能消息重分布

如果在某个节点上消费者比另外一个节点块，消息将在群集中的不同节点间自动转移。这有助于防止 starvation 或者在某个节点上堆积消息。

消息顺序保护

如果你想确保生产者（Producer）产生消息的顺序和消费者（Consumer）消费消息的顺序一致，那就把这设置为 true。这甚至在消息重分布时也可以实现。

完全透明的失效切换

当某个服务器发生故障时，你的会话将在新的节点上继续进行，就好像什么都没有发生一样。（这是完全可配置的 – 如果你不希望这样，你可以设置为抛出异常并手工在另外节点上重新创建连接）

高可用性和无缝式失效切换

如果你连接的节点出现了故障，你将自动切换至另外一个节点且不会丢失任何持久性消息。你可以从断开的地方开始无缝地继续你的会话。在任何时候，持久性消息都只会被递送一次。

消息桥（Message Bridge）

JBoss Messaging 包含一个消息桥组件，它让你可以在处于相同或不同（如由 WAN 分隔）位置上的两个 JMS1.1 目的地之间连接消息。这允许你连接物理上独立的群集系统，从而组成大型的全球分布式逻辑队列和主题。

3.4. 对 JBossMQ 的兼容性

既然 JBoss Messaging 兼容 JMS 1.1 和 JMS 1.0.2b，针对 JBossMQ 编写的 JMS 代码不作修改就可以在 JBoss Messaging 里运行。

JBoss Messaging 不具备和 JBossMQ 的线格式（wire format）兼容，JBoss MQ 客户端有必要用 JBoss Messaging 客户端的 jar 文件进行升级。

重要

即使 JBoss Messaging 和 JBoss MQ 的部署描述符非常相似，但它们*并不是*相同的，所以要进行少许调整才能在 JBoss Messaging 里使用。而且，数据库数据模型完全不一样，所以请不要试图用 JBoss MQ 的数据模型来使用 JBoss Messaging，反之亦然。

注意

JBoss Messaging 是采用基于 Java 5 的 JBoss AS 4.2 类库进行构建的，因此 JBoss Messaging 只能运行在 Java 5 或更新版本里。

第 4 章 JBoss Messaging 示例

4.1. 关于这些示例

在 JBoss 企业级应用程序平台的安装目录里，你可以找到位于 **JBOSS_DIST/doc/examples/jboss-messaging-examples** 的一系列示例。这些例子演示了 JBoss Messaging 的运行。在 **JBOSS_DIST/doc/examples/jboss-messaging-examples** 里，你将看到下面的子目录：

- docs/examples/jboss-messaging-examples/queue
  这个例子演示了一个 JMS 客户端，它对远程队列进行简单的 '发送' 和 '接收'。
- docs/examples/jboss-messaging-examples/topic
  这个例子演示使用 JMS 客户端对远程主题（Topic）进行简单的 '发送' 和 '接收'。
- docs/examples/jboss-messaging-examples/mdb
  这个例子演示 JBoss Messaging 对 EJB2.1 MDB 的使用。
- docs/examples/jboss-messaging-examples/ejb3mdb
  这个例子演示 JBoss Messaging 对 EJB3 MDB 的使用。
- docs/examples/jboss-messaging-examples/stateless
  这个例子演示 EJB2.1 stateless session bean 和 JBoss Messaging 的交互。
- docs/examples/jboss-messaging-examples/mdb-failure
  这个例子演示了 EJB2.1 MDB 里发生的回滚（rollback）和重递送（redelivery）。
- docs/examples/jboss-messaging-examples/secure-socket
  这个例子演示了 JMS 客户端用 SSL 加密传输和 JBoss Messaging 服务器进行交互。
- docs/examples/jboss-messaging-examples/http
  这个例子演示了 JMS 客户端通过 HTTP 协议和 JBoss Messaging 服务器进行交互。
- docs/examples/jboss-messaging-examples/web-service
  这个例子演示了 JBoss web-service 和 JBoss Messaging 的交互。
- docs/examples/jboss-messaging-examples/distributed-queue
  这个例子演示了 JMS 客户端和 JBoss Messaging 分布式队列的交互 — 它要求运行两个 JBoss 应用服务器实例。
- docs/examples/jboss-messaging-examples/distributed-topic
  这个例子演示了 JMS 客户端和 JBoss Messaging 分布式主题的交互 — 它要求运行两个 JBoss 应用服务器实例。
- docs/examples/jboss-messaging-examples/stateless-clustered
  这个例子演示了 JMS 客户端和群集 EJB2.1 stateless session bean 的交互、以及它反过来和 JBoss Messaging 的交互。这个例子使用 HAJNDI 来查找连接工厂。它要求运行两个 JBoss 应用服务器实例。
- docs/examples/jboss-messaging-examples/bridge
  这个例子演示了消息桥（Message Bridge）的使用。它在 JBoss 应用服务器里部署消息桥，然后把消息从来源移至一个目标队列。

4.2. 运行示例程序

4.2.1. 安装 Ant

要编译这些例子，你必须安装 Apache Ant 1.6+。你可以从 http://ant.apache.org 上下载并按下列步骤安装：

- 把下载的文件解压到你选定的目录。
- 创建名为 **ANT_HOME** 的环境变量，使其指向 Ant 的安装目录。你可以在 **.bashrc** 文件里添加如下一行来实现（你需要替换成实际的 Ant 目录）：

```
export ANT_HOME=/home/user/apache-ant-1.7.0
```

在 Windows 里，你可以从启动菜单里打开控制面板（如有需要则切换至经典视图），然后打开系统/高级/环境变量。创建一个新的变量，命名为 **ANT_HOME** 并设置为 Ant 的安装目录。

- 在系统路径里添加 **$ANT_HOME/bin**，这样就能够在命令行下运行 **ant** 了。你可以在 **.bashrc** 文件里添加如下一行来实现这一点：

```
export PATH=$PATH:$ANT_HOME/bin
```

在 Windows 里，你可以从启动菜单里打开控制面板（如有需要则切换至经典视图），然后在系统/高级/环境变量/路径中找到并编辑环境变量 **PATH**。添加一个分号以及 Ant 的 **bin** 目录路径。

- 你可以在命令行提示下输入 **ant -version** 来验证 Ant 安装。输出应该类似于：

```
Apache Ant version 1.7.0 compiled on December 13 2006
```

4.2.2. 设置 JBOSS_HOME 环境变量

在 Linux 平台上

创建一个指向安装目录（**JBOSS_DIST/jboss-as**）的环境变量并命名为 **JBOSS_HOME**。把 **$JBOSS_HOME/bin** 添加至系统路径以便从命令行下运行服务器。你可以在自己主目录里的 **.bashrc** 文件添加如下一样来实现：

```
#In this example /home/vrenish/EnterprisePlatform-4.3/jboss-as is the installation
directory.
export JBOSS_HOME=/home/vrenish/EnterprisePlatform-4.3/jboss-as
export PATH=$PATH:$JBOSS_HOME/bin
```

在 Microsoft Windows 平台上

创建一个指向安装目录的环境变量（如 C:\Program Files\EnterprisePlatform-4.3\jboss-as\），命名为 **JBOSS_HOME**。为了从命令行下运行服务器，你需要把 **bin** 目录添加至系统路径，如 C:\Program Files\EnterprisePlatform-4.3\jboss-as\bin。你可以这样来完成：从启动菜单里打开控制面板（如有需要则切换至经典视图），打开“系统”，然后选择“高级”标签页并点击“环境变量”按钮。

4.2.3. 运行这个例子

请确保在运行这些例子之前启动了 JBoss 应用服务器。非群集的示例需要单个的 JBoss 应用服务器实例以缺省设置运行。群集示例则需要两个 JBoss 应用服务器实例并设置端口（如 ports-01 和 ports-02）。对于每个示例，你都可以通过编辑其相应目录下的 **jndi.properties** 来覆盖它试图连接的缺省端口。

要运行 **JBOSS_DIST/doc/examples/jboss-messaging-examples** 目录里的任何示例，你可以进入该示例的目录，并输入 **ant**。

第 5 章 配置

JMS API 指定消息客户端怎样和服务器进行交互。消息服务的确切定义和实现，如消息目的地和连接工厂，都是 JMS 提供者所专有的。JBoss Messaging 具有自己的配置文件来对服务进行配置。如果你准备从 JBossMQ（或其他 JMS 提供者）移植服务到 JBoss Messaging 里，你需要理解这些配置文件。在本章我们将讨论如何配置 JBoss Messaging 里的不同服务，使其为客户应用程序提供 JMS API 级的服务。

JBoss Messaging 服务的配置分布在几个配置文件里。根据所配置服务的功能，配置数据分布在 **`messaging-service.xml`**、**`remoting-bisocket-service.xml`**、 **`xxx-persistence-service.xml`（这里的 `xxx` 是数据库名）**、**`connection-factories-service.xml`** 和 **`destinations-service.xml`** 里。

AOP 客户端和服务器端的拦截器栈在 **`aop-messaging-client.xml`** 和 **`aop-messaging-server.xml`** 里进行配置。通常你不用改动它们，但你可以删除某些不需要的拦截器来小幅提高性能。在删除安全拦截器（security interceptor）之前，请先仔细考虑安全隐患。

5.1. 配置 ServerPeer

Server Peer 是 JBoss Messaging JMS facade 的核心。相关服务器的配置位于 **`messaging-service.xml`** 文件里。

所有的 JBoss Messaging 服务在 server peer 里都以根用户运行。

下面是一个 Server Peer 配置的例子。请注意这个例子并没有指定所有 Server Peer 属性的值。

```
      <mbean code="org.jboss.jms.server.ServerPeer"
      name="jboss.messaging:service=ServerPeer"
      xmbean-dd="xmdesc/ServerPeer-xmbean.xml">

      <!-- The unique id of the server peer - in a cluster each node MUST have a
unique value - must be an integer -->

      <attribute name="ServerPeerID">0</attribute>

      <!-- The default JNDI context to use for queues when they are deployed
without specifying one -->

      <attribute name="DefaultQueueJNDIContext">/queue</attribute>

      <!-- The default JNDI context to use for topics when they are deployed
without specifying one -->

      <attribute name="DefaultTopicJNDIContext">/topic</attribute>

      <attribute name="PostOffice">jboss.messaging:service=PostOffice</attribute>

      <!-- The JAAS security domain to use for JBoss Messaging -->

      <attribute name="SecurityDomain">java:/jaas/messaging</attribute>

      <!-- The default security configuration to apply to destinations - this can
be overridden on a per destination basis -->

      <attribute name="DefaultSecurityConfig">
        <security>
            <role name="guest" read="true" write="true" create="true"/>
        </security>
      </attribute>

      <!-- The default Dead Letter Queue (DLQ) to use for destinations.
           This can be overridden on a per destinatin basis -->

      <attribute
name="DefaultDLQ">jboss.messaging.destination:service=Queue,name=DLQ</attribute>

      <!-- The default maximum number of times to attempt delivery of a message
before sending to the DLQ (if configured).
           This can be overridden on a per destinatin basis -->

      <attribute name="DefaultMaxDeliveryAttempts">10</attribute>

      <!-- The default Expiry Queue to use for destinations. This can be
overridden on a per destinatin basis -->

      <attribute
name="DefaultExpiryQueue">jboss.messaging.destination:service=Queue,name=ExpiryQueu
e</attribute>

      <!-- The default redelivery delay to impose. This can be overridden on a
per destination basis -->

      <attribute name="DefaultRedeliveryDelay">0</attribute>

      <!-- The periodicity of the message counter manager enquiring on queues for
statistics -->

      <attribute name="MessageCounterSamplePeriod">5000</attribute>

      <!-- The maximum amount of time for a client to wait for failover to start
on the server side after
           it has detected failure -->
```

```
      <attribute name="FailoverStartTimeout">60000</attribute>

      <!-- The maximum amount of time for a client to wait for failover to
complete on the server side after
           it has detected failure -->

      <attribute name="FailoverCompleteTimeout">300000</attribute>

      <!-- The maximum number of days results to maintain in the message counter
history -->

      <attribute name="DefaultMessageCounterHistoryDayLimit">-1</attribute>

      <!-- The name of the connection factory to use for creating connections
between nodes to pull messages -->

      <attribute
name="ClusterPullConnectionFactoryName">jboss.messaging.connectionfactory:service=C
lusterPullConnectionFactory</attribute>

      <!-- When redistributing messages in the cluster. Do we need to preserve the
order of messages received
            by a particular consumer from a particular producer? -->

      <attribute name="DefaultPreserveOrdering">false</attribute>

      <!-- Max. time to hold previously delivered messages back waiting for
clients to reconnect after failover -->

      <attribute name="RecoverDeliveriesTimeout">300000</attribute>

      <attribute name="EnableMessageCounters">false</attribute>

      <!-- The password used by the message sucker connections to create
connections.
           THIS SHOULD ALWAYS BE CHANGED AT INSTALL TIME TO SECURE SYSTEM
      <attribute name="SuckerPassword"></attribute>
      -->

      <depends optional-attribute-
name="PersistenceManager">jboss.messaging:service=PersistenceManager</depends>

      <depends optional-attribute-
name="JMSUserManager">jboss.messaging:service=JMSUserManager</depends>

      <depends>jboss.messaging:service=Connector,transport=bisocket</depends>

   </mbean>
```

警告

安全风险！要避免安全风险，你必须指定 Server Peer 配置文件（messaging-service.xml）里的 SuckerPassword 属性的值。如果你没有进行指定，缺省值将被使用。任何知道缺省值的人都可以访问服务器上的所有目的地。只有管理员才应该知道这个密码。

5.1.1. ServerPeer 的属性

现在我们将讨论 ServerPeer MBean 的 MBean 属性。

5.1.1.1. ServerPeerID

Server peer 的唯一 ID。你部署的每个节点都必须有唯一的 ID。这适用于组成群集的不同节点，也适用于仅通过消息桥链接的节点。这个 ID 必须为有效的整型值。

5.1.1.2. DefaultQueueJNDIContext

绑定队列时使用的缺省 JNDI 上下文。缺省为 /queue。

5.1.1.3. DefaultTopicJNDIContext

绑定主题时使用的缺省 JNDI 上下文。缺省为 /topic。

5.1.1.4. PostOffice

这是 ServerPeer 使用的邮局（Post Office）。通常，你不需要修改这个属性。邮局负责把消息转发至队列并维护地址和队列间的映射。

5.1.1.5. SecurityDomain

本 server peer 使用的 JAAS 安全域名。

5.1.1.6. DefaultSecurityConfig

当特定队列或主题的安全配置没有覆盖目的地的部署描述符时，缺省的安全配置将被使用。它与在 JBossMQ 里的语法和模式完全相同。

DefaultSecurityConfig 属性元素应该包含一个 **<security>** 元素。**<security>** 元素可以包含多个 **<role>** 元素。每个 **<role>** 元素都定义了特定角色的缺省访问权限。

如果 **read** 属性为 **true**，那么这个角色将缺省可以读取目的地（创建消费者、接收或浏览消息）。

如果 **write** 属性为 **true**，那么这个角色将缺省可以写入目的地（创建生产者或发送消息）。

如果 **create** 属性为 **true**，那么这个角色将缺省可以创建主题上的可持久订阅。

5.1.1.7. DefaultDLQ

这是 Server Peer 用于目的地的缺省 DLQ（Dead Letter Queue）。每个目的地的 DLQ 可以被覆盖 – 更多细节请参考目的地 MBean 的配置。DLQ 是一个特殊的目的地，当服务器递送信息连续失败指定的次数之后，这些信息将被发送到这里。如果根本就没有指定 DLQ，那么这些消息在进行最多次数的递送尝试后将被删除。递送的最多次数可以用属性 DefaultMaxDeliveryAttempts 指定，它可以是全局的缺省属性也可以基于每个目的地进行设置。

5.1.1.8. DefaultMaxDeliveryAttempts

如果被设置的话，它表示消息在被发送到 DLQ 之前所尝试的最多次数。

它的缺省值是 **10**。

这个值也可以基于每个目的地进行覆盖。

5.1.1.9. DefaultExpiryQueue

这是 Server Peer 用于目的地的缺省 Expiry Queue 的名称。过期时间可以基于每个目的地进行覆盖 – 更多细节请参考目的地的 MBean 配置。Expiry Queue 是一个特殊的目的地，当消息过期时将被发送到这里。消息是否过期是由 Message::getJMSExpiration() 的值决定的，如果根本没有指定 Expiry Queue，那么消息在过期后将被删除。

5.1.1.10. DefaultRedeliveryDelay

在前一次递送失败后进行重新递送前，采用一定的延迟是有益处的，这是为了阻止连续的递送失败。

它的缺省值是 **0**，表示没有延迟。

修改这个值将使你的应用程序受益于重新递送前的延迟。这个值也可以基于每个目的地进行覆盖。

5.1.1.11. MessageCounterSamplePeriod

服务器将定期低查询每个队列来获取统计信息。这个值表示对应的时间间隔。

它的缺省值是 **10000** 毫秒。

5.1.1.12. FailoverStartTimeout

当检测到问题时，客户将等待服务器端启动失效切换的最长时间（毫秒）。

它的缺省值是 **60000**（亦即 1 分钟）。

5.1.1.13. FailoverCompleteTimeout

当失效切换已经启动时，客户将等待服务器端完成失效切换的最长时间（毫秒）。

它的缺省值是 **300000**（亦即 5 分钟）。

5.1.1.14. DefaultMessageCounterHistoryDayLimit

JBoss Messaging 提供消息计数器的历史记录，它显示一段时间内到达每个队列的消息数量。这个属性代表消息计数器历史存储的最多天数。这个属性可以基于目的地进行覆盖。

5.1.1.15. ClusterPullConnectionFactory

用于在节点间传送消息的连接工厂的名称。

如果你希望关闭队列间的消息 sucking，但保留失效切换，你可以忽略这个属性。

5.1.1.16. DefaultPreserveOrdering

如果为 true，群集里将应用严格的 JMS 顺序。更多细节请参考群集配置部分的内容。它的缺省值是 false。

5.1.1.17. RecoverDeliveriesTimeout

当发生失效切换时，已经递送的消息将被置于一旁，等待客户端的重新连接。如果客户端不在重新连接（如客户端崩溃），那么这些消息最终将超时并放回队列。它以毫秒为单位，缺省值是 5 毫秒。

5.1.1.18. EnableMessageCounters

设置为 true 可以在服务器启动时启用消息计数器。

5.1.1.19. SuckerPassword

JBoss Messaging 在节点间创建内部的连接来在群集目的地间重新分发消息。这些连接使用特殊的保留用户的名称进行创建。这个参数指定该用户使用的密码。

警告

这必须在安装时指定，否则缺省密码将被使用。任何知道缺省密码的人都可以访问服务器上的任何目的地。这个值必须在安装时进行修改。

5.1.1.20. StrictTCK

如果你想应用严格的 JMS TCK 模式，将其设置为 true。

5.1.1.21. Destinations

返回目前部署的目的地（队列和主题）列表。

5.1.1.22. MessageCounters

JBoss Messaging 为每个队列提供了一个消息计数器。

5.1.1.23. MessageCountersStatistics

JBoss Messaging 也为每个队列的每个消息计数器提供统计信息。

5.1.1.24. SupportsFailover

如设置为 false，当节点崩溃时会阻止群集里服务器端的失效切换。

5.1.1.25. PersistenceManager

这是 ServerPeer 使用的 persistence manager。你通常不需要修改这个属性。

5.1.1.26. JMSUserManager

这是 ServerPeer 使用的 JMS user manager。你通常不需要修改这个属性。

5.1.1.27. ServerPeer MBean 的 MBean 操作。

5.1.1.27.1. DeployQueue

这个操作可让你在程序里部署队列。

这个操作有两个重载的版本

如果队列存在但已卸载，它将被部署。否则队列将被创建并部署。

name 参数代表要部署的目的地的名称。

jndiName 参数（可选）代表目的地绑定的 JNDI 全名。如果没有被指定，那么这个目的地将绑定在 <DefaultQueueJNDIContext>/<name>。

这个操作的第一个版本用缺省的页面参数部署目的地。第二个重载的版本用指定的页面参数部署目的地。关于页面参数的定义，请参考目的地配置部分的内容。

5.1.1.27.2. UndeployQueue

这个操作让你可以在程序里卸载队列。

这个队列将被卸载但不会从持久性存储里删除。

如果队列被成功卸载，那么这个操作返回 **true**。否则返回 **false**。

5.1.1.27.3. DestroyQueue

这个操作可让你在程序里销毁队列。

队列被卸载，然后所有的数据将从数据库里销毁。

警告

在使用这个方法要小心，因为它将删除队列的所有数据。

如果队列被成功销毁，这个操作返回 **true**。否则返回 **false**。

5.1.1.27.4. DeployTopic

这个操作让你在程序里部署主题。

这个操作有两个重载的版本。

如果主题已存在但未卸载，那么它将被部署。否则它将被创建并部署。

name 参数代表要部署的目的地的名称。

jndiName 参数（可选）代表绑定目的地的 JNDI 全名。如果它没有被指定，那么这个目的地将绑定在 <DefaultTopicJNDIContext>/<name>。

这个操作的第一个版本用缺省的页面参数部署目的地。第二个重载的版本用指定的页面参数部署目的地。关于页面参数的定义，请参考目的地配置部分的内容。

5.1.1.27.5. UndeployTopic

这个操作让你可以在程序里卸载主题。

这个队列将被卸载但不会从持久性存储里删除。

如果主题被成功卸载，这个操作返回 **true**。否则返回 **false**。

5.1.1.27.6. DestroyTopic

这个操作可让你在程序里部署主题。

主题被卸载，然后它的所有数据都会从数据库里销毁。

警告

请小心使用这个方法，因为它将删除主题的所有数据。

如果主题被成功销毁，这个操作返回 **true**，否则返回 **false**。

5.1.1.27.7. ListMessageCountersHTML

这个操作返回 HTML 格式的消息计数器。

5.1.1.27.8. ResetAllMesageCounters

这个操作将所有消息计数器清零。

5.1.1.27.9. ResetAllMesageCounters

这个操作将所有消息计数器历史清零。

5.1.1.27.10. EnableMessageCounters

这个操作为所有目的地启用消息计数器。消息计数器缺省是禁用的。

5.1.1.27.11. DisableMessageCounters

这个操作禁用所有目的地的消息计数器。消息计数器缺省是禁用的。

5.1.1.27.12. RetrievePreparedTransactions

获取节点上当前处于 prepared 状态的所有事务的 XID 列表。

5.1.1.27.13. ShowPreparedTransactions

获取节点上当前处于 prepared 状态的所有事务的 XID 列表，并以 HTML 形式显示。

5.2. 修改数据库

JBoss AS 里的 JMS 服务使用关系型数据库来存储消息。为了改进性能，你应该修改 JMS 服务以利用外部的数据库。你需要用外部数据库对应的 **jboss-as/docs/examples/jms/** 里的文件替换 **jboss-as/server/production/deploy/jboss-messaging.sar/clustered-hsqldb-persistence-service.xml** 并重启服务器。

- MySQL: **mysql-persistence-service.xml**
- PostgreSQL: **postgresql-persistence-service.xml**
- Oracle: **oracle-persistence-service.xml**
- Sybase: **sybase-persistence-service.xml**
- MS SQL Server: **mssql-persistence-service.xml**

对于 **default** 和 **all** 配置，请分别替换 **jboss-as/server/default/deploy/jboss-messaging.sar/hsqldb-persistence-service.xml** 和 **jboss-as/server/all/deploy/jboss-messaging.sar/clustered-hsqldb-persistence-**

service.xml。

而且，请注意 Messaging 服务缺省依赖于以 **"java:/DefaultDS"** 引用的数据源。如果你采用不同的 JNDI 名称部署数据源，你需要更新持久化配置文件里所有的 **DataSource** 属性。本发行版本里包含了所有流行数据库的数据源配置示例。

你可以把 **jboss-as/docs/examples/jca** 里的例子复制到 **jboss-as/server/<*config-name*>/deploy** 里来配置 JCA 数据源。JBoss Messaging 缺省使用 **DefaultDS**。

5.3. 配置邮局

邮局（Post Office）的职责是把消息传递至目的地。

邮局维护消息可发送到的地址和最终队列之间的映射关系。

例如，当发送带有代表 JMS 队列名称的地址的消息时，邮局会将其传递到单一的对列 - 该 JMS 对列。而当带有代表 JMS 主题名称的地址的消息时，邮局会将其传递到一组对列 - 每个 JMS 订阅对应一个。

邮局也处理地址映射的持久化。

JBoss Messaging 的邮局也可用于群集。在群集环境里，它们将在节点间自动传递和获取消息来提供完全分布式的 JMS 队列和主题。

邮局的配置在 xxx-persistence-service.xml（xxx 是数据库的名称）文件里进行。

下面是一个邮局的配置示例：

```
   <mbean code="org.jboss.messaging.core.jmx.MessagingPostOfficeService"
      name="jboss.messaging:service=PostOffice"
      xmbean-dd="xmdesc/MessagingPostOffice-xmbean.xml">

      <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>

      <depends>jboss.jca:service=DataSourceBinding,name=DefaultDS</depends>

      <depends optional-attribute-
name="TransactionManager">jboss:service=TransactionManager</depends>

      <!-- The name of the post office -->

      <attribute name="PostOfficeName">JMS post office</attribute>

      <!-- The datasource used by the post office to access it's binding
information -->

      <attribute name="DataSource">java:/DefaultDS</attribute>

      <!-- If true will attempt to create tables and indexes on every start-up -->

      <attribute name="CreateTablesOnStartup">true</attribute>

      <!-- If true then we will automatically detect and reject duplicate messages
sent during failover -->

      <attribute name="DetectDuplicates">true</attribute>

      <!-- The size of the id cache to use when detecting duplicate messages -->

      <attribute name="IDCacheSize">500</attribute>

      <attribute name="SqlProperties"><![CDATA[
CREATE_POSTOFFICE_TABLE=CREATE TABLE JBM_POSTOFFICE (POSTOFFICE_NAME VARCHAR(255),
NODE_ID INTEGER, QUEUE_NAME VARCHAR(255), COND VARCHAR(1023), SELECTOR
VARCHAR(1023), CHANNEL_ID BIGINT, CLUSTERED CHAR(1), ALL_NODES CHAR(1), PRIMARY
KEY(POSTOFFICE_NAME, NODE_ID, QUEUE_NAME)) ENGINE = INNODB
INSERT_BINDING=INSERT INTO JBM_POSTOFFICE (POSTOFFICE_NAME, NODE_ID, QUEUE_NAME,
COND, SELECTOR, CHANNEL_ID, CLUSTERED, ALL_NODES) VALUES (?, ?, ?, ?, ?, ?, ?, ?)
DELETE_BINDING=DELETE FROM JBM_POSTOFFICE WHERE POSTOFFICE_NAME=? AND NODE_ID=?
AND QUEUE_NAME=?
LOAD_BINDINGS=SELECT QUEUE_NAME, COND, SELECTOR, CHANNEL_ID, CLUSTERED, ALL_NODES
FROM JBM_POSTOFFICE WHERE POSTOFFICE_NAME=? AND NODE_ID=?
      ]]></attribute>

      <!-- This post office is clustered. If you don't want a clustered post
office then set to false -->

      <attribute name="Clustered">true</attribute>

      <!-- All the remaining properties only have to be specified if the post
office is clustered.
           You can safely comment them out if your post office is non clustered --
>

      <!-- The JGroups group name that the post office will use -->

      <attribute
name="GroupName">${jboss.messaging.groupname:MessagingPostOffice}</attribute>

      <!-- Max time to wait for state to arrive when the post office joins the
cluster -->

      <attribute name="StateTimeout">5000</attribute>
```

```
      <!-- Max time to wait for a synchronous call to node members using the
MessageDispatcher -->

      <attribute name="CastTimeout">50000</attribute>

      <!-- Set this to true if you want failover of connections to occur when a
node is shut down -->

      <attribute name="FailoverOnNodeLeave">false</attribute>


      <!-- JGroups stack configuration for the data channel - used for sending data
across the cluster -->

      <!-- By default we use the TCP stack for data -->
      <attribute name="DataChannelConfig">
         <config>
            <TCP start_port="7900"
                 loopback="true"
                 recv_buf_size="20000000"
                 send_buf_size="640000"
                 discard_incompatible_packets="true"
                 max_bundle_size="64000"
                 max_bundle_timeout="30"
                 use_incoming_packet_handler="true"
                 use_outgoing_packet_handler="false"
                 down_thread="false" up_thread="false"
                 enable_bundling="false"
                 use_send_queues="false"
                 sock_conn_timeout="300"
                 skip_suspected_members="true"/>
            <MPING timeout="4000"
           bind_to_all_interfaces="true"
           mcast_addr="${jboss.messaging.datachanneludpaddress:228.6.6.6}"
           mcast_port="${jboss.messaging.datachanneludpport:45567}"
           ip_ttl="8"
           num_initial_members="2"
           num_ping_requests="1"/>
            <MERGE2 max_interval="100000"
                    down_thread="false" up_thread="false" min_interval="20000"/>
            <FD_SOCK down_thread="false" up_thread="false"/>
            <VERIFY_SUSPECT timeout="1500" down_thread="false" up_thread="false"/>
            <pbcast.NAKACK max_xmit_size="60000"
                           use_mcast_xmit="false" gc_lag="0"
                           retransmit_timeout="300,600,1200,2400,4800"
                           down_thread="false" up_thread="false"
                           discard_delivered_msgs="true"/>
            <pbcast.STABLE stability_delay="1000" desired_avg_gossip="50000"
                           down_thread="false" up_thread="false"
                           max_bytes="400000"/>
            <pbcast.GMS print_local_addr="true" join_timeout="3000"
                        down_thread="false" up_thread="false"
                        join_retry_timeout="2000" shun="false"
                        view_bundling="true"/>
        </config>
      </attribute>

      <!-- JGroups stack configuration to use for the control channel - used for
control messages -->

      <!-- We use udp stack for the control channel -->
      <attribute name="ControlChannelConfig">
         <config>
            <UDP
                 mcast_addr="${jboss.messaging.controlchanneludpaddress:228.7.7.7}"
                 mcast_port="${jboss.messaging.controlchanneludpport:45568}"
```

```
                tos="8"
                ucast_recv_buf_size="20000000"
                ucast_send_buf_size="640000"
                mcast_recv_buf_size="25000000"
                mcast_send_buf_size="640000"
                loopback="false"
                discard_incompatible_packets="true"
                max_bundle_size="64000"
                max_bundle_timeout="30"
                use_incoming_packet_handler="true"
                use_outgoing_packet_handler="false"
                ip_ttl="2"
                down_thread="false" up_thread="false"
                enable_bundling="false"/>
            <PING timeout="2000"
                  down_thread="false" up_thread="false" num_initial_members="3"/>
            <MERGE2 max_interval="100000"
                    down_thread="false" up_thread="false" min_interval="20000"/>
            <FD_SOCK down_thread="false" up_thread="false"/>
            <FD timeout="10000" max_tries="5" down_thread="false"
up_thread="false" shun="true"/>
            <VERIFY_SUSPECT timeout="1500" down_thread="false" up_thread="false"/>
            <pbcast.NAKACK max_xmit_size="60000"
                           use_mcast_xmit="false" gc_lag="0"
                           retransmit_timeout="300,600,1200,2400,4800"
                           down_thread="false" up_thread="false"
                           discard_delivered_msgs="true"/>
            <UNICAST timeout="300,600,1200,2400,3600"
                     down_thread="false" up_thread="false"/>
            <pbcast.STABLE stability_delay="1000" desired_avg_gossip="50000"
                           down_thread="false" up_thread="false"
                           max_bytes="400000"/>
            <pbcast.GMS print_local_addr="true" join_timeout="3000"
use_flush="true" flush_timeout="3000"
                        down_thread="false" up_thread="false"
                        join_retry_timeout="2000" shun="false"
                        view_bundling="true"/>
            <FRAG2 frag_size="60000" down_thread="false" up_thread="false"/>
            <pbcast.STATE_TRANSFER down_thread="false" up_thread="false"
use_flush="true" flush_timeout="3000"/>
            <pbcast.FLUSH down_thread="false" up_thread="false" timeout="20000"
auto_flush_conf="false"/>
        </config>
     </attribute>

   </mbean>
```

5.3.1. 邮局具有下列的属性

5.3.1.1. DataSource

邮局用来持久化其映射数据的数据源。

5.3.1.2. SQLProperties

这是为特定数据库指定的 DDL 和 DML。如果 DDL 和 DML 没有根据特定数据库覆盖，那么缺省的 Hypersonic 配置将被使用。

5.3.1.3. CreateTablesOnStartup

如果你希望邮局在启动时创建表（和索引），设置它为 **true**。如果这个表（或索引）已经存在，那么 JDBC 驱动将抛出 **SQLException**，但 Persistence Manager 会忽略它并允许继续运行。

CreateTablesOnStartup 属性的缺省值是 **true**。

5.3.1.4. DetectDuplicates

如果你希望邮局检测在服务器故障后重发的重复消息，可将其设置为 **true**。

DetectDuplicates 属性的缺省值是 **true**。

5.3.1.5. IDCacheSize

如果启用了对重复消息的检测（请参考 **DetectDuplicates**），那么服务器将记住最后发送的 **n** 条消息的 ID。这将阻止发生故障后发送重复的消息。**n** 由这个属性指定。

IDCacheSize 属性的缺省值为 **500**。

5.3.1.6. PostOfficeName

邮局的名称。

5.3.1.7. NodeIDView

它返回包含群集里所有节点的 ID 的集合。

5.3.1.8. GroupName

群集里的所有具有相同组名的邮局将组成一个群集。请确保组名匹配你希望组成群集的所有节点。

5.3.1.9. Clustered

如果为 true，邮局将参与群集来组成分布式的队列和主题。如果为 false 则不参与群集，且所有和群集相关的属性都会被忽略。

5.3.1.10. StateTimeout

当某个节点加入现有的群集时，在组状态到达前等待的最长时间。

它的缺省值为 **5000** 毫秒。

5.3.1.11. CastTimeout

同步转换消息回复的最长等待时间。

它的缺省值为 **5000** 毫秒。

5.3.1.12. FailoverOnNodeLeave

如果这个属性为 **true**，当服务器节点关闭后，任何到这个节点的连接都会切换到另外一个节点上。

这个属性的缺省值是 **false**。

5.3.1.13. MaxConcurrentReplications

在阻塞回复前，最大的并行复制请求数量。这可以防止 JGroups 的负担过重。我们通常不需要修改它。

它的缺省值是 **50**。

5.3.1.14. ControlChannelConfig

JBoss Messaging 将 JGroups 用于所有的组管理。这包含了用于控制频道的 JGroups 栈配置。

控制频道（Control Channel）被用来往/从群集里的其他节点发送请求/接收响应。

既然这只是标准的 JGroups 配置，我们不会在这里讨论其细节。你可以在 JGroups 发行文档 http://www.jgroups.org 或 http://wiki.jboss.org/wiki/Wiki.jsp?page=JGroups 里找到 JGroups 的详细信息。

5.3.1.15. DataChannelConfig

JBoss Messaging 将 JGroups 用于所有的组管理。这包含了用于数据频道（Data Channel）的 JGroups 栈配置。

数据频道（Data Channel）被用来往/从群集里的其他节点发送请求/接收响应以及复制会话数据。

既然这只是标准的 JGroups 配置，我们不会在这里讨论其细节。你可以在 JGroups 发行文档 http://www.jgroups.org 或 http://wiki.jboss.org/wiki/Wiki.jsp?page=JGroups 里找到 JGroups 的详细信息。

5.4. 配置 Persistence Manager

Persistence Manager 的职责是管理所有和消息相关的持久化。

JBoss Messaging 带有一个在通过 JDBC 访问的关系型数据库里处理消息数据的持久化的 JDBC Persistence Manager。这个 Persistence Manager 实现是可插拔的（Persistence Manager 是一个消息服务器的插件），这使得在非关系型数据库里（如文件系统）提供消息数据的持久化成为可能。

"persistent" 服务的配置以 **`xxx-persistence-service.xml`** 文件的形式分组，其中的 xxx 对应数据库名称。在缺省情况下，JBoss Messaging 附带一个 **`hsqldb-persistence-service.xml`** 文件，它配置任何 JBossAS 实例缺省使用的 in-VM Hypersonic 数据库。

警告

Persistence Manager 缺省使用 Hypersonic。然而，我们需要强调的是，Hypersonic 不应该用在产品环境里，这是因为它只有限地支持事务隔离，且在高负荷下表现不稳定。

警告

Critique of Hypersonic wiki 页面概述了使用这个数据库的一些众所周知的问题。

JBoss Messaging 也附带用于 MySQL、Oracle、PostgreSQL、Sybase 和 MS SQL Server 的 Persistence Manager 配置。示例的 **`mysql-persistence-service.xml`**、**`ndb-persistence-service.xml`**、**`oracle-persistence-service.xml`**、**`postgres-persistence-service.xml`**、 **`sybase-persistence-service.xml`** 和 **`mssql-persistence-service.xml`** 文件位于 **`examples/config`** 目录之下。

我们鼓励用户贡献自己的配置文件，我们将在认证之前进行详尽测试。JDBC Persistence Manager 被设计成对 DML 使用标准的 SQL，所以为其他数据库编写 JDBC Persistence Manager 配置通常只需要修改其中的 DDL（对于不同的数据库，这些语句很可能有差别）就可以了。

JBoss Messaging 也带有一个 'Null Persistence Manager' 配置 — 它可以在你根本不需要任何持久化时使用。

下面是缺省的 Hypersonic persistence 配置文件：

```
  <mbean code="org.jboss.messaging.core.jmx.JDBCPersistenceManagerService"
      name="jboss.messaging:service=PersistenceManager"
      xmbean-dd="xmdesc/JDBCPersistenceManager-xmbean.xml">

      <depends>jboss.jca:service=DataSourceBinding,name=DefaultDS</depends>

      <depends optional-attribute-
name="TransactionManager">jboss:service=TransactionManager</depends>

      <!-- The datasource to use for the persistence manager -->

      <attribute name="DataSource">java:/DefaultDS</attribute>

      <!-- If true will attempt to create tables and indexes on every start-up -->

      <attribute name="CreateTablesOnStartup">true</attribute>

      <!-- If true then will use JDBC batch updates -->

      <attribute name="UsingBatchUpdates">true</attribute>

      <attribute name="SqlProperties"><![CDATA[
   CREATE_DUAL=CREATE TABLE JBM_DUAL (DUMMY INTEGER, PRIMARY KEY (DUMMY)) ENGINE =
INNODB
   CREATE_MESSAGE_REFERENCE=CREATE TABLE JBM_MSG_REF (CHANNEL_ID BIGINT,
MESSAGE_ID BIGINT, TRANSACTION_ID BIGINT, STATE CHAR(1), ORD BIGINT, PAGE_ORD
BIGINT, DELIVERY_COUNT INTEGER, SCHED_DELIVERY BIGINT, PRIMARY KEY(CHANNEL_ID,
MESSAGE_ID)) ENGINE = INNODB
   CREATE_IDX_MESSAGE_REF_TX=CREATE INDEX JBM_MSG_REF_TX ON JBM_MSG_REF
(TRANSACTION_ID)
   CREATE_IDX_MESSAGE_REF_ORD=CREATE INDEX JBM_MSG_REF_ORD ON JBM_MSG_REF (ORD)
   CREATE_IDX_MESSAGE_REF_PAGE_ORD=CREATE INDEX JBM_MSG_REF_PAGE_ORD ON
JBM_MSG_REF (PAGE_ORD)
   CREATE_IDX_MESSAGE_REF_MESSAGE_ID=CREATE INDEX JBM_MSG_REF_MESSAGE_ID ON
JBM_MSG_REF (MESSAGE_ID)
   CREATE_IDX_MESSAGE_REF_SCHED_DELIVERY=CREATE INDEX JBM_MSG_REF_SCHED_DELIVERY
ON JBM_MSG_REF (SCHED_DELIVERY)
   CREATE_MESSAGE=CREATE TABLE JBM_MSG (MESSAGE_ID BIGINT, RELIABLE CHAR(1),
EXPIRATION BIGINT, TIMESTAMP BIGINT, PRIORITY TINYINT, TYPE TINYINT, HEADERS
MEDIUMBLOB, PAYLOAD LONGBLOB, PRIMARY KEY (MESSAGE_ID)) ENGINE = INNODB
   CREATE_IDX_MESSAGE_TIMESTAMP=CREATE INDEX JBM_MSG_REF_TIMESTAMP ON JBM_MSG
(TIMESTAMP)
   CREATE_TRANSACTION=CREATE TABLE JBM_TX (NODE_ID INTEGER, TRANSACTION_ID BIGINT,
BRANCH_QUAL VARBINARY(254), FORMAT_ID INTEGER, GLOBAL_TXID VARBINARY(254), PRIMARY
KEY (TRANSACTION_ID)) ENGINE = INNODB
   CREATE_COUNTER=CREATE TABLE JBM_COUNTER (NAME VARCHAR(255), NEXT_ID BIGINT,
PRIMARY KEY(NAME)) ENGINE = INNODB
   INSERT_DUAL=INSERT INTO JBM_DUAL VALUES (1)
   CHECK_DUAL=SELECT 1 FROM JBM_DUAL
   INSERT_MESSAGE_REF=INSERT INTO JBM_MSG_REF (CHANNEL_ID, MESSAGE_ID,
TRANSACTION_ID, STATE, ORD, PAGE_ORD, DELIVERY_COUNT, SCHED_DELIVERY) VALUES (?,
?, ?, ?, ?, ?, ?, ?, ?)
   DELETE_MESSAGE_REF=DELETE FROM JBM_MSG_REF WHERE MESSAGE_ID=? AND CHANNEL_ID=?
AND STATE='C'
   UPDATE_MESSAGE_REF=UPDATE JBM_MSG_REF SET TRANSACTION_ID=?, STATE='-' WHERE
MESSAGE_ID=? AND CHANNEL_ID=? AND STATE='C'
   UPDATE_PAGE_ORDER=UPDATE JBM_MSG_REF SET PAGE_ORD = ? WHERE MESSAGE_ID=? AND
CHANNEL_ID=?
   COMMIT_MESSAGE_REF1=UPDATE JBM_MSG_REF SET STATE='C', TRANSACTION_ID = NULL
WHERE TRANSACTION_ID=? AND STATE='+'
   COMMIT_MESSAGE_REF2=DELETE FROM JBM_MSG_REF WHERE TRANSACTION_ID=? AND
STATE='-'
   ROLLBACK_MESSAGE_REF1=DELETE FROM JBM_MSG_REF WHERE TRANSACTION_ID=? AND
STATE='+'
   ROLLBACK_MESSAGE_REF2=UPDATE JBM_MSG_REF SET STATE='C', TRANSACTION_ID = NULL
WHERE TRANSACTION_ID=? AND STATE='-'
```

```
   LOAD_PAGED_REFS=SELECT MESSAGE_ID, DELIVERY_COUNT, PAGE_ORD, SCHED_DELIVERY
FROM JBM_MSG_REF WHERE CHANNEL_ID = ? AND PAGE_ORD BETWEEN ? AND ? ORDER BY
PAGE_ORD
   LOAD_UNPAGED_REFS=SELECT MESSAGE_ID, DELIVERY_COUNT, SCHED_DELIVERY FROM
JBM_MSG_REF WHERE STATE = 'C' AND CHANNEL_ID = ? AND PAGE_ORD IS NULL ORDER BY
ORD
   LOAD_REFS=SELECT MESSAGE_ID, DELIVERY_COUNT, SCHED_DELIVERY FROM JBM_MSG_REF
WHERE STATE = 'C' AND CHANNEL_ID = ? ORDER BY ORD
   UPDATE_REFS_NOT_PAGED=UPDATE JBM_MSG_REF SET PAGE_ORD = NULL WHERE PAGE_ORD
BETWEEN ? AND ? AND CHANNEL_ID=?
   SELECT_MIN_MAX_PAGE_ORD=SELECT MIN(PAGE_ORD), MAX(PAGE_ORD) FROM JBM_MSG_REF
WHERE CHANNEL_ID = ?
   SELECT_EXISTS_REF_MESSAGE_ID=SELECT MESSAGE_ID FROM JBM_MSG_REF WHERE
MESSAGE_ID = ?
   UPDATE_DELIVERY_COUNT=UPDATE JBM_MSG_REF SET DELIVERY_COUNT = ? WHERE
CHANNEL_ID = ? AND MESSAGE_ID = ?
   UPDATE_CHANNEL_ID=UPDATE JBM_MSG_REF SET CHANNEL_ID = ? WHERE CHANNEL_ID = ?
   LOAD_MESSAGES=SELECT MESSAGE_ID, RELIABLE, EXPIRATION, TIMESTAMP, PRIORITY,
HEADERS, PAYLOAD, TYPE FROM JBM_MSG
   INSERT_MESSAGE=INSERT INTO JBM_MSG (MESSAGE_ID, RELIABLE, EXPIRATION,
TIMESTAMP, PRIORITY, TYPE, HEADERS, PAYLOAD) VALUES (?, ?, ?, ?, ?, ?, ?, ?)
   INSERT_MESSAGE_CONDITIONAL=INSERT INTO JBM_MSG (MESSAGE_ID, RELIABLE,
EXPIRATION, TIMESTAMP, PRIORITY, TYPE, INST_TIME) SELECT ?, ?, ?, ?, ?, ?, ? FROM
JBM_DUAL WHERE NOT EXISTS (SELECT MESSAGE_ID FROM JBM_MSG WHERE MESSAGE_ID = ?)
   UPDATE_MESSAGE_4CONDITIONAL=UPDATE JBM_MSG SET HEADERS=?, PAYLOAD=? WHERE
MESSAGE_ID=?
   INSERT_MESSAGE_CONDITIONAL_FULL=INSERT INTO JBM_MSG (MESSAGE_ID, RELIABLE,
EXPIRATION, TIMESTAMP, PRIORITY, TYPE, HEADERS, PAYLOAD) SELECT ?, ?, ?, ?, ?, ?,
?, ? FROM JBM_DUAL WHERE NOT EXISTS (SELECT MESSAGE_ID FROM JBM_MSG WHERE
MESSAGE_ID = ?)
   MESSAGE_ID_COLUMN=MESSAGE_ID
   DELETE_MESSAGE=DELETE FROM JBM_MSG WHERE MESSAGE_ID = ? AND NOT EXISTS (SELECT
* FROM JBM_MSG_REF WHERE JBM_MSG_REF.MESSAGE_ID = ?)
   INSERT_TRANSACTION=INSERT INTO JBM_TX (NODE_ID, TRANSACTION_ID, BRANCH_QUAL,
FORMAT_ID, GLOBAL_TXID) VALUES(?, ?, ?, ?, ?)
   DELETE_TRANSACTION=DELETE FROM JBM_TX WHERE NODE_ID = ? AND TRANSACTION_ID = ?
   SELECT_PREPARED_TRANSACTIONS=SELECT TRANSACTION_ID, BRANCH_QUAL, FORMAT_ID,
GLOBAL_TXID FROM JBM_TX WHERE NODE_ID = ?
   SELECT_MESSAGE_ID_FOR_REF=SELECT MESSAGE_ID, CHANNEL_ID FROM JBM_MSG_REF WHERE
TRANSACTION_ID = ? AND STATE = '+' ORDER BY ORD
   SELECT_MESSAGE_ID_FOR_ACK=SELECT MESSAGE_ID, CHANNEL_ID FROM JBM_MSG_REF WHERE
TRANSACTION_ID = ? AND STATE = '-' ORDER BY ORD
   UPDATE_COUNTER=UPDATE JBM_COUNTER SET NEXT_ID = ? WHERE NAME=?
   SELECT_COUNTER=SELECT NEXT_ID FROM JBM_COUNTER WHERE NAME=? FOR UPDATE
   INSERT_COUNTER=INSERT INTO JBM_COUNTER (NAME, NEXT_ID) VALUES (?, ?)
   SELECT_ALL_CHANNELS=SELECT DISTINCT(CHANNEL_ID) FROM JBM_MSG_REF
   UPDATE_TX=UPDATE JBM_TX SET NODE_ID=? WHERE NODE_ID=?
      ]]></attribute>

      <!-- The maximum number of parameters to include in a prepared statement --
>

      <attribute name="MaxParams">500</attribute>

      <attribute name="UseNDBFailoverStrategy">true</attribute>

   </mbean>
```

5.4.1. PersistenceManager MBean 的 MBean 属性

5.4.1.1. CreateTablesOnStartup

如果你希望 Persistence Manager 在启动时创建表（以及索引），请将其设置为 **true**。如果这些表（以及索引）已经存在，JDBC 驱动将抛出一个 **SQLException**，Persistence Manager 会忽略这个异常并继续

运行。

CreateTablesOnStartup 属性的缺省值是 **true**。

5.4.1.2. UsingBatchUpdates

如果数据库支持 JDBC 批量更新，请设置它为 **true**。JDBC Persistence Manager 将对批量的数据库更新进行分组来提供性能。

UsingBatchUpdates 属性的缺省值是 **false**。

5.4.1.3. UsingBinaryStream

如果你希望用 JDBC 二进制流而不是 getBytes()、setBytes() 存储和读取消息，请设置它为 **true**。有些数据库对用 getBytes()/setBytes() 读取/写的字节数有限制。

UsingBinaryStream 属性的缺省值是 **true**。

5.4.1.4. UsingTrailingByte

Sybase 的某些版本会把尾部的零截为圆点。要防止这种情况，这个属性应该设置为 **true**，这样尾部的非零字节将在持久化之前和之后添加和删除以防止数据库删除它们。目前这仅对于 Sybase 是必需的。

UsingTrailingByte 属性的缺省值是 **false**。

5.4.1.5. SupportsBlobOnSelect

Oracle（可能还有其他数据库）不允许用 INSERT INTO ... SELECT FROM 语句插入 BLOB，且要求两阶段插入消息。如果这个值为 false，那么将使用两阶段的插入。

SupportsBlobOnSelect 属性缺省值是 **true**。

5.4.1.6. SQLProperties

这是为特定数据库指定的 DDL 和 DML。如果 DDL 和 DML 没有根据特定数据库覆盖，那么缺省的 Hypersonic 配置将被使用。

5.4.1.7. MaxParams

当加载消息时，Persistence manager 将生成带有很多参数的 prepared 语句。这个值通知 Persistence manager 每个 prepared 语句所允许的参数的最大个数。

MaxParams 属性的缺省值是**100**。

5.4.1.8. UseNDBFailoverStrategy

当在群集的数据库环境里运行时，某些数据库如 MySQL，可能在提交数据库事务时失败。如果在提交阶段数据库节点停止运行了，那么就无法获知该事务的最后状态。如果设置了 **UseNDBFailoverStrategy** 为 true，那么相关 SQL 语句将被重新执行。然而，任何进一步的错误将被忽略，这是因为我们假设这个错误是由于前面成功提交的事务所引起的。

UseNDBFailoverStrategy 的缺省值是 **false**。

5.5. 配置 JMS User Manager

JMS user manager 处理预配置的客户 ID 和用户的映射，它也管理用户和角色表（其是否被使用取决于你所配置的登录模块）。

下面是一个配置 JMSUserManager 的例子

```
   <mbean code="org.jboss.jms.server.plugin.JDBCJMSUserManagerService"
      name="jboss.messaging:service=JMSUserManager"
      xmbean-dd="xmdesc/JMSUserManager-xmbean.xml">
      <depends>jboss.jca:service=DataSourceBinding,name=DefaultDS</depends>
      <depends optional-attribute-name="TransactionManager">
               jboss:service=TransactionManager
      </depends>
      <attribute name="DataSource">java:/DefaultDS</attribute>
      <attribute name="CreateTablesOnStartup">true</attribute>
      <attribute name="SqlProperties"><![CDATA[
               CREATE_USER_TABLE=CREATE TABLE JBM_USER (USER_ID VARCHAR(32) NOT
NULL,
               PASSWD VARCHAR(32) NOT NULL, CLIENTID VARCHAR(128),
               PRIMARY KEY(USER_ID)) ENGINE = INNODB
               CREATE_ROLE_TABLE=CREATE TABLE JBM_ROLE (ROLE_ID VARCHAR(32) NOT
NULL,
               USER_ID VARCHAR(32) NOT NULL, PRIMARY KEY(USER_ID, ROLE_ID))
               ENGINE = INNODB
               SELECT_PRECONF_CLIENTID=SELECT CLIENTID FROM JBM_USER WHERE
USER_ID=?
               POPULATE.TABLES.1=INSERT INTO JBM_USER (USER_ID,PASSWD,CLIENTID)
               VALUES ('dilbert','dogbert','dilbert-id')
      ]]></attribute>
   </mbean>
```

5.5.1. JMSUserManager MBean 的 MBean 属性

5.5.1.1. CreateTablesOnStartup

如果你希望 JMS user manager 试图在启动时创建表（和索引），请设置它为 **true**。如果这个表（或索引）已经存在，那么 JDBC 驱动将抛出 **SQLException** 而 Persistence Manager 会忽略这个异常来使其继续运行。

CreateTablesOnStartup 属性的缺省值是 **true**。

5.5.1.2. UsingBatchUpdates

如果数据库支持 JDBC 批量更新，请设置它为 **true**。JDBC Persistence Manager 将对批量的数据库更新进行分组来提供性能。

UsingBatchUpdates 属性的缺省值是 **false**。

5.5.1.3. SQLProperties

这是为特定数据库指定的 DDL 和 DML。如果 DDL 和 DML 没有根据特定数据库覆盖，那么缺省的 Hypersonic 配置将被使用。

缺省的用户和角色数据也可以在这里指定。如上例一样，任何被插入的数据必须用名称以 **POPULATE.TABLES** 开始的属性来指定。

5.6. 配置目的地

5.6.1. 预配置的目的地

JBoss Messaging 带有一套缺省的预配置目的地，它将在服务器启动过程中被部署。针对这些目的地的包含配置信息的文件是 **destinations-service.xml**。下面是这个文件的一部分内容：

```
   <!--
      The Default Dead Letter Queue. This destination is a dependency of an EJB MDB
container.
   -->

   <mbean code="org.jboss.jms.server.destination.QueueService"
      name="jboss.messaging.destination:service=Queue,name=DLQ"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
                  jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
   </mbean>


   <mbean code="org.jboss.jms.server.destination.TopicService"
      name="jboss.messaging.destination:service=Topic,name=testTopic"
      xmbean-dd="xmdesc/Topic-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
                  jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
      <attribute name="SecurityConfig">
         <security>
            <role name="guest" read="true" write="true"/>
            <role name="publisher" read="true" write="true" create="false"/>
            <role name="durpublisher" read="true" write="true" create="true"/>
         </security>
      </attribute>
   </mbean>

   <mbean code="org.jboss.jms.server.destination.TopicService"
      name="jboss.messaging.destination:service=Topic,name=securedTopic"
      xmbean-dd="xmdesc/Topic-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
                  jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
      <attribute name="SecurityConfig">
         <security>
            <role name="publisher" read="true" write="true" create="false"/>
         </security>
      </attribute>
   </mbean>


   <mbean code="org.jboss.jms.server.destination.QueueService"
      name="jboss.messaging.destination:service=Queue,name=testQueue"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
                  jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
      <attribute name="SecurityConfig">
         <security>
            <role name="guest" read="true" write="true"/>
            <role name="publisher" read="true" write="true" create="false"/>
            <role name="noacc" read="false" write="false" create="false"/>
         </security>
      </attribute>
   </mbean>

   <mbean code="org.jboss.jms.server.destination.QueueService"
      name="jboss.messaging.destination:service=Queue,name=A"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
```

```
                    jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
   </mbean>


   <!-- It's possible for indiviual queues and topics to use a specific queue for
   an expiry or DLQ -->

   <mbean code="org.jboss.jms.server.destination.QueueService"
      name="jboss.messaging.destination:service=Queue,name=PrivateDLQ"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
                    jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
   </mbean>

   <mbean code="org.jboss.jms.server.destination.QueueService"
      name="jboss.messaging.destination:service=Queue,name=PrivateExpiryQueue"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
                    jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
   </mbean>

   <mbean code="org.jboss.jms.server.destination.QueueService"

name="jboss.messaging.destination:service=Queue,name=QueueWithOwnDLQAndExpiryQueue"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
                    jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
      <attribute name="DLQ">
                    jboss.messaging.destination:service=Queue,name=PrivateDLQ
      </attribute>
      <attribute name="ExpiryQueue">

jboss.messaging.destination:service=Queue,name=PrivateExpiryQueue
      </attribute>
   </mbean>

   <mbean code="org.jboss.jms.server.destination.TopicService"

name="jboss.messaging.destination:service=Topic,name=TopicWithOwnDLQAndExpiryQueue"
      xmbean-dd="xmdesc/Topic-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
                    jboss.messaging:service=ServerPeer
      </depends>
      <depends>jboss.messaging:service=PostOffice</depends>
      <attribute name="DLQ">
                    jboss.messaging.destination:service=Queue,name=PrivateDLQ
      </attribute>
      <attribute name="ExpiryQueue">

jboss.messaging.destination:service=Queue,name=PrivateExpiryQueue
      </attribute>
   </mbean>


   <mbean code="org.jboss.jms.server.destination.TopicService"

name="jboss.messaging.destination:service=Topic,name=TopicWithOwnRedeliveryDelay"
      xmbean-dd="xmdesc/Topic-xmbean.xml">
      <depends optional-attribute-name="ServerPeer">
```

```
                    jboss.messaging:service=ServerPeer
        </depends>
        <depends>jboss.messaging:service=PostOffice</depends>
        <attribute name="RedeliveryDelay">5000</attribute>
    </mbean>


    <mbean code="org.jboss.jms.server.destination.TopicService"
        name="jboss.messaging.destination:service=Topic,name=testDistributedTopic"
        xmbean-dd="xmdesc/Topic-xmbean.xml">
        <depends optional-attribute-name="ServerPeer">
                    jboss.messaging:service=ServerPeer
        </depends>
        <depends>jboss.messaging:service=PostOffice</depends>
        <attribute name="Clustered">true</attribute>
    </mbean>
....
```

5.6.2. 配置队列

5.6.2.1. Queue MBean 的属性

5.6.2.1.1. Name

队列的名称

5.6.2.1.2. JNDIName

队列绑定的 JNDI 名称

5.6.2.1.3. DLQ

用于这个队列的 DLQ。它将覆盖 ServerPeer 配置里的任何值。

5.6.2.1.4. ExpiryQueue

用于这个队列的过期队列（Expiry Queue）。它将覆盖 ServerPeer 配置里的任何值。

5.6.2.1.5. RedeliveryDelay

这个队列使用的重递送延迟。它将覆盖 ServerPeer 配置里的任何值。

5.6.2.1.6. MaxDeliveryAttempts

在把消息发送到 DLQ 之前，尝试递送消息的最大次数。如果为 -1（缺省值），ServerPeer 配置里值将被使用。任何其他值将覆盖 ServerPeer 配置里的值。

5.6.2.1.7. 目的地的安全性配置

`SecurityConfig` - 允许你决定哪些角色可以在目的地读、写和创建。它具有和 JBossMQ 目的地的安全配置完全相同的语法。

`SecurityConfig` 元素里应该包含一个 **`<security>`** 元素。而 **`<security>`** 元素可以包含多个 **`<role>`** 元素。每个 **`<role>`** 元素都定义特定角色的访问权限。

如果 **`read`** 属性为 **`true`**，那么这个角色可以读取（创建消费者、接收消息或浏览消息）目的地。

如果 **`write`** 属性为 **`true`**，那么该角色可以写这个目的地（创建生产者或发送消息）。

如果 **`create`** 属性为 **`true`**，那么该角色可以在这个目的地上创建持久性订阅。

请注意目的地的安全性配置是可选的。如果没有指定 **`SecurityConfig`** 元素，那么 Server Peer 里的缺省安全配置将被使用。

5.6.2.1.8. 目的地的页面参数

'Pageable Channels' 是 JBoss Messaging 里的一个复杂的新功能。

如果你的应用程序需要支持超大的对列或可能包含数百万条消息的订阅，那么一次性进行存储就不可能。

JBoss Messaging 没有解决这个问题，但它可以让你指定可以一次性存储在内存里的每个队列或主题的消息的最大个数。JBoss Messaging 然后以块的形式透明地将这些消息分页，允许队列和订阅非常大而不影响性能。

这已经在基本的硬件上使用超过一千万零两千条消息进行了测试，且有可能扩展至更大的数目。

其各个参数为：

FullSize - 这是内存里的队列或主题订阅在任何时刻可容纳的最大消息数目。实际的队列或订阅可以容纳比这更多的消息，因为在添加或消费消息时可以按需要将这些消息分页至存储里。

PageSize - 当从队列或订阅加载消息时，这是在某个操作里可以预加载的最大消息数目。

DownCacheSize - 当把消息从队列里分页至存储里时，在实际上写入前它们首先进入一个 "Down Cache" 里。这使写入成为单个操作，因此提高了性能。这个设置决定了 Down Cache 在冲刷消息至存储前可容纳的消息的最大数目。

如果没有指定 **FullSize**、**PageSize** 或 **DownCacheSize**，它们的缺省值将分别是 75000、2000、2000。

如果你向指定用于临时队列的页面参数，那么你需要在合适的连接工厂上对它们进行指定。详情请参考连接工厂的配置。

5.6.2.1.9. CreatedProgrammatically

如果队列是在程序里创建的话，返回 **true**

5.6.2.1.10. MessageCount

返回队列里的消息总数，它等于正在递送的消息数目 + 未递送的消息数目 + 正被调度的消息数目。

5.6.2.1.11. ScheduledMessageCount

返回队列里已调度的消息的数目。这些消息预定在以后的时间进行递送。

调度递送是 JBoss Messaging 的一个功能，你可以发送消息并指定递送的最早时间。例如，你可以现在就发送一条消息，但这个消息实际上会在两小时后被递送。

要实现这一点，你只需要在消息里设置下面的消息头部信息：

```
              long now = System.currentTimeMillis();

              Message msg = sess.createMessage();


msg.setLongProperty(JBossMessage.JMS_JBOSS_SCHEDULED_DELIVERY_PROP_NAME,
                      now + 1000 * 60 * 60 * 2);

              prod.send(msg);
```

5.6.2.1.12. MaxSize

指定队列里消息的最大数目。任何超过这个数量的消息将被忽略。它的缺省值为 **-1**，表示不加限制。

5.6.2.1.13. Clustered

群集的目的地必须把它设置为 **true**。

5.6.2.1.14. MessageCounter

每个队列都维护一个消息计数器。

5.6.2.1.15. MessageCounterStatistics
消息计数器的统计信息

5.6.2.1.16. MessageCounterHistoryDayLimit
消息计数器历史记录保留的最多天数。它将覆盖 ServerPeer 配置里的任何值。

5.6.2.1.17. ConsumerCount
目前在队列里消费的消费者的数量。

5.6.2.2. Queue MBean 的 MBean 操作

5.6.2.2.1. RemoveAllMessages
从队列里移除（并删除）所有消息。

警告

请小心使用这个操作。它将永久地删除队列里的所有消息。

.

5.6.2.2.2. ListAllMessages
列出队列里当前的所有消息

这个操作有两个重载版本：其中一个以 JMS selector 为参数，另外一个则不以其为参数。通过使用 selector，你可以获取队列里符合标准的消息子集。

5.6.2.2.3. ListDurableMessages
和 listAllMessages 一样但它只列出持久性消息

这个操作有两个重载版本：其中一个以 JMS selector 为参数，另外一个则不以其为参数。通过使用 selector，你可以获取队列里符合标准的消息子集。

5.6.2.2.4. ListNonDurableMessages
和 listAllMessages 一样但它只列出非持久性消息

这个操作有两个重载版本：其中一个以 JMS selector 为参数，另外一个则不以其为参数。通过使用 selector，你可以获取队列里符合标准的消息子集。

5.6.2.2.5. ResetMessageCounter
将消息计数器清零。

5.6.2.2.6. ResetMessageCounterHistory
将消息计数器历史记录清零。

5.6.2.2.7. ListMessageCounterAsHTML
以 HTML 形式列出消息计数器

5.6.2.2.8. ListMessageCounterHistoryAsHTML
以 HTML 形式列出消息计数器历史记录

5.6.3. 配置主题

5.6.3.1. Topic MBean 的 MBean 属性

5.6.3.1.1. Name
状态的名称

5.6.3.1.2. JNDIName
主题所绑定的 JNDI 名称

5.6.3.1.3. DLQ
用于该主题的 DLQ。它将覆盖 ServerPeer 配置里的任何值。

5.6.3.1.4. ExpiryQueue
用于该主题的 Expiry queue。它将覆盖 ServerPeer 配置里的任何值。

5.6.3.1.5. RedeliveryDelay
用于该主题的重递送延迟。它将覆盖 ServerPeer 配置里的任何值。

5.6.3.1.6. MaxDeliveryAttempts
在把消息发送到 DLQ 之前，尝试递送消息的最大次数。如果为 -1（缺省值），ServerPeer 配置里值将被使用。任何其他值将覆盖 ServerPeer 配置里的值。

5.6.3.1.7. 目的地的安全性配置
SecurityConfig - 允许你决定哪些角色可以在目的地读、写和创建。它具有和 JBossMQ 目的地的安全配置完全相同的语法。

SecurityConfig 元素里应该包含一个 **<security>** 元素。而 **<security>** 元素可以包含多个 **<role>** 元素。每个 **<role>** 元素都定义特定角色的访问权限。

如果 **read** 属性为 **true**，那么这个角色可以读取（创建消费者、接收消息或浏览消息）目的地。

如果 **write** 属性为 **true**，那么该角色可以写这个目的地（创建生产者或发送消息）。

如果 **create** 属性为 **true**，那么该角色可以在这个目的地上创建持久性订阅。

请注意目的地的安全性配置是可选的。如果没有指定 **SecurityConfig** 元素，那么 Server Peer 里的缺省安全配置将被使用。

5.6.3.1.8. 目的地的页面参数
'Pageable Channels' 是 JBoss Messaging 里的一个复杂的新功能。

如果你的应用程序需要支持超大的对列或可能包含数百万条消息的订阅，那么一次性进行存储就不可能。

JBoss Messaging 没有解决这个问题，但它可以让你指定可以一次性存储在内存里的每个队列或主题的消息的最大个数。JBoss Messaging 然后以块的形式透明地将这些消息分页，允许队列和订阅非常大而不影响性能。

这已经在基本的硬件上使用超过一千万零两千条消息进行了测试，且有可能扩展至更大的数目。

其各个参数为：

FullSize - 这是内存里的队列或主题订阅在任何时刻可容纳的最大消息数目。实际的队列或订阅可以容纳比这更多的消息，因为在添加或消费消息时可以按需要将这些消息分页至存储里。

PageSize - 当从队列或订阅加载消息时，这是在某个操作里可以预加载的最大消息数目。

DownCacheSize - 当把消息从队列里分页至存储里时，在实际上写入前它们首先进入一个 "Down Cache" 里。这使写入成为单个操作，因此提高了性能。这个设置决定了 Down Cache 在冲刷消息至存储前可容纳的消息的最大数目。

如果没有指定 **FullSize**、**PageSize** 或 **DownCacheSize**，它们的缺省值将分别是 75000、2000、

2000。

如果你向指定用于临时队列的页面参数，那么你需要在合适的连接工厂上对它们进行指定。详情请参考连接工厂的配置。

5.6.3.1.9. CreatedProgrammatically

如果主题在程序里被成功创建，返回 **true**。

5.6.3.1.10. MaxSize

主题订阅里的消息的最大数量。任何超过这个数量的消息将被忽略。缺省值为 **-1**，表示不加限制。

5.6.3.1.11. Clustered

群集的目的地将它必须设置为 **true**。

5.6.3.1.12. MessageCounterHistoryDayLimit

消息计数器历史记录保留的最多天数。它将覆盖 ServerPeer 配置里的任何值。

5.6.3.1.13. MessageCounters

返回该主题的订阅的消息计数器列表。

5.6.3.1.14. AllMessageCount

返回该主题的订阅的所有消息的个数。

5.6.3.1.15. DurableMessageCount

返回该主题的订阅的所有持久性消息的个数。

5.6.3.1.16. NonDurableMessageCount

返回该主题的订阅的所有非持久性消息的个数。

5.6.3.1.17. AllSubscriptionsCount

该主题上所有订阅的个数。

5.6.3.1.18. DurableSubscriptionsCount

该主题上所有持久性订阅的个数。

5.6.3.1.19. NonDurableSubscriptionsCount

该主题上所有非持久性订阅的个数。

5.6.3.2. Topic MBean 的 MBean 操作

5.6.3.2.1. RemoveAllMessages

从该主题的订阅里移除（并删除）所有消息。

警告

请小心使用这个操作。它将永久地删除主题里的所有消息。

5.6.3.2.2. ListAllSubscriptions

列出该主题的所有订阅。

5.6.3.2.3. ListDurableSubscriptions

列出该主题的所有持久性订阅。

5.6.3.2.4. ListNonDurableSubscriptions

列出该主题的所有非持久性订阅。

5.6.3.2.5. ListAllSubscriptionsAsHTML

以 HTML 形式列出该主题的所有订阅。

5.6.3.2.6. ListDurableSubscriptionsAsHTML

以 HTML 形式列出该主题的所有持久性订阅。

5.6.3.2.7. ListNonDurableSubscriptionsAsHTML

以 HTML 形式列出该主题的所有非持久性订阅。

5.6.3.2.8. ListAllMessages

列出指定订阅的所有消息。

这个操作有两个重载版本。其中一个使用 selector 而另外一个则不使用。通过指定 selector，你可以限制返回的消息。

5.6.3.2.9. ListNonDurableMessages

列出指定订阅的所有非持久性消息。

这个操作有两个重载版本。其中一个使用 selector 而另外一个则不使用。通过指定 selector，你可以限制返回的消息。

5.6.3.2.10. ListDurableMessages

列出指定订阅的所有持久性消息。

这个操作有两个重载版本。其中一个使用 selector 而另外一个则不使用。通过指定 selector，你可以限制返回的消息。

5.7. 配置连接工厂

在启动时，JBoss Messaging 用缺省的配置在 JNDI 里构建了两个连接工厂。

第一个连接工厂是缺省的非群集连接工厂，它绑定在下面的 JNDI 上下文里：**`/ConnectionFactory`**，**`/XAConnectionFactory`**，**`java:/ConnectionFactory`**，**`java:/XAConnectionFactory`**。这个连接工厂用来兼容原来针对 JBoss MQ 编写的没有自动失效切换或负载平衡功能的应用程序。如果你不要求客户端的自动失效切换或负载平衡，你应该使用它。

第二个连接工厂是缺省的群集连接工厂，它绑定在下面的 JNDI 上下文里：**`/ClusteredConnectionFactory`**，**`/ClusteredXAConnectionFactory`**，**`java:/ClusteredConnectionFactory`**，**`java:/ClusteredXAConnectionFactory`**。

你可能想配置其他的连接工厂，例如，你想为连接工厂提供缺省的客户 ID、或者想绑定到不同的 JNDI 位置、或让不同的连接工厂使用不同的传输协议、或选择性地为某个特定的连接工厂启用或禁止负载平衡和/或自动失效切换。在 **`connection-factories-service.xml`** 里添加新的 ConnectionFactory MBean 配置就可以部署新的连接工厂。

你也可以创建一个完全新的服务部署描述符 **`xxx-service.xml`** 并部署在 **`$JBOSS_HOME/server/messaging/deploy`** 里。

通过设置相关属性，连接工厂也可以支持自动失效切换和/或负载平衡。

下面是一个连接工厂配置的例子：

```
<mbean code="org.jboss.jms.server.connectionfactory.ConnectionFactory"
       name="jboss.messaging.connectionfactory:service=MyConnectionFactory"
       xmbean-dd="xmdesc/ConnectionFactory-xmbean.xml">
       <depends optional-attribute-name="ServerPeer">
                  jboss.messaging:service=ServerPeer
       </depends>
       <depends optional-attribute-name="Connector">
                  jboss.messaging:service=Connector,transport=bisocket
       </depends>
       <depends>jboss.messaging:service=PostOffice</depends>

       <attribute name="JNDIBindings">
          <bindings>
             <binding>/MyConnectionFactory</binding>
             <binding>/factories/cf</binding>
          </bindings>
       </attribute>

       <attribute name="ClientID">myClientID</attribute>

       <attribute name="SupportsFailover">true</attribute>

       <attribute name="SupportsLoadBalancing">false</attribute>

       <attribute
name="LoadBalancingFactory">org.acme.MyLoadBalancingFactory</attribute>

       <attribute name="PrefetchSize">1000</attribute>

       <attribute name="SlowConsumers">false</attribute>

       <attribute name="StrictTck">true</attribute>

       <attribute name="DefaultTempQueueFullSize">50000</attribute>

       <attribute name="DefaultTempQueuePageSize">1000</attribute>

       <attribute name="DefaultTempQueueDownCacheSize">1000</attribute>

       <attribute name="DupsOKBatchSize">10000</attribute>
   </mbean>
```

上面的例子将创建一个带有预配置客户 ID **myClientID** 的连接工厂并绑定在 JNDI 树的两个地方：**/MyConnectionFactory** 和 **/factories/cf**。这个连接工厂覆盖了 PreFetchSize、DefaultTempQueueFullSize、DefaultTempQueuePageSize、DefaultTempQueueDownCacheSize 和 DupsOKBatchSize 的缺省值。它将使用缺省的远程连接器（remoting connector）。要使用不同的远程连接器，你可以修改 **Connector** 属性来指定连接器的服务名。

5.7.1. ConnectionFactory MBean 的 MBean 属性

5.7.1.1. ClientID

连接工厂可用 client id 进行预配置。使用这个连接工厂创建的任何连接都将获得这个 ID。

5.7.1.2. JNDIBindings

这个连接工厂使用的 JNDI 绑定的列表

5.7.1.3. PrefetchSize

这个参数为消费者（Consumer）流控制指定窗口大小。这个窗口大小决定服务器可以发送到某个消费者而不阻塞的消息数量。每个消费者都维护一个它从中消费的消息缓冲。请注意，TCP 也实现自己的流控制，所以如果你把它设置为过大的值，TCP 窗口大小限制可能在 **prefetchSize** 之前就被突破，这样会导致写

阻塞。

5.7.1.4. SlowConsumers

如果存在非常慢的消费者，你可能希望它们不缓冲任何消息，因为这会阻止这些消息被更快的消费者所消费。把它设置为 true 和设置 **`PrefetchSize`** 为 1 是相等的。

5.7.1.5. StrictTck

如果你期望 TCK 所要求的严格的 JMS 行为的话，你可以将它设置为 true。

5.7.1.6. 临时队列的页面参数

DefaultTempQueueFullSize、DefaultTempQueuePageSize、DefaultTempQueueDownCacheSize 是决定缺省页面参数的可选属性，它们用于连接工厂创建的作用域为临时目的连接。关于这些参数的更多信息，请参考 paging channels 部分的内容。如果忽略的话，它们的缺省值为 200000, 2000 和 2000。

5.7.1.7. DupsOKBatchSize

当使用带有 DUPS_OK_ACKNOWLEDGE 确认模式的会话时，设置它可以决定在发送前确认多少次后就在本地进行缓冲。它的缺省值为 **`2000`**。

5.7.1.8. SupportsLoadBalancing

当使用带有群集 JBoss Messaging 安装的连接工厂时，你可以选择是否启用客户端连接的负载平衡。这可以通过设置连接工厂的 supportsLoadBalancing 属性来实现。

如果连接工厂启用了负载平衡，那么用它创建的任何连接都将在群集里的节点间进行负载平衡。一旦在某个节点上创建了一个连接，它就会呆在那个节点上。

连接使用何种策略进行负载平衡是由 LoadBalancingFactory 属性决定的

它的缺省值为 **`false`**

5.7.1.9. SupportsFailover

当使用带有群集 JBoss Messaging 安装的连接工厂时，你可以选择是否启用客户端自动失效切换。这可以通过设置连接工厂的 supportsFailover 属性来完成。

如果连接工厂启用了自动失效切换，且某个连接出现了连接问题，JBoss Messaging 将自动和透明地切换至群集里的另外一个节点上。

透明的失效切换意味着客户可以象故障发生前一样地使用会话、消费者、生产者和连接对象。

如果不要求自动失效切换，那可以把这个属性设置为 false。如果禁用了自动失效切换，那么程序代码要在同步 JMS 操作里捕获异常或安装 JMS ExceptionListener 来异步捕获异常。当某个连接被捕获时，客户端代码应该查找一个使用 HAJNDI 的连接工厂并用它来重新创建连接。

它的缺省值为 **`false`**

5.7.1.10. DisableRemotingChecks

在缺省情况下，在部署一个连接工厂时，JBoss Messaging 将检查对应的 JBoss Remoting Connector 是否具有某些敏感的（"sensible"）值。JBoss Messaging 对这些值非常敏感且很少有修改这些值的时候。要禁用这样的检查，可将其设置为 false。

警告

我们通常没什么理由禁用这个检查。请在有明确目的时才这么做。

它的缺省值为 **`false`**

5.7.1.11. LoadBalancingFactory

如果你在使用带有客户端负载平衡的连接工厂，你可以通过覆盖这个属性来指定实施方法。这个值必须对应

实现 org.jboss.jms.client.plugin.LoadBalancingFactory 接口的类名。

它的缺省值是 org.jboss.jms.client.plugin.RoundRobinLoadBalancingFactory，这将以 round-robin 模式在群集里平衡连接负载。

5.7.1.12. 连接器

它指定连接工厂使用的远程连接器。不同的连接工厂可以使用不同的连接器。

例如，你部署的连接工厂可以创建使用 HTTP 传输在服务器和客户端间通信的连接、也可以创建使用双向套接字传输通信的连接。

5.8. 配置远程连接器

对于所有的客户端和服务器之间的通信，JBoss Messaging 都使用 JBoss Remoting。关于 JBoss Remoting 的功能和配置，请参考 JBoss Remoting 文档。

缺省的配置包括单一的远程连接器，它被单一的缺省连接工厂使用。你可以配置每个连接工厂都使用自己的连接器。

缺省的连接器使用远程双向套接字传输（bisocket transport）。双向套接字传输基于 TCP 套接字，它只在服务器端侦听和接受连接。也就是说，连接总是从客户端发起。这意味着它适合于典型的防火墙环境里，因为此时只允许服务器上的转入连接。或者说只允许客户端的转出连接。

如果需要更高级别的安全性，我们可以配置双向套接字传输使用 SSL。

另外一种被支持的传输协是 HTTP 传输。它使用 HTTP 协议在服务器和客户端之间通信。客户端通过定期地轮询服务器来接收数据。这种传输特别适合于在客户端和服务器之间存在防火墙的情况，此时只允许服务器上的转入 HTTP 通信。请注意，这种传输不具有和双向传输相同的性能，这是由于轮询的性质以及 HTTP 协议的采用。也请注意它不是为高负载环境设计的。

JBoss Messaging 目前不支持其他远程传输模式。

你可以在这里查看远程配置：

<JBoss>/server/<YourMessagingServer>/deploy/jboss-messaging.sar/remoting-bisocket-service.xml

下面是一个关于双向远程传输的配置示例：

```
         <config>
            <invoker transport="bisocket">

               <!-- There should be no reason to change these parameters -
warning!
                    Changing them may stop JBoss Messaging working correctly -->
               <attribute name="marshaller"
isParam="true">org.jboss.jms.wireformat.JMSWireFormat</attribute>
               <attribute name="unmarshaller"
isParam="true">org.jboss.jms.wireformat.JMSWireFormat</attribute>
               <attribute name="dataType" isParam="true">jms</attribute>
               <attribute name="socket.check_connection"
isParam="true">false</attribute>
               <attribute name="timeout" isParam="true">0</attribute>
               <attribute
name="serverBindAddress">${jboss.bind.address}</attribute>
               <attribute name="serverBindPort">4457</attribute>
               <attribute name="clientSocketClass"
isParam="true">org.jboss.jms.client.remoting.ClientSocketWrapper</attribute>
               <attribute name="serverSocketClass"
isParam="true">org.jboss.jms.server.remoting.ServerSocketWrapper</attribute>
               <attribute name="numberOfCallRetries" isParam="true">1</attribute>
               <attribute name="pingFrequency"
isParam="true">214748364</attribute>
               <attribute name="pingWindowFactor" isParam="true">10</attribute>
               <attribute
name="onewayThreadPool">org.jboss.jms.server.remoting.DirectThreadPool</attribute>

               <!-- Periodicity of client pings. Server window by default is
twice this figure -->
               <attribute name="clientLeasePeriod"
isParam="true">10000</attribute>

               <!-- Number of seconds to wait for a connection in the client pool
to become free -->
               <attribute name="numberOfRetries" isParam="true">10</attribute>

               <!-- Max Number of connections in client pool. This should be
significantly higher than
                    the max number of sessions/consumers you expect -->
               <attribute name="clientMaxPoolSize" isParam="true">200</attribute>

               <!-- Use these parameters to specify values for binding and
connecting control connections to
                    work with your firewall/NAT configuration
               <attribute name="secondaryBindPort">xyz</attribute>
               <attribute name="secondaryConnectPort">abc</attribute>
               -->

            </invoker>
            <handlers>
               <handler
subsystem="JMS">org.jboss.jms.server.remoting.JMSServerInvocationHandler</handler>
            </handlers>
         </config>
```

请注意，除非你有明确的目的，否则不要修改其中一些属性。我们将讨论你可能需要修改的属性：

- clientLeasePeriod - 客户端定期地往服务器发送 heartbeat 来告知它处于活动状态。如果服务器在一段时间后没有接收到 heartbeat，那么它将关掉连接并删除这个会话对应的所有服务器上的资源。clientLeasePeriod 决定了发送 heartbeat 的间隔。如果服务器在 2 * clientLeasePeriod 毫秒后还没有收到 heartbeat 信息，它缺省将关闭这个客户连接。实际的因子会根据系统负载自动进行调节。这个值以毫秒为单位。它的缺省值是 10000 毫秒。
- numberOfRetries - 它对应 JBoss Remoting 在等待客户连接池有空闲的连接时阻塞的时间（秒）。如果有数目众多的会话并行地访问服务器，且你遇到不能从连接池获得连接的问题，那么你就可以考虑增大

这个值。

- clientMaxPoolSize - JBoss Remoting 维护一个客户端的对应服务请求的 TCP 连接池。如果有数目众多的会话并行地访问服务器，且你遇到不能及时地从连接池获得连接的问题，那么你就可以考虑增大这个值。
- secondaryBindPort - 双向传输使用控制连接（control connection）在服务器和客户端之间传递控制消息。如果你想在防火墙后工作，你可能需要按照防火墙配置为这个连接指定特殊的值。它是第二个 ServerSocket 所绑定的地址。
- secondaryConnectPort - 这是客户端用户连接的端口。你可能希望指定它来允许客户使用 NAT 路由器。
- maxPoolSize - 这是服务器端用于处理请求的线程数量。

在缺省情况下，JBoss Messaging 绑定到 ${jboss.bind.address}，它可用 ./run.sh -c <yourconfig> -b yourIP 进行定义。

如果你想使用不同的通信端口，你可以修改 remoting-bisocket-service.xml。

警告

除了 clientLeasePeriod、clientMaxPoolSize、maxRetries、numberOfRetries、secondaryBindPort 和 secondaryConnectPort 以外，我们通常不需修改 bisocket 或 sslbisocket 连接器配置里的其他值。修改这些值可能导致 JBoss Messaging 不能正常运行。

5.9. ServiceBindingManager

如果你要使用 JBoss AS ServiceBindingManager 来提供带有不同端口范围俄的不同服务器，你必须确保 ServiceBindingManager xml 文件里的 JBoss Messaging 部分里的 JBoss Messaging 远程配置完全符合 remoting-bisocket-service.xml 里的配置。

如果你在较旧版本的 JBAS 里使用更新版本的 JBM，那么 AS 发行版本里的例子可能是过期的。因此相关部分急需用 JBM 发行版本里的远程配置进行覆盖。

第 6 章 JBoss Messaging 群集配置

在大部分情况下，JBoss Messaging 群集应该可以直接运行而不需要修改任何配置。然而为每个节点都分配一个唯一的服务器 ID 是很重要的。

警告

对于基于群集的安装来说，群集里的每个节点都有一个共享的数据库是强制的。JBoss AS 缺省使用的 HSQLDB 是一个本地的共享数据库。因此，如果要使用群集，你必须用另外的数据库来替换它。

如果你想在同一台机器上运行多个 JBoss Messaging 实例，如出于开发目的，你可以用 ServiceBindingManager 来实现：

- 在 $JBOSS_CONFIG/conf/jboss-service.xml 里注释 binding manager 服务
- 指定端口范围（如ports-01、ports-02 等）
- 来看看 $JBOSS_HOME/docs/examples/binding-manager/sample-bindings.xml。针对每个端口范围的 JBoss Remoting 配置都应该类似于：

```
      <service-config
name="jboss.messaging:service=Connector,transport=bisocket"

delegateClass="org.jboss.services.binding.AttributeMappingDelegate">
         <delegate-config>
            <attribute name="Configuration"><![CDATA[
    <config>
            <invoker transport="bisocket">
               <attribute name="marshaller" isParam="true">
                  org.jboss.jms.wireformat.JMSWireFormat
               </attribute>
               <attribute name="unmarshaller" isParam="true">
                  org.jboss.jms.wireformat.JMSWireFormat
               </attribute>
               <attribute name="dataType" isParam="true">jms</attribute>
               <attribute name="socket.check_connection"
isParam="true">false</attribute>
               <attribute name="timeout" isParam="true">0</attribute>
               <attribute
name="serverBindAddress">${jboss.bind.address}</attribute>
               <attribute name="serverBindPort">4657</attribute>
               <attribute name="leasePeriod">10000</attribute>
               <attribute name="clientSocketClass" isParam="true">
                  org.jboss.jms.client.remoting.ClientSocketWrapper
               </attribute>
               <attribute name="serverSocketClass">
                  org.jboss.jms.server.remoting.ServerSocketWrapper
               </attribute>
               <attribute name="numberOfRetries" isParam="true">1</attribute>
               <attribute name="numberOfCallRetries"
isParam="true">1</attribute>
               <attribute name="clientMaxPoolSize"
isParam="true">50</attribute>
            </invoker>
            <handlers>
               <handler subsystem="JMS">
                  org.jboss.jms.server.remoting.JMSServerInvocationHandler
               </handler>
            </handlers>
         </config>
            ]]></attribute>
         </delegate-config>
         <binding port="4657"/>
      </service-config>
```

警告

你必须确保这个配置（如上所示）和 **remoting-bisocket-service.xml** 里的一样，除了每个端口范围的实际 serverBindPort 必须不同。请注意，JBAS 4.2.0 附带的**sample-bindings.xml** 里的缺省 JBoss Messaging 服务 binding manager 绑定是过期的，你需要从 **remoting-bisocket-service.xml** 里把相关配置复制过来。

你应该确保每个节点使用不同的端口范围。

6.1. 唯一的 server peer id

每个部署的节点必须具有唯一的 ID，这包括特殊局域网群集里的节点，以及只通过消息桥链接的节点。

注意

确保 **messaging-service.xml** 里针对群集里的每个节点的 **ServerPeerID** MBean 属性值是唯一的。**ServerPeerID** 必须为整型值。

6.2. 群集目的地

JBoss Messaging 透明地群集了 JMS 队列和主题。发送到分布式队列或主题的消息也可在其他节点上获取。要指定特定节点为群集的，只要设置目的地部署描述符里的 clustered 属性为 true 就可以了。

JBoss Messaging 在节点间平衡消息的负载，这是为了有效地调整群集里较快和较慢消费者的负载。

如果你不想在节点间重新分发消息，但仍然想保留群集目的地的其他特点，你可以通过不指定 server peer 上的 ClusterPullConnectionFactoryName 属性来实现这一点。

6.3. 群集的持久性订阅

JBoss Messaging 的持久性订阅（durable subscription）也可以进行群集。这意味着多个订阅者可以在不同节点里访问相同的持久性订阅。如果持久性订阅的主题是群集的，那么它也将是群集的。

6.4. 群集的临时目的地

JBoss Messaging 也支持群集的临时主题和队列。如果邮局（Post Office）是群集的，那么所有的临时主题和队列将被群集。

6.5. 非群集的服务器

如果你不希望你的节点参与某个群集，或者你只有一个非群集的服务器，你可以设置邮局的 clustered 属性为 false。

6.6. 管理群集里的顺序

如果你想对消息应用严格的 JMS 顺序，亦即特定的 JMS 消费者（consumer）将按照和生产者（producer）产生消息的顺序来消费消息，你可以设置 server peer 里的 DefaultPreserveOrdering 属性为 true。它的缺省值是 false。设置它为 true 的副作用是消息不能自由地在群集里进行分发。

6.7. Idempotent 操作

如果对发送持久性消息到持久性目的地的调用成功返回，那么你可以确认消息已经被持久化了。然而，如果这个调用没有成功返回，例如有异常被抛出，你并不能确认消息没有被持久化。因为故障可能发生在消息已经被持久化后，而在响应调用者之前。这是任何 RPC 类型的调用的一个共同问题：调用没有返回并不代表它没有成功执行。不管它是一个 web 服务调用、HTTP 的 GET 请求，还是 EJB 调用，都是如此。编程的诀窍是使这些操作成为幂等的（idempotent），也就是它们可以重复而不会使系统的状态不一致。对于消息系统来说，你可以在应用程序级别上来实现这一点，也就是检查重复的消息并在其到达时丢弃它们。对重复的检查是一种非常强大的技术，在许多情况下它可以消除对 XA 事务的需要。

而在群集环境里，JBM 将缺省自动检测重复的消息。

6.8. 群集的连接工厂

如果连接工厂的 supportsLoadBalancing 属性为 true，那么它将在可用的服务器间循环地尝试创建连接。第一个尝试的节点将被随机地选择。

如果连接工厂的 supportsFailover 属性为 true，那么自动失效切换将被启用。在故障发生是，这将自动地从一个服务器切换到另外一个服务器，而对于用户来说是透明的。

如果不要求自动失效切换或者你想手工来完成（JBoss MQ 风格），你可以将其设置为 false，而且你可以为连接提供一个标准的 JMS ExceptionListener，它在连接发生故障时将被调用。然后你需要手工关闭这个连接，从 HA JNDI 里查找一个新的连接工厂并重新创建连接。

第 7 章 JBoss Messaging XA 恢复的配置

本节描述了如何在 JBoss AS 4.2.0 里配置 JBoss Transactions 来处理 JBoss Messaging 资源的 XA 恢复。

你可以很容易地配置 JBoss Transactions Recovery Manager 来轮询和恢复 JBoss Messaging 的 XA 资源，这为事务提供了极高级别的持久性。

启用 JBoss Transactions Recovery Manager 来恢复 JBoss Messaging 资源很简单，你只需在 ${JBOSS_CONFIG}/conf/jbossjta-properties.xml 里添加一行就可以了。

下面是一个已经添加了这一行的 jbossjta-properties.xml 文件的示例片段（注意这里并未显示整个文件）：

```
     <properties depends="arjuna" name="jta">
        <!--
        Support subtransactions in the JTA layer?
        Default is NO.
        -->
        <property name="com.arjuna.ats.jta.supportSubtransactions" value="NO"/>
        <property name="com.arjuna.ats.jta.jtaTMImplementation"

value="com.arjuna.ats.internal.jta.transaction.arjunacore.TransactionManagerImple"/>
        <property name="com.arjuna.ats.jta.jtaUTImplementation"

value="com.arjuna.ats.internal.jta.transaction.arjunacore.UserTransactionImple"/>
        <!--
           *** Add this line to enable recovery for JMS resources using
DefaultJMSProvider ***
        -->
        <property
name="com.arjuna.ats.jta.recovery.XAResourceRecovery.JBMESSAGING1"

value="org.jboss.jms.server.recovery.MessagingXAResourceRecovery;java:/DefaultJMSPr
ovider"/>

    </properties>
```

在上面的例子里，Recovery Manager 将试图用 JMSProviderLoader "DefaultJMSProvider" 恢复 JMS 资源。

DefaultJMSProvider 是 JBoss AS 附带的缺省 JMS 提供者的加载器，它在 jms-ds.xml（群集环境里使用 hajndi-jms-ds.xml）里进行定义。如果你想用不同的 JMS 提供者加载器进行恢复 — 例如对应远程 JMS 提供者的加载器，你只需要再添加一行来指定远程 JSM 提供者（如其 MBean 配置里所指定的）的名字来代替 DefaultJMSProvider 就可以了。

对于你添加的每一行，名字必须是唯一的，所以你可以指定 "com.arjuna.ats.jta.recovery.XAResourceRecovery.JBMESSAGING1", "com.arjuna.ats.jta.recovery.XAResourceRecovery.JBMESSAGING2, ..." 等等。

实际上，恢复也可以使用实现了可恢复 XAResources（也就是需要正确实现 XAResource.recover()）的任何 JMS 提供者来进行，而不仅仅是 JBoss Messaging。

请注意，要配置 Recovery Manager 来恢复群集里任何节点的事务，你有必要在配置文件里用一行为每个节点进行指定。

第 8 章 JBoss Messaging 消息桥的配置

8.1. 消息桥概述

JBoss Messaging 包含一个功能完整的消息桥。

消息桥的功能是从源队列或主题里提取消息，然后发送到通常在另外一个服务器上的目标队列或主题。

源和目标服务器不需要位于同一个群集里，消息桥适合于把消息可靠地从一个群集发送到另外一个群集，例如跨越网络连接可能不稳定的广域网。

消息桥部署在 JBoss AS 实例里。这个实例可以是源或目标服务器里的实例，也可以是第三个独立的 JBoss AS 实例。

消息桥作为 MBean 部署在 JBoss AS 实例里。部署很简单 — 只要把 MBean 部署描述符放入包含 JBoss Messaging 的 JBoss 配置的 deploy 目录下就可以了。

docs/example/bridge 里的例子演示了部署在 JBoss AS 里的一个简单的消息桥，它把消息从源服务器一移动到目标目的地里。

消息桥也可以用转移非 JBoss Messaging JMS 服务器的消息，只要这个服务器兼容 JMS 1.1。

消息桥具有处理故障的能力，如果源或目标服务器连接丢失了（例如网络故障），消息桥将试图恢复连接，直到双方都在线为止。当连接恢复后，消息桥将继续正常的操作。

消息桥可以用可选的 JMS Selector 进行配置，所以它只提取符合该 JMS selector 的消息。

它可以配置成从队列或主题里提取消息。当它从主题里提取消息时，你还可以配置它使用非持久的或持久的订阅。

消息桥可以使用三个级别的服务品质来转发消息，它们是：

- QOS_AT_MOST_ONCE

  使用这种 QoS 模式，消息最多可以从源服务器到达目的地一次。在发送到目的地之前，消息从源服务器里提取并确认。因此，如果在从源服务器提取和到达目的地之间如果发生故障，消息就有可能会丢失。因此，递送最多只能进行一次。这种模式对持久性和非持久性消息都适用。
- QOS_DUPLICATES_OK

  使用这种 QoS 模式，消息从源服务器提取并在成功送达目的地后确认收到。因此，如果故障发生在消息发送到目的地之后、而在确认收到之前，系统恢复后消息可能会被再发送一次。也就是说，在发生故障时，目的地可能收到重复的消息。持久性和非持久性消息都可以使用这种模式。
- QOS_ONCE_AND_ONLY_ONCE

  这种 QoS 模式确保消息将只到达目的地一次（有时候这种模式被称为"exactly once"）。如果源和目的地都在同一个 JBoss Messaging 服务器实例上，那么这可以通过在同一个本地事务内发送和确认消息来实现。如果源和目的地在不同的服务器上，这可以通过在 JTA 事务里支持发送和消费会话来实现。JTA 事务由 JBoss Transactions JTA 实现所控制，它是一个可完全恢复的事务管理者（Transaction Manager），因此提供了级别非常高的持久性。如果 JTA 时必需的，那么这两个连接工厂都需要是 XAConnectionFactory 的实现。这种模式仅适用于持久性消息。它可能是最慢的模式，因为它要求登录事务管理者和用于恢复的资源。如果你需要这种级别的 QoS，请确保启用 JBoss Transactions 的 XA 恢复功能。

注意

对于特殊的应用程序，我们可能提供 once and only once 模式而无需使用 QOS_ONCE_AND_ONLY_ONCE QoS 级别。这可以通过使用 QOS_DUPLICATES_OK 模式并在目的地检查重复消息来实现。我们也可以在应用程序级别来实现：在磁盘上保留一个接收到的消息的 ID 的缓存并比较收到的消息。因为这个缓存将只在某段时间内有效，所以这个方法并不象 QOS_ONCE_AND_ONLY_ONCE 模式一样无懈可击，但也不失为一个好的选择，这取决于特殊应用程序的需要。

8.2. 部署消息桥

只要把相关的 MBean 描述符放入包含 JBoss Messaging 的 JBoss AS 的 deploy 目录下，你就可以轻易地部署消息桥了。

8.3. 消息桥的配置

本节我们将描述如何配置消息桥

下面是一个消息桥配置的例子，它展示了所有的属性。请注意，由于有些配置不适合全部一次被指定，所以它们被注释掉了。这要根据你的需要来取舍。

```
   <mbean code="org.jboss.jms.server.bridge.BridgeService"
          name="jboss.messaging:service=Bridge,name=TestBridge"
          xmbean-dd="xmdesc/Bridge-xmbean.xml">

      <!-- The JMS provider loader that is used to lookup the source destination -
->
      <depends optional-attribute-name="SourceProviderLoader">
          jboss.messaging:service=JMSProviderLoader,name=JMSProvider</depends>

      <!-- The JMS provider loader that is used to lookup the target destination -
->
      <depends optional-attribute-name="TargetProviderLoader">
          jboss.messaging:service=JMSProviderLoader,name=JMSProvider</depends>

      <!-- The JNDI lookup for the source destination -->
      <attribute name="SourceDestinationLookup">/queue/A</attribute>

      <!-- The JNDI lookup for the target destination -->
      <attribute name="TargetDestinationLookup">/queue/B</attribute>

      <!-- The username to use for the source connection
      <attribute name="SourceUsername">bob</attribute>
      -->

      <!-- The password to use for the source connection
      <attribute name="SourcePassword">cheesecake</attribute>
      -->

      <!-- The username to use for the target connection
      <attribute name="TargetUsername">mary</attribute>
      -->

      <!-- The password to use for the target connection
      <attribute name="TargetPassword">hotdog</attribute>
      -->

      <!-- Optional: The Quality Of Service mode to use, one of:
           QOS_AT_MOST_ONCE = 0;
           QOS_DUPLICATES_OK = 1;
           QOS_ONCE_AND_ONLY_ONCE = 2; -->
      <attribute name="QualityOfServiceMode">0</attribute>

      <!-- JMS selector to use for consuming messages from the source
      <attribute name="Selector">specify jms selector here</attribute>
      -->

      <!-- The maximum number of messages to consume from the source
          before sending to the target -->
      <attribute name="MaxBatchSize">5</attribute>

      <!-- The maximum time to wait (in ms) before sending a batch to the target
          even if MaxBatchSize is not exceeded.
           -1 means wait forever -->
      <attribute name="MaxBatchTime">-1</attribute>

      <!-- If consuming from a durable subscription this is the subscription name
      <attribute name="SubName">mysub</attribute>
      -->

      <!-- If consuming from a durable subscription this is the client ID to use
      <attribute name="ClientID">myClientID</attribute>
      -->

      <!-- The number of ms to wait between connection retrues in the event
connections
          to source or target fail -->
```

```
      <attribute name="FailureRetryInterval">5000</attribute>

      <!-- The maximum number of connection retries to make in case of failure,
          before giving up -1 means try forever-->
      <attribute name="MaxRetries">-1</attribute>

      <!-- If true then the message id of the message before bridging will be
added
          as a header to the message so it is available to the receiver. Can then
be
          sent as correlation id to correlate in a distributed request-response --
>
      <attribute name="AddMessageIDInHeader">false</attribute>

    </mbean>
```

现在我们逐个来讨论这些属性

8.3.1. SourceProviderLoader

这是 JMSProviderLoader MBean 的对象名，消息桥将用它来查找源连接工厂和源目的地。

JBoss AS 缺省带有一个 JMSProviderLoader，它部署在 **`jms-ds.xml`** 文件里 — 这是缺省的本地 JMSProviderLoader（群集配置里的对应文件是 **`hajndi-jms-ds.xml`**）。

如果源目的地位于不同的服务器上或者甚至对应非 JBoss JMS 提供者，那你可以部署引用远程 JMS 提供者的其他 JMSProviderLoader MBean 实例并用属性引用它。然后消息桥将使用那个远程 JMS 提供者来联系源目的地。

请注意，如果你正在使用远程的非 JBoss Messaging 源或目标服务器，且你希望 once and only once 模式的递送，那么该远程的 JMS 提供者必须提供可远程运行的具备完整功能的 JMS XA 资源实现 — 有些非 JBoss JMS 提供者并不提供这样的资源。

8.3.2. TargetProviderLoader

这是 JMSProviderLoader MBean 的对象名，消息桥将用它来查找目标连接工厂和目标目的地。

JBoss AS 缺省带有一个 JMSProviderLoader，它部署在 **`jms-ds.xml`** 文件里 — 这是缺省的本地 JMSProviderLoader（群集配置里的对应文件是 **`hajndi-jms-ds.xml`**）。

如果你的目标目的地位于不同的服务器上或者甚至对应非 JBoss JMS 提供者，那你可以部署引用远程 JMS 提供者的其他 JMSProviderLoader MBean 实例并用属性引用它。然后消息桥将使用那个远程 JMS 提供者来联系目标目的地。

请注意，如果你正在使用远程的非 JBoss Messaging 源或目标服务器，且你希望 once and only once 模式的递送，那么该远程的 JMS 提供者必须提供可远程运行的具备完整功能的 JMS XA 资源实现 — 有些非 JBoss JMS 提供者并不提供这样的资源。

8.3.3. SourceDestinationLookup

这是用于使用 SourceProviderLoader 的源目的地的完整的 JNDI 查找

例如 /queue/mySourceQueue

8.3.4. TargetDestinationLookup

这是用于使用 TargetProviderLoader 的目标目的地的完整的 JNDI 查找

例如 /topic/myTargetTopic

8.3.5. SourceUsername

这个可选属性是创建源连接时所使用的用户名

8.3.6. SourcePassword
这个可选属性是创建源连接时所使用的密码

8.3.7. TargetUsername
这个可选属性是创建目标连接时所使用的用户名

8.3.8. TargetPassword
这个可选属性是创建源连接时所使用的密码

8.3.9. QualityOfServiceMode
这个整型值代表服务模式的级别

其可能的取值为：

- QOS_AT_MOST_ONCE = 0
- QOS_DUPLICATES_OK = 1
- QOS_ONCE_AND_ONLY_ONCE = 2

具体解释请参考 第 8.1 节 “消息桥概述”。

8.3.10. Selector
这个可选属性可以包含一个用于从源目的地提取消息的 JMS selector 表达式。只有匹配这个表达式的消息才会从源服务器转移到目标目的地。

请注意，在源主题订阅上应用 Selector 比在源队列消费者总是具有更好的性能。

Selector 表达式必须遵循 JMS selector 语法
：http://java.sun.com/j2ee/1.4/docs/api/javax/jms/Message.html。

8.3.11. MaxBatchSize
这个属性指定在批量发送至目标目的地前从源目的地提取的消息的最大数量，它的值必须大于 1。

8.3.12. MaxBatchTime
这个属性指定在批量发送到目标前等待的最大毫秒数，即使消费的消息数量没有达到 MaxBatchSize。它的值为 -1 则表示“永远等待”，而 >=1 则表示实际的时间。

8.3.13. SubName
如果源目的地是一个主题，且你想用持久性订阅（Durable Subscription）从这个主题消费消息，那么这个属性就是这个订阅的名字。

8.3.14. ClientID
如果源目的地是一个主题，而你想用一个持久性订阅从这个主题消费消息，那么这个属性就是用来创建/查找该订阅的 JMS 客户 ID。

8.3.15. FailureRetryInterval
它表示当消息桥检测到连接发生了故障时，在试图与源或目标服务器重建连接前等待的毫秒数。

8.3.16. MaxRetries
它表示当消息桥检测到连接发生了故障时，与源或目标服务器重建连接的最多尝试次数。消息桥在尝试这个次数后将放弃。-1 表示“永远地尝试”。

8.3.17. AddMessageIDInHeader
它如果为 true，那么原始消息的 ID 将附加在发送到目的地的消息的消息头部 JBossMessage.JBOSS_MESSAGING_BRIDGE_MESSAGE_ID_LIST。如果这个消息被多次转发，那么每

个 message-id 都将被附加。这使分布式的请求-应答（request-response）模式被使用。

修订历史记录

修订 4.3.0-3.402	**Fri Oct 25 2013**	**Rüdiger Landmann**
Rebuild with Publican 4.0.0		
修订 4.3.0-3.1	**2013-06-11**	**Misty Stanley-Jones**
Rebuild for updated legal template		
修订 4.3.0-3	**2012-07-18**	**Anthony Towns**
Rebuild for Publican 3.0		
修订 4.3-0	**Wed Sep 15 2010**	