7.5

规划 *IBM WebSphere MQ*

注

在使用本资料及其支持的产品之前，请阅读第 133 页的『声明』中的信息。

内容

规划

规划 IBM WebSphere MQ 环境时，必须考虑要配置的 IBM WebSphere MQ 体系结构，资源需求，日志记录需求和备份工具。 使用本主题中的链接来规划 IBM WebSphere MQ 运行所在的环境。

在规划 IBM WebSphere MQ 环境之前，请熟悉基本 IBM WebSphere MQ 概念，请参阅 技术概述中的主题。

相关概念
可用性、恢复和重新启动
相关任务
迁移
安装
配置
管理 WebSphere MQ
确保消息不丢失 (日志记录)

针对 GDPR 就绪性的 IBM MQ 和 IBM MQ Appliance 本地注意事项

适用于下列 PID：

- 5724-H72 IBM MQ
- 5655-AV9 IBM MQ Advanced for z/OS
- 5655-AV1 IBM MQ Advanced for z/OSValue Unit Edition
- 5655-AM9 IBM MQ Advanced Message Security for z/OS
- 5725-Z09 IBM MQ Appliance M2001
- 5725-S14 IBM MQ Appliance M2000
- 5655-MQ9 IBM MQ for z/OS
- 5655-VU9 IBM MQ for z/OS Value Unit Edition
- 5639-L92 IBM MQ Internet Pass-Thru
- 5655-MF9 IBM MQ Managed File Transfer for z/OS
- 5655-ADV IBM WebSphere MQ Advanced for z/OS
- 5655-AMS IBM WebSphere MQ Advanced Message Security for z/OS
- 5724-R10 IBM WebSphere MQ File Transfer Edition for Multiplatforms
- 5724-A39 IBM WebSphere MQ for HP NonStop Server
- 5724-A38 IBM WebSphere MQ for HP OpenVMS
- 5655-W97 IBM WebSphere MQ for z/OS
- 5655-VU8 IBM WebSphere MQ for z/OS Value Unit Edition
- 5655-VUE IBM WebSphere MQ for z/OS Value Unit Edition
- 5725-C79 IBM WebSphere MQ Hypervisor Edition for Red Hat Enterprise Linux® for x86
- 5725-F22 IBM WebSphere MQ Hypervisor for AIX
- 5655-MFT IBM WebSphere MQ Managed File Transfer for z/OS

声明：

此文档旨在帮助您做好 GDPR 准备工作。 它提供有关您可以配置的 IBM MQ 的功能以及产品使用方面的信息，您应该考虑这些信息来帮助组织实现 GDPR 就绪状态。 由于客户选择和配置功能部件的方式有很多种，并且产品单独使用以及与第三方应用程序和系统配合使用的方式也多种多样，因此这些信息并不是一份详尽的列表。

客户负责确保自己遵守各种法律法规，包括欧盟一般数据保护条例。 客户须自行负责从合格的法律顾问那里，就可能会影响客户业务和客户为了遵守此类法律和法规需要采取的任何行动，获得关于任何相关法律和法规的认定和解释的意见。

本文描述的产品、服务和其他功能不适用于所有客户情况，可能具有受限可用性。 IBM 不提供法律、会计或审计建议，或者代表或保证其服务或产品将确保客户遵守任何法律或条例。

目录

欧洲通用数据保护条例

欧盟（“EU”）已采用了《一般数据保护条例》(GDPR) 并于 2018 年 5 月 25 日起实施。

为什么 GDPR 很重要?

GDPR 建立了用于处理个人数据的更强大的数据保护监管框架。 GDPR 实现：

- 新增及增强的个人权利
- 更广泛的个人数据定义
- 新增的处理商义务
- 可能会因不合规而遭受重大经济处罚
- 强制性数据违规通知

请阅读关于 GDPR 的更多信息：

- 欧盟 GDPR 信息门户网站
- ibm.com/GDPR Web 站点

产品配置 - 为 GDPR 做准备时的注意事项

以下部分提供了配置 IBM MQ 以帮助贵组织实现 GDPR 就绪状态的注意事项。

数据生命周期

IBM MQ 是面向事务性消息的中间件产品，使应用程序能够异步交换应用程序提供的数据。 IBM MQ 支持一系列用于连接应用程序的消息传递 API，协议和网桥。 因此，可以使用 IBM MQ 来交换多种形式的数据，其中一些数据可能受 GDPR 约束。 有几个第三方产品可以与 IBM MQ 交换数据。 其中部分产品为 IBM 所有，但许多其他产品均由其他技术供应商提供。 软件产品兼容性报告 Web 站点 提供关联软件的列表。 有关第三方产品的 GDPR 就绪性的注意事项，请参阅该产品的文档。 IBM MQ 管理员通过队列，主题和预订的定义来控制 IBM MQ 与通过其传递的数据进行交互的方式。

哪些类型的数据流流经 IBM MQ?

由于 IBM MQ 为应用程序数据提供异步消息传递服务，因此此问题没有一个确定的答案，因为用例因应用程序部署而异。 应用程序消息数据持久存储在队列文件 (页集或 z/OS 上的耦合设施)，日志和归档以及消息本

身可能包含由 GDPR 管理的数据。 应用程序提供的消息数据也可能包含在为问题确定目的而收集的文件中，例如错误日志，跟踪文件和 FFST。 在 z/OS 应用程序上，提供的消息数据也可能包含在地址空间或耦合设施转储中。

以下是可使用 IBM MQ 交换的个人数据的一些典型示例:

- 客户的员工 (例如， IBM MQ 可用于连接客户的工资单或 HR 系统)
- 客户自己的客户个人数据 (例如， IBM MQ 可能由客户用于在与其客户相关的应用程序之间交换数据，例如，获取销售线索并将数据存储在其 CRM 系统中)。
- 客户自己的客户的敏感个人数据 (例如， IBM MQ 可能在需要交换个人数据的行业上下文中使用，例如，在集成临床应用程序时基于 HL7-based 的医疗保健记录)。

除了应用程序提供的消息数据外， IBM MQ 还会处理以下类型的数据:

- 认证凭证 (例如用户名和密码， API 密钥等)
- 技术上可识别的个人信息（例如，设备标识、基于使用情况的标识、IP 地址等 - 当链接到个人时）

用于与 IBM 在线联系的个人数据

IBM MQ 客户机可以通过多种方式提交在线评论/反馈/请求，以联系 IBM 有关 IBM MQ 主题的信息，主要包括:

- IBM Developer 上的 IBM MQ 区域 中的页面上的公共评论区域
- IBM Documentation 中的 IBM MQ 产品信息 页面上的公共评论区域
- IBM 支持论坛 中的公共评论
- IBM RFE 社区 IBM Developer 中的公共评论

通常，只有客户姓名和电子邮件地址用来启用联系主题的个人回复，而且个人数据的使用符合 IBM 在线隐私声明。

数据 收集

IBM MQ 可用于收集个人数据。 在评估 IBM MQ 的使用情况以及满足 GDPR 需求的需求时，您应该考虑在您的环境中通过 IBM 传递的个人数据类型。 您可能想要考虑如下几个方面:

- 数据如何到达队列管理器? (跨哪些协议? 数据是否已加密？ 数据是否已签名？)
- 如何从队列管理器发送数据? (跨哪些协议? 数据是否已加密？ 数据是否已签名？)
- 数据在通过队列管理器时如何存储? (任何消息传递应用程序都有可能将消息数据写入有状态介质，即使消息是非持久的。 您是否知道消息传递功能可能如何公开通过产品传递的应用程序消息数据的各个方面?)
- IBM MQ 在需要时如何收集和存储凭证以访问第三方应用程序?

IBM MQ 可能需要与需要认证的其他系统和服务 (例如 LDAP) 进行通信。 需要时，认证数据 (用户标识和密码) 由 IBM MQ 进行配置和存储，以便在此类通信中使用。 应尽可能避免将个人凭证用于 IBM MQ 认证。 请考虑保护用于认证数据的存储器。 (请参阅下面的 "数据存储"。)

数据存储

当消息数据通过队列管理器传输时， IBM MQ 将持久存储 (可能是该数据的多个副本) 直接到有状态介质。 IBM MQ 用户可能希望考虑在消息数据处于静止状态时对其进行保护。

以下项目突出显示了 IBM MQ 持久存储应用程序提供的数据的区域，用户在确保符合 GDPR 时可能希望考虑这些区域。

- 应用程序消息队列:

  IBM MQ 提供消息队列以允许应用程序之间进行异步数据交换。 存储在队列上的非持久和持久消息将写入有状态介质。

- 文件传输代理队列:

  IBM MQ Managed File Transfer 利用消息队列来协调文件数据的可靠传输，包含个人数据和传输记录的文件存储在这些队列上。

- 传输队列:

  为了在队列管理器之间可靠地传输消息，将消息临时存储在传输队列上。
- 死信队列:

  在某些情况下，如果队列管理器配置了一条消息，那么无法将消息放入目标队列并将其存储在死信队列中。
- 回退队列:

  JMS 和 XMS 消息传递接口提供了一种功能，允许在发生多次回退后将有害消息移至回退队列，以允许处理其他有效消息。
- AMS 错误队列:

  IBM MQ Advanced Message Security 会将不符合安全策略的消息移动到 SYSTEM.PROTECTION.ERROR.QUEUE 错误队列与死信队列相似。
- 保留出版物:

  IBM MQ 提供保留的发布功能，以允许预订应用程序重新调用先前的发布。

阅读更多:

- 日志记录: 确保消息不会丢失
- MFT 代理队列设置
- 定义传输队列
- 使用死信队列
- 在 IBM MQ classes for JMS 中处理有害消息
- AMS 错误处理
- 保留的发布内容

以下项目突出显示了 IBM MQ 可能间接持久存储应用程序提供的数据的区域，用户在确保遵守 GDPR 时也可能希望考虑这些数据。

- 跟踪路由消息传递:

  IBM MQ 提供跟踪路由功能，用于记录消息在应用程序之间采用的路由。 生成的事件消息可能包括技术上可识别的个人信息，例如 IP 地址。
- 应用程序活动跟踪:

  IBM MQ 提供了应用程序活动跟踪，用于记录应用程序和通道的消息传递 API 活动，应用程序活动跟踪可以将应用程序提供的消息数据的内容记录到事件消息中。
- 服务跟踪:

  IBM MQ 提供服务跟踪功能，用于记录消息数据流通过的内部代码路径。 作为这些功能的一部分， IBM MQ 可以记录应用程序提供的消息数据的内容，以跟踪存储在磁盘上的文件。
- 队列管理器事件:

  IBM MQ 可以生成可包含个人数据 (例如权限，命令和配置事件) 的事件消息。

阅读更多:

- 跟踪路由消息传递
- 使用跟踪
- 事件监视
- 队列管理器事件

要保护对应用程序提供的消息数据副本的访问权，请考虑以下操作:

- 限制特权用户对文件系统中 IBM MQ 数据的访问权，例如在 UNIX 平台上限制 "mqm" 组的用户成员资格。

- 通过专用队列和访问控制限制应用程序对 IBM MQ 数据的访问。 在适当情况下，避免不必要的资源 (例如应用程序之间的队列) 共享，并提供对队列和主题资源的细粒度访问控制。
- 使用 IBM MQ Advanced Message Security 提供消息数据的端到端签名和/或加密。
- 使用文件或卷级别加密来保护用于存储跟踪日志的目录的内容。
- 将服务跟踪上载到 IBM 后，如果您担心可能包含个人数据的内容，那么可以删除服务跟踪文件和 FFST 数据。

阅读更多:

- 特权用户
- 在 Multiplatforms 版上规划文件系统支持

IBM MQ 管理员可以使用凭证 (用户名和密码， API 密钥等) 配置队列管理器。 针对 3rd 参与方服务，例如 LDAP， IBM Cloud Product Insights 和 Salesforce 等。 此数据通常存储在通过文件系统许可权保护的队列管理器数据目录中。

创建 IBM MQ 队列管理器时，将使用基于组的访问控制来设置数据目录，以便 IBM MQ 可以读取配置文件并使用凭证来连接到这些系统。 IBM MQ 管理员被视为特权用户，并且是此组的成员，因此具有对文件的读访问权。 某些文件已加密，但未加密。 因此，要完全保护对凭证的访问权，您应该考虑以下操作:

- 限制特权用户对 IBM MQ 数据的访问权，例如，在 UNIX 平台上限制 "mqm" 组的成员资格。
- 使用文件或卷级别加密来保护队列管理器数据目录的内容。
- 对生产配置目录的备份进行加密，并使用相应的访问控制来存储这些备份。
- 考虑通过安全性，命令和配置事件为认证失败，访问控制和配置更改提供审计跟踪。

阅读更多:

- 保护 IBM MQ

数据访问

可以通过以下产品接口访问 IBM MQ 队列管理器数据，其中一些设计用于通过远程连接进行访问，其他设计用于通过本地连接进行访问。

- IBM MQ 控制台 [仅远程]
- IBM MQ REST API [仅远程]
- MQI [本地和远程]
- JMS [本地和远程]
- XMS [本地和远程]
- IBM MQ Telemetry (MQTT) [仅远程]
- IBM MQ Light (AMQP) [仅远程]
- IBM MQ IMS 网桥 [仅本地]
- IBM MQ CICS 网桥 [仅本地]
- IBM MQ Bridge for HTTP [Only Remote]
- IBM MQ MFT 协议网桥 [仅远程]
- IBM MQ Connect:Direct 网桥 [仅远程]
- IBM MQ Bridge to Salesforce [仅远程]
- IBM MQ Bridge to Blockchain [Only Remote]
- IBM MQ MQAI [本地和远程]
- IBM MQ PCF 命令 [本地和远程]
- IBM MQ MQSC 命令 [本地和远程]
- IBM MQ Explorer [本地和远程]

这些接口旨在允许用户对 IBM MQ 队列管理器以及存储在其上的消息进行更改。 管理和消息传递操作是安全的，因此在发出请求时涉及三个阶段;

- 认证
- 角色映射
- 授权

认证：

如果从本地连接请求了消息或管理操作，那么此连接的源是同一系统上正在运行的进程。 运行该进程的用户必须已通过操作系统提供的任何认证步骤。 将从中建立连接的进程的所有者的用户名声明为身份。 例如，这可能是运行已从中启动应用程序的 shell 的用户的名称。 本地连接的可能认证形式为:

1. 已声明的用户名 (本地操作系统)
2. 可选用户名和密码 (操作系统， LDAP 或定制 3rd 存储库)

如果从远程连接请求了管理操作，那么将通过网络接口与 IBM MQ 进行通信。 以下形式的身份可以通过网络连接进行认证;

1. 已声明的用户名 (来自远程操作系统)
2. 用户名和密码 (操作系统， LDAP 或定制 3rd 存储库)
3. 源网络地址 (例如 IP 地址)
4. X.509 数字证书 (相互 SSL/TLS 认证)
5. 安全性令牌 (例如 LTPA2 令牌)
6. 其他定制安全性 (3rd 参与方出口提供的功能)

角色映射：

在角色映射阶段，可以将认证阶段中提供的凭证映射到备用用户标识。 如果允许映射的用户标识继续 (例如，管理用户可能被通道认证规则阻止) ，那么在授权针对 IBM MQ 资源的活动时，映射的用户标识将转入最终阶段。

授权：

IBM MQ 使不同用户能够对不同的消息传递资源 (例如队列，主题和其他队列管理器对象) 具有不同的权限。

日志记录活动:

IBM MQ 的某些用户可能需要创建访问 MQ 资源的审计记录。 理想审计日志的示例可能包括配置更改，其中包含有关更改及其请求者的信息。

以下信息来源可用于实现此要求:

1. 可以将 IBM MQ 队列管理器配置为在成功运行管理命令时生成命令事件。
2. 可以将 IBM MQ 队列管理器配置为在创建，更改或删除队列管理器资源时生成配置事件。
3. 可以将 IBM MQ 队列管理器配置为在资源的授权检查失败时生成权限事件。
4. 指示授权检查失败的错误消息将写入队列管理器错误日志。
5. 当认证，授权检查失败或创建，启动，停止或删除队列管理器时， IBM MQ Web 控制台会将审计消息写入其日志。

在考虑这些类型的解决方案时， IBM MQ 用户可能希望考虑以下几点:

- 事件消息是非持久的，因此当队列管理器重新启动时，信息将丢失。 应将任何事件监视器配置为持续使用任何可用消息并将内容传输到持久介质。
- IBM MQ 特权用户具有足够的特权来禁用事件，清除日志或删除队列管理器。

有关保护对 IBM MQ 数据的访问并提供审计跟踪的更多信息，请参阅以下主题:

- IBM MQ 安全性机制
- 配置事件
- 命令事件

- 错误日志

数据处理

使用公共密钥基础结构进行加密：

您可以保护与 IBM MQ 的网络连接以使用 TLS，这还可以提供连接起始端的相互认证。

使用传输机制提供的 PKI 安全设施是使用 IBM MQ 保护数据处理的第一步。 但是，在不启用进一步的安全功能的情况下，消费应用程序的行为是处理传递给它的所有消息，而不验证消息的来源或在传输过程中是否发生了更改。

获得许可使用 Advanced Message Security (AMS) 功能的 IBM MQ 用户可以通过安全策略的定义和配置来控制应用程序处理消息中保存的个人数据的方式。 安全策略允许将数字签名和/或加密应用于应用程序之间的消息数据。

在使用消息时，可以使用安全策略来要求和验证数字签名，以确保消息是真实的。 AMS 加密提供了一种方法，通过该方法将消息数据从可读格式转换为编码版本，只有当它是预期的接收方或消息并且有权访问正确的解密密钥时，才能由另一个应用程序进行解码。

有关使用 SSL 和证书来保护网络连接的更多信息，请参阅 IBM MQ V9 产品文档中的以下主题:

- 为 IBM MQ
- AMS 概述

数据删除

IBM MQ 提供了用于删除已提供给产品的数据的命令和用户界面操作。 这使 IBM MQ 的用户能够在需要时删除与特定个人相关的数据。

- 要考虑遵守 GDPR 客户机数据删除的 IBM MQ 行为区域
  - 通过以下方法删除存储在应用程序队列上的消息数据:
    - 使用消息传递 API 或工具或使用消息到期来除去个别消息。
    - 指定消息是非持久消息，保留在非持久消息类正常的队列上，并重新启动队列管理器。
    - 以管理方式清除队列。
    - 正在删除队列。
  - 通过以下方法删除存储在主题上的保留发布数据:
    - 指定消息是非持久消息，并重新启动队列管理器。
    - 将保留的数据替换为新数据或使用消息到期。
    - 以管理方式清除主题字符串。
  - 通过删除整个队列管理器来删除存储在队列管理器上的数据。
  - 通过删除跟踪目录中的文件来删除服务跟踪命令所存储的数据。
  - 删除通过删除 errors 目录中的文件存储的 FFST 数据。
  - 删除地址空间和耦合设施转储 (在 z/OS 上)。
  - 删除此类数据的归档，备份或其他副本。
- 要考虑遵守 GDPR 帐户数据删除的 IBM MQ 行为区域
  - 您可以删除 IBM MQ 存储的用于连接到队列管理器和 3rd 服务的帐户数据和首选项 (包括其归档，备份或复制副本):
    - 用于存储凭证的队列管理器认证信息对象。
    - 队列管理器权限记录引用用户标识。
    - 用于映射或阻止特定 IP 地址，证书 DN 或用户标识的队列管理器通道认证规则。
    - 由 IBM MQ Managed File Transfer 代理，记录器和 MQ Explorer MFT 插件用于向队列管理器和文件服务器进行认证的凭证文件。

- X.509 数字证书，用于表示或包含有关密钥库中个人的信息， SSL/TLS 连接或 IBM MQ Advanced Message Security (AMS) 可使用这些密钥库。
- 来自 IBM MQ Appliance 的个人用户帐户，包括系统日志文件中对这些帐户的引用。
- IBM MQ Explorer 工作空间元数据和 Eclipse 设置。
- IBM MQ Explorer 密码存储，如 密码首选项中所指定。
- IBM MQ 控制台和 mqweb 服务器配置文件。
- Salesforce 连接数据配置文件。
- 区块链连接数据配置文件。
- IBM Cloud Product Insights 连接数据，位于 qm.ini 和 APIKeyFile 中的 ReportingService 节下。

阅读更多：

- 配置 IBM MQ Bridge to Salesforce
- 配置 IBM MQ 以用于区块链
- MFT 和 IBM MQ 连接认证
- 使用 ProtocolBridgeCredentials.xml 文件映射文件服务器的凭证
- 配置 IBM MQ Console 用户和角色

数据监视

IBM MQ 提供了一系列监视功能，用户可以利用这些功能来更好地了解应用程序和队列管理器的执行方式。

IBM MQ 还提供了一些有助于管理队列管理器错误日志的功能。

阅读更多：

- 监视 IBM MQ 网络
- 诊断消息服务
- QMErrorLog 服务

IBM MQ 提供了一项功能，使用户能够将信息发布到 IBM Cloud Product Insights 服务，以便 IBM MQ 用户可以查看队列管理器启动和使用情况信息。

阅读更多：

- 配置 IBM MQ 以与 IBM Cloud 中的 IBM Cloud Product Insights 服务配合使用

限制使用个人数据的功能

通过使用本文档中概述的工具， IBM MQ 使最终用户能够限制其个人数据的使用。

IBM MQ 消息队列不应以与数据库相同的方式用作永久数据存储器，在处理受 GDPR 约束的应用程序数据时尤其如此。

与可以通过搜索查询找到数据的数据库不同，除非您知道消息的队列，消息和相关标识，否则很难找到消息数据。

提供的包含个人数据的消息可以随时识别和定位，可以使用标准 IBM MQ 消息传递功能来访问或修改消息数据。

文件处理

1. IBM MQ Managed File Transfer 不会对传输的文件执行恶意软件扫描。 按原样传输文件，并执行完整性检查以确保在传输期间不修改文件数据。 源和目标校验和作为传输状态发布的一部分发布。 建议最终用户在 MFT 传输文件之前以及 MFT 将文件交付到远程端点之后，针对其环境实施适当的恶意软件扫描。
2. IBM MQ Managed File Transfer 不会根据 MIME 类型或文件扩展名执行操作。 MFT 读取文件并传输的字节与从输入文件读取的字节完全相同。

设计 IBM WebSphere MQ 体系结构

了解 IBM WebSphere MQ 针对点到点和发布/预订消息传递样式支持的不同体系结构。

在规划 IBM WebSphere MQ 体系结构之前，请熟悉基本 IBM WebSphere MQ 概念，请参阅 IBM WebSphere MQ 技术概述中的主题。

IBM WebSphere MQ 体系结构从使用单个队列管理器的简单体系结构到相互连接的队列管理器的更复杂网络。 使用分布式排队技术将多个队列管理器连接在一起。 有关规划单个队列管理器和多个队列管理器体系结构的更多信息，请参阅以下主题:

- 第 13 页的『基于单个队列管理器的体系结构』
- 第 14 页的『基于多个队列管理器的体系结构』
- 第 14 页的『网络和网络规划』
- WebSphere MQ 分布式消息传递方法

如果需要多个逻辑相关的队列管理器，并且需要共享数据和应用程序，那么可以将这些队列管理器分组到集群中。 使用集群可以使队列管理器相互通信，而无需设置额外的通道定义或远程队列定义，从而简化其配置和管理。 有关使用集群的更多信息，请参阅 集群的工作方式。

相关概念

第 5 页的『规划』

规划 IBM WebSphere MQ 环境时，必须考虑要配置的 IBM WebSphere MQ 体系结构，资源需求，日志记录需求和备份工具。 使用本主题中的链接来规划 IBM WebSphere MQ 运行所在的环境。

相关任务

配置

基于单个队列管理器的体系结构

最简单的 IBM WebSphere MQ 体系结构涉及单个队列管理器的配置和使用。

在规划 IBM WebSphere MQ 体系结构之前，请熟悉基本 IBM WebSphere MQ 概念，请参阅 IBM WebSphere MQ 简介。

以下部分描述了使用单个队列管理器的许多可能的体系结构:

- 第 13 页的『具有访问服务的本地应用程序的单个队列管理器』
- 第 13 页的『具有作为客户机访问服务的远程应用程序的单个队列管理器』
- 第 13 页的『具有发布/预订配置的单个队列管理器』

具有访问服务的本地应用程序的单个队列管理器

基于单个队列管理器的第一个体系结构是访问服务的应用程序与提供服务的应用程序在同一系统上运行。 IBM WebSphere MQ 队列管理器提供请求服务的应用程序与提供服务的应用程序之间的异步相互通信。 这意味着即使其中一个应用程序处于脱机状态很长一段时间，应用程序之间的通信也可以继续。

具有作为客户机访问服务的远程应用程序的单个队列管理器

基于单个队列管理器的第二个体系结构具有从提供服务的应用程序远程运行的应用程序。 远程应用程序在服务的不同系统上运行。 应用程序作为客户机连接到单个队列管理器。 这意味着可以通过单个队列管理器向多个系统提供对服务的访问权。

此体系结构的一个限制是网络连接必须可供应用程序运行。 应用程序与队列管理器之间通过网络连接进行的交互是同步的。

具有发布/预订配置的单个队列管理器

使用单个队列管理器的备用体系结构是使用发布/预订配置。 在发布/预订消息传递中，可以将信息提供者与该信息的使用者分离。 这与先前描述的体系结构中的点到点消息传递样式不同，其中应用程序必须知道有关目标应用程序的信息，例如要将消息放入的队列名称。 通过使用 IBM WebSphere MQ 发布/预订，发送应用

程序将根据信息主题发布具有指定主题的消息。 IBM WebSphere MQ 处理将消息分发到已通过预订在该主题中注册兴趣的应用程序。 接收应用程序也不需要知道要接收这些应用程序的消息源的任何信息。 有关发布/预订消息传递的更多信息，请参阅 WebSphere MQ 发布/预订消息传递简介。 有关使用单个队列管理器的发布/预订消息传递的示例，请参阅 单个队列管理器发布/预订配置的示例。

相关概念

第 13 页的『设计 IBM WebSphere MQ 体系结构』
了解 IBM WebSphere MQ 针对点到点和发布/预订消息传递样式支持的不同体系结构。

相关信息

WebSphere MQ 简介
创建和管理队列管理器

基于多个队列管理器的体系结构

您可以使用分布式消息排队技术来创建涉及配置和使用多个队列管理器的 IBM WebSphere MQ 体系结构。

在规划 IBM WebSphere MQ 体系结构之前，请熟悉基本 IBM WebSphere MQ 概念，请参阅 IBM WebSphere MQ 简介。

可以通过添加其他队列管理器来更改 IBM WebSphere MQ 体系结构，而无需更改提供服务的应用程序。

应用程序可以与队列管理器托管在同一台机器上，然后与另一个系统上的另一个队列管理器上托管的服务进行异步通信。 或者，访问服务的应用程序可以作为客户机连接到队列管理器，然后在另一个队列管理器上提供对服务的异步访问。

用于连接不同队列管理器及其队列的路由是使用分布式排队技术定义的。 体系结构中的队列管理器使用通道进行连接。 通道用于根据队列管理器的配置在一个方向上自动将消息从一个队列管理器移动到另一个队列管理器。

有关规划 IBM WebSphere MQ 网络的高级概述，请参阅 第 14 页的『网络和网络规划』。

有关如何为 IBM WebSphere MQ 体系结构规划通道的信息，请参阅 WebSphere MQ 分布式消息传递技术。

分布式队列管理使您能够创建和监视队列管理器之间的通信。 有关分布式队列管理的更多信息，请参阅 分布式队列管理简介。

相关概念

WebSphere MQ 简介
第 13 页的『设计 IBM WebSphere MQ 体系结构』
了解 IBM WebSphere MQ 针对点到点和发布/预订消息传递样式支持的不同体系结构。

相关任务

创建和管理队列管理器

网络和网络规划

WebSphere MQ 在应用程序之间以及通过使用队列管理器和通道的网络发送和接收数据。 网络规划涉及定义需求，以创建用于通过网络连接这些系统的框架。

可以在您的系统与需要进行通信的任何其他系统之间创建通道。 可以创建多中继段通道以连接到没有直接连接的系统。 方案中描述的消息通道连接在 第 15 页的图 1 中显示为网络图。

通道和传输队列名称

可以为传输队列提供任何名称。 但是为了避免混淆，您可以为它们提供与目标队列管理器名称或队列管理器别名相同的名称 (视情况而定)。 这使传输队列与它们使用的路由相关联，提供了通过中间 (多跳跃) 队列管理器创建的并行路由的清晰概述。

对于通道名称而言，它并不是那么清晰。 例如， 第 15 页的图 1 中针对 QM2 的通道名称对于入局和出局通道必须不同。 所有通道名称仍可包含其传输队列名称，但必须对其进行限定以使其唯一。

例如，在 QM2 上，有一个 QM3 通道来自 QM1，还有一个 QM3 通道转至 QM3。 要使名称唯一，第一个名称可以命名为 "QM3_from_QM1"，第二个名称可以命名为 "QM3_from_QM2"。 这样，通道名称在名称的第一部分中显示传输队列名称。 方向和相邻队列管理器名称显示在名称的第二部分中。

第 15 页的表 1 中提供了 第 15 页的图 1 的建议通道名称表。

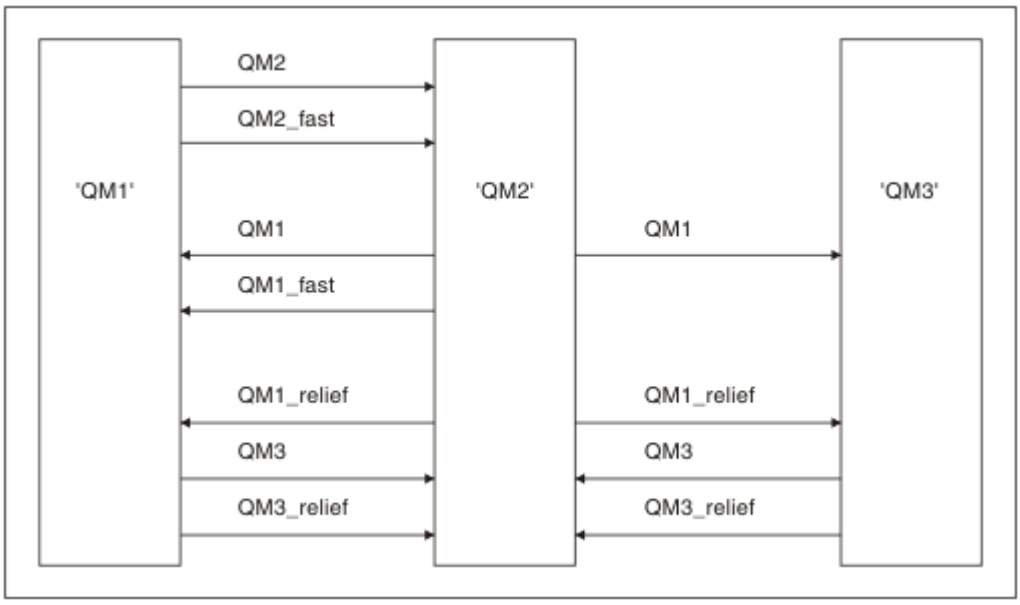


图 *1:* 显示所有通道的网络图

表 *1:* 通道名称示例

路由名称	**队列管理器托管通道**	**传输队列的名称**	**建议的通道名称**
QM1	QM1 & QM2	QM1 (位于 QM2)	QM1.from.QM2
QM1	QM2 & QM3	QM1 (位于 QM3)	QM1.from.QM3
QM1_fast	QM1 & QM2	QM1_fast (位于 QM2)	QM1_fast.from.QM2
QM1_relief	QM1 & QM2	QM1_relief (位于 QM2)	QM1_relief.from.QM2
QM1_relief	QM2 & QM3	QM1_relief (位于 QM3)	QM1_relief.from.QM3
QM2	QM1 & QM2	QM2 (位于 QM1)	QM2.from.QM1
QM2_fast	QM1 & QM2	QM2_fast (位于 QM1)	QM2_fast.from.QM1
QM3	QM1 & QM2	QM3 (位于 QM1)	QM3.from.QM1
QM3	QM2 & QM3	QM3 (位于 QM2)	QM3.from.QM2
QM3_relief	QM1 & QM2	QM3_relief (位于 QM1)	QM3_relief.from.QM1
QM3_relief	QM2 & QM3	QM3_relief (位于 QM2)	QM3_relief.from.QM2

注:

1. 在 WebSphere MQ for z/OS 上，队列管理器名称限制为四个字符。
2. 对网络中的所有通道进行唯一命名。 如 第 15 页的表 1 中所示，在通道名称中包含源队列管理器名称和目标队列管理器名称是一种很好的方法。

网络策划员

创建网络假定有另一个更高级别的 网络规划员 功能，其计划由团队的其他成员实施。

对于广泛使用的应用程序，使用本地访问站点之间的宽带链路进行消息流量集中的本地访问站点方面的思考更为经济，如 第 16 页的图 2 所示。

在这个例子中，有两个主要系统和一些卫星系统。 实际配置将取决于业务注意事项。 有两个集中器队列管理器位于方便的中心。 每个 QM 集中器都具有到本地队列管理器的消息通道:

- QM 集中器 1 具有到三个本地队列管理器 (QM1， QM2 和 QM3) 中每一个的消息通道。 使用这些队列管理器的应用程序可以通过 QM 集中器相互通信。
- QM 集中器 2 具有到三个本地队列管理器 (QM4， QM5 和 QM6) 的消息通道。 使用这些队列管理器的应用程序可以通过 QM 集中器相互通信。
- QM 集中器之间有消息通道，因此允许队列管理器中的任何应用程序与另一个队列管理器中的任何其他应用程序交换消息。

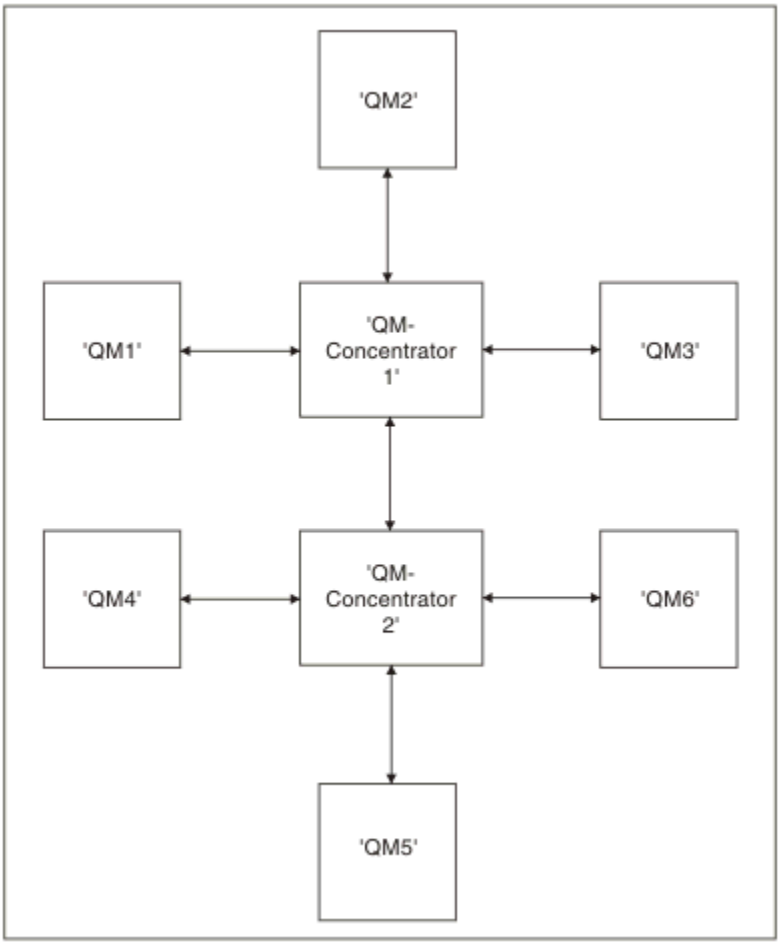


图 *2:* 显示 *QM* 集中器的网络图

集群

本主题提供有关规划和管理 IBM WebSphere MQ 集群的指导。 此信息是基于测试和客户反馈的指南。

以下信息假定用户基本了解 IBM WebSphere MQ 集群。 此信息并非旨在作为 “一刀切” 解决方案，而是尝试共享常见问题的常见方法。

集群提供了一种用于以简化设置系统所需的初始配置和所需的持续管理的方式互连队列管理器的机制。 配置越大，受益越大。

在规划集群系统时需要谨慎，以确保它们正常运行，并确保系统所需的可用性和响应能力水平，特别是对于更大或更复杂的集群系统。

成功的集群设置依赖于良好的规划和对 IBM WebSphere MQ 基础知识 (例如，良好的应用程序管理和网络设计) 的全面了解。 确保您熟悉 相互通信的概念 和 集群的工作方式 中的信息。

什么是集群以及为何使用这些集群?

集群提供了两个关键优势:

- 集群简化了 IBM WebSphere MQ 网络的管理，这些网络通常需要为要配置的通道，传输队列和远程队列提供许多对象定义。 在许多队列管理器需要互连的大型网络中，尤其存在这种情况。 这种架构特别难以配置和积极维护。
- 集群可用于在集群中的队列和队列管理器之间分配消息流量的工作负载。 此类分发允许将单个队列的消息工作负载分布到位于多个队列管理器上的该队列的等效实例中。 工作负载的这种分布可用于实现对系统故障的更大弹性，以及提高系统中特别活跃的消息流的缩放性能。 在这种环境中，分布式队列的每个实例都具有处理消息的使用应用程序。

相关信息

集群: 最佳实践

点到点消息传递

IBM WebSphere MQ 中最简单的消息传递形式是点到点消息传递。

在点到点消息传递中，发送应用程序必须先知道有关接收应用程序的信息，然后才能向该应用程序发送消息。 例如，发送应用程序可能需要知道要将信息发送到的队列的名称，还可能指定队列管理器名称。

可用于 IBM WebSphere MQ 的备用消息传递样式是发布/预订消息传递。 发布/预订消息传递允许您使信息的提供者与该信息的使用者分离开来。 发送应用程序和接收应用程序不需要相互了解即可发送和接收信息。 有关发布/预订消息传递的更多信息，请参阅 WebSphere MQ 发布/预订消息传递简介。

相关信息

开发应用程序

WebSphere MQ 消息

IBM WebSphere MQ 发布/预订消息传递简介

发布/预订消息传递允许您使信息的提供者与该信息的使用者分离开来。 发送应用程序和接收应用程序不需要相互了解即可发送和接收信息。

在点到点 IBM WebSphere MQ 应用程序可以向另一个应用程序发送消息之前，它需要了解有关该应用程序的一些信息。 例如，它需要知道要向其发送信息的队列的名称，并且还可以指定队列管理器名称。

IBM WebSphere MQ 发布/预订使您的应用程序无需了解有关目标应用程序的任何信息。 所有发送应用程序都必须执行以下操作: 放入一条 IBM WebSphere MQ 消息，其中包含它想要的信息，并为它分配一个主题，该主题表示信息的主题，并让 IBM WebSphere MQ 处理该信息的分发。 类似地，目标应用程序不必知道有关其接收的信息源的任何信息。

第 17 页的图 3 显示了最简单的发布/预订系统。 有一个发布程序，一个队列管理器和一个订户。 预订从订户发送到队列管理器，发布从发布者发送到队列管理器，然后由队列管理器将发布转发到订户。

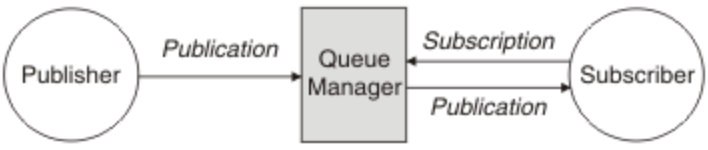


图 *3:* 简单发布/预订配置

典型的发布/预订系统具有多个发布者和多个订户，并且通常具有多个队列管理器。 应用程序可以既是发布者也是订户。

发布/预订组件概述

发布/订阅是订阅者以消息形式从发布者接收信息的机制。 发布者和订户之间的交互由队列管理器使用标准 WebSphere MQ 工具进行控制。

典型的发布/预订系统具有多个发布者和多个订户，并且通常具有多个队列管理器。 应用程序可以既是发布者也是订户。

信息的提供者称为发布者。 发布者提供关于主题的信息，而不必了解任何关于对该信息有兴趣的应用程序的情况。 发布者以消息形式生成此信息，称为 发布 ，他们要发布这些消息并定义这些消息的主题。

信息的使用者称为订户。 订户创建描述订户感兴趣的主题的 预订 。 因此，订阅确定了转发给订阅者的发布内容。 订户可以创建多个预订，并可以接收来自许多不同发布者的信息。

发布的信息在 WebSphere MQ 消息中发送，信息的主题由其 主题标识。 发布者在发布信息时指定主题，订户指定要接收发布的主题。 仅向订户发送有关其预订的主题的信息。

它是存在的主题，允许信息的提供者和使用者在发布/预订消息传递中解耦，方法是根据点到点消息传递中的要求，消除在每条消息中包含特定目标的需求。

发布者和订户之间的交互都由队列管理器控制。 队列管理器从发布者接收消息，并从订户接收预订 (针对一系列主题)。 队列管理器的作业是将已发布的消息路由到已在消息主题中注册兴趣的订户。

标准 WebSphere MQ 工具用于分发消息，因此应用程序可以使用可供现有 WebSphere MQ 应用程序使用的所有功能。 这意味着您可以使用持久消息来获取仅一次有保证的传递，并且您的消息可以是事务性工作单元的一部分，以确保仅当消息由发布者落实时才将其传递给订户。

单个队列管理器发布/预订配置的示例

第 18 页的图 4 说明了基本的单个队列管理器发布/预订配置。 此示例显示了新闻服务的配置，其中发布者提供了有关多个主题的信息:

- 发布程序 1 正在使用 "运动" 主题发布有关运动结果的信息
- 发布程序 2 正在使用 "股票" 主题发布有关股票价格的信息
- 发布者 3 正在使用 "电影" 主题发布有关电影评论的信息，并使用 "电视" 主题发布有关电视列表的信息

三个订户已注册了对不同主题的兴趣，因此队列管理器会向他们发送他们感兴趣的信息:

- 订户 1 接收体育结果和股价
- 订户 2 接收影片评论
- 订户 3 接收体育结果

没有一个用户登记了对电视列表的兴趣，所以这些都没有分发。

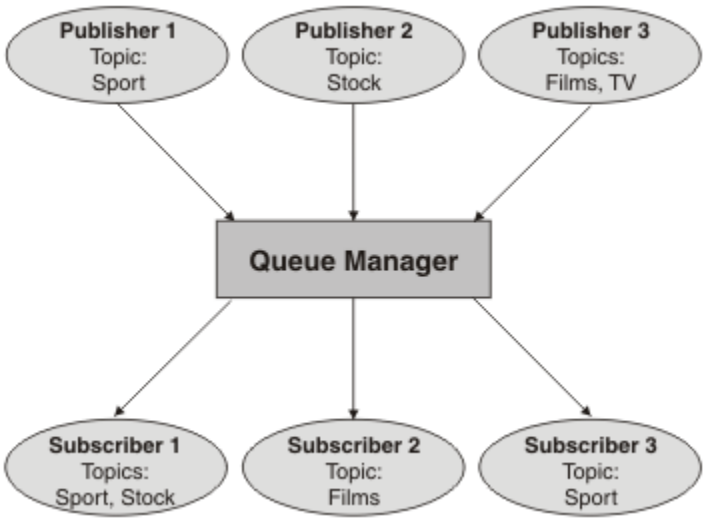


图 *4:* 单个队列管理器发布/预订示例

出版商和出版物

在 WebSphere MQ 发布/预订中，发布者是一个应用程序，它以称为发布的标准 WebSphere MQ 消息的形式向队列管理器提供有关指定主题的信息。 发布者可以发布有关多个主题的信息。

发布者使用 MQPUT 动词将消息放入先前打开的主题，此消息是发布。 然后，本地队列管理器会将发布路由到预订该发布主题的任何订户。 已发布的消息可以由多个订户使用。

除了将发布分发给具有相应预订的所有本地订户外，队列管理器还可以直接或通过具有主题订户的队列管理器网络将发布分发给与其连接的任何其他队列管理器。

在 WebSphere MQ 发布/预订网络中，发布应用程序也可以是订户。

同步点下的出版物

发布者可以在同步点中发出 MQPUT 或 MQPUT1 调用，以在工作单元中包含传递给订户的所有消息。 如果指定了值为 ALL 或 ALLDUR 的 MQPMO_RETAIN 选项或主题交付选项 NPMSGDLV 或 PMSGDLV ，那么队列管理器将在发布程序 MQPUT 或 MQPUT1 调用的作用域内同步点中使用内部 MQPUT 或 MQPUT1 调用。

状态和事件信息

可以将出版物分类为状态出版物 (例如，库存的当前价格) 或事件出版物 (例如，该库存的贸易)。

状态发布

状态发布 包含有关某事物的当前状态的信息，例如，库存价格或足球比赛中的当前分数。 当发生某件事（如股票价格更改或足球比分更改）时，将不再需要先前的状态信息，因为它已被新的信息取代。

订户将希望在启动时接收当前版本的状态信息，并在状态更改时被发送新信息。

如果发布内容包含了状态信息，其通常作为保留发布内容被发布。 新订户通常希望立即获取当前状态信息; 订户不希望等待导致重新发布信息的事件。 除非预订程序使用 MQSO_PUBLIC ICATIONS_ON_REQUEST 或 MQSO_NEW_PUBLIC ICATIONS_ONLY 选项，否则预订程序在预订时将自动接收主题的保留发布。

事件发布

事件发布 包含有关发生的个别事件的信息，例如某个库存的交易或特定目标的评分。 每个事件独立于其他事件。

订户将希望在事件发生时接收有关这些事件的信息。

保留的发布内容

缺省情况下，在将发布发送到所有感兴趣的订户之后，将废弃该发布。 但是，发布者可以指定保留发布的副本，以便可以将其发送给在主题中注册兴趣的未来订户。

在将发布发送给所有感兴趣的订户后删除这些发布适用于事件信息，但并不总是适用于状态信息。 通过保留消息，新订阅者不必等待再次发布信息即可接收到初始状态信息。 例如，预订股票价格的订户将直接收到当前价格，而无需等待股票价格更改 (并因此重新发布)。

队列管理器只能为每个主题保留一个发布，因此当新的保留发布到达队列管理器时，将删除主题的现有保留发布。 但是，删除现有发布可能不会在新保留发布到达时同步发生。 因此，只要有可能，就有不超过一个发布者发送有关任何主题的保留发布。

订户可以使用 MQSO_NEW_publicATIONS_ONLY 预订选项指定他们不希望接收保留发布。 现有订户可以要求向其发送保留发布的重复副本。

有时，您可能不希望保留发布，即使是针对状态信息:

- 如果在对某个主题进行任何发布之前对该主题进行了所有预订，并且您不期望或不允许新预订，那么不需要保留发布，因为这些发布在第一次发布时交付到完整的订户集。
- 如果发布频繁发生 (例如每秒) ，那么新订户 (或从故障中恢复的订户) 几乎在其初始预订之后立即接收到当前状态，因此不需要保留这些发布。

- 如果发布内容较大，那么最终可能需要大量存储空间来存储每个主题的保留发布内容。 在多队列管理器环境中，保留发布由具有匹配预订的网络中的所有队列管理器存储。

在决定是否使用保留发布时，请考虑预订应用程序如何从故障中恢复。 如果发布者不使用保留发布，那么订户应用程序可能需要在本地存储其当前状态。

要确保保留发布，请使用 MQPMO_RETAIN put-message 选项。 如果使用此选项并且无法保留发布，那么将不会发布消息，并且调用将失败并返回 MQRC_PUT_NOT_保留。

如果消息是保留发布，那么此消息由 MQIsRetained 消息属性指示。 消息的持久性与最初发布消息时一样。

同步点下的出版物

在 IBM WebSphere MQ 发布/预订中，同步点可以由发布者使用，也可以由队列管理器在内部使用。

发布程序在使用 MQPMO_SYNCPOINT 选项发出 MQPUT/MQPUT1 调用时使用同步点。 传递到订户的所有消息将计入 work.The MAXUMSGS 队列管理器属性指定此限制。 如果达到限制，那么发布程序将接收到 2024 (07E8) (RC2024) :MQRC_SYNCPOINT_LIMIT_已达到 原因码。

当发布者使用带有 MQPMO_RETAIN 选项的 MQPMO_NO_SYNCPOINT 或带有值 ALL 或 ALLDUR 的主题传递选项 NPMSGDV/PMSGDV 发出 MQPUT/MQPUT1 调用时，队列管理器将使用内部同步点来保证按请求传递消息。 如果在发布程序 MQPUT/MQPUT1 调用范围内达到限制，那么发布程序可以接收 2024 (07E8) (RC2024): MQRC_SYNCPOINT_LIMIT_已达到 原因码。

订户和预订

在 WebSphere MQ 发布/预订中，订户是从发布/预订网络中的队列管理器请求有关特定主题的信息的应用程序。 订户可以从多个发布程序接收关于相同或不同主题的消息。

可以使用 MQSC 命令手动创建预订，也可以由应用程序手动创建预订。 这些预订将向本地队列管理器发出，并包含有关订户要接收的发布的信息:

- 订户感兴趣的主题; 如果使用通配符，那么这可以解析为多个主题。
- 要应用于已发布消息的可选的选择字符串。
- 队列 (称为 订户队列) 的句柄，应该将所选发布放在该队列上，以及可选的 CorrelId。

本地队列管理器存储预订信息，当它接收到发布时，扫描该信息以确定是否存在与发布的主题和选择字符串相匹配的预订。 对于每个匹配的预订，队列管理器会将发布定向到订户的订户队列。 可以使用 DIS SUB 和 DIS SBSTATUS 命令来查看队列管理器存储的有关预订的信息。

仅当发生下列其中一个事件时，才会删除预订:

- 订户使用 MQCLOSE 调用取消预订 (如果以非持久方式进行预订)。
- 预订将到期。
- 系统管理员使用 DELETE SUB 命令删除预订。
- 订户应用程序结束 (如果预订以非持久方式进行)。
- 队列管理器将停止或重新启动 (如果预订以非持久方式进行)。

获取消息时，请在 MQGET 调用上使用相应的选项。 如果应用程序仅处理一个预订的消息，那么至少应使用 `get-by-correlid`，如 C 样本程序 `amqssbxa.c` 和 非受管 MQ 订户中所示。 要使用的 **CorrelId** 从 MQSD 中的 MQSUB 返回。**SubCorrelId** 字段。

受管队列和发布/预订

创建预订时，可以选择使用受管排队。 如果使用受管队列，那么将在创建预订时自动创建预订队列。 将根据预订的耐久性自动对受管队列进行分层。 使用受管队列意味着您不必担心创建队列以接收发布，如果非持久预订连接已关闭，那么将自动从订户队列中除去任何未使用的发布。

如果应用程序不需要将特定队列用作其订户队列 (它接收的发布的目标) ，那么它可以使用 MQSO_MANAGED 预订选项来使用 受管预订 。 如果您创建受管预订，那么队列管理器会将对象句柄返回给订户队列的订户，队列管理器将在该订户队列中创建将接收发布的对象句柄。 将返回队列的对象句柄，允许您浏览，获取或查询队列 (除非已显式授予您对临时动态队列的访问权，否则无法放入或设置受管队列的属性)。

预订的耐久性确定在预订应用程序与队列管理器的连接中断时受管队列是否保留。

与非持久预订配合使用时，受管预订特别有用，因为当应用程序的连接结束时，未使用的消息将无限期地保留在订户队列上占用队列管理器中的空间。 如果您正在使用受管预订，那么该受管队列将是临时动态队列，因此，当由于以下任何原因导致连接中断时，将删除该受管队列以及任何未使用的消息:

- 使用带有 MQCO_REMOVE_SUB 的 MQCLOSE ，并关闭受管 Hobj。
- 与使用非持久预订 (MQSO_NON_耐久性) 的应用程序的连接丢失。
- 由于预订已到期并且受管 Hobj 已关闭，因此将除去该预订。

受管预订也可以与持久预订配合使用，但您可能希望在订户队列中保留未使用的消息，以便在重新打开连接时可以检索这些消息。 因此，持久预订的受管队列采用永久动态队列形式，并将在预订应用程序与队列管理器的连接中断时保留。

如果要使用永久动态受管队列，那么可以在预订上设置到期时间，以便尽管该队列在连接中断后仍将存在，但它不会无限期地继续存在。

如果删除受管队列，那么将接收到错误消息。

创建的受管队列在结束时使用数字 (时间戳记) 进行命名，因此每个队列都是唯一的。

预订耐久性

可以将预订配置为持久或非持久。 预订耐久性确定在预订应用程序与队列管理器断开连接时对预订执行的操作。

持久预订

持久预订在预订应用程序与队列管理器之间的连接关闭后继续存在。 如果预订是持久的，那么当预订应用程序断开连接时，预订将保留在原位置，并且可以由预订应用程序在使用创建预订时返回的 SubName 重新连接请求预订时使用。

持久预订时，需要预订名称 (SubName)。 预订名称在队列管理器中必须唯一，以便可用于标识预订。 如果您有意关闭了预订的句柄 (使用 MQCO_KEEP_SUB 选项) 或已与队列管理器断开连接，那么在指定要恢复的预订时，此标识方法是必需的。 您可以使用带有 MQSO_RESUME 选项的 MQSUB 调用来恢复现有预订。 如果将 DISPLAY SBSTATUS 命令与 SUBTYPE ALL 或 ADMIN 配合使用，那么还会显示预订名称。

当应用程序不再需要持久预订时，可以使用带有 MQCO_REMOVE_SUB 选项的 MQCLOSE 函数调用将其除去，也可以使用 MQSC 命令 DELETE SUB 手动将其删除。

是否可以使用 **DURSUB** 主题属性来控制对主题进行持久预订。

使用 MQSO_RESUME 选项从 MQSUB 调用返回时，预订到期将设置为预订的原始到期时间，而不是剩余到期时间。

队列管理器继续发送发布以满足持久预订，即使该订户应用程序未连接也是如此。 这将导致在订户队列上构建消息。 避免此问题的最简单方法是在适当情况下使用非持久预订。 但是，在需要使用持久预订的情况下，如果订户使用 保留发布 选项进行预订，那么可以避免构建消息。 然后，订户可以通过使用 MQSUBRQ 调用来控制何时接收发布。

非持久预订

仅当预订应用程序与队列管理器的连接保持打开状态时，非持久预订才存在。 当预订应用程序有意地断开或由于连接中断而断开与队列管理器的连接时，将除去此预订。 关闭连接时，将从队列管理器中除去有关预订的信息，如果使用 DISPLAY SBSTATUS 命令显示预订，那么将不再显示这些信息。 不会将更多消息放入订户队列。

对于非持久预订的订户队列上的任何未使用的发布，将按如下所示进行确定。

- 如果预订应用程序正在使用 受管目标，那么将自动除去尚未使用的任何发布。
- 如果预订应用程序在预订时向其自己的订户队列提供句柄，那么不会自动除去未使用的消息。 如果需要，应用程序负责清除队列。 如果队列由多个订户或其他点到点应用程序共享，那么可能不适合完全清除该队列。

虽然非持久预订不需要预订名称，但队列管理器会使用预订名称 (如果提供)。 预订名称在队列管理器中必须唯一，以便可用于标识预订。

选择字符串

选择字符串 是应用于发布以确定其是否与预订匹配的表达式。 选择字符串可以包含通配符。

预订时，除了指定主题外，还可以指定选择字符串以根据其消息属性选择发布。

主题

主题是发布/订阅消息中发布的信息的论题。

将点到点系统中的消息发送到特定目标地址。 基于主题的发布/预订系统中的消息将根据描述消息内容的主题发送给订户。 在基于内容的系统中，根据消息本身的内容向订户发送消息。

IBM WebSphere MQ 发布/预订系统是基于主题的发布/预订系统。 发布者创建一条消息，并使用最适合发布主题的主题字符串来发布该消息。 要接收发布内容，订户创建带有模式匹配主题字符串的预订，该字符串用于选择发布内容主题。 队列管理器将发布内容传递给其预订与发布内容主题匹配并有权接收发布内容的订户。 文章 第 22 页的『主题字符串』描述了用于标识出版物主题的主题字符串的语法。 订户还创建主题字符串以选择要接收的主题。 订户创建的主题字符串可以包含两个备用通配符方案中的任何一个，以便与发布内容中的主题字符串进行模式匹配。 第 23 页的『通配方案』中描述了模式匹配。

在基于主题的发布/预订中，发布者或管理员负责将主题分类为主题。 通常，主题以分层方式组织到主题树中，使用 '/' 字符在主题字符串中创建子主题。 请参阅 第 28 页的『主题树』 以获取主题树的示例。 主题是主题树中的节点。 主题可以是没有进一步子主题的叶节点，也可以是具有子主题的中间节点。

在将主题组织到分层主题树中的同时，可以将主题与管理主题对象相关联。 通过将主题与管理主题对象相关联，可以将属性分配给该主题，例如，该主题是否分布在集群中。 通过使用管理主题对象的 TOPICSTR 属性对主题进行命名来进行关联。 如果未显式将管理主题对象与主题关联，那么该主题将继承其在 具有 与管理主题对象关联的主题树中最接近的祖代的属性。 如果您根本未定义任何父主题，那么它将从 SYSTEM.BASE.TOPIC。 第 30 页的『管理主题对象』中描述了管理主题对象。

注: 即使您从 SYSTEM.BASE.TOPIC，为直接继承自 SYSTEM.BASE.TOPIC。 例如，在美国各州的主题空间中， USA/Alabama USA/Alaska 等， USA 是根主题。 根主题的主要用途是创建离散的非重叠主题空间，以避免发布与错误的预订匹配。 这也意味着您可以更改根主题的属性以影响整个主题空间。 例如，可以设置 **CLUSTER** 属性的名称。

以发布者或订户身份引用主题时，您可以选择提供主题字符串，也可以同时引用主题对象，在这种情况下，您提供的主题字符串将定义主题对象的子主题。 队列管理器通过将主题字符串附加到主题对象中指定的主题字符串前缀，在两个主题字符串之间插入额外的 '/' (例如， 主题字符串/对象字符串) 来标识主题。 第 27 页的『组合主题字符串』 进一步对此进行了描述。 生成的主题字符串用于标识主题并将其与管理主题对象相关联。 管理主题对象不一定是与主主题对应的主题对象相同的主题对象。

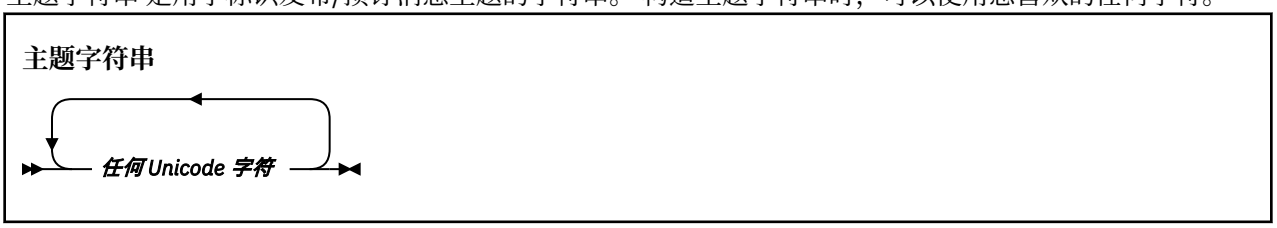
在基于内容的发布/预订中，您可以通过提供用于搜索每条消息内容的选择字符串来定义要接收的消息。 WebSphere MQ 使用扫描消息属性而不是消息的完整内容的消息选择器提供基于内容的发布/预订的中间形式，请参阅 选择器。 消息选择器的原型用途是预订主题，然后使用数字属性限定选择。 选择器使您能够指定您只对特定范围内的值感兴趣; 不能使用字符或基于主题的通配符来执行任何操作。 如果需要根据消息的完整内容进行过滤，那么需要使用 WebSphere Message Broker。

主题字符串

您使用主题字符串将标签信息发布为主题。 然后，使用基于字符或主题的通配主题字符串来预订一组主题。

主题

主题字符串 是用于标识发布/预订消息主题的字符串。 构造主题字符串时，可以使用您喜欢的任何字符。



三个字符在 V 7 发布/预订中具有特殊含义。 在主题字符串中的任何位置都允许这些值，但请谨慎使用这些值。 在 第 23 页的『基于主题的通配方案』中说明了特殊字符的用法。

正斜杠 (/)

主题级别分隔符。 使用 '/' 字符将主题构造为主题树。

如果可以，请避免使用空的主题级别 '//'。 这些节点对应于主题层次结构中没有主题字符串的节点。 主题字符串中的前导或尾部 '/' 对应于前导或尾部空节点，也应避免。

散列符号 (#)

与 '/' 结合使用以在预订中构造多级通配符。 请注意在用于命名已发布主题的主题字符串中使用与 '/' 相邻的 '#'。 第 23 页的『主题字符串示例』 显示 '#'的合理用法。

字符串 '.../#/...'， '#/...' 和 '.../#' 在预订主题字符串中具有特殊含义。 这些字符串与主题层次结构中的一个或多个级别的所有主题匹配。 因此，如果您创建了具有其中一个序列的主题，那么无法预订该主题，同时也无法预订主题层次结构中多个级别的所有主题。

加号 (+)

与 '/' 结合使用以在预订中构造单层通配符。 请注意在用于命名已发布主题的主题字符串中使用与 '/' 相邻的 '+'。

字符串 '.../+/...'， '+/...' 和 '.../+' 在预订主题字符串中具有特殊含义。 这些字符串与主题层次结构中的一个级别的所有主题匹配。 因此，如果您创建了具有其中一个序列的主题，那么无法预订该主题，也无法在主题层次结构中的一个级别预订所有主题。

主题字符串示例

```
IBM/Business Area#/Results
IBM/Diversity/%African American
```

通配方案
有两种用于预订多个主题的通配方案。 对方案的选择是预订选项。

MQSO_WILDCARD_TOPIC
选择要使用基于主题的通配符方案预订的主题。

如果未显式选择通配符模式，那么这是缺省值。

MQSO_WILDCARD_CHAR
选择要使用基于字符的通配符方案预订的主题。

通过在 DEFINE SUB 命令上指定 **wschema** 参数来设置任一方案。 有关更多信息，请参阅 DEFINE SUB 。

注: 在 WebSphere MQ 版本 7.0 之前创建的预订始终使用基于字符的通配符方案。

示例

```
IBM/+/Results
#/Results
IBM/Software/Results
IBM/*ware/Results
```

基于主题的通配方案
基于主题的通配符允许订户每次预订多个主题。

基于主题的通配符是 WebSphere MQ 发布/预订中主题系统的强大功能。 多点传送通配符和单一级别通配符可用于预订，但不能由消息的发布者在主题中使用。

基于主题的通配场景允许您选择按主题级别进行分组的发布内容。 对于主题层次结构中的每个级别，您可以选择该主题级别的订阅中的字符串是否必须与发布内容中的该字符串完全匹配。 例如，预订 IBM/+/Results 选择所有主题，

```
IBM/Software/Results
IBM/Services/Results
IBM/Hardware/Results
```

有两种类型的通配符。

多级别通配符

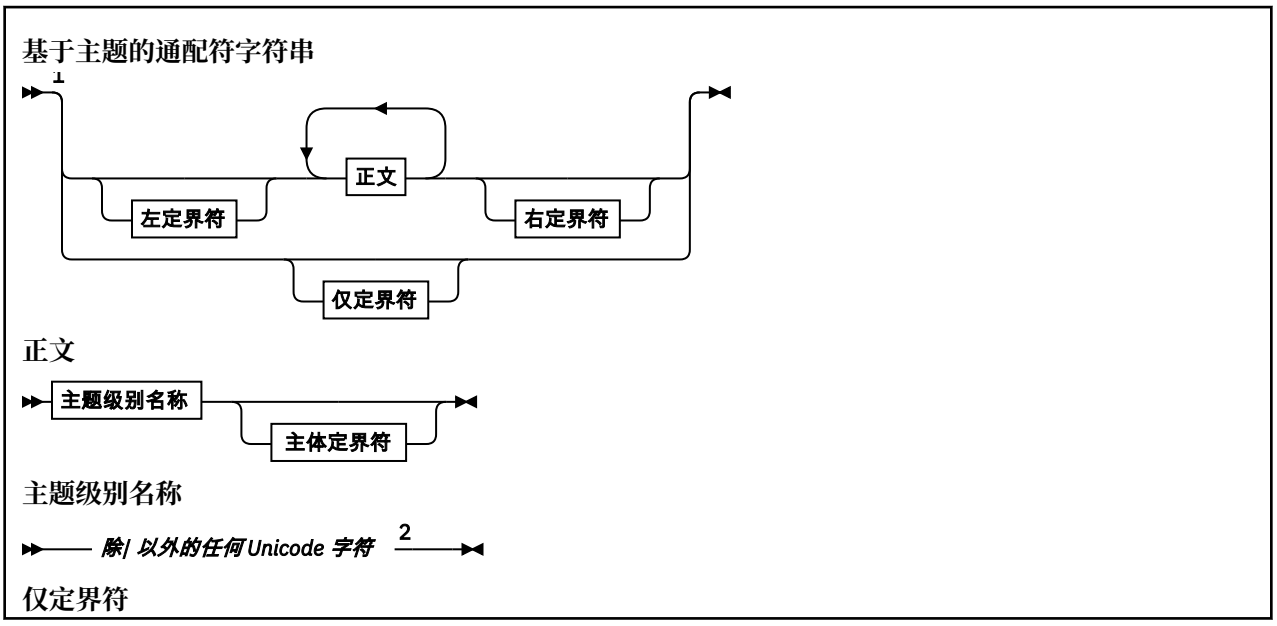
- 多级别通配符用于订阅中。 在发布中使用时，会将其视为文字。
- 多级通配符 '#' 用于匹配主题中的任意数量的级别。 例如，使用示例主题树，如果预订 'USA/Alaska/#'，那么将接收有关主题 'USA/Alaska' 和 'USA/Alaska/Juneau'的消息。
- 多级别通配符可以不表示任何级别，也可以表示多个级别。 因此， 'USA/#' 还可以与奇异 'USA'匹配，其中 '#' 表示零级别。 因为没有要分隔的级别，所以主题级别分隔符在此上下文中没有含义。
- 多级别通配符仅当单独指定或紧随主题级别分隔符指定时才有效。 因此， '#' 和 'USA/#' 是将 '#' 字符视为通配符的有效主题。 但是，虽然 'USA#' 也是有效的主题字符串，但 '#' 字符不会被视为通配符，并且没有任何特殊含义。 请参阅 第 25 页的『当基于主题的通配符无效时』，以了解更多信息。

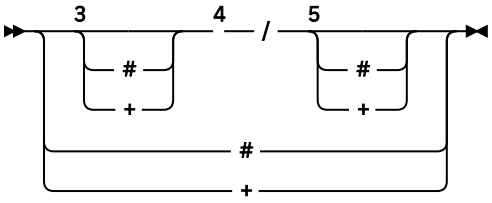
单一级别通配符

- 单一级别通配符用于订阅中。 在发布中使用时，会将其视为文字。
- 单级别通配符 '+' 与一个且仅与一个主题级别匹配。 例如， 'USA/+' 与 'USA/Alabama'匹配，但与 'USA/Alabama/Auburn'不匹配。 因为单级别通配符仅与单个级别匹配，所以 'USA/+' 与 'USA'不匹配。
- 可以在主题树中的任何级别使用单层通配符，并将其与多层通配符结合使用。 除了单独指定单一级别通配符时以外，必须紧随主题级别分隔符指定单一级别通配符。 因此， '+' 和 'USA/+' 是将 '+' 字符视为通配符的有效主题。 但是，虽然 'USA+' 也是有效的主题字符串，但 '+' 字符不会被视为通配符，并且没有任何特殊含义。 请参阅 第 25 页的『当基于主题的通配符无效时』，以了解更多信息。

基于主题的通配符方案的语法没有转义字符。 是否将 '#' 和 '+' 视为通配符取决于它们的上下文。 请参阅 第 25 页的『当基于主题的通配符无效时』，以了解更多信息。

注: 以特殊方式处理主题字符串的开头和结尾。 使用 '$' 来表示字符串的结尾，那么 '$#/...' 是多级通配符， '$/#/..'。 是根处的空节点，后跟多级通配符。

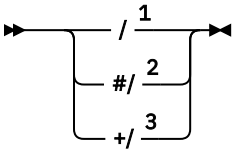




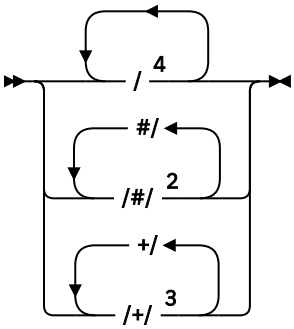
注:

[1] 空或零长度主题字符串无效
[2] 建议您不要在级别名称字符串中使用任何 *, ?, % ， 以实现基于字符的通配符方案与基于主题的通配符方案之间的兼容性。
[3] 这些个案等同于 左定界符 模式。
[4] 没有通配符的 / 与单个空主题匹配。
[5] 这些个案等同于 右定界符 模式。
[6] 匹配每个主题。
[7] 匹配每个只有一个级别的主题。

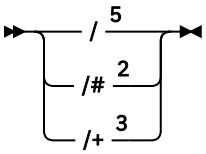
左定界符



主体定界符



右定界符



注:

[1] 主题字符串以空主题开头
[2] 与零个或多个级别匹配。 多个多级匹配字符串与一个多级匹配字符串具有相同的效果。
[3] 正好匹配一个级别。
[4] // 是空主题-没有主题字符串的主题对象。
[5] 主题字符串以空主题结尾

当基于主题的通配符无效时

当通配符 '+' 和 '#' 与主题级别中的其他字符 (包括它们本身) 混合时，它们没有特殊含义。

这意味着可以发布主题级别中包含 '+' 或 '#' 以及其他字符的主题。

例如以下两个主题：

1. level0/level1/+/level4/#
2. level0/level1/#+/level4/level#

在第一个示例中，字符 '+' 和 '#' 被视为通配符，因此在要发布到的主题字符串中无效，但在预订中有效。

在第二个示例中，不会将字符 '+' 和 '#' 视为通配符，因此可以发布和预订主题字符串。

示例

```
IBM/+/Results
#/Results
IBM/Software/Results
```

基于字符的通配符方案
基于字符的通配符方案允许您根据传统字符匹配来选择主题。

您可以使用字符串 '*'在主题层次结构中选择多个级别的所有主题。 在基于字符的通配符方案中使用 '*' 等同于使用基于主题的通配符字符串 '#'

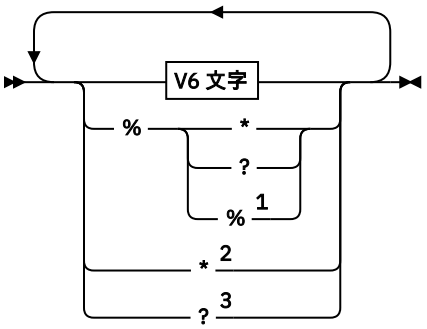
'*x*/*/*y*' 等同于基于主题的方案中的 '*x*/#/*y*'，并在级别 '*x* 和 *y*'之间选择主题层次结构中的所有主题，其中 '*x*' 和 '*y*' 是不在通配符返回的级别集中的主题名称。

基于主题的方案中的 '/+/' 在基于字符的方案中没有完全相同的内容。 'IBM/*/Results' 还将选择 'IBM/Patents/Software/Results'。 仅当层次结构的每个级别的主题名称集都是唯一的时，才能始终使用生成相同匹配项的两个方案来构造查询。

以一般方式使用，基于字符的方案中的 '*' 和 '?' 在基于主题的方案中没有等效项。 基于主题的方案不会使用通配符执行部分匹配。 基于字符的通配符预订 'IBM/*ware/Results' 没有基于主题的等效项。

注: 使用字符通配符预订的匹配比使用基于主题的预订的匹配慢。

基于字符的通配符字符串



V6 文字

除*和*以外的任何 Unicode 字符？ *和*%

注：

[1] 表示 "转义以下字符"，以便将其视为字面值。 '%' 必须后跟 '*'， '?' 或 '%'。 请参阅 第 23 页的『主题字符串示例』。
[2] 表示预订中的 "匹配零个或多个字符"。
[3] 表示预订中的 "只匹配一个字符"。

示例

```
IBM/*/Results
IBM/*ware/Results
```

组合主题字符串

创建预订或打开主题以使您可以向其发布消息时，可以通过组合两个单独的子主题字符串 (即 “子主题”) 来构成主题字符串。 一个子主题由应用程序或管理命令作为主题字符串提供，另一个子主题是与主题对象相关联的主题字符串。 您可以单独使用子主题作为主题字符串，也可以将其组合以形成新的主题名称。

例如，使用 MQSC 命令 **DEFINE SUB** 定义预订时，该命令可以将 **TOPICSTR** (主题字符串) 和/或 **TOPICOBJ** (主题对象) 作为属性。 如果仅提供了 **TOPICOBJ**，那么与该主题对象关联的主题字符串将用作主题字符串。 如果仅提供了 **TOPICSTR**，那么将用作主题字符串。 如果同时提供了这两个字符串，那么将它们并置以形成格式为 **TOPICOBJ**/**TOPICSTR** 的单个主题字符串，其中 **TOPICOBJ** 配置的主题字符串始终是第一个，并且该字符串的两个部分始终以 “/” 字符分隔。

同样，在 MQI 程序中，完整主题名称由 MQOPEN 创建。 它由发布/预订 MQI 调用中使用的两个字段组成，按列出的顺序:

1. 主题对象的 **TOPICSTR** 属性，在 **ObjectName** 字段中指定。
2. 定义应用程序提供的子主题的 **ObjectString** 参数。

生成的主题字符串将在 **ResObjectString** 参数中返回。

如果每个字段的第一个字符不是空白或空字符，并且字段长度大于零，那么这些字段被视为存在。 如果仅存在其中一个字段，那么将其用作主题名称。 如果两个字段都没有值，那么调用将失败，原因码为 MQRC_UNKNOWN_OBJECT_NAME; 如果完整主题名称无效，那么调用将失败 MQRC_TOPIC_STRING_ERROR 。

如果两个字段都存在，那么将在生成的组合主题名称的两个元素之间插入 “/” 字符。

第 27 页的表 2 显示了主题字符串并置的示例:

表 *2:* 主题字符串并置示例

主题对象的 TOPICSTR	**由应用程序或 DEFINE SUB 命令提供的主题字符串**	**完整主题名称**	**注释**
足球/苏格兰人	''	足球/苏格兰人	单独使用主题对象的 TOPICSTR。
''	足球/苏格兰人	足球/苏格兰人	单独使用 ObjectString/ TOPICSTR。
美式足球	评分	足球/苏格兰人	在并置点添加 “/” 字符。
美式足球	/得分	足球//苏格兰人	在两个字符串之间生成 "空节点"。 这与 “Football/Scores”不同。
/足球	评分	/足球/苏格兰人	主题以 "空节点" 开头。 这与 “Football/Scores” 不同。

“/” 字符被视为特殊字符，为 第 28 页的『主题树』中的完整主题名称提供结构。 不得将 “/” 字符用于任何其他原因，因为主题树的结构受到影响。 主题 “/Football” 与主题 “Football”不同。

注: 如果在创建预订时使用主题对象，那么主题对象主题字符串的值在定义时在预订中固定。 对主题对象的任何后续更改都不会影响预订所定义到的主题字符串。

主题字符串中的通配符

以下通配符是特殊字符:

- 加号 (+)
- 编号符号 (#)
- 星号 (*)
- 问号 (?)

仅当预订使用通配符时，通配符才具有特殊含义。 在其他位置使用时，这些字符不会被视为无效，但是您必须确保了解如何使用这些字符，并且在发布或定义主题对象时，您可能不希望在主题字符串中使用这些字符。

如果在主题级别中使用 # 或 + 与其他字符 (包括其本身) 混合的主题字符串进行发布，那么可以使用通配符方案来预订该主题字符串。

如果发布主题字符串时使用 # 或 + 作为两个 / 字符之间的唯一字符，那么应用程序无法使用通配符方案 MQSO_WILDCARD_TOPIC 显式预订主题字符串。 这种情况会导致应用程序获得比预期更多的发布。

不应在定义的主题对象的主题字符串中使用通配符。 如果执行此操作，那么当发布程序使用对象时，该字符将被视为字面值字符，当预订使用对象时，该字符将被视为通配符。 这会导致混乱。

示例代码片段

从示例程序 Example 2: Publisher to a variable topic 中抽取的此代码片段将主题对象与变量主题字符串组合在一起:

```
MQOD     td = {MQOD_DEFAULT};  /* Object Descriptor            */
td.ObjectType = MQOT_TOPIC;    /* Object is a topic            */
td.Version = MQOD_VERSION_4;   /* Descriptor needs to be V4    */
strncpy(td.ObjectName, topicName,  MQ_TOPIC_NAME_LENGTH);
td.ObjectString.VSPtr = topicString;
td.ObjectString.VSLength = (MQLONG)strlen(topicString);
td.ResObjectString.VSPtr = resTopicStr;
td.ResObjectString.VSBufSize = sizeof(resTopicStr)-1;
MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
```

主题树

您定义的每个主题都是主题树中的一个元素或节点。 主题树可以为空以开始，也可以包含先前使用 MQSC 或 PCF 命令定义的主题。 通过使用“创建主题”命令或通过指定主题，可以在发布或预订中首次定义新的主题。

虽然可以使用任何字符串来定义主题的主题字符串，但建议选择适合分层树结构的主题字符串。 经过深思熟虑的主题刺和主题树设计可以帮助您执行以下操作:

- 预订多个主题。
- 建立安全策略。

虽然您可以将主题树构造为平面的线性结构，但最好在具有一个或多个根主题的分层结构中构建主题树。 有关安全规划和主题的更多信息，请参阅 发布/预订安全性。

第 28 页的图 5 显示了具有一个根主题的主题树的示例。

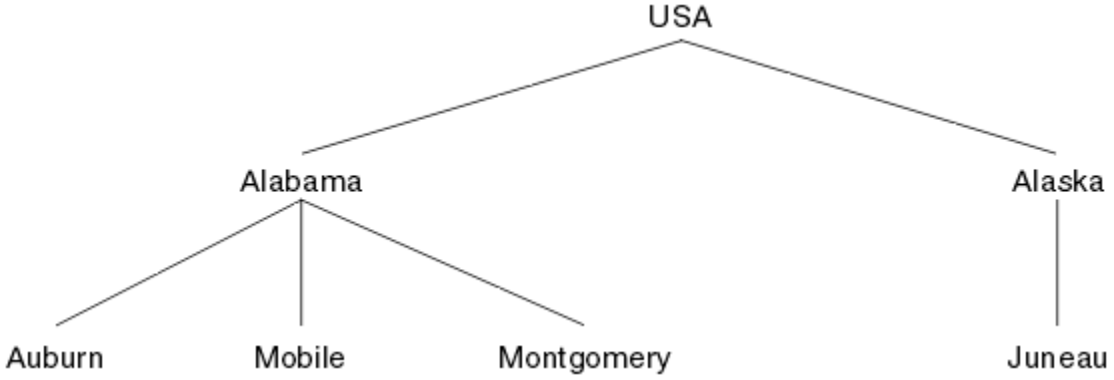


图 *5:* 主题树的示例

图中的每个字符串表示主题树中的一个节点。 通过聚集主题树中一个或多个级别的节点来创建完整的主题字符串。 级别由 "/" 字符分隔。 完全指定的主题字符串的格式为: "root/level2/level3"。

第 28 页的图 5 中显示的主题树中的有效主题包括:

"美国"
"美国/阿拉巴马州"
"美国/阿拉斯加"
"USA/Alabama/Auburn"
"USA/Alabama/Mobile"
"USA/Alabama/Montgomery"
"USA/Alaska/Juneau"

在设计主题字符串和主题树时，请记住队列管理器不会解释或尝试派生主题字符串本身的含义。 它只是使用主题字符串将所选消息发送到该主题的订户。

下列原则应用到主题树的构造和内容:

- 对主题树中级别的数目没有限制。
- 对主题树中级别的名称长度没有限制。
- 对“根”节点的数量无限制，即对主题树的数量无限制。

减少主题树中不需要的主题数
通过减少主题树中不需要的主题数来提高发布/预订系统的性能。 什么是不需要的主题以及如何除去这些主题?

您可以创建大量主题，而不会对性能产生负面影响。 但是，使用发布/预订的某些方法会导致不断扩展主题树。 将创建一次异常大的主题，并且不再使用这些主题。 越来越多的主题可能成为性能问题。

如何避免设计导致大量且越来越多的不需要的主题? 您可以执行哪些操作来帮助队列管理器从主题树中除去不需要的主题?

队列管理器可识别不需要的主题，因为它已未使用 30 分钟。 队列管理器将从主题树中为您除去未使用的主题。 可以通过改变队列管理器属性 **TREELIFE** 来更改 30 分钟的持续时间。 您可以帮助队列管理器除去不需要的主题，方法是确保该主题对队列管理器显示为未使用。 第 29 页的『什么是未使用的主题?』 部分说明了什么是未使用的主题。

一个程序员，设计任何应用程序，尤其是设计一个长时间运行的应用程序，会考虑它的资源使用情况: 程序需要多少资源，是否有任何无界的需求，以及任何资源泄漏? 主题是发布/预订程序使用的资源。 与程序使用的任何其他资源一样，仔细检查主题的使用情况。

什么是未使用的主题?

在定义未使用的主题之前，什么是主题?

将主题字符串 (例如 `USA/Alabama/Auburn`) 转换为主题时，会将该主题添加到主题树中。 如果需要，将在树中创建其他主题节点及其相应的主题。 主题字符串 `USA/Alabama/Auburn` 将转换为具有三个主题的树。

- `USA`
- `USA/Alabama`
- `USA/Alabama/Auburn`

要显示主题树中的所有主题，请使用 **runmqsc** 命令 `DISPLAY TPSTATUS('#') TYPE(TOPIC)`。

主题树中未使用的主题具有以下属性。

它未与主题对象关联

管理主题对象具有将其与主题关联的主题字符串。 当您定义主题对象 `Alabama` 时，如果要与主题 `USA/Alabama` 关联的主题不存在，那么将根据主题字符串创建主题。 如果主题确实存在，那么将使用主题字符串将主题对象与主题关联在一起。

它没有保留发布

具有保留发布的主题由发布程序生成，该发布程序使用选项 MQPMO_RETAIN 将消息放入主题。

使用 **runmqsc** 命令 DISPLAY TPSTATUS('USA/Alabama') RETAINED 来检查 USA/Alabama 是否具有保留发布。 响应为 YES 或 NO。

使用 **runmqsc** 命令 CLEAR TOPICSTR('USA/Alabama') CLTRTYPE(RETAINED) 从 USA/Alabama 中除去保留发布。

它没有子主题

USA/Alabama/Auburn 是没有子主题的主题。 USA/Alabama/Auburn 是 USA/Alabama 的直接子主题。

使用 **runmqsc** 命令 DISPLAY TPSTATUS('USA/Alabama/+') 显示 USA/Alabama 的直接子代。

节点没有活动发布程序

节点的活动发布程序是打开主题以进行输出的应用程序。

例如，应用程序使用打开选项 MQOO_OUTPUT 打开名为 **Alabama** 的主题对象。

要向 USA/Alabama 及其所有子代显示活动发布程序，请使用 **runmqsc** 命令 DISPLAY TPSTATUS('USA/Alabama/#') TYPE(PUB) ACTCONN。

没有节点的活动订户

活动订户可以是持久预订，也可以是已向 MQSUB 注册主题预订但未将其关闭的应用程序。

要显示 USA/Alabama 的活动预订，请使用 **runmqsc** 命令 DISPLAY TPSTATUS('USA/Alabama') TYPE(SUB) ACTCONN。

要显示 USA/Alabama 及其所有子代的活动预订，请使用 **runmqsc** 命令 DISPLAY TPSTATUS('USA/Alabama/#') TYPE(SUB) ACTCONN。

减少主题树中的主题数

总之，有多种方法可以减少主题树中的主题数。

修改 TREELIFE

缺省情况下，未使用的主题的生存期为 30 分钟。 您可以使未使用的主题的生存期变小。

例如， **runmqsc** 命令 ALTER QMGR TREELIFE(900)将未使用的主题的生存期从 30 分钟缩短到 15 分钟。

在异常情况下，重新启动队列管理器

重新启动队列管理器时，将从主题对象，具有保留发布的节点和持久预订重新初始化主题树。 将消除由发布程序和订户程序的操作创建的主题。

定期使用 **runmqsc** 命令 DISPLAY TPSTATUS('#') TYPE(TOPIC) 来列出所有主题，并检查数字是否正在增长。

作为最后的手段，如果不需要的主题的增长是过去导致性能问题的原因，请重新启动队列管理器。

管理主题对象

通过使用管理主题对象，可以将特定非缺省属性分配给主题。

第 30 页的图 6 显示了如何将划分为不同体育项目的不同主题的 Sport 高级别主题可视化为主题树:

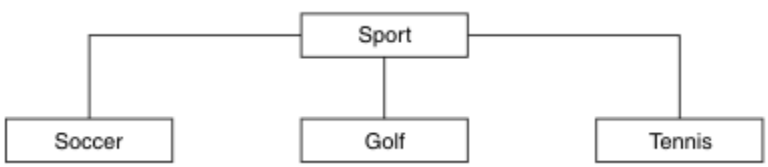


图 *6:* 主题树的可视化

第 31 页的图 7 显示了如何进一步划分主题树，以分隔有关每项运动的不同类型的信息:

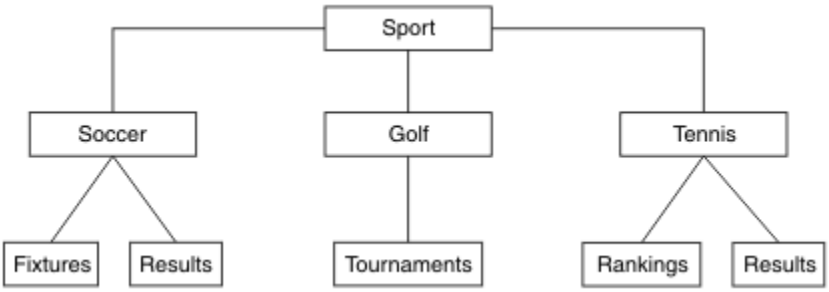


图 *7:* 扩展主题树

要创建说明的主题树，不需要定义任何管理主题对象。 此树中的每个节点都由在发布或预订操作中创建的主题字符串定义。 树中的每个主题都从其父代继承其属性。 属性继承自父主题对象，因为缺省情况下，所有属性都设置为 ASPARENT。 在此示例中，每个主题都具有与 Sport 主题相同的属性。 Sport 主题没有管理主题对象，并且从 SYSTEM.BASE.TOPIC。

请注意，在主题树的根节点 (即 SYSTEM.BASE.TOPIC，因为权限是继承的，但不能限制。 因此，通过授予此级别的权限，您将授予整个树的权限。 您应该在层次结构中的较低主题级别授予权限。

管理主题对象可用于定义主题树中特定节点的特定属性。 在以下示例中，定义了管理主题对象以将足球主题的持久预订属性 DURSUB 设置为值 NO:

```
DEFINE TOPIC(FOOTBALL.EUROPEAN)
       TOPICSTR('Sport/Soccer')
       DURSUB(NO)
       DESCR('Administrative topic object to disallow durable subscriptions')
```

现在可以将主题树可视化为:

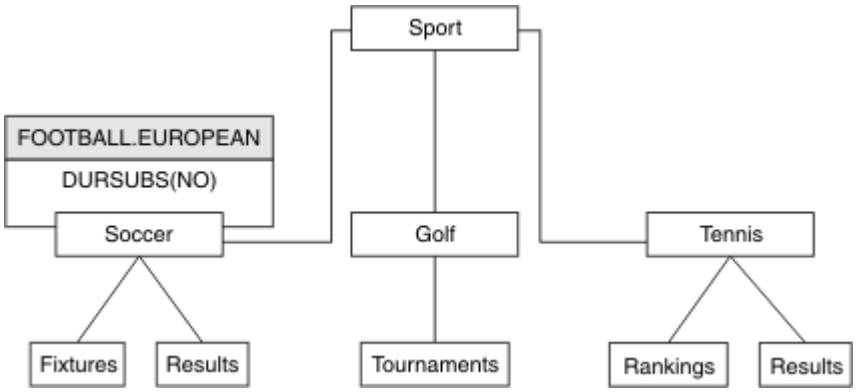


图 *8:* 与 *Sport/Soccer* 主题关联的管理主题对象的可视化

预订树中 Soccer 下的主题的任何应用程序仍可以使用它们在添加管理主题对象之前使用的主题字符串。 但是，现在可以使用对象名 FOOTBALL.EUROPEAN (而不是字符串 /Sport/Soccer) 来编写应用程序以进行预订。 例如，要预订 /Sport/Soccer/Results，应用程序可以将 MQSD.ObjectName 指定为 FOOTBALL.EUROPEAN，将 MQSD.ObjectString 指定为 Results。

通过此功能，您可以向应用程序开发者隐藏主题树的部分内容。 在主题树中的特定节点上定义管理主题对象，然后应用程序开发者可以将自己的主题定义为该节点的子代。 开发者必须了解父主题，但不能了解父树中的任何其他节点。

继承属性

如果主题树具有许多管理主题对象，那么缺省情况下，每个管理主题对象都将从其最接近的父管理主题继承其属性。 先前的示例已在 第 32 页的图 9 中扩展:

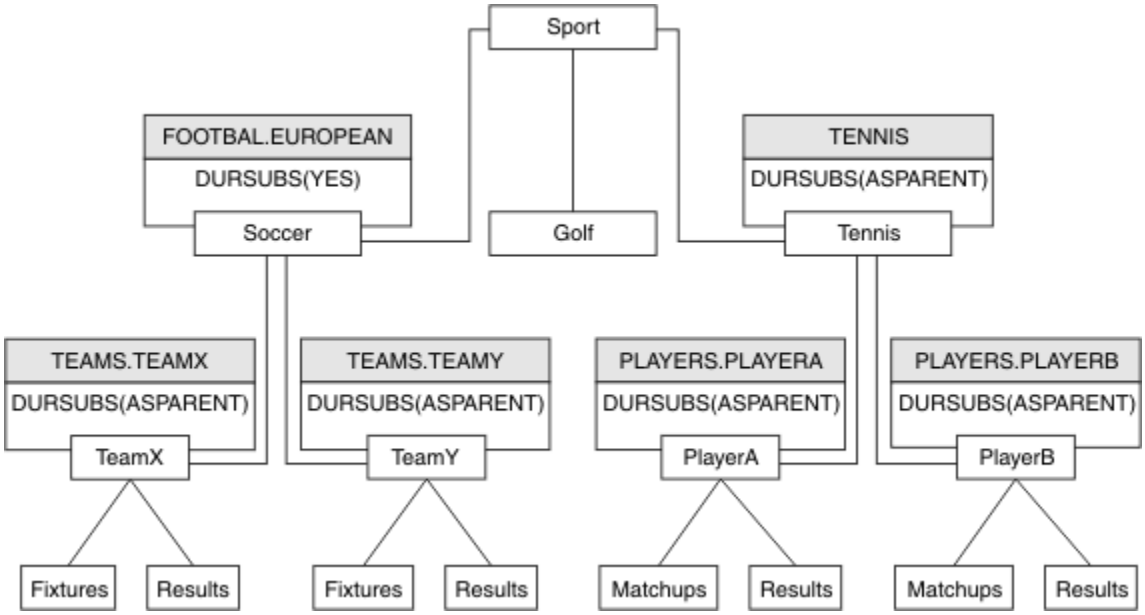


图 *9:* 具有多个管理主题对象的主题树

例如，使用继承为 /Sport/Soccer 的所有子主题提供预订为非持久预订的属性。 将 FOOTBALL.EUROPEAN 的 DURSUB 属性更改为 NO。

可使用以下命令设置此属性:

```
ALTER TOPIC(FOOTBALL.EUROPEAN) DURSUB(NO)
```

Sport/Soccer 的子主题的所有管理主题对象都将属性 DURSUB 设置为缺省值 ASPARENT。 将 FOOTBALL.EUROPEAN 的 DURSUB 属性值更改为 NO 后， Sport/Soccer 的子主题将继承 DURSUB 属性值 NO 。 Sport/Tennis 的所有子主题从 SYSTEM.BASE.TOPIC 对象继承 DURSUB 的值。 SYSTEM.BASE.TOPIC 的值为 YES。

尝试对主题 Sport/Soccer/TeamX/Results 进行持久预订现在将失败; 但是，尝试对 Sport/Tennis/PlayerB/Results 进行持久预订将成功。

使用 通配符 属性控制通配符使用

使用 MQSC **Topic** 通配符 属性或等效 PCF 主题 WildcardOperation 属性来控制向使用通配符主题字符串名称的订户应用程序传递发布内容。 通配符 属性可以具有以下两个可能的值之一:

WILDCARD
与此主题有关的通配符预订的行为。

PASSTHRU
对没有此主题对象中主题字符串具体的通配主题进行的预订将接收对此主题以及比此主题更具体的主题字符串进行的发布。

BLOCK
对没有此主题对象中主题字符串具体的通配主题进行的预订不会接收对此主题或比此主题更具体的主题字符串进行的发布。

在定义预订时将使用此属性的值。 如果改变此属性，那么现有预订涵盖的主题集不会因为此修改而受到影响。 如果在创建或删除主题对象时拓扑发生更改，此场景也适用；将使用修改后的拓扑来创建与修改 WILDCARD 属性后创建的预订匹配的主题集。 如果要针对现有预订强制重新评估匹配的主题集，那么必须重新启动队列管理器。

在示例 第 36 页的『示例: 创建 Sport 发布/预订集群』中，您可以遵循步骤来创建 第 33 页的图 10 中显示的主题树结构。

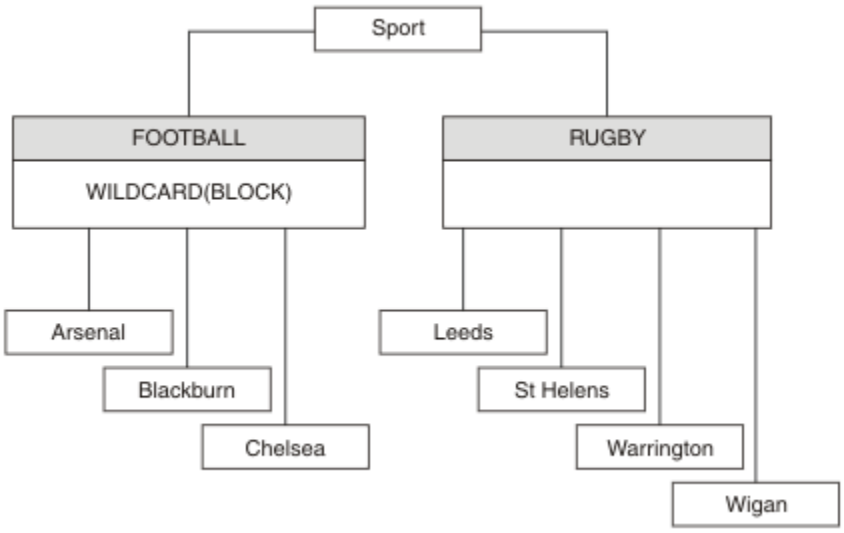


图 *10:* 使用 通配符 属性 *BLOCK* 的主题树

使用通配符主题字符串 *#* 的订户将接收到 Sport 主题和 Sport/Rugby 子树的所有发布。 订户不会接收到 Sport/Football 子树的发布，因为 Sport/Football 主题的 **通配符** 属性值为 BLOCK。

PASSTHRU 是缺省设置。 您可以将 **通配符** 属性值 PASSTHRU 设置为 Sport 树中的节点。 如果节点没有 **通配符** 属性值 BLOCK，那么设置 PASSTHRU 不会改变 Sports 树中节点的订户观察到的行为。

在此示例中，创建预订以查看通配符设置如何影响交付的发布; 请参阅 第 37 页的图 14。 在 第 38 页的图 17 中运行发布命令以创建一些发布。

```
pub QMA
```

图 *11:* 发布到 *QMA*

第 33 页的表 3 显示了结果。 请注意，如何设置 **通配符** 属性值 BLOCK，以防止具有通配符的预订将发布内容接收到通配符作用域内的主题。

表 *3: QMA* 上收到的出版物

预订	**主题字符串**	**收到的出版物**	**注**
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	WILDCARD(BLOCK) 在 Sports/Football 上阻止了对 Football 子树的所有发布
SARSENAL	Sports/#/Arsenal	-	Sports/Football 上的 WILDCARD(BLOCK) 会阻止 Arsenal 上的通配符预订
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby 上的缺省 WILDCARD 不会阻止 Leeds 上的通配符预订。

注:

假设预订具有通配符，该通配符与具有 **通配符** 属性值 BLOCK 的主题对象相匹配。 如果预订还具有匹配通配符右侧的主题字符串，那么预订将从不接收发布内容。 未被阻止的发布集合是作为被阻止通配符的父代的主题的发布。 通配符将阻止发布到作为具有 BLOCK 属性值的主题的子代的主题。 因此，包含通配符右侧主题的预订主题字符串永远不会收到任何要匹配的发布内容。

将 WILDCARD 属性值设置为 BLOCK 并不意味着您无法使用包含通配符的主题字符串进行预订。 这样的订阅是正常的。 预订具有一个显式主题，该主题与具有 **通配符** 属性值 BLOCK 的主题对象匹配。 它将通配符用

于作为具有 **通配符** 属性值 BLOCK 的主题的父代或子代的主题。 在 第 33 页的图 10 中的示例中，预订 (例如 Sports/Football/#) 可以接收发布内容。

通配符和集群主题

集群主题定义将传播到集群中的每个队列管理器。 在集群中的一个队列管理器上预订集群主题会导致队列管理器创建代理预订。 将在集群中的每个其他队列管理器上创建代理预订。 使用包含通配符的主题字符串 (与集群主题结合使用) 的预订可能难以预测行为。 此行为在以下示例中进行了说明。

在为示例 第 36 页的『示例: 创建 Sport 发布/预订集群』设置的集群中， QMB 具有与 QMA 相同的预订集，但 QMB 在发布者发布到 QMA 之后未收到任何发布，请参阅 第 33 页的图 11。 虽然 Sports/Football 和 Sports/Rugby 主题是集群主题，但 fullsubs.tst 中定义的预订不会引用集群主题。 不会将代理预订从 QMB 传播到 QMA。 如果没有代理预订，那么不会将 QMA 的任何发布转发到 QMB 。

某些预订 (例如 Sports/#/Leeds) 可能似乎引用了集群主题，在本例中为 Sports/Rugby 。 Sports/#/Leeds 预订实际解析为主题对象 SYSTEM.BASE.TOPIC。

用于解析预订 (例如， Sports/#/Leeds) 所引用的主题对象的规则如下所示。 将主题字符串截断为第一个通配符。 通过主题字符串向左扫描以查找具有关联管理主题对象的第一个主题。 主题对象可以指定集群名称，也可以定义本地主题对象。 在示例 Sports/#/Leeds 中，截断后的主题字符串是 Sports，它没有主题对象，因此 Sports/#/Leeds 继承自 SYSTEM.BASE.TOPIC(这是本地主题对象)。

要查看预订集群主题如何更改通配符传播的工作方式，请运行批处理脚本 upsubs.bat。 该脚本将清除预订队列，并在 fullsubs.tst 中添加集群主题预订。 再次运行 puba.bat 以创建一批发布; 请参阅 第 33 页的图 11。

第 34 页的表 4 显示了将两个新预订添加到发布发布的同一队列管理器的结果。 结果与预期相同，新预订各接收一个发布，其他预订接收的发布数量保持不变。 意外结果发生在其他集群队列管理器上; 请参阅 第 35 页的表 5。

表 *4: QMA* 上收到的出版物

预订	**主题字符串**	**收到的出版物**	**注**
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	WILDCARD(BLOCK) 在 Sports/Football 上阻止了对 Football 子树的所有发布
SARSENAL	Sports/#/Arsenal	-	Sports/Football 上的 WILDCARD(BLOCK) 会阻止 Arsenal 上的通配符预订
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby 上的缺省 WILDCARD 不会阻止 Leeds 上的通配符预订。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal 接收发布，因为预订没有通配符。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds 将在任何情况下接收发布。

第 35 页的表 5 显示了在 QMB 上添加两个新预订以及在 QMA 上发布的结果。 请注意，在没有这两个新预订的情况下， QMB 未收到任何发布。 正如预期的那样，这两个新预订将接收发布内容，因为 Sports/FootBall 和 Sports/Rugby 都是集群主题。 QMB 将 Sports/Football/Arsenal 和 Sports/Rugby/Leeds 的代理预订转发到 QMA，然后将发布发送到 QMB。

意外的结果是，先前未收到任何发布的两个预订 Sports/# 和 Sports/#/Leeds 现在接收发布。 原因是针对其他预订转发到 QMB 的 Sports/Football/Arsenal 和 Sports/Rugby/Leeds 发布现在可用于连接到 QMB 的任何订户。 因此，本地主题 Sports/# 和 Sports/#/Leeds 的预订将接收 Sports/Rugby/Leeds 发布。 Sports/#/Arsenal 继续不接收发布内容，因为 Sports/Football 已将其 **通配符** 属性值设置为 BLOCK。

表 *5: QMB* 上收到的出版物			
预订	**主题字符串**	**收到的出版物**	**注**
SPORTS	*Sports/#*	*Sports/Rugby/Leeds*	*WILDCARD(BLOCK)* 在 *Sports/Football* 上阻止了对 *Football* 子树的所有发布
SARSENAL	Sports/#/Arsenal	-	Sports/Football 上的 WILDCARD(BLOCK) 会阻止 Arsenal 上的通配符预订
SLEEDS	*Sports/#/Leeds*	*Sports/Rugby/Leeds*	*Sports/Rugby* 上的缺省 *WILDCARD* 不会阻止 *Leeds* 上的通配符预订。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal 接收发布，因为预订没有通配符。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds 将在任何情况下接收发布。

在大多数应用程序中，不希望一个预订影响另一个预订的行为。 **通配符** 属性与值 BLOCK 的一个重要用途是使包含通配符的同一主题字符串的预订行为一致。 无论预订是在与发布程序相同的队列管理器上，还是在不同的队列管理器上，预订的结果都是相同的。

通配符和流

WebSphere MQ V 6 流由 WebSphere MQ V 7 映射到主题; 请参阅 第 38 页的『流和主题』。 在 **strmqbrk** 在 V 7 中执行的缺省映射中，流 Sports 中的所有主题都将映射到主题 Sports。 流 Business 中的所有主题都将映射到主题 Business。

Sports 流中的 WebSphere MQ V 6 到 * 的预订将接收 Sports 树中的所有发布内容，而 Business 树中没有发布内容。 版本 7 中的同一预订将接收 Sports 树中的所有发布以及 Business 树中的所有发布。 要阻止此行为，当流迁移到 V 7 时， **strmqbrk** 会设置 **通配符** 属性。 对于从流迁移的每个顶级主题，它会将其设置为值 BLOCK。 通过从名为 Sports 和 Business 的版本 6 流进行转换，将 Sports 和 Business 的 **通配符** 属性设置为值 BLOCK。

对于写入发布/预订 API 的新应用程序，结果是对 * 的预订不会收到任何发布。 要接收所有体育出版物，必须预订 Sports/*或 Sports/# 以及类似的 Business 出版物。

将发布/预订代理迁移到 WebSphere MQ V 7 时，现有已排队的发布/预订应用程序的行为不会更改。 **Publish**， **Register Publisher** 或 **Subscriber** 命令中的 **StreamName** 属性将映射到流已迁移到的主题的名称。

通配符和预订点

WebSphere Message Broker 预订点由 WebSphere MQ V 7 映射到主题; 请参阅 第 40 页的『预订点和主题』。 在 **migmqbrk** 在 V 7 中执行的缺省映射中，预订点 Sports 中的所有主题都将映射到主题 Sports。 预订点 Business 中的所有主题都将映射到主题 Business。

WebSphere Message Broker V 6 到 Sports 预订点中 * 的预订将接收 Sports 树中的所有发布内容，而 Business 树中没有发布内容。 版本 7 中的同一预订将接收 Sports 树中的所有发布以及 Business 树中的所有发布。 要阻止此行为，当预订点迁移到 V 7 时， **migmqbrk** 会设置 **通配符** 属性。 对于从预订点迁移的每个顶级主题，它会将其设置为值 BLOCK。 通过从名为 Sports 和 Business 的 WebSphere Message Broker 预订点进行转换，将 Sports 和 Business 的 **通配符** 属性设置为值 BLOCK。

对于写入发布/预订 API 的新应用程序，迁移的效果是，对 * 的预订不会收到任何发布。 要接收所有体育出版物，必须预订 Sports/*或 Sports/# 以及类似的 Business 出版物。

将发布/预订代理迁移到 WebSphere MQ V 7 时，现有已排队的发布/预订应用程序的行为不会更改。 **Publish**， **Register Publisher** 或 **Subscriber** 命令中的 **SubPoint** 属性将映射到预订已迁移到的主题的名称。

示例: 创建 Sport 发布/预订集群

后续步骤将创建一个具有四个队列管理器的集群 CL1: 两个完整存储库 CL1A 和 CL1B 以及两个部分存储库 QMA 和 QMB。 完整存储库仅用于保存集群定义。 QMA 被指定为集群主题主机。 持久预订同时在 QMA 和 QMB 上定义。

注: 此示例针对 Windows 进行编码。 您必须重新编码 Create qmgrs.bat 和 create pub.bat 以在其他平台上配置和测试示例。

1. 创建脚本文件。
   a. 创建 topics.tst
   b. 创建 wildsubs.tst
   c. 创建 fullsubs.tst
   d. 创建 qmgrs.bat
   e. 创建 pub.bat
2. 运行 Create qmgrs.bat 以创建配置。

```
qmgrs
```

在 第 33 页的图 10 中创建主题。 图 5 中的脚本将创建集群主题 Sports/Football 和 Sports/Rugby。

注: REPLACE 选项不会替换主题的 TOPICSTR 属性。 TOPICSTR 是在示例中进行有效更改以测试不同主题树的属性。 要更改主题，请先删除该主题。

```
DELETE TOPIC ('Sports')
DELETE TOPIC ('Football')
DELETE TOPIC ('Arsenal')
DELETE TOPIC ('Blackburn')
DELETE TOPIC ('Chelsea')
DELETE TOPIC ('Rugby')
DELETE TOPIC ('Leeds')
DELETE TOPIC ('Wigan')
DELETE TOPIC ('Warrington')
DELETE TOPIC ('St. Helens')

DEFINE TOPIC ('Sports')     TOPICSTR('Sports')
DEFINE TOPIC ('Football')   TOPICSTR('Sports/Football') CLUSTER(CL1) WILDCARD(BLOCK)
DEFINE TOPIC ('Arsenal')    TOPICSTR('Sports/Football/Arsenal')
DEFINE TOPIC ('Blackburn')  TOPICSTR('Sports/Football/Blackburn')
DEFINE TOPIC ('Chelsea')    TOPICSTR('Sports/Football/Chelsea')
DEFINE TOPIC ('Rugby')      TOPICSTR('Sports/Rugby')    CLUSTER(CL1)
DEFINE TOPIC ('Leeds')      TOPICSTR('Sports/Rugby/Leeds')
DEFINE TOPIC ('Wigan')      TOPICSTR('Sports/Rugby/Wigan')
DEFINE TOPIC ('Warrington') TOPICSTR('Sports/Rugby/Warrington')
DEFINE TOPIC ('St. Helens') TOPICSTR('Sports/Rugby/St. Helens')
```

图 *12:* 删除并创建主题*: topics.tst*

注: 删除主题，因为 REPLACE 不会替换主题字符串。

使用通配符创建预订。 通配符将主题与 第 33 页的图 10 中的主题对象对应。 为每个预订创建一个队列。 运行或重新运行脚本时，将清除队列并删除预订。

注: REPLACE 选项不会替换预订的 TOPICOBJ 或 TOPICSTR 属性。 TOPICOBJ 或 TOPICSTR 是在用于测试不同预订的示例中进行有效更改的属性。 要对其进行更改，请先删除预订。

```
DEFINE QLOCAL(QSPORTS)   REPLACE
DEFINE QLOCAL(QSARSENAL) REPLACE
DEFINE QLOCAL(QSLEEDS)   REPLACE
CLEAR  QLOCAL(QSPORTS)
CLEAR  QLOCAL(QSARSENAL)
CLEAR  QLOCAL(QSLEEDS)

DELETE SUB   (SPORTS)
DELETE SUB   (SARSENAL)
DELETE SUB   (SLEEDS)
DEFINE SUB   (SPORTS)   TOPICSTR('Sports/#')         DEST(QSPORTS)
DEFINE SUB   (SARSENAL) TOPICSTR('Sports/+/Arsenal') DEST(QSARSENAL)
DEFINE SUB   (SLEEDS)   TOPICSTR('Sports/+/Leeds')   DEST(QSLEEDS)
```

图 *13:* 创建通配符预订*: wildsubs.tst*

创建引用集群主题对象的预订。

注:

将在 TOPICOBJ 引用的主题字符串与 TOPICSTR 定义的主题字符串之间自动插入定界符 /。

定义 DEFINE SUB(FARSENAL) TOPICSTR('Sports/Football/Arsenal') DEST(QFARSENAL) 将创建相同的预订。 TOPICOBJ 用作引用您已定义的主题字符串的快速方法。 创建预订时，不再引用主题对象。

```
DEFINE QLOCAL(QFARSENAL) REPLACE
DEFINE QLOCAL(QRLEEDS)   REPLACE
CLEAR  QLOCAL(QFARSENAL)
CLEAR  QLOCAL(QRLEEDS)

DELETE SUB   (FARSENAL)
DELETE SUB   (RLEEDS)
DEFINE SUB   (FARSENAL) TOPICOBJ('Football') TOPICSTR('Arsenal') DEST(QFARSENAL)
DEFINE SUB   (RLEEDS)   TOPICOBJ('Rugby')    TOPICSTR('Leeds')   DEST(QRLEEDS)
```

图 *14:* 删除并创建预订*: fullsubs.tst*

创建具有两个存储库的集群。 创建两个用于发布和预订的部分存储库。 重新运行脚本以删除所有内容，然后再次启动。 该脚本还会创建主题层次结构和初始通配符预订。

注:

在其他平台上，编写类似的脚本，或者输入所有命令。 通过使用脚本，可以快速删除所有内容，并使用相同的配置重新开始。

```
@echo off
set port.CL1B=1421
set port.CLIA=1420
for %%A in (CL1A CL1B QMA QMB) do call :createQM %%A
call :configureQM CL1A CL1B %port.CL1B% full
call :configureQM CL1B CL1A %port.CL1A% full
for %%A in (QMA QMB) do call :configureQM %%A CL1A %port.CL1A% partial
for %%A in (topics.tst wildsubs.tst) do runmqsc QMA < %%A
for %%A in (wildsubs.tst) do runmqsc QMB < %%A
goto:eof

:createQM
echo Configure Queue manager %1
endmqm -p %1
for %%B in (dlt crt str) do %%Bmqm %1
goto:eof

:configureQM
if %1==CL1A set p=1420
if %1==CL1B set p=1421
if %1==QMA  set p=1422
if %1==QMB  set p=1423
echo configure %1 on port %p% connected to repository %2 on port %3 as %4 repository
echo DEFINE LISTENER(LST%1) TRPTYPE(TCP) PORT(%p%) CONTROL(QMGR) REPLACE | runmqsc %1
echo START LISTENER(LST%1) | runmqsc %1
if full==%4 echo ALTER  QMGR REPOS(CL1) DEADQ(SYSTEM.DEAD.LETTER.QUEUE) | runmqsc %1
echo DEFINE CHANNEL(TO.%2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('LOCALHOST(%3)') CLUSTER(CL1)
REPLACE | runmqsc %1
echo DEFINE CHANNEL(TO.%1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('LOCALHOST(%p%)')
CLUSTER(CL1) REPLACE | runmqsc %1
goto:eof
```

图 *15:* 创建队列管理器*: qmgrs.bat*

通过将预订添加到集群主题来更新配置。

```
@echo off
for %%A in (QMA QMB) do runmqsc %%A < wildsubs.tst
for %%A in (QMA QMB) do runmqsc %%A < upsubs.tst
```

图 *16:* 更新预订*: upsubs.bat*

运行以队列管理器作为参数的 pub.bat，以发布包含发布主题字符串的消息。 Pub.bat 使用样本程序 **amqspub**。

```
@echo off
@rem Provide queue manager name as a parameter
set S=Sports
set S=6 Sports/Football Sports/Football/Arsenal
set S=6 Sports/Rugby Sports/Rugby/Leeds
for %%B in (6) do echo %%B | amqspub %%B %1
```

图 *17:* 发布*: pub.bat*

流和主题

已排队的发布/预订具有集成发布/预订模型中不存在的发布流的概念。 在已排队的发布/预订中，流提供了一种分隔不同主题的信息流的方法。 在 IBM WebSphere MQ Version 6.0 中，流实现为队列，在支持流的每个代理上定义。 每个队列具有相同的名称（流的名称）。 从 IBM WebSphere MQ Version 7.0 开始，流作为顶级主题实现，可通过管理方式映射到其他主题标识。

将为网络上的所有代理和队列管理器自动设置缺省流 SYSTEM.BROKER.DEFAULT.STREAM，并且无需其他配置即可使用缺省流。 将缺省流视为未命名的缺省主题空间。 发布到缺省流的主题立即可供所有已连接的 Version 6.0 代理以及从 Version 7.0 开始的所有队列管理器使用，并且已启用排队的发布/预订。 指定的流类似于单独的，指定的主题空间。 必须在使用指定流的每个代理上定义该流。

定义主题时，该主题可供 Version 6.0 发布/预订代理以及在更高版本的 IBM WebSphere MQ 上运行的发布者和订户使用，并且没有特殊配置。

如果发布者和订户位于不同的队列管理器上，那么在同一代理层次结构中连接代理之后，不需要进一步配置这些发布以及预订之间的流。 相同的互操作性也会逆向工作。

指定的流

解决方案设计人员在使用排队的发布/预订编程模型时，可能会决定将所有体育出版物放入名为 Sport 的指定流中。 在 Version 6.0 中，流通常会自动复制到使用模型队列 SYSTEM.BROKER.MODEL.STREAM 的其他代理上。 但是，要使流可供在 Version 7.0 上运行且已启用排队发布/预订的队列管理器使用，必须手动添加该流。

如果从 Version 6.0 迁移队列管理器，请运行 **strmqbrk** 迁移 Version 6.0 指定流到主题的命令。 流 Sport 已映射到主题 Sport。 这不适用于 z/OS。

在流 Sport 上预订 Soccer/Results 的已排队发布/预订应用程序工作而不进行更改。 使用 MQSUB 预订主题 Sport 并提供主题字符串 Soccer/Results 的集成发布/预订应用程序也会接收相同的发布。

当 **strmqbrk** 创建主题 Soccer/Result 时，它将定义为主题 Sport 的子代，主题字符串为 Sport。 对 Soccer/Results 的预订是作为对 Sport/Soccer/Results 的预订实现的，因此 Sport 流的发布将映射到主题空间中的不同位置，以发布到其他流 (例如 Business)。

对于某些场景， **strmqbrk** 执行的自动迁移不是答案，您需要手动添加流。 添加流主题中描述了添加流的任务。 由于以下三个原因，您可能需要手动添加流。

1. 您继续在 V 6 队列管理器上维护发布/预订应用程序，这将与在后续队列管理器上运行的新编写的发布/预订应用程序进行互操作。
2. 继续开发在更高版本队列管理器上运行的已排队发布/预订应用程序，而不是将这些应用程序迁移到集成发布/预订 MQI 接口。
3. 流到主题的缺省映射会导致主题空间中的 "冲突" ，并且流上的发布与来自其他位置的发布具有相同的主题字符串。

权限

缺省情况下，在主题树的根目录中有多个主题对象: SYSTEM.BASE.TOPIC， SYSTEM.BROKER.DEFAULT.STREAM 和 SYSTEM.BROKER.DEFAULT.SUBPOINT 。 权限 (例如，用于发布或预订) 由 SYSTEM.BASE.TOPIC 上的权限确定; 将忽略 SYSTEM.BROKER.DEFAULT.STREAM 或 SYSTEM.BROKER.DEFAULT.SUBPOINT 上的任何权限。 如果删除 SYSTEM.BROKER.DEFAULT.STREAM 或 SYSTEM.BROKER.DEFAULT.SUBPOINT 中的任何一个并使用非空主题字符串重新创建，那么将以与普通主题对象相同的方式使用对这些对象定义的权限。

流与主题之间的映射

从 Version 7.0 开始，通过创建队列并为其提供与流相同的名称来模拟已排队的发布/预订流。 有时该队列被称为流队列，因为这就是它在排队的发布/预订应用程序中的显示方式。 通过将队列添加到名为 SYSTEM.QPUBSUB.QUEUE.NAMELIST 的特殊名称列表，可将该队列标识到发布/预订引擎。 通过向名称列表添加其他特殊队列，可以根据需要添加任意数量的流。 最后需要添加主题，名称与流相同，主题字符串与流名称相同，因此可以发布和预订主题。

但是，在特殊情况下，您可以为与流对应的主题提供您在定义主题时选择的任何主题字符串。 主题字符串的目的是在主题空间中为主题提供唯一的名称。 通常，流名称完全满足此目的。 有时，流名称与现有主题名称冲突。 要解决此问题，可以为与流关联的主题选择另一个主题字符串。 选择任何主题字符串，确保其唯一。

主题定义中定义的主题字符串以常规方式作为发布程序和订户使用 MQOPEN 或 MQSUB MQI 调用提供的主题字符串的前缀。 使用主题对象引用主题的应用程序不受前缀主题字符串选项的影响-因此，您可以选择任何在主题空间中保持出版物唯一的主题字符串。

将不同流重新映射到不同主题依赖于用于唯一主题字符串的前缀，以将一组主题与另一组主题完全分开。 您必须定义严格遵循的通用主题命名约定，才能使映射生效。 在 Version 7.0 中，如果主题字符串发生冲突，那么可以使用流来分隔主题空间。 从 Version 7.0 开始，使用前缀机制将主题字符串重新映射到主题空间中的其他位置。

注: 删除流时，请先删除该流上的所有预订。 如果任何预订源自代理层次结构中的其他代理，那么此操作最为重要。

示例

在 第 40 页的图 18 中，主题 'Sport' 具有主题字符串 'xyz'，导致源自流 'Sport' 的发布以版本 7 队列管理器主题空间中的字符串 'xyz' 作为前缀。 正在将 V 7 中的主题 'Sport' 前缀 'xyz' 发布或预订到主题字符串。 如果发布流向 V 6 订户，那么将从发布中除去前缀 'xyz'，并将其放置在 'Sport' 流中。 相反，当发布从 V 6 流到 V 7 时，从 'Sport' 流到 'Sport' 主题，前缀 'xyz' 将添加到主题字符串。

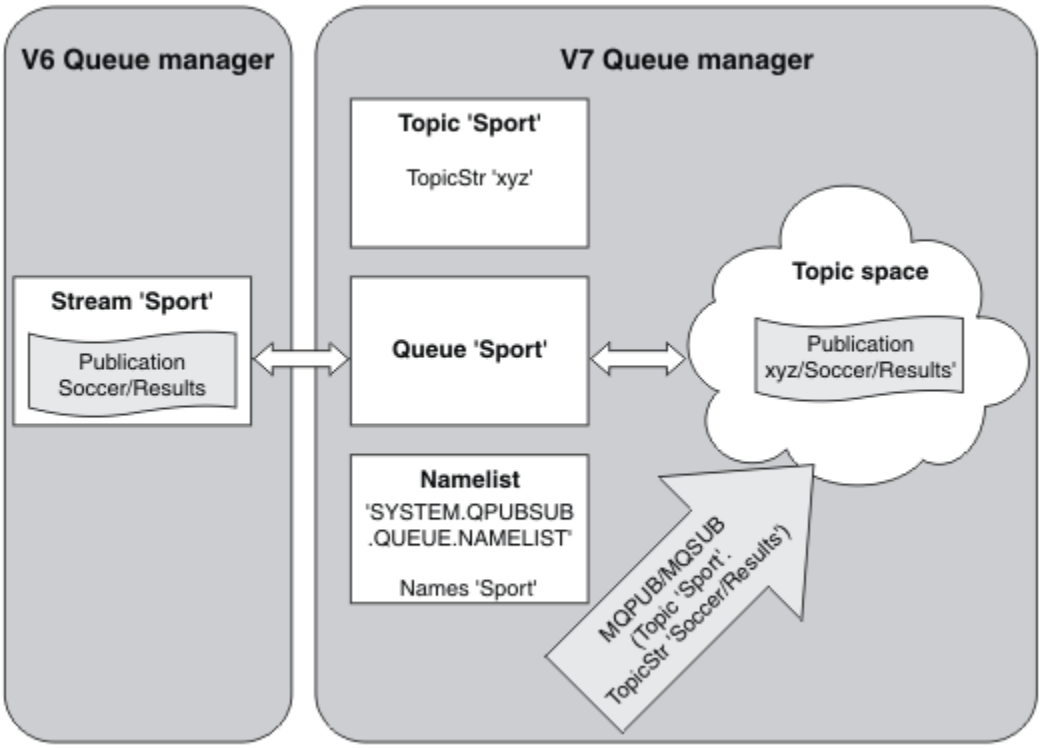


图 *18: 与 V 7* 主题共存的 *V 6* 流

预订点和主题

一个预订点，用于从 WebSphere MQ Event Broker 和 Message Broker 中的一组特定发布节点请求发布。 指定的预订点由主题和主题对象进行仿真。

WebSphere MQ Event Broker V6.0 到 WebSphere MQ V7.0.1 迁移过程 **migmbbrk** 将指定的预订点转换为主题和主题对象。 如果预订点具有保留发布或已注册订户，那么将自动迁移该预订点。 **migmbbrk** 从指定的预订点创建主题对象。 预订点的名称将成为主题对象的名称以及主题字符串本身。 主题对象将添加到 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST。

如果具有相同名称的主题对象存在，那么 **migmbbrk** 将执行下列其中一项操作。

1. 如果主题对象具有不同的主题字符串，或者如果预订点名称比对象名称长，那么 **migmbbrk** 将创建具有生成的名称的主题对象。
2. 如果主题对象具有相同的主题字符串，那么 **migmbbrk** 会将现有对象添加到名称列表。

要手动添加预订点，请参阅 添加预订点。

WebSphere MQ Event Broker 中的预订点

在 WebSphere MQ Event and Message Broker 消息流中使用发布节点来过滤消息并将其传输到订户。 发布者通常不会在发布节点上设置预订点。 订户注册对特定主题集的兴趣，并且通常也不指定预订点。

预订点是一种选择将消息转发到预订的发布节点的方法。 订户使用预订点的名称限定他们对一组主题的兴趣。

将名称分配给发布节点的 **Subscription point** 属性以设置其预订点名称。

预订点属性控制是否将主题的发布转发给同一主题的订户。 来自具有指定预订点的发布节点的发布仅转发给同一预订点的订户。 来自没有指定预订点 (缺省值) 的发布节点的发布仅转发给未指定预订点的订户。

具有指定预订点的节点以 MQRFH2 格式发送 Publish 命令消息，并设置 **SubPoint** 属性。 对指定预订点的预订必须在 MQRFH2 Register subscriber 命令消息中设置 **SubPoint** 属性。

WebSphere MQ 中的预订点

WebSphere MQ 将预订点映射到 WebSphere MQ 主题树中的不同主题空间。 没有预订点的命令消息中的主题将保持不变地映射到 WebSphere MQ 主题树的根，并从 SYSTEM.BASE.TOPIC 继承属性。

具有预订点的命令消息正在使用 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 中的主题对象列表进行处理。 命令消息中的预订点名称与列表中每个主题对象的主题字符串相匹配。 如果找到匹配项，那么会将预订点名称作为主题节点附加到主题字符串。 主题从 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 中找到的关联主题对象继承其属性。

使用预订点的效果是为每个预订点创建单独的主题空间。 主题空间植根于与预订点同名的主题中。 每个主题空间中的主题从与预订点同名的主题对象继承其属性。

将以正常方式从 SYSTEM.BASE.TOPIC 继承匹配主题对象中未设置的任何属性。

现有已排队的发布/预订应用程序使用 MQRFH2 消息头，通过在 Publish 或 Register subscriber 命令消息中设置 **SubPoint** 属性来继续工作。 预订点与命令消息中的主题字符串组合在一起，并与任何其他主题一样处理生成的主题。

新的 WebSphere MQ V7 应用程序不受预订点影响。 如果它使用从其中一个匹配主题对象继承的主题，那么它将使用匹配的预订点与排队的应用程序进行互操作。

示例

集合体中的现有 WebSphere MQ Event Broker 发布/预订应用程序使用预订点以不同货币发布股价。 IBM 股票的美元现货价格是使用预订点 USD 和主题 NYSE/IBM/SPOT 发布的。 使用同一主题和预订点 GBP 发布英镑价格。

WebSphere MQ 上的迁移过程将创建两个主题对象 GBP 和 USD 以及相应的主题字符串 'GBP' 和 'USD'。

现有发布者到主题 NYSE/IBM/SPOT，已迁移到 WebSphere MQ 上运行，这些发布者使用预订点 USD 在主题上创建发布 USD/NYSE/IBM/SPOT。 与 NYSE/IBM/SPOT 的现有订户类似，使用预订点 USD 创建对 USD/NYSE/IBM/SPOT 的预订。

通过调用 MQSUB 来预订版本 7 发布/预订程序中的美元现货价格。 使用 USD 主题对象和主题字符串 'NYSE/IBM/SPOT' 创建预订，如 "C" 代码片段中所示。

```
strncpy(sd.ObjectName, "USD", MQ_TOPIC_NAME_LENGTH);
sd.ObjectString.VSPtr = "NYSE/IBM/SPOT";
sd.ObjectString.VSLength = MQVS_NULL_TERMINATED;
MQSUB(Hconn, &sd, &Hobj, &Hsub, &CompCode, &Reason);
```

请考虑集合体中的 WebSphere MQ Event Broker 应用程序是否始终使用预订点 USD 和 GBP。 如果已创建，那么仅创建一次 USD 和 GBP 主题对象，作为集群主题主机上的集群主题。 您无需执行迁移过程的步骤 ../com.ibm.mq.mig.doc/q007670_.dita#q007670_/clusterstep ，即可将集群中每个队列管理器上的 SYSTEM.BASE.TOPIC 更改为集群主题。 而是执行以下步骤:

1. 在集群主题主机上设置 USD 和 GBP 主题对象的 CLUSTER 属性。
2. 删除集群中其他队列管理器上 USD 和 GBP 主题对象的所有副本。
3. 确保在集群中的每个队列管理器上的 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 中定义了 USD 和 GBP 。

分布式发布/预订

本部分包含有关如何在队列管理器与可用于连接队列管理器，集群和层次结构的两种不同队列管理器拓扑之间执行发布/预订消息传递的信息。

队列管理器可以与 WebSphere MQ 发布/预订系统中的其他队列管理器进行通信，以便订户可以预订一个队列管理器并接收最初发布到另一个队列管理器的消息。 第 42 页的图 19 对此作了说明。

第 42 页的图 19 显示了具有两个队列管理器的发布/预订系统。

- 队列管理器 2 由发布者 4 使用 "天气" 主题发布天气预报信息，并使用 "交通" 主题发布有关主要道路交通状况的信息。
- 订户 4 还使用此队列管理器，并使用主题 "流量" 预订有关流量条件的信息。
- 订户 3 还预订有关天气状况的信息，即使它使用与发布程序不同的队列管理器也是如此。 这是可能的，因为队列管理器相互链接。

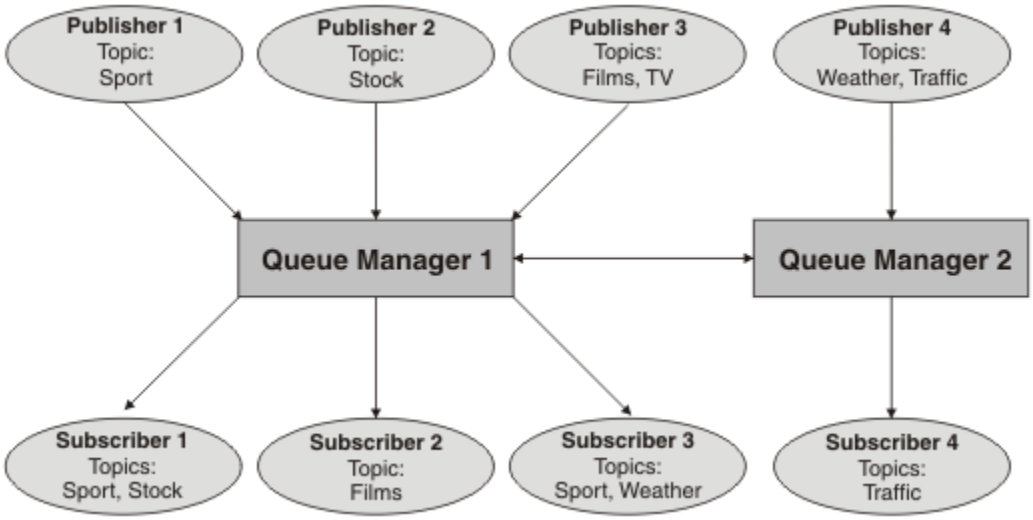


图 *19:* 具有两个队列管理器的发布/预订示例

分布式发布/预订如何工作?

WebSphere MQ 发布/预订使用代理预订来确保订户可以接收发布到远程队列管理器的消息。

分布式发布/预订使用与分布式排队相同的组件来连接队列管理器网络，从而连接到这些队列管理器的应用程序。 要了解有关队列管理器与在队列管理器之间建立连接所涉及的组件之间的消息传递的更多信息，请参阅 *Intercommunication* 文档。

订户无需在分布式发布/预订系统中执行标准预订操作以外的任何操作。 在队列管理器上进行预订时，队列管理器将管理将预订传播到已连接的队列管理器的过程。 代理预订流向网络中的所有队列管理器。 创建这些发布是为了确保将发布路由回创建原始预订的队列管理器; 请参阅 第 43 页的图 20。

仅当该远程队列管理器上存在对该主题的预订时，才会将发布传播到该远程队列管理器。

队列管理器将合并在其上创建的所有预订，无论是从本地应用程序还是从远程队列管理器。 它为具有其邻居的预订的主题创建代理预订，除非存在预订; 请参阅 第 43 页的图 21。

当应用程序发布信息时，接收队列管理器会将其转发到远程队列管理器上具有有效预订的任何应用程序。 它可能会通过一个或多个中间队列管理器将其转发; 请参阅 第 44 页的图 22。

订户 1 在亚洲队列管理器 (1) 上注册特定主题的预订。 此主题的预订将转发到网络中的所有其他队列管理器 (2,3, 4)。

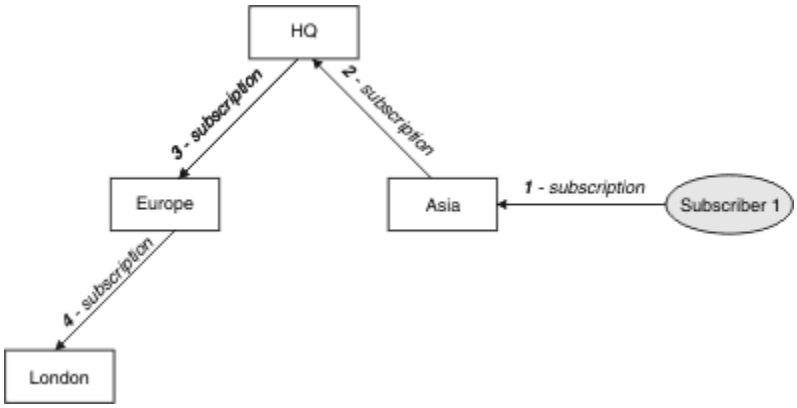


图 *20:* 通过队列管理器网络传播预订

订户 2 在总部队列管理器 (5) 上注册与 第 43 页的图 20 中相同的主题的预订。 此主题的预订将转发到亚洲队列管理器，以便它知道网络 (6) 上的其他位置存在预订。 未将预订转发到欧洲队列管理器，因为已注册此主题的预订; 请参阅 第 43 页的图 20 中的步骤 3。

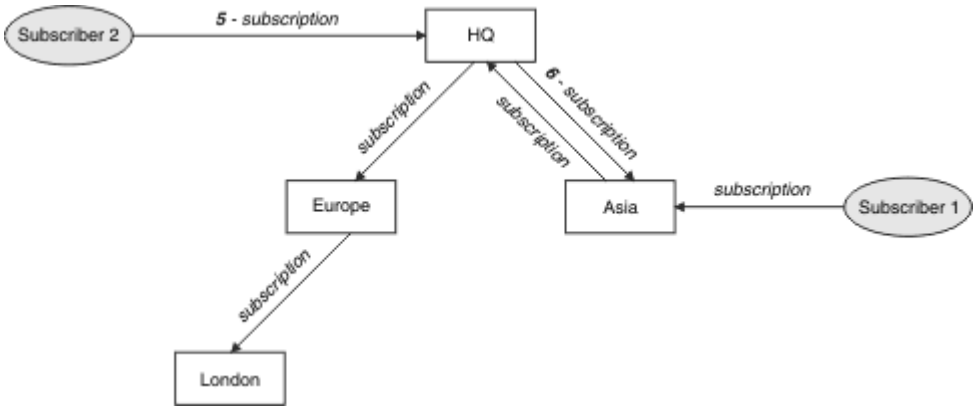


图 *21:* 多个预订

发布者将发布内容 (与 第 43 页的图 21 中的主题相同) 发送到欧洲队列管理器 (7)。 此主题存在从总部到欧洲的预订，因此该发布将转发到总部队列管理器 (8)。 但是，不存在从伦敦到欧洲 (仅从欧洲到伦敦) 的预订，因此不会将该发布转发到伦敦队列管理器。 HQ 队列管理器将发布直接发送到订户 2 和亚洲队列管理器 (9)。 该出版物将转发给来自亚洲的订户 1 (10)。

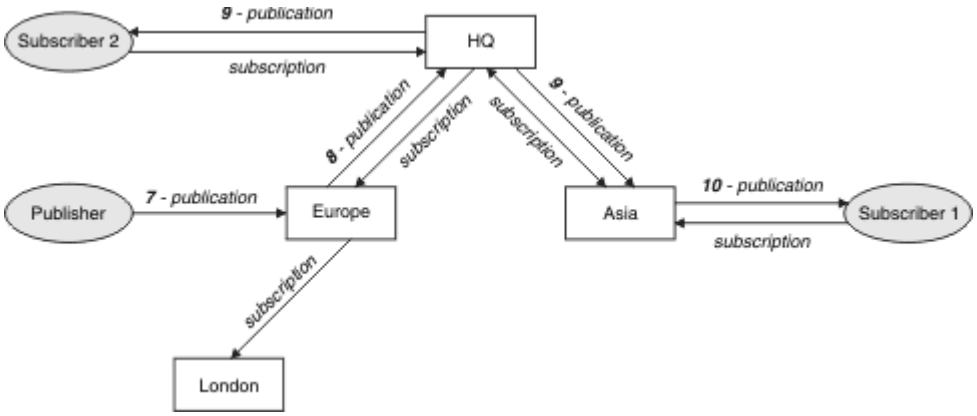


图 *22:* 通过队列管理器网络传播发布

当队列管理器将任何发布或预订发送到另一个队列管理器时，它会在消息中设置自己的用户标识。 如果您正在使用发布/预订层次结构，并且如果设置了入局通道以将具有用户标识权限的消息放入消息中，那么必须对发送队列管理器的用户标识进行授权; 请参阅 第 90 页的『将缺省用户标识与队列管理器层次结构配合使用』。 如果您正在使用发布/预订集群，那么授权由集群处理。

由于发布/预订队列管理器的互连性质，代理预订在网络中的所有节点之间传播需要时间。 远程发布不一定立即开始预订。 您可以通过将 **Topic** 属性 PROXYSUB 与值 FORCE 配合使用来消除预订延迟，如 第 45 页的『有关路由机制的更多信息』中所述。

当代理预订已放入每个直接连接的队列管理器的相应传输队列时，预订操作将完成。 预订操作不会等待代理预订传播到拓扑的其余部分。

代理预订与创建它们的队列管理器名称相关联。 如果层次结构中的队列管理器具有相同的名称，那么可能会导致发布无法访问这些队列管理器。 为了避免此问题 (与点到点消息传递一样)，请为队列管理器提供唯一名称，尤其是在它们直接或间接连接到 WebSphere MQ 网络时。

在分布式发布/预订网络中，可以使用发布和预订范围来控制发布和预订的流，并在适当情况下对其进行限制。

代理预订聚集和发布聚集

将聚集分布式发布/预订发布和代理预订，以最小化在发布/预订队列管理器之间传递的消息数量。

代理预订是一个队列管理器为另一个队列管理器上发布的主题所作的预订。 您不会显式创建代理预订，队列管理器代表您创建代理预订; 请参阅 第 42 页的『分布式发布/预订如何工作?』。

您可以将队列管理器一起连接到发布/预订层次结构中，或者连接到发布/预订集群中。 代理预订在已连接的队列管理器之间流动。 代理预订导致连接到一个队列管理器的发布程序所创建的主题的发布由连接到其他队列管理器的该主题的订户接收; 请参阅 第 52 页的『发布/预订拓扑』。

代理预订会针对预订所预订的每一个主题字符串在队列管理器间流动。

您可以使用 **Topic** 属性 PUBSCOPE 和 SUBSCOPE 来限制已连接队列管理器之间的代理预订和发布流。 您还可以通过将 **Topic** 属性 **通配符** 设置为 BLOCK 来限制包含通配符的代理预订流; 请参阅 第 46 页的『通配符规则』。

代理预订在队列管理器之间以异步方式流向创建预订。 通过将主题上的 **Topic** 属性 PROXYSUB 设置为 FORCE 或者将预订的主题的父代，可以减少等待代理预订传播到所有已连接队列管理器的等待时间; 请参阅 第 45 页的『有关路由机制的更多信息』。

代理预订聚集

代理预订是使用重复消除系统聚集的。 对于特定的已解析主题字符串，将在第一个本地预订或接收到的代理预订上发送代理预订。 对同一主题字符串的后续预订将使用此现有代理预订。

在取消最后一个本地预订或接收到的代理预订之后，将取消代理预订。

在具有针对各个主题字符串的数千个预订的发布/预订拓扑中，或者在这些预订的存在可能正在快速变化的情况下，必须考虑代理预订传播的开销。 可通过使用设置为 FORCE 的主题属性 **PROXYSUB** 来合并各个代理预订。 有关路由机制和集群主题性能的更多详细信息，请参阅 第 45 页的『有关路由机制的更多信息』。

发布聚集

当队列管理器上存在对同一主题字符串的多个预订时，仅从发布/预订拓扑中的其他队列管理器发送与该主题字符串匹配的每个发布的单个副本。 当消息到达时，本地队列管理器会将消息副本传递到每个匹配的预订。

当代理预订包含通配符时，可能有多个代理预订与单个发布的主题字符串匹配。 如果在与单个连接的队列管理器创建的两个或多个代理预订匹配的队列管理器上发布消息，那么仅会将该发布的一个副本转发到远程队列管理器以满足多个代理预订。

有关路由机制的更多信息

随时随地发布 是单个代理预订/转发的备用路由机制。 个别代理预订转发意味着只有在主题字符串上具有匹配预订的发布才会发送到远程消息传递服务器。 通过将发布到消息传递服务器的所有发布转发到分布式发布/预订网络中的所有其他消息传递服务器，可以随时随地发布或广播。 然后，接收消息传递服务器将交付与本地预订匹配的发布。

每个机制都有其优点，但也有局限性。

个别代理预订转发

此机制导致队列管理器间发布流量最小，因为仅发送与队列管理器上的预订匹配的那些发布。

但是:

- 预订的每个单独主题字符串都会导致将代理预订发送到发布/预订拓扑中的所有其他队列管理器。 如果要创建或删除数千个预订 (例如，重新启动队列管理器后的所有非持久预订)，或者如果预订集正在快速更改，并且每个预订都具有不同的主题字符串，那么此消息传递开销可能很大。
- 使用异步消息传递将代理预订流至其他队列管理器，因此，在创建预订与代理预订创建，传递和由其他队列管理器处理之间存在延迟。 在该时间间隔内在这些队列管理器上发布的消息不会传递到远程预订。

到处发布

使用此机制:

- 系统上没有每个主题字符串代理预订开销，这意味着快速预订创建，删除或更改不会导致网络负载和处理增加。
- 在创建预订和将发布流至队列管理器之间没有延迟，因为它们始终流至所有队列管理器。 因此，没有将发布传递到新创建的远程预订的窗口。

但是:

- 所有发布都将发送到发布/预订拓扑中的所有队列管理器，这可能导致发布在每个队列管理器上没有匹配预订的网络流量过大。

当您期望从集群或层次结构中的大量队列管理器预订发布时，或者由于预订更改的频率而导致代理预订开销过大时，您可能希望使用 "随时随地发布" 机制。 此工作方法在这些情况下可能比在将发布发送到所有队列管理器 (而不是具有匹配预订的队列管理器) 时迂到消息传递流量增加的其他情况下更有效。

通过将高级别主题对象的 **PROXYSUB** 属性设置为 FORCE ，可以在 IBM WebSphere MQ 分布式发布/预订拓扑中启用发布无处不在机制。

有关禁用个别代理预订的详细信息，请参阅 第 64 页的『禁用个别代理预订』。

在整个拓扑中传播此强制代理预订时，任何新预订都将立即从其他已连接的队列管理器接收任何发布，而不会迂到等待时间。

配置此类系统时必须小心。 **PROXYSUB** 设置为 FORCE 的主题下的任何主题对象都不得位于与 **PROXYSUB** 设置为 FORCE 的节点不同的集群或层次结构流中。 同样，下主题对象不得将其 **WILDCARD** 属性设置为 BLOCK。 在这两种情况下，这都可能导致已发布的消息无法从一个队列管理器正确流向另一个队列管理器。

即使 **PROXYSUB** 设置为 FORCE，也会继续传播预订的每个单独主题字符串的代理预订。 如果预订的数量和频率足以对系统造成重大开销，那么可以对队列管理器上的所有主题禁用这些预订。 有关禁用个别代理预订的详细信息，请参阅 第 64 页的『禁用个别代理预订』。

多点广播和预订等待时间

预订等待时间和 PROXYSUB (FORCE) 选项可用于维护代理预订。

例如，存在从 QM_B 到 QM_A 的代理预订在所有订户断开连接后被取消的潜在问题。 如果您要求多点广播流量继续，即使与队列管理器的单点广播连接终止，也可能不希望出现这种情况。 WebSphere MQ 的多点广播通过向每个代理预订添加几分钟的等待时间，在新订户进行连接时，会将该代理预订保持在短时间内，以便这些代理预订不会取消最后一个订户终止的即时时间。

您还可以对主题使用 PROXYSUB (FORCE) 选项，以确保未完成的代理预订始终处于未完成状态。 您必须确保在预订处于活动状态的大部分时间内，至少有一个订户需要流经队列的消息。 如果设置了 PROXYSUB (FORCE)，那么可能会在第一个本地预订或接收到的代理预订之前发送代理预订，并且即使在取消最后一个本地预订或接收到的代理预订之后也不会取消代理预订。

如果仍未预订，那么可以使用对等通信来确保消息传输继续进行; 有关更多信息，请参阅 多点广播的高可用性。

通配符规则
代理预订中的通配符将转换为使用主题通配符。

如果接收到通配符的预订，那么它可以是 WebSphere MQ Version 6.0 所使用的字符。 它也可以是 WebSphere Message Broker Version 6.0 和 WebSphere MQ Version 7.0 所使用的主题。

- 字符通配符使用 * 来表示任何字符，包括 /。
- 主题通配符使用 # 来表示 / 个字符之间的主题空间的一部分。

在 WebSphere MQ V 7.0 中，所有代理预订都将转换为使用主题通配符。 如果找到字符通配符，那么会将其替换为 # 字符，返回到最近的 /。 例如， /aaa/bbb/c*d 将转换为 /aaa/bbb/#。 此转换导致远程队列管理器发送的发布数略多于显式预订的发布数。 当本地队列管理器将发布内容传递给其本地订户时，会将其他发布内容过滤掉。

使用 通配符 属性控制通配符使用

使用 MQSC **Topic** 通配符 属性或等效 PCF 主题 `WildcardOperation` 属性来控制向使用通配符主题字符串名称的订户应用程序传递发布内容。 通配符 属性可以具有以下两个可能的值之一:

WILDCARD
与此主题有关的通配符预订的行为。

PASSTHRU
对没有此主题对象中主题字符串具体的通配主题进行的预订将接收对此主题以及比此主题更具体的主题字符串进行的发布。

BLOCK
对没有此主题对象中主题字符串具体的通配主题进行的预订不会接收对此主题或比此主题更具体的主题字符串进行的发布。

在定义预订时将使用此属性的值。 如果改变此属性，那么现有预订涵盖的主题集不会因为此修改而受到影响。 如果在创建或删除主题对象时拓扑发生更改，此场景也适用；将使用修改后的拓扑来创建与修改 WILDCARD 属性后创建的预订匹配的主题集。 如果要针对现有预订强制重新评估匹配的主题集，那么必须重新启动队列管理器。

在示例 第 36 页的『示例: 创建 Sport 发布/预订集群』中，您可以遵循步骤来创建 第 33 页的图 10 中显示的主题树结构。

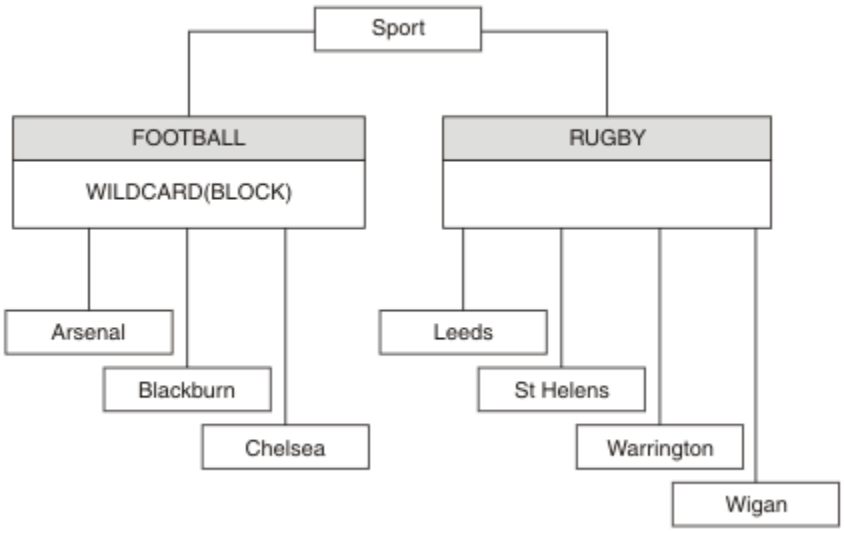


图 *23:* 使用 通配符 属性 *BLOCK* 的主题树

使用通配符主题字符串 *#* 的订户将接收到 Sport 主题和 Sport/Rugby 子树的所有发布。 订户不会接收到 Sport/Football 子树的发布，因为 Sport/Football 主题的 **通配符** 属性值为 BLOCK。

PASSTHRU 是缺省设置。 您可以将 **通配符** 属性值 PASSTHRU 设置为 Sport 树中的节点。 如果节点没有 **通配符** 属性值 BLOCK，那么设置 PASSTHRU 不会改变 Sports 树中节点的订户观察到的行为。

在此示例中，创建预订以查看通配符设置如何影响交付的发布; 请参阅 第 37 页的图 14。 在 第 38 页的图 17 中运行发布命令以创建一些发布。

```
pub QMA
```

图 *24:* 发布到 *QMA*

第 33 页的表 3 显示了结果。 请注意，如何设置 **通配符** 属性值 BLOCK，以防止具有通配符的预订将发布内容接收到通配符作用域内的主题。

表 *6: QMA* 上收到的出版物

预订	**主题字符串**	**收到的出版物**	**注**
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	WILDCARD(BLOCK) 在 Sports/Football 上阻止了对 Football 子树的所有发布
SARSENAL	Sports/#/Arsenal	-	Sports/Football 上的 WILDCARD(BLOCK) 会阻止 Arsenal 上的通配符预订
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby 上的缺省 WILDCARD 不会阻止 Leeds 上的通配符预订。

注:

假设预订具有通配符，该通配符与具有 **通配符** 属性值 BLOCK 的主题对象相匹配。 如果预订还具有匹配通配符右侧的主题字符串，那么预订将从不接收发布内容。 未被阻止的发布集合是作为被阻止通配符的父代的主题的发布。 通配符将阻止发布到作为具有 BLOCK 属性值的主题的子代的主题。 因此，包含通配符右侧主题的预订主题字符串永远不会收到任何要匹配的发布内容。

将 WILDCARD 属性值设置为 BLOCK 并不意味着您无法使用包含通配符的主题字符串进行预订。 这样的订阅是正常的。 预订具有一个显式主题，该主题与具有 **通配符** 属性值 BLOCK 的主题对象匹配。 它将通配符用

于作为具有 **通配符** 属性值 BLOCK 的主题的父代或子代的主题。 在 第 33 页的图 10 中的示例中，预订 (例如 Sports/Football/#) 可以接收发布内容。

通配符和集群主题

集群主题定义将传播到集群中的每个队列管理器。 在集群中的一个队列管理器上预订集群主题会导致队列管理器创建代理预订。 将在集群中的每个其他队列管理器上创建代理预订。 使用包含通配符的主题字符串 (与集群主题结合使用) 的预订可能难以预测行为。 此行为在以下示例中进行了说明。

在为示例 第 36 页的『示例: 创建 Sport 发布/预订集群』设置的集群中， QMB 具有与 QMA 相同的预订集，但 QMB 在发布者发布到 QMA 之后未收到任何发布，请参阅 第 33 页的图 11。 虽然 Sports/Football 和 Sports/Rugby 主题是集群主题，但 fullsubs.tst 中定义的预订不会引用集群主题。 不会将代理预订从 QMB 传播到 QMA。 如果没有代理预订，那么不会将 QMA 的任何发布转发到 QMB 。

某些预订 (例如 Sports/#/Leeds) 可能似乎引用了集群主题，在本例中为 Sports/Rugby 。 Sports/#/Leeds 预订实际解析为主题对象 SYSTEM.BASE.TOPIC。

用于解析预订 (例如， Sports/#/Leeds) 所引用的主题对象的规则如下所示。 将主题字符串截断为第一个通配符。 通过主题字符串向左扫描以查找具有关联管理主题对象的第一个主题。 主题对象可以指定集群名称，也可以定义本地主题对象。 在示例 Sports/#/Leeds 中，截断后的主题字符串是 Sports，它没有主题对象，因此 Sports/#/Leeds 继承自 SYSTEM.BASE.TOPIC(这是本地主题对象)。

要查看预订集群主题如何更改通配符传播的工作方式，请运行批处理脚本 upsubs.bat。 该脚本将清除预订队列，并在 fullsubs.tst 中添加集群主题预订。 再次运行 puba.bat 以创建一批发布; 请参阅 第 33 页的图 11。

第 34 页的表 4 显示了将两个新预订添加到发布发布的同一队列管理器的结果。 结果与预期相同，新预订各接收一个发布，其他预订接收的发布数量保持不变。 意外结果发生在其他集群队列管理器上; 请参阅 第 35 页的表 5。

表 7: *QMA* 上收到的出版物

预订	主题字符串	收到的出版物	注
SPORTS	Sports/#	Sports Sports/Rugby Sports/Rugby/Leeds	WILDCARD(BLOCK) 在 Sports/Football 上阻止了对 Football 子树的所有发布
SARSENAL	Sports/#/Arsenal	-	Sports/Football 上的 WILDCARD(BLOCK) 会阻止 Arsenal 上的通配符预订
SLEEDS	Sports/#/Leeds	Sports/Rugby/Leeds	Sports/Rugby 上的缺省 WILDCARD 不会阻止 Leeds 上的通配符预订。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal 接收发布，因为预订没有通配符。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds 将在任何情况下接收发布。

第 35 页的表 5 显示了在 QMB 上添加两个新预订以及在 QMA 上发布的结果。 请注意，在没有这两个新预订的情况下， QMB 未收到任何发布。 正如预期的那样，这两个新预订将接收发布内容，因为 Sports/FootBall 和 Sports/Rugby 都是集群主题。 QMB 将 Sports/Football/Arsenal 和 Sports/Rugby/Leeds 的代理预订转发到 QMA，然后将发布发送到 QMB。

意外的结果是，先前未收到任何发布的两个预订 Sports/# 和 Sports/#/Leeds 现在接收发布。 原因是针对其他预订转发到 QMB 的 Sports/Football/Arsenal 和 Sports/Rugby/Leeds 发布现在可用于连接到 QMB 的任何订户。 因此，本地主题 Sports/# 和 Sports/#/Leeds 的预订将接收 Sports/Rugby/Leeds 发布。 Sports/#/Arsenal 继续不接收发布内容，因为 Sports/Football 已将其 **通配符** 属性值设置为 BLOCK。

表 *8: QMB* 上收到的出版物			
预订	**主题字符串**	**收到的出版物**	**注**
SPORTS	*Sports/#*	*Sports/Rugby/Leeds*	*WILDCARD(BLOCK)* 在 *Sports/Football* 上阻止了对 *Football* 子树的所有发布
SARSENAL	Sports/#/Arsenal	-	Sports/Football 上的 WILDCARD(BLOCK) 会阻止 Arsenal 上的通配符预订
SLEEDS	*Sports/#/Leeds*	*Sports/Rugby/Leeds*	*Sports/Rugby* 上的缺省 *WILDCARD* 不会阻止 *Leeds* 上的通配符预订。
FARSENAL	Sports/Football/Arsenal	Sports/Football/Arsenal	Arsenal 接收发布，因为预订没有通配符。
FLEEDS	Sports/Rugby/Leeds	Sports/Rugby/Leeds	Leeds 将在任何情况下接收发布。

在大多数应用程序中，不希望一个预订影响另一个预订的行为。 **通配符** 属性与值 BLOCK 的一个重要用途是使包含通配符的同一主题字符串的预订行为一致。 无论预订是在与发布程序相同的队列管理器上，还是在不同的队列管理器上，预订的结果都是相同的。

通配符和流

WebSphere MQ V 6 流由 WebSphere MQ V 7 映射到主题; 请参阅 第 38 页的『流和主题』 。 在 **strmqbrk** 在 V 7 中执行的缺省映射中，流 Sports 中的所有主题都将映射到主题 Sports。 流 Business 中的所有主题都将映射到主题 Business。

Sports 流中的 WebSphere MQ V 6 到 * 的预订将接收 Sports 树中的所有发布内容，而 Business 树中没有发布内容。 版本 7 中的同一预订将接收 Sports 树中的所有发布以及 Business 树中的所有发布。 要阻止此行为，当流迁移到 V 7 时， **strmqbrk** 会设置 **通配符** 属性。 对于从流迁移的每个顶级主题，它会将其设置为值 BLOCK 。 通过从名为 Sports 和 Business 的版本 6 流进行转换，将 Sports 和 Business 的 **通配符** 属性设置为值 BLOCK 。

对于写入发布/预订 API 的新应用程序，结果是对 * 的预订不会收到任何发布。 要接收所有体育出版物，必须预订 Sports/*或 Sports/# 以及类似的 Business 出版物。

将发布/预订代理迁移到 WebSphere MQ V 7 时，现有已排队的发布/预订应用程序的行为不会更改。 **Publish**， **Register Publisher** 或 **Subscriber** 命令中的 **StreamName** 属性将映射到流已迁移到的主题的名称。

通配符和预订点

WebSphere Message Broker 预订点由 WebSphere MQ V 7 映射到主题; 请参阅 第 40 页的『预订点和主题』 。 在 **migmqbrk** 在 V 7 中执行的缺省映射中，预订点 Sports 中的所有主题都将映射到主题 Sports。 预订点 Business 中的所有主题都将映射到主题 Business。

WebSphere Message Broker V 6 到 Sports 预订点中 * 的预订将接收 Sports 树中的所有发布内容，而 Business 树中没有发布内容。 版本 7 中的同一预订将接收 Sports 树中的所有发布以及 Business 树中的所有发布。 要阻止此行为，当预订点迁移到 V 7 时， **migmqbrk** 会设置 **通配符** 属性。 对于从预订点迁移的每个顶级主题，它会将其设置为值 BLOCK 。 通过从名为 Sports 和 Business 的 WebSphere Message Broker 预订点进行转换，将 Sports 和 Business 的 **通配符** 属性设置为值 BLOCK 。

对于写入发布/预订 API 的新应用程序，迁移的效果是，对 * 的预订不会收到任何发布。 要接收所有体育出版物，必须预订 Sports/*或 Sports/# 以及类似的 Business 出版物。

将发布/预订代理迁移到 WebSphere MQ V 7 时，现有已排队的发布/预订应用程序的行为不会更改。 **Publish**， **Register Publisher** 或 **Subscriber** 命令中的 **SubPoint** 属性将映射到预订已迁移到的主题的名称。

示例: 创建 Sport 发布/预订集群

后续步骤将创建一个具有四个队列管理器的集群 CL1: 两个完整存储库 CL1A 和 CL1B 以及两个部分存储库 QMA 和 QMB。 完整存储库仅用于保存集群定义。 QMA 被指定为集群主题主机。 持久预订同时在 QMA 和 QMB 上定义。

注: 此示例针对 Windows 进行编码。 您必须重新编码 Create qmgrs.bat 和 create pub.bat 以在其他平台上配置和测试示例。

1. 创建脚本文件。
   a. 创建 topics.tst
   b. 创建 wildsubs.tst
   c. 创建 fullsubs.tst
   d. 创建 qmgrs.bat
   e. 创建 pub.bat
2. 运行 Create qmgrs.bat 以创建配置。

```
qmgrs
```

在 第 33 页的图 10 中创建主题。 图 5 中的脚本将创建集群主题 Sports/Football 和 Sports/Rugby。

注: REPLACE 选项不会替换主题的 TOPICSTR 属性。 TOPICSTR 是在示例中进行有效更改以测试不同主题树的属性。 要更改主题，请先删除该主题。

```
DELETE TOPIC ('Sports')
DELETE TOPIC ('Football')
DELETE TOPIC ('Arsenal')
DELETE TOPIC ('Blackburn')
DELETE TOPIC ('Chelsea')
DELETE TOPIC ('Rugby')
DELETE TOPIC ('Leeds')
DELETE TOPIC ('Wigan')
DELETE TOPIC ('Warrington')
DELETE TOPIC ('St. Helens')

DEFINE TOPIC ('Sports')     TOPICSTR('Sports')
DEFINE TOPIC ('Football')   TOPICSTR('Sports/Football') CLUSTER(CL1) WILDCARD(BLOCK)
DEFINE TOPIC ('Arsenal')    TOPICSTR('Sports/Football/Arsenal')
DEFINE TOPIC ('Blackburn')  TOPICSTR('Sports/Football/Blackburn')
DEFINE TOPIC ('Chelsea')    TOPICSTR('Sports/Football/Chelsea')
DEFINE TOPIC ('Rugby')      TOPICSTR('Sports/Rugby')    CLUSTER(CL1)
DEFINE TOPIC ('Leeds')      TOPICSTR('Sports/Rugby/Leeds')
DEFINE TOPIC ('Wigan')      TOPICSTR('Sports/Rugby/Wigan')
DEFINE TOPIC ('Warrington') TOPICSTR('Sports/Rugby/Warrington')
DEFINE TOPIC ('St. Helens') TOPICSTR('Sports/Rugby/St. Helens')
```

图 *25:* 删除并创建主题*: topics.tst*

注: 删除主题，因为 REPLACE 不会替换主题字符串。

使用通配符创建预订。 通配符将主题与 第 33 页的图 10 中的主题对象对应。 为每个预订创建一个队列。 运行或重新运行脚本时，将清除队列并删除预订。

注: REPLACE 选项不会替换预订的 TOPICOBJ 或 TOPICSTR 属性。 TOPICOBJ 或 TOPICSTR 是在用于测试不同预订的示例中进行有效更改的属性。 要对其进行更改，请先删除预订。

```
DEFINE QLOCAL(QSPORTS)   REPLACE
DEFINE QLOCAL(QSARSENAL) REPLACE
DEFINE QLOCAL(QSLEEDS)   REPLACE
CLEAR  QLOCAL(QSPORTS)
CLEAR  QLOCAL(QSARSENAL)
CLEAR  QLOCAL(QSLEEDS)

DELETE SUB   (SPORTS)
DELETE SUB   (SARSENAL)
DELETE SUB   (SLEEDS)
DEFINE SUB   (SPORTS)   TOPICSTR('Sports/#')         DEST(QSPORTS)
DEFINE SUB   (SARSENAL) TOPICSTR('Sports/+/Arsenal') DEST(QSARSENAL)
DEFINE SUB   (SLEEDS)   TOPICSTR('Sports/+/Leeds')   DEST(QSLEEDS)
```

图 *26:* 创建通配符预订*: wildsubs.tst*

创建引用集群主题对象的预订。

注:

将在 TOPICOBJ 引用的主题字符串与 TOPICSTR 定义的主题字符串之间自动插入定界符 /。

定义 DEFINE SUB(FARSENAL) TOPICSTR('Sports/Football/Arsenal') DEST(QFARSENAL) 将创建相同的预订。 TOPICOBJ 用作引用您已定义的主题字符串的快速方法。 创建预订时，不再引用主题对象。

```
DEFINE QLOCAL(QFARSENAL) REPLACE
DEFINE QLOCAL(QRLEEDS)   REPLACE
CLEAR  QLOCAL(QFARSENAL)
CLEAR  QLOCAL(QRLEEDS)

DELETE SUB   (FARSENAL)
DELETE SUB   (RLEEDS)
DEFINE SUB   (FARSENAL) TOPICOBJ('Football') TOPICSTR('Arsenal') DEST(QFARSENAL)
DEFINE SUB   (RLEEDS)   TOPICOBJ('Rugby')    TOPICSTR('Leeds')   DEST(QRLEEDS)
```

图 *27:* 删除并创建预订*: fullsubs.tst*

创建具有两个存储库的集群。 创建两个用于发布和预订的部分存储库。 重新运行脚本以删除所有内容，然后再次启动。 该脚本还会创建主题层次结构和初始通配符预订。

注:

在其他平台上，编写类似的脚本，或者输入所有命令。 通过使用脚本，可以快速删除所有内容，并使用相同的配置重新开始。

```
@echo off
set port.CL1B=1421
set port.CLIA=1420
for %%A in (CL1A CL1B QMA QMB) do call :createQM %%A
call :configureQM CL1A CL1B %port.CL1B% full
call :configureQM CL1B CL1A %port.CL1A% full
for %%A in (QMA QMB) do call :configureQM %%A CL1A %port.CL1A% partial
for %%A in (topics.tst wildsubs.tst) do runmqsc QMA < %%A
for %%A in (wildsubs.tst) do runmqsc QMB < %%A
goto:eof

:createQM
echo Configure Queue manager %1
endmqm -p %1
for %%B in (dlt crt str) do %%Bmqm %1
goto:eof

:configureQM
if %1==CL1A set p=1420
if %1==CL1B set p=1421
if %1==QMA  set p=1422
if %1==QMB  set p=1423
echo configure %1 on port %p% connected to repository %2 on port %3 as %4 repository
echo DEFINE LISTENER(LST%1) TRPTYPE(TCP) PORT(%p%) CONTROL(QMGR) REPLACE | runmqsc %1
echo START LISTENER(LST%1) | runmqsc %1
if full==%4 echo ALTER  QMGR REPOS(CL1) DEADQ(SYSTEM.DEAD.LETTER.QUEUE) | runmqsc %1
echo DEFINE CHANNEL(TO.%2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('LOCALHOST(%3)') CLUSTER(CL1)
REPLACE | runmqsc %1
echo DEFINE CHANNEL(TO.%1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('LOCALHOST(%p%)')
CLUSTER(CL1) REPLACE | runmqsc %1
goto:eof
```

图 *28:* 创建队列管理器*: qmgrs.bat*

通过将预订添加到集群主题来更新配置。

```
@echo off
for %%A in (QMA QMB) do runmqsc %%A < wildsubs.tst
for %%A in (QMA QMB) do runmqsc %%A < upsubs.tst
```

图 *29:* 更新预订*: upsubs.bat*

运行以队列管理器作为参数的 pub.bat，以发布包含发布主题字符串的消息。 Pub.bat 使用样本程序 **amqspub**。

```
@echo off
@rem Provide queue manager name as a parameter
set S=Sports
set S=6 Sports/Football Sports/Football/Arsenal
set S=6 Sports/Rugby Sports/Rugby/Leeds
for %%B in (6) do echo %%B | amqspub %%B %1
```

图 *30:* 发布*: pub.bat*

发布/预订拓扑

发布/预订拓扑 由支持发布/预订应用程序的队列管理器及其之间的连接组成。

发布/预订应用程序可以由连接在一起的队列管理器网络组成。 队列管理器可以全部位于同一物理系统上，也可以分布在多个物理系统上。 通过将队列管理器连接在一起，应用程序可以使用网络中的任何队列管理器来接收发布。

这提供了以下优点:

- 客户机应用程序可以与附近的队列管理器通信，而不是与遥远的队列管理器通信，从而获得更好的响应时间。
- 通过使用多个队列管理器，可以支持更多订户。

您可以通过两种不同的方式 (集群和层次结构) 安排正在执行发布/预订消息传递的队列管理器。 有关简单集群和简单层次结构的示例，请参阅 第 53 页的图 31 和 第 53 页的图 32。 有关这两种拓扑的更多信息以及要了解最适合您的拓扑，请参阅产品文档的本部分中的信息。

通过在层次结构中将集群连接在一起，可以组合使用这两种拓扑。

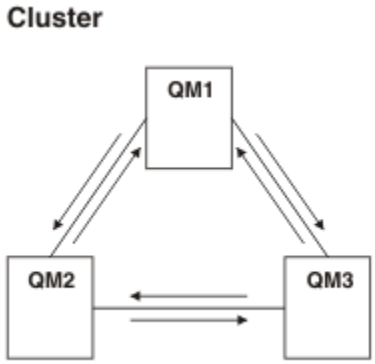


图 *31:* 简单发布/预订集群

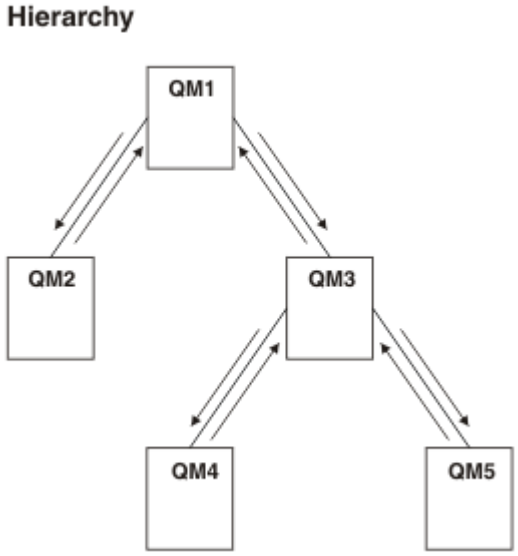
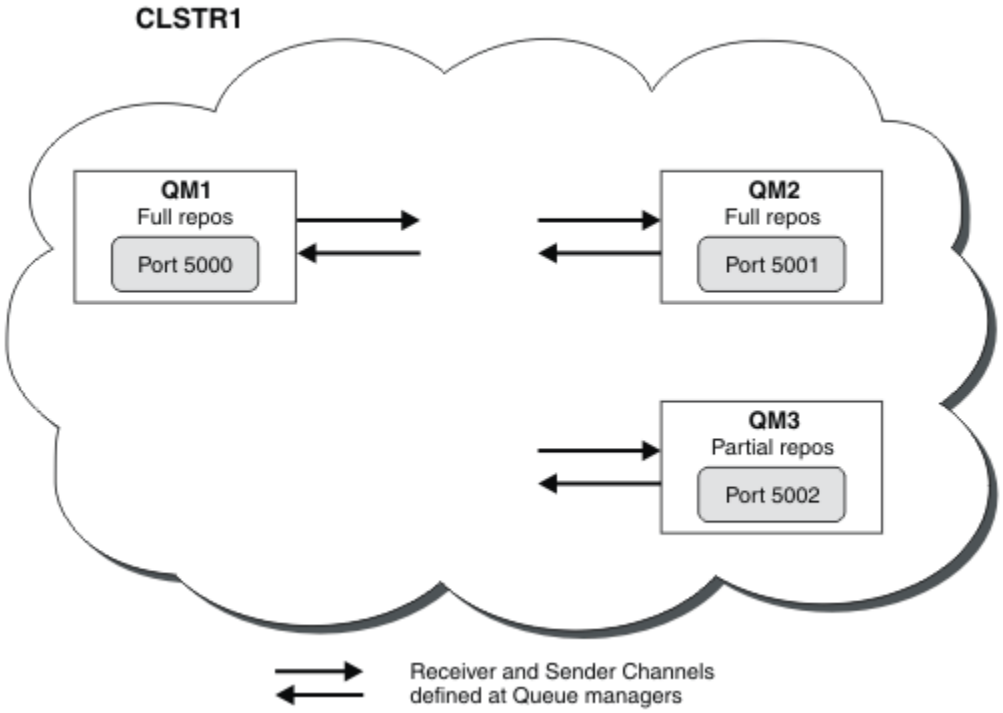


图 *32:* 简单发布/预订层次结构

设置发布/预订集群*:* 方案 *1*
将两个队列管理器作为完整存储库添加到集群，并定义它们之间的通道。

关于此任务

下图包含三个队列管理器: QM1 ， QM2 和 QM3:



QM1 和 QM2 是集群中的完整存储库，而 QM3 是部分存储库。

方案 1 将 QM1 和 QM2 作为完整存储库添加到集群 DEMO 。

方案 2 将 QM3 作为部分存储库添加到集群 DEMO 。

这些任务至少需要一个命令窗口。

过程

1. 将 QM1 和 QM2 设置为 DEMO 集群的完整存储库:

```
alter QMGR REPOS(DEMO)
```

2. 定义并启动 QM1 的侦听器:

```
define listener(QM1_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5000)
start listener(QM1_LS)
```

3. 定义并启动 QM2 的侦听器:

```
define listener(QM2_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5001)
start listener(QM2_LS)
```

4. 为 QM1 定义接收方通道:

```
DEFINE CHANNEL(DEMO.QM1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')
CLUSTER(DEMO) DESCR('TCP Cluster-receiver channel for queue manager QM1')
```

5. 定义从 QM1 到 QM2 的发送方通道:

```
DEFINE CHANNEL(DEMO.QM2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5001)')
CLUSTER(DEMO) DESCR('TCP Cluster-sender channel from QM1 to  queue manager QM2')
```

6. 为 QM2 定义接收方通道:

```
DEFINE CHANNEL(DEMO.QM2) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5001)')
CLUSTER(DEMO) DESCR('TCP Cluster-receiver channel for queue manager QM2')
```

7. 定义从 QM2 到 QM1 的发送方通道:

```
DEFINE CHANNEL(DEMO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')
CLUSTER(DEMO) DESCR('TCP Cluster-sender channel from qm2 to qm1')
```

8. 在 QM1 上定义集群主题 scores :

```
define topic(scores) TOPICSTR(/football) CLUSTER(DEMO)
```

9. 使用以下命令验证设置:

```
display topic(scores) type(all) clusinfo
display clusqmgr(*)
display chstatus(*)
```

10. 使用两个命令窗口测试设置:
    a. 在第一个命令窗口中输入以下命令:

```
/opt/mqm/samp/bin/amqspub /FOOTBALL/scores QM1
```

    b. 在第二个命令窗口中输入以下命令:

```
/opt/mqm/samp/bin/amqssub /FOOTBALL/scores QM2
```

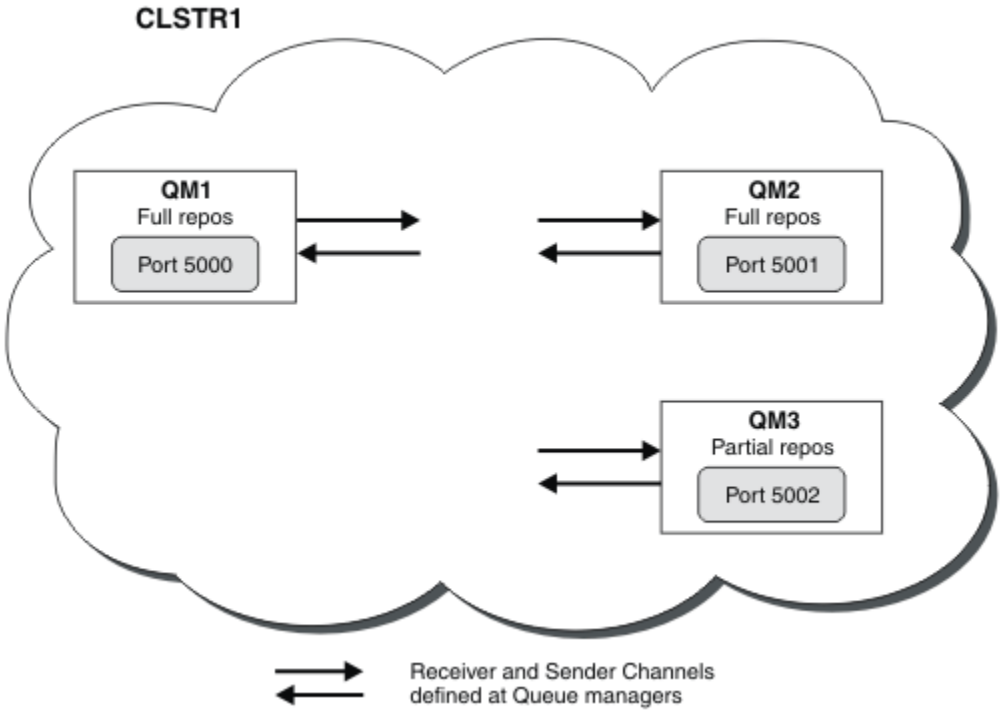
相关任务
管理 WebSphere MQ 集群
设置新集群

设置发布/预订集群*: 方案 2*
将第三个队列管理器作为部分存储库添加到集群。

开始之前
必须先在 第 53 页的『设置发布/预订集群: 方案 1』 中完成该任务，然后才能完成此任务。

关于此任务

下图具有 3 个队列管理器; QM1， QM2 和 QM3:



QM1 和 QM2 是集群中的完整存储库，而 QM3 是部分存储库。

方案 1 将 QM1 和 QM2 作为完整存储库添加到集群 DEMO 。

方案 2 将 QM3 作为部分存储库添加到集群 DEMO 。

这些任务至少需要 1 命令窗口。

过程

1. 定义并启动 QM3 的侦听器:

```
define listener(QM3_LS) TRPTYPE(TCP) CONTROL(QMGR) PORT(5002)
start listener(QM3_LS)
```

2. 为 QM3 定义接收方通道:

```
DEFINE CHANNEL(DEMO.QM3) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5002)') CLUSTER
(DEMO) DESCR('TCP Cluster-receiver channel for queue manager QM3')
```

3. 定义从 QM3 到 QM1 的发送方通道:

```
DEFINE CHANNEL(DEMO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('$HOSTNAME(5000)')
CLUSTER(DEMO) DESCR('TCP Cluster-sender channel from qm3 to qm1')
```

4. 使用以下命令验证设置:

```
display topic(scores) type(all) clusinfo
display clusqmgr(*)
display chstatus(*)
```

5. 使用 2 命令窗口测试设置:
   a. 在第一个命令窗口中输入以下命令:

```
/opt/mqm/samp/bin/amqspub /FOOTBALL/scores QM2
```

   b. 在第二个命令窗口中输入以下命令:

```
/opt/mqm/samp/bin/amqssub /FOOTBALL/scores QM3
```

发布/预订集群

发布/预订集群是互连队列管理器的标准 IBM WebSphere MQ 集群，其中发布将自动从发布应用程序移至集群中任何队列管理器上存在的预订。

用于发布/预订消息传递的集群与标准 IBM WebSphere MQ 集群没有任何不同。 因此，发布/预订集群中的队列管理器可以存在于物理上独立的计算机上，必要时，集群通道会自动将每对队列管理器连接在一起。 有关如何规划和配置 IBM WebSphere MQ 集群的信息，请参阅 集群如何工作。

定义集群主题对象时，将通过在集群中的任何队列管理器配置的主题上设置 **CLUSTER** 属性来创建发布/预订集群。 主题定义将传播到集群的所有成员。 您可以在集群中的任何队列管理器上发布和预订主题，以及在主题树中该主题下方的任何主题字符串。 发布将自动传播到连接到集群中其他队列管理器的订户。

通过使用不属于集群主题对象的主题字符串，还可以在发布/预订集群中执行非集群发布/预订活动。 此安排与发布/预订层次结构不同，在该层次结构中，所有预订都将在整个层次结构中传播。 在这两种情况下，都可以使用 "预订" 和 "发布" 作用域进行更精细的控制。

在发布/预订拓扑中使用集群具有以下优点:

- 发往同一集群中特定队列管理器上的预订的消息将直接传输到该队列管理器，并且不需要通过中间队列管理器。 与分层拓扑相比，这将提高性能并优化队列管理器之间的发布/预订流量。
- 由于所有队列管理器都直接相互连接，因此此拓扑中没有单点故障。 如果一个队列管理器不可用，那么集群中其他队列管理器上的预订仍能够从可用队列管理器上的发布程序接收消息。
- 在包含多个独立集群的系统中 (例如，集群在地理上分散) ，可以将集群连接到集群的层次结构中。 此连接是通过加入每个集群中的单个队列管理器来创建的，以启用通过网络的发布和预订流; 请参阅 第 82 页的『组合多个集群的主题空间』。 您还可以控制哪些发布从一个集群流向另一个集群; 请参阅 第 83 页的『组合和隔离多个集群中的主题空间』。
- 预订应用程序可以连接到其最近的队列管理器，以提高其自身的性能。 队列管理器从集群中的所有队列管理器接收与客户机的预订注册相匹配的所有消息。

  对于从此队列管理器请求的其他服务，还会提高客户机应用程序的性能。 客户机应用程序可以同时使用发布/预订和点到点消息传递。
- 通过向集群添加更多队列管理器以共享工作负载，可以减少每个队列管理器的客户机和预订数。 发布会自动分发到新队列管理器上的客户机。 对于某些使用模式，此过程可以使发布/预订集群拓扑具有高度可伸缩性。

在发布/预订中使用集群时要考虑的事项:

- 将自动使发布/预订集群中的所有队列管理器了解集群中的所有其他队列管理器。 对于点到点集群，此过程有所不同，只有队列管理器所关注的队列管理器是已知的。
- 发布/预订集群中的队列管理器托管对集群主题的一个或多个预订，自动创建集群中所有其他队列管理器的集群发送方通道。 即使接收队列管理器未在任何集群主题上发布消息，队列管理器也会向每个队列管理器发送有关预订的信息。
- 队列管理器上对集群主题下的主题字符串的第一个预订会导致将消息发送到集群中的每个其他队列管理器。 同样，要删除的主题字符串上的最后一个预订也会生成一条消息。 在集群主题下使用的单个主题字符串越多，发生的队列管理器间通信就越多。

警告:

由于本主题中先前列出的原因，将集群主题引入大型 IBM WebSphere MQ 集群 (即，包含许多队列管理器的集群) 会立即导致集群中每个队列管理器上的额外负载，并且在某些情况下会导致性能降低。 有关更多信息，请参阅第 62 页的『集群主题性能』。

必须仔细规划将发布/预订引入队列管理器集群，特别是现有集群，以适应这些性能下降的情况。

如果已知集群无法降低发布/预订的性能，那么可以使用 **PSCLUS** 参数在队列管理器中禁用集群发布/预订功能。 **PSCLUS** 参数主要用于通过意外或不正确地定义集群主题来停止创建发布/预订集群时可能发生的严重问题。 有关禁用此功能的更多信息，请参阅 第 61 页的『禁止集群中的集群发布/预订』。

发布/预订集群: 最佳实践
本主题提供有关规划和管理 IBM WebSphere MQ 发布/预订集群的指导。 这些信息基于客户的测试和反馈。

以下信息假定用户基本了解 IBM WebSphere MQ 集群，发布/预订，并且熟悉 第 41 页的『分布式发布/预订』中的主题。 这些信息并非旨在作为 "一刀切" 的解决方案，而是试图分享共同解决常见问题的方法。

发布/预订集群

通过集群，您可以在需要时在集群中的队列管理器之间进行 "任何到任何" 直接连接。 当集群用于点到点消息传递时，集群中的每个队列管理器仅知道有关其他集群资源 (例如集群中的其他队列管理器和集群队列) 的信息，当连接到这些资源的应用程序请求使用这些资源时; 即，它们在需要知道的基础上工作。

发布/预订集群是具有常规 CLUSSDR 和 CLUSRCVR 通道定义的队列管理器集群。 但是，发布/预订集群还包含至少一个 TOPIC 对象，该对象是在主题对象已标识集群名称的集群中的至少一个队列管理器上定义的。

通过在集群中定义主题对象，连接到集群中一个队列管理器的应用程序可以预订该主题或该主题下的主题树中的任何节点，并从集群中的其他队列管理器接收有关该主题的发布。 此过程通过在集群中标识存在预订的队列管理器的所有其他队列管理器上创建代理预订来实现。 因此，当有关主题的发布在其队列管理器上发生时，他们知道将其转发到集群的其他相应成员，然后从那里将其交付到各个应用程序预订。

要实现此传递，只要将主题添加到集群中，集群中的每个队列管理器都需要知道集群中其他每个队列管理器的身份。 此知识通过集群的完整存储库队列管理器进行传播。 一个队列管理器上的已发布消息仅发送到集群中已知用于托管同一主题的预订的其他队列管理器。 要实现此过程，当应用程序创建对集群主题的预订时，该队列管理器必须通过集群发送方通道与集群中的每个其他队列管理器直接通信以传播代理预订。

此过程与使用集群进行点对点交付时所需的有限的需要了解的信息和连接有很大不同。 因此，发布/预订集群上的需求与点到点集群 (没有任何集群主题的集群) 上的需求不同。

使用集群主题可使在队列管理器之间扩展发布/预订域变得简单，但如果不了解机制和影响，并且考虑了用于发布/预订的集群，那么可能会导致问题。 以下最佳实践旨在帮助理解和准备工作。

总之，集群发布/预订的性能影响可能对大型集群不利，在尝试在现有集群中使用发布/预订之前，需要仔细考虑和了解这些影响。 例如，即使是简单创建集群主题对象。 最好从一个专用于发布/预订活动的小型新集群开始，并从该集群中扩展该集群。

设计发布/预订拓扑

如前所述，在集群中使用发布/预订时，存在容量和性能注意事项。 因此，最好仔细考虑是否需要跨队列管理器进行发布/预订，并将其限制为仅需要它的队列管理器数量。 在确定需要发布和预订一组主题的最小队列管理器集后，可以使这些队列管理器成为仅包含它们的集群的成员，而不包含其他队列管理器。

在已建立的集群中尤其如此，对于点到点消息传递而言，该集群已正常运行。 因此，当您将现有大型集群转变为发布/预订集群时，最好先为发布/预订工作创建一个单独的集群，在该集群中可以尝试使用应用程序，而不是使用当前集群。 可以继续使用已在一个或多个点到点集群中的现有队列管理器，这些队列管理器的子集需要成为新发布/预订集群的成员。 但是，此新集群必须具有单独的队列管理器，这些队列管理器配置为完整存储库，以隔离现有集群完整存储库中的额外负载。

如果您确定集群由于其大小或当前负载而不会用于发布/预订，那么最好通过在集群中的任何队列管理器上简单创建集群主题来防止此集群意外地进入发布/预订集群。 使用 **PSCLUS** 队列管理器属性来实现此设计，有关详细信息，请参阅 禁止集群中的集群发布/预订。

同样重要的是，要仔细选择要添加到集群中的主题: 这些主题的主题树越高，其范围就越广。 因此，建议不要在不考虑所看到的行为的情况下将主题根节点放入集群中。 尽可能使全局主题变得明显，例如，在主题字符串中使用高级限定符: `/global` 或 `/cluster`。

如何调整系统大小

发布/预订集群需要许多通道，因为模型与点到点消息传递不同: 每个队列管理器都需要与该集群中的所有其他队列管理器进行通信。 点对点模型是 "选择性加入" 模型，但发布/预订集群与预订扇出具有不分青红皂白的性质。 因此，完整存储库队列管理器以及在发布/预订集群中托管本地预订的任何队列管理器都必须具有同时建立到集群中每个成员的通道的能力。

最好确保发布/预订集群中的每个队列管理器都可以实现此容量，但已确认已知从不托管预订的队列管理器不需要与其他每个队列管理器建立通道，因此不需要此容量级别。

但是，必须小心，因为在此类队列管理器上创建的意外预订，或者任何尝试手动将此类队列管理器与集群中的其他队列管理器再同步，都会导致同时启动所有通道。 请参阅 第 59 页的『重新同步代理预订』，以了解更多信息。

集群发布/预订支持将一个队列管理器上的已发布消息传递到其他队列管理器上的预订。 但对于点到点消息传递，在队列管理器之间传输消息的成本可能对性能不利。 因此，必须尽可能尝试在发布消息的队列管理器上创建对主题的预订。

另一个考虑因素是对传播代理预订的系统性能的影响。 通常，当创建特定集群主题字符串 (而不仅仅是配置的主题对象) 的第一个预订时，队列管理器会向集群中的每个其他队列管理器发送代理预订消息。 如果发布/预订解决方案包含许多正在预订的唯一主题字符串，或者这些主题经常被预订和取消预订，那么可以在集群中的所有队列管理器之间生成大量代理预订流量，从而对系统的整体性能产生负面影响。 有关减少代理预订开销的方法的信息，请参阅 第 62 页的『集群主题性能』 。

重新同步代理预订

在正常情况下，队列管理器会自动确保系统中的代理预订正确反映集群中每个队列管理器上的预订。

但是，如果需要，您可以使用 REFRESH QMGR TYPE (PROXYSUB) 命令将队列管理器的本地预订与在集群中传播的代理预订手动再同步。

注: 再同步将临时在集群上创建突然的额外代理预订负载，该负载源自发出该命令的队列管理器。 因此，除非 IBM WebSphere MQ 服务， IBM WebSphere MQ 文档或错误日志记录指示您这样做，否则请勿使用它。

需要再同步的一个示例是当队列管理器无法正确传播其代理预订时，这可能是因为通道已停止并且所有消息都无法排队进行传输，或者因为操作员错误导致从 SYSTEM.CLUSTER.TRANSMIT.QUEUE 队列中错误地删除了消息。 在这种情况下，首先纠正原始问题 (例如，通过重新启动通道) ，然后在队列管理器上发出 **REFRESH QMGR TYPE(PROXYSUB)** 命令。 请注意，由于代理预订未就位而丢失的发布不会为受影响的预订恢复。 必须考虑到这一缺点。

再同步要求队列管理器启动到集群中所有其他队列管理器的通道。 因此，刷新的队列管理器必须具备足够强大的功能，来应对与集群中的所有其他队列管理器通信的情况。

集群主题
集群主题是定义了 **cluster** 属性的管理主题。 有关集群主题的信息将推送到集群的所有成员，并与本地主题组合以在每个队列管理器上创建不同的主题空间。

在队列管理器上定义集群主题时，会将集群主题定义发送到完整存储库队列管理器。 然后，完整存储库会将集群主题定义传播到集群中的所有队列管理器，从而使相同的集群主题可用于集群中任何队列管理器处的发布者和订户。 您在其中创建集群主题的队列管理器称为集群主题主机。 集群主题可以由集群中的任何队列管理器使用，但必须在定义该主题的队列管理器 (主机) 上对集群主题进行任何修改，在该队列管理器上，修改将通过完整存储库传播到集群的所有成员。

在每个队列管理器上，将根据它知道的本地和集群主题定义来构造单个主题名称空间。 当应用程序预订解析为集群主题的主题时， IBM WebSphere MQ 会创建代理预订，并将其直接从创建该预订的队列管理器发送到集群的所有其他成员。 与集群主题本身不同，代理预订不会流经完整的存储库队列管理器。

在主题上发布的消息将发送到发布程序所连接的队列管理器已知的每个预订。 如果其中任何预订是代理预订，那么将向发起代理预订的队列管理器发送已发布消息的副本。 然后，接收队列管理器将消息的副本发送到每个本地预订。 此过程确保集群主题的订户接收来自连接到集群中任何队列管理器的发布者的发布，并确保通过集群传播最小数量的已发布消息。

如果您具有集群主题和本地主题对象，那么本地主题优先。 请参阅第 60 页的『多个集群主题定义』以获取更多信息。

有关用于显示集群主题的命令的更多信息，请参阅以下相关链接:

通配符预订

当对在集群主题对象处或下方解析的主题字符串进行本地预订时，将创建代理预订。 如果在主题层次结构中进行的通配符预订高于任何集群主题，那么不会在集群周围为匹配的集群主题发送代理预订，因此不会从集群的其他成员接收任何发布。 但是，它会从本地队列管理器接收发布。

但是，如果另一个应用程序预订解析为集群主题或低于集群主题的主题字符串，那么将生成代理预订并将发布内容传播到此队列管理器。 原件到达时，更高的通配符预订将被视为这些出版物的合法接收方并接收副本。

此行为与本地发布的相同主题的消息不同。 如果不需要此行为，那么在集群主题上设置 **WILDCARD(BLOCK)** 会使原始通配符不被视为合法预订，并且不会在集群主题或其子主题上接收任何发布(本地发布或来自集群中的其他位置)。

相关概念

处理管理主题

使用预订

相关参考

DISPLAY TOPIC

DISPLAY TPSTATUS

显示子项

集群主题属性

设计和管理发布/预订集群需要对集群主题属性有很好的了解。

主题对象具有许多适用于多队列管理器发布/预订拓扑的属性。 当您使用 IBM WebSphere MQ 集群来创建此类拓扑时，这些属性具有以下行为。

PROXYSUB

- **PROXYSUB** 是用于控制何时进行代理预订的属性。 有关您可能希望将此属性更改为缺省值 FIRSTUSE 的原因的详细信息，请参阅 第 45 页的『有关路由机制的更多信息』。
- 通过与集群主题的其他属性相同的方式， **PROXYSUB** 属性将传播到集群中的每个队列管理器，而不仅仅是定义主题的队列管理器。 这将立即导致集群中的每个队列管理器创建对每个其他队列管理器的通配符代理预订。 此过程的结果是，每个队列管理器都将创建集群发送方通道到每个其他队列管理器，并且发布的任何消息都将发送到每个队列管理器。

PUBSCOB 和 SUBSCOPE

PUBSCOPE 和 **SUBSCOPE** 确定此队列管理器是将发布传播到拓扑 (发布/预订集群或层次结构) 中的队列管理器，还是将作用域限制为仅其本地队列管理器。 您可以使用 MQPMO_SCOPE_QMGR/ MQSO_SCOPE_QMGR 以编程方式执行等效作业。

- **PUBSCOPE** 如果使用 **PUBSCOPE(QMGR)** 定义了集群主题对象，那么该定义将与集群共享，但基于该主题的发布的作用域仅为本地，并且不会将其发送到集群中的其他队列管理器。
- **SUBSCOPE** 如果使用 **SUBSCOPE(QMGR)** 定义了集群主题对象，那么该定义将与集群共享，但基于该主题的预订的作用域仅为本地，因此不会向集群中的其他队列管理器发送任何代理预订。

这两个属性通常一起用于隔离队列管理器，使其不与特定主题上的集群其他成员进行交互。 队列管理器既不向集群的其他成员发布这些主题的出版物，也不从集群的其他成员接收这些主题的出版物。 如果在子主题上定义了主题对象，那么此情境不会阻止发布或预订。

在主题的本地定义上将 **SUBSCOPE** 设置为 QMGR 不会阻止集群中的其他队列管理器将其代理预订传播到队列管理器 (如果他们正在将该主题的集群版本与 **SUBSCOPE(ALL)** 配合使用)。 但是，如果本地定义还将 **PUBSCOPE** 设置为 QMGR ，那么不会从此队列管理器向这些代理预订发送发布。

多个集群主题定义

本地主题定义覆盖远程定义的同名集群主题定义。 还可以在集群中的不同队列管理器上创建同一集群主题的多个定义。 但是，这两种情况都需要谨慎，原因在本主题中进行了说明。

就像集群队列一样，在集群中具有同一集群主题对象的多个定义会引入在每个集群上定义不同属性的可能性。 确定集群中每个队列管理器看到的主题定义版本并不容易，因此很难确定期望的行为。

如果针对单个主题字符串的两个或多个集群主题定义具有不同的属性或存在于多个集群中，那么会将消息 (AMQ5465 & AMQ5466) 写入错误日志，并使用最近接收到的集群主题定义。

集群主题主机队列管理器不得删除主题定义，并保留在集群中以确保集群主题继续为集群的所有成员所知。 由于集群主题定义由完整存储库队列管理器以及其部分集群存储库中的所有其他队列管理器进行高速缓存，因此此主机队列管理器持续可用并不重要。 通过这种高速缓存机制，可以在主机队列管理器不可用的情况下至少 60 天内可以使用集群主题定义。 有关此主题的更多信息，请参阅 第 65 页的『发布/预订集群队列管理器的关键角色』。

在本地覆盖集群主题定义

可能需要在集群中的特定队列管理器上覆盖集群主题的行为。 可以通过定义本地主题对象来覆盖具有相同主题字符串的集群主题对象，并将其用于仅发布到本地连接的订户来实现此覆盖。

即使创建了主题的本地定义以覆盖队列管理器上的集群主题，队列管理器也会继续使用集群主题定义从集群的其他成员接收代理预订。 缺省情况下，本地发布的消息将继续发送到远程队列管理器以实现代理预订。 如果不需要此安排，请在本地主题对象上指定 **PUBSCOPE**(QMGR)，以确保连接到此队列管理器的发布程序应用程序仅发布到本地订户。

修改集群主题定义

如果需要变更集群主题定义，请在与集群主题主机定义相同的队列管理器上对其进行修改。 请勿在集群中的另一个队列管理器上创建同一集群主题的定义。 再次定义主题会导致同一集群主题有两个集群主题主机。

多次定义集群主题会产生潜在的冲突定义，并且不同队列管理器可能会在不同时间使用不同的定义。

将集群主题定义移至集群中的其他队列管理器

您可能需要将集群主题定义从集群中的一个队列管理器移至另一个队列管理器，例如在从集群中停用队列管理器时。 要将集群主题定义移至集群中的其他队列管理器而不中断发布流，您需要执行以下步骤。 此示例将定义从 QM1 移至 QM2。

1. 使用与 QM1 定义相同的属性在 QM2 上创建重复的集群主题定义。
2. 等待完整存储库队列管理器在整个集群中传播新定义。 可以通过使用 **(DISPLAY CLUSTER)** 命令在每个集群成员上显示集群主题并检查源自 QM2 的定义来确定传播。
3. 从 QM1 中删除集群主题定义。

从 QM1 中删除原始定义后，如果需要，可以在 QM2 上修改该定义，而不会在属性中引入冲突。

在失败的队列管理器上替换集群主题定义

在先前的 方案中，如果 QM1 在一段时间内不可用，那么可能无法从 QM1 中删除定义。 在此场景中，可以接受在存在这两个定义的情况下运行。

如果这样就需要修改集群主题定义，那么可以修改 QM2 上的版本，因为知道 QM2 定义比 QM1 定义更新，因此占上风。 但是，在此时间段内，会将错误写入队列管理器的错误日志，因为存在冲突的集群主题定义。 通过从 QM1 中移除可重新启动的重复集群主题定义，尽快解决该错误。

或者，如果 QM1 永远不会返回到集群 (例如，灾难性硬件故障后意外停用)，那么可以使用 RESET CLUSTER 命令来强制弹出队列管理器。 **RESET CLUSTER** 会自动删除目标队列管理器上托管的所有主题对象。

禁止集群中的集群发布/预订
必须仔细规划将发布/预订引入队列管理器集群，尤其是现有集群，以适应性能的任何降低。

将集群主题引入到大型 IBM WebSphere MQ 集群 (其中包含许多队列管理器) 可能会立即导致集群中的每个队列管理器上的额外负载，并且在某些情况下会降低性能。 因此，必须仔细规划发布/订阅的引入。 请参阅 第 62 页的『集群主题性能』以获取更多信息。

如果已知集群无法容纳发布/预订的开销，那么可以通过将队列管理器属性 **PSCLUS** 设置为 DISABLED 来禁用队列管理器中的集群发布/预订功能。

将 **PSCLUS** 设置为 DISABLED 将修改队列管理器功能的三个方面:

- 此队列管理器的管理员不再能够将主题对象定义为集群对象。
- 将拒绝来自其他队列管理器的入局主题定义或代理预订 (将记录一条警告消息以通知管理员配置不正确)。
- 当完整存储库接收到主题定义时，它们不再自动与所有其他部分存储库共享有关每个队列管理器的信息。

虽然 **PSCLUS** 是集群中每个单独队列管理器的参数，但它并非旨在选择性地禁用集群中队列管理器子集中的发布/预订。 除其他外，此方法将导致频繁的错误消息被视为代理预订，并且将不断地看到和拒绝主题定义。 理想情况下，使用此选项时，请始终将集群中的所有队列管理器设置为禁用。 当队列管理器参与一个或多个发布预订集群以及一个或多个传统集群时， **PSCLUS** 必须在该队列管理器上设置为 ENABLED 。 请参阅以下有关在完整存储库中禁用的信息。

重要的是，在集群中的所有完整存储库队列管理器上将 **PSCLUS** 设置为 DISABLED 可防止未正确配置的部分存储库上的任何集群主题定义影响集群中的其他队列管理器。 在此类情况下，将在完整存储库队列管理器的错误日志中报告不一致情况。

将传统的点到点集群与发布预订集群重叠时，请务必在每个集群中使用一组单独的完整存储库。 此安排允许主题定义和 "所有队列管理器" 信息仅在发布/预订集群中流动。

对此参数的使用有一些注意事项，这有助于避免配置不一致。 在从 **已启用** 修改为 **已禁用**时，此队列管理器是其成员的任何集群中都不能存在任何集群主题对象。 在禁用此功能之前，必须删除任何此类主题 (即使是远程定义的主题)

有关 **PSCLUS** 的更多信息，请参阅 ALTER QMGR (PSCLUS)。

集群主题性能
集群主题的性能特征需要特别考虑，因为它们与集群队列的性能特征不同，考虑不周的使用可能是大型或不平衡集群中性能问题的根源。

降低发布/预订对性能的影响

集群中的队列管理器上有两个工作负载源: 直接处理应用程序的消息，以及处理管理集群所需的消息和通道。 在典型的点到点集群中，集群系统工作负载主要限于集群成员根据需要明确请求的信息 (请参阅 第 64 页的『发布/预订集群的性能特征』中的比较)。 因此，在非常大的集群 (例如包含数千个队列管理器的集群) 之外的任何其他集群中，在考虑队列管理器性能时，您可以在很大程度上降低管理集群的性能效果。

在发布/预订集群中，将信息 (例如集群主题和代理预订) 推送到集群的所有成员，而不考虑是否所有集群队列管理器都在主动参与发布/预订消息传递。 此过程可能会在系统上产生显着的额外负载。 因此，您需要考虑集群管理对队列管理器性能的影响，包括其计时和大小。

要降低发布/预订集群管理对集群性能的影响，请考虑以下两个建议:

1. 在一天中的非高峰时间执行集群，主题和预订更新。
2. 如果仅因为集群已存在而考虑向现有大型集群添加发布/预订主题，请考虑是否可以定义参与发布/预订的队列管理器的小得多的子集，并使该集群成为 "重叠的" 集群。 然后，此集群是定义了集群主题的集群。 虽然某些队列管理器现在位于两个集群中，但发布/预订的总体效果会降低:
   a. 发布/预订集群的大小较小。
   b. 不在发布/预订集群中的队列管理器受集群管理流量的影响要小得多。

平衡生产者和消费者

异步消息传递性能中的一个重要概念是 *balance*。 除非消息使用者与消息生产者平衡，否则积压的未使用消息可能会累积并严重影响多个应用程序的性能。

在点到点消息传递拓扑中，易于理解消息使用者与消息生产者之间的关系。 您可以获取消息生产和使用估算，按队列排列，按通道排列。 如果缺乏平衡，就容易发现瓶颈，然后加以补救。

在发布/预订拓扑中，更难确定发布者和订户是否均衡。 从解析为集群主题的每个预订开始，并返回到具有主题发布程序的队列管理器。 计算从每个队列管理器流向每个订户的发布数。

与集群中远程队列管理器上的预订匹配的每个发布 (基于代理预订) 都将放入 SYSTEM.CLUSTER.TRANSMIT.QUEUE 。 如果多个远程队列管理器具有该发布的代理预订，那么会将消息的多个副本放入传输队列，目标是不同的集群发送方通道。

这些发布以远程队列管理器上的 SYSTEM.INTER.QMGR.PUBS 队列为目标。 每个队列管理器处理到达该队列的消息，并将它们传递到该队列管理器上的正确预订。

因此，请在以下可能出现瓶颈的位置监视负载:

- 各个预订队列本身:
  - 此瓶颈将意味着预订应用程序不会像正在发布的发布一样快速使用这些发布。
- SYSTEM.INTER.QMGR.PUBS 队列:
  - 队列管理器接收来自一个或多个远程队列管理器的发布的速度比它可以将它们分发到本地预订的速度要快。
- 发布队列管理器上的发布队列管理器，预订队列管理器和集群传输队列之间的集群通道。(缺省情况下为 SYSTEM.CLUSTER.TRANSMIT.QUEUE):
  - 一个或多个集群通道未在运行，或者消息发布到本地队列管理器的速度超过了这些通道可以将它们传递到远程队列管理器的速度。
- 如果发布应用程序正在使用已排队的发布/预订接口，那么还需要考虑 SYSTEM.BROKER.DEFAULT.STREAM 队列和 SYSTEM.QPUBSUB.QUEUE.NAMELIST 中列出的任何其他流队列，以及 SYSTEM.BROKER.DEFAULT.SUBPOINT 队列和 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 中列出的任何其他子点队列:
  - 本地发布应用程序将消息放入的速度比本地队列管理器可以处理消息的速度要快。

预订性能注意事项

如前所述，在队列管理器上针对解析为集群主题的主题字符串进行预订时，该队列管理器必须确保集群中的每个其他队列管理器都具有针对该主题的代理预订。 要实现此结果，队列管理器将创建代理预订消息并将其发送到集群中的每个其他队列管理器。

使用缺省配置时，创建对集群主题的预订不会导致发送新的代理预订的唯一时间是在本地队列管理器上已存在对完全相同的主题字符串的预订时。 在此情况下，不需要其他代理预订，因为到达的发布将传递到所有匹配的预订，而不仅仅是主题字符串的原始预订。

有关缺省配置的替代方法，请参阅 第 64 页的『禁用个别代理预订』。

不考虑预订选择器，因此对同一主题字符串的两个预订 (但使用不同的选择器) 仍共享代理预订。 此情境还可能意味着将与主题字符串匹配的发布传播到订户队列管理器，即使该发布与预订的选择器不匹配也是如此。

当从队列管理器中删除主题字符串的最后一个预订时，将创建与代理预订消息相同的消息并将其发送至所有队列管理器。 此过程将从远程队列管理器中除去代理预订。

由于这些原因，集群的大小和不同主题字符串的预订频率会对集群本身造成很大的负载，在规划集群和要由发布/预订应用程序使用的主题时必须考虑这些负载。

在考虑来自代理预订流量的系统负载时，除了监视 第 62 页的『平衡生产者和消费者』 部分中列出的队列外，还要监视以下队列。

- SYSTEM.INTER.QMGR.FANREQ 队列。
- SYSTEM.INTER.QMGR.CONTROL 队列。

这些队列上的任何重要消息积压意味着预订更改速率对于系统太大，或者队列管理器在集群中未正常工作。由于已禁用发布/预订支持 (请参阅 ALTER QMGR 中的 **PSMODE**) 或发生需要进一步调查的问题，此时请检查队列管理器错误日志。

减少代理预订流量

如果代理预订开销较高，那么应采取步骤来减少此开销。 这可能通过常规主题合并或更改为队列间管理器发布的广播模型来实现。

一般的发布/预订建议是评估主题字符串的使用情况，以确定它们是否可以合并，从而降低系统资源的总体负载。 使用许多不同的瞬态主题字符串会在连接发布程序或预订的系统中的每个队列管理器上引入某种级别的管理开销。 减少主题字符串的数量和瞬态性质，因此对它们的发布者和预订将减少对系统的影响。

减少代理预订流量的一种方法是在同一队列管理器上查找同一主题字符串的预订。 此方法允许此队列管理器将单个代理预订发送到其他队列管理器，而不是让多个队列管理器发送代理预订，每个队列管理器都针对同一主题字符串上自己的一组预订。 此实践还会优化集群中的发布路由。

禁用个别代理预订

在某些情况下，如果在集群中预订的一组不同的主题字符串很大且不断更改，那么最好从预订传播模型更改为发布广播模型。 在此首选模型中，任何集群主题上的每个发布都会自动发送到集群中的每个队列管理器，而不考虑这些队列管理器上是否存在预订。

然后，接收队列管理器可以将消息传递到存在的本地预订，或者废弃该消息。 在此模型中，不需要根据预订的存在来创建和删除单个代理预订。 在此方式下运行时，发布的消息资源负载很可能随着所有发布发送到所有队列管理器而增加。 因此，集群中的队列管理器必须具有处理此额外负载的能力。

使用以下配置步骤启用广播模型:

1. 每个托管预订的队列管理器都必须配置为不发送与集群主题的本地预订匹配的代理预订。 在定义集群主题或在集群中创建预订之前，此配置需要在每个队列管理器 `qm.ini` 文件中设置以下调整参数:

   ```
   TuningParameters:
       pscProxySubFlags=1
   ```

2. 设置调整参数后，必须重新启动所有队列管理器。
3. 在重新启动队列管理器之后，可以定义集群主题。 每个集群主题都必须将 **PROXYSUB** 设置为 FORCE。

逆转行为

要撤销先前在 第 64 页的『禁用个别代理预订』中描述的操作方式，请使用以下步骤:

1. 从每个队列管理器的 `qm.ini` 文件中除去调整参数。
2. 重新启动每个队列管理器。
3. 在每个托管预订的队列管理器上发出 **REFRESH QMGR TYPE(PROXYSUB)** 命令。
4. 将 **PROXYSUB** 设置为集群主题上的 FIRSTUSE 。

警告: 在此行为的启用和撤销过程中，如果未按记录的顺序完成所有步骤，那么可能不会发生针对预订的正确发布流。

注: 设置 PROXYSUB (到 FORCE) 的影响

如本主题中先前所述， **PROXYSUB(FORCE)** 主题属性可以减少代理预订流量，但必须谨慎使用此属性。 **PROXYSUB(FORCE)** 属性将传播到集群中的每个队列管理器，而不仅仅是定义主题的队列管理器。 这将立即导致集群中的每个队列管理器创建对每个其他队列管理器的通配符代理预订。 此过程的结果是，每个队列管理器都将创建集群发送方通道到每个其他队列管理器，并且任何已发布的消息都将发送到每个队列管理器。

在大型或繁忙集群中设置此属性可能会导致系统资源上的额外负载。

发布/预订集群的性能特征

请务必考虑更改发布/预订集群的属性 (例如，向集群添加队列管理器，主题或预订) 如何影响集群中运行的应用程序的性能。

将点到点集群与发布/预订集群就两个管理任务进行比较。

首先，点对点集群:

1. 定义新集群队列时，会将目标信息推送到完整存储库队列管理器，并且仅当其他集群成员首次引用集群队列时 (例如，应用程序尝试将其打开) 才会将目标信息发送给其他集群成员。 然后，队列管理器将在本地高速缓存此信息，以消除每次访问队列时远程检索此信息的需要。
2. 将队列管理器添加到集群不会直接影响其他队列管理器上的负载。 有关新队列管理器的信息将推送到完整存储库，但仅当流量开始流入或流出新队列管理器时，才会创建并启动从集群中的其他队列管理器到新队列管理器的通道。

简而言之，点到点集群中队列管理器上的负载与它为应用程序处理的消息流量相关，而与集群的大小不直接相关。

第二，发布/预订集群:

1. 定义新集群主题时，会将信息推送到完整存储库队列管理器，并立即从该存储库直接推送到集群的所有成员，从而导致通道从完整存储库启动到集群的每个成员 (如果尚未启动)。
2. 在新主题字符串上创建对集群主题的预订时，会立即将该信息从该队列管理器直接推送到集群的所有其他成员，从而导致通道从该队列管理器启动到该集群的每个成员 (如果尚未启动)。
3. 当新队列管理器加入现有集群时，会将有关所有集群主题的信息从完整存储库队列管理器推送到该集群。 然后，新队列管理器将集群中集群主题的所有预订的知识与集群的所有成员同步，从而创建通道并从新队列管理器启动到集群的每个成员。

总之，集群中任何队列管理器上的集群管理负载都会随集群中的队列管理器，集群主题和代理预订的数量而增加，而不考虑在每个队列管理器上本地使用这些集群主题。

发布/预订集群队列管理器的关键角色
与点到点集群类似，发布/预订集群中的队列管理器有两个关键角色: 作为完整存储库队列管理器和作为集群主题主机。

完整存储库
完整存储库队列管理器具有将对象定义推送到集群的其他成员的角色; 在发布/预订集群的情况下，将集群主题对象定义推送到集群的其他成员。

集群主题主机
集群主题主机是定义了集群主题对象的队列管理器。 您可以在发布/预订集群中的任何队列管理器上定义集群主题对象。 将集群主题对象推送到完整存储库队列管理器，然后将其推送到集群中的所有其他队列管理器，在这些队列管理器中对其进行高速缓存，以供集群中任何队列管理器上运行的发布者和订户使用。

可用性和管理

您应该在集群中定义两个完整存储库，以最大限度提高集群中集群主题定义的可用性。

对于排队的消息传递集群，在许多计算机中只有两台高可用性计算机的发布/预订集群中，最好将高可用性计算机定义为完整存储库。

在已排队的集群中，您可以通过在集群中的多个队列管理器上 定义 同一集群队列来提高集群队列的可用性和吞吐量。 然后在消息之间进行工作负载均衡。 相反，在发布/预订集群中，集群主题在集群中的所有队列管理器上 可用 ，但不会执行发布/预订流量的工作负载均衡。 相反，应该将单独的预订和发布程序分布在不同的队列管理器中，以分布发布/预订负载。 如果定义了集群主题的队列管理器变为不可用，那么其他队列管理器将继续处理该主题的发布/预订请求。

但是，如果定义了集群主题对象的队列管理器再也不可用，那么最终将删除其他队列管理器上的高速缓存主题对象，并且该主题变为不可用。 此过程将在至少 60 天后 (取决于上次刷新主题定义的时间) 从主题定义变为不可用的时间开始执行。

使用 60 天时间段来恢复定义了集群主题对象的队列管理器，因此几乎不需要采取特殊措施来使集群主题主机具有高可用性。 60 天的时间足以应付技术上的问题; 60 天的时间可能只会因为行政上的错误而超过。 为了降低这种可能性，如果集群主题主机不可用，那么集群的所有成员每小时都会写入错误日志消息，表明其高速缓存的集群主题对象未刷新。 通过确保定义了集群主题对象的队列管理器正在运行来响应此消息。

您可以采用在其他队列管理器上定义同一集群主题对象的做法。 每个定义都会导致将额外的集群主题对象推送到集群中的其他队列管理器 (包括其他集群主题主机)。 现在，如果集群主题主机超过 60 天变为不可用，那么将仅从其他主机中除去其集群主题对象的版本。 将保留集群主题对象的其他版本。 要求集群中特定主题的所有定义都相同，否则很难确定队列管理器正在使用哪个主题定义。 任何主机上的最新副本始终是所使用的集群主题对象。

将多集群主题定义的新增保护与增加的管理复杂性进行权衡: 随着复杂性的增加，出现人为错误的可能性会更大。

与托管集群队列不同，作为集群主题定义的主机队列管理器不会引入任何其他应用程序消息流量。 该流量仅限于创建预订和发布消息的队列管理器。 可以在既不执行任何操作的队列管理器上托管集群主题。 这种情

况意味着虽然不是必需的，但通常应该在集群的完整存储库队列管理器上托管集群主题，因为这些队列管理器可能具有更高级别的可用性，并且对它们具有更严格的管理控制。 这种安排降低了错误地修改或删除定义甚至队列管理器的可能性。

重叠集群支持和发布/预订

通过 IBM WebSphere MQ 集群，单个队列管理器可以是多个集群的成员。 这种安排称为重叠集群。 当集群在队列管理器中重叠时，发布/预订集群中的集群主题与队列的行为不同。 使用具有重叠集群的集群发布/预订时，必须清楚地了解此行为。

与队列不同，无法将主题定义与多个集群相关联。 因此，在集群中创建的代理预订的作用域仅限于在其中定义集群主题的单个集群。 但是，每个队列管理器都有一个主题树，该树包含所有本地主题和任何已知的集群主题 (来自它们是其成员的任何集群)。 因此，可以构建这样一个发布/预订行为可能难以理解的系统。

集成多个发布/预订集群

对于点到点消息，使单个队列管理器成为多个集群的成员的原因是在两个集群之间创建集群网关。 有关此主题的更多信息，请参阅 重叠集群 。 此集群网关支持将源自一个集群的点到点消息路由到另一个集群中的查询。 发布/预订集群继承从传统队列管理器集群重叠的功能。 但是，不能使用此机制将发布和预订从一个集群路由到另一个集群。

相反，要将发布和预订从一个集群中的队列管理器传递到另一个集群，必须使用发布/预订层次结构将队列管理器链接在一起。 可以通过在一个集群中的一个队列管理器与另一个集群中的另一个队列管理器之间显式创建父子分层关系来实现此安排。 此关系支持集群之间的所有代理预订的流，从而支持任何匹配的发布。 有关此关系的更多信息，请参阅 第 68 页的『发布/预订层次结构』。

限制集群之间的发布和预订流的方法是使用既不在集群中的网关队列管理器; 请参阅 第 83 页的『组合和隔离多个集群中的主题空间』。

重叠集群，单个主题树

每个队列管理器都有一个包含本地主题和所有已知集群主题的 主题树 。 使用发布/预订的两个集群重叠的另一个考虑因素是，每个集群中的队列管理器可以定义具有相同名称的集群主题，也可以定义具有相同主题字符串的不同名称的集群主题。 在作为两个集群成员的队列管理器上，当向它们通知多个集群主题定义 (每个集群一个定义) 时，会发生冲突。 队列管理器报告了问题，但队列管理器继续运行，仅使用最新的集群主题定义。 因此，行为将变为非确定性行为，无法依赖此行为。

因此，使用集群发布/预订的重叠集群必须考虑其主题定义名称空间以跨所有集群，并对其主题对象进行命名，并相应地构造其主题字符串。 然后，可以使用重叠中的队列管理器以可预测方式发布和预订这两个集群。

在 第 67 页的图 33 中， T_B 和 T_C 是不重叠的主题定义。 连接到 QM3 的发布程序在集群中重叠，能够发布到它们各自集群中的两个主题。 连接到重叠中的 QM3 的订户能够预订两个集群中的主题。

另一种考虑 第 67 页的图 33 的方法是考虑代理预订。 连接到队列管理器 QM3 的应用程序对解析为主题对象 T_B (仅在 CLUSTER 1 中存在) 的主题进行预订，导致仅将代理预订从队列管理器 QM3 发送到队列管理器 QM1 和 QM2 。 连接到队列管理器 QM3 的应用程序预订了解析为主题对象 T_C (仅存在于 CLUSTER 2 中) 的主题。 预订导致仅将代理预订从队列管理器 QM3 发送到队列管理器 QM4 和 QM5 。

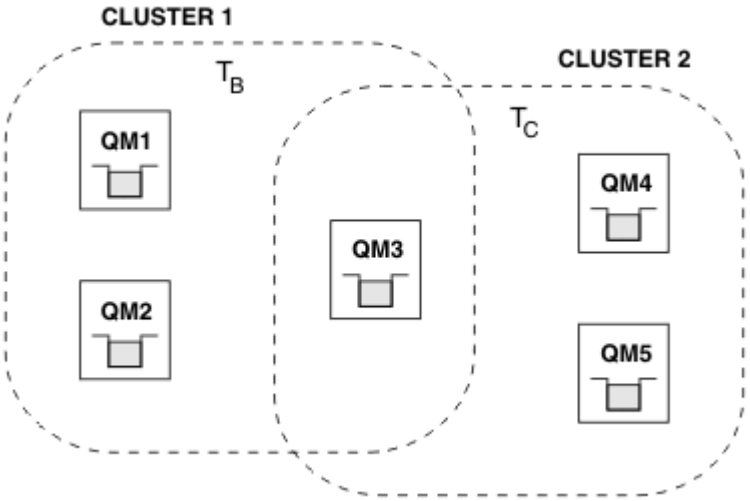


图 *33:* 重叠集群*:* 两个集群各预订不同的主题

不在重叠中的队列管理器的发布者和订户只能发布和预订其集群中的主题，例如， QM2 上的主题字符串的订户不会接收发布到从 QM5 发布的相同主题字符串的消息，而不考虑对主题进行集群。 要实现此安排，需要发布/预订层次结构。

重叠集群，通配符预订

由于本主题的前一部分中的原因，使用通配符来预订属于多个集群的队列管理器上的主题时必须小心。

在上一个示例中，假定两个主题对象配置为:

- T_B: 主题名称 'Football'，集群 'CLUSTER1'. 主题字符串 '/Sport/Football'
- T_C: 主题名称 'Tennis'，集群 'CLUSTER2'. 主题字符串 '/Sport/Tennis'

在此场景中，两个集群主题显然是分开的，主题名称或主题字符串中没有重叠。

连接到 QM3 的应用程序可以创建对 '/Sport/Football' 的预订和对 '/Sport/Tennis'的预订。 然后，他们将接收来自两个集群的任何出版物。 但是，如 第 30 页的『管理主题对象』中所述，如果要预订 '/Sport/#'，以便同时在 '/Sport/Football' 和 '/Sport/Tennis'上接收发布，那么在任一集群中都不会将此模型识别为集群主题，因此不会创建代理预订。 然后，它们将错过来自任一集群中的其他队列管理器的发布。

如前所述，在 CLUSTER 1 和 CLUSTER 2 中都为 '/Sport/#' 创建集群主题是无效的，因为这些集群主题会发生冲突，并且会将参考消息写入错误日志以指示此情况。 但是，仅允许 "允许" 在其中一个集群 (例如 CLUSTER 1) 中创建此类主题。 现在，对 QM3 中 '/Sport/#' 的预订将导致仅将代理预订发送到 CLUSTER 1 中的队列管理器，因此仍将无法接收来自 QM4 或 QM5 的发布到 '/Sport/Tennis' 。

此场景中的唯一解决方案是继续创建两个单独的预订。

针对发布/预订集群的 *REFRESH CLUSTER* 注意事项

发出 **REFRESH CLUSTER** 命令会导致队列管理器临时废弃本地保存的有关集群的信息，包括任何集群主题及其关联的代理预订。

从发出 **REFRESH CLUSTER** 命令到队列管理器完全了解集群发布/预订的必要信息所花费的时间取决于集群的大小，可用性以及完整存储库队列管理器的响应能力。

在刷新处理期间，会中断发布/预订集群中的发布/预订流量。 对于大型集群，使用 **REFRESH CLUSTER** 命令可能会在集群进行中时中断集群，并且在集群对象自动向所有相关队列管理器发送状态更新后的 27 天时间间隔内再次中断集群。 请参阅在大型集群中刷新可能会影响集群的性能和可用性。 由于这些原因，只有在 IBM 支持中心的指导下，才必须在发布/预订集群中使用 **REFRESH CLUSTER** 命令。

对集群的中断可能会在外部显示为以下症状:

- 此队列管理器上集群主题的预订未从连接到集群中其他队列管理器的发布程序接收发布。
- 发布到此队列管理器上的集群主题的消息不会传播到其他队列管理器上的预订。

- 在此期间创建的此队列管理器上的集群主题预订未一致地将代理预订发送给集群的其他成员。
- 在此期间删除的此队列管理器上的集群主题预订无法一致地从集群的其他成员中除去代理预订。
- 在消息传递中暂停 10 秒或更长时间。
- **MQPUT** 失败，例如， MQRC_PUBLICATION_FAILURE。
- 放置在死信队列上的发布，原因为 MQRC_UNKNOWN_REMOTE_Q_MGR

由于这些原因，需要先停顿发布/预订应用程序，然后再发出 **REFRESH CLUSTER** 命令。

另请参阅 **REFRESH CLUSTER** 和 集群使用说明: 使用 REFRESH CLUSTER 最佳实践。

在发布/预订集群中的队列管理器上发出 **REFRESH CLUSTER** 命令后，请等待直到成功刷新所有集群队列管理器和集群主题，然后再同步代理预订，如 第 59 页的『重新同步代理预订』中所述。 此安排要求集群发送方通道从此队列管理器启动到集群中的所有其他队列管理器。 当所有代理预订都已正确再同步时，请重新启动发布/预订应用程序。

如果 **REFRESH CLUSTER** 命令需要很长时间才能完成，请通过查看 SYSTEM.CLUSTER.COMMAND.QUEUE 的 CURDEPTH 来对其进行监视。

相关概念

运行 REFRESH CLUSTER 时发现的应用程序问题

集群：使用 REFRESH CLUSTER 最佳实践

相关参考

MQSC 命令参考：REFRESH CLUSTER

发布/预订层次结构

队列管理器可以在层次结构中分组在一起，其中层次结构包含一个或多个直接连接的队列管理器。 队列管理器使用连接时间父代和子关系连接在一起。 首次将两个队列管理器连接在一起时，子队列管理器将连接到父队列管理器。

在层次结构中连接父队列管理器和子队列管理器时，它们之间没有功能差异，直到您断开队列管理器与层次结构的连接。

注: IBM WebSphere MQ 分层连接要求将队列管理器属性 PSMODE 设置为 ENABLED。

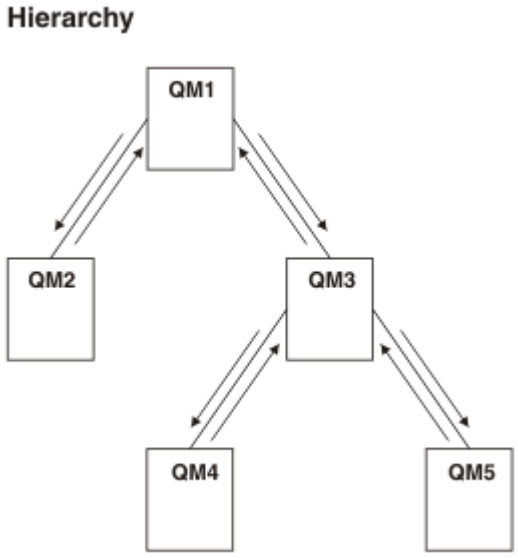


图 *34:* 简单发布/预订层次结构

将队列管理器连接到代理层次结构

您可以将本地队列管理器连接到父队列管理器以修改代理层次结构。

开始之前

1. 启用排队的发布/预订方式。 请参阅 启动排队的发布/预订。
2. 此更改将使用 IBM WebSphere MQ 连接传播到父队列管理器。 有两种方法可以建立连接。
   - 将队列管理器连接到 IBM WebSphere MQ 集群，请参阅 将队列管理器添加到集群
   - 使用与父队列管理器同名的传输队列或队列管理器别名来建立点到点通道连接。 有关如何建立点到点通道连接的更多信息，请参阅 WebSphere MQ 分布式消息传递技术。

关于此任务

使用 `ALTER QMGR PARENT (PARENT_NAME)` runmqsc 命令将子代连接到父代。

分布式发布/预订通过使用队列管理器集群和集群主题定义来实现。 对于与 IBM WebSphere MQ Version 6.0 和 WebSphere Message Broker Version 6.1 和 WebSphere Event Broker Version 6.1 及更低版本的互操作性，只要启用了排队发布/预订方式，您还可以将 Version 7.1 或更高版本的队列管理器连接到代理层次结构。

过程

ALTER QMGR PARENT (PARENT)

示例

第一个示例显示了如何将 QM2 作为 QM1 的子代进行连接，然后查询 QM2 以获取其连接:

```
C:>runmqsc QM2
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM2
alter qmgr parent(QM1)
     1 : alter qmgr parent(QM1)
AMQ8005: WebSphere MQ queue manager changed.
display pubsub all
     2 : display pubsub all
AMQ8723: Display pub/sub status details.
   QMNAME(QM2)                             TYPE(LOCAL)
   STATUS(ACTIVE)
AMQ8723: Display pub/sub status details.
   QMNAME(QM1)                             TYPE(PARENT)
   STATUS(ACTIVE)
```

下一个示例显示了查询 QM1 以获取其连接的结果:

```
C:\Documents and Settings\Admin>runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM1.
display pubsub all
     2 : display pubsub all
AMQ8723: Display pub/sub status details.
   QMNAME(QM1)                             TYPE(LOCAL)
   STATUS(ACTIVE)
AMQ8723: Display pub/sub status details.
   QMNAME(QM2)                             TYPE(CHILD)
   STATUS(ACTIVE)
```

下一步做什么

您可以在一个代理或队列管理器上定义可供已连接队列管理器上的发布者和订户使用的主题。 有关更多信息，请参阅 定义管理主题

相关概念

流和主题

WebSphere MQ 发布/预订消息传递简介

相关参考

显示发布预订

断开队列管理器与代理层次结构的连接
将子队列管理器与代理层次结构中的父队列管理器断开连接。

关于此任务

使用 **ALTER QMGR** 命令将队列管理器与代理层次结构断开连接。 您可以随时按任何顺序断开队列管理器的连接。

当队列管理器之间的连接正在运行时，将发送更新父代的相应请求。

过程

```
ALTER QMGR PARENT('')
```

示例

```
C:\Documents and Settings\Admin>runmqsc QM2
5724-H72 (C) Copyright IBM Corp. 1994, 2024. ALL RIGHTS RESERVED.
Starting MQSC for queue manager QM2.
    1 : alter qmgr parent('')
AMQ8005: WebSphere MQ queue manager changed.
    2 : display pubsub type(child)
AMQ8147: WebSphere MQ object  not found.
display pubsub type(parent)
    3 : display pubsub type(parent)
AMQ8147: WebSphere MQ object  not found.
```

下一步做什么

您可以删除不再需要的任何流，队列和手动定义的通道。

发布/预订层次结构示例: 方案 *1*
使用具有队列管理器名称别名的点到点通道设置发布/预订层次结构拓扑。

关于此任务

这些方案以不同方式设置发布/预订层次结构，以建立队列管理器之间的连接。 这些方案都使用名为 QM1 的父队列管理器，以及名为 QM2 和 QM3 的两个子队列管理器。

方案 1 将拆分为更小的部分，以使过程更易于遵循。

方案 *1 part 1:* 创建队列管理器

过程

1. 使用以下命令创建并启动三个名为 QM1， QM2 和 QM3 的队列管理器:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
strmqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
strmqm QM3
```

2. 通过在所有三个队列管理器上使用以下命令来启用队列管理器发布/预订方式:

```
ALTER QMGR PSMODE(ENABLED)
```

方案 *1 part 2:* 点到点通道连接

关于此任务

使用与父队列管理器同名的队列管理器别名在队列管理器之间建立点到点通道连接。

过程

1. 在 QM2 到 QM1 上定义传输队列和队列管理器别名。 为 QM1 定义发送方通道，为 QM2 定义在 QM1 上创建的发送方通道的接收方通道:

```
DEFINE QLOCAL(QM1.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM1) RNAME('') RQMNAME(QM1) XMITQ(QM1.XMITQ)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(RCVR) TRPTYPE(TCP)
```

2. 在 QM3 到 QM1 上定义传输队列和队列管理器别名。 定义 QM1 的发送方通道以及在 QM1 上为 QM3 创建的发送方通道的接收方通道:

```
DEFINE QLOCAL(QM1.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM1) RNAME('') RQMNAME(QM1) XMITQ(QM1.XMITQ)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(RCVR) TRPTYPE(TCP)
```

3. 在 QM1 到 QM2 和 QM3 上定义传输队列和队列管理器别名。 定义到 QM2 和 QM3 的发送方通道，以及在 QM2 和 QM3 上为 QM1 创建的发送方通道的接收方通道:

```
DEFINE QLOCAL(QM2.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM2) RNAME('') RQMNAME(QM2) XMITQ(QM2.XMITQ)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(SDR) CONNAME('localhost(7777)') XMITQ(QM2.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)

DEFINE QLOCAL(QM3.XMITQ) USAGE(XMITQ)

DEFINE QREMOTE (QM3) RNAME('') RQMNAME(QM3) XMITQ(QM3.XMITQ)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(SDR) CONNAME('localhost(8888)') XMITQ(QM3.XMITQ)
TRPTYPE(TCP)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)
```

4. 在队列管理器上启动相应的侦听器:

```
runmqlsr -m QM1 -t TCP -p 9999 &
runmqlsr -m QM2 -t TCP -p 7777 &
runmqlsr -m QM3 -t TCP -p 8888 &
```

5. 启动以下通道:

   a. 在 QM1 上:

   ```
   START CHANNEL('QM1.TO.QM2')

   START CHANNEL('QM1.TO.QM3')
   ```

   b. 在 QM2 上:

   ```
   START CHANNEL('QM2.TO.QM1')
   ```

   c. 在 QM3 上:

   ```
   START CHANNEL('QM3.TO.QM1')
   ```

6. 检查所有通道是否已启动:

```
DISPLAY CHSTATUS('QM1.TO.QM2')

DISPLAY CHSTATUS('QM1.TO.QM3')
```

```
DISPLAY CHSTATUS('QM2.TO.QM1')
DISPLAY CHSTATUS('QM3.TO.QM1')
```

方案 *1* 部件 *3:* 连接队列管理器并定义主题

关于此任务

将子队列管理器 QM2 和 QM3 连接到父队列管理器 QM1。

过程

1. 在 QM2 和 QM3 上，将父队列管理器设置为 QM1:

```
ALTER QMGR PARENT (QM1)
```

2. 在所有队列管理器上运行以下命令以检查子队列管理器是否已连接到父队列管理器:

```
DISPLAY PUBSUB TYPE(ALL)
```

3. 定义主题对象:

```
define topic(FOOTBALL) TOPICSTR('Sport/Soccer')
```

方案 *1 part 4:* 发布和预订主题

关于此任务

使用 amqspub.exe 和 amqssub.exe 应用程序来发布和预订主题。

过程

1. 在第一个命令窗口中运行以下命令:

```
amqspub Sport/Soccer QM2
```

2. 在第二个命令窗口中运行此命令:

```
amqssub Sport/Soccer QM1
```

3. 在第三个命令窗口中运行以下命令:

```
amqssub Sport/Soccer QM3
```

结果

第二个和第三个命令窗口中的 amqssub.exe 应用程序接收在第一个命令窗口中发布的消息。

相关任务

第 72 页的『发布/预订层次结构示例: 方案 2』
使用具有与远程队列管理器相同的传输队列名称的点到点通道来设置发布/预订层次结构拓扑。

第 75 页的『发布/预订层次结构示例: 方案 3』
使用集群通道将队列管理器添加到层次结构拓扑。

发布/预订层次结构示例*: 方案 2*
使用具有与远程队列管理器相同的传输队列名称的点到点通道来设置发布/预订层次结构拓扑。

关于此任务

这些方案以不同方式设置发布/预订层次结构，以建立队列管理器之间的连接。 这些方案都使用名为 QM1 的父队列管理器，以及名为 QM2 和 QM3 的两个子队列管理器。

方案 2 将拆分为更小的部分，以使过程更易于遵循。 此方案复用 第 70 页的『发布/预订层次结构示例: 方案 1』 中的方案 1 部件 1 ，方案 1 部件 3 和方案 1 部件 4。

方案 *2 part 1:* 创建队列管理器并设置 *PSMODE*

过程

1. 使用以下命令创建并启动三个名为 QM1， QM2 和 QM3 的队列管理器:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
strmqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
strmqm QM3
```

2. 通过在所有三个队列管理器上使用以下命令来启用队列管理器发布/预订方式:

```
ALTER QMGR PSMODE(ENABLED)
```

方案 *2 part 2:* 点到点通道连接

关于此任务

使用与父队列管理器同名的传输队列在队列管理器之间建立点到点通道连接。

过程

1. 在 QM2 到 QM1 上定义传输队列。 为 QM1 上创建的 QM2 的发送方通道定义到 QM1 的发送方通道和接收方通道:

```
DEFINE QLOCAL(QM1) USAGE(XMITQ)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1) TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(RCVR) TRPTYPE(TCP)
```

2. 在 QM3 到 QM1 上定义传输队列。 定义 QM1 的发送方通道以及在 QM1 上为 QM3 创建的发送方通道的接收方通道:

```
DEFINE QLOCAL(QM1) USAGE(XMITQ)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(SDR) CONNAME('localhost(9999)') XMITQ(QM1) TRPTYPE(TCP)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(RCVR) TRPTYPE(TCP)
```

3. 在 QM1 到 QM2 和 QM3 上定义传输队列。 定义到 QM2 和 QM3 的发送方通道，以及在 QM2 和 QM3 上为 QM1 创建的发送方通道的接收方通道:

```
DEFINE QLOCAL(QM2) USAGE(XMITQ)

DEFINE CHANNEL('QM1.TO.QM2') CHLTYPE(SDR) CONNAME('localhost(7777)') XMITQ(QM2) TRPTYPE(TCP)

DEFINE CHANNEL('QM2.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)

DEFINE QLOCAL(QM3) USAGE(XMITQ)

DEFINE CHANNEL('QM1.TO.QM3') CHLTYPE(SDR) CONNAME('localhost(8888)') XMITQ(QM3) TRPTYPE(TCP)

DEFINE CHANNEL('QM3.TO.QM1') CHLTYPE(RCVR) TRPTYPE(TCP)
```

4. 在队列管理器上启动相应的侦听器:

```
runmqlsr -m QM1 -t TCP -p 9999 &
runmqlsr -m QM2 -t TCP -p 7777 &
runmqlsr -m QM3 -t TCP -p 8888 &
```

5. 启动以下通道:

   a. 在 QM1 上:

   ```
   START CHANNEL('QM1.TO.QM2')
   START CHANNEL('QM1.TO.QM3')
   ```

   b. 在 QM2 上:

   ```
   START CHANNEL('QM2.TO.QM1')
   ```

   c. 在 QM3 上:

   ```
   START CHANNEL('QM3.TO.QM1')
   ```

6. 检查所有通道是否已启动:

```
DISPLAY CHSTATUS('QM1.TO.QM2')
DISPLAY CHSTATUS('QM1.TO.QM3')
DISPLAY CHSTATUS('QM2.TO.QM1')
DISPLAY CHSTATUS('QM3.TO.QM1')
```

方案 *2* 部件 *3:* 连接队列管理器并定义主题

关于此任务

将子队列管理器 QM2 和 QM3 连接到父队列管理器 QM1。

过程

1. 在 QM2 和 QM3 上，将父队列管理器设置为 QM1:

```
ALTER QMGR PARENT (QM1)
```

2. 在所有队列管理器上运行以下命令以检查子队列管理器是否已连接到父队列管理器:

```
DISPLAY PUBSUB TYPE(ALL)
```

3. 定义主题对象:

```
define topic(FOOTBALL) TOPICSTR('Sport/Soccer')
```

方案 *2 part 4:* 发布和预订主题

关于此任务

使用 amqspub.exe 和 amqssub.exe 应用程序来发布和预订主题。

过程

1. 在第一个命令窗口中运行以下命令:

```
amqspub Sport/Soccer QM2
```

2. 在第二个命令窗口中运行此命令:

```
amqssub Sport/Soccer QM1
```

3. 在第三个命令窗口中运行以下命令:

```
amqssub Sport/Soccer QM3
```

结果

第二个和第三个命令窗口中的 amqssub.exe 应用程序接收在第一个命令窗口中发布的消息。

相关任务

第 70 页的『发布/预订层次结构示例: 方案 1』
使用具有队列管理器名称别名的点到点通道设置发布/预订层次结构拓扑。

第 75 页的『发布/预订层次结构示例: 方案 3』
使用集群通道将队列管理器添加到层次结构拓扑。

发布/预订层次结构示例*: 方案 3*
使用集群通道将队列管理器添加到层次结构拓扑。

关于此任务

这些方案以不同方式设置发布/预订层次结构，以建立队列管理器之间的连接。 这些方案都使用名为 QM1 的父队列管理器，以及名为 QM2 和 QM3 的两个子队列管理器。

方案 3 将拆分为更小的部分，以使过程更易于遵循。 此方案复用 第 70 页的『发布/预订层次结构示例: 方案 1』中的方案 1 部件 1，方案 1 部件 3 和方案 1 部件 4。

此方案创建名为 DEMO 的集群，其中 QM1 和 QM2 是完整存储库， QM3 是部分存储库。 队列管理器 QM1 是队列管理器 QM2 和 QM3 的父代。

方案 *2 part 1:* 创建队列管理器并设置 *PSMODE*

过程

1. 使用以下命令创建并启动三个名为 QM1， QM2 和 QM3 的队列管理器:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM1
strmqm QM1

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM2
strmqm QM2

crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM3
strmqm QM3
```

2. 通过在所有三个队列管理器上使用以下命令来启用队列管理器发布/预订方式:

```
ALTER QMGR PSMODE(ENABLED)
```

方案 *2 part 2:* 点到点通道连接

关于此任务

在集群的队列管理器之间建立点到点通道连接。

过程

1. 在 QM1 和 QM2 上，将 **REPOS** 参数设置为集群 DEMO 的名称:

```
ALTER QMGR REPOS(DEMO)
```

2. 在队列管理器上启动相应的侦听器:

```
runmqlsr -m QM1 -t TCP -p 9999 &
runmqlsr -m QM2 -t TCP -p 7777 &
runmqlsr -m QM3 -t TCP -p 8888 &
```

3. 在每个队列管理器上定义集群接收方通道:

   a. 在 QM1 上:

   ```
   DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(9999)')
   CLUSTER(DEMO)
   ```

   b. 在 QM2 上:

   ```
   DEFINE CHANNEL(TO.QM2) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(7777)')
   CLUSTER(DEMO)
   ```

   c. 在 QM3 上:

   ```
   DEFINE CHANNEL(TO.QM3) CHLTYPE(CLUSRCVR) TRPTYPE(TCP) CONNAME('localhost(8888)')
   CLUSTER(DEMO)
   ```

4. 定义集群中每个队列管理器上的完整存储库的集群发送方通道:

   a. 在 QM1 上:

   ```
   DEFINE CHANNEL(TO.QM2) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(7777)')
   CLUSTER(DEMO)
   ```

   b. 在 QM2 上:

   ```
   DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(9999)')
   CLUSTER(DEMO)
   ```

   c. QM3 可以具有到 QM1 或 QM2 上的完整存储库的集群发送方通道。 此示例定义到 QM1 的通道:

   ```
   DEFINE CHANNEL(TO.QM1) CHLTYPE(CLUSSDR) TRPTYPE(TCP) CONNAME('localhost(9999)')
   CLUSTER(DEMO)
   ```

方案 *2* 部件 *3:* 连接队列管理器并定义主题

关于此任务

将子队列管理器 QM2 和 QM3 连接到父队列管理器 QM1。

过程

1. 在 QM2 和 QM3 上，将父队列管理器设置为 QM1:

   ```
   ALTER QMGR PARENT (QM1)
   ```

2. 在所有队列管理器上运行以下命令以检查子队列管理器是否已连接到父队列管理器:

   ```
   DISPLAY PUBSUB TYPE(ALL)
   ```

3. 定义主题对象:

   ```
   define topic(FOOTBALL) TOPICSTR('Sport/Soccer')
   ```

方案 *2 part 4:* 发布和预订主题

关于此任务

使用 amqspub.exe 和 amqssub.exe 应用程序来发布和预订主题。

过程

1. 在第一个命令窗口中运行以下命令:

```
amqspub Sport/Soccer QM2
```

2. 在第二个命令窗口中运行此命令:

```
amqssub Sport/Soccer QM1
```

3. 在第三个命令窗口中运行以下命令:

```
amqssub Sport/Soccer QM3
```

结果

第二个和第三个命令窗口中的 amqssub.exe 应用程序接收在第一个命令窗口中发布的消息。

相关任务

第 70 页的『发布/预订层次结构示例: 方案 1』
使用具有队列管理器名称别名的点到点通道设置发布/预订层次结构拓扑。

第 72 页的『发布/预订层次结构示例: 方案 2』
使用具有与远程队列管理器相同的传输队列名称的点到点通道来设置发布/预订层次结构拓扑。

控制发布和预订的流

连接到分布式发布/预订拓扑的队列管理器共享公共联合主题空间。 您可以通过选择每个发布和预订是本地发布和预订还是全局发布和预订，控制拓扑中的发布和预订流。

本地发布和预订不会传播到发布者或订户所连接的队列管理器之外。

您可以控制通过将集群或层次结构中的队列管理器连接在一起而创建的主题空间的范围。 在发布/预订集群中，主题对象必须为 "集群"，或者所有元素都保持本地，并且发布或预订不起作用。

预订在与不同发布中的主题字符串匹配时，可以解析为不同的主题对象。 这些称为重叠主题。 与特定匹配项的发布相关联的主题对象提供主题属性，并确定例如订户是否要接收发布内容。

发布作用域

发布的作用域控制队列管理器是否将发布转发到远程队列管理器。 使用 **PUBSCOPE** 主题属性来管理发布的作用域。

如果未将发布转发至远程队列管理器，那么只有本地订户才会接收该发布。

PUBSCOPE 主题属性用于确定对特定主题进行的发布的作用域。 可以将属性设置为下列其中一个值:

QMGR
: 该出版物仅提供给本地订户。 这些出版物称为 本地出版物。 本地发布不会转发至远程队列管理器，因此连接至远程队列管理器的订户不会接收本地发布。

ALL
: 该发布将传递给本地订户以及连接到远程队列管理器的订户。 这些出版物称为 全局出版物。

ASPARENT
: 使用父代的 **PUBSCOPE** 设置。

发布者还可以使用 MQPMO_SCOPE_QMGR put 消息选项来指定发布是本地发布还是全局发布。 如果使用此选项，那么它将覆盖已使用 **PUBSCOPE** 主题属性设置的任何行为。

预订作用域

预订的作用域控制一个队列管理器上的预订是接收在发布/预订集群或层次结构中的另一个队列管理器上发布的发布，还是仅接收来自本地发布程序的发布。

将预订作用域限制为队列管理器将阻止代理预订转发到发布/预订拓扑中的其他队列管理器。 这将减少队列管理器之间的发布/预订消息传递流量。

SUBSCOPE 主题属性用于确定对特定主题进行的预订的作用域。 可以将属性设置为下列其中一个值:

QMGR
预订仅接收本地发布，并且不会将代理预订传播到远程队列管理器。

ALL
代理预订会传播至远程队列管理器，订户将接收本地和远程发布。

ASPARENT
使用父代的 **SUBSCOPE** 设置。

单个订户可以通过在创建预订时指定 MQSO_SCOPE_QMGR 预订选项来覆盖 ALL 的 **SUBSCOPE** 设置。 预订可以覆盖主题的 **SUBSCOPE** 设置 ALL。

注: 个别订户只能限制主题的 **SUBSCOPE** 。 当单个预订将 **SUBSCOPE** 设置为 ALL 时，该预订将采用匹配主题的 **SUBSCOPE** 设置。

组合发布和预订作用域
在 WebSphere MQ V 7 及更高版本中，发布和预订作用域独立工作以确定队列管理器之间的发布流。

发布可以流向在发布/预订拓扑中连接的所有队列管理器，也可以仅流向本地队列管理器。 同样适用于代理预订。 与预订匹配的发布由这两个流的组合管理。

出版物和预订都可以限定为 QMGR 或 ALL。 如果发布者和订户都连接到同一队列管理器，那么作用域设置不会影响订户从该发布者接收到的发布。

如果发布者和订户连接到不同的队列管理器，那么这两个设置都必须为 ALL 才能接收远程发布。

假设发布程序连接到不同的队列管理器。 如果希望订户接收来自任何发布者的发布，请将预订作用域设置为 ALL。 然后，对于每个发布者，您可以决定是否将其发布的范围限制为发布者本地的订户。

假设订户连接到不同的队列管理器。 如果您希望将来自发布者的发布发送到所有订户，请将发布范围设置为 ALL。 如果您希望订户仅从连接到同一队列管理器的发布程序接收发布，请将预订作用域设置为 QMGR。

在 V 6 和更低版本中，发布和预订作用域不仅管理所流动的发布。 此外，出版物的范围必须与订阅的范围相匹配。

示例: 足球结果服务

假设你是足球联赛的成员球队。 每个团队都有一个队列管理器连接到发布/预订集群中的所有其他团队。

各代表队使用主题 Football/result/*Home team name/Away team name* 公布在主场进行的所有比赛的结果。 斜体字中的字符串是变量主题名称，而发布是匹配的结果。

每个俱乐部还使用主题字符串 Football/myteam/*Home team name/Away team name* 重新发布仅针对俱乐部的结果。

这两个主题都将发布到整个集群。

联盟设置了以下订阅，以便任何球队的球迷都可以通过三种有趣的方式订阅结果。

请注意，您可以使用 SUBSCOPE(QMGR)设置集群主题。 主题定义将传播到集群的每个成员，但预订的作用域只是本地队列管理器。 因此，每个队列管理器上的订户从同一预订接收不同的发布。

接收所有结果

```
DEFINE TOPIC(A) TOPICSTR('Football/result/') CLUSTER SUBSCOPE(ALL)
```

接收所有主目录结果

```
DEFINE TOPIC(B) TOPICSTR('Football/result/') CLUSTER SUBSCOPE(QMGR)
```

由于预订具有 QMGR 作用域，因此仅会匹配在主目录中发布的结果。

接收我的所有团队结果

```
DEFINE TOPIC(C) TOPICSTR('Football/myteam/') CLUSTER SUBSCOPE(QMGR)
```

由于预订具有 QMGR 作用域，因此仅匹配本地重新发布的本地团队结果。

主题空间

主题空间是您可以预订的主题集。 连接到分布式发布/预订拓扑中队列管理器的订户具有一个主题空间，该主题空间可能包含在已连接的队列管理器上定义的主题。

最初以管理方式创建主题，在您定义主题对象或持久预订时，或者在应用程序动态创建发布或预订时，以动态方式创建主题。

主题通过代理预订和通过创建管理集群主题对象传播到其他队列管理器。 代理预订导致发布从发布程序所连接的队列管理器转发到订户的队列管理器。 代理预订是将不同队列管理器上定义的主题组合到公共主题空间中的机制。

代理预订将在通过队列管理器层次结构中的父子关系连接在一起的所有队列管理器之间传播。 结果是，您可以在一个队列管理器上预订层次结构中任何其他队列管理器上定义的主题。 只要队列管理器之间存在已连接的路径，那么队列管理器的连接方式就无关紧要。

还会在集群的所有成员之间针对 *cluster* 主题传播代理预订。 集群主题是附加到具有 **CLUSTER** 属性或从其父代继承该属性的主题对象的主题。 不是集群主题的主题称为本地主题，不会复制到集群。 不会将任何代理预订从预订传播到集群到本地主题。

总之，在两种情况下会为订户创建代理预订。

1. 队列管理器是层次结构的成员，代理预订将转发给队列管理器的父代和子代。
2. 队列管理器是集群的成员，预订主题字符串解析为与集群主题对象关联的主题。 代理预订将转发到集群的所有成员。 请参阅 第 86 页的『重叠主题』，以获取有关并发症的更多信息。

如果队列管理器是集群和层次结构的成员，那么代理预订由两种机制传播，而不会将重复的发布传递给订户。

创建集群主题对象的效果是双重的。 当预订解析为集群主题时，会将对主题的代理预订发送到集群的其他成员。 它还会将主题对象的副本发送到集群的其他成员。 转发集群主题对象的作用是简化主题管理。 通常，集群主题对象是在集群中的单个队列管理器 (称为集群主题主机) 上定义的。

以下列表中描述了三个发布/预订拓扑的主题空间:

- 第 79 页的『案例 1。 发布/预订集群』.
- 第 80 页的『案例 2。 版本 7 中的发布/预订层次结构』.
- 第 80 页的『案例 3。 版本 6 中的发布/预订层次结构和流』.

在单独的主题中，以下任务描述了如何组合主题空间。

- 第 81 页的『在发布/预订集群中创建单个主题空间』.
- 第 81 页的『将 V 7 队列管理器添加到现有 V 6 主题空间』.
- 第 82 页的『组合多个集群的主题空间』.
- 第 83 页的『组合和隔离多个集群中的主题空间』
- 第 85 页的『发布和预订多个集群中的主题空间』

案例 1。 发布/预订集群

在此示例中，假定队列管理器 未 连接到发布/预订层次结构。

如果队列管理器是发布/预订集群的成员，那么其主题空间由本地主题和集群主题组成。 本地主题与没有 **CLUSTER** 属性的主题对象相关联。 如果队列管理器具有本地主题对象定义，那么其主题空间与集群中另一个也具有其自己的本地定义主题对象的队列管理器不同。

在发布/预订集群中，无法预订另一个队列管理器上定义的主题，除非您预订的主题解析为集群主题对象。

将解析集群中其他位置定义的集群主题的冲突定义，以支持最新定义。 在任何时间点，如果已多次定义集群主题，那么不同队列管理器上的集群主题定义可能不同。

主题对象的本地定义 (无论该定义是针对集群主题还是本地主题) 优先于集群中其他位置定义的相同主题对象。 使用本地定义的主题，即使在其他位置定义的对象是最新的。

将 **PUBSCOPE** 和 **SUBSCOPE** 选项设置为 QMGR，以防止集群主题上的发布或预订流向集群中的不同队列管理器。

假设您在集群主题主机上使用主题字符串 USA/Alabama 定义集群主题对象 Alabama 。 结果如下:

1. 现在，集群主题主机上的主题空间包含集群主题对象 Alabama 和主题 USA/Alabama。
2. 集群主题对象 Alabama 将复制到集群中与每个队列管理器上的主题空间组合在一起的所有队列管理器。 集群中每个队列管理器上发生的情况取决于队列管理器上是否存在主题对象 Alabama 。
   - 如果 Alabama 是新主题对象，那么队列管理器会将集群主题对象 Alabama 和主题 USA/Alabama 添加到其主题空间。
   - 如果 Alabama 是本地定义，那么将添加集群主题对象 Alabama 。 除非删除本地定义，否则将忽略远程定义的集群主题对象。 队列管理器将保留这两个定义。
   - 如果 Alabama 是在其他位置定义的较旧的集群主题对象，那么会将其替换为较新的集群主题对象。
3. 应用程序或管理员可以在集群中的任何位置通过引用 Alabama 主题对象来创建对 USA/Alabama 的预订。
4. 在集群中的任何位置，直接使用主题字符串 USA/Alabama 的应用程序可以创建继承主题对象 Alabama 的属性的预订。 Alabama 主题对象由从以 USA/Alabama 开头的任何主题字符串构成的预订继承。

   如果在另一个队列管理器上存在另一个 Alabama 主题对象定义，那么它优先于集群主题主机上的定义。 本地对象可能具有集群属性，也可能没有集群属性。 集群属性可能引用同一个集群或另一个集群。 尽量避免这些多定义案例。 它们导致行为上的差异。
5. 如果主题对象 Alabama 具有 **PUBSCOPE** 属性 ALL，那么解析为 Alabama 的预订将发送到集群中的所有其他队列管理器。

   将 Alabama **PUBSCOPE** 属性设置为 QMGR ，以防止发布从发布程序流向连接到集群中不同队列管理器的订户。

   Alabama 主题对象将复制到集群中的每个队列管理器，因此 **PUBSCOPE** 和 **PUBSCOPE** 属性将应用于集群中的所有队列管理器。

集群主题对象与集群中所有位置的相同主题字符串相关联很重要。 不能修改与主题对象关联的主题字符串。 要将同一主题对象与另一主题字符串相关联，必须删除该主题对象，并使用新的主题字符串重新创建该主题对象。 如果主题是集群的，那么效果是删除存储在集群的其他成员上的主题对象的副本，然后在集群中的所有位置创建新主题对象的副本。 主题对象的副本都引用同一个主题字符串。

但是，您可以使用不同的主题字符串在集群中的另一个队列管理器上创建主题对象的重复定义。 始终尝试通过在一个队列管理器上管理集群主题主机来避免重复。 请参阅 第 60 页的『多个集群主题定义』 ，以获取有关此要点的更多信息。 具有不同主题字符串的同一主题对象的多个定义可能会产生不同的结果，具体取决于引用主题的方式和位置。

案例 2。 版本 7 中的发布/预订层次结构

在此示例中，假定队列管理器 不是 发布/预订集群的成员。

在 V 7 中，如果队列管理器是发布/预订层次结构的成员，那么其主题空间由本地定义的所有主题以及在已连接的队列管理器上定义的所有主题组成。 层次结构中所有队列管理器的主题空间都相同。 没有将主题划分为本地主题和集群主题。

将 **PUBSCOPE** 和 **SUBSCOPE** 选项中的任一选项设置为 QMGR，以防止主题上的发布从发布者流向连接到层次结构中不同队列管理器的订户。

假设您在队列管理器 QMA 上使用主题字符串 USA/Alabama 定义主题对象 Alabama 。 结果如下:

1. QMA 上的主题空间现在包含主题对象 Alabama 和主题字符串 USA/Alabama。
2. 应用程序或管理员可以使用主题对象名称 Alabama 在 QMA 上创建预订。
3. 应用程序可以在层次结构中的任何队列管理器上创建对任何主题 (包括 USA/Alabama) 的预订。 如果未在本地定义 QMA ，那么主题 USA/Alabama 将解析为主题对象 SYSTEM.BASE.TOPIC。

案例 3。 版本 6 中的发布/预订层次结构和流

在 V 7 之前，主题空间分为不同的流，其中包括所有队列管理器上存在的缺省流。 发布不能在不同流之间流动。 如果使用指定的流，那么不同队列管理器上的主题空间可能不同。 主题分为缺省流中的主题和不同命名流中的主题。

注: 每个指定的流构成一个单独的主题空间。 要构成已连接的拓扑，每个指定的流都必须存在于已连接的队列管理器上。 假设流 X 是在 QMA 和 QMC 上定义的，而不是在 QMB 上定义的。 如果 QMA 是 QMB 的父代，而 QMB 是 QMC 的父代，那么流 X 中的任何主题都不能在 QMA 和 QMC 之间流动。

将 **PUBSCOPE** 和 **SUBSCOPE** 选项都设置为 QMGR 或 ALL 要求主题的发布者和订户仅将发布内容交换为本地使用，或仅将发布内容交换为全局使用。

从 V 7 开始，流不可使用发布/预订 API。 如果在 V 7 队列管理器上使用排队的发布/预订，那么会将流映射到可模拟流效果的不同主题对象。 通过创建作为流中所有主题的根主题的主题对象来模拟流。 队列管理器在流与每个树的相应根主题之间映射发布和预订。

组合主题空间
将队列管理器的主题空间与发布/预订集群或层次结构中的其他队列管理器组合在一起。 将发布/预订集群以及发布/预订集群与层次结构组合在一起。

您可以使用 **CLUSTER**， **PUBSCOPE** 和 **SUBSCOPE** 属性，发布/预订集群以及发布/预订层次结构的构建块来创建不同的发布/预订主题空间。

从从单个队列管理器扩展到发布/预订集群的示例开始，以下场景说明了不同的发布/预订拓扑。

在发布/预订集群中创建单个主题空间
扩展发布/预订系统以在多个队列管理器上运行。 使用发布/预订集群为每个发布者和订户提供单个相同的主题空间。

开始之前
您已在单个版本 7 队列管理器上实现发布/预订系统。

始终使用自己的根主题创建主题空间，而不是依赖继承 SYSTEM.BASE.TOPIC 的属性。 如果将发布/预订系统扩展至集群，那么可以在集群主题主机上将根主题定义为集群主题，然后在整个集群中共享所有主题。

关于此任务
现在，您希望扩展系统以支持更多发布者和订户，并在整个集群中显示每个主题。

过程
1. 创建要与发布/预订系统配合使用的集群。
   如果您具有现有传统集群，那么出于性能原因，最好为新的发布预订系统设置新集群。 可以将相同的服务器用于两个集群的集群存储库
2. 选择一个队列管理器 (可能是其中一个存储库) 作为集群主题主机。
3. 确保要在整个发布/预订集群中显示的每个主题都解析为管理主题对象。
   设置用于命名发布/预订集群的 **CLUSTER** 属性。

下一步做什么
将发布程序和订户应用程序连接到集群中的任何队列管理器。

创建具有 **CLUSTER** 属性的管理主题对象。 还会在整个集群中传播这些主题。 发布程序和订户程序使用管理主题，以便不会通过连接到集群中的不同队列管理器来改变其行为

如果需要 SYSTEM.BASE.TOPIC 在每个队列管理器上充当集群主题，那么需要在每个队列管理器上对其进行修改。

将 *V 7* 队列管理器添加到现有 *V 6* 主题空间
扩展现有版本 6 发布/预订系统以与版本 7 队列管理器进行互操作，共享相同的主题空间。

开始之前
您具有现有版本 6 发布/预订系统。

您已在新服务器上安装 WebSphere MQ V 7 并配置了队列管理器。

关于此任务

您想要扩展现有版本 6 发布/预订系统以使用版本 7 队列管理器。

您已决定稳定开发使用排队的发布/预订接口的版本 6 发布/预订系统。 您打算使用 V 7 MQI 向系统添加扩展。 您现在没有重写已排队的发布/预订应用程序的计划。

您打算将来将 V 6 队列管理器升级到 V 7。 现在，您将继续在 V 7 队列管理器上运行现有已排队的发布/预订应用程序。

过程

1. 创建一组发送方/接收方通道，以将 V 7 队列管理器与 V 6 队列管理器中的一个双向连接。
2. 使用目标队列管理器的名称创建两个传输队列。 如果由于某种原因无法使用目标队列管理器的名称作为传输队列名称，请使用队列管理器别名。
3. 配置传输队列以触发发送方通道。
4. 如果 V 6 发布/预订系统使用流，请将流添加到 V 7 队列管理器，如 添加流 中所述。
5. 检查 V 7 queue manager **PSMODE** 是否设置为 ENABLE。
6. 更改其 **PARENT** 属性以引用其中一个版本 6 队列管理器。
7. 检查队列管理器之间的父子关系在两个方向上都处于活动状态。

下一步做什么

完成该任务后， V 6 和 V 7 队列管理器将共享相同的主题空间。 例如，您可以执行以下所有任务。

- 在 V 6 和 V 7 队列管理器之间交换发布和预订。
- 在 V 7 队列管理器上运行现有版本 6 发布/预订程序。
- 在 V 6 或 V 7 队列管理器上查看和修改主题空间。
- 编写 V 7 发布/预订应用程序，并在 V 7 队列管理器上运行这些应用程序。
- 使用 V 7 应用程序创建新的发布和预订，并与 V 6 应用程序交换这些发布和预订。

组合多个集群的主题空间

创建跨多个集群的主题空间。 发布到一个集群中的主题，然后在另一个集群中预订该主题。

开始之前

您有现有的发布/预订集群，并且要将一些集群主题传播到所有集群中。

关于此任务

要将发布从一个集群传播到另一个集群，您需要在层次结构中将这些集群连接在一起; 请参阅 第 83 页的图 35。 分层连接在已连接的队列管理器之间传播预订和发布，集群在每个集群中传播集群主题，而不是在集群之间传播集群主题。

这两种机制的组合将在所有集群之间传播集群主题。 您需要在每个集群中重复集群主题定义。

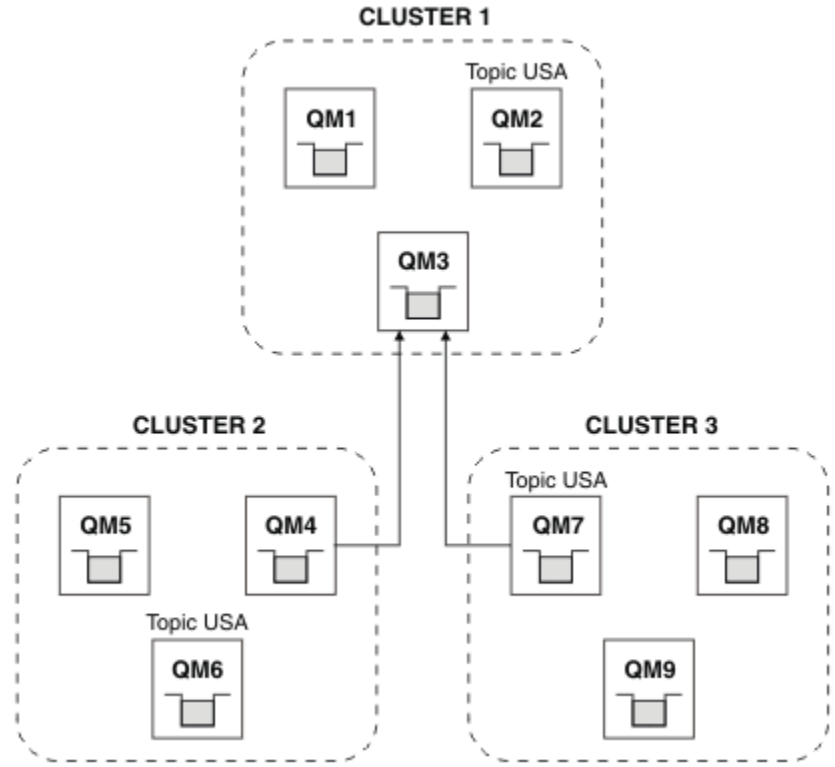


图 *35:* 使用层次结构连接集群

以下步骤将集群连接到层次结构中。

过程

1. 创建两组发送方/接收方通道以双向连接 QM3 和 QM4 以及 QM3 和 QM7。 必须使用传统发送方/接收方通道和传输队列 (而不是集群) 来连接层次结构。
2. 使用目标队列管理器的名称创建三个传输队列。 如果由于某种原因无法使用目标队列管理器的名称作为传输队列名称，请使用队列管理器别名。
3. 配置传输队列以触发发送方通道。
4. 检查 QM3 的 **PSMODE** ， QM4 和 QM7 是否设置为 ENABLE。
5. 将 QM4 和 QM7 的 **PARENT** 属性变更为 QM3。
6. 检查队列管理器之间的父子关系在两个方向上都处于活动状态。
7. 在集群 1， 2 和 3 中的三个集群主题主机上，使用属性 **CLUSTER**('CLUSTER 1')， **CLUSTER**('CLUSTER 2') 和 **CLUSTER**('CLUSTER 3') 创建管理主题 USA 。 集群主题主机不需要是分层连接的队列管理器。

下一步做什么

现在，您可以在 第 83 页的图 35 中发布或预订集群主题 USA 。 发布预订流向所有三个集群中的发布者和订户。

假定您未在其他集群中创建 USA 作为集群主题。 如果仅在 QM7 上定义 USA ，那么将在 QM7， QM8， QM9 和 QM3 之间交换 USA 的发布和预订。 在 QM7 QM8 上运行的发布者和订户 QM9 将继承管理主题 USA 的属性。 QM3 上的发布者和订户将继承 QM3 上的 SYSTEM.BASE.TOPIC 的属性。

组合和隔离多个集群中的主题空间
将某些主题空间与特定集群隔离，并组合其他主题空间以使它们在所有已连接的集群中都可访问。

开始之前

检查主题 第 82 页的『组合多个集群的主题空间』。 这可能足以满足您的需求，而无需添加额外的队列管理器作为网桥。

关于此任务

第 82 页的『组合多个集群的主题空间』 中的 第 83 页的图 35 中显示的拓扑的潜在改进是隔离未在所有集群之间共享的集群主题。 通过创建不在任何集群中的桥接队列管理器来隔离集群; 请参阅 第 84 页的图 36。 使用桥接队列管理器来过滤哪些发布和预订可以从一个集群流向另一个集群。

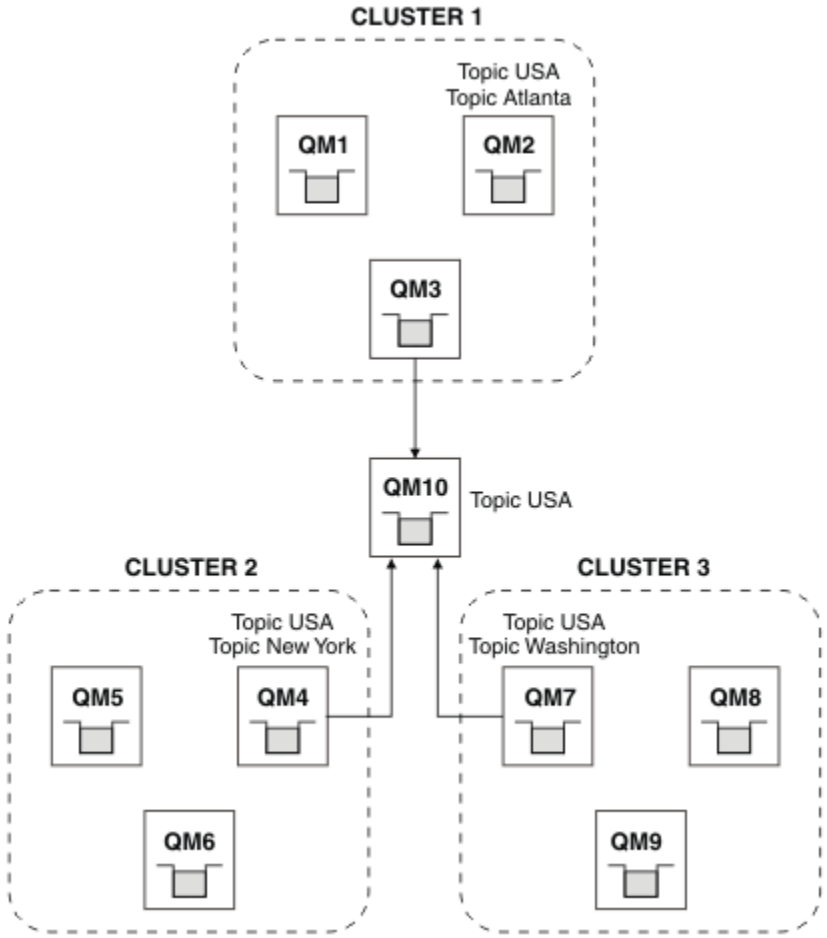


图 *36:* 桥接集群

使用网桥来隔离您不希望在其他集群上的网桥中公开的集群主题。 在 第 84 页的图 36 中， USA 是所有集群中共享的集群主题， Atlanta， New York 和 Washington 是仅在一个集群中共享的集群主题。

使用以下过程对配置进行建模:

过程

1. 将所有 SYSTEM.BASE.TOPIC 主题对象修改为在所有队列管理器上具有 **SUBSCOPE**(QMGR) 和 **PUBSCOPE**(QMGR)。

   除非在集群主题的根主题上显式设置 **SUBSCOPE**(ALL) 和 **PUBSCOPE**(ALL)， 否则不会将任何主题 (甚至集群主题) 传播到其他队列管理器。
2. 定义要在每个集群中与属性 **CLUSTER**(*clustername*)， **SUBSCOPE**(ALL) 和 **PUBSCOPE**(ALL) 共享的三个集群主题主机上的主题。

   如果要在所有集群之间共享某些集群主题，请在每个集群中定义相同的主题。 使用每个集群的集群名称作为集群属性。

3. 对于要在所有集群之间共享的集群主题，请在网桥队列管理器 (QM10) 上使用属性 **SUBSCOPE**(ALL) 和 **PUBSCOPE**(ALL) 再次定义这些主题。

示例

在 第 84 页的图 36 中的示例中，仅从 USA 继承的主题在所有三个集群之间传播。

下一步做什么

在集群之间传播使用 **SUBSCOPE**(ALL) 和 **PUBSCOPE**(ALL) 在网桥队列管理器上定义的主题的预订。

将在每个集群中传播具有属性 **CLUSTER**(*clustername*) ， **SUBSCOPE**(ALL) 和 **PUBSCOPE**(ALL) 的每个集群中定义的主题的预订。

任何其他预订都是队列管理器的本地预订。

发布和预订多个集群中的主题空间

使用重叠的集群发布和预订多个集群中的主题。 只要集群中的主题空间不重叠，就可以使用此方法。

开始之前

在集群之间的交集中使用一些队列管理器创建多个传统集群。

关于此任务

由于各种不同的原因，您可能选择了重叠集群。

1. 您有数量有限的高可用性服务器或队列管理器。 您决定将所有集群存储库以及集群主题主机部署到这些存储库。
2. 您具有使用网关队列管理器进行连接的现有传统队列管理器集群。 您希望将发布/预订应用程序部署到同一集群拓扑。
3. 您有几个自包含的发布/预订应用程序。 出于性能原因，最好保持发布/预订集群较小且与传统集群分开。 您已决定将应用程序部署到不同的集群。 但是，您还希望在一个队列管理器上监视所有发布/预订应用程序，因为您只许可了该监视应用程序的一个副本。 此队列管理器必须有权访问所有集群中的集群主题的发布。

通过确保在非重叠主题空间中定义主题，可以将主题部署到重叠的发布/预订集群中，请参阅 第 85 页的图 37。 如果主题空间重叠，那么部署到重叠的集群会导致问题。

由于发布/预订集群重叠，因此您可以使用重叠中的队列管理器来发布和预订任何主题空间。

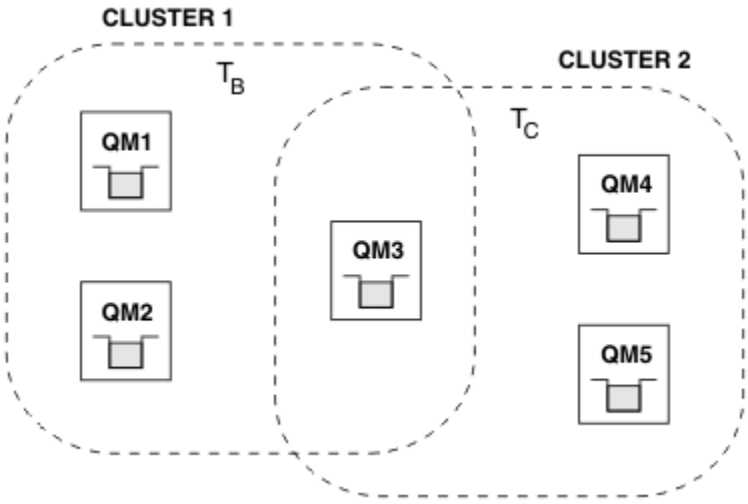


图 *37:* 重叠集群，非重叠主题空间

过程

创建确保主题空间不重叠的方法。

例如，为每个主题空间定义唯一的根主题。 使根主题成为集群主题。

a) `DEFINE TOPIC(B) TOPICSTR('B') CLUSTER('CLUSTER 1') ...`

b) `DEFINE TOPIC(C) TOPICSTR('C') CLUSTER('CLUSTER 2') ...`

示例

在 第 85 页的图 37 中，连接到 QM3 的发布者和订户可以发布或预订 T_B 或 T_C

下一步做什么

将两个集群中使用主题的发布者和订户连接到重叠中的队列管理器。

将只能使用特定集群中的主题的发布者和订户连接到不在重叠中的队列管理器。

重叠主题
根据分布式发布/预订拓扑，发布和预订主题字符串，当发布可以与不同的主题对象相关联时，会发生重叠主题。

如果可以将主题解析为多个主题对象，那么必须考虑主题之间的重叠。

集群中的本地主题
可以在集群中的任何队列管理器上定义主题。 如果该主题是在本地定义的，那么它优先于在其他位置定义并解析为同一主题字符串的集群主题。

集群中的集群主题
可以在集群中的任何队列管理器上定义主题。 如果主题已集群化，那么会将其复制到集群的其他成员。 如果在集群中的另一个队列管理器上将该主题定义为集群主题，那么将发生错误。 将错误消息写入具有现有集群定义的队列管理器的错误日志。

作为规则，仅在集群中的一个队列管理器 ("集群主题主机") 上定义集群主题，以确保集群主题只有一个定义。

如果重新定义集群主题，那么更改需要时间才能到达每个队列管理器。 最终，最新定义将覆盖已复制到非集群主题主机的较早集群主题定义。

如果在具有不同属性的集群中的多个队列管理器上定义集群主题，那么最新定义不会覆盖任何先前的本地定义。

通配符预订解析为多个主题字符串

当预订包含通配符时，主题空间中可能有不同的主题可以与预订匹配，并导致预订解析为不同的主题对象。

例如，请考虑集群 SPORTS 中的以下主题定义。

```
DEFINE TOPIC(A) TOPICSTR('Football/result/#') SUBSCOPE(QMGR) CLUSTER(SPORTS)
DEFINE TOPIC(B) TOPICSTR('Football/#') SUBSCOPE(ALL) CLUSTER(SPORTS)
DEFINE TOPIC(C) TOPICSTR('Football/result/Newport/Cardiff') PUBSCOPE(ALL) SUBSCOPE(ALL)
CLUSTER(SPORTS)
DEFINE TOPIC(D) TOPICSTR('Football/matches/Newport/Cardiff') PUBSCOPE(ALL) SUBSCOPE(QMGR)
CLUSTER(SPORTS)
```

假设集群中有两个队列管理器 QM1 和 QM2 。 主题 C 和 D 发布于 QM1。

请考虑 QM2 上的订户接收的内容 (如果这些预订未分组)。

- 对主题 A 的预订不接收任何内容。
  - SUBSCOPE(QMGR)，发布在另一个队列管理器上。
- 对主题 B 的预订将同时接收这两个发布。
  - 在这两种情况下都为 SUBSCOPE(ALL) 和 PUBSCOPE(ALL) 。
- 主题 C 的预订接收到一个发布。

  - SUBSCOPE(ALL) 和 PUBSCOPE(ALL)，以及与主题 C 上的出版物的匹配项。
- 对主题 D 的预订不接收任何内容。
  - SUBSCOPE(QMGR)，发布在另一个队列管理器上。

如果这些预订已分组，请考虑 QM2 上的订户接收的内容。

- 订户在主题 C 上接收到一个发布内容。
  - 具有 SUBSCOPE(QMGR) 的主题 A 上的匹配预订将被具有 SUBSCOPE(ALL)的主题 C 上的匹配预订覆盖。 更具体的订阅获胜，将收到该出版物。
  - 将拒绝主题 B 上的匹配预订以支持主题 C 上的匹配预订，因为这些预订已分组，并且 C 更具体。 将废弃重复的发布。
- 订户未收到有关主题 D 的发布
  - 具有 SUBSCOPE(ALL) 的主题 B 上的匹配预订将被具有 SUBSCOPE(QMGR)的主题 D 上的匹配预订覆盖。 更具体的订阅获胜，发布将被废弃。

循环检测的工作方式

在分布式发布/预订网络中，发布和代理预订不能循环很重要，因为这将导致网络泛滥，连接的订户接收同一原始发布的多个副本。

第 44 页的『代理预订聚集和发布聚集』 中描述的代理预订聚集系统不会阻止形成循环，尽管它会阻止代理预订的永久循环。 由于发布内容的传播由代理预订的存在确定，因此，它们可能形成永久回路。 Websphere MQ V7.0 使用以下技术来防止发布内容形成永久回路：

当发布内容在发布/预订拓扑中移动时，每个队列管理器都将对消息头添加唯一的指纹。 每当发布/预订队列管理器接收到来自另一发布/预订队列管理器的发布内容时，都将检查消息头中的指纹。 如果它自己的指纹已存在，那么表明该发布内容已在回路中完整巡回，因此该队列管理器将废弃该消息并在错误日志中添加条目。

注: 在回路中，发布内容将沿回路双向传播，回路中的每个队列管理器在始发队列管理器废弃回路的发布内容之前，都将接收到这两份发布内容。 这将导致预订应用程序在回路中断前接收到发布内容的重复副本。

回路检测指纹格式

循环检测指纹作为 V7.0 协议的一部分插入到 RFH2 头或流中。 RFH2 程序员需要了解此头原封不动地传递指纹信息。 WebSphere MessageBroker 使用不会包含指纹信息的 RFH1 头。

```
<ibm>
  <Rfp>uuid1</Rfp>
  <Rfp>uuid2</Rfp>
  <Rfp>uuid3</Rfp>
  . . .
</ibm>
```

<ibm> 是一个文件夹的名称，此文件夹存放路由指纹的列表，这些路由指纹包含已访问的每个队列管理器的唯一用户标识 (UUID)。

队列管理器每次发布消息时，都使用 <Rfp>（路由指纹）标记将其 UUID 添加到 <ibm> 文件夹中。 Whenever a publication is received, WebSphere MQ uses the message properties API to iterate through the <Rfp> tags to see if that particular uuid value is present. 由于 WebSphere MQ 的 WebSphere Platform Messaging 组件在使用排队的发布/预订接口时通过通道和 RFH2 预订连接到 Websphere Message Broker 的方式，因此 WebSphere MQ 在通过该路由接收发布时也会创建指纹。

目标是，如果应用程序不需要任何 RFH2，那么不要仅仅由于已添加指纹信息而将任何 RFH2 传递给应用程序。

Whenever an RFH2 is converted into message properties, it will also be necessary to convert the <ibm> folder; this removes the fingerprint information from the RFH2 that is passed on or delivered to applications that have used the Websphere MQ V7.0 API.

每当将具有指纹信息的消息传递到 RFH1 订户或传递到 Websphere Message Broker V6.0 时，都会将指纹信息转换为 RFH1。

当 Websphere Message Broker V6.0 将此消息传递给 RFH2 订户 (例如 SIB) 时，它必须将指纹信息转换回 RFH2 格式。

JMS 应用程序看不到指纹信息，因为 JMS 接口不会从 RFH2 中抽取该信息，因此不会将其交给其应用程序。

Rfp 消息属性是使用 propDesc.CopyOptions = MQCOPY_FORWARD and MQCOPY_PUBLISH 创建的。这将对接收并重新发布同一消息的应用程序产生影响。 这意味着此类应用程序可以使用 PutMsgOpts.Action = MQACTP_FORWARD 继续路由指纹的链，但必须进行适当编码以从链中除去其自己的指纹。 缺省情况下，应用程序使用 PutMsgOpts.Action = MQACTP_NEW 并启动新链。

分布式发布/预订拓扑中的保留发布

在分布式发布/预订拓扑中使用保留发布时，最佳实践是仅从拓扑中的单个队列管理器发布同一主题上的保留发布。

否则，不同的保留发布可能在同一主题的不同队列管理器上处于活动状态，从而导致意外行为。 由于分发了多个代理预订，因此可能会接收到多个保留发布。

队列管理器之间的发布/预订安全性

使用正常通道安全规则将发布/预订内部消息 (例如代理预订和发布) 放入发布/预订系统队列。 本主题中的信息和图突出显示了传递这些消息所涉及的各种流程和用户标识。

本地访问控制

对发布和预订主题的访问权由 发布/预订安全性中描述的本地安全性定义和规则进行管理。 在 z/OS 上，不需要本地主题对象来建立访问控制。 其他平台上的访问控制也不需要本地主题。 管理员可以选择将访问控制应用于集群主题对象，而不管它们是否存在于集群中。

系统管理员负责对其本地系统进行访问控制。 他们必须信任层次结构或集群集合体的其他成员的管理员对其访问控制策略负责。 由于为每台单独的机器定义了访问控制，因此如果需要精细的级别控制，可能会造成负担。 可能不需要强制实施任何访问控制，或者可以在主题树中的高级对象上定义访问控制。 可以为主题名称空间的每个子部分定义精细级别访问控制。

进行代理预订

组织将其队列管理器连接到队列管理器的信任由正常通道认证方法确认。 如果还允许该可信组织执行分布式发布/预订，那么将执行权限检查。 当通道将消息放入分布式发布/预订队列时，将执行此检查。 例如，如果将消息放入 SYSTEM.INTER.QMGR.CONTROL 队列。 队列权限检查的用户标识取决于接收通道的 PUTAUT 值。 例如，通道的用户标识 MCAUSER，消息上下文，具体取决于值和平台。 有关通道安全性的更多信息，请参阅 通道安全性。

使用远程队列管理器上分布式发布/预订代理程序的用户标识进行代理预订。 例如， 第 89 页的图 38 中的 QM2 。 然后，很容易授予用户对本地主题对象概要文件的访问权，因为该用户标识是在系统中定义的，因此不存在域冲突。

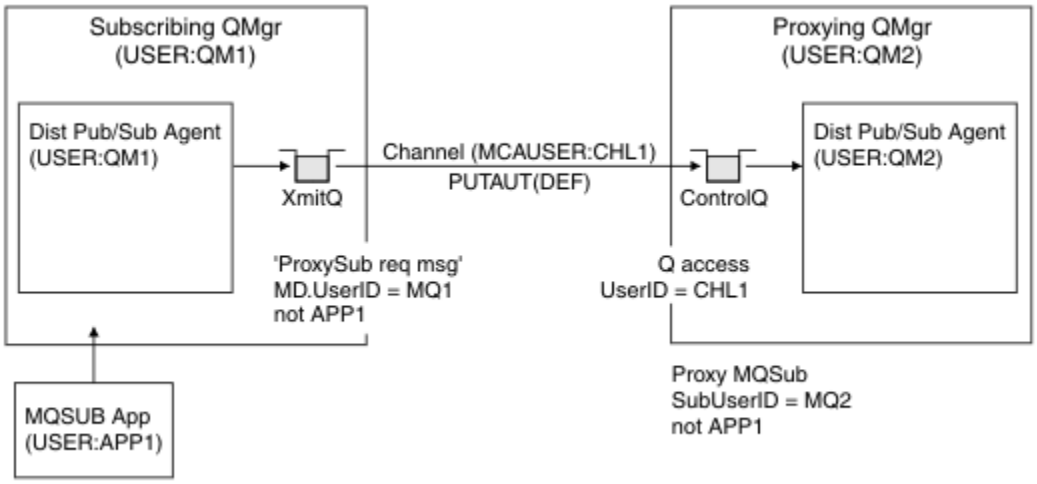


图 *38:* 代理预订安全性，进行预订

发送回远程发布

在发布队列管理器上创建发布时，将为任何代理预订创建发布的副本。 复制的发布的上下文包含进行预订的用户标识的上下文; QM2 位于 第 89 页的图 39 中。 使用作为远程队列的目标队列创建代理预订，因此会将发布消息解析到传输队列。

组织将其队列管理器 QM2 连接到另一个队列管理器 QM1 的信任已通过正常通道认证方法确认。 如果随后允许该可信组织执行分布式发布/预订，那么当通道将发布消息放入分布式发布/预订发布队列 SYSTEM.INTER.QMGR.PUBS 时，将执行权限检查。 队列权限检查的用户标识取决于接收通道的 PUTAUT 值 (例如，通道的用户标识， MCAUSER，消息上下文等，具体取决于值和平台)。 有关通道安全性的更多信息，请参阅 通道安全性。

当发布消息到达预订队列管理器时，将在该队列管理器的权限下完成对主题的另一个 MQPUT，并且消息的上下文将替换为每个本地订户的上下文，因为每个订户都具有该消息。

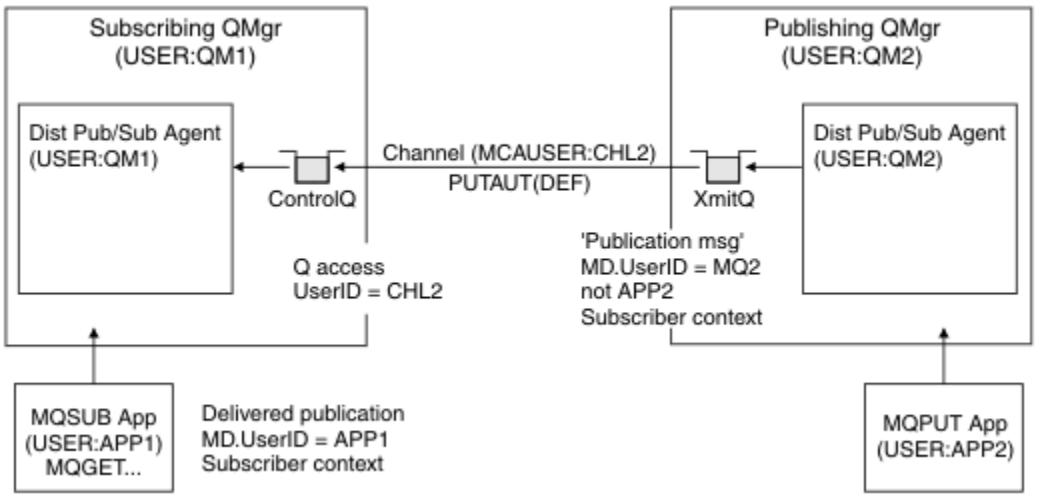


图 *39:* 代理预订安全性，转发发布

在很少考虑安全性的系统上，分布式发布/预订进程可能在 mqm 组中的用户标识下运行，通道上的 MCAUSER 参数为空 (缺省值)，并且根据需要将消息传递到各种系统队列。 无安全保护的系统便于设置概念证明以演示分布式发布/订阅。

在更认真考虑安全性的系统上，这些内部消息受与通过通道的任何消息相同的安全性控制。

如果使用非空白 MCAUSER 和指定必须检查 MCAUSER 的 PUTAUT 值来设置通道，那么必须授予所讨论的 MCAUSER 对 SYSTEM.INTER.QMGR.* 队列的访问权。 如果存在多个不同的远程队列管理器，并且通道在

不同的 MCAUSER 标识下运行，那么需要授予所有这些用户标识对 SYSTEM.INTER.QMGR.* 队列的访问权。 例如，在单个队列管理器上配置了多个分层连接时，可能会发生在不同 MCAUSER 标识下运行的通道。

如果使用指定使用消息上下文的 PUTAUT 值设置通道，那么将根据内部消息中的用户标识来检查对 SYSTEM.INTER.QMGR.* 队列的访问。 由于所有这些消息都与来自发送内部消息或发布消息的队列管理器的分布式发布/预订代理程序的用户标识一起放置 (请参阅 第 89 页的图 39)，因此如果要通过此方式设置分布式发布/预订安全性，那么授予对各种系统队列 (每个远程队列管理器一个) 的访问权的用户标识集不会太大。 它仍然具有通道上下文安全性始终存在的所有相同问题; 不同用户标识域的问题以及消息中的用户标识可能未在接收系统上定义的事实。 但是，如果需要，这是完全可以接受的运行方式。

用于分布式发布/预订的所有队列间管理器消息传递都使用正常通道安全性运行。

有关在主题级别限制发布和代理预订的信息，请参阅 发布/预订安全性 。

将缺省用户标识与队列管理器层次结构配合使用

如果您具有在不同平台上运行的队列管理器层次结构，并且正在使用缺省用户标识，请注意，这些缺省用户标识在不同平台之间有所不同，并且在目标平台上可能未知。 因此，在一个平台上运行的队列管理器会拒绝从其他平台上的队列管理器接收到的消息，原因码为 MQRC_NOT_AUTHORIZED。

为了避免消息被拒绝，至少需要将以下权限添加到其他平台上使用的缺省用户标识:

- SYSTEM.BROKER。 队列
- * SYSTEM.BROKER 上的 PUB *SUB 权限。 主题
- SYSTEM.BROKER.CONTROL.QUEUE 队列。

缺省用户标识如下所示:

平台	缺省用户标识
Windows	MUSR_MQADMIN **注:** MUSR_MQADMIN 是仅用于第一次安装的缺省用户标识。 对于后续安装， "准备 IBM WebSphere MQ 向导" 将创建一个为 MUSR_MQADMINx 命名的用户帐户，其中 x 是表示不存在的用户标识的下一个可用数字。
UNIX and Linux 系统	mqm
IBM i	QMQM
z/OS	通道启动程序地址空间用户标识

如果针对 Windows， UNIX， Linux 和 z/OS 平台上的队列管理器分层连接到 IBM i 上的队列管理器，请创建并授予对 "qmqm" 用户标识的访问权。

如果以分层方式连接到 Windows， UNIX 或 Linux 上的队列管理器 (针对 IBM i 和 z/OS 平台上的队列管理器)，请创建并授予对 "mqm" 用户标识的访问权。

如果分层连接到 Windows， UNIX， Linux 和 IBM i 平台上队列管理器的 z/OS 上的队列管理器，请创建并授予用户对 z/OS 通道启动程序地址空间用户标识的访问权。

用户标识可以区分大小写。 发端队列管理器 (如果 IBM i， Windows， UNIX 或 Linux 系统) 强制用户标识全大写。 接收队列管理器 (如果 Windows， UNIX 或 Linux 系统) 强制用户标识全部为小写。 因此，必须以小写形式创建在 UNIX and Linux 系统上创建的所有用户标识。 如果已安装消息出口，那么不会强制将用户标识设置为大写或小写。 必须注意了解消息出口如何处理用户标识。

要避免用户标识转换的潜在问题，请执行以下操作:

- 在 UNIX， Linux 和 Windows 系统上，确保以小写形式指定用户标识。
- 在 IBM i 和 z/OS 上，确保以大写形式指定用户标识。

分布式发布/预订系统队列

队列管理器使用四个系统队列进行发布/预订消息传递。 您需要了解它们是否存在，仅用于问题确定或容量规划目的。

表 *9:* 分布式平台上的发布/预订系统队列	
系统队列	**用途**
SYSTEM.INTER.QMGR.CONTROL	WebSphere MQ 分布式发布/预订控制队列
SYSTEM.INTER.QMGR.FANREQ	WebSphere MQ 分布式发布/预订内部代理预订扇出进程输入队列
SYSTEM.INTER.QMGR.PUBS	WebSphere MQ 分布式发布/预订出版物
SYSTEM.HIERARCHY.STATE	WebSphere MQ 分布式发布/预订层次结构关系状态

发布/预订系统队列的属性显示在 第 91 页的表 10 中。

表 *10:* 发布/预订系统队列的属性	
属性	**缺省值**
DEFPSIST	Yes
DEFSOPT	扩展
MAXMSGL	在 AIX， HP-UX， Linux， IBM i， Solaris 和 Windows 平台上 :ALTER QMGR 命令的 MAXMSGL 参数值
MAXDEPTH	999999999
SHARE	不适用
STGCLASS	此属性仅在 z/OS 平台上使用

发布/预订系统队列错误
当分布式发布/预订队列管理器队列不可用时，可能会发生错误。

如果扇出请求队列 SYSTEM.INTER.QMGR.FANREQ 不可用， MQSUB API 接收写入错误日志的原因码和错误消息，在需要将代理预订传递到直接连接的队列管理器的情况下。

如果层次结构关系状态队列为 SYSTEM.HIERARCHY.STATE 不可用，将错误消息写入错误日志并将发布/预订引擎置于 COMPAT 方式。

如果存在任何其他 SYSTEM.INTER.QMGR 队列不可用，错误消息将写入错误日志，虽然功能未禁用，但发布/预订消息很可能将在远程队列管理器上的队列上构建。

如果到父级，子级或发布/预订集群队列管理器的传输队列不可用:

1. MQPUT API 接收原因码，但未交付发布。
2. 如果达到回退阈值，那么接收到的队列间管理器发布将回退到输入队列，然后重新尝试将其放在死信队列上。
3. 如果达到回退阈值，那么会将代理预订回退到扇出请求队列，然后再次尝试将其放在死信队列上; 在这种情况下，不会将代理预订传递到任何已连接的队列管理器。
4. 层次结构关系协议消息失败，并且在 PUBSUB 命令上将连接状态标记为 ERROR。

使用 WebSphere MQ 死信队列处理程序处理未传递的消息

什么是死信队列，消息是如何放入的，以及如何管理它?

死信队列 (DLQ) (有时称为 未传递的消息队列) 是无法传递到其目标队列的消息的保留队列。 网络中的每个队列管理器都应该具有关联的 DLQ。

消息可由队列管理器，消息通道代理程序 (MCA) 和应用程序放在 DLQ 上。 死信队列上的所有消息都必须以死信头 结构 MQDLH 作为前缀。

队列管理器或消息通道代理程序放在 DLQ 上的消息始终具有 MQDLH; 将消息放在 DLQ 上的应用程序必须提供 MQDLH。 MQDLH 结构的 原因 字段包含标识消息在 DLQ 上的原因的原因码。

所有 WebSphere MQ 环境都需要一个例程来定期处理 DLQ 上的消息。 WebSphere MQ 提供了一个称为 死信队列处理程序 (DLQ 处理程序) 的缺省例程，您可以使用 `runmqdlq` 命令来调用该例程。

通过用户编写的 规则表向 DLQ 处理程序提供有关在 DLQ 上处理消息的指示信息。 即， DLQ 处理程序将 DLQ 上的消息与规则表中的条目相匹配; 当 DLQ 消息与规则表中的条目相匹配时， DLQ 处理程序将执行与该条目相关联的操作。

调用 DLQ 处理程序

使用 `runmqdlq` 命令调用 DLQ 处理程序。 您可以通过两种方式来命名要处理的 DLQ 和要使用的队列管理器。

这两种方式如下:

- 作为命令提示符中 `runmqdlq` 的参数。 例如:

```
runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER <qrule.rul
```

- 在规则表中。 例如:

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
```

这些示例适用于名为 ABC1.DEAD.LETTER.QUEUE，由队列管理器 ABC1.QUEUE.MANAGER。

如果未如所示指定 DLQ 或队列管理器，那么会将安装的缺省队列管理器与属于该队列管理器的 DLQ 一起使用。

`runmqdlq` 命令从 `stdin` 获取其输入; 通过从规则表重定向 `stdin` 将规则表与 `runmqdlq` 相关联。

要运行 DLQ 处理程序，您必须有权访问 DLQ 本身以及将 DLQ 上的消息转发到的任何消息队列。 要使 DLQ 处理程序将消息放在消息上下文中具有用户标识权限的队列上，您还必须有权采用其他用户的身份。

有关 `runmqdlq` 命令的更多信息，请参阅 runmqdlq。

样本 *DLQ* 处理程序 *amqsdlq*

除了使用 `runmqdlq` 命令调用的 DLQ 处理程序外， WebSphere MQ 还提供了样本 DLQ 处理程序 amqsdlq 的源，其函数类似于 `runmqdlq` 提供的函数。

您可以定制 amqsdlq 以提供符合您需求的 DLQ 处理程序。 例如，您可能决定需要一个可处理不带死信头的消息的 DLQ 处理程序。 (缺省 DLQ 处理程序和样本 amqsdlq 都仅处理 DLQ 上以死信头 MQDLH 开头的那些消息。 未以 MQDLH 开头的消息被标识为出错，并且无限期地保留在 DLQ 上。)

MQ_INSTALLATION_PATH 表示安装 WebSphere MQ 的高级目录。

在 WebSphere MQ for Windows 中， amqsdlq 的源在以下目录中提供:

```
MQ_INSTALLATION_PATH\tools\c\samples\dlq
```

并且在目录中提供了编译版本:

```
MQ_INSTALLATION_PATH\tools\c\samples\bin
```

在 WebSphere MQ for UNIX and Linux 系统中， amqsdlq 的源在以下目录中提供:

MQ_INSTALLATION_PATH/samp/dlq

并且在目录中提供了编译版本:

MQ_INSTALLATION_PATH/samp/bin

DLQ 处理程序规则表

DLQ 处理程序规则表定义 DLQ 处理程序如何处理到达 DLQ 的消息。

规则表中有两种类型的条目:

- 表中的第一个条目 (可选) 包含 控制数据。
- 表中的所有其他条目都是供 DLQ 处理程序遵循的 规则 。 每个规则都由一个 模式 (一组消息特征) 组成，该模式与消息匹配，并且当 DLQ 上的消息与指定的模式匹配时，将执行 操作 。 规则表中必须至少有一个规则。

规则表中的每个条目都包含一个或多个关键字。

控制数据

本部分描述了可以包含在 DLQ 处理程序规则表中的控制数据条目中的关键字。

注:

- 垂直线 (|) 分隔替代项，只能指定其中一个替代项。
- 所有关键字都是可选的。

INPUTQ (*QueueName*| "__")
要处理的 DLQ 的名称:

1. 作为 `runmqdlq` 命令参数提供的任何 INPUTQ 值都将覆盖规则表中的任何 INPUTQ 值。
2. 如果未将 INPUTQ 值指定为 `runmqdlq` 命令的参数，但 **请勿** 在规则表中指定值，那么将使用规则表中的 INPUTQ 值。
3. 如果未指定 DLQ 或在规则表中指定 INPUTQ ('') ，那么将使用属于队列管理器的 DLQ 的名称以及作为 `runmqdlq` 命令的参数提供的名称。
4. 如果未将 INPUTQ 值指定为 `runmqdlq` 命令的参数或规则表中的值，那么将使用属于规则表中 INPUTQM 关键字上指定的队列管理器的 DLQ。

INPUTQM (*QueueManager* 名称| "__")
拥有 INPUTQ 关键字上指定的 DLQ 的队列管理器的名称:

1. 作为参数提供给 `runmqdlq` 命令的任何 INPUTQM 值都将覆盖规则表中的任何 INPUTQM 值。
2. 如果未将 INPUTQM 值指定为 `runmqdlq` 命令的参数，那么将使用规则表中的 INPUTQM 值。
3. 如果未指定队列管理器，或者您在规则表中指定 INPUTQM ('') ，那么将使用安装的缺省队列管理器。

RETRYINT (时间间隔| 60)
这是一个时间间隔 (以秒为单位) ， DLQ 处理程序应在此时间间隔内重新处理第一次尝试时无法处理的 DLQ 上的消息，并且已请求重复尝试。 缺省情况下，重试时间间隔为 60 秒。

WAIT (YES| NO |*nnn*)
当 DLQ 处理程序检测到没有可处理的其他消息时，是否应等待更多消息到达 DLQ。

YES
DLQ 处理程序无限期等待。

否
当检测到 DLQ 为空或不包含可处理的消息时， DLQ 处理程序结束。

nnn
在 DLQ 处理程序检测到队列为空或不包含它可以处理的消息后，它将等待 *nnn* 秒以等待新工作到达，然后再结束。

为繁忙的 DLQ 指定 WAIT (YES) ，为活动级别较低的 DLQ 指定 WAIT (NO) 或 WAIT (*nnn*)。 如果允许 DLQ 处理程序终止，请使用触发再次调用该处理程序。 有关触发的更多信息，请参阅 使用触发器启动 WebSphere MQ 应用程序 。

在规则表中包含控制数据的替代方法是提供 DLQ 及其队列管理器的名称作为 `runmqdlq` 命令的输入参数。 如果在规则表中同时指定值并将其作为 `runmqdlq` 命令的输入，那么在 `runmqdlq` 命令上指定的值优先。

如果在规则表中包含控制数据条目，那么它必须是表中的 **第一个** 条目。

规则 (模式和操作)

模式匹配关键字 (与 DLQ 上的消息匹配的关键字) 和操作关键字 (确定 DLQ 处理程序如何处理匹配消息的关键字) 的描述。 还提供了示例规则。

模式匹配关键字

用于指定与 DLQ 上的消息匹配的值的模式匹配关键字如下所示。 (所有模式匹配关键字都是可选的):

APPLIDAT (*ApplIdentity* 数据|_*)
在 DLQ 上的消息的消息描述符 MQMD 中指定的 *ApplIdentity* 数据 值。

APPLNAME (*PutAppl* 名称|_*)
发出 MQPUT 或 MQPUT1 调用的应用程序的名称，如 DLQ 上消息的消息描述符 MQMD 的 *PutApplName* 字段中所指定。

APPLTYPE (*PutAppl* 类型|_*)
在 DLQ 上的消息的消息描述符 MQMD 中指定的 *PutApplType* 值。

DESTQ (*QueueName*|_*)
以消息为目标的消息队列的名称。

DESTQM (*QueueManager* 名称|_*)
要为其发送消息的消息队列的队列管理器的名称。

FEEDBACK (反馈|_*)
当 *MsgType* 值为 MQFB_REPORT 时， 反馈 描述报告的性质。

您可以使用符号名称。 例如，您可以使用符号名称 MQFB_COA 来标识 DLQ 上需要确认其是否到达其目标队列的那些消息。

FORMAT (格式|_*)
消息发送方用于描述消息数据格式的名称。

MSGTYPE (*MsgType*|_*)
DLQ 上消息的消息类型。

您可以使用符号名称。 例如，可以使用符号名称 MQMT_REQUEST 来标识 DLQ 上需要应答的那些消息。

PERSIST (持久性|_*)
消息的持久性值。 (消息的持久性确定它是否在队列管理器重新启动后仍然存在。)

您可以使用符号名称。 例如，可以使用符号名称 MQPER_PERSISTENT 来标识 DLQ 上持久的消息。

原因 (*ReasonCode*|_*)
描述将消息放入 DLQ 的原因码。

您可以使用符号名称。 例如，您可以使用符号名称 MQRC_Q_FULL 来标识放置在 DLQ 上的那些消息，因为它们的目标队列已满。

REPLYQ (*QueueName*|_*)
在 DLQ 上消息的消息描述符 MQMD 中指定的应答队列的名称。

REPLYQM (*QueueManager* 名称|_*)
应答队列的队列管理器的名称，如 DLQ 上消息的消息描述符 MQMD 中所指定。

USERID (*UserIdentifier*|_*)
在 DLQ 上生成消息的用户的用户标识，如 DLQ 上消息的消息描述符 MQMD 中所指定。

操作关键字

用于描述如何处理匹配消息的操作关键字如下所示:

ACTION (DISCARD | IGNORE | RETRY | FWD)
要对 DLQ 上与此规则中定义的模式匹配的任何消息执行的操作。

丢弃
从 DLQ 中删除消息。

IGNORE
将消息保留在 DLQ 上。

重试
如果第一次尝试将消息放在其目标队列上失败，请重试。 RETRY 关键字设置为实现操作而进行的尝试次数。 控制数据的 RETRYINT 关键字控制尝试之间的时间间隔。

转发
将消息转发到 FWDQ 关键字上指定的队列。

必须指定 ACTION 关键字。

FWDQ (*QueueName*|&DESTQ|&REPLYQ)
请求 ACTION (FWD) 时要将消息转发到的消息队列的名称。

QueueName
消息队列的名称。 FWDQ ('') 无效。

&DESTQ
从 MQDLH 结构中的 *DestQName* 字段获取队列名称。

&REPLYQ
从消息描述符 MQMD 中的 *ReplyToQ* 字段获取队列名称。

To avoid error messages when a rule specifying FWDQ (&REPLYQ) matches a message with a blank *ReplyToQ* field, specify REPLYQ (?*) in the message pattern.

FWDQM (*QueueManager* 名称|&DESTQM|&REPLYQM|'__')
要将消息转发到的队列的队列管理器。

***QueueManager* 名称**
请求 ACTION (FWD) 时要将消息转发到的队列的队列管理器的名称。

&DESTQM
从 MQDLH 结构中的 *DestQMgrName* 字段获取队列管理器名称。

&REPLYQM
从消息描述符 MQMD 中的 *ReplyToQMgr* 字段获取队列管理器名称。

''
FWDQM ("") (缺省值) 标识本地队列管理器。

HEADER (YES| NO)
MQDLH 是否应保留在请求 ACTION (FWD) 的消息上。 缺省情况下， MQDLH 保留在消息上。 HEADER 关键字对于 FWD 以外的操作无效。

PUTAUT (DEF| CTX)
应由 DLQ 处理程序放置消息的权限:

DEF
使用 DLQ 处理程序本身的权限放入消息。

CTX
将具有用户标识权限的消息放在消息上下文中。 如果指定 PUTAUT (CTX) ，那么您必须有权采用其他用户的身份。

RETRY (*RetryCount*|1)
尝试操作 (在控制数据的 RETRYINT 关键字上指定的时间间隔) 的次数，范围为 1-999,999,999。 DLQ 处理程序为实现任何特定规则而进行的尝试计数特定于 DLQ 处理程序的当前实例; 该计数不会在重新启动之间持久存在。 如果重新启动 DLQ 处理程序，那么应用规则的尝试计数将重置为零。

示例规则

以下是来自 DLQ 处理程序规则表的示例规则:

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
 ACTION (RETRY) RETRY (3)
```

此规则指示 DLQ 处理程序进行三次尝试，以将由于禁止 MQPUT 和 MQPUT1 而放置在 DLQ 上的任何持久消息传递到其目标队列。

在此部分的其余部分中描述了可用于规则的所有关键字。 请注意下列事项：

- 关键字的缺省值 (如果有) 带有下划线。 对于大多数关键字，缺省值为 * (星号)，这与任何值匹配。
- 垂直线 (|) 分隔替代项，只能指定其中一个替代项。
- 除 ACTION 以外的所有关键字都是可选的。

规则表约定

DLQ 处理程序规则表的语法，结构和内容必须遵循这些约定。

规则表必须遵循以下约定：

- 规则表必须至少包含一个规则。
- 关键字可以按任意顺序出现。
- 关键字只能包含在任何规则中一次。
- 关键字不区分大小写。
- 关键字及其参数值必须与其他关键字至少用一个空格或逗号分隔。
- 在规则的开头或结尾以及关键字，标点和值之间可以有任意数目的空格。
- 每个规则都必须在新行上开始。
- 在 Windows 系统上，表中的最后一条规则必须以回车符/换行符结尾。 您可以通过确保在规则末尾按 Enter 键来实现此目的，以便表的最后一行是空白行。
- 出于可移植性的原因，行的显着长度不得大于 72 个字符。
- 使用加号 (+) 作为行上的最后一个非空白字符，以指示规则从下一行中的第一个非空白字符继续。 使用减号 (-) 作为行上的最后一个非空白字符，以指示规则从下一行的开头开始继续。 可以在关键字和参数中出现连续字符。

  例如：

  ```
  APPLNAME('ABC+
    D')
  ```

  生成 "ABCD"，并且

  ```
  APPLNAME('ABC-
    D')
  ```

  结果为 " ABC D"。
- 以星号 (*) 开头的注释行可以出现在规则表中的任何位置。
- 空白行予以忽略。
- DLQ 处理程序规则表中的每个条目都包含一个或多个关键字及其关联参数。这些参数必须遵循以下语法规则：
  - 每个参数值必须至少包含一个有效字符。 用引号括起的值中的定界单引号不会被认为是重要的。 例如，以下参数有效：

FORMAT('ABC')	3 个有效字符
FORMAT(ABC)	3 个有效字符
FORMAT('A')	1 有效字符
FORMAT(A)	1 有效字符

```
FORMAT(' ')          1 有效字符
```

这些参数无效，因为它们不包含重要字符:

```
FORMAT('')
FORMAT( )
FORMAT()
FORMAT
```

- 支持通配符。 可以使用问号 (?) 代替任何单个字符，但尾部空格除外; 可以使用星号 (*) 代替零个或多个相邻字符。 星号 (*) 和问号 (?) **始终** 解释为参数值中的通配符。
- 这些关键字的参数中不能包含通配符 :ACTION， HEADER， RETRY， FWDQ， FWDQM 和 PUTAUT。
- 在执行通配符匹配时，参数值以及 DLQ 上的消息中相应字段中的尾部空格不重要。 但是，以单引号括起的字符串中的前导空格和嵌入空格对于通配符匹配很重要。
- 数字参数不能包含问号 (?) 通配符。 可以使用星号 (*) 代替整个数字参数，但不能将其作为数字参数的一部分。 例如，以下是有效的数字参数:

```
MSGTYPE(2)           只有应答消息才符合条件
MSGTYPE(*)           任何消息类型都符合条件
MSGTYPE('*')         任何消息类型都符合条件
```

  但是， MSGTYPE('2*') 无效，因为它包含星号 (*) 作为数字参数的一部分。
- 数字参数必须在范围 0-999 999 999 之间。 如果参数值在此范围内，那么将接受该参数值，即使该参数值在与关键字相关的字段中当前无效也是如此。 可以将符号名称用于数字参数。
- 如果字符串值比与关键字相关的 MQDLH 或 MQMD 中的字段短，那么该值将用空白填充到字段的长度。 如果该值 (不包括星号) 比字段长，那么将诊断错误。 例如，以下是 8 字符字段的所有有效字符串值:

```
'ABCDEFGH'           8 个字符
'A*C*E*G*I'          5 个字符 (不包括星号)
'*A*C*E*G*I*K*M*O    8 个字符 (不包括星号)
*'
```

- 将包含空格，小写字符或除句点 (.)，正斜杠 (?)，下划线 (_) 和百分号 (%) 以外的特殊字符的字符串括在单引号中。 未括在单引号内的小写字符将转换为大写。 如果字符串包含引号，请使用两个单引号来表示引号的开头和结尾。 计算字符串的长度时，每次出现的双引号都算作单个字符。

处理规则表的方式

DLQ 处理程序在规则表中搜索模式与 DLQ 上的消息匹配的规则。

搜索以表中的第一个规则开始，并在表中按顺序继续。 当 DLQ 处理程序找到具有匹配模式的规则时，它将从该规则执行操作。 无论何时应用规则， DLQ 处理程序都会将规则的重试计数递增 1。 如果第一次尝试失败，那么 DLQ 处理程序将再次尝试，直到尝试次数与 RETRY 关键字上指定的次数匹配为止。 如果所有尝试都失败，那么 DLQ 处理程序将搜索表中的下一个匹配规则。

将针对后续匹配规则重复此过程，直到操作成功为止。 当尝试每个匹配规则时，在其 RETRY 关键字上指定的次数，并且所有尝试都失败了，那么假定为 ACTION (IGNORE)。 如果找不到匹配的规则，那么也将采用 ACTION (IGNORE)。

注:

1. 仅针对以 MQDLH 开头的 DLQ 上的消息查找匹配的规则模式。 不以 MQDLH 开头的消息会定期报告为出错，并无限期地保留在 DLQ 上。
2. 可以允许所有模式关键字为缺省值，以便规则只能由操作组成。 但是，请注意，仅操作规则将应用于队列上具有 MQDLH 且尚未根据表中的其他规则进行处理的所有消息。

3. 当 DLQ 处理程序启动时，将验证规则表，并在该时间标记错误。 您可以随时对规则表进行更改，但这些更改直到 DLQ 处理程序重新启动后才生效。
4. DLQ 处理程序不会更改消息， MQDLH 或消息描述符的内容。 DLQ 处理程序始终使用消息选项 MQPMO_PASS_ALL_CONTEXT 将消息放入其他队列。
5. 可能无法识别规则表中的连续语法错误，因为规则表旨在避免在验证期间生成重复错误。
6. DLQ 处理程序使用 MQOO_INPUT_AS_Q_DEF 选项打开 DLQ。
7. 可使用相同的规则表对同一队列并发运行 DLQ 处理程序的多个实例。 但是，在 DLQ 和 DLQ 处理程序之间存在一对一关系是比较常见的。

确保处理所有 *DLQ* 消息

DLQ 处理程序保留 DLQ 上已看到但未除去的所有消息的记录。

如果使用 DLQ 处理程序作为过滤器从 DLQ 中抽取一小部分消息，那么 DLQ 处理程序仍必须在其未处理的 DLQ 上保留这些消息的记录。 此外， DLQ 处理程序无法保证会看到到达 DLQ 的新消息，即使 DLQ 定义为先进先出 (FIFO) 也是如此。 如果队列不为空，那么将定期重新扫描 DLQ 以检查所有消息。

出于这些原因，请尝试确保 DLQ 包含尽可能少的消息; 如果允许无法丢弃或转发到其他队列的消息 (无论出于何种原因) 在队列上累积，那么 DLQ 处理程序的工作负载会增加，并且 DLQ 本身会填满。

您可以采取特定措施，使 DLQ 处理程序能够清空 DLQ。 例如，尝试不使用 ACTION (IGNORE) ，这会将消息保留在 DLQ 上。 (请记住，对于表中的其他规则未显式处理的消息，将采用 ACTION (IGNORE)。) 相反，对于那些要忽略的消息，请使用将消息移至另一个队列的操作。 例如:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

同样，使表中的最终规则成为用于处理表中先前规则未处理的消息的 catchall。 例如，表中的最终规则可能如下所示:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

这会将落到表中最终规则的消息转发到队列 REALLY.DEAD.QUEUE，在该队列中可以手动处理这些消息。 如果您没有这样的规则，那么消息可能会无限期地保留在 DLQ 上。

一个示例 DLQ 处理程序规则表

runmqdlq 命令的示例规则表，其中包含单个控制数据条目和多个规则。

```
*********************************************************************
*        An example rules table for the runmqdlq command            *
*********************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* runmqdlq, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to runmqdlq,
* use the DLQ defined for the local queue manager.
*
inputqm(' ')  inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.
* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).
```

```
REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation are always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
  ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never do things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
  action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
  ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)
```

```
* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it should be able
* to cope with the message being lost, so we can afford to
* discard the message.   PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)
* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We don't have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
  ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

多个安装

在 UNIX, Linux, and Windows 上，可以在系统上具有多个 IBM WebSphere MQ 副本。

您可以选择安装 IBM WebSphere MQ 的每个副本的位置，但每个副本必须位于单独的安装位置。 一次最多可以在一台机器上存在 128 个 IBM WebSphere MQ 安装。 一个安装可以是 IBM WebSphere MQ Version 7.0.1 在修订包 6 或更高版本上的安装。 您现在可以选择:

- 保持在机器上维护和管理单个 IBM WebSphere MQ 安装的简单性。
- 通过启用多个 IBM WebSphere MQ 安装来利用所提供的灵活性。

在安装 IBM WebSphere MQ 的多个副本之前，必须做出若干决策:

- 系统上是否有 IBM WebSphere MQ Version 7.0.1 的副本?

  在系统上安装 IBM WebSphere MQ Version 7.0.1 at fix pack 6 或更高版本时，需要考虑一些限制:

  – 在 UNIX and Linux 系统上， IBM WebSphere MQ Version 7.0.1 必须安装在缺省位置。

  - IBM WebSphere MQ Version 7.0.1 必须是系统上的第一个安装。 在安装 V 7.1 或更高版本之后，无法安装 IBM WebSphere MQ Version 7.0.1 。 如果卸载 V 7.0.1，那么在安装更高版本的 WebSphere MQ 时无法重新安装该版本。
  - IBM WebSphere MQ Version 7.0.1 是自动主安装。 在安装 IBM WebSphere MQ Version 7.0.1 时，不能选择其他安装作为主安装。
- 您将在何处安装 IBM WebSphere MQ 的每个副本?

  您可以选择安装版本为 7.1 或更高版本的安装位置。 有关更多信息，请参阅 选择安装位置。
- 需要主安装吗?

  主安装是系统范围位置所引用的安装。 有关更多信息，请参阅第 100 页的『选择主安装』。
- 您的应用程序将如何连接?

  您需要考虑应用程序如何找到相应的 IBM WebSphere MQ 库。 有关更多信息，请参阅 在多安装环境中连接应用程序和 在多安装环境中连接 .NET 应用程序。
- 是否需要更改现有出口?

  如果 IBM WebSphere MQ 未安装在缺省位置中，那么需要更新出口。 有关更多信息，请参阅 编写和编译出口和可安装服务 。
- 哪个队列管理器将与哪个安装相关联?

  每个队列管理器都与特定安装相关联。 队列管理器与该队列管理器的限制相关联的安装，以便该队列管理器只能由该安装中的命令进行管理。 有关更多信息，请参阅 使队列管理器与安装相关联。
- 您将如何设置环境以处理每个安装?

  对于系统上的多个安装，您需要考虑如何处理特定安装，以及如何从该安装发出命令。 您可以指定命令的完整路径，也可以使用 **setmqenv** 或 **crtmqenv** 命令来设置环境变量。 设置环境变量允许您省略该安装的命令的路径。 有关更多信息，请参阅 setmqenv 和 crtmqenv。

回答完这些问题后，可以使用 安装 IBM WebSphere MQ 中提供的步骤来安装 IBM WebSphere MQ 。

如果您具有 IBM WebSphere MQ 的现有安装，并且要使用多个安装功能从 IBM WebSphere MQ 的一个版本迁移到另一个版本，请参阅 UNIX， Linux 和 Windows 上的多安装队列管理器共存。

IBM Message Service Client for .NET 支持包和多个安装

对于多版本支持，必须随 IBM WebSphere MQ 产品一起安装 *Java* 和 *.NET Messaging and Web Services* 功能部件。 此功能部件包含 *IBM Message Service Client for .NET* 支持包 (IA9H) 中包含的所有功能。 如果支持包安装在系统上，那么不支持多个版本。 必须先卸载支持包，然后才能安装 IBM WebSphere MQ。 有关安装 .NET 功能部件的更多信息，请参阅 安装 WebSphere MQ classes for .NET。

相关概念

UNIX， Linux 和 Windows: 从 V 7.0.1 到 V 7.5 的并行迁移

UNIX， Linux 和 Windows: 从 V 7.0.1 到 V 7.5 的多阶段迁移

相关任务

配置多个安装

在系统上查找 WebSphere MQ 的安装

选择主安装

在支持 IBM WebSphere MQ (UNIX, Linux, and Windows) 的多个安装的系统上，主安装是 IBM WebSphere MQ 系统范围位置所引用的安装。 具有主安装是可选的，但很方便。

在 IBM WebSphere MQ Version 7.1 之前，任何时候都只能安装产品的一个实例。 在 Windows 系统上，设置了多个全局环境变量以指向该安装。 在 UNIX and Linux 系统上，向 `/usr/lib`， `/usr/bin` 和 `/usr/include` 添加了符号链接，也指向该单一安装。

从 Version 7.1 开始，可以在 UNIX, Linux, and Windows 上安装多个版本的 IBM WebSphere MQ 。 可以随时在其中一个系统上安装多个 IBM WebSphere MQ ，也可以选择性地将其中一个安装配置为主安装。 当存在多个版本时，指向单个安装的环境变量和符号链接不太有意义。 但是，某些功能需要这些系统范围的位置

才能工作。 例如，用于管理 IBM WebSphere MQ 和第三方产品的定制用户脚本。 这些功能仅在主安装上起作用。

在 UNIX and Linux 系统上，如果将安装设置为主安装，那么会将指向该安装的外部库和控制命令的符号链接添加到 /usr/lib 和 /usr/bin 中。 如果您没有主安装，那么不会创建符号链接。 要获取到主安装的符号链接的列表，请参阅 外部库和控制命令链接到 UNIX and Linux 上的主安装。

在 Windows 系统上，全局环境变量指向安装了主安装的目录。 这些环境变量用于查找 IBM WebSphere MQ 库，控制命令和头文件。 此外，在 Windows 系统上，操作系统的某些功能需要集中注册接口库，然后将这些接口库装入到单个进程中。 使用多个版本的 IBM WebSphere MQ 时， IBM WebSphere MQ 库将存在冲突集。 这些功能部件将尝试将这些有冲突的库集装入到单个进程中。 因此，此类功能部件只能用于主安装。 有关限于与主安装配合使用的某些功能部件的详细信息，请参阅 在 Windows 上只能与主安装配合使用的功能部件。

如果在系统上安装了 IBM WebSphere MQ Version 7.0.1 ，那么此安装将自动成为主安装。 安装 Version 7.0.1 时，无法更改主安装。 如果系统上的所有安装都位于 Version 7.1 或更高版本，那么您可以选择是否进行主安装。 请考虑 第 101 页的表 11 中的选项。

表 *11:* 主安装选项.

此表显示主安装的有效安装配置。 对于单个 Version 7.1 或更高版本，它可以是主项或非主项。 对于多个安装，一个安装在 Version 7.0.1 ，一个或多个安装在 Version 7.1 或更高版本，Version 7.0.1 必须是主安装，而其他安装必须是非主安装。 对于 Version 7.1 或更高版本的多个安装，一个安装可以是主安装，或者所有安装都可以是非主安装。

选项	**有效的安装配置**		**详细信息**
	主	非主	
Version 7.1 或更高版本的单次安装。	Version 7.1 或更高版本。	None	如果要以与先前发行版相同的方式继续使用单个安装，请将安装配置为主安装。 有关此选项的信息，请参阅 配置为主安装的 IBM WebSphere MQ Version 7.1 或更高版本的单一安装
	None	Version 7.1 或更高版本。	如果要继续使用单个安装，但不希望为您创建符号链接或全局环境变量，请将安装配置为非主安装。 有关此选项的含义的信息，请参阅 配置为非主项的 IBM WebSphere MQ Version 7.1 或更高版本的单一安装
多个安装: Version 7.0.1 和 Version 7.1 或更高版本。	Version 7.0.1	Version 7.1 或更高版本。	如果要多次安装 IBM WebSphere MQ(其中一个版本为 7.0.1) ，那么版本 7.0.1 安装将自动成为主安装。 安装 IBM WebSphere MQ V 7.0.1 时，您无法更改哪个安装是主安装。 有关此选项及其影响的信息，请参阅 IBM WebSphere MQ 的多个安装，其中一个安装位于 Version 7.0.1
多个安装: Version 7.1 或更高版本。	Version 7.1 或更高版本。	Version 7.1 或更高版本。	如果要具有 WebSphere MQ V 7.1 或更高版本的多个安装，那么可以选择是否将其中一个安装设置为主安装。 有关此选项的信息，请参阅 IBM WebSphere MQ Version 7.1 或更高版本的多个安装
	None	Version 7.1 或更高版本。	

相关概念

单个安装 WebSphere MQ V7.1 或更高版本，配置为主安装

配置为非主服务器的 WebSphere MQ V 7.1 或更高版本的单次安装

WebSphere MQ V 7.1 或更高版本的多个安装

WebSphere MQ 的多个安装，一个版本为 7.0.1

相关任务

更改主安装

选择安装位置
规划您的安装
选择安装名称

配置为主安装的 IBM WebSphere MQ Version 7.1 或更高版本的单次安装

将 IBM WebSphere MQ 安装标记为主安装会向系统添加符号链接或全局环境变量，以便应用程序使用的 IBM WebSphere MQ 命令和库在需要最低系统设置时自动可用。

您可以决定安装 IBM WebSphere MQ 的位置。

在可能的情况下，配置应用程序和脚本以使用系统搜索路径来查找 IBM WebSphere MQ 控制命令或 IBM WebSphere MQ 库。 应用程序和脚本的此配置为执行将来的任务 (例如，迁移到 IBM WebSphere MQ 的下一个发行版或安装第二个安装) 提供了最大的灵活性。 有关用于连接应用程序的选项的更多信息，请参阅 在多安装环境中连接应用程序。

在 Windows 上，第一次安装会自动配置为主安装。 在 UNIX and Linux 平台上，必须手动将系统上的第一个安装配置为主安装。 使用 **setmqinst** 命令设置主安装。 有关更多信息，请参阅 卸载，升级和维护主安装。

相关任务
更改主安装
选择安装位置
规划您的安装
选择安装名称

配置为非主安装的 IBM WebSphere MQ Version 7.1 或更高版本的单个安装

如果安装 IBM WebSphere MQ Version 7.1 或更高版本，那么作为非主项，您可能必须为应用程序配置库路径以装入 IBM WebSphere MQ 库。 在 Windows 上，仅当 IBM WebSphere MQ 配置为主产品时，某些产品功能才可用。

UNIX 和 Linux 系统

在 UNIX and Linux 上运行非主安装的含义如下:

- 如果满足以下条件，那么使用嵌入式库路径 (例如， RPATH) 找到其 IBM WebSphere MQ 库的应用程序将找不到这些库:
  - IBM WebSphere MQ 安装在与 RPATH 中指定的目录不同的目录中
  - /usr 中没有符号链接
- 如果应用程序使用外部库路径 (例如 LD_LIBRARY_PATH) 找到其库，那么必须配置外部库路径以包含 *MQ_INSTALLATION_PATH*/lib 或 *MQ_INSTALLATION_PATH*/lib64 目录。 **setmqenv** 和 **crtmqenv** 命令可以在当前 shell 中配置许多环境变量，包括外部库路径。
- 大多数 IBM WebSphere MQ 进程作为 setuid/setgid 运行。 因此，在装入用户出口时，它们将忽略外部库路径。 仅当在 IBM WebSphere MQ 库中嵌入的库路径中找到这些库时，引用这些库的用户出口才能找到这些库。 如果 /usr 中存在符号链接，那么将解决这些问题。 现在，可以构建要在 IBM WebSphere MQ Version 7.1 或更高版本上运行的用户出口，以便它们完全不引用 IBM WebSphere MQ 库。 而是依赖于 IBM WebSphere MQ 将函数指针传递到 IBM WebSphere MQ 函数，然后出口可以使用这些函数。 有关更多信息，请参阅 编写和编译出口和可安装服务 。

有关用于连接应用程序的选项的更多信息，请参阅 在多安装环境中连接应用程序。

在 UNIX and Linux 平台上，系统上的第一个安装不会自动配置为主安装。 但是， /usr/bin 中包含单个符号链接以查找 **dspmqver** 命令。 如果不需要任何符号链接，那么必须使用以下命令除去此链接:

```
setmqinst -x -p MQ_INSTALLATION_PATH
```

在 Windows 系统上

在 Windows 上运行非主安装的含义如下:

- 应用程序通常使用外部库路径 PATH 来查找其库。 没有嵌入式库路径或显式库位置的概念。 如果安装是非主安装，那么全局 PATH 环境变量不包含 IBM WebSphere MQ 安装目录。 对于要查找 IBM WebSphere MQ 库的应用程序，请更新 PATH 环境变量以引用 IBM WebSphere MQ 安装目录。 **setmqenv** 和 **crtmqenv** 命令可以在当前 shell 中配置许多环境变量，包括外部库路径。
- 仅当将安装配置为主安装时，某些产品功能才可用; 请参阅 在 Windows 上只能与主安装配合使用的功能部件。

缺省情况下，在 Windows 上，第一次安装会自动配置为主安装。 您必须手动取消选择它作为主安装。

相关任务
更改主安装
选择安装位置
规划您的安装
选择安装名称
相关参考
setmqenv
克特蒙琴夫

IBM WebSphere MQ Version 7.1 或更高版本的多个安装

您可以选择将其中一个 IBM WebSphere MQ Version 7.1 或更高版本的安装配置为主安装。 您的选择取决于应用程序如何查找库。

IBM WebSphere MQ Version 7.1 随附的 IBM WebSphere MQ 库 (例如 mqm) 会自动使用它们所连接到的队列管理器所需的级别的库。 这意味着如果提供了应用程序从 IBM WebSphere MQ Version 7.1 安装中查找其 IBM WebSphere MQ 库，那么它可以连接到该系统上的任何队列管理器。 将一个 IBM WebSphere MQ Version 7.1 安装配置为主安装可确保如果应用程序找到其 IBM WebSphere MQ 接口库，那么应用程序可以连接到任何队列管理器。

有关在多安装环境中连接应用程序的更多信息，请参阅 在多安装环境中连接应用程序。

卸载主安装时，不会自动更改主安装。 如果您希望另一个安装是主安装，那么必须使用 **setmqinst** 命令手动设置主安装。 有关更多信息，请参阅 卸载，升级和维护主安装。

相关概念
多个安装
相关任务
更改主安装
选择安装位置
规划您的安装
选择安装名称

IBM WebSphere MQ 的多个安装，一个安装在 Version 7.0.1

IBM WebSphere MQ Version 7.1 或更高版本可以与 IBM WebSphere MQ Version 7.0.1 共存，但存在一些限制。

- 在 UNIX and Linux 系统上， Version 7.0.1 只能安装在固定缺省位置，因此不能在该缺省位置安装 Version 7.1 或更高版本。
- IBM WebSphere MQ Version 7.0.1 会自动配置为主安装。 在 UNIX and Linux 系统上，将自动创建指向相应 IBM WebSphere MQ 目录的符号链接。 在 Windows 上，产品提供的所有内容都在全局注册。 IBM WebSphere MQ Version 7.0.1 必须以这种方式安装才能工作。 因此，在安装了 IBM WebSphere MQ Version 7.0.1 ， IBM WebSphere MQ Version 7.1 或更高版本的情况下，无法使安装成为主安装。

来自 IBM WebSphere MQ Version 7.1 或更高版本的库可以使用在 IBM WebSphere MQ Version 7.0.1 或更高版本下运行的任何队列管理器。 如果应用程序需要连接到在 Version 7.0.1 以及更高版本下运行的队列管理器，那么在满足以下条件时，它可以继续正常运行:

- 它在运行时查找 IBM WebSphere MQ Version 7.1 或更高版本的库。
- 它仅使用 Version 7.0.1 中可用的功能。

有关在多安装环境中连接应用程序的更多信息，请参阅 在多安装环境中连接应用程序。

卸载 IBM WebSphere MQ Version 7.0.1 时，不会自动更改主安装。 如果您希望另一个安装是主安装，那么必须使用 **setmqinst** 命令手动设置主安装。 有关更多信息，请参阅 卸载，升级和维护主安装。

相关概念
多个安装
相关任务
选择安装位置
规划您的安装
选择安装名称

规划存储和性能需求

您必须为 IBM WebSphere MQ 系统设置切合实际且可实现的存储和性能目标。 使用这些链接可了解影响平台上的存储和性能的因素。

根据您在其中使用 IBM WebSphere MQ 的系统以及要使用的组件，需求会有所不同。

有关受支持的硬件和软件环境的最新信息，请参阅 IBM WebSphere MQ 的系统需求 Web 站点:

```
www.ibm.com/software/integration/wmq/requirements/
```

IBM WebSphere MQ 将队列管理器数据存储在文件系统中。 使用以下链接来了解有关规划和配置目录结构以用于 IBM WebSphere MQ 的信息:

- 第 106 页的『规划文件系统支持』
- 第 106 页的『共享文件系统的需求』
- 第 114 页的『共享 IBM WebSphere MQ 文件』
- 第 116 页的『UNIX and Linux 系统上的目录结构』
- 第 125 页的『Windows 系统上的目录结构』

使用以下链接以获取有关 UNIX and Linux 上的系统资源，共享内存和进程优先级的信息:

- 第 128 页的『IBM WebSphere MQ 和 UNIX System V IPC 资源』
- 第 129 页的『 AIX 上的共享内存』
- 第 129 页的『 WebSphere MQ 和 UNIX 进程优先级』

相关概念
第 5 页的『规划』
规划 IBM WebSphere MQ 环境时，必须考虑要配置的 IBM WebSphere MQ 体系结构，资源需求，日志记录需求和备份工具。 使用本主题中的链接来规划 IBM WebSphere MQ 运行所在的环境。

第 13 页的『设计 IBM WebSphere MQ 体系结构』
了解 IBM WebSphere MQ 针对点到点和发布/预订消息传递样式支持的不同体系结构。

UNIX 和 Linux 上的硬件和软件需求
Windows 上的硬件和软件需求

磁盘空间需求

WebSphere MQ 的存储需求取决于您安装的组件以及所需的工作空间量。

您选择安装的可选组件 (包括所需的任何必备组件) 需要磁盘存储器。 总存储需求还取决于您使用的队列数，队列上消息的数量和大小以及消息是否持久。 您还需要磁盘，磁带或其他介质上的归档容量以及您自己的应用程序的空间。

下表显示了在不同平台上安装产品的各种组合时所需的大致磁盘空间。 (值向上取整为最接近的 5 MB，其中 MB 为 1,048,576 字节。)

平台	客户机安装 [1]	服务器安装 [2]	WebSphere MQ MFT 安装 [3]	完全安装 [4]
AIX	145 MB	190 MB	705 MB	915 MB
HP-UX	225 MB	310 MB	1075 MB	1340 MB
IBM i	215 MB	450 MB	80 MB	655 MB
Linux for System x (32 位)	85 MB	N/A	N/A	120 MB
Linux for System x (64 位)	125 MB	170 MB	575 MB	935 MB
Linux on POWER Systems-大尾数法	130 MB	170 MB	565 MB	715 MB
Solaris x86-64, AMD64， EM64T 和兼容处理器	105 MB [5]	150 MB [5]	695 MB	860 MB
Solaris SPARC	105 MB [5]	150 MB [5]	680 MB	820 MB
Windows (32 位安装) [6]	390 MB	N/A	N/A	475 MB
Windows (64 位安装) [6]	445 MB	555 MB	710 MB	1005 MB

使用说明

1. 客户机安装包含以下组件:
   - 运行时
   - 客户机
2. 服务器安装包含以下组件:
   - 运行时
   - 服务器
3. IBM WebSphere MQ Managed File Transfer 安装包含以下组件:
   - IBM WebSphere MQ Managed File Transfer 服务，记录器，代理程序，工具和基本组件
   - 运行时
   - 服务器
   - Java
   - JRE
4. 完整安装包含所有可用组件。
5. Solaris 在 Solaris 平台上，您必须以静默方式进行安装以获取此组件组合。
6. Windows 并非此处列出的所有组件都是 Windows 系统上的可安装功能部件; 它们的功能有时包含在其他功能部件中。 请参阅 Windows 系统的 WebSphere MQ 功能部件。

相关任务

选择要安装的内容

规划文件系统支持

队列管理器数据存储在文件系统中。 队列管理器使用文件系统锁定来防止多实例队列管理器的多个实例同时处于活动状态。

共享文件系统

共享文件系统使多个系统能够同时访问同一物理存储设备。 如果多个系统直接访问同一物理存储设备，而没有某种强制锁定和并行控制的方法，那么将发生损坏。 操作系统为本地文件系统提供对本地进程的锁定和并行控制; 网络文件系统为分布式系统提供锁定和并行控制。

历史上，网络文件系统的执行速度不够快，或者提供了足够的锁定和并行控制，无法满足日志记录消息的要求。 如今，联网的文件系统可以提供良好的性能，实现可靠的网络文件系统协议，如 *RFC 3530* ，网络文件系统 *(NFS)* 版本 *4* 协议，满足可靠记录消息的需求。

共享文件系统和 WebSphere MQ

多实例队列管理器的队列管理器数据存储在共享网络文件系统中。 在 Microsoft Windows UNIX and Linux 系统上，队列管理器的数据文件和日志文件必须放在共享网络文件系统中。

在发行版 v7.0.1 之前， WebSphere MQ 不支持存储在作为共享文件系统访问的网络存储器上的队列管理器数据。 如果将队列管理器数据放在共享网络存储器上，那么您需要确保队列管理器数据不会被同时运行的队列管理器的另一个实例访问。

从 v7.0.1 开始， WebSphere MQ 使用锁定来防止同一多实例队列管理器的多个实例同时处于活动状态。 同一锁定还可确保两个单独的队列管理器不能无意中使用同一组队列管理器数据文件。 一次只能有一个队列管理器实例具有其锁定。 因此， WebSphere MQ 支持存储在作为共享文件系统访问的联网存储器上的队列管理器数据。

由于并非网络文件系统的所有锁定协议都是健全的，并且由于可能针对性能而非数据完整性配置了文件系统，因此您必须运行 **amqmfsck** 命令来测试网络文件系统是否将正确控制对队列管理器数据和日志的访问。 此命令仅适用于 UNIX 和 IBM i 系统。 在 Microsoft Windows 上，只有一个受支持的网络文件系统，不需要 **amqmfsck** 命令。

相关任务

第 108 页的『验证共享文件系统行为』

运行 **amqmfsck** 以检查 UNIX 系统上的共享文件系统是否满足存储多实例队列管理器的队列管理器数据的需求。 与 **amqmfsck** 并行运行 IBM WebSphere MQ MQI client 样本程序 **amqsfhac** 以演示队列管理器在故障期间维护消息完整性。

共享文件系统的需求

共享文件系统必须提供数据写完整性，保证对文件的互斥访问权，并在无法可靠地使用 IBM WebSphere MQ 时释放锁定。

共享文件系统必须满足的需求

共享文件系统必须满足三个基本要求才能可靠地记录消息:

1. 数据写入完整性

   数据写入完整性有时称为 清空时写入磁盘。 队列管理器必须能够与成功落实到物理设备的数据同步。 在事务系统中，您需要确保在继续其他处理之前已安全地落实某些写操作。

   更具体地说， UNIX 平台上的 IBM WebSphere MQ 使用 *O_SYNC* 打开选项和 `fsync()` 系统调用来显式强制写入可恢复介质，并且依赖于这些选项正常运行。

注意: Linux 您应该使用 async 选项安装文件系统，该选项仍然支持同步写入选项，并提供比 sync 选项更好的性能。

但是，请注意，如果已从 Linux 导出文件系统，那么仍必须使用 sync 选项导出文件系统。

2. 保证对文件的独占访问权

   为了使多个队列管理器同步，需要有一种队列管理器获取对文件的互斥锁定的机制。

3. 发生故障时释放锁定

   如果队列管理器发生故障，或者如果与文件系统发生通信故障，那么队列管理器锁定的文件需要解锁并可供其他进程使用，而无需等待队列管理器重新连接到文件系统。

共享文件系统必须满足这些要求才能使 IBM WebSphere MQ 可靠运行。 如果没有，那么在多实例队列管理器配置中使用共享文件系统时，队列管理器数据和日志会损坏。

对于 Microsoft Windows 上的多实例队列管理器，必须通过 Microsoft Windows 网络使用的通用因特网文件系统 (CIFS) 协议来访问联网存储器。 通用因特网文件系统 (CIFS) 客户机不满足 IBM WebSphere MQ 在 Microsoft Windows 以外的平台上锁定语义的要求，因此在 Microsoft Windows 以外的平台上运行的多实例队列管理器不得将通用因特网文件系统 (CIFS) 用作其共享文件系统。

对于其他受支持平台上的多实例队列管理器，必须通过符合 Posix 的网络文件系统协议来访问存储器，并支持基于租赁的锁定。 现代文件系统 (例如网络文件系统 (NFS) V 4) 使用租赁的锁定来检测故障，然后在发生故障后释放锁定。 不能将较旧的文件系统 (例如，网络文件系统版本 3) 与多实例队列管理器配合使用，这些系统没有在发生故障后释放锁定的可靠机制。

检查共享文件系统是否满足需求

您必须检查计划使用的共享文件系统是否满足这些需求。 您还必须检查是否正确配置了文件系统以实现可靠性。 共享文件系统有时会提供配置选项，以牺牲可靠性来提高性能。

在正常情况下， IBM WebSphere MQ 使用属性高速缓存正常运行，不需要禁用高速缓存，例如通过在 NFS 安装上设置 NOAC。 当多个文件系统客户机争用对文件系统服务器上同一文件的写访问权时，属性高速缓存可能会导致问题，因为每个客户机使用的高速缓存属性可能与服务器上的那些属性不同。 以此方式访问的文件的一个示例是多实例队列管理器的队列管理器错误日志。 队列管理器错误日志可能由活动队列管理器实例和备用队列管理器实例写入，并且高速缓存的文件属性可能导致在发生文件回滚之前，错误日志的大小大于预期。

要帮助检查文件系统，请运行任务 第 108 页的『验证共享文件系统行为』。 此任务检查共享文件系统是否满足需求 2 和 3。 您需要验证共享文件系统文档中的需求 1，或者通过尝试将数据记录到磁盘。

在写入磁盘时，磁盘故障会导致错误， IBM WebSphere MQ 会将这些错误报告为 "首次故障数据捕获" 错误。 您可以针对操作系统运行文件系统检查程序，以检查共享文件系统是否存在任何磁盘故障。 例如，在 UNIX 和 Linux 平台上，文件系统检查程序称为 fsck。 在 Windows 平台上，文件系统检查程序称为 CHKDSK 或 SCANDISK。

NFS 服务器安全性

注: 应仅将队列管理器数据放在网络文件系统 (NFS) 服务器上。 在 NFS 上，将以下三个选项与 mount 命令配合使用以使系统安全:

诺埃克
通过使用此选项，可以阻止二进制文件在 NFS 上运行，这将阻止远程用户在系统上运行不需要的代码。

诺苏伊德
通过使用此选项，可防止使用 set-user-identifier 和 set-group-identifier 位，这将阻止远程用户获取更高的特权。

节点 v
通过使用此选项，可以停止字符并阻止使用或定义特殊设备，这将阻止远程用户走出 chroot 监狱。

验证共享文件系统行为

运行 **amqmfsck** 以检查 UNIX 系统上的共享文件系统是否满足存储多实例队列管理器的队列管理器数据的需求。 与 **amqmfsck** 并行运行 IBM WebSphere MQ MQI client 样本程序 **amqsfhac** 以演示队列管理器在故障期间维护消息完整性。

开始之前

您需要一个具有联网存储器的服务器，以及另外两个安装了 WebSphere MQ 的连接到该服务器的服务器。 您必须具有管理员 (root) 权限才能配置文件系统，并且必须是 WebSphere MQ 管理员才能运行 **amqmfsck**。

关于此任务

第 106 页的『共享文件系统的需求』 描述了将共享文件系统与多实例队列管理器配合使用的文件系统需求。 IBM WebSphere MQ 技术说明 针对 WebSphere MQ 多实例队列管理器的测试和支持语句 列出了 IBM 已使用的共享文件系统。 本任务中的过程描述了如何测试文件系统以帮助您评估未列出的文件系统是否保持数据完整性。

多实例队列管理器的故障转移可由硬件或软件故障触发，包括阻止队列管理器写入其数据或日志文件的网络问题。 主要是您有兴趣在文件服务器上引起故障。 但是，您还必须导致 IBM WebSphere MQ 服务器失败，以测试是否成功释放了任何锁定。 要对共享文件系统充满信心，请测试以下所有故障以及特定于您的环境的任何其他故障:

1. 在文件服务器上关闭操作系统，包括同步磁盘。
2. 正在停止文件服务器上的操作系统而不同步磁盘。
3. 按每个服务器上的重置按钮。
4. 从每个服务器拔出网络电缆。
5. 从每个服务器拔出电源线。
6. 关闭每个服务器。

在要用于共享队列管理器数据和日志的网络存储器上创建目录。 目录所有者必须是 WebSphere MQ 管理员，换言之，是 UNIX 上 `mqm` 组的成员。 运行测试的用户必须具有 WebSphere MQ 管理员权限。

使用在 在 Linux 上创建多实例队列管理器 中导出和安装文件系统的示例来帮助您配置文件系统。 不同的文件系统要求执行不同的配置步骤。 请阅读文件系统文档。

过程

在每个检查中，当文件系统检查程序正在运行时，将导致先前列表中的所有故障。 如果您打算与 **amqmfsck** 同时运行 **amqsfhac** ，请执行与此任务并行的任务 第 112 页的『运行 amqsfhac 以测试消息完整性』 。

1. 将导出的目录安装在两个 IBM WebSphere MQ 服务器上。

   在文件系统服务器上，创建共享目录 `shared` 和子目录以保存多实例队列管理器 `qmdata` 的数据。 有关在 Linux 上为多实例队列管理器设置共享目录的示例，请参阅 在 Linux 上创建多实例队列管理器 中的 示例
2. 检查基本文件系统行为。

   在一个 IBM WebSphere MQ 服务器上，运行不带参数的文件系统检查程序。

   ```
   amqmfsck /shared/qmdata
   ```

   图 *40:* 在 *IBM WebSphere MQ* 服务器 *1* 上
3. 检查是否同时从两个 IBM WebSphere MQ 服务器写入同一目录。

   在两个 IBM WebSphere MQ 服务器上，使用 `-c` 选项同时运行文件系统检查程序。

```
amqmfsck -c /shared/qmdata
```

图 *41:* 在 *IBM WebSphere MQ* 服务器 *1* 上

```
amqmfsck -c /shared/qmdata
```

图 *42:* 在 *IBM WebSphere MQ* 服务器 *2* 上

4. 检查是否正在等待和释放两个 IBM WebSphere MQ 服务器上的锁定。

   在两个 IBM WebSphere MQ 服务器上，使用 -w 选项同时运行文件系统检查程序。

```
amqmfsck -w /shared/qmdata
```

图 *43:* 在 *IBM WebSphere MQ* 服务器 *1* 上

```
amqmfsck -w /shared/qmdata
```

图 *44:* 在 *IBM WebSphere MQ* 服务器 *2* 上

5. 检查数据完整性。

   a) 格式化测试文件。

   在要测试的目录中创建大文件。 文件已格式化，以便后续阶段可以成功完成。 该文件必须足够大，以便有足够的时间来中断第二阶段以模拟故障转移。 尝试缺省值 262144 页 (1 GB)。 程序会自动减少慢速文件系统上的此缺省值，以便在大约 60 秒内完成格式化

```
amqmfsck -f /shared/qmdata
```

   服务器通过以下消息进行响应:

```
Formatting test file for data integrity test.
Test file formatted with 262144 pages of data.
```

   图 *45:* 在 *IBM WebSphere MQ* 服务器 *1* 上

   b) 在导致失败时，使用文件系统检查程序将数据写入测试文件。

   同时在两台服务器上运行测试程序。 在将要经历故障的服务器上启动测试程序，然后在将要经历故障的服务器上启动测试程序。 导致您正在调查的故障。

   第一个测试程序停止，并显示错误消息。 第二个测试程序获取对测试文件的锁定，并从第一个测试程序离开的位置开始将数据写入测试文件。 让第二个测试程序运行到完成。

表 *12:* 同时在两台服务器上运行数据完整性检查

IBM WebSphere MQ 服务器 1	**IBM WebSphere MQ 服务器 2**
`amqmfsck -a /shared/qmdata`	

表 *12:* 同时在两台服务器上运行数据完整性检查 *(*继续*)*

IBM WebSphere MQ 服务器 1	**IBM WebSphere MQ 服务器 2**
	`amqmfsck -a /shared/qmdata`

`Please start this program on a second machine`
`with the same parameters.`
`File lock acquired.`
`Start a second copy of this program with the same parameters on another server.`
 	
`Writing data into test file.`
 	
`To increase the effectiveness of the test,`
`interrupt the writing by ending the process,`
`temporarily breaking the network connection`
`to the networked storage,`
`rebooting the server of turning off the power.`
`To increase the effectiveness of the test,`
`interrupt the writing by ending the process,`
`temporarily breaking the network connection`
`to the networked storage,`
`rebooting the server of turning off the power.`
	`Waiting for lock...`
	`Waiting for lock...`
	`Waiting for lock...`
	`Waiting for lock...`
	`Waiting for lock...`
	`Waiting for lock...`

Turn the power off here	

	`File lock acquired.`
	`Reading test file`
	`Checking the integrity of the data read.`
	`Appending data into the test file`
	`after data already found.`
	`The test file is full of data.`
	`It is ready to be inspected for data integrity.`

测试的时间取决于文件系统的行为。 例如，在断电后，文件系统通常需要 30 到 90 秒来释放由第一个程序获取的文件锁定。 如果在第一个测试程序填充该文件之前，您没有太多时间来引入该故障，请使用 **amqmfsck** 的 -x 选项来删除该测试文件。 使用更大的测试文件从头开始尝试测试。

c) 验证测试文件中数据的完整性。

```
amqmfsck -i /shared/qmdata
```

服务器通过以下消息进行响应:

```
File lock acquired
Reading test file checking the integrity of the data read.
The data read was consistent.
The tests on the directory completed successfully.
```

图 *46:* 在 *IBM WebSphere MQ* 服务器 *2* 上

6. 删除测试文件。

```
amqmfsck -x /shared/qmdata
Test files deleted.
```

图 *47:* 在 *IBM WebSphere MQ* 服务器 *2* 上

服务器通过以下消息进行响应:

```
Test files deleted.
```

结果

如果测试成功完成，那么程序返回的退出代码为零，否则返回非零。

示例

第一组三个示例显示了生成最小输出的命令。

在一台服务器上成功测试基本文件锁定

```
> amqmfsck /shared/qmdata
The tests on the directory completed successfully.
```

在一个服务器上测试基本文件锁定失败

```
> amqmfsck /shared/qmdata
AMQ6245: Error Calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck' error '2'.
```

在两台服务器上成功进行锁定测试

表 *13:* 在两台服务器上成功锁定

IBM WebSphere MQ 服务器 1	**IBM WebSphere MQ 服务器 2**

`> amqmfsck -w /shared/qmdata`
`Please start this program on a second`
`machine with the same parameters.`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

	`> amqmfsck -w /shared/qmdata` `Waiting for lock...`
`[ Return pressed ]` `Lock released.`	

	`Lock acquired.`
	`The tests on the directory completed`
	`successfully`

第二组三个示例显示了使用详细方式的相同命令。

在一台服务器上成功测试基本文件锁定

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")'
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd1 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd1, F_SETLK, F_RDLCK)
System call: fd2 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd2, F_SETLK, F_RDLCK)
System call: close(fd2)
System call: write(fd1)
```

```
System call: close(fd1)
The tests on the directory completed successfully.
```

在一个服务器上测试基本文件锁定失败

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_RDLCK)
System call: fdSameFile = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fdSameFile, F_SETLK, F_RDLCK)
System call: close(fdSameFile)
System call: write(fd)
AMQxxxx: Error calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck', errno 2
(Permission denied).
```

在两台服务器上成功进行锁定测试

表 *14:* 在两个服务器上成功锁定-详细方式

IBM WebSphere MQ 服务器 1	**IBM WebSphere MQ 服务器 2**

`> amqmfsck -wv /shared/qmdata`
`Calling 'stat("/shared/qmdata")'`
`Calling 'fd = open("/shared/qmdata/`
`amqmfsck.lkw",`
`O_EXCL | O_CREAT | O_RDWR, 0666)'`
`Calling 'fchmod(fd, 0666)'`
`Calling 'fstat(fd)'`
`Please start this program on a second`
`machine with the same parameters.`
`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

	`> amqmfsck -wv /shared/qmdata`
	`Calling 'stat("/shared/qmdata")'`
	`Calling 'fd = open("/shared/qmdata/`
	`amqmfsck.lkw",`
	`O_EXCL | O_CREAT | O_RDWR,0666)'`
	`Calling 'fd = open("/shared/qmdata/amqmfsck.lkw,`
	`O_RDWR, 0666)'`
	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)`
	`'Waiting for lock...`

`[ Return pressed ]`
`Calling 'close(fd)'`
`Lock released.`	

	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
	`Lock acquired.`
	`The tests on the directory completed`
	`successfully`

相关参考

amqmfsck（文件系统检查）

运行 ***amqsfhac*** 以测试消息完整性

amqsfhac 将检查使用联网存储器的队列管理器在发生故障后是否维护数据完整性。

开始之前

此测试需要四个服务器。 两个服务器用于多实例队列管理器，一个用于文件系统，另一个用于将 **amqsfhac** 作为 IBM WebSphere MQ MQI client 应用程序运行。

遵循 过程 中的步骤 第 108 页的『1』，为多实例队列管理器设置文件系统。

关于此任务

过程

1. 使用您在 过程中的步骤 第 108 页的『1』 中创建的文件系统，在另一服务器 QM1 上创建多实例队列管理器。

   请参阅 创建多实例队列管理器。
2. 在两个服务器上启动队列管理器，使其具有高可用性。

   在服务器 1 上:

   ```
   strmqm -x QM1
   ```

   在服务器 2 上:

   ```
   strmqm -x QM1
   ```
3. 设置客户机连接以运行 **amqsfhac**。

   a) 使用 验证客户机安装 中的过程来设置客户机连接，或者使用 可重新连接的客户机样本 中的示例脚本。

   b) 修改客户机通道以具有两个 IP 地址，对应于运行 QM1 的两个服务器。

   在示例脚本中，修改:

   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) +
   CONNAME('LOCALHOST(2345)') QMNAME(QM1) REPLACE
   ```

   到:

   ```
   DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)  TRPTYPE(TCP) +
   CONNAME('server1(2345),server2(2345)') QMNAME(QM1) REPLACE
   ```

   其中 server1 和 server2 是两个服务器的主机名， 2345 是通道侦听器正在侦听的端口。 通常，此值缺省为 1414。 您可以将 1414 与缺省侦听器配置配合使用。
4. 在 QM1 上为测试创建两个本地队列。
   运行以下 MQSC 脚本:

   ```
   DEFINE QLOCAL(TARGETQ) REPLACE
   DEFINE QLOCAL(SIDEQ) REPLACE
   ```
5. 使用 **amqsfhac** 测试配置

   ```
   amqsfhac QM1 TARGETQ SIDEQ 2 2 2
   ```
6. 测试文件系统完整性时测试消息完整性。

   在 过程的步骤 第 109 页的『5』 期间运行 **amqsfhac** 。

   ```
   amqsfhac QM1 TARGETQ SIDEQ 10 20 0
   ```

   如果停止活动队列管理器实例，那么 **amqsfhac** 将在另一个队列管理器实例处于活动状态后重新连接到该实例。 再次重新启动已停止的队列管理器实例，以便您可以在下一次测试中反转故障。 您可能需要根据对环境的试验增加迭代次数，以便测试程序运行足够的时间进行故障转移。

结果

第 114 页的图 48 中显示了在步骤 第 113 页的『6』 中运行 **amqsfhac** 的示例。 测试是成功的。

如果测试检测到问题，那么输出将报告故障。 在某些测试中， MQRC_CALL_INTERRUPTED 可能会报告 “Resolving to backed out”。 它与结果没有区别。 结果取决于是在发生故障之前还是之后由联网文件存储器落实对磁盘的写入。

```
Sample AMQSFHAC start
qmname = QM1
qname = TARGETQ
sidename = SIDEQ
transize = 10
iterations = 20
verbose = 0
Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Resolving MQRC_CALL_INTERRUPTED
MQGET browse side tranid=14 pSideinfo->tranid=14
Resolving to committed
Iteration 7
Iteration 8
Iteration 9
Iteration 10
Iteration 11
Iteration 12
Iteration 13
Iteration 14
Iteration 15
Iteration 16
Iteration 17
Iteration 18
Iteration 19
Sample AMQSFHAC end
```

图 *48:* 成功运行 ***amqsfhac*** 的输出

相关参考

“高可用性”样本程序

共享 IBM WebSphere MQ 文件

某些 IBM WebSphere MQ 文件由活动队列管理器独占访问，其他文件共享。

WebSphere MQ 文件拆分为程序文件和数据文件。 程序文件通常本地安装在运行 WebSphere MQ 的每个服务器上。 队列管理器共享对缺省数据目录中的数据文件和目录的访问权。 他们需要对包含在 第 115 页的图 49 中显示的每个 qmgrs 和 log 目录中的自己的队列管理器目录树进行独占访问。

第 115 页的图 49 是 WebSphere MQ 目录结构的高级视图。 它显示了可以在队列管理器之间共享并使其成为远程目录的目录。 详细信息因平台而异。 虚线指示可配置路径。

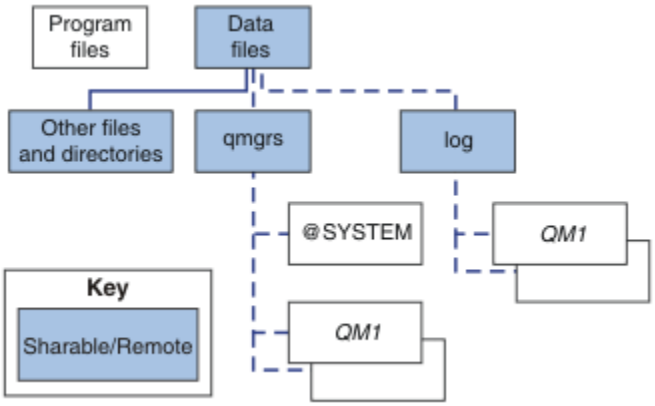


图 *49: WebSphere MQ* 目录结构的整体视图

程序文件

程序文件目录通常保留在缺省位置，是本地的，并且由服务器上的所有队列管理器共享。

数据文件

数据文件目录通常位于缺省位置 /var/mqm (在 UNIX and Linux 系统上) 本地，并且可在 Windows 上安装时进行配置。 它在队列管理器之间共享。 您可以使缺省位置成为远程位置，但不要在 WebSphere MQ 的不同安装之间共享该位置。 WebSphere MQ 配置中的 DefaultPrefix 属性指向此路径。

qmgrs

从 v7.0.1 开始，可以通过两种替代方法来指定队列管理器数据的位置。

使用 前缀

Prefix 属性指定 qmgrs 目录的位置。 WebSphere MQ 根据队列管理器名称构造队列管理器目录名称，并将其创建为 qmgrs 目录的子目录。

Prefix 属性位于 QueueManager 节中，并从 DefaultPrefix 属性中的值继承。 缺省情况下，为了简化管理，队列管理器通常共享同一 qmgrs 目录。

QueueManager 节位于 mqs.ini 文件中。

如果更改任何队列管理器的 qmgrs 目录的位置，那么必须更改其 Prefix 属性的值。

UNIX and Linux 平台的 第 115 页的图 49 中 QM1 目录的 Prefix 属性为:

```
Prefix=/var/mqm
```

使用 DataPath

DataPath 属性指定队列管理器数据目录的位置。

DataPath 属性指定完整路径，包括队列管理器数据目录的名称。 DataPath 属性与 Prefix 属性不同，后者指定队列管理器数据目录的不完整路径。

DataPath 属性 (如果已指定) 位于 QueueManager 节中。 如果已指定，那么它优先于 Prefix 属性中的任何值。

QueueManager 节位于 mqs.ini 文件中。

如果更改任何队列管理器的队列管理器数据目录的位置，那么必须更改 DataPath 属性的值。

对于 UNIX 或 Linux 平台， 第 115 页的图 49 中 QM1 目录的 DataPath 属性为:

```
DataPath=/var/mqm/qmgrs/QM1
```

log

将在队列管理器配置的 Log 节中为每个队列管理器单独指定日志目录。 队列管理器配置在 qm.ini 中。

DataPath/*QmgrName*/@IPCC 子目录

DataPath/*QmgrName*/@IPCC 子目录位于共享目录路径中。 它们用于构造 IPC 文件系统对象的目录路径。 当在系统之间共享队列管理器时，它们需要区分队列管理器的名称空间。 在 V7.0.1 之前，仅在一个系统上使用了队列管理器。 一组子目录足以定义 IPC 文件系统对象的目录路径，请参阅 第 116 页的图 50。

```
DataPath/QmgrName/@IPCC/esem
```

图 *50:* 示例 *IPC* 子目录 *pre-V7.0.1*

在 V7.0.1 和更高版本中， IPC 文件系统对象必须由系统进行区分。 对于运行队列管理器的每个系统，会将一个子目录添加到目录路径中，请参阅 第 116 页的图 51。

```
DataPath/QmgrName/@IPCC/esem/myHostName/
```

图 *51:* 示例 *IPC* 子目录 *V7.0.1* 和后续发行版

myHostName 最多包含操作系统返回的主机名的前 20 个字符。 在某些系统上，在截断之前，主机名的长度可能最多为 64 个字符。 生成的 *myHostName* 值可能由于以下两个原因导致问题:

1. 前 20 个字符不唯一。
2. 主机名由 DHCP 算法生成，该算法并不总是将相同的主机名分配给系统。

在这些情况下，请使用环境变量 MQC_IPC_HOST 设置 *myHostName* ; 请参阅 第 116 页的图 52。

```
export MQS_IPC_HOST=myHostName
```

图 *52:* 示例: 设置 *MQC_IPC_HOST*

其他文件和目录

其他文件和目录 (例如包含跟踪文件的目录和公共错误日志) 通常共享并保留在本地文件系统上。

直到 v7.0.1 之前， WebSphere MQ 依靠外部管理来保证队列管理器对队列管理器数据和日志文件的独占访问权。 从 v7.0.1 开始，通过支持共享文件系统， WebSphere MQ 使用文件系统锁定来管理对这些文件的独占访问。 文件系统锁定一次仅允许特定队列管理器的一个实例处于活动状态。

当您启动特定队列管理器的第一个实例时，它将获取其队列管理器目录的所有权。 如果启动第二个实例，那么仅当第一个实例已停止时，才能获取所有权。 如果第一个队列管理器仍在运行，那么第二个实例无法启动，并报告队列管理器正在其他位置运行。 如果第一个队列管理器已停止，那么第二个队列管理器将接管队列管理器文件的所有权并成为正在运行的队列管理器。

您可以自动执行第二个队列管理器从第一个队列管理器接管的过程。 使用 `strmqm -x` 选项启动第一个队列管理器，该选项允许另一个队列管理器从该队列管理器接管。 然后，第二个队列管理器将等待队列管理器文件解锁，然后再尝试接管队列管理器文件的所有权，然后启动。

UNIX and Linux 系统上的目录结构

UNIX and Linux 系统上的 WebSphere MQ 目录结构可以映射到不同的文件系统，以实现更简单的管理，更好的性能和更好的可靠性。

使用 WebSphere MQ 的灵活目录结构来利用共享文件系统来运行多实例队列管理器。

使用命令 **crtmqm** *QM1* 来创建 第 117 页的图 53 中显示的目录结构，其中 R 是产品的发行版。 它是从 v7.0.1 开始在 WebSphere MQ 系统上创建的队列管理器的典型目录结构。 为了清晰起见，省略了某些目录，文件和 .ini 属性设置，另一个队列管理器名称可能会被修改。 文件系统的名称在不同的系统上会有所不同。

在典型安装中，您创建的每个队列管理器都指向本地文件系统上的公共 `log` 和 `qmgrs` 目录。 在多实例配置中， `log` 和 `qmgrs` 目录位于与另一个 WebSphere MQ 安装共享的网络文件系统上。

第 117 页的图 53 显示了 AIX 上 WebSphere MQ v7.R 的缺省配置，其中 R 是产品的发行版。 有关备用多实例配置的示例，请参阅 第 121 页的『UNIX and Linux 系统上的示例目录配置』。

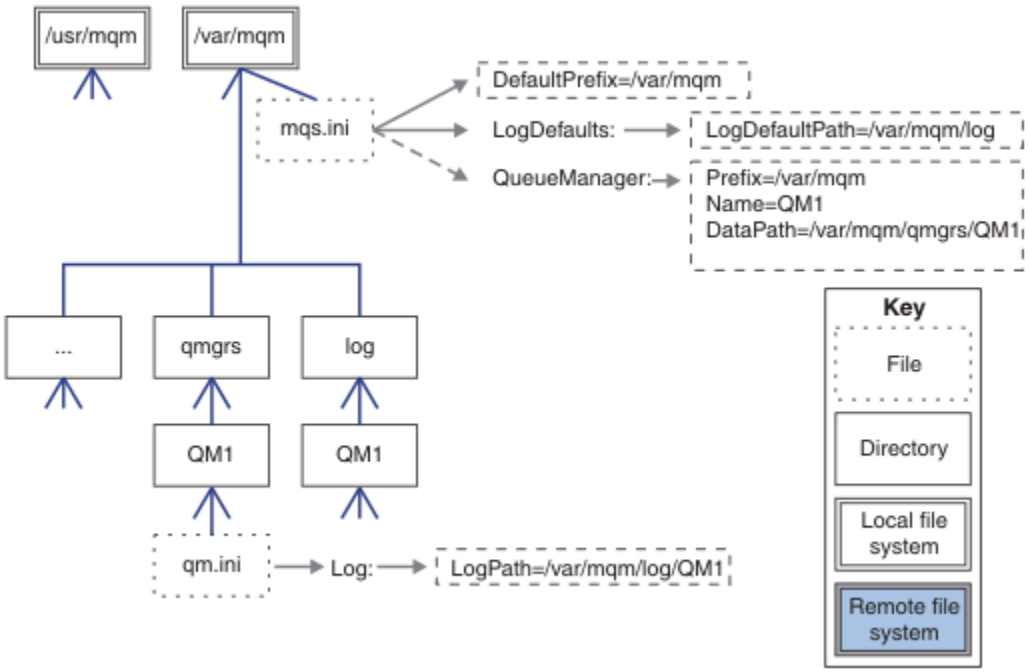


图 *53: UNIX and Linux* 系统的缺省 *WebSphere MQ v7.R* 目录结构示例

缺省情况下，该产品安装到 AIX 上的 /usr/mqm 和其他系统上的 /opt/mqm 中。 工作目录将安装到 /var/mqm 目录中。

注: 如果在安装 IBM WebSphere MQ 之前创建了 /var/mqm 文件系统，请确保 mqm 用户具有完整目录许可权，例如，文件方式 755。

log 和 qmgrs 目录显示在其缺省位置中，如 mqs.ini 文件中的 LogDefault **路径** 和 DefaultPrefix 属性的缺省值所定义。 创建队列管理器时，缺省情况下将在 *DefaultPrefix*/qmgrs 中创建队列管理器数据目录，并在 *LogDefaultPath* /log 中创建日志文件目录。 LogDefault **路径** 和 DefaultPrefix 仅影响缺省情况下创建队列管理器和日志文件的情况。 队列管理器目录的实际位置保存在 mqs.ini 文件中，日志文件目录的位置保存在 qm.ini 文件中。

队列管理器的日志文件目录在 LogPath 属性的 qm.ini 文件中定义。 在 **crtmqm** 命令上使用 -ld 选项来设置队列管理器的 LogPath 属性; 例如， **crtmqm** -ld *LogPath* QM1。 如果省略 ld 参数，那么将改为使用 LogDefaultPath 的值。

队列管理器数据目录在 mqs.ini 文件的 QueueManager 节中的 DataPath 属性中定义。 在 **crtmqm** 命令上使用 -md 选项为队列管理器设置 DataPath ; 例如， **crtmqm** - md *DataPath* QM1 。 如果省略 md 参数，那么将改为使用 DefaultPrefix 或 Prefix 属性的值。 **前缀** 优先于 DefaultPrefix 。

通常，在单个命令中同时指定日志和数据目录来创建 QM1 。

```
crtmqm
 -md DataPath -ld
 LogPath QM1
```

当队列管理器停止时，您可以通过编辑 qm.ini 文件中的 DataPath 和 LogPath 属性来修改现有队列管理器的队列管理器日志和数据目录的位置。

errors 目录的路径 (与 /var/mqm 中所有其他目录的路径一样) 不可修改。 但是，这些目录可以安装在不同的文件系统上，也可以象征性地链接到不同的目录。

UNIX and Linux 系统上的目录内容

与队列管理器关联的目录的内容。

有关产品文件位置的信息，请参阅 选择安装位置

有关备用目录配置的信息，请参阅 第 106 页的『规划文件系统支持』。

在 第 119 页的图 54 中，在队列管理器已使用一段时间之后，布局代表 WebSphere MQ 。 您拥有的实际结构取决于在队列管理器上执行的操作。

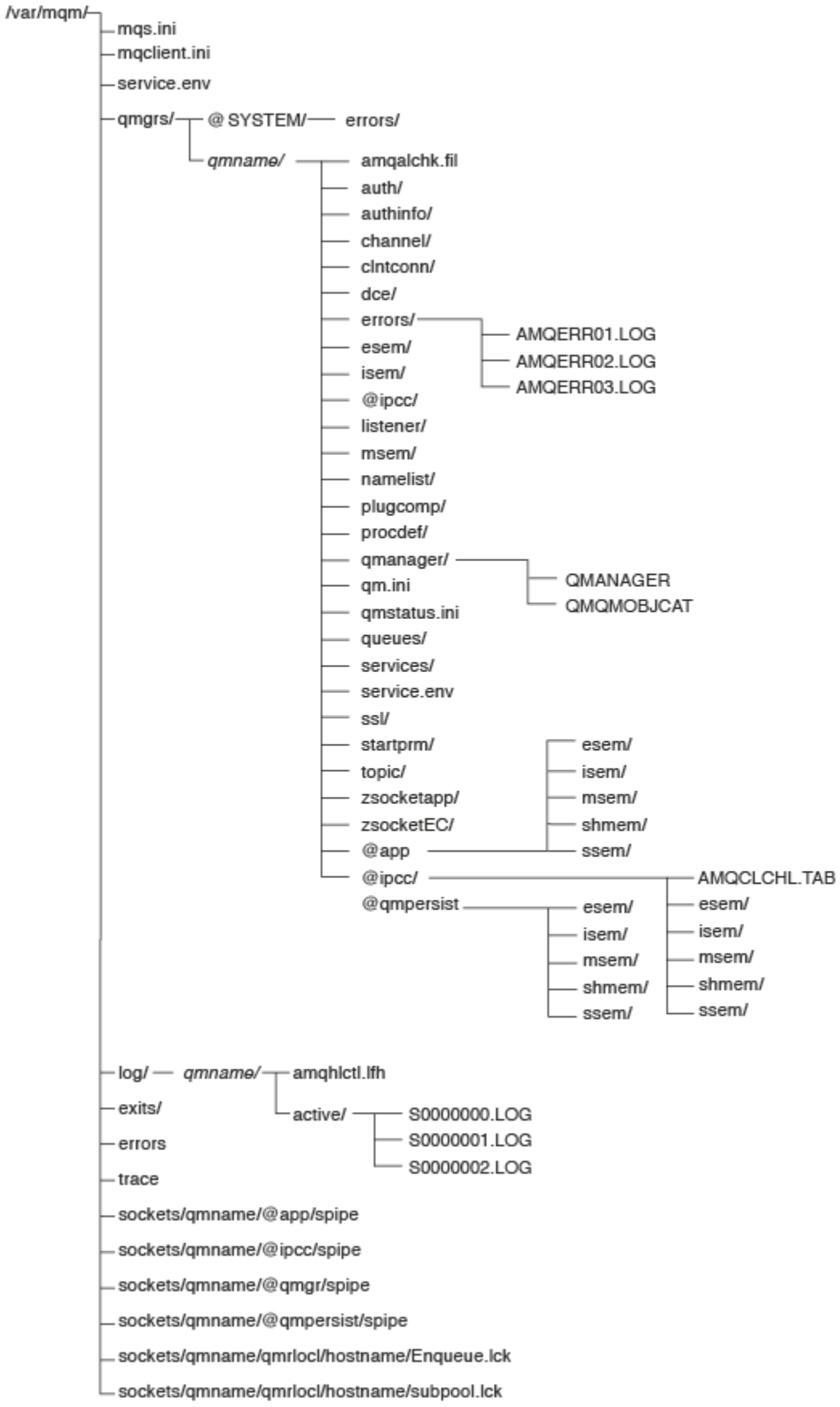


图 *54:* 启动队列管理器后的缺省目录结构 *(UNIX* 系统*)*

/var/mqm/

/var/mqm 目录包含应用于整个 WebSphere MQ 安装而不是单个队列管理器的配置文件和输出目录。

表 15: UNIX 系统上 /var/mqm 目录的已记录内容	
mqs.ini	WebSphere MQ 安装范围配置文件，在队列管理器启动时读取。 可使用 **AMQ_MQS_INI_LOCATION** 环境变量修改文件路径。 确保在运行 **strmqm** 命令的 shell 中设置并导出此命令。
mqclient.ini	WebSphere MQ MQI 客户机程序读取的缺省客户机配置文件。 可使用 **MQCLNTCF** 环境变量修改文件路径。
service.env	包含服务进程的机器作用域环境变量。 文件路径已固定。
个错误/	机器作用域错误日志和 FFST 文件。 目录路径已修正。 另请参阅 FFST: IBM WebSphere MQ for UNIX and Linux systems。
套接字/	包含每个队列管理器的信息，仅供系统使用。
跟踪/	跟踪文件。 目录路径已修正。
个出口/ **exits64/**	包含用户通道出口程序的缺省目录。 可在 mqs.ini 文件的 ApiExit 节中修改位置。

/var/mqm/qmgrs/*qmname*/

/var/mqm/qmgrs/*qmname*/ 包含队列管理器的目录和文件。 该目录已锁定，以供活动队列管理器实例独占访问。 可以在 mqs.ini 文件中直接修改目录路径，也可以使用 **crtmqm** 命令的 **md** 选项进行修改。

表 16: UNIX 系统上 /var/mqm/qmgrs/qmname 目录的已记录内容	
qm.ini	队列管理器配置文件，在队列管理器启动时读取。
个错误/	队列管理器作用域错误日志。 *qmname* = @system 包含未知或不可用队列管理器的通道相关消息。
@ipcc/ **AMQCLCHL.TAB**	缺省客户机通道控制表，由 WebSphere MQ 服务器创建，并由 WebSphere MQ MQI 客户机程序读取。 可使用 **MQCHLLIB** 和 **MQCHLTAB** 环境变量修改文件路径。
QMANAGER	队列管理器对象文件: QMANAGER 队列管理器对象目录: QMQMOBJCAT

表 16: UNIX 系统上 /var/mqm/qmgrs/qmname 目录的已记录内容 (继续)	
authinfo/	队列管理器中定义的每个对象都与这些目录中的一个文件相关联。 文件名与定义名称大致匹配; 请参阅 了解 WebSphere MQ 文件名 。
通道/	
clntconn/	
侦听器/	
名称列表/	
进程/	
队列/	
服务/	
主题/	
...	WebSphere MQ 使用的其他目录 (例如 @ipcc) 将仅由 WebSphere MQ 修改。

/var/mqm/log/*qmname*/

/var/mqm/log/*qmname*/ 包含队列管理器日志文件。 该目录已锁定，以供活动队列管理器实例独占访问。 可以在 qm.ini 文件中修改目录路径，也可以使用 **crtmqm** 命令的 **ld** 选项进行修改。

表 17: UNIX 系统上 /var/mqm/log/qmname 目录的已记录内容	
amqhlctl.lfh	日志控制文件。
活动/	此目录包含编号为 S0000000.LOG， S0000001.LOG， S0000002.LOG，依此类推。

UNIX and Linux 系统上的示例目录配置

UNIX and Linux 系统上备用文件系统配置的示例。

您可以通过各种方式定制 WebSphere MQ 目录结构，以实现许多不同的目标。

- 将 qmgrs 和 log 目录放在远程共享文件系统上以配置多实例队列管理器。
- 将单独的文件系统用于数据和日志目录，并将这些目录分配给不同的磁盘，以通过减少 I/O 争用来提高性能。
- 将更快的存储设备用于对性能影响更大的目录。 物理设备等待时间通常是持久消息传递性能中比本地还是远程安装设备更重要的因素。 以下列表显示哪些目录最敏感和最不敏感的性能。

  1. log
  2. qmgrs
  3. 其他目录，包括 /usr/mqm

- 在分配给具有良好弹性的存储器 (例如冗余磁盘阵列) 的文件系统上创建 qmgrs 和 log 目录。
- 最好将公共错误日志存储在 var/mqm/errors 本地，而不是存储在网络文件系统上，以便可以记录与网络文件系统相关的错误。

第 122 页的图 55 是从中派生备用 WebSphere MQ 目录结构的模板。 在模板中，虚线表示可配置的路径。 在示例中，虚线将替换为与 AMQ_MQS_INI_LOCATION 环境变量以及 mqs.ini 和 qm.ini 文件中存储的配置信息相对应的实线。

注: 路径信息显示在 mqs.ini 或 qm.ini 文件中。 如果在 **crtmqm** 命令中提供路径参数，请省略队列管理器目录的名称: 队列管理器名称在被管理后由 WebSphere MQ 添加到路径中。

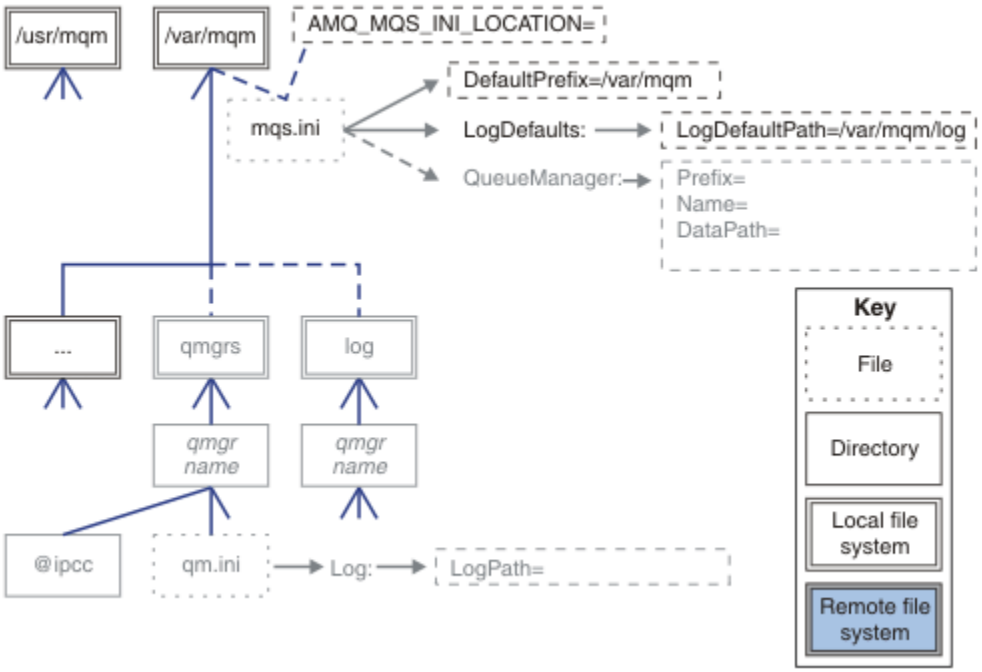


图 *55:* 目录结构模式模板

以下是已配置的目录结构的示例。 第一个示例显示了通过发出 **crtmqm** *QM1* 命令创建的 WebSphere MQ v7.0.1 的典型缺省目录结构。 第二个示例显示如何显示使用低于 v7.0.1 的 WebSphere MQ 发行版创建的队列管理器的典型目录结构。 目录结构未更改。

在 V 7.0.1 中新创建的队列管理器具有与 v7 的较早发行版不同的配置文件。 如果需要除去 v7.0.1 修订包以还原为 v7.0.0.2，那么需要重新创建配置文件。 您可能只需要使用 Prefix 属性来定义新队列管理器数据目录的路径，或者可能需要将队列管理器数据目录和日志目录移至其他位置。 重新配置队列管理器的最安全方法是保存队列管理器数据和日志目录，删除并重新创建队列管理器，然后将其新位置中的数据和日志目录替换为已保存的数据和日志目录。

发行版 v7.0.1 及更高版本的典型目录结构

第 123 页的图 56 是在 v7.0.1 中通过发出命令 **crtmqm** *QM1* 创建的缺省目录结构。

`mqs.ini` 文件具有 QM1 队列管理器的节，此节是通过引用 `DefaultPrefix` 的值创建的。 `qm.ini` 文件中的 `Log` 节具有 `LogPath` 的值，通过引用 `mqs.ini` 中的 `LogDefaultPath` 进行设置。

使用可选 **crtmqm** 参数可覆盖 `DataPath` 和 `LogPath` 的缺省值。

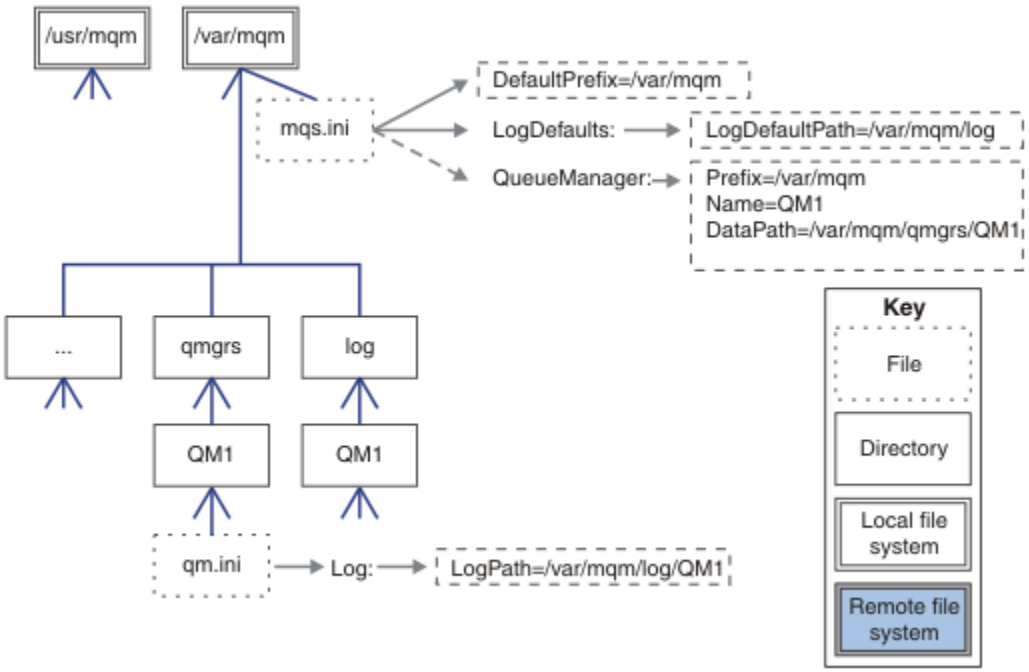


图 *56: UNIX and Linux* 系统的缺省 *WebSphere MQ v7.R* 目录结构示例

低于 v7.0.1 的发行版的典型目录结构

DataPath 属性在 WebSphere MQ v7.0.1; 该属性在 mqs.ini 文件中不存在。 qmgrs 目录的位置是使用 Prefix 属性配置的。 可以使用符号链接来配置各个目录的位置，以指向不同的文件系统位置。

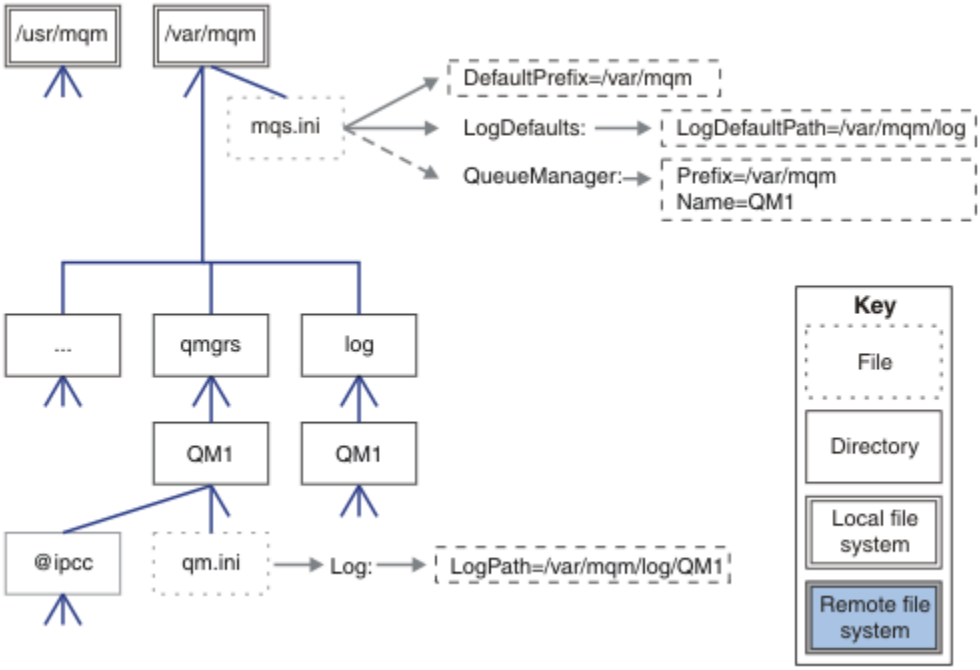


图 *57:* 低于 *v7.0.1* 的发行版的典型目录结构

共享缺省 qmgrs 和 log 目录 (发行版 v7.0.1 及更高版本)

第 125 页的『共享所有内容 (发行版 v7.0.1 及更高版本)』 的替代方法是单独共享 qmgrs 和 log 目录 (第 124 页的图 58)。 在此配置中，无需设置 AMQ_MQS_INI_LOCATION，因为缺省 mqs.ini 存储在本地 /var/mqm 文件系统中。 文件和目录 (例如 mqclient.ini 和 mqserver.ini) 也不会共享。

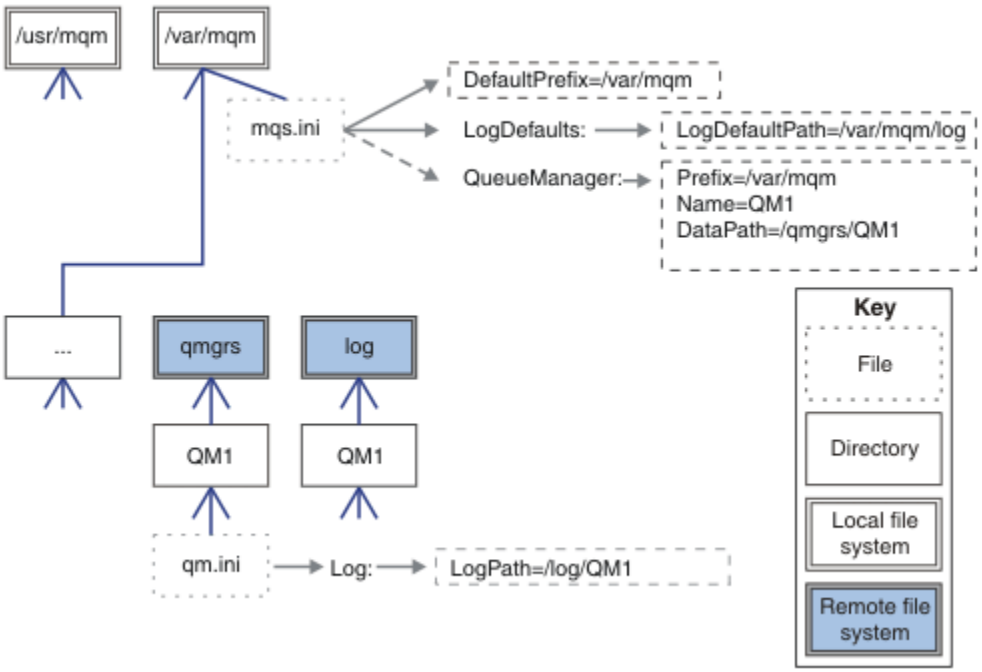


图 *58:* 共享 *qmgrs* 和 *log* 目录

共享名为 qmgrs 和 log 的目录 (发行版 v7.0.1 及更高版本)

第 124 页的图 59 中的配置将 log 和 qmgrs 放置在名为 /ha 的公共命名远程共享文件系统中。 可以通过两种不同的方式创建相同的物理配置。

1. 设置 LogDefaultPath=/ha，然后运行命令 **crtmqm** -md */ha/qmgrs* QM1。 结果与 第 124 页的图 59 中的说明完全相同。
2. 保留缺省路径不变，然后运行命令 **crtmqm** -ld */ha/log* - md */ha/qmgrs* QM1。

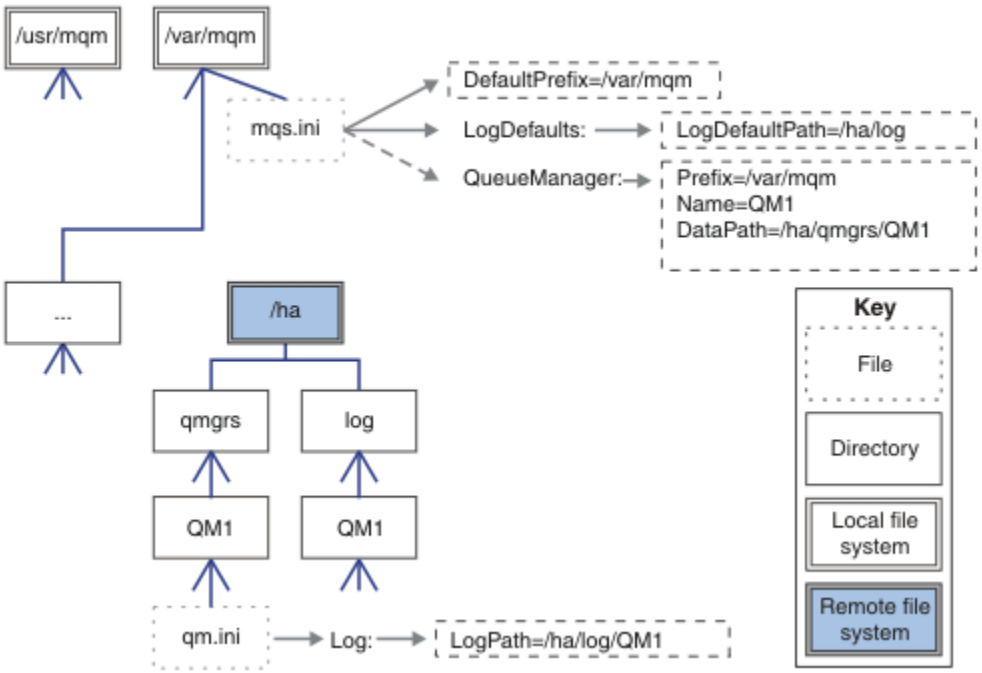


图 *59:* 共享名为 *qmgrs* 和 *log* 的目录

共享所有内容 (发行版 v7.0.1 及更高版本)

第 125 页的图 60 是具有快速联网文件存储器的系统的简单配置。

将 /var/mqm 安装为远程共享文件系统。 缺省情况下，当您启动 QM1 时，它将查找 /var/mqm，在共享文件系统上查找并读取 /var/mqm 中的 mqs.ini 文件。 您可以将每个服务器上的 AMQ_MQS_INI_LOCATION 环境变量设置为指向不同的 mqs.ini 文件，而不是将单个 /var/mqm/mqs.ini 文件用于所有服务器上的队列管理器。

注: /var/mqm/errors/ 中通用错误文件的内容在不同服务器上的队列管理器之间共享。

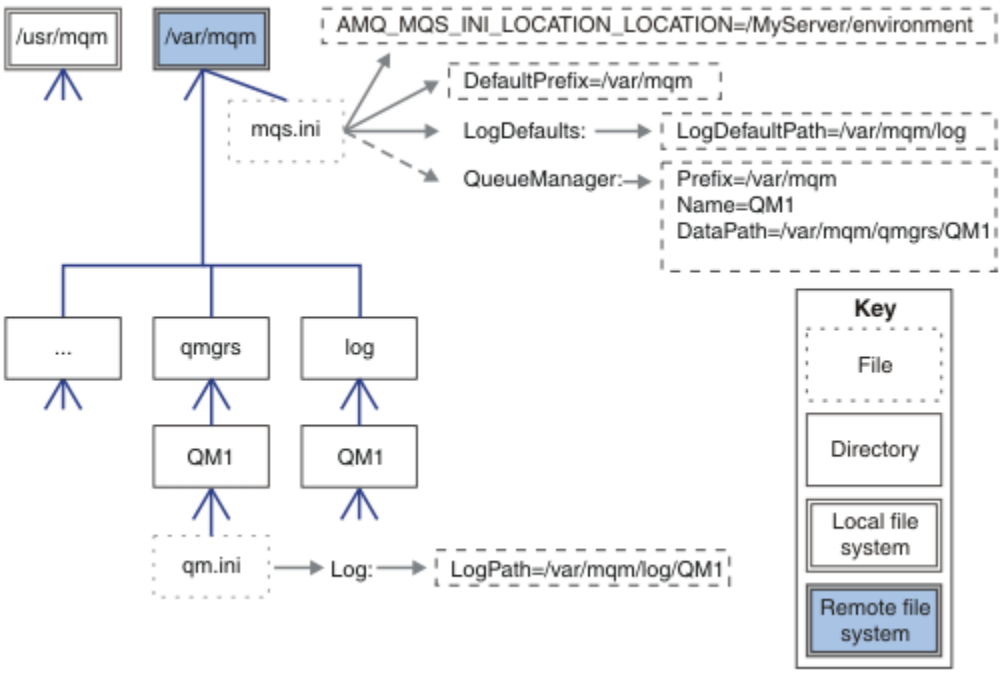


图 *60:* 共享所有内容

请注意，不能将此用于多实例队列管理器。 原因是多实例队列管理器中的每个主机都必须具有其自己的 /var/mqm 本地副本，以跟踪本地数据 (例如信号量和共享内存)。 无法在主机之间共享这些实体。

Windows 系统上的目录结构

如何在 Windows 上查找队列管理器配置信息和目录。

IBM WebSphere MQ for Windows 安装的缺省目录为:

32 位
```
C:\Program Files\IBM\WebSphere MQ
```
64 位
```
C:\Program Files (x86)\IBM\WebSphere MQ
```
安装信息存储在 Windows 注册表中。 存储 IBM WebSphere MQ 信息的注册表键为:

32 位
```
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphere MQ\
```
64 位
```
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\IBM\WebSphere MQ\
```
每个安装都有一个特定的子键:

```
Installation\<InstallationName>\
```

指向 IBM WebSphere MQ 数据目录的路径存储在名为 WorkPath 的字符串值中，日志的缺省目录存储在 LogDefaultPath 中。 队列管理器数据目录是在 *WorkPath*\qmgrs*Qmgrname* 中创建的。 在 *LogDefaultPath**QmgrName* 中创建队列管理器日志。 请参阅第 126 页的图 61。

如果在安装 IBM WebSphere MQ 时定义队列管理器数据和日志目录，那么将使用定制路径信息更新 *WorkPath* 和 *LogDefaultPath* 。

WorkPath 和 *LogDefault* 路径 仅用于创建队列管理器。

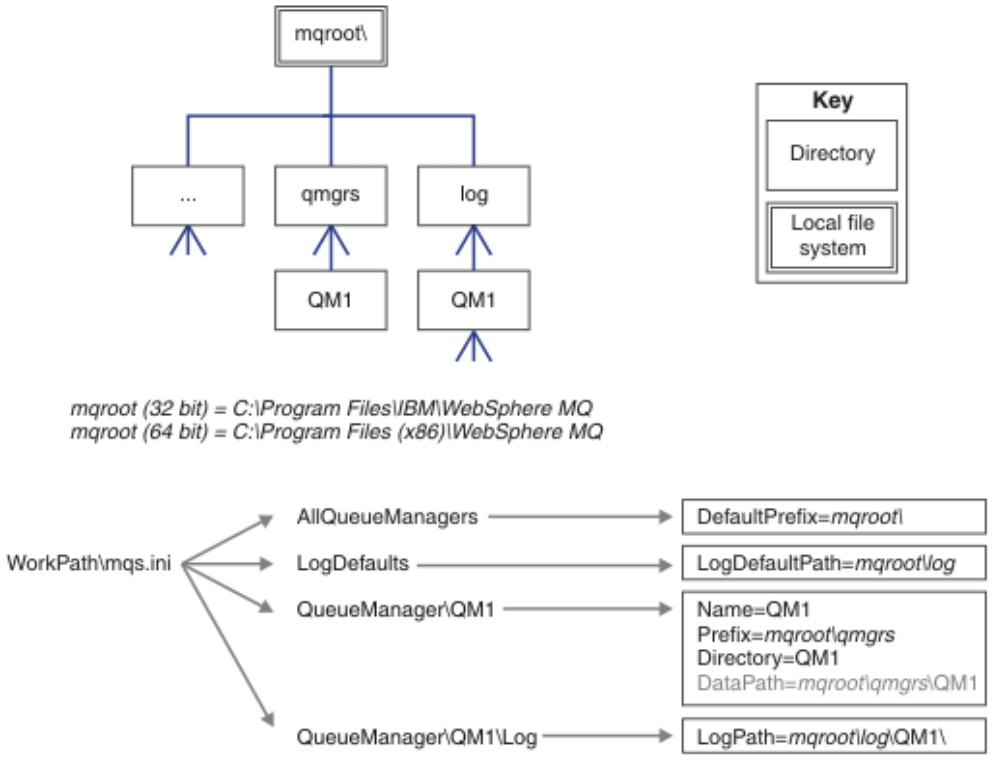


图 *61: Windows* 的缺省目录结构

多实例队列管理器

要配置多实例队列管理器，必须将日志和数据目录放在联网存储器上，最好是放在与正在运行队列管理器实例的任何服务器不同的服务器上。

在 **crtmqm** 命令 **-md** 和 **-ld** 上提供了两个参数，以便更轻松地指定队列管理器数据和日志目录的位置。 指定 **-md** 参数的效果是四重:

1. mqs.ini 节 QueueManager*QmgrName* 包含指向队列管理器数据目录的新变量 *DataPath*。 与 *Prefix* 变量不同，路径包含队列管理器目录的名称。
2. 存储在 mqs.ini 文件中的队列管理器配置信息将缩减为 *Name*， *Prefix*， *Directory* 和 *DataPath*。

目录内容

列出 WebSphere MQ 目录的位置和内容。

WebSphere MQ 配置具有三组主要文件和目录，

1. 仅在应用维护时更新的可执行文件和其他只读文件，例如自述文件， WebSphere MQ Explorer 插件和帮助文件以及许可证文件。 第 127 页的表 18 中描述了这些文件。
2. 可能可修改的文件和目录并非特定于特定队列管理器。 第 127 页的表 19 中描述了这些文件和目录。
3. 特定于服务器上每个队列管理器的文件和目录。 第 127 页的表 18 中描述了这些文件和目录

资源目录和文件

资源目录和文件包含用于运行队列管理器的所有可执行代码和资源。 特定于安装的 IBM WebSphere MQ 配置注册表键中的变量 *FilePath* 包含资源目录的路径。

表 *18: FilePath* 目录中的目录和文件	
文件路径	**内容**
FilePath\bin	命令和 DLL
FilePath\bin64	命令和 DLL (64 位)
FilePath\conv	数据转换表
FilePath\doc	向导帮助文件
FilePath\MQExplorer	Explorer 和 Explorer 帮助 Eclipse 插件
FilePath\gskit8	Global Security Kit
FilePath\java	Java 资源，包括 JRE
FilePath\licenses	许可证信息
FilePath\Non_IBM_License	许可证信息
FilePath\properties	内部使用
FilePath\Tivoli	
FilePath\tools	开发资源和样本
FilePath\Uninst	内部使用
FilePath\README.TXT	自述文件

不特定于队列管理器的目录

某些目录包含并非特定于特定队列管理器的文件，例如跟踪文件和错误日志。 *DefaultPrefix* 变量包含这些目录的路径。 *DefaultPrefix* 是 AllQueueManagers 节的一部分。

表 *19: DefaultPrefix* 目录中的目录和文件	
文件路径	**内容**
DefaultPrefix\Config	内部使用
DefaultPrefix\conv	ccsid.tbl 数据转换控制文件，如 数据转换 中所述
DefaultPrefix\errors	非队列管理器错误日志， AMQERR*nn*.LOG
DefaultPrefix\exits	通道出口程序
DefaultPrefix\exits64	通道出口程序 (64 位)
DefaultPrefix\ipc	未使用
DefaultPrefix\Qmgrs	在 第 128 页的表 20 中描述
DefaultPrefix\trace	跟踪文件
DefaultPrefix\amqmjpse.txt	内部使用

队列管理器目录

创建队列管理器时，将创建一组特定于队列管理器的新目录。

如果使用 **-md** *filepath* 参数创建队列管理器，那么路径将存储在 mqs.ini 文件的队列管理器节中的 *DataPath* 变量中。 如果在不设置 **-md** *filepath* 参数的情况下创建队列管理器，那么将在存储在 *DefaultPrefix* 中的路径中创建队列管理器目录，并将该路径复制到 mqs.ini 文件的队列管理器节中的 *Prefix* 变量中。

表 *20: DataPath* 和 *Prefix/Qmgrs/QmgrName* 目录中的目录和文件

文件路径	**内容**
DataPath\@ipcc	AMQCLCHL.TAB(客户机连接表) 的缺省位置。
DataPath\authinfo	内部使用
DataPath\channel	
DataPath\clntconn	
DataPath\errors	错误日志， AMQERR*nn*.LOG
DataPath\listener	内部使用
DataPath\namelist	
DataPath\plugcomp	
DataPath\procdef	
DataPath\qmanager	
DataPath\queues	
DataPath\services	
DataPath\ssl	
DataPath\startprm	
DataPath\topic	
DataPath\active	
DataPath\active.dat	
DataPath\amqalchk.fil	
DataPath\master	
DataPath\master.dat	
DataPath\qm.ini	队列管理器配置
DataPath\qmstatus.ini	队列管理器状态
Prefix\Qmgrs*QmgrName*	内部使用
Prefix\Qmgrs\@SYSTEM	未使用
Prefix\Qmgrs\@SYSTEM\errors	

IBM WebSphere MQ 和 UNIX System V IPC 资源

队列管理器使用一些 IPC 资源。 使用 **ipcs -a** 来查找正在使用的资源。

此信息仅适用于在 UNIX and Linux 系统上运行的 IBM WebSphere MQ 。

IBM WebSphere MQ 使用 System V 进程间通信 (IPC) 资源 (*semaphores* 和 共享内存段) 在系统组件之间存储和传递数据。 这些资源由连接到队列管理器的队列管理器进程和应用程序使用。 IBM WebSphere MQ MQI 客户机不使用 IPC 资源，但 IBM WebSphere MQ 跟踪控制除外。 使用 UNIX 命令 **ipcs -a** 可获取有关机器上当前正在使用的 IPC 资源的数量和大小的完整信息。

AIX 上的共享内存

如果某些应用程序类型由于 AIX 内存限制而无法连接，那么在大多数情况下，可以通过设置环境变量 EXTSHM=ON 来解决此问题。

AIX 上的某些 32 位进程可能会迁到影响其连接到 WebSphere MQ 队列管理器的能力的操作系统限制。 到 WebSphere MQ 的每个标准连接都使用共享内存，但与其他 UNIX and Linux 平台不同， AIX 仅允许 32 位进程连接 11 个共享内存集。

大多数 32 位进程不会迁到此限制，但是具有高内存需求的应用程序可能无法连接到 WebSphere MQ ，原因码为 2102: MQRC_RESOURCE_PROBLEM。 以下应用程序类型可能会看到此错误:

- 在 32 位 Java 虚拟机中运行的程序
- 使用大型或超大型内存模型的程序
- 连接到多个队列管理器或数据库的程序
- 独立连接到共享内存集的程序

AIX 为 32 位进程提供扩展共享内存功能，允许它们连接更多共享内存。 要使用此功能运行应用程序，请在启动队列管理器和程序之前导出环境变量 EXTSHM=ON。 EXTSHM=ON 功能部件在大多数情况下可防止此错误，但它与使用 shmctl 函数的 SHM_SIZE 选项的程序不兼容。

WebSphere MQ MQI 客户机应用程序和所有 64 位进程不受此限制影响。 无论是否已设置 EXTSHM ，它们都可以连接到 WebSphere MQ 队列管理器。

WebSphere MQ 和 UNIX 进程优先级

设置进程优先级 *nice* 值时的良好实践。

此信息适用于仅在 UNIX and Linux 系统上运行的 WebSphere MQ 。

如果在后台运行进程，那么调用 shell 可以为该进程提供更高的 *nice* 值 (从而降低优先级)。 这可能具有一般 WebSphere MQ 性能影响。 在高度紧张的情况下，如果有许多就绪可运行的线程处于较高优先级，而有些线程处于较低优先级，那么操作系统调度特性会使较低优先级的线程失去处理器时间。

与队列管理器 (例如 `runmqlsr`) 关联的独立启动的进程具有与它们关联的队列管理器相同的 *nice* 值是最佳实践。 请确保 shell 未将更高的 *nice* 值分配给这些后台进程。 例如，在 ksh 中，使用 “`set +o bgnice`” 设置来阻止 ksh 提高后台进程的 *nice* 值。 您可以通过检查 “ps -efl” 列表的 *NI* 列来验证正在运行的进程的 *nice* 值。

此外，使用与队列管理器相同的 *nice* 值来启动 WebSphere MQ 应用程序进程。 如果它们使用不同的 *nice* 值运行，那么应用程序线程可能会阻止队列管理器线程，反之亦然，从而导致性能下降。

在 HP Integrity NonStop Server 上规划 IBM WebSphere MQ 客户机环境

规划 IBM WebSphere MQ 环境时，必须考虑 HP Integrity NonStop Server 环境和 HP NonStop TMF。 使用此信息来规划运行 IBM WebSphere MQ Client for HP Integrity NonStop Server 的环境。

在规划 IBM WebSphere MQ Client for HP Integrity NonStop Server 体系结构之前，请先熟悉基本 IBM WebSphere MQ client for HP Integrity NonStop Server 概念，请参阅 IBM WebSphere MQ Client for HP Integrity NonStop Server 技术概述中的主题。

准备 HP Integrity NonStop Server 环境

在安装之前，必须根据是否要立即验证安装来准备环境。

对于安装，您需要以下项:

- 满足需求的用户标识。 有关用户标识需求的详细信息，请参阅 在 HP Integrity NonStop Server 上设置用户和组。
- OSS 和监护器文件系统中可用于安装文件的已验证位置。
- 可操作的 OSS shell 和 OSS 文件系统。 您可以通过执行以下任务来验证文件系统:

– 登录到 OSS 环境 (shell)。 确保您对打算使用的 OSS 安装根目录具有写访问权。
– 使用 MQM 组中的用户标识登录到 TACL 环境。 请验证您打算使用的卷是否满足需求并且可供您访问，以及该子卷是否不存在。

您可以使用别名 (如果有) 或完整主体登录到 OSS 或 TACL。

如果您打算立即继续验证安装是否可用，那么可能还需要以下可选项:

- OSS 环境中可操作且可访问的本地套接字子系统。
- 可操作的 TCP/IP 子系统。

如果您打算使用 TMF 协调全局工作单元，那么将需要以下项:

- 可操作的 TMF 子系统。
- 可操作的 Pathway (TS/MP) 子系统。

如果您对这些关键子系统的状态有任何疑问，请与系统管理员一起工作。

IBM WebSphere MQ 和 HP NonStop TMF

HP Integrity NonStop Server 的 IBM WebSphere MQ 客户机可以参与 HP NonStop 事务管理设施 (HP NonStop TMF) 协调工作单元。 仅当队列管理器位于 IBM WebSphere MQ Version 7.1 或更高版本时，才支持使用 HP NonStop TMF 来协调事务。

IBM WebSphere MQ 提供的 TMF/Gateway 将事务从 TMF 协调转换为 eXtended 体系结构 (XA) 事务协调，以与远程队列管理器进行通信。 IBM WebSphere MQ 提供的 TMF/Gateway 是 TMF 与队列管理器事务之间的网桥，使用 HP NonStop TMF 提供的服务，并且已设计为在 Pathway 环境中运行。

HP NonStop TMF 软件在严苛的环境中提供事务保护和数据库一致性。 有关 HP NonStop TMF 的更多信息，请参阅 HP NonStop TMF 简介。

有关如何配置 IBM WebSphere MQ 提供的 TMF/Gateway 的信息，请参阅 配置 HP Integrity NonStop Server。

使用 HP NonStop TMF

HP NonStop Transaction Management Facility (TMF) 是 HP Integrity NonStop Server 上的本机事务管理器，与文件系统和关系数据库管理器， SQL/MP 和 SQL/MX 集成。

HP Integrity NonStop Server 的 IBM WebSphere MQ 客户机可以使用 TMF 来协调全局工作单元。

要协调全局工作单元， TMF 充当事务管理器，应用程序必须使用 TMF 提供的 API 来启动，落实和回退全局工作单元。 应用程序通过调用 BEGINTRANSACTION 来启动全局工作单元，然后通过在同步点控制中发出 MQPUT， MQPUT1 和 MQGET 调用来更新全局工作单元中的 IBM WebSphere MQ 资源。 然后，应用程序可以通过调用 ENDTRANSACTION 来落实全局工作单元，或者通过调用 ABORTTRANSACTION 来回退全局工作单元。

使用 TMF 事务的应用程序在任何时候都只能主动处理一个事务，但是使用 RESUMETRANSACTION 允许应用程序从一个活动事务切换到另一个活动事务，或者与没有 TMF 事务关联，而不完成或中止先前活动的事务。 对 MQPUT， MQPUT1 或 MQGET 的任何调用都在当前活动的 TMF 事务 (如果存在) 或本地工作单元 (如果不存在) 下进行。 因此，必须在应用程序中小心，以确保在正确的工作单元中进行这些调用。

在全局工作单元中，以及更新 IBM WebSphere MQ 资源时，应用程序可以更新 Enscribe 文件， SQL/MP 数据库或 SQL/MX 数据库。

使用全局工作单元

全局工作单元作为 TMF 事务实现。 应用程序通过调用 BEGINTRANSACTION 来启动全局工作单元，并通过调用 ENDTRANSACTION 来落实工作单元，或者通过调用 ABORTTRANSACTION 来回退工作单元。 应用程序还可以使用其他 TMF API 调用。

应用程序可以从另一个应用程序继承 TMF 事务。 例如，应用程序 (第一个应用程序) 可在事务中执行工作，然后回复事务并将其传递回第二个应用程序以进行进一步处理。 因此，第一个和第二个应用程序都可以参与同一全局工作单元，该工作单元涉及对 IBM WebSphere MQ 队列的更新以及对文件和数据库的更新。 在应

用程序之间传递 TMF 事务的能力意味着多个 IBM WebSphere MQ 应用程序可以在同一全局工作单元中执行消息传递操作。

一个应用程序可以同时管理和控制多个活动的 TMF 事务。 事务可以由应用程序本身启动，也可以从其他应用程序继承，也可以同时从这两个应用程序启动。 这意味着一个应用程序可以同时参与多个全局工作单元。

每个进程的最大并发活动 TMF 事务数为 1000，这是体系结构限制。 如果应用程序正在管理多个 TMF 事务，那么在任何时间点只能有一个事务是最新的。 或者，任何事务都不能是当前事务。 应用程序可以使用 TMF API 调用 (例如 RESUMETRANSACTION，ACTIVAT 勃起 EIVETRANSID 和 TMF_SET_TX_ID) 将当前状态从一个事务移动到另一个事务，或者指定没有任何事务是当前事务。 应用程序使用此级别的控制来确定是在本地工作单元，全局工作单元还是在同步点控制外部执行消息传递操作:

- 如果应用程序在当前没有 TMF 事务时在同步点控制中调用 MQPUT， MQPUT1 或 MQGET，那么 IBM WebSphere MQ 将在本地工作单元中处理该调用。
- 当应用程序具有当前 TMF 事务时，如果应用程序在同步点控制中调用 MQPUT， MQPUT1 或 MQGET，那么 IBM WebSphere MQ 将在当前 TMF 事务实现的全局工作单元内处理该调用。
- 如果应用程序在同步点控制外部调用 MQPUT， MQPUT1 或 MQGET，那么 IBM WebSphere MQ 会在同步点控制外部处理调用，而无论应用程序在调用时是否具有当前 TMF 事务。

IBM WebSphere MQ 从不在 MQI 调用期间更改应用程序的 TMF 事务的状态，除非在处理期间发生软件或硬件故障，并且 IBM WebSphere MQ 或操作系统确定必须回退该事务以保留数据完整性。 在将控制权返回给应用程序之前，每个 MQI 调用都会复原应用程序的事务状态。

避免长时间运行的事务

避免设计 TMF 事务在其中保持活动数十秒以上的应用程序。 长时间运行的事务可能会导致 TMF 的循环审计跟踪被填满。 由于 TMF 是关键的系统范围资源，因此 TMF 通过回退活动时间过长的应用程序事务来保护自己。

假设通过从队列中获取消息来驱动应用程序中的处理，并且应用程序从队列中获取消息并在工作单元中处理该消息。 通常，应用程序使用 wait 选项并在同步点控制内调用 MQGET 以从队列中获取消息。

如果应用程序正在使用全局工作单元，那么 MQGET 调用上指定的等待时间间隔必须短，以避免长时间运行的事务。 这意味着应用程序在检索消息之前可能需要多次发出 MQGET 调用。

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能。 有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。 任何对 IBM 产品、程序或服务的引用并非意在明示或默示只能使用 IBM 的产品、程序或服务。 只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务都可以代替 IBM 产品、程序或服务。 但是， 评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。 提供本文档并未授予用户使用这些专利的任何许可。 您可以以书面形式将许可查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

知识产权许可
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 063-8506 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状” 提供本出版物，不附有任何种类的 (无论是明示的还是暗含的) 保证，包括但不限于暗含的有关非侵权，适销和适用于某种特定用途的保证。 某些国家或地区在某些交易中不允许免除明示或暗含的保证。 因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。 此处的信息将定期更改；这些更改将编入本资料的新版本中。 IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式 充当对那些 Web 站点的保证。 那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Corporation
软件互操作性协调员，部门 49XA
北纬 3605 号公路
罗切斯特，明尼苏达州 55901
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。 因此，在其他操作环境中获得的数据可能会有明显的不同。 有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。 此外，有些测量是通过推算而估计的， 实际结果可能会有差异。 本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中 获取。 IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。 有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含日常商业运作所使用的数据和报表的示例。 为了尽可能全面地说明这些数据和报表，这些示例包括个人、公司、品牌和产品的名称。 所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有任何雷同，纯属巧合。

版权许可：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。 如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。 这些示例并未在所有条件下作全面测试。 因此，IBM 不能担保或默示这些程序的可靠性、可维护性或功能。

如果您正在查看本信息的软拷贝，图片和彩色图例可能无法显示。

编程接口信息

编程接口信息 (如果提供) 旨在帮助您创建用于此程序的应用软件。

本书包含有关允许客户编写程序以获取 IBM WebSphere MQ 服务的预期编程接口的信息。

但是，该信息还可能包含诊断、修改和调优信息。 提供诊断、修改和调优信息是为了帮助您调试您的应用程序软件。

要点: 请勿将此诊断，修改和调整信息用作编程接口，因为它可能会发生更改。

商标

IBM IBM 徽标 ibm.com 是 IBM Corporation 在全球许多管辖区域的商标。 当前的 IBM 商标列表可从 Web 上的 “Copyright and trademark information”www.ibm.com/legal/copytrade.shtml 获取。 其他产品和服务名称可能是 IBM 或其他公司的商标。

Microsoft 和 Windows 是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 Open Group 在美国和其他国家或地区的注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

此产品包含由 Eclipse 项目 (http://www.eclipse.org/) 开发的软件。

Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其附属公司的商标或注册商标。

部件号:

(1P) P/N: