9.3

IBM MQ 技术概述

注

在使用本资料及其支持的产品之前，请阅读第 251 页的『声明』中的信息。

内容

IBM MQ 技术概述

使用 IBM MQ 可连接应用程序并管理整个组织中的信息分发。

IBM MQ 使程序能够使用一致的应用程序编程接口在由不同组件 (处理器，操作系统，子系统和通信协议) 组成的网络中相互通信。 使用此接口设计和编写的应用程序称为消息排队应用程序。

使用以下子主题来了解消息排队以及 IBM MQ 提供的其他功能。

相关概念

IBM MQ 简介

在何处查找产品需求和支持信息

相关任务

规划 IBM MQ 体系结构

相关参考

第 6 页的『消息排队的主要功能和优点』

此信息突出显示了消息排队的一些功能和优点。 它描述了诸如消息排队的安全性和数据完整性之类的功能。

消息排队简介

IBM MQ 产品支持程序使用一致的应用程序编程接口在不同组件 (处理器，操作系统，子系统和通信协议) 的网络中相互通信。

使用此接口设计和编写的应用程序称为 消息排队 应用程序，因为它们使用 消息传递 和 排队 样式:

- 消息传递意味着程序通过在消息中相互发送数据而不是直接相互调用来进行通信。
- 排队意味着将消息放在存储器中的队列上，允许程序以不同的速度和时间，在不同的位置独立运行，并且没有它们之间的逻辑连接。

消息排队已在数据处理中使用多年。 现在最常用的是电子邮件。 无需排队，长途发送电子消息需要路由上的每个节点都可用于转发消息，并且收件人要登录并意识到您正在尝试向他们发送消息。 在排队系统中，消息存储在中间节点，直到系统准备好转发这些消息为止。 在他们的最终目的地，他们被存储在一个电子邮箱中，直到收件人准备好阅读它们。

即便如此，今天很多复杂的业务交易都是在不排队的情况下处理的。 在大型网络中，系统可能正在维护数千个处于即用即用状态的连接。 如果系统的一个部分迂到问题，那么系统的许多部分将变为不可用。

您可以将消息排队视为营销计划的电子邮件。 在消息排队环境中，构成应用程序套件的一部分的每个程序都会执行明确定义的自包含函数以响应特定请求。 要与另一个程序通信，程序必须将消息放在预定义的队列上。 另一个程序从队列中检索消息，并处理消息中包含的请求和信息。 所以消息排队是一种程序间通信的样式。

排队是在应用程序准备好处理消息之前保留消息的机制。 排队允许您:

- 在程序之间进行通信 (可能每个程序都在不同的环境中运行) ，而不必编写通信代码。
- 选择程序处理消息的顺序。
- 当消息数超过阈值时，通过安排多个程序为队列提供服务来平衡系统上的负载。
- 通过安排备用系统在主系统不可用时为队列提供服务，提高应用程序的可用性。

什么是消息队列?

消息队列 (简称为队列) 是可将消息发送至的指定目标。 消息会在队列上累积，直到服务这些队列的程序检索到这些消息为止。

队列驻留在队列管理器中并由其管理 (请参阅 第 8 页的『消息排队术语』)。 队列的物理性质取决于运行队列管理器的操作系统。 队列可以是计算机内存中的易失性缓冲区，也可以是永久存储设备 (例如磁盘) 上的数据集。 队列的物理管理由队列管理器负责，不会对参与的应用程序显示。

程序仅通过队列管理器的外部服务访问队列。 他们可以打开队列，将消息放在队列上，从中获取消息，然后关闭队列。 它们还可以设置和查询队列的属性。

消息排队的不同样式

点到点 (point-to-point)

一条消息放置在队列上，一个应用程序接收该消息。

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。

发布/预订

由发布应用程序发布的每条消息的副本将传递到每个感兴趣的应用程序。 可能有许多，一个或没有感兴趣的应用程序。 在发布/预订中，感兴趣的应用程序称为订户，并且消息在由预订标识的队列上排队。

发布/预订消息传递允许您将信息提供者与该信息的使用者分离。 对于要发送和接收的信息，发送申请和接收申请不需要相互了解那么多。 有关更多信息，请参阅 第 51 页的『发布/预订消息传递』。

消息排队对应用程序设计者和开发者的好处

IBM MQ 允许应用程序使用 消息排队 来参与消息驱动的处理。 应用程序可以使用相应的消息排队软件产品在不同平台之间进行通信。 例如， z/OS 应用程序可以通过 IBM MQ for z/OS 进行通信。 应用程序与底层通信的机制相屏蔽。 消息排队的其他优点包括:

- 您可以使用可在许多应用程序之间共享的小程序来设计应用程序。
- 您可以通过复用这些构建块来快速构建新应用程序。
- 为使用消息排队技术而编写的应用程序不受队列管理器工作方式更改的影响。
- 您不需要使用任何通信协议。 队列管理器为您处理通信的所有方面。
- 在向其发送消息时，接收消息的程序无需运行。 这些消息保留在队列上。

设计人员可以降低其应用程序的成本，因为开发速度更快，需要的开发者更少，对编程技能的要求也比不使用消息排队的应用程序低。

IBM MQ 在应用程序运行的任何位置实现称为 消息队列接口 (或 MQI) 的公共应用程序编程接口。 这使您可以更轻松地将应用程序从一个平台移植到另一个平台。

有关 MQI 的详细信息，请参阅 消息队列接口概述。

消息排队的主要功能和优点

此信息突出显示了消息排队的一些功能和优点。 它描述了诸如消息排队的安全性和数据完整性之类的功能。

使用消息排队技术的应用程序的主要功能包括:

- 第 7 页的『程序之间没有直接连接』
- 第 7 页的『独立于时间的通信』
- 第 7 页的『小程序』
- 第 7 页的『消息驱动的处理』
- 第 7 页的『事件驱动的处理』
- 第 8 页的『消息优先级』
- 第 8 页的『安全性』
- 第 8 页的『数据完整性』
- 第 8 页的『恢复支持』

注: 在考虑 IBM MQ 客户机和服务器时，您不必更改服务器应用程序以支持新平台上的其他 IBM MQ MQI clients 。 同样， IBM MQ MQI client 可以在不进行更改的情况下使用其他类型的服务器。

程序之间没有直接连接

消息排队是一种用于间接程序间通信的技术。 它可以在程序相互通信的任何应用程序中使用。 通信由一个将消息放入队列 (由队列管理器拥有) 的程序和另一个从队列中获取消息的程序进行。

程序可以获取由其他程序放入队列中的消息。 其他程序可以连接到与接收程序相同的队列管理器，也可以连接到其他队列管理器。 此另一个队列管理器可能位于另一个系统上，另一个计算机系统上，甚至位于另一个业务或企业内。

使用消息队列进行通信的程序之间没有物理连接。 程序将消息发送到队列管理器拥有的队列，而另一个程序从队列中检索消息 (请参阅 第 7 页的图 1)。

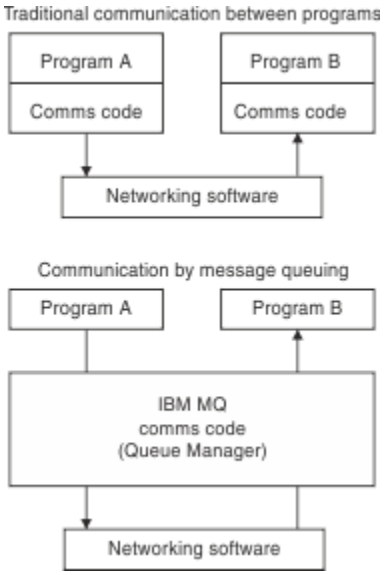


图 *1:* 与传统通信相比的消息排队

与电子邮件一样，作为事务一部分的个别消息通过商店上的网络传输。基础。 如果节点之间的链接失败，那么将保留该消息，直到恢复该链接，或者操作员或程序重定向该消息。

在程序中隐藏消息从队列移动到队列的机制。 因此，程序更简单。

独立于时间的通信

请求他人执行工作的程序不必等待对请求的应答。 他们可以执行其他工作，并在应答到达时或稍后处理该应答。 在编写消息传递应用程序时，当程序发送消息时，或者当目标能够接收消息时，您无需知道 (或关心)。 此消息不会丢失; 队列管理器将保留此消息，直到目标准备好对其进行处理为止。 消息将保留在队列中，直到程序将其除去。 这意味着发送和接收应用程序已解耦; 发送方可以继续处理，而无需等待接收方确认接收消息。 发送消息时，目标应用程序甚至不必正在运行。 它可以在消息启动后检索该消息。

小程序

消息排队允许您使用小型自包含程序的优点。 您可以将作业分布在几个较小的独立程序上，而不是单个大型程序按顺序执行作业的所有部分。 请求程序将消息发送到每个单独的程序，要求它们执行其功能; 当每个程序完成时，结果将作为一条或多条消息发送回。

消息驱动的处理

当消息到达队列时，它们可以使用 触发自动启动应用程序。 如果需要，可以在处理消息 (或消息) 时停止应用程序。

事件驱动的处理

可以根据队列的状态来控制程序。 例如，您可以安排程序在消息到达队列时立即启动，或者您可以指定程序在队列上具有高于特定优先级的 10 条消息或队列上具有任何优先级的 10 条消息之前不启动。

消息优先级

当程序将消息放入队列中时，它可以将优先级分配给消息。 这将确定添加新消息的队列中的位置。

程序可以按消息在队列中的顺序或通过获取特定消息从队列中获取消息。 (如果程序正在查找对其先前发送的请求的应答，那么该程序可能希望获取特定消息。)

安全性

提供了安全设施，包括当应用程序使用队列管理器时对其进行认证，当它们使用资源 (例如队列管理器上的队列) 时对其进行授权检查，以及当消息数据在网络上传输时对其进行加密，以及当消息数据驻留在队列上时对其进行加密。 有关安全性的更多信息，请参阅 安全性概述。

数据完整性

数据完整性由工作单元提供。 在每个 MQGET 或 MQPUT 上，完全支持将工作单元的开始和结束同步作为选项，从而允许落实或回滚工作单元的结果。 同步点支持在内部或外部针对 IBM MQ 运行，具体取决于为应用程序选择的同步点协调形式。

恢复支持

为了实现恢复，将记录所有持久 IBM MQ 更新。 如果需要恢复，那么将复原所有持久消息，回滚所有正在进行的事务，并以控制中的同步点管理器的正常方式处理任何同步点落实和回退。 有关持久消息的更多信息，请参阅 消息持久性。

消息排队术语

此信息提供了对消息排队中使用的一些术语的洞察。

它们包括:

- 通道
- 集群
- IBM MQ MQI client
- z/OS 组内排队
- 消息
- 消息通道代理程序
- 消息描述符
- 点到点
- 发布/预订
- 队列
- 队列管理器
- z/OS 队列共享组
- z/OS 共享队列
- 预订
- 主题

通道

通道用于将消息从一个队列管理器移动到另一个队列管理器，它们保护应用程序免受底层通信协议的保护。 队列管理器可能存在于同一系统上，也可能存在于同一平台上或不同平台上的不同系统上。 发送的消息可能源自许多位置:

- 用户编写的应用程序，用于将数据从一个节点传输到另一个节点。

- 使用 PCF 命令或 MQAI 的用户编写的管理应用程序。
- IBM MQ Explorer。
- 将检测事件消息发送到另一个队列管理器的队列管理器。
- 将远程管理命令发送到另一个队列管理器的队列管理器。 例如，使用 MQSC 命令或 administrative REST API。

有关通道的更多信息，请参阅 第 27 页的『通道定义』。

集群

集群 是以某种方式逻辑关联的队列管理器网络。

在不使用集群的分布式排队的 IBM MQ 网络中，每个队列管理器都是独立的。 如果一个队列管理器需要将消息发送到另一个队列管理器，那么它必须已定义传输队列和到远程队列管理器的通道。

使用集群有两种不同的原因: 减少系统管理和提高可用性和工作负载均衡。

一旦建立了即使是最小的集群，您也会从简化的系统管理中获益。 属于集群的队列管理器需要较少的定义，因此减少了在定义中发生错误的风险。

有关集群的更多信息，请参阅 集群。

IBM MQ MQI client

IBM MQ MQI 客户机 是 IBM MQ 的独立可安装组件。 MQI 客户机允许您使用通信协议运行 IBM MQ 应用程序，与其他平台上的一个或多个消息队列接口 (MQI) 服务器进行交互并连接到其队列管理器。

有关如何安装和使用 IBM MQ MQI client 组件的完整详细信息，请参阅以下主题:

- AIX 在 AIX 上安装 IBM MQ 客户机
- Linux 在 Linux® 上安装 IBM MQ 客户机
- Windows 在 Windows 上安装 IBM MQ 客户机
- IBM i 在 IBM i 上安装 IBM MQ 客户机

和 配置服务器与客户机之间的连接。

组内排队

z/OS

队列共享组中的队列管理器可以使用正常通道进行通信，也可以使用称为 组内排队 (IGQ) 的技术，这使您能够在不定义通道的情况下执行快速消息传输。 这仅适用于 IBM MQ for z/OS。

有关组内排队的更多信息，请参阅 第 173 页的『组内排队』。

消息

在消息排队中，消息是由一个程序发送并用于另一个程序的数据集合。 请参阅 IBM MQ 消息。

有关消息类型的信息，请参阅 消息类型。

消息通道代理程序

消息通道代理程序是通道的一端。 一对消息通道代理 (一个发送和一个接收) 组成一个通道并将消息从一个队列管理器移动到另一个队列管理器。

有关如何使用消息通道代理程序的信息，请参阅 分布式队列管理简介。

消息描述符

IBM MQ 消息由控制信息和应用程序数据组成。

控制信息在消息描述符结构 (MQMD) 中定义，并包含如下内容:

- 消息类型
- 消息的标识
- 消息传递的优先级

应用程序数据的结构和内容由参与程序确定，而不是由 IBM MQ 确定。

有关更多信息，请参阅 MQMD。

点到点消息传递

在点到点消息传递中，每条消息从一个生产应用程序传输到一个消费应用程序。 通过将消息放入队列中的生产应用程序来传输消息，并且使用应用程序从该队列中获取消息。

发布/预订消息传递

在发布/预订消息传递中，由发布应用程序发布的每条消息的副本将传递到每个感兴趣的应用程序。 可能有许多，一个或不感兴趣的应用程序。 在发布/预订中，感兴趣的应用程序称为订户，并且消息在由预订标识的队列上排队。

有关更多信息，请参阅第 51 页的『发布/预订消息传递』。

队列

可将消息发送到的指定目标。 消息会在队列上累积，直到服务这些队列的程序检索到这些消息为止。

有关更多信息，请参阅第 16 页的『队列』。

队列管理器

队列管理器 是向应用程序提供排队服务的系统程序。

它提供了一个应用程序编程接口，以便程序可以将消息放入队列并从队列中获取消息。 队列管理器提供了其他功能，以便管理员可以创建新队列，更改现有队列的属性以及控制队列管理器的操作。

要使 IBM MQ 消息排队服务在系统上可用，必须有一个正在运行的队列管理器。 您可以在单个系统上运行多个队列管理器 (例如，将测试系统与 实时 系统分开)。 对于应用程序，每个队列管理器由 连接句柄 (*Hconn*) 标识。

许多不同的应用程序可以同时使用队列管理器的服务，而这些应用程序可能完全不相关。 要使程序使用队列管理器的服务，它必须建立与该队列管理器的连接。

要使应用程序向连接到其他队列管理器的应用程序发送消息，队列管理器必须能够在它们之间进行通信。 IBM MQ 实现 *store-and-forward* 协议，以确保在此类应用程序之间安全地传递消息。

有关更多信息，请参阅第 23 页的『队列管理器』。

队列共享组

z/OS

可以访问同一组共享队列的队列管理器构成一个称为 队列共享组 (QSG) 的组。 它们通过用于存储共享队列的耦合设施 (CF) 相互通信。 这仅适用于 IBM MQ for z/OS。

有关更多信息，请参阅第 137 页的『共享队列和队列共享组』。

共享队列

z/OS

共享队列 是一种本地队列，其消息可由综合系统中的一个或多个队列管理器访问。 这与多个应用程序使用同一队列管理器共享的队列不同。 这仅适用于 IBM MQ for z/OS。

预订

发布/预订应用程序可以注册对有关特定主题的消息的兴趣。 当应用程序执行此操作时，它称为订户，术语预订定义匹配消息排队等待处理的方式。

预订包含有关订户的身份以及要将发布放置到的目标队列的身份的信息。 它还包含有关如何将发布放在目标队列上的信息。

有关更多信息，请参阅第 53 页的『订户和预订』。

Topic

主题是描述在发布/预订消息中所发布信息的主题的字符串。

主题是发布/预订系统中成功发送消息的关键。 发布者为每条消息分配一个主题，而不是在每条消息中包含一个特定目标地址。 队列管理器使该主题与一组已预订该主题的订户匹配，并将消息传递至其中每个订户。

有关更多信息，请参阅第 56 页的『主题』。

消息和队列

消息和队列是消息排队系统的基本组件。

什么是消息?

消息 是对使用它的应用程序有意义的字节字符串。 消息用于将信息从一个应用程序传输到另一个应用程序(或者在同一应用程序的不同部分之间)。 应用程序可以在同一平台上运行，也可以在不同平台上运行。

IBM MQ 消息由以下内容组成:

- 应用程序数据。应用程序数据的内容和结构由使用它的应用程序定义。
- 消息描述符。消息描述符标识消息并包含其他控制信息，例如消息类型以及发送应用程序分配给消息的优先级。

  消息描述符的格式由 IBM MQ 定义。 有关消息描述符的完整描述，请参阅 MQMD-消息描述符。
- 消息属性。 有关消息的元数据。 消息属性的内容由使用它们的应用程序定义。 有关更多信息，请参阅 消息属性。

消息长度

缺省最大消息长度为 4 MB ，尽管您可以将其增大到最大长度 100 MB (其中 1 MB 等于 1 048 576 字节)。 在实践中，消息长度可能受以下限制:

- 为接收队列定义的最大消息长度
- 为队列管理器定义的最大消息长度
- 队列定义的最大消息长度
- 发送或接收应用程序定义的最大消息长度
- 可用于消息的存储量

可能需要多条消息才能发送应用程序所需的所有信息。

应用程序如何发送和接收消息?

应用程序使用 **MQI 调用**发送和接收消息。

例如，要将消息放入队列，应用程序:

1. 通过发出 MQI MQOPEN 调用打开所需队列

2. 发出 MQI MQPUT 调用以将消息放入队列

另一个应用程序可以通过发出 MQI MQGET 调用从同一队列检索消息

有关 MQI 调用的更多信息，请参阅 MQI 调用。

什么是队列?

队列 是用于存储消息的数据结构。

每个队列都由 队列管理器拥有。 队列管理器负责维护它所拥有的队列，并负责将它接收到的所有消息存储到适当的队列中。 这些消息可由应用程序或队列管理器作为其正常操作的一部分放在队列上。

预定义队列和动态队列

可以通过创建队列的方式来描述这些队列:

- **预定义队列** 由管理员使用相应的 MQSC 或 PCF 命令创建。预定义队列是永久队列; 它们独立于使用它们的应用程序而存在，并且在 IBM MQ 重新启动后仍存在。
- **动态队列** 是在应用程序创建时创建的 发出 MQOPEN 请求，指定 模型队列的名称。 创建的队列基于称为模型队列的 模板队列定义。您可以使用 MQSC 命令 DEFINE QMODEL 来创建模型队列。 模型队列的属性 (例如，可存储在其上的最大消息数) 由从中创建的任何动态队列继承。

  模型队列具有一个属性，该属性指定动态队列是永久队列还是临时队列。 永久队列在应用程序和队列管理器重新启动后仍然存在; 临时队列在重新启动时丢失。

从队列中检索消息

适当授权的应用程序可以根据以下检索算法从队列中检索消息:

- 先进先出 (FIFO)。
- 消息优先级，如消息描述符中所定义。 将根据 FIFO 检索具有相同优先级的消息。
- 针对特定消息的程序请求。

来自应用程序的 MQGET 请求确定所使用的方法。

IBM MQ 对象

队列管理器定义 IBM MQ 对象的属性。 这些属性的值会影响 IBM MQ 处理这些对象的方式。 您可以使用 IBM MQ 命令和界面来创建和管理对象。 在应用程序中，使用消息队列接口 (MQI) 来控制对象。 从程序寻址时，对象由 IBM MQ 对象描述符 (MQOD) 标识。

对象管理

对象管理包括以下任务:

- 启动和停止队列管理器。
- 为应用程序创建对象 (尤其是队列)。
- 显示或改变对象的属性。
- 正在删除对象。
- 使用通道来创建到其他 (远程) 系统上的队列管理器的通信路径。
- 创建队列管理器的 集群 以简化整体管理过程并平衡工作负载。

除了动态队列之外，必须先向队列管理器定义对象，然后才能使用这些对象。

使用 IBM MQ 命令执行对象管理操作时，队列管理器将检查您是否具有执行该操作所需的权限级别。 同样，当应用程序使用 MQOPEN 调用打开对象时，队列管理器将检查应用程序是否具有所需的权限级别，然后才能访问此对象。 针对要打开的对象的名称执行检查。

您可以使用以下方法来定义和管理对象:

- 可编程命令格式参考 和 自动化管理任务 中描述的 PCF 命令
- MQSC 命令 中描述的 MQSC 命令
- z/OS IBM MQ for z/OS 操作和控制面板，如 管理 IBM MQ for z/OS 中所述
- Windows Linux IBM MQ Explorer (仅适用于 Intel 系统的 Windows 和 Linux)。 有关更多信息，请参阅 MQ Explorer 简介。

您还可以使用以下方法来管理对象:

- 从键盘输入的控制命令。 请参阅 使用控制命令管理 IBM MQ for Multiplatforms。
- IBM MQ 程序中的管理接口 (MQAI) 调用。 请参阅 IBM MQ 管理接口 (MQAI)。

ALW 对于 AIX, Linux, and Windows 上的 IBM MQ 命令序列，可以使用 MQSC 工具来运行文件中保存的一系列命令。 有关更多信息，请参阅 使用 MQSC 命令管理 IBM MQ。

IBM i 对于定期使用的 IBM MQ for IBM i 命令序列，可以编写 CL 程序。 有关更多信息，请参阅 使用 CL 命令管理 IBM MQ for IBM i。

z/OS 对于您定期使用的 IBM MQ for z/OS 命令序列，您可以编写用于创建包含命令的消息并将这些消息放入系统命令输入队列的管理程序。 队列管理器处理此队列上的消息的方式与处理从命令行或操作和控制面板中输入的命令的方式相同。 此技术在 编写程序以管理 IBM MQ 中进行了描述，并在随 IBM MQ for z/OS 提供的 Mail Manager 样本应用程序中进行了演示。 有关此样本的描述，请参阅 IBM MQ for z/OS 的样本程序 。

对象属性

对象的属性由其属性定义。 某些可以指定，其他只能查看。

例如，队列可容纳的最大消息长度由其 **MaxMsgLength** 属性定义; 您可以在创建队列时指定此属性。 **DefinitionType** 属性指定如何创建队列; 您只能显示此属性。

在 IBM MQ 中，有两种引用属性的方法:

- 使用其 PCF 名称，例如 **MaxMsgLength**。
- 使用其 MQSC 命令名，例如 MAXMSGL。

队列共享组

z/OS

可以访问同一组共享队列的队列管理器构成一个称为 队列共享组 (QSG) 的组，它们使用存储共享队列的耦合设施 (CF) 相互通信。 请注意， QSG 并非严格意义上的对象。

共享队列是具有可由队列共享组中的一个或多个队列管理器访问的消息的本地队列类型。 这与多个应用程序使用同一队列管理器共享的队列不同。

队列共享组具有最多为四个字符的名称。 该名称在网络中必须是唯一的，并且必须与所有队列管理器名称不同。

要点: 共享队列和队列共享组仅在 IBM MQ for z/OS 上受支持。

请参阅 第 137 页的『共享队列和队列共享组』 以获取更多信息。

系统缺省对象

系统缺省对象 是一组对象定义，每当创建队列管理器时都会自动创建这些对象定义。 您可以复制和修改这些对象定义中的任何对象定义，以便在安装时在应用程序中使用。

缺省对象名具有主干 SYSTEM; 例如，缺省本地队列为 SYSTEM.DEFAULT.LOCAL.QUEUE，缺省接收方通道为 SYSTEM.DEF.RECEIVER。 无法重命名这些对象; 需要这些名称的缺省对象。

定义对象时，将从相应的缺省对象复制未显式指定的任何属性。 例如，如果定义本地队列，那么将从缺省队列 SYSTEM.DEFAULT.LOCAL.QUEUE。

请参阅 系统和缺省对象 以获取更多信息。

对象类型

许多管理任务涉及处理各种类型的 IBM MQ 对象。

有关命名 IBM MQ 对象的信息，请参阅 第 30 页的『对 IBM MQ 对象进行命名』。

有关在队列管理器上创建的缺省对象的信息，请参阅 第 13 页的『系统缺省对象』。

有关不同类型的 IBM MQ 对象的信息，请参阅以下内容:

认证信息对象

认证信息对象提供执行证书撤销检查所需的定义。

队列管理器认证信息对象构成对传输层安全性 (TLS) 的 IBM MQ 支持的一部分。 它提供了检查已撤销证书所需的定义。 认证中心撤销不再可信的证书。

您可以使用 MQSC 命令 **DEFINE AUTHINFO** 来定义认证信息对象。 有关认证信息对象的属性的更多信息，请参阅 **DEFINE AUTHINFO**。

您可以将以下 IBM MQ 控制命令与认证信息对象配合使用:

- **setmqaut** (授予或撤销权限)
- **dspmqaut** (显示对象授权)
- **dmpmqaut** (转储授权)
- **rcrmqobj** (重新创建对象)
- **rcdmqimg** (记录介质图像)
- **dspmqfls** (显示文件名)

有关 TLS 以及使用认证信息对象的概述，请参阅 传输层安全性 (TLS) 概念 和 IBM MQ 中的 TLS 安全协议。

通道

通道是提供从一个队列管理器到另一个队列管理器的通信路径的对象。

请参阅 第 24 页的『通道』 以获取更多信息。

通信信息对象

IBM MQ 多点广播提供低等待时间、高扇出和可靠的多点广播消息传递。 需要通信信息 (COMMINFO) 对象才能使用多点广播传输。

请参阅 第 87 页的『IBM MQ 多点广播』 以获取更多信息。

COMMINFO 对象是包含与多点广播传输关联的属性的 IBM MQ 对象。 有关这些属性的更多信息，请参阅 DEFINE COMMINFO。 有关创建 COMMINFO 对象的更多信息，请参阅 多点广播入门。

侦听器

侦听器 是接受来自其他队列管理器或客户机应用程序的网络请求并启动关联通道的进程。

可以使用 **runmqlsr** 控制命令来启动 侦听器进程 。

侦听器对象 是 IBM MQ 对象，允许您在队列管理器的作用域内管理侦听器进程的启动和停止。 通过定义侦听器对象的属性，可以执行以下操作:

- 配置侦听器进程。
- 指定当队列管理器启动和停止时，侦听器进程是否自动启动和停止。

要点: z/OS IBM MQ for z/OS 上不支持侦听器对象。 有关 IBM MQ for z/OS 如何通过使用通道启动程序实现侦听的更多信息，请参阅 第 134 页的『z/OS 上的通道启动程序』。

名称列表

名称列表 是一个 IBM MQ 对象，其中包含集群名称，队列名称或认证信息对象名称的列表。 在集群中，它可用于标识队列管理器为其保存存储库的集群的列表。

名称列表是包含其他 IBM MQ 对象列表的 IBM MQ 对象。 通常，名称列表由诸如触发器监视器之类的应用程序使用，在那些应用程序中，名称列表用来标识一组队列。 使用名称列表的优点是它独立于应用程序进行维护; 可以在不停止使用它的任何应用程序的情况下进行更新。 另外，如果一个应用程序失败，那么名称列表不受影响，其他应用程序可以继续使用该名称列表。

名称列表还与队列管理器集群配合使用，以维护由多个 IBM MQ 对象引用的集群列表。

您可以使用 MQSC 命令 DEFINE NAMELIST 和 ALTER NAMELIST 来定义和修改名称列表。

注: z/OS 或者，在 z/OS 上，可以使用 IBM MQ for z/OS 操作和控制面板

程序可以使用 MQI 来查找这些名称列表中包含的队列。 名称列表的组织由应用程序设计者和系统管理员负责。

要获取可供使用的名称列表属性的列表，请参阅 名称列表的属性。

进程定义

进程定义对象允许通过定义应用程序的属性以供队列管理器使用来启动应用程序，而无需操作员干预。

进程定义对象定义用于响应 IBM MQ 队列管理器上的触发器事件而启动的应用程序。 进程定义属性包括应用程序标识，应用程序类型以及特定于应用程序的数据。 有关更多信息，请参阅 第 22 页的『IBM MQ 用于特定用途的队列』中的 启动队列 。

要允许在不需要操作员干预的情况下启动应用程序，如 使用触发器启动 IBM MQ 应用程序中所述，队列管理器必须知道应用程序的属性。 这些属性是在 进程定义对象中定义的。

创建对象时， **ProcessName** 属性是固定的。 但是，您可以使用 IBM MQ 命令来更改其他属性。

注: z/OS 或者，在 z/OS 上，可以使用 IBM MQ for z/OS 操作和控制面板。

您可以使用 MQINQ-Inquire object attributes 来查询所有属性的值。

有关可供使用的进程定义属性的列表，请参阅 进程定义的属性。

队列

IBM MQ 队列 是一个指定对象，应用程序可以在该对象上放置消息，并且应用程序可以从中获取消息。

请参阅 第 16 页的『队列』 以获取更多信息。

队列管理器

IBM MQ 队列管理器向应用程序提供排队服务，并管理属于它们的队列。

请参阅 第 23 页的『队列管理器』 以获取更多信息。

服务

服务 对象是定义在队列管理器启动或停止时要运行的程序的一种方法。

程序可以是下列其中一种类型:

服务器
服务器 是将参数 SERVTYPE 指定为 SERVER 的服务对象。 服务器服务对象是将在启动指定队列管理器时执行的程序的定义。 只能同时执行服务器进程的一个实例。 运行时，可以使用 MQSC 命令 DISPLAY SVSTATUS 来监视服务器进程的状态。 通常，服务器服务对象是程序 (例如，死信处理程序或触发器监

视器) 的定义，但是可以运行的程序不限于 IBM MQ 随附的程序。 此外，可以定义服务器服务对象以包含将在指定队列管理器关闭以结束程序时运行的命令。

命令

命令 是将参数 SERVTYPE 指定为 COMMAND 的服务对象。 命令服务对象是将在启动或停止指定队列管理器时执行的程序的定义。 可以同时执行命令进程的多个实例。 命令服务对象与服务器服务对象不同，因为一旦执行了程序，队列管理器就不会监视该程序。 通常，命令服务对象是生命周期较短的程序的定义，并且将执行特定任务 (例如，启动一个或多个其他任务)。

要点: z/OS IBM MQ for z/OS 上不支持服务对象。

有关更多信息，请参阅 使用服务。

存储类

z/OS

存储类将一个或多个队列映射至页集。

这意味着该队列的消息将存储在该页集上 (可进行缓冲)。

仅在 IBM MQ for z/OS 上支持存储类。

有关存储类的更多信息，请参阅 在 z/OS 上规划 IBM MQ 环境。

Topic 对象

主题对象 是一个 IBM MQ 对象，允许您将特定非缺省属性分配给主题。

主题 由发布或预订特定 主题字符串的应用程序定义。 主题字符串可以通过使用正斜杠字符 (/) 分隔主题来指定主题层次结构。 这可以通过 主题树进行可视化。 例如，如果应用程序发布到主题字符串 `/Sport/American Football` 和 `/Sport/Soccer`，那么将创建一个主题树，其中包含具有两个子代 `American Football` 和 `Soccer` 的父节点 `Sport` 。

主题从其主题树中找到的第一个父管理节点继承其属性。 如果特定主题树中没有管理主题节点，那么所有主题都将从基本主题对象 SYSTEM.BASE.TOPIC。

您可以在主题树中的任何节点上创建主题对象，方法是在主题对象的 TOPICSTR 属性中指定该节点的主题字符串。 您还可以为管理主题节点定义其他属性。 有关这些属性的更多信息，请参阅 MQSC 命令或 使用 PCF 命令自动管理。 缺省情况下，每个主题对象都从其最接近的父管理主题节点继承其属性。

主题对象还可用于向应用程序开发者隐藏完整主题树。 如果为主题 `/Sport/American Football` 创建了名为 `FOOTBALL.US` 的主题对象，那么应用程序可以发布或预订名为 `FOOTBALL.US` 的对象，而不是具有相同结果的字符串 `/Sport/American Football` 。

如果在主题对象的主题字符串中输入 `#` ， `+` ， `/` 或 `*` 字符，那么该字符将被视为字符串中的正常字符，并被视为与主题对象关联的主题字符串的一部分。

有关主题对象的更多信息，请参阅 第 51 页的『发布/预订消息传递』。

相关概念

第 5 页的『消息排队简介』
IBM MQ 产品支持程序使用一致的应用程序编程接口在不同组件 (处理器，操作系统，子系统和通信协议) 的网络中相互通信。

相关参考

MQSC 命令

队列

IBM MQ 队列和队列属性简介。

消息存储在队列上，因此，如果放置应用程序期望对其消息进行应答，那么在等待该应答时可以执行其他工作。 应用程序使用消息队列接口 (MQI) 访问队列，如 消息队列接口概述中所述。

必须先创建队列，然后才能将消息放入队列中。 队列由队列管理器拥有，并且该队列管理器可以拥有许多队列。 但是，每个队列必须具有在该队列管理器中唯一的名称。

通过队列管理器维护队列。 在大多数情况下，每个队列都由其队列管理器进行物理管理，但这对于应用程序并不明显。 IBM MQ for z/OS 共享队列可以由队列共享组中的任何队列管理器管理。

要创建队列，可以使用 IBM MQ 命令 (MQSC)， PCF 命令或特定于平台的接口。 例如， IBM MQ for z/OS 操作和控制面板特定于平台。

您可以从应用程序 动态 为临时作业创建本地队列。 例如，您可以创建 *reply-to* 队列 (在应用程序结束后不需要这些队列)。 有关更多信息，请参阅 第 21 页的『动态队列和模型队列』。

在使用队列之前，必须打开队列，并指定要对其执行的操作。 例如，您可以打开以下对象的队列:

- 仅浏览消息 (不检索消息)
- 检索消息 (并与其他程序共享访问权或独占访问权)
- 将消息放入队列
- 查询队列的属性
- 设置队列的属性

要获取打开队列时可以指定的选项的完整列表，请参阅 MQOPEN-Open object。

队列的属性

队列的某些属性是在定义队列时指定的，之后无法更改 (例如，队列的类型)。 可以将队列的其他属性分组到可以更改的属性中:

- 在队列处理期间由队列管理器执行 (例如，队列的当前深度)
- 仅通过命令 (例如，队列的文本描述)
- 由应用程序使用 MQSET 调用 (例如，是否允许在队列上执行 put 操作)

您可以使用 MQINQ 调用来查找所有属性的值。

对于多种类型的队列通用的属性包括:

QName
队列的名称。

QTYPE
队列的类型。

QDesc
队列的文本描述。

InhibitGet
是否允许程序从队列获取消息。 但是，您永远无法从远程队列获取消息。

InhibitPut
是否允许程序将消息放入队列。

DefPriority
放入队列的消息的缺省优先级。

DefPersistence
放入队列中的消息的缺省持久性

作用域
控制此队列的条目是否也存在于名称服务中。

z/OS **Scope** 属性在 z/OS 上不受支持

有关这些属性的完整描述，请参阅 队列的属性。

定义队列的方法

您可以使用 MQSC DEFINE 命令或 PCF 创建队列 命令将队列定义到 IBM MQ 。 这些命令指定队列的类型及其属性。 例如，本地队列对象具有一些属性，这些属性指定应用程序在 MQI 调用中引用该队列时发生的情况。 属性的示例包括:

- 应用程序是否可以从队列中检索消息 (已启用 GET)
- 应用程序是否可以将消息放入队列 (已启用 PUT)
- 对队列的访问是由一个应用程序独占还是在应用程序之间共享
- 可同时存储在队列上的最大消息数 (最大队列深度)
- 可以放到队列中的消息的最大长度

您还可以使用各种特定于平台的接口来定义队列。

相关概念

第 48 页的『集群队列』
集群队列是由集群队列管理器托管并可供集群中其他队列管理器使用的队列。

第 41 页的『死信队列』
死信队列 (或未传递的消息队列) 是无法将消息路由到其正确目标时将消息发送到的队列。 每个队列管理器通常都有一个死信队列。

使用 PCF 命令自动管理

在 IBM MQ Console 中使用队列

相关任务

使用 MQSC 命令管理 IBM MQ

使用 MQ Explorer 创建和配置队列管理器和对象

IBM i 使用 CL 命令管理 IBM MQ for IBM i

z/OS 可在 IBM MQ for z/OS 上发出 MQSC 和 PCF 命令的源

相关参考

第 48 页的『共享队列与集群队列之间的比较』
此信息旨在帮助您比较共享队列和集群队列，并确定哪些可能更适合您的系统。

相关信息

第 137 页的『什么是共享队列?』

本地队列

传输，启动，死信，命令，缺省，通道和事件队列是本地队列的类型。

如果队列归程序连接到的队列管理器所有，那么该队列对程序而言是本地队列。 您可以从本地队列中获取消息及将消息放在本地队列上。

队列定义对象保存队列的定义信息以及放在该队列上的物理消息。

每个队列管理器都可以有一些用于特殊目的的本地队列:

传输队列

当应用程序将消息发送到远程队列时，本地队列管理器将消息存储在称为 传输队列的特殊本地队列中。 应用程序可以将消息直接放置在传输队列上，也可以通过远程队列定义间接放置。

当队列管理器向远程队列管理器发送消息时，它将使用以下顺序来标识传输队列:

1. 在远程队列的本地定义的 XMITQ 属性上指定的传输队列。
2. 与远程队列管理器同名的传输队列。 此值是远程队列的本地定义的 XMITQ 上的缺省值。
3. 在本地队列管理器的 DEFXMITQ 属性上指定的传输队列。

消息通道代理程序 是与传输队列相关联的通道程序，它将消息传递到其下一个目标。 下一个目标是消息通道所连接到的队列管理器。 它不一定是与消息的最终目标相同的队列管理器。 将消息传递到其下一个目标时，将从传输队列中删除该消息。 该消息可能必须在到达其最终目标的过程中通过许多队列管理

器。 您必须在路由的每个队列管理器上定义一个传输队列，每个队列管理器都保存等待传输到下一个目标的消息。 正常传输队列保存下一个目标的消息，尽管这些消息可能具有不同的最终目标。 集群传输队列保存多个目标的消息。 每条消息的 `correlID` 标识放置该消息以将其传输到其下一个目标的通道。

您可以在队列管理器上定义多个传输队列。 您可以为同一目标定义多个传输队列，每个传输队列用于不同的服务类。 例如，您可能想要为发送到同一目标的小消息和大消息创建不同的传输队列。 然后，您可以使用不同的消息通道来传输消息，以便大型消息不会容纳较小的消息。 缺省情况下，集群队列或集群主题的所有消息都放置在单集群传输队列 `SYSTEM.CLUSTER.TRANSMIT.QUEUE` 上。 作为一个选项，您可以更改缺省值，并将流向不同集群队列管理器的消息流量分隔到不同的集群传输队列上。 如果将队列管理器属性 `DEFCLXQ` 设置为 `CHANNEL`，那么每个集群发送方通道都会创建一个单独的集群传输队列。 作为替代方法，您可以手动定义集群传输队列以供集群发送方通道使用。

传输队列可以触发消息通道代理程序以继续发送消息; 请参阅 使用触发器启动 IBM MQ 应用程序。

z/OS 在 IBM MQ for z/OS 上，如果使用组内排队，那么传输队列由 组内排队代理提供服务。 在 IBM MQ for z/OS 上使用组内排队时，将使用共享传输队列。

启动队列

启动队列 是一个本地队列，当应用程序队列上发生触发器事件时，队列管理器在该队列上放置触发器消息。

触发器事件是旨在使程序开始处理队列的事件。 例如，事件可能超过 10 条消息到达。 有关触发如何工作的更多信息，请参阅 使用触发器启动 IBM MQ 应用程序。

死信（未传递的消息）队列

死信 *(*未传递的消息*)* 队列 是队列管理器将无法传递的消息放入其中的本地队列。

当队列管理器将消息放入死信队列时，它会向消息添加头。 头信息包括队列管理器将消息放入死信队列的原因。 它还包含原始消息的目标，日期以及队列管理器将消息放入死信队列的时间。

应用程序还可以将队列用于它们无法传递的消息。 有关更多信息，请参阅 使用死信 (未传递的消息) 队列。

系统命令队列

系统命令队列 是适当授权的应用程序可以向其发送 IBM MQ 命令的队列。 这些队列接收平台上支持的 PCF， MQSC 和 CL 命令，以便队列管理器能够对它们进行操作。

z/OS 在 IBM MQ for z/OS 上，队列称为 `SYSTEM.COMMAND.INPUT` ; 在其他平台上称为 `SYSTEM.ADMIN.COMMAND.QUEUE`。 接受的命令因平台而异。 有关详细信息，请参阅可编程命令格式参考。

系统缺省队列

系统缺省队列 包含系统的队列的初始定义。 创建队列定义时，队列管理器将从相应的系统缺省队列复制该定义。 创建队列定义与创建动态队列不同。 动态队列的定义基于您选择作为动态队列模板的模型队列。

事件队列

事件队列 用于保存事件消息。 这些消息由队列管理器或通道报告。

远程队列

对于程序，如果队列由与程序所连接的队列管理器不同的队列管理器拥有，那么该队列是 远程 队列。

在已建立通信链路的情况下，程序可以向远程队列发送消息。 程序永远无法从远程队列获取消息。

定义远程队列时创建的队列定义对象仅保存本地队列管理器查找您希望消息进入的队列所需的信息。 此对象称为 远程队列的本地定义。 远程队列的所有属性都由拥有它的队列管理器挂起，因为它是该队列管理器的本地队列。

打开远程队列时，要标识该队列，必须指定以下任一项:

- 定义远程队列的本地定义的名称。 从应用程序的角度来看，这与打开本地队列相同。 应用程序不需要知道队列是本地队列还是远程队列。

  要在除 IBM i 以外的所有平台上创建远程队列的本地定义，请使用 DEFINE QREMOTE 命令。

  IBM i 在 IBM i 上，使用 CRTMQMQ 命令。
- 远程队列管理器的名称以及该远程队列管理器已知的队列的名称。

除了 第 17 页的『队列的属性』中描述的公共属性外，远程队列的本地定义还具有三个属性。 这三个属性是:

RemoteQName
队列拥有的队列管理器识别该队列的名称。

RemoteQmgrName
拥有队列管理器的名称。

XmitQName
将消息转发到其他队列管理器时使用的本地传输队列的名称。

有关这些属性的更多信息，请参阅 队列的属性。

如果对远程队列的本地定义使用 MQINQ 调用，那么队列管理器仅返回本地定义的属性，即远程队列名称，远程队列管理器名称和传输队列名称，而不是远程系统中匹配的本地队列的属性。

另请参阅 传输队列 (Transmission queue)。

别名队列

别名队列 是可用于访问另一个队列或主题的 IBM MQ 对象。 这意味着多个程序可以使用同一个队列，使用不同的名称访问该队列。

通过解析别名 (称为基本队列) 而产生的队列。 可以是平台支持的以下任何类型的队列:

- 本地队列
- 远程队列的本地定义。
- z/OS 共享队列，这是仅在 IBM MQ for z/OS 上可用的本地队列类型。
- 预定义队列
- 动态队列

别名也可以解析为主题。 如果应用程序当前将消息放入队列，那么可以通过使队列名称成为主题的别名来将其发布到主题。 无需更改应用程序代码。

注: 别名不能直接解析为同一队列管理器上的另一个别名。

使用别名队列的一个示例是系统管理员对基本队列名称 (即，别名解析到的队列) 和别名队列名称授予不同的访问权限。 这意味着可以授权程序或用户使用别名队列，但不能使用基本队列。

或者，可以设置授权以禁止对别名执行放置操作，但允许对基本队列执行这些操作。

在某些应用程序中，使用别名队列意味着系统管理员可以轻松更改别名队列对象的定义，而不必更改应用程序。

当程序尝试使用别名时， IBM MQ 会针对别名进行授权检查。 它不会检查程序是否有权访问别名解析为的名称。 因此，可以授权程序访问别名队列名称，但不能访问已解析的队列名称。

除了 第 16 页的『队列』中描述的常规队列属性外，别名队列还具有 **BaseQName** 属性。 这是别名解析到的基本队列的名称。 有关此属性的更完整描述，请参阅 BaseQName (MQCHAR48)。

InhibitGet 和 **InhibitPut** 属性 (请参阅 第 16 页的『队列』) 别名队列属于别名。 例如，如果别名队列名称 ALIAS1 解析为基本队列名称 BASE ，那么对 ALIAS1 的禁止仅影响 ALIAS1 ，并且不会禁止 BASE。 但是，禁止 BASE 也会影响 ALIAS1。

DefPriority 和 **DefPersistence** 属性也属于别名。 例如，您可以将不同的缺省优先级分配给同一基本队列的不同别名。 此外，您可以更改这些优先级，而不必更改使用别名的应用程序。

动态队列和模型队列

此信息可深入了解动态队列，临时和永久动态队列的属性，动态队列的使用，使用动态队列时的一些注意事项以及模型队列。

当应用程序发出 MQOPEN 调用以打开模型队列时，队列管理器会动态创建与模型队列具有相同属性的本地队列实例。 根据模型队列的 *DefinitionType* 字段的值，队列管理器将创建临时或永久动态队列 (请参阅创建动态队列)。

临时动态队列的属性

临时动态队列 具有以下属性:

- z/OS 它们不能是共享队列，可从队列共享组中的队列管理器访问。

  请注意，队列共享组仅在 IBM MQ for z/OS 上可用。
- 它们仅保存非持久消息。
- 它们是不可恢复的。
- 当队列管理器启动时，将删除它们。
- 当发出创建队列的 MQOPEN 调用的应用程序关闭队列或终止时，将删除这些消息。
  - 如果队列上存在任何已落实的消息，那么将删除这些消息。
  - 如果此时有任何未落实的 MQGET， MQPUT 或 MQPUT1 调用未针对队列执行，那么该队列将标记为正在逻辑上删除，并且仅在落实这些调用后作为关闭处理的一部分进行物理删除，或者在应用程序终止时进行物理删除。
  - 如果此时该队列正在使用 (由创建或另一个应用程序)，那么该队列将标记为正在逻辑上被删除，并且仅当最后一个使用该队列的应用程序关闭时才会物理上被删除。
  - 尝试访问逻辑删除的队列 (而不是关闭该队列) 失败，原因码为 MQRC_Q_DELETED。
  - 在针对创建队列的相应 MQOPEN 调用的 MQCLOSE 调用上指定 MQCO_NONE 时， MQCO_DELETE 和 MQCO_DELETE_PURGE 都将被视为 MQCO_NONE。

永久动态队列的属性

永久动态队列 具有以下属性:

- 它们保存持久或非持久消息。
- 在发生系统故障时，它们是可恢复的。
- 当应用程序 (不一定是发出了创建队列的 MQOPEN 调用的应用程序) 使用 MQCO_DELETE 或 MQCO_DELETE_PURGE 选项成功关闭队列时，将删除这些消息。
  - 如果队列上仍有任何消息 (已落实或未落实)，那么带有 MQCO_DELETE 选项的关闭请求将失败。 带有 MQCO_DELETE_PURGE 选项的关闭请求成功，即使队列上存在已落实的消息 (要在关闭过程中删除的消息)，但如果针对队列有任何未落实的 MQGET， MQPUT 或 MQPUT1 调用未完成，那么将失败。
  - 如果删除请求成功，但该队列恰好正在使用 (由创建应用程序或其他应用程序)，那么该队列将标记为逻辑上已删除，并且仅当最后一个使用该队列的应用程序关闭时才会实际删除该队列。
- 如果未被授权删除队列的应用程序关闭，那么不会将其删除，除非关闭应用程序发出了创建队列的 MQOPEN 调用。 将对用于验证相应 MQOPEN 调用的用户标识 (或备用用户标识 (如果指定了 MQOO_ALTERNATE_USER_AUTHORITY)) 执行授权检查。
- 可以使用与正常队列相同的方式删除这些队列。

动态队列的使用

您可以将动态队列用于:

- 在应用程序终止后不需要保留队列的应用程序。
- 需要由另一个应用程序处理对消息的回复的应用程序。 此类应用程序可以通过打开模型队列来动态创建应答队列。 例如，客户机应用程序可以:

  1. 创建动态队列。

2. 在请求消息的消息描述符结构的 **ReplyToQ** 字段中提供其名称。
3. 将请求放在服务器正在处理的队列上。

然后，服务器可以将应答消息放在应答队列上。 最后，客户机可以处理应答，并使用删除选项关闭应答队列。

使用动态队列时的注意事项

使用动态队列时，请考虑以下几点:

- 在客户机/服务器模型中，每个客户机都必须创建并使用其自己的动态应答队列。 如果在多个客户机之间共享动态应答队列，那么可能会延迟删除应答队列，因为针对该队列存在未落实的活动，或者该队列正由另一个客户机使用。 此外，该队列可能被标记为正在逻辑上被删除，并且对于后续 API 请求 (MQCLOSE 除外) 不可访问。
- 如果应用程序环境要求必须在应用程序之间共享动态队列，请确保仅当已落实针对队列的所有活动时才关闭该队列 (使用删除选项)。 这应该是最后一个用户。 这将确保不会延迟删除该队列，并将由于该队列已标记为已被逻辑删除而无法访问的时间段最小化。

模型队列

模型队列 是您在创建动态队列时使用的队列定义的模板。

您可以从 IBM MQ 程序动态创建本地队列，命名要用作队列属性模板的模型队列。 此时，您可以更改新队列的某些属性。 但是，您无法更改 **DefinitionType**。 例如，如果需要永久队列，请选择定义类型设置为永久的模型队列。 某些会话式应用程序可以使用动态队列来保存对其查询的应答，因为在处理这些应答之后，它们可能不需要维护这些队列。

在 MQOPEN 调用的 对象描述符 (MQOD) 中指定模型队列的名称。 通过使用模型队列的属性，队列管理器会动态地为您创建本地队列。

您可以指定动态队列的名称 (完整) 或名称的主干 (例如， ABC) ，并让队列管理器为此添加唯一部分，也可以让队列管理器为您分配完整的唯一名称。 如果队列管理器指定名称，那么它会将其放入 MQOD 结构中。

不能直接向模型队列发出 MQPUT1 调用，但可以向已通过打开模型队列创建的动态队列发出 MQPUT1 。

无法对模型队列发出 MQSET 和 MQINQ。 使用 MQOO_INQUIRE 或 MQOO_SET 打开模型队列会导致对动态创建的队列进行后续 MQINQ 和 MQSET 调用。

模型队列的属性是本地队列的属性的子集。 有关更完整的描述，请参阅 队列的属性。

IBM MQ 用于特定用途的队列

IBM MQ 将某些本地队列用于与其操作相关的特定用途。

您必须先定义这些队列，然后 IBM MQ 才能使用这些队列。

启动队列

启动队列是在触发中使用的队列。 当发生触发器事件时，队列管理器将触发器消息放在启动队列上。 触发器事件是队列管理器检测到的条件的逻辑组合。 例如，当队列上的消息数达到预定义的深度时，可能会生成触发器事件。 此事件会导致队列管理器将触发器消息放在指定的启动队列上。 此触发器消息由触发器监视器(用于监视启动队列的特殊应用程序) 检索。 然后，触发器监视器启动触发器消息中指定的应用程序。

如果队列管理器要使用触发，那么必须至少为该队列管理器定义一个启动队列。 请参阅 管理用于触发的对象， runmqtrm 和 使用触发器启动 IBM MQ 应用程序

传输队列

传输队列是临时存储发往远程队列管理器的消息的队列。 必须为本地队列管理器要直接向其发送消息的每个远程队列管理器定义至少一个传输队列。 这些队列也用于远程管理; 请参阅 来自本地队列管理器的远程管理。 有关在分布式排队中使用传输队列的信息，请参阅 IBM MQ 分布式排队技术。

每个队列管理器都可以具有缺省传输队列。

- 如果不属于集群的队列管理器将消息放入远程队列，那么缺省操作是使用缺省传输队列。
- 如果存在与目标队列管理器同名的传输队列，那么消息将放置在该传输队列上。

- 如果存在队列管理器别名定义，其中 **RQMNAME** 参数与目标队列管理器匹配，并且指定了 **XMITQ** 参数，那么会将消息放在由 **XMITQ** 指定的传输队列上。
- 如果没有 **XMITQ** 参数，则会将信息置于信息中指定的本地队列中。
- 如果没有队列，而报文的目的地是远程队列管理器，则 IBM MQ 会使用默认传输队列，即 **SYSTEM.DEFAULT.XMIT.QUEUE**。

集群传输队列

集群中的每个队列管理器都有一个名为 SYSTEM.CLUSTER.TRANSMIT.QUEUE 的集群传输队列和一个模型集群传输队列 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE。 缺省情况下，在定义队列管理器时将创建这些队列的定义。 如果队列管理器属性 **DEFCLXQ** 设置为 CHANNEL，那么将自动为创建的每个集群发送方通道创建永久动态集群传输队列。 这些队列称为 SYSTEM.CLUSTER.TRANSMIT.*ChannelName*。 您还可以手动定义集群传输队列。

属于集群的队列管理器将其中一个队列上的消息发送到同一集群中的其他队列管理器。

在名称解析期间，集群传输队列优先于缺省传输队列，而特定集群传输队列优先于 SYSTEM.CLUSTER.TRANSMIT.QUEUE。

死信队列

死信 (undelivered-message) 队列是存储无法路由到其正确目标的消息的队列。 例如，当目标队列已满时，无法路由消息。 提供的死信队列称为 SYSTEM.DEAD.LETTER.QUEUE。

对于分布式排队，请在涉及的每个队列管理器上定义死信队列。

命令队列

命令队列 SYSTEM.ADMIN.COMMAND.QUEUE 是一个本地队列，适当授权的应用程序可以将 MQSC 命令发送到该队列以进行处理。 然后，这些命令将由称为命令服务器的 IBM MQ 组件检索。 命令服务器验证命令，传递有效的命令以供队列管理器处理，并将任何响应返回到相应的应答队列。

创建队列管理器时，将自动为每个队列管理器创建命令队列。

应答队列

当应用程序发送请求消息时，接收消息的应用程序可以将应答消息发送回发送应用程序。 此消息放在队列上，称为应答队列，通常是发送应用程序的本地队列。 应答队列的名称由发送应用程序指定为消息描述符的一部分。

事件队列

检测事件可用于独立于 MQI 应用程序监视队列管理器。

发生检测事件时，队列管理器会将事件消息放入事件队列中。 然后，监视应用程序可以读取此消息，这可能会通知管理员或在事件指示问题时启动一些补救操作。

注: 触发器事件与检测事件不同。 触发器事件不是由相同的条件引起的，并且不会生成事件消息。

有关检测事件的更多信息，请参阅 检测事件。

队列管理器

队列管理器 及其提供给应用程序的排队服务简介。

程序必须具有与队列管理器的连接，然后才能使用该队列管理器的服务。 程序可以显式建立此连接 (使用 MQCONN 或 MQCONNX 调用) ，也可以隐式建立此连接 (这取决于运行程序的平台和环境)。

IBM MQ 队列管理器确保执行以下操作:

- 根据接收到的命令更改对象属性。
- 满足相应条件时，将生成特殊事件 (例如，触发器事件或检测事件)。
- 根据发出 MQPUT 调用的应用程序的请求，将消息放入正确的队列中。 如果无法执行此操作，那么将通知应用程序，并提供相应的原因码。

每个队列都属于单个队列管理器，并且被认为是该队列管理器的 本地队列 。 应用程序所连接的队列管理器被认为是该应用程序的 本地队列管理器 。 对于应用程序，属于其本地队列管理器的队列是本地队列。

远程队列 是属于另一个队列管理器的队列。 远程队列管理器 是除本地队列管理器以外的任何队列管理器。 远程队列管理器可以存在于网络中的远程机器上，也可以存在于本地队列管理器所在的机器上。 IBM MQ 支持同一机器上的多个队列管理器。

可以在某些 MQI 调用中使用队列管理器对象。 例如，您可以使用 MQI 调用 MQINQ 来查询队列管理器对象的属性。

队列管理器的属性

与每个队列管理器关联的是一组用于定义其特征的属性 (或属性)。队列管理器的某些属性在创建时是固定的; 您可以使用 IBM MQ 命令来更改其他属性。 您可以使用 MQINQ 调用来查询所有属性 (用于传输层安全性 (TLS) 加密的属性除外) 的值。

固定属性包括:

- 队列管理器的名称
- 运行队列管理器的平台 (例如， Windows)
- 队列管理器支持的系统控制命令的级别
- 可以分配给队列管理器处理的消息的最大优先级
- 程序可以向其发送 IBM MQ 命令的队列的名称
- 队列管理器可处理的最大消息长度 z/OS (仅在 IBM MQ for z/OS 中修正)
- 当程序放入和获取消息时，队列管理器是否支持同步

可更改 属性包括:

- 队列管理器的文本描述
- 队列管理器在处理 MQI 调用时用于字符串的字符集的标识
- 队列管理器用于限制触发器消息数的时间间隔
- z/OS 队列管理器用于确定队列扫描到期消息的频率的时间间隔 (仅限 IBM MQ for z/OS)
- 队列管理器的死信 (未传递的消息) 队列的名称
- 队列管理器的缺省传输队列的名称
- 任何一个连接的最大打开句柄数
- 启用和禁用各种类别的事件报告
- 工作单元中未落实的最大消息数

队列管理器和工作负载管理

您可以设置具有相同队列的多个定义的队列管理器集群 (例如，集群中的队列管理器可以相互克隆)。 特定队列的消息可由主管队列实例的任何队列管理器处理。 工作负载管理算法决定哪个队列管理器处理消息，从而在队列管理器之间传播工作负载; 请参阅 集群工作负载管理算法 以获取更多信息。

通道

通道 是分布式队列管理器在 IBM MQ MQI client 与 IBM MQ 服务器之间或两个 IBM MQ 服务器之间使用的逻辑通信链路。

通道用于将消息从一个队列管理器移动到另一个队列管理器，它们保护应用程序免受底层通信协议的保护。 队列管理器可能存在于同一系统上，也可能存在于同一平台上或不同平台上的不同系统上。 发送的消息可能源自许多位置:

- 用户编写的应用程序，用于将数据从一个节点传输到另一个节点。
- 使用 PCF 命令或 MQAI 的用户编写的管理应用程序。
- IBM MQ Explorer。
- 将检测事件消息发送到另一个队列管理器的队列管理器。

- 将远程管理命令发送到另一个队列管理器的队列管理器。 例如，使用 MQSC 命令或 administrative REST API。

通道有两个定义: 连接的每个端都有一个定义。 要使队列管理器相互通信，必须在要发送消息的队列管理器上定义一个通道对象，在要接收消息的队列管理器上定义另一个补充对象。 在连接的每一端都必须使用相同的 通道名称 ，并且所使用的 通道类型 必须兼容。

IBM MQ 中有三个类别的通道，这些类别中有不同的通道类型:

- 消息通道 (单向) ，将消息从一个队列管理器传输到另一个队列管理器。
- MQI 通道 (双向) ，用于将 MQI 调用从 IBM MQ MQI client 传输到队列管理器，并将响应从队列管理器传输到 IBM MQ 客户机。
- AMQP 通道，这是双向通道，用于将 AMQP 客户机连接到服务器上的队列管理器。 IBM MQ 使用 AMQP 通道在 AMQP 应用程序和队列管理器之间传输 AMQP 调用和响应

消息通道

消息通道的用途是将消息从一个队列管理器传输到另一个队列管理器。 客户机服务器环境不需要消息通道。

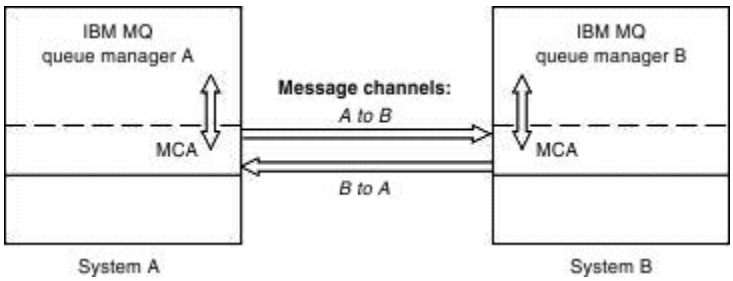


图 *2:* 两个队列管理器之间的消息通道

消息通道是单向链接。 如果您希望远程队列管理器响应本地队列管理器发送的消息，那么必须设置第二个通道以将响应发送回本地队列管理器。

消息通道使用 消息通道代理程序 (MCA) 连接两个队列管理器。 在通道的每一端都有一个消息通道代理程序。 您可以允许 MCA 使用多个线程来传输消息。 此过程称为 流水线。 Pipelining 使 MCA 能够更高效地传输消息，从而提高通道性能。 有关管道传输的更多信息，请参阅 通道属性。

有关通道的更多信息，请参阅 通道出口调用和数据结构和 第 38 页的『分布式排队组件』。

MQI 通道

消息队列接口 (MQI) 通道将 IBM MQ MQI client 连接到服务器上的队列管理器，并在您从 IBM MQ MQI client 应用程序发出 MQCONN 或 MQCONNX 调用时建立。

它是双向链接，仅用于传输 MQI 调用和响应，包括包含消息数据的 MQPUT 调用和导致返回消息数据的 MQGET 调用。 有不同的方法来创建和使用通道定义 (请参阅 定义 MQI 通道)。

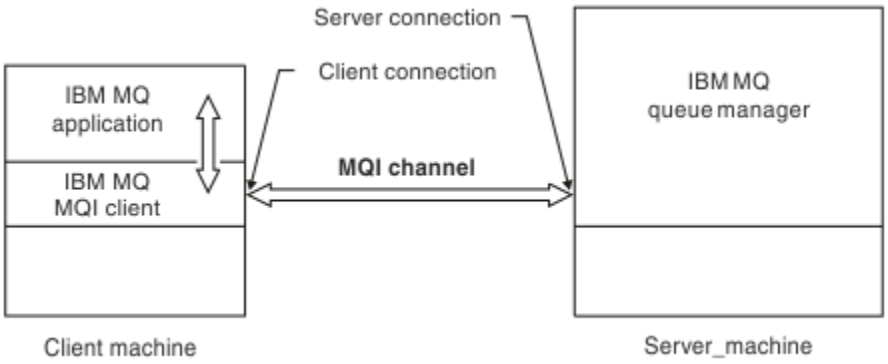


图 *3: MQI* 通道上的客户机连接和服务器连接

z/OS MQI 通道可用于将客户机连接到单个队列管理器，或连接到属于队列共享组的队列管理器 (请参阅 将客户机连接到队列共享组)。

MQI 通道定义有两种通道类型。 它们定义双向 MQI 通道。

客户机连接通道
此类型适用于 IBM MQ MQI client。

服务器连接通道
此类型适用于运行队列管理器的服务器，在 IBM MQ MQI client 环境中运行的 IBM MQ 应用程序将与该服务器进行通信。

AMQP 通道

Multi

只有一种类型的 AMQP 通道。

使用此通道连接 AMQP 消息传递应用程序和队列管理器，从而使该应用程序能够与 IBM MQ 应用程序交换消息。 AMQP 允许您使用 MQ Light 开发应用程序，然后将其部署为企业应用程序，从而利用 IBM MQ 提供的企业级功能。

客户机连接通道

客户机连接通道 是提供从 IBM MQ MQI client 到队列管理器的通信路径的对象。

客户机连接通道用于分布式排队，以在队列管理器和客户机之间移动消息。 它们保护应用程序免受底层通信协议的保护。 客户机可能存在于与队列管理器相同的或不同的平台上。

通道定义

请参阅 第 27 页的『通道定义』 以获取每种类型的通道的描述。

相关概念

第 35 页的『分布式排队和集群』
分布式排队意味着将消息从一个队列管理器发送到另一个队列管理器。 接收队列管理器可以在同一台机器上，也可以在另一台机器上; 可以在附近，也可以在世界的另一侧。 它可以在与本地队列管理器相同的平台上运行，也可以在 IBM MQ 支持的任何平台上运行。 您可以手动定义分布式排队环境中的所有连接，也可以创建集群并让 IBM MQ 为您定义许多连接详细信息。

消息队列接口概述

相关任务

管理远程 IBM MQ 对象
停止 MQI 通道
配置服务器与客户机之间的连接

相关参考

通道出口调用和数据结构

第 29 页的『通信』

IBM MQ MQI clients 使用 MQI 通道与服务器通信。

通道定义

描述 IBM MQ 使用的不同类型的消息通道和 MQI 通道的表。

当提到消息通道时，通道一词常用作通道定义的同义词。 通常从上下文中能清楚地分辨出我们正在谈论的是一个完整的通道（带有两端）还是一个通道定义（只有一端）。

消息通道

消息通道定义可以是下列其中一个类型：

消息通道定义类型	描述
发送方	发送方通道是队列管理器用来将消息发送至其他队列管理器的消息通道。 要使用发送方通道来发送消息，还必须在另一队列管理器上创建与此发送方通道同名的接收方通道。 如果要实现“回叫”机制，那么还可以将发送方通道与请求方通道配合使用。
服务器	服务器通道是队列管理器用来将消息发送至其他队列管理器的消息通道。 要使用服务器通道来发送消息，还必须在另一队列管理器上创建与此服务器通道同名的接收方通道。 您还可以将服务器通道与请求方通道配合使用。 在此情况下，在通道另一端的请求方通道定义将请求要启动的服务器通道定义。 服务器会发送消息给请求方。 只要服务器知道对方通道的连接名称，它还可启动通信。
接收方	接收方通道是队列管理器用来从其他队列管理器接收消息的消息通道。 要使用接收方通道来接收消息，还必须在另一队列管理器上创建与此接收方通道同名的发送方通道或服务器通道。
请求者	请求方通道是队列管理器用来从其他队列管理器接收消息的消息通道。 请求方通道可以请求启动在远端定义的伙伴通道。 如果伙伴通道是服务器通道，那么服务器通道会接受启动请求，并开始从服务器通道定义中标识的传输队列向请求方通道发送消息。 如果伙伴通道是发送方通道，那么发送方通道会接受启动请求，然后关闭与请求方的连接。 然后，发送方通道开始与伙伴请求方通道协商会话，并开始从发送方通道定义中标识的传输队列发送消息。 后一种情况实质上提供了一种回调机制，其中请求方通道可以请求发送方通道进行回调。
集群发送方	集群发送方 (CLUSSDR) 通道定义用来定义通道发送端，集群队列管理器可通过它将集群信息发送至其中一个完整存储库。 集群发送方通道用于告知存储库有关队列管理器状态的任何更改，例如，添加或除去队列。 它还用于传输消息。 完整存储库队列管理器自身拥有指向彼此的集群发送方通道。 它们使用这些通道以就集群状态的更改进行相互通信。 队列管理器的 CLUSSDR 通道定义指向哪个完整存储库不是很重要。 在进行了最初的接触之后，会根据需要自动定义更多的集群队列管理器对象，以便队列管理器可将集群信息发送至每个完整存储库并将消息发送至每个队列管理器。

消息通道定义类型	描述
集群接收方	集群接收方 (CLUSRCVR) 通道定义用来定义通道接收端，集群队列管理器可通过它从集群中的其他队列管理器接收消息。 集群接收方通道还可传送有关集群的信息（发往存储库）。 通过定义集群接收方通道，该队列管理器对其他集群队列管理器表示它可用于接收消息。 每个集群队列管理器至少需要有一个集群接收方通道。

对于每个通道，您必须定义两端以便获取通道每一端的通道定义。 通道的两端必须是兼容类型。

您可使用下列通道定义的组合：

- 发送方-接收方
- 服务器-接收方
- 请求方-服务器
- 请求方-发送方（回调）
- 集群发送方-集群接收方

消息通道代理程序

您创建的每个通道定义都属于特定队列管理器。 队列管理器可具有同一类型或不同类型的几个通道。 通道的每一端是一个程序，即消息通道代理程序 (MCA)。 在通道的一端，调用方 MCA 从传输队列获取消息并通过通道发送它们。 在通道的另一端，响应方 MCA 接收这些消息并将它们传递至远程队列管理器。

调用方 MCA 可与发送方通道、服务器通道或请求方通道关联。 响应方 MCA 可与任何类型的消息通道关联。

IBM MQ 在连接的两端支持通道类型的以下组合：

调用方		消息流方向	响应方	
通道类型	需要侦听器吗?		需要侦听器吗?	通道类型
发送方	否	从调用方至响应方	Yes	接收方
服务器	否	从调用方至响应方	Yes	接收方
服务器	否	从调用方至响应方	Yes	请求者
请求者	否	从响应方至调用方	Yes	服务器
请求者	Yes	从响应方至调用方	Yes	发送方

MQI 通道

MQI 通道可以是下列其中一个类型：

MQI 通道类型	描述
服务器连接	服务器连接通道是双向 MQI 通道，用于将 IBM MQ 客户机连接至 IBM MQ 服务器。 服务器连接通道是通道的服务器端。
客户机连接	客户机连接通道是双向 MQI 通道，用于将 IBM MQ 客户机连接至 IBM MQ 服务器。 IBM MQ Explorer 还使用客户机连接来连接至远程队列管理器。 客户机连接通道是通道的客户机端。 当您创建客户机连接通道时，在主管队列管理器的计算机上创建一个文件。 然后，您必须将该客户机连接文件复制到 IBM MQ 客户端计算机。

Multi 多线程支持-流水线

您可以选择允许消息通道代理 (MCA) 使用多个线程来传输消息。 此过程称为 流水线，使 MCA 能够更高效地传输消息，减少等待状态，从而提高通道性能。 每个 MCA 限制为最多两个线程。

您可以使用 qm.ini 文件中的 *PipeLineLength* 参数来控制管道。 此参数将添加到 Channels 节。

注: 流水线仅对 TCP/IP 通道有效。

使用管道时，必须将通道两端的队列管理器配置为具有大于 1 的 *PipeLineLength* 。

通道出口注意事项

管道可能导致某些出口程序失败，因为:

- 可能未按顺序调用出口。
- 可以从不同的线程交替调用出口。

在使用管道之前，请检查出口程序的设计:

- 出口必须在其执行的所有阶段都可重入。
- 使用 MQI 调用时，请记住，从不同线程调用出口时，不能使用相同的 MQI 句柄。

请考虑打开队列的消息出口，并将其句柄用于对该出口的所有后续调用的 MQPUT 调用。 这在管道方式下失败，因为从不同的线程调用了出口。 要避免此故障，请为每个线程保留一个队列句柄，并在每次调用出口时检查线程标识。

通信

IBM MQ MQI clients 使用 MQI 通道与服务器通信。

必须在连接的 IBM MQ MQI client 和服务器端创建通道定义。 定义 MQI 通道中说明了如何创建通道定义。

下表显示了可能的传输协议:

表 *1: MQI* 通道的传输协议

客户机平台	LU 6.2	TCP/IP	NetBIOS	SPX
IBM i IBM i		Yes		
Linux AIX AIX and Linux 系统	是 [1]	Yes		
Windows Windows	Yes	Yes	Yes	Yes

注:

1. Linux LU6.2 在以下平台上不受支持:
   - Linux (POWER 平台)
   - Linux (x86-64 平台)
   - Linux (zSeries s390x 平台)

传输协议- IBM MQ MQI client 和服务器平台的组合 显示了使用这些传输协议的 IBM MQ MQI client 和服务器平台的可能组合。

IBM MQ MQI client 上的 IBM MQ 应用程序可以使用与队列管理器为本地时相同的所有 MQI 调用。 **MQCONN** 或 **MQCONNX** 将 IBM MQ 应用程序与所选队列管理器相关联，从而创建 连接句柄。 然后，连接的队列管理器将处理使用该连接句柄的其他调用。 IBM MQ MQI client 通信需要客户机与服务器之间的活动连接，而队列管理器之间的通信是独立于连接和独立于时间的。

传输协议是使用通道定义指定的，不会影响应用程序。 例如， Windows 应用程序可以通过 TCP/IP 连接到一个队列管理器，并通过 NetBIOS 连接到另一个队列管理器。

性能注意事项

您使用的传输协议可能会影响 IBM MQ 客户机和服务器系统的性能。 在传输速度较慢的某些情况下，可以使用 IBM MQ 通道压缩。

对 IBM MQ 对象进行命名

针对 IBM MQ 对象采用的命名约定取决于对象。 与 IBM MQ 配合使用的机器的名称和用户标识也受到一些命名限制。

队列管理器的每个实例都由其名称识别。 此名称在相互连接的队列管理器的网络中必须唯一，以便一个队列管理器可以毫不含糊地标识将任何给定消息发送到的目标队列管理器。

对于其他类型的对象，每个对象都有一个与其关联的名称，可以通过该名称引用该名称。 这些名称在一个队列管理器和对象类型中必须唯一。 例如，可以有一个队列和一个同名的进程，但不能有两个同名的队列。

在 IBM MQ 中，名称最多可以包含 48 个字符，但最多包含 20 个字符的 通道 除外。 有关命名 IBM MQ 对象的更多信息，请参阅 第 30 页的『用于命名 IBM MQ 对象的规则』。

与 IBM MQ 配合使用的机器的名称和用户标识也受到一些命名限制:

- 确保机器名不包含任何空格。 IBM MQ 不支持包含空格的机器名。 如果在此类机器上安装 IBM MQ，那么无法创建任何队列管理器。
- 对于 IBM MQ 权限，用户标识和组的名称不得超过 20 个字符 (不允许使用空格)。
- Windows 如果客户机以包含 @ 字符的用户标识 (例如， abc@d.) 运行，那么 IBM MQ for Windows 服务器不支持连接 IBM MQ MQI client 。

相关概念

第 33 页的『IBM MQ 文件名』
每个 IBM MQ 队列管理器，队列，进程定义，名称列表，通道，客户机连接通道，侦听器，服务和认证信息对象都由一个文件表示。 由于对象名不一定是有效的文件名，因此队列管理器会在必要时将对象名转换为有效的文件名。

相关参考

第 30 页的『用于命名 IBM MQ 对象的规则』
IBM MQ 对象名具有最大长度并且区分大小写。 并非每个对象类型都支持所有字符，并且许多对象都有关于名称唯一性的规则。

用于命名 IBM MQ 对象的规则

IBM MQ 对象名具有最大长度并且区分大小写。 并非每个对象类型都支持所有字符，并且许多对象都有关于名称唯一性的规则。

有许多不同类型的 IBM MQ 对象，并且每种类型的对象都可以具有相同的名称，因为它们存在于不同的对象名称空间中: 例如，本地队列和发送方通道都可以具有相同的名称。 但是，一个对象不能与同一名称空间中的另一个对象具有相同的名称：例如，本地队列不能与模型队列具有相同的名称，发送方通道不能与接收方通道具有相同的名称。

以下 IBM MQ 对象存在于单独的对象名称空间中:

- 认证信息
- 通道
- 客户机通道
- 侦听器
- 名称列表
- 进程
- 队列
- 服务
- 存储类

- 预订
- Topic

对象名称的字符长度

通常， IBM MQ 对象名的长度最多可以为 48 个字符。 此规则适用于以下对象:

- 认证信息
- 集群
- 侦听器
- 名称列表
- 进程定义
- 队列
- 队列管理器
- 服务
- 预订
- Topic

存在以下限制:

1. z/OS 在 z/OS 系统上，队列管理器必须最多为 4 个字符，并且必须仅为大写字符和数字字符。
2. 通道对象名和客户机连接通道名的最大长度为 20 个字符。 请参阅 定义通道 以获取有关通道的更多信息。
3. 主题字符串最多可以为 10240 个字节。 所有 IBM MQ 对象名都区分大小写。
4. 预订名称最多可以为 10240 字节，并且可以包含空格。
5. 存储类名的最大长度为 8 个字符。
6. CF 结构名称的最大长度为 12 个字符。

对象名称中的字符

IBM MQ 对象名的有效字符为:

个字符之后	限制
大写 A-Z	• 无
小写 a-z	• 在 MQSC 脚本中，必须将带有小写字符的名称括在单引号中。 这将防止将小写字符转换为大写。 • 使用 EBCDIC 片假名的系统不能在对象名中使用小写 a-z 字符。 • z/OS 在 z/OS 系统上使用小写字符时可能存在限制，例如，队列管理器名称不能包含小写字符。 • IBM i 在 IBM i 系统上使用 CL 命令时，必须将带有小写字符的名称括在单引号中。 这将防止将小写字符转换为大写。
数字 0-9	• 无
句点 (.)	• 无

个字符之后	限制
下划线 (_)	• Multi 无 • z/OS 避免使用带有前导或尾部下划线的名称，因为 IBM MQ for z/OS 操作和控制面板无法处理这些名称。
正斜杠 (/)	• Windows 在 Windows 系统上，队列管理器名称的第一个字符不能是正斜杠。 • IBM i 在 IBM i 系统上使用 CL 命令时，必须将包含正斜杠的名称括在单引号中。 • z/OS 无
百分号 (%)	• ALW 无 • z/OS 如果要将 RACF 用作 IBM MQ for z/OS 的外部安全性管理器，请不要在对象名中使用%，因为在使用 RACF 通用概要文件时，这些名称不会包含在安全性检查中。 • IBM i 在 IBM i 系统上使用 CL 命令时，必须将包含百分号的名称括在单引号中。

还有一些关于对象名上的字符的一般规则:

1. 不允许前导空白或嵌入空白。
2. 不允许使用本地语言字符。
3. 任何小于完整字段长度的名称都可以用空格填充到右边。 队列管理器返回的所有短名称总是用空格填充到右边。

队列名称

队列的名称有两个部分:

- 队列管理器的名称
- 队列管理器已知的队列的局部名

队列名称的每个部分都有 48 个字符长。

要引用本地队列，可以省略队列管理器的名称 (通过将其替换为空白字符或使用前导空字符)。 但是，IBM MQ 返回到程序的所有队列名称都包含队列管理器的名称。

z/OS 共享队列 (可供其队列共享组中的任何队列管理器访问) 不能与同一队列共享组中的任何非共享本地队列同名。 此限制避免了应用程序在打算打开本地队列时误打开共享队列的可能性，反之亦然。 共享队列和队列共享组仅在 IBM MQ for z/OS 上可用。

要引用远程队列，程序必须在完整队列名称中包含队列管理器的名称，或者必须存在远程队列的本地定义。

当应用程序使用队列名称时，该名称可以是本地队列的名称 (或一个队列的别名)，也可以是远程队列的本地定义的名称，但应用程序不需要知道哪个，除非它需要从队列中获取消息 (当队列必须是本地队列时)。 当应用程序打开队列对象时， MQOPEN 调用将执行名称解析函数以确定要在哪个队列上执行后续操作。 这一点的重要意义在于，应用程序不依赖于在队列管理器网络中的特定位置定义的特定队列。 因此，如果系统管理员在网络中重新定位队列并更改其定义，那么不需要更改使用这些队列的应用程序。

保留对象名

将为队列管理器定义的对象保留以 SYSTEM. 开头的对象名。 您可以使用 **Alter**， **Define** 和 **Replace** 命令来更改这些对象定义以适合您的安装。 为 IBM MQ 定义的名称在 队列名称中完整列出。

z/OS 在 IBM MQ for z/OS 上，保留耦合设施应用程序结构名称 CSQSYSAPPL。

相关概念

AIX, Linux, and Windows 上的安装名称

IBM MQ 文件名

每个 IBM MQ 队列管理器，队列，进程定义，名称列表，通道，客户机连接通道，侦听器，服务和认证信息对象都由一个文件表示。 由于对象名不一定是有效的文件名，因此队列管理器会在必要时将对象名转换为有效的文件名。

队列管理器目录的缺省路径如下所示:

- 在 IBM MQ 配置信息中定义的前缀:
  - Linux AIX 在 AIX and Linux 上，缺省前缀为 `/var/mqm`。 这是在 mqs.ini 配置文件的 `DefaultPrefix` 节中配置的。
  - Windows 在 Windows 32 位系统上，缺省前缀为 `C:\Program Files (x86)\IBM\WebSphere MQ`。 在 Windows 64 位系统上，缺省前缀为 `C:\Program Files\IBM\MQ`。 对于 32 位和 64 位安装，数据目录将安装到 `C:\ProgramData\ IBM \ MQ` 中。 这是在 mqs.ini 配置文件的 `DefaultPrefix` 节中配置的。

  如果可用，可以使用 IBM MQ Explorer 中的 IBM MQ 属性页面来更改前缀，否则请手动编辑 `mqs.ini` 配置文件。
- 队列管理器名称将变换为有效的目录名称。 例如，队列管理器:

```
queue.manager
```

  将表示为:

```
queue!manager
```

  此过程称为 名称变换。

在 IBM MQ 中，可以为队列管理器提供最多包含 48 个字符的名称。

例如，可以命名队列管理器:

```
QUEUE.MANAGER.ACCOUNTING.SERVICES
```

但是，每个队列管理器都由一个文件表示，并且对文件名的最大长度以及名称中可以使用的字符都有限制。因此，将自动变换表示对象的文件的名称以满足文件系统的要求。

管理队列管理器名称变换的规则如下所示:

1. 变换单个字符:
   - 从。 到!
   - From / to &
2. 如果名称仍然无效:
   a. 将其截断为 8 个字符
   b. 附加三字符数字后缀

例如，假定缺省前缀和名为 `queue.manager` 的队列管理器:

- Windows 在具有 NTFS 或 FAT32 的 Windows 上，队列管理器名称变为:

```
C:\Program Files\IBM\MQ\qmgrs\queue!manager
```

- Windows 在具有 FAT 的 Windows 上，队列管理器名称变为:

```
C:\Program Files\IBM\MQ\qmgrs\queue!ma
```

- Linux AIX 在 AIX and Linux 上，队列管理器名称变为:

```
/var/mqm/qmgrs/queue!manager
```

变换算法还区分仅在不区分大小写的文件系统上不同的名称。

对象名变换

对象名不一定是有效的文件系统名。 您可能需要变换对象名。 所使用的方法与队列管理器名称的方法不同，因为虽然每台机器上只有几个队列管理器名称，但每个队列管理器可能有大量其他对象。 队列，进程定义，名称列表，通道，客户机连接通道，侦听器，服务和认证信息对象在文件系统中表示。

当变换过程生成新名称时，与原始对象名没有简单关系。 您可以使用 **dspmqfls** 命令在实对象名和已变换对象名之间进行转换。

相关参考
dspmqfls (显示文件名)
相关信息
mqs.ini 文件的 AllQueueManagers 节段

IBM i IBM i 上的对象名

队列管理器具有具有唯一名称的关联队列管理器库。 可能需要变换队列管理器名称和对象名以满足 IBM i Integrated File System (IFS) 的需求。

创建队列管理器时， IBM MQ 会将队列管理器库与其相关联。 此队列管理器库具有唯一名称，长度不超过 10 个字符，这在很大程度上取决于用户定义的队列管理器名称。 队列管理器和队列管理器库都放在一个目录中，该目录也基于具有前缀 /QIBM/UserData/mqm 的队列管理器名称。 以下是队列管理器，队列管理器库和目录的示例:

队列管理器名称	橙色
队列管理器库名	Qmorange
目录	/QIBM/UserData/mqm/ORANGE

所有队列管理器名称和队列管理器库名都将写入文件 /QIBM/UserData/mqm/mqs.ini 中的节。

IBM MQ IFS 目录和文件

IBM i Integrated File System (IFS) 由 IBM MQ 广泛用于存储数据。 有关 IFS 的更多信息，请参阅 *Integrated File System* 简介。

每个 IBM MQ 对象 (例如，通道或队列管理器) 都由一个文件表示。 由于对象名不一定是有效的文件名，因此队列管理器会在必要时将对象名转换为有效的文件名。

队列管理器目录的路径由以下内容构成:

- 在队列管理器配置文件 qm.ini 中定义的前缀。 缺省前缀为 /QIBM/UserData/mqm。
- 文字 qmgrs。
- 编码的队列管理器名称，即转换为有效目录名的队列管理器名称。 例如，队列管理器 queue/manager 由 queue&manager 表示。

此过程称为名称变换。

IFS 队列管理器名称变换

在 IBM MQ 中，可以为队列管理器提供最多包含 48 个字符的名称。

例如，可以将队列管理器命名为 QUEUE/MANAGER/ACCOUNTING/SERVICES。 与为每个队列管理器创建库的方式相同，每个队列管理器也由一个文件表示。 由于 EBCDIC 中的变体代码点，因此可以在名称中使用的字符存在限制。 因此，表示对象的 IFS 文件的名称会自动变换以满足文件系统的要求。

使用名称为 queue/manager 的队列管理器示例，将字符 / 转换为 &，并采用缺省前缀， IBM MQ for IBM i 中的队列管理器名称将变为 /QIBM/UserData/mqm/qmgrs/queue&manager。

对象名变换

对象名不一定是有效的文件系统名，因此可能需要变换对象名。 所使用的方法与队列管理器名称的方法不同，因为虽然每台机器只有几个队列管理器名称，但每个队列管理器可能有大量其他对象。 只有进程定义，队列和名称列表在文件系统中表示; 通道不受这些注意事项影响。

当变换过程生成新名称时，与原始对象名没有简单关系。 可以使用 DSPMQMOBJN 命令来查看 IBM MQ 对象的已变换名称。

分布式排队和集群

分布式排队意味着将消息从一个队列管理器发送到另一个队列管理器。 接收队列管理器可以在同一台机器上，也可以在另一台机器上; 可以在附近，也可以在世界的另一侧。 它可以在与本地队列管理器相同的平台上运行，也可以在 IBM MQ 支持的任何平台上运行。 您可以手动定义分布式排队环境中的所有连接，也可以创建集群并让 IBM MQ 为您定义许多连接详细信息。

分布式 排队

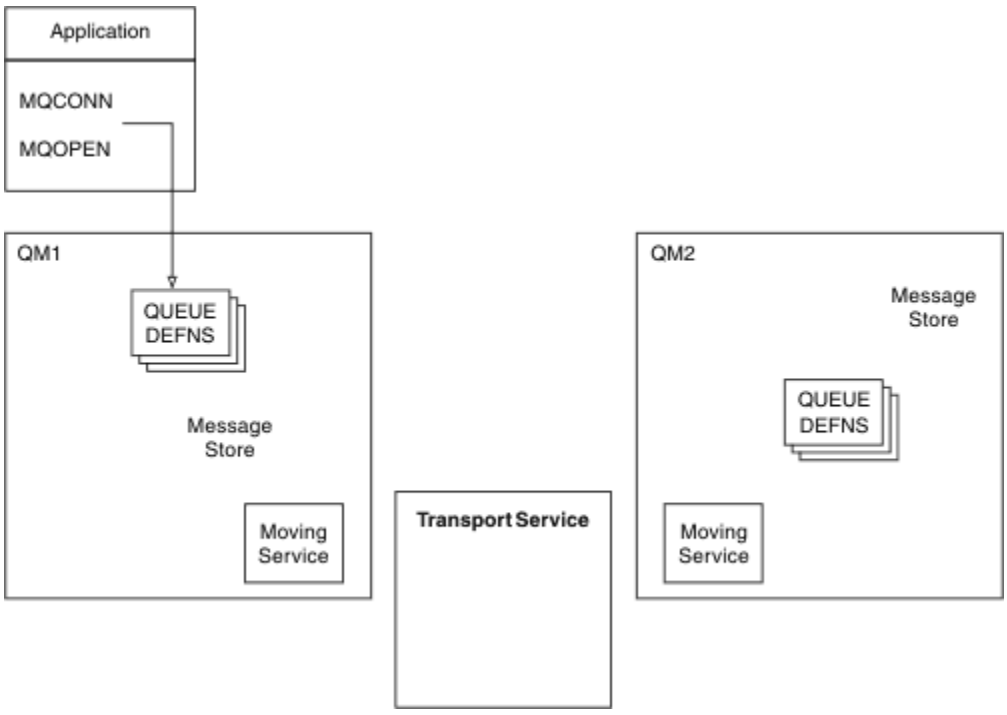


图 *4:* 分布式排队组件概述

在上图中:

- 应用程序使用 MQCONN 调用来连接到队列管理器。 然后，应用程序使用 MQOPEN 调用来打开队列，以便它可以将消息放入队列中。

- 每个队列管理器都有其每个队列的定义。 它可以具有 本地队列 (即，由此队列管理器托管) 的定义以及 远程队列 (即，由其他队列管理器托管) 的定义。
- 如果消息以远程队列为目标，那么本地队列管理器会将它们保存在 传输队列上，该队列会将它们持久存储在消息存储器中，直到可以将它们转发到远程队列管理器为止。
- 每个队列管理器都包含通信软件 (称为 移动服务) ，队列管理器使用这些软件与其他队列管理器进行通信。
- 传输服务 独立于队列管理器，可以是下列任何一项 (取决于平台):
  - 系统网络体系结构高级程序间通信 (SNA APPC)
  - 传输控制协议/因特网协议 (Transmission Control Protocol/Internet Protocol，TCP/IP)
  - 网络基本输入/输出系统 (NetBIOS)
  - 顺序包交换 (SPX)

发送消息所需的组件

如果要将消息发送到远程队列管理器，那么本地队列管理器需要 传输队列 和 通道的定义。 通道是两个队列管理器之间的单向通信链路。 它可以携带发往远程队列管理器中任意数量的队列的消息。

通道的每个端都具有单独的定义，例如，将其定义为发送端或接收端。 简单通道由本地队列管理器上的 发送方 通道定义和远程队列管理器上的 接收方 通道定义组成。 这两个定义必须具有相同的名称，并且它们共同构成一个通道。

用于处理消息发送和接收的软件称为 消息通道代理程序 (MCA)。 在通道的每一端都有一个 消息通道代理程序 (MCA)。

每个队列管理器都应该具有 死信队列 (也称为 未传递的消息队列)。 如果无法将消息传递到其目标，那么会将这些消息放入此队列。

下图显示了队列管理器，传输队列，通道和 MCA 之间的关系:

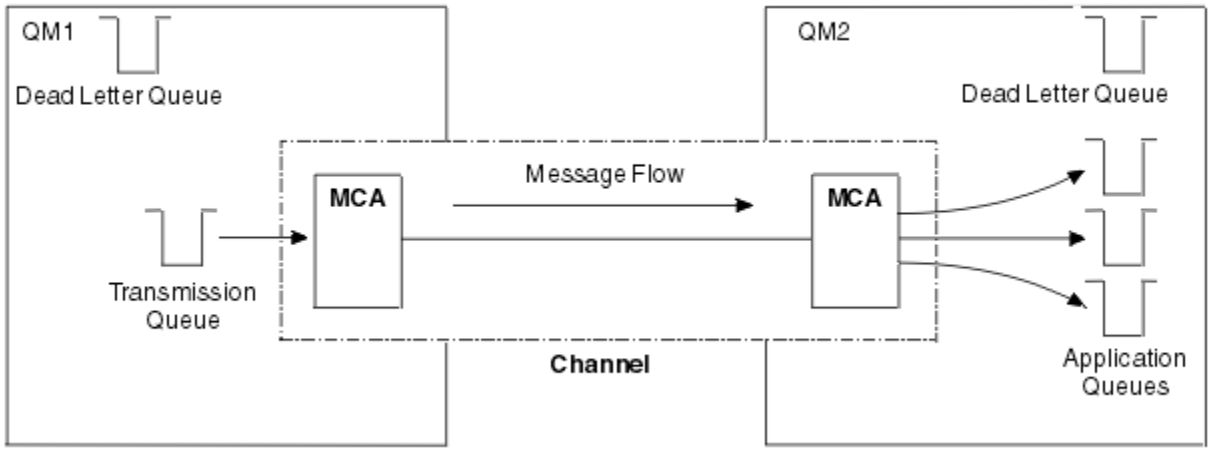


图 *5:* 发送消息

返回消息所需的组件

如果应用程序要求从远程队列管理器返回消息，那么需要定义另一个通道，以便在队列管理器之间以相反的方向运行，如下图所示:

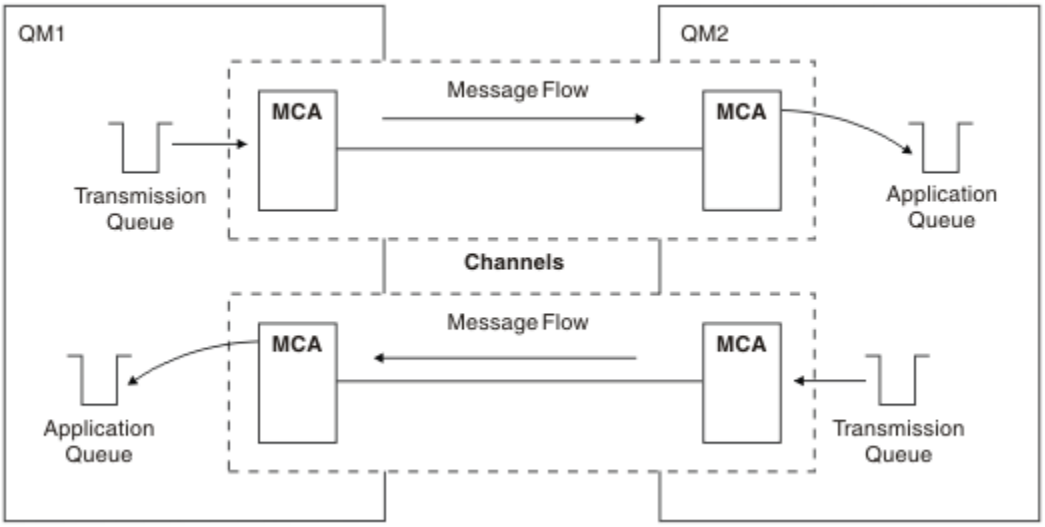


图 *6:* 双向发送消息

集群

您可以在集群中对一组队列管理器进行分组，而不是手动定义分布式排队环境中的所有连接。 执行此操作时，队列管理器可以使其托管的队列可供集群中的其他队列管理器使用，而无需针对每个目标的显式通道定义，远程队列定义或传输队列。 集群中的每个队列管理器都有一个传输队列，用于将消息传输到集群中的任何其他队列管理器。 对于每个队列管理器，您只需要定义一个集群接收方通道和一个集群发送方通道; 任何其他通道都由集群自动管理。

IBM MQ 客户机可以连接到属于集群的队列管理器，就像它可以连接到任何其他队列管理器一样。 与手动配置的分布式排队一样，您使用 MQPUT 调用将消息放入任何队列管理器中的队列。 您可以使用 MQGET 调用从本地队列中检索消息。

支持集群的平台上的队列管理器不必是集群的一部分。 您可以继续手动配置分布式排队，也可以使用集群来代替分布式排队。

使用集群的优点

集群提供了两个关键优势:

- 集群简化了 IBM MQ 网络的管理，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。 有关更多信息，请参阅 使用集群进行工作负载管理。

如何在集群中路由消息

您可以将集群视为由良心系统管理员维护的队列管理器网络。 无论何时定义集群队列，系统管理员都会根据需要在其他队列管理器上自动创建相应的远程队列定义。

您不需要进行传输队列定义，因为 IBM MQ 在集群中的每个队列管理器上提供了传输队列。 此单个传输队列可用于将消息传递到集群中的任何其他队列管理器。 您不限于使用单个传输队列。 队列管理器可以使用多个传输队列来分隔发送到集群中每个队列管理器的消息。 通常，队列管理器使用单个集群传输队列。 您可以更改队列管理器属性 DEFCLXQ，以便队列管理器对集群中的每个队列管理器使用不同的集群传输队列。 您还可以手动定义集群传输队列。

所有加入集群的队列管理器都同意以这种方式工作。 他们发送有关自己和他们托管的队列的信息，并接收有关集群的其他成员的信息。

要确保在队列管理器变为不可用时不会丢失任何信息，请在集群中指定两个队列管理器以充当 完整存储库。 这些队列管理器存储有关集群中所有队列管理器和队列的完整信息集。 集群中的所有其他队列管理器仅存储

有关这些队列管理器及其交换消息的队列的信息。 这些队列管理器称为 部分存储库。 有关更多信息，请参阅 第 47 页的『集群存储库』。

要成为集群的一部分，队列管理器必须具有两个通道: 集群发送方通道和集群接收方通道:

- 集群发送方通道是类似于发送方通道的通信通道。 必须在队列管理器上手动创建一个集群发送方通道，以将其连接到已是集群成员的完整存储库。
- 集群接收方通道是类似于接收方通道的通信通道。 必须手动创建一个集群接收方通道。 通道充当队列管理器接收集群通信的机制。

然后，将自动创建此队列管理器与集群的其他成员之间的通信所需的所有其他通道。

下图显示了名为 CLUSTER 的集群的组件:

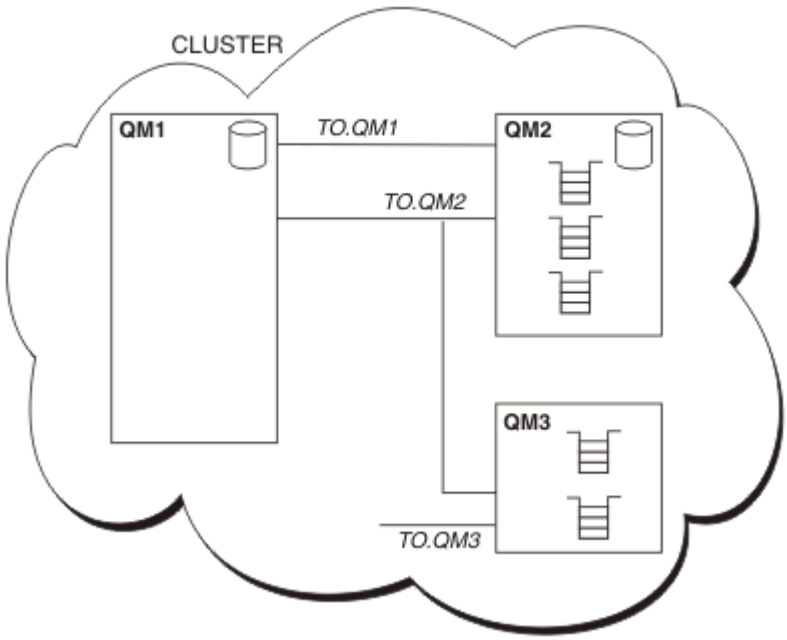


图 7: 队列管理器集群

- CLUSTER 包含三个队列管理器: QM1， QM2 和 QM3。
- QM1 和 QM2 托管有关集群中的队列管理器和队列的信息的完整存储库。
- QM2 和 QM3 托管一些集群队列，即可供集群中任何其他队列管理器访问的队列。
- 每个队列管理器都有一个名为 TO.qmgr 的集群接收方通道，可以在该通道上接收消息。
- 每个队列管理器还具有一个集群发送方通道，在该通道上，它可以将信息发送到其中一个存储库队列管理器。
- QM1 和 QM3 发送到 QM2 上的存储库， QM2 发送到 QM1 上的存储库。

分布式排队组件

分布式排队的组件包括消息通道，消息通道代理程序，传输队列，通道启动器和侦听器以及通道出口程序。 消息通道的每个端的定义可以是多种类型之一。

消息通道是将消息从一个队列管理器传递到另一个队列管理器的通道。 请勿将消息通道与 MQI 通道混淆。 有两种类型的 MQI 通道: 服务器连接 (SVRCONN) 和客户机连接 (CLNTCONN)。 有关更多信息，请参阅 通道。

消息通道的每个端的定义可以是下列其中一种类型:

- 发件人 (SDR)
- 接收器 (RCVR)

- 服务器 (SVR)
- 请求者 (RQSTR)
- 集群发送方 (CLUSSDR)
- 集群接收方 (CLUSRCVR)

使用在一端定义的其中一种类型和在另一端定义的兼容类型来定义消息通道。 可能的组合包括:

- 发送方-接收方
- 请求方-服务器
- 请求方-发送方（回调）
- 服务器-接收方
- 集群发送方-集群接收方

有关创建发送方/接收方通道的详细指示信息包含在 定义通道中。 有关设置发送方/接收方通道所需的参数的示例，请参阅适用于您的平台的 示例配置信息 。 有关定义任何类型的通道所需的参数，请参阅 DEFINE CHANNEL。

发送方-接收方通道

一个系统中的发送方启动通道，以便可以将消息发送到另一个系统。 发送方请求通道另一端的接收方启动。 发送方将消息从其传输队列发送到接收方。 接收方将消息放在目标队列上。 第 39 页的图 8 对此进行了说明。

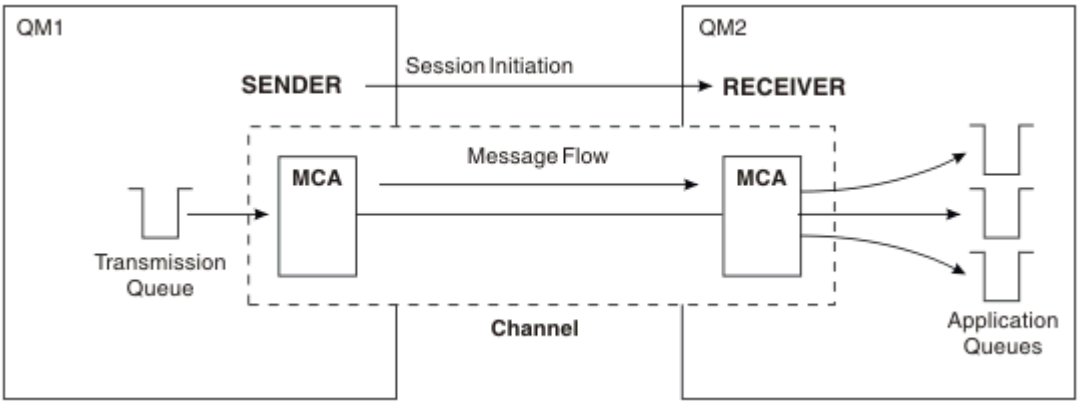


图 *8:* 一种发送方-接收方通道

请求方-服务器通道

一个系统中的请求者启动通道，以便它可以从另一个系统接收消息。 请求者请求通道另一端的服务器启动。 服务器从其通道定义中定义的传输队列向请求者发送消息。

服务器通道还可以启动通信并向请求者发送消息。 这仅适用于 标准 服务器，即在通道定义中指定了伙伴的连接名称的服务器通道。 标准服务器可以由请求者启动，也可以启动与请求者的通信。

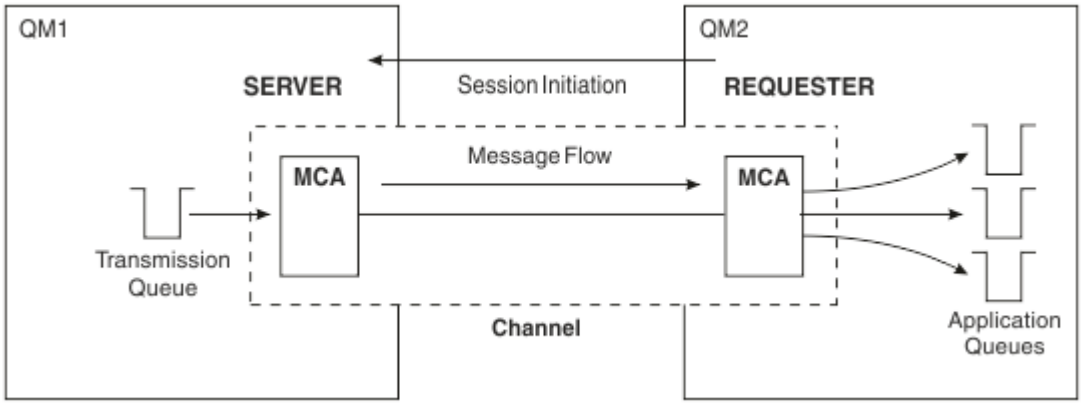


图 *9:* 请求者-服务器通道

请求方-发送方通道

请求者启动通道，发送方终止呼叫。 然后，发送方根据其通道定义 (称为 回调) 中的信息重新启动通信。 它将消息从传输队列发送到请求者。

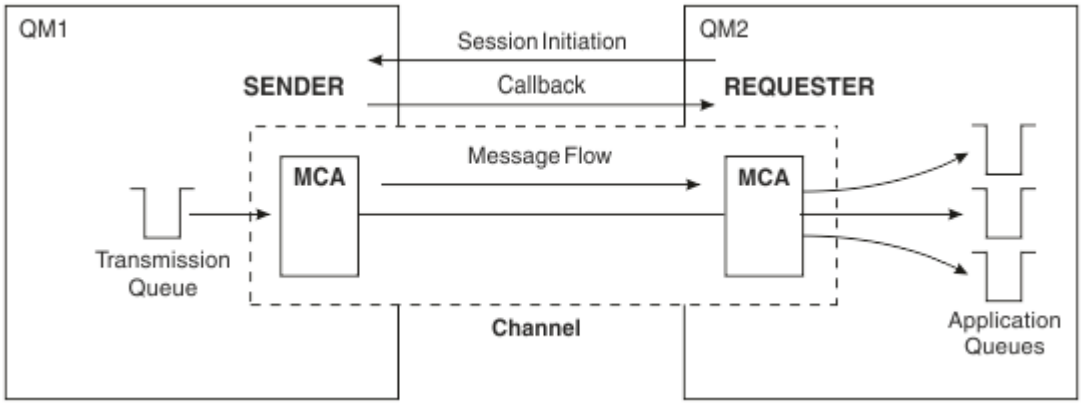


图 *10:* 请求者-发送方通道

服务器-接收方通道

这类似于发送方/接收方，但仅适用于 标准 服务器，即具有在通道定义中指定的伙伴的连接名称的服务器通道。 必须在链路的服务器端启动通道启动。 此图的插图与 第 39 页的图 8 中的插图类似。

集群发送方通道

在集群中，每个队列管理器都有一个集群发送方通道，在该通道上可以将集群信息发送到其中一个完整的存储库队列管理器。 队列管理器还可以将消息发送到集群发送方通道上的其他队列管理器。

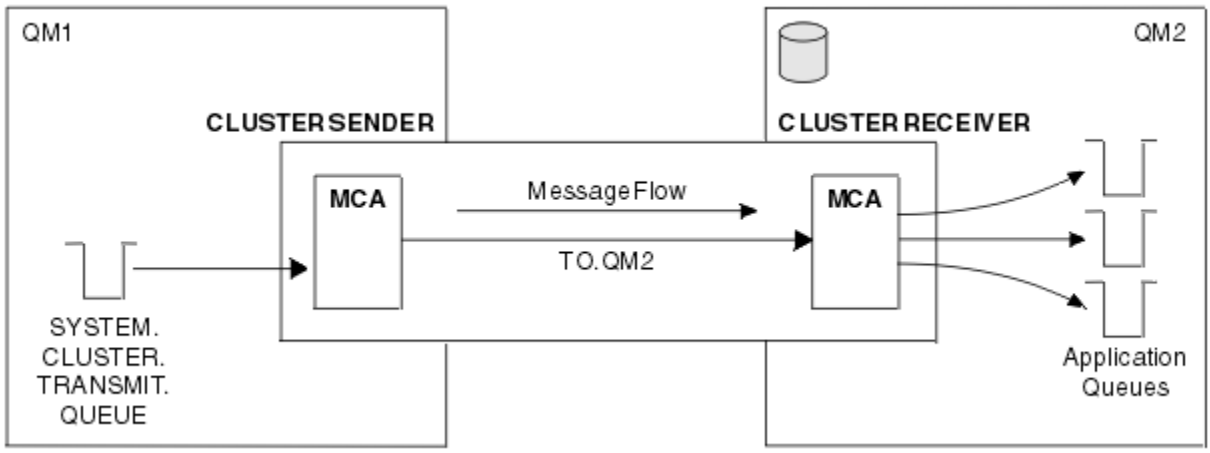


图 *11:* 集群发送方通道

集群接收方通道

在集群中，每个队列管理器都有一个集群接收方通道，在该通道上可以接收有关集群的消息和信息。 此图的插图与 第 41 页的图 11 中的插图类似。

死信队列

死信队列 (或未传递的消息队列) 是无法将消息路由到其正确目标时将消息发送到的队列。 每个队列管理器通常都有一个死信队列。

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，例如，由于该队列不存在或已满。 对于数据转换错误，在通道的发送端也会使用死信队列。 网络中的每个队列管理器通常都有一个要用作死信队列的本地队列，以便无法传递到其正确目标的消息可以存储以供以后检索。

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。 MQDLH 结构的 原因 字段包含标识消息在 DLQ 上的原因的原因码。

通常应该为每个队列管理器定义死信队列。 如果没有，并且 MCA 无法放入消息，那么会将其保留在传输队列上并停止通道。 此外，如果是快速的非持久消息 (请参阅 快速的非持久消息) 无法传递，并且目标系统上不存在死信队列，将废弃这些消息。

但是，使用死信队列可能会影响传递消息的顺序，因此您可能选择不使用这些消息。

相关任务

使用死信队列

未送达消息故障诊断

相关参考

runmqdlq (运行死信队列处理程序)

远程队列定义

远程队列定义是另一个队列管理器所拥有的队列的定义。

而应用程序只能从本地队列检索消息，它们可以将消息放在本地队列或远程队列上。 因此，除了其每个本地队列的定义外，队列管理器还可以具有 远程队列定义。 远程队列定义的优点是使应用程序能够将消息放入远程队列，而不必指定远程队列或远程队列管理器的名称或传输队列的名称。 远程队列定义使您具有位置独立性。

远程队列定义还有其他用途，稍后将进行描述。

如何访问远程队列管理器

您可能并非总是在每个源队列管理器与目标队列管理器之间具有一个通道。 两者之间还有很多其他的链接方式，包括多跳，共享通道，使用不同的通道和集群。

多跳跃

如果源队列管理器与目标队列管理器之间没有直接通信链路，那么可以在到达目标队列管理器的路上通过一个或多个 中间队列管理器 。 这称为 多跳。

您需要定义所有队列管理器之间的通道，以及中间队列管理器上的传输队列。 第 42 页的图 12 对此作了说明。

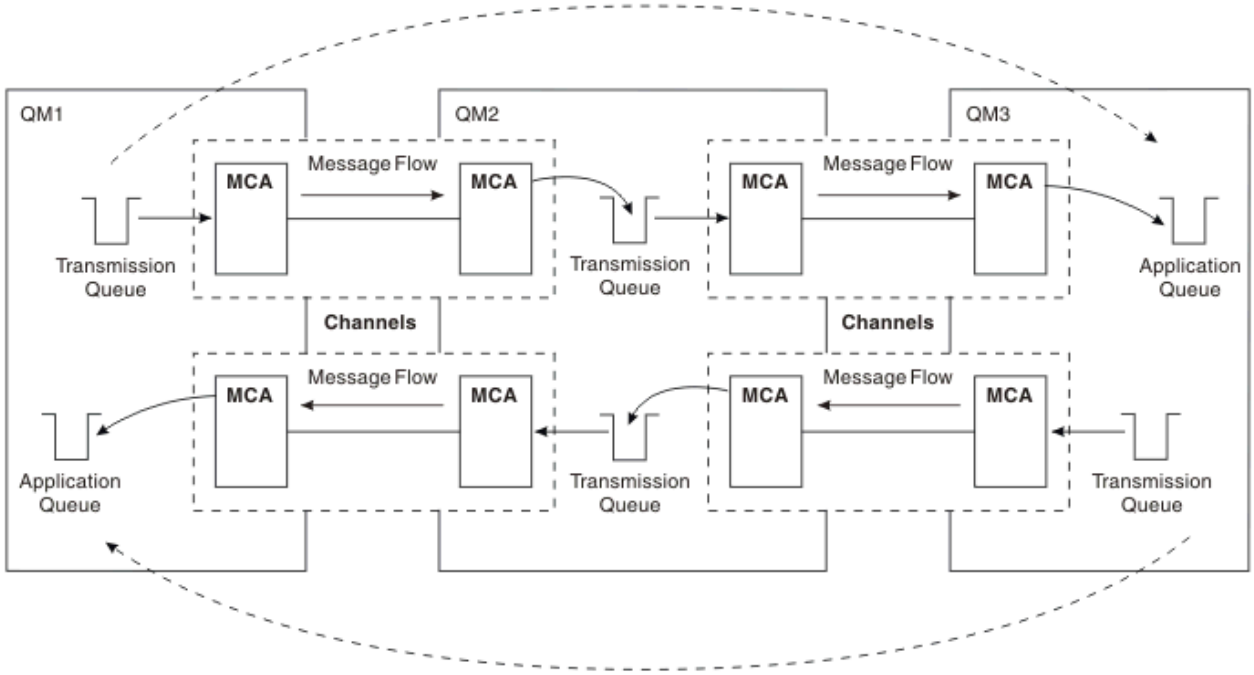


图 *12:* 通过中间队列管理器

共享通道

作为应用程序设计者，您可以选择强制应用程序指定远程队列管理器名称以及队列名称，或者为每个远程队列创建 远程队列定义 。 此定义保存远程队列管理器名称，队列名称和传输队列的名称。 无论哪种方式，来自同一远程位置的所有寻址队列的应用程序的所有消息都通过同一传输队列发送其消息。 第 42 页的图 13 对此作了说明。

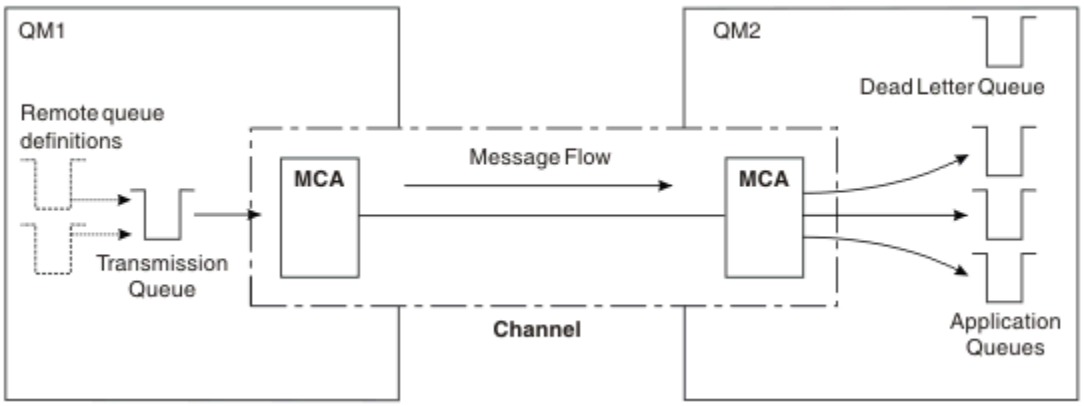


图 *13:* 共享传输队列

第 42 页的图 13 说明从多个应用程序到多个远程队列的消息可以使用同一通道。

使用不同的通道

如果要在两个队列管理器之间发送不同类型的消息，那么可以在这两个队列管理器之间定义多个通道。 有时，您需要备用通道 (可能出于安全目的) ，或者需要将交付速度与大量消息流量进行比较。

要设置第二个通道，您需要定义另一个通道和另一个传输队列，并创建指定位置和传输队列名称的远程队列定义。 然后，应用程序可以使用任一通道，但消息仍会传递到相同的目标队列。 第 43 页的图 14 对此作了说明。

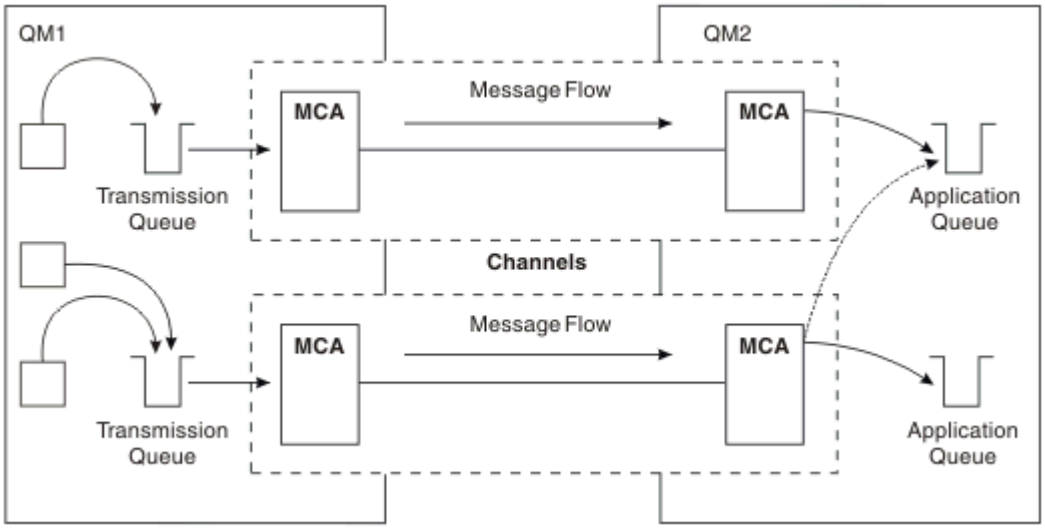


图 *14:* 使用多个通道

使用远程队列定义来指定传输队列时，应用程序 **不得** 指定位置 (即目标队列管理器) 本身。 如果这样做，那么队列管理器不会使用远程队列定义。 远程队列定义使您具有位置独立性。 应用程序可以在不知道队列所在位置的情况下将消息放入 逻辑 队列，并且您可以更改 物理 队列，而不必更改应用程序。

使用集群

集群中的每个队列管理器都定义一个集群接收方通道。 当另一个队列管理器想要向该队列管理器发送消息时，它会自动定义相应的集群发送方通道。 例如，如果集群中有多个队列实例，那么可以向托管该队列的任何队列管理器定义集群发送方通道。 IBM MQ 使用工作负载管理算法，该算法使用循环法例程来选择要将消息路由到的可用队列管理器。 有关更多信息，请参阅 集群。

寻址信息

当应用程序将发往远程队列管理器的消息放入时，本地队列管理器会在将这些消息放入传输队列之前向它们添加传输头。 此头包含目标队列和队列管理器的名称，即 寻址信息。

在单个队列管理器环境中，当应用程序打开要将消息放入的队列时，将建立目标队列的地址。 由于目标队列位于同一队列管理器上，因此不需要任何寻址信息。

在分布式排队环境中，队列管理器不仅需要知道目标队列名称，还需要知道该队列的位置 (即，队列管理器名称) 以及到该远程位置的路径 (即，传输队列)。 此寻址信息包含在传输头中。 接收通道除去传输头并使用其中的信息来查找目标队列。

如果使用远程队列定义，那么可以避免应用程序需要指定目标队列管理器的名称。 此定义指定远程队列的名称，将消息发送至的远程队列管理器的名称以及用于传输消息的传输队列的名称。

什么是别名?

别名用于为消息提供服务质量。 队列管理器别名使系统管理员能够更改目标队列管理器的名称，而不必更改应用程序。 它还使系统管理员能够更改到目标队列管理器的路由，或者设置涉及通过多个其他队列管理器 (多跳跃) 的路由。 应答队列别名为应答提供服务质量。

队列管理器别名和应答队列别名是使用具有空白 RNAME 的远程队列定义创建的。 这些定义不定义实际队列; 队列管理器使用这些定义来解析物理队列名称，队列管理器名称和传输队列。

别名定义的特征是具有空白 RNAME。

队列名称解析

每次打开队列时，都会在每个队列管理器上解析队列名称。 其目的是标识目标队列，目标队列管理器 (可能是本地队列) 以及到该队列管理器的路径 (可能为空)。 解析的名称有三个部分: 队列管理器名称，队列名称以及传输队列 (如果队列管理器是远程的)。

当存在远程队列定义时，不会引用任何别名定义。 应用程序提供的队列名称解析为目标队列，远程队列管理器和远程队列定义中指定的传输队列的名称。 有关队列名称解析的更多详细信息，请参阅 队列名称解析。

如果没有远程队列定义，并且指定了队列管理器名称或由名称服务解析，那么队列管理器将查看是否存在与提供的队列管理器名称匹配的队列管理器别名定义。 如果存在，那么其中的信息将用于将队列管理器名称解析为目标队列管理器的名称。 队列管理器别名定义还可用于确定到目标队列管理器的传输队列。

如果解析的队列名称不是本地队列，那么队列管理器名称和队列名称都包含在应用程序放入传输队列的每条消息的传输头中。

除非由远程队列定义或队列管理器别名定义更改，否则所使用的传输队列通常具有与已解析的队列管理器相同的名称。 如果您尚未定义此类传输队列，但已定义缺省传输队列，那么将使用此传输队列。

z/OS 在 z/OS 上运行的队列管理器的名称限制为四个字符。

队列管理器别名定义

队列管理器别名定义适用于打开队列放置信息的应用程序，指定队列名称**和**队列管理器名称。

队列管理器别名定义有三种用途:

- 发送信息时，重新映射队列管理器名称
- 发送信息时，更改或指定传输队列
- 接收报文时，确定本地队列管理器是否是这些报文的预定目的地

出站信息 - 重新映射队列管理器名称

队列管理器别名定义可用于重新映射 MQOPEN 调用中指定的队列管理器名称。 例如，MQOPEN 调用指定队列名称为 "THISQ，队列管理器名称为 "YOURQM。 在本地队列管理器中，有一个队列管理器别名定义，如下所示:

```
DEFINE QREMOTE (YOURQM) RQMNAME(REALQM)
```

这表明，当应用程序向队列管理器 "YOURQM 发送信息时，要使用的真正队列管理器是 "REALQM。 如果本地队列管理器是 "REALQM，它就会把信息放到本地队列 "THISQ。 如果本地队列管理器不叫 "REALQM，它就会将信息路由到一个叫 "REALQM 的传输队列。 队列管理器会更改传输头，将 "REALQM 改为 "YOURQM。

出站信息 - 更改或指定传输队列

第 45 页的图 15 显示的情况是，信息到达队列管理器 "QM1，传输头显示队列管理器 "QM3 的队列名称。 在这种情况下，"QM3 可通过 "QM2 的多重跳转到达。

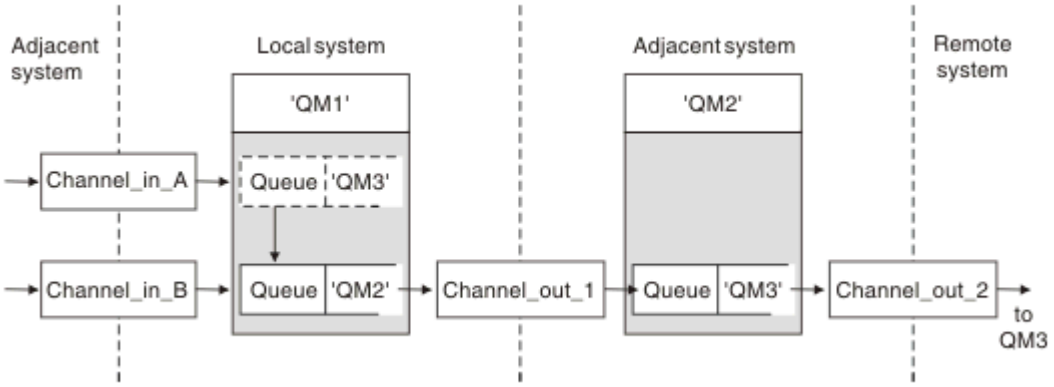


图 *15:* 队列管理器别名

QM3"的所有报文都通过队列管理器别名在 "QM1 捕获。 队列管理器别名为 "QM3，包含通过传输队列 "QM2 定义的 "QM3。 定义如下

```
DEFINE QREMOTE (QM3) RNAME(' ') RQMNAME(QM3) XMITQ(QM2)
```

队列管理器将报文放入传输队列 "QM2，但不会更改传输队列头，因为目的地队列管理器的名称 "QM3 不会更改。

所有到达 "QM1 并显示包含 "QM2 队列名称的传输标题的报文，也会被放入 "QM2 传输队列。 这样，不同目的地的信息就会被收集到一个共同的传输队列中，传送到适当的邻近系统，再转发到目的地。

入站信息 - 确定目的地

接收 MCA 打开传输标题中引用的队列。 如果队列管理器别名定义与所引用的队列管理器名称相同，那么传输头中接收到的队列管理器名称就会被该定义中的 RQMNAME 所替换。

这个过程有两个用途:

- 将信息导向另一个队列管理器
- 更改队列管理器名称，使其与本地队列管理器相同

回复队列别名定义

回复队列别名定义指定了报文描述符中回复信息的替代名称。 这样做的好处是可以更改队列或队列管理器的名称，而无需更改应用程序。

队列名称解析

当应用程序回复消息时，它会使用收到的消息描述符中的数据来查找要回复的队列名称。 发送应用程序会指明回复的发送目的地，并将此信息附加到其信息中。 这一概念必须作为应用设计的一部分加以协调。

队列名称解析是在应用程序的发送端进行的，然后才将报文放入队列。 因此，队列名称解析是在与发送信息的远程应用程序交互之前进行的。 这是唯一一种在队列未打开时进行名称解析的情况。

使用队列管理器别名解析队列名称

通常，应用程序会指定一个回复到队列，并将回复到队列管理器名称留空。 队列管理器在投放时完成自己的命名。 这种方法效果很好，除非您希望使用其他通道进行回复，例如，使用传输队列 "QM1_relief 的通道，而不是使用传输队列 "QM1 的默认返回通道。 在这种情况下，传输队列头中指定的队列管理器名称与 "真实 "队列管理器名称不匹配，而是使用队列管理器别名定义重新指定的。 为了沿其他路径返回回复，有必要使用回复到队列别名定义来映射回复到队列数据。

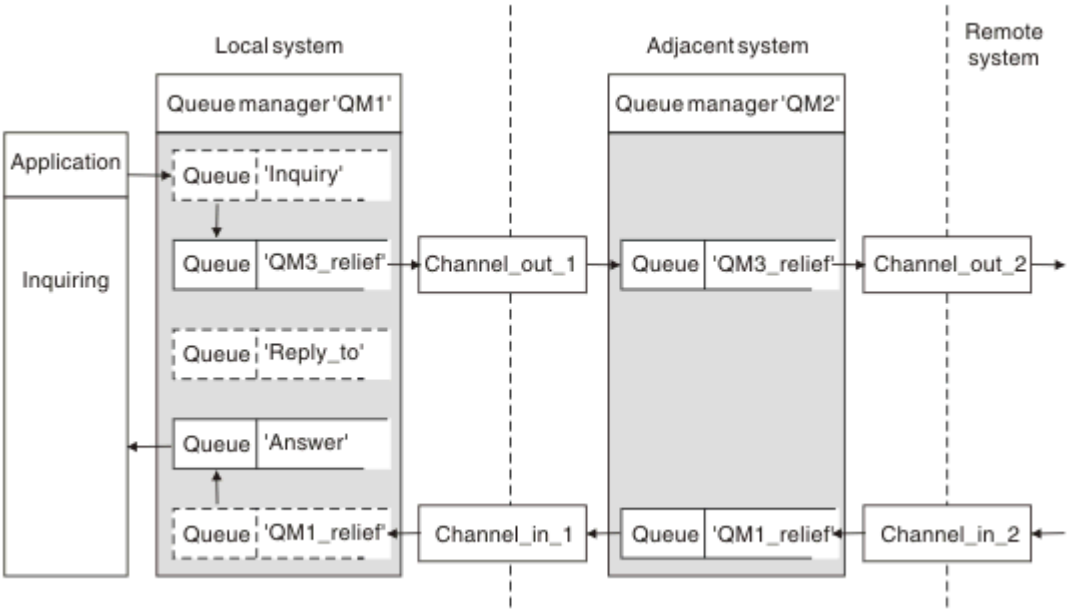


图 *16:* 用于更改回复位置的回复队列别名

在 "第 46 页的图 16 的示例中:

1. 应用程序使用 MQPUT 调用发送消息，并在消息描述符中指定以下信息:

   ```
   ReplyToQ='Reply_to'
   ReplyToQMgr=' '
   ```

   ReplyToQMgr 必须为空，以便使用回复到队列别名。
2. 您创建了一个名为 "Reply_to 的回复队列别名定义，其中包含名称 "Answer 和队列管理器名称 "QM1_relief。

   ```
   DEFINE QREMOTE ('Reply_to') RNAME ('Answer')
   RQMNAME ('QM1_relief')
   ```
3. 发送信息时，信息描述符会显示 "ReplyToQ='Answer' 和 "ReplyToQMgr='QM1_relief'。
4. 应用程序规范必须包括在队列 "Answer 而不是 "Reply_to 中查找回复的信息。

为了准备回复，您必须创建平行返回通道，并进行定义:

- 在 QM2，传输队列名为 "QM1_relief

  ```
  DEFINE QLOCAL ('QM1_relief') USAGE(XMITQ)
  ```
- 在 QM1，队列管理器别名 "QM1_relief

  ```
  DEFINE QREMOTE ('QM1_relief') RNAME() RQMNAME(QM1)
  ```

  该队列管理器别名会终止并行返回通道链，并为 QM1 捕捉信息。

如果您认为将来可能需要这样做，请确保应用程序从一开始就使用别名。 目前，这是回复到队列的普通队列别名，但以后可以改为队列管理器别名。

应答队列名称

命名回复到队列时需要谨慎。 应用程序之所以在报文中加入回复到队列名称，是因为它可以指定发送回复的队列。 创建具有此名称的回复到队列别名定义时，不能使用具有相同名称的实际回复到队列（即本地队列定

义）。 因此，回复到队列别名定义必须包含一个新队列名称以及队列管理器名称，而且应用程序规范必须包含其回复在另一个队列中找到的信息。

现在，应用程序必须从不同的队列检索信息，而这些队列与它们在放置原始信息时命名为回复到队列的队列不同。

集群组件

集群由队列管理器，集群存储库，集群通道和集群队列组成。

请参阅以下子主题以获取有关每个集群组件的信息:

相关概念
集群与分布式排队的比较
相关任务
配置队列管理器集群
设置新集群

集群存储库

存储库是有关作为集群成员的队列管理器的信息集合。

存储库信息包括队列管理器名称，它们的位置，它们的通道，它们托管的队列以及其他信息。 该信息以消息形式存储在名为 SYSTEM.CLUSTER.REPOSITORY.QUEUE 的队列上。 此队列是其中一个缺省对象。 Multi 在 多平台上，它是在创建 IBM MQ 队列管理器时定义的。 z/OS 在 IBM MQ for z/OS 上，它定义为队列管理器定制的一部分。

完整存储库和部分存储库

通常，集群中的两个队列管理器保存完整存储库。 其余队列管理器都持有部分存储库。

托管集群中每个队列管理器的完整信息集的队列管理器具有完整的存储库。 集群中的其他队列管理器具有部分存储库，其中包含完整存储库中的一部分信息。

部分存储库仅包含有关队列管理器需要与之交换消息的那些队列管理器的信息。 队列管理器请求更新它们需要的信息，因此如果它发生更改，那么完整存储库队列管理器将向它们发送新信息。 在大部分时间内，部分存储库包含队列管理器在集群中需要执行的所有信息。 当队列管理器需要某些其他信息时，它可查询完整存储库并更新其部分存储库。 队列管理器使用 SYSTEM.CLUSTER.COMMAND.QUEUE 队列来请求和接收对存储库的更新。

当 迁移 作为集群成员的队列管理器时，请先迁移完整存储库，然后再迁移部分存储库。 这是因为较旧的存储库无法存储较新发行版中引入的较新属性。 它容忍它们，但不存储它们。

集群队列管理器

集群队列管理器是属于集群的队列管理器。

一个队列管理器可以隶属于多个集群。 每个集群队列管理器都必须具有在其所属的所有集群中唯一的名称。

集群队列管理器可以托管队列，它会将这些队列通告给集群中的其他队列管理器。 但是，它不必这样做。 它可以改为将消息馈送到集群中其他位置托管的队列中，并且仅接收显式定向到该队列的响应。

z/OS 在 IBM MQ for z/OS 中，集群队列管理器可以是队列共享组的成员。 在这种情况下，它与同一队列共享组中的其他队列管理器共享其队列定义。

集群队列管理器是自主的。 他们可以完全控制他们定义的队列和通道。 其他队列管理器 (同一队列共享组中的队列管理器除外) 无法修改它们的定义。 存储库队列管理器不控制集群中其他队列管理器中的定义。 它们包含所有定义的完整集合，以便在需要时使用。 集群是队列管理器的联合。

在集群队列管理器上创建或更改定义后，会将信息发送到完整存储库队列管理器。 稍后将更新集群中的其他存储库。

完整存储库队列管理器

完整存储库队列管理器是一个集群队列管理器，用于保存集群资源的完整表示。 要确保可用性，请在每个集群中设置两个或更多完整存储库队列管理器。 完整存储库队列管理器接收集群中其他队列管理器发送的信息并更新其存储库。 它们相互发送消息以确保它们都与有关集群的新信息保持一致。

队列管理器和存储库

每个集群都有至少一个 (最好是两个) 队列管理器，这些队列管理器保存有关集群中的队列管理器，队列和通道的完整信息存储库。 这些存储库还包含来自集群中其他队列管理器的请求，用于更新信息。

其他队列管理器各自保存部分存储库，其中包含有关需要与其通信的队列和队列管理器的子集的信息。 队列管理器通过在首次需要访问另一个队列或队列管理器时进行查询来构建其部分存储库。 他们请求将有关该队列或队列管理器的任何新信息通知他们。

每个队列管理器都将其存储库信息存储在名为 SYSTEM.CLUSTER.REPOSITORY.QUEUE 的队列上的消息中。 队列管理器在名为 SYSTEM.CLUSTER.COMMAND.QUEUE 的队列上的消息中交换存储库信息。

连接集群的每个队列管理器都定义了一个集群发送方 CLUSSDR 通道，用于连接到其中一个存储库。 它会立即了解集群中的哪些其他队列管理器保存完整存储库。 从此，队列管理器可以从任何存储库请求信息。 当队列管理器将信息发送到所选存储库时，它还会将信息发送到另一个存储库 (如果有)。

当托管完整存储库的队列管理器从与其链接的其中一个队列管理器接收新信息时，将更新完整存储库。 还会将新信息发送到另一个存储库，以降低在存储库队列管理器无法使用时将其延迟的风险。 由于将两次发送所有信息，因此存储库必须废弃重复项。 每个信息项都带有一个序号，存储库使用该序号来标识重复项。 通过交换消息使所有存储库保持一致。

集群队列

集群队列是由集群队列管理器托管并可供集群中其他队列管理器使用的队列。

集群队列定义将播发给集群中的其他队列管理器。 集群中的其他队列管理器无需相应的远程队列定义即可将消息放入集群队列。 可以使用集群名称列表在多个集群中播发集群队列。

在播发队列时，集群中的任何队列管理器都可以将消息放入该队列中。 要放入消息，队列管理器必须从完整存储库中查明托管该队列的位置。 然后，它会将一些路由信息添加到消息，并将消息放到集群传输队列上。

z/OS 集群队列可以是 IBM MQ for z/OS 中队列共享组的成员所共享的队列。

相关任务

定义集群队列

共享队列与集群队列之间的比较

此信息旨在帮助您比较共享队列和集群队列，并确定哪些可能更适合您的系统。

通道启动程序成本

在集群队列中，消息由通道发送，因此除了应用程序成本外，还允许通道启动程序成本。 由于通道获取和放置消息，因此网络中存在成本。 共享队列不存在这些成本，因此，在队列共享组中的队列管理器之间移动消息时，使用的处理能力比集群队列低。

消息的可用性

放入队列时，集群队列会将消息发送到其中一个具有连接到队列管理器的活动通道的队列管理器。 在远程队列管理器上，如果用于处理消息的应用程序不工作，那么不会处理消息并等待应用程序启动。 同样，如果队列管理器已关闭，那么在队列管理器重新启动之前，队列管理器上的任何消息都不可用。 这些实例显示的消息可用性低于使用共享队列时的可用性。

使用共享队列时，队列共享组中的任何应用程序都可以获取发送的消息。 如果关闭队列共享组中的一个队列管理器，那么消息可供其他队列管理器使用，从而提供比使用集群队列时更高的消息可用性。

容量

耦合设施比磁盘更昂贵; 因此，在本地队列中存储 1,000,000 条消息的成本低于具有足够容量来存储相同数量的消息的耦合设施。

发送到其他队列管理器

共享队列消息仅在队列共享组中可用。 如果要在队列共享组外部使用队列管理器，那么必须使用通道。 您可以使用集群在多个远程分布式队列管理器之间实现工作负载均衡。

工作负载均衡

您可以使用集群来赋予哪些通道和队列管理器获取所发送消息的比例的权重。 例如，可以将 60% 的消息发送到一个队列管理器，将 40% 的消息发送到另一个队列管理器。 此实例不依赖于远程队列管理器处理工作的能力。 具有第一个队列管理器的系统可能超负荷，具有第二个队列管理器的系统可能处于空闲状态，但大多数消息仍转至第一个队列管理器。

通过共享队列，两个 CICS 系统可以获取消息。 如果一个系统超负荷，那么另一个系统将接管大部分工作负载。

集群通道

在每个完整存储库上，手动定义一个集群接收方通道和一组集群发送方通道，以连接到集群中的每个其他完整存储库。 添加部分存储库时，请手动定义集群接收方通道以及连接到其中一个完整存储库的单个集群发送方通道。 需要时，集群会自动定义其他集群发送方通道。 自动定义的集群发送方通道从接收队列管理器上的相应集群接收方通道定义中获取其属性。

集群接收方通道: CLUSRCVR

CLUSRCVR 通道定义定义了一个通道的末尾，集群队列管理器可以在该通道上接收来自集群中其他队列管理器的消息。

必须为每个集群队列管理器至少定义一个 CLUSRCVR 通道。 通过定义 CLUSRCVR 通道，队列管理器将显示它可用于接收消息的其他集群队列管理器。

CLUSRCVR 通道定义还使其他队列管理器能够自动定义相应的集群发送方通道定义。 请参阅本文的 第 49 页的『自动定义的集群发送方通道』 部分。

集群发送方通道: CLUSSDR

您可以手动定义从每个完整存储库队列管理器到集群中每个其他完整存储库队列管理器的 CLUSSDR 通道。 由完整存储库交换的所有更新仅在这些通道上流动。 通过手动定义这些通道，可以显式地控制完整存储库的网络。

将部分存储库队列管理器添加到集群时，请手动定义单个 CLUSSDR 通道以连接到其中一个完整存储库。 这与您选择的完整存储库差别不大，因为在进行初始联系之后，将根据需要自动定义队列管理器的更多集群队列管理器对象 (包括 CLUSSDR 通道)。 这使队列管理器能够将集群信息发送到任何完整存储库，并将消息发送到集群中的任何队列管理器。

如本文部分所述，自动定义的发送方通道基于集群接收方通道的配置。 因此，您在集群通道上设置的任何通道属性都应该在匹配的 CLUSSDR 和集群接收方通道上进行相同的设置，或者仅在集群接收方通道上设置。

由于先前描述的原因，您应该仅手动定义 CLUSSDR 通道。 即，最初将部分存储库连接到完整存储库，或者将两个完整存储库连接到一起。 手动配置连接到部分存储库或不在集群中的队列管理器的 CLUSSDR 通道会导致发出错误消息，例如 AMQ9427 和 AMQ9428 。 虽然这有时可能作为临时情况不可避免，例如在修改完整存储库的位置时，应该尽快删除手动定义。

自动定义的集群发送方通道

通常，将部分存储库队列管理器添加到集群时，仅手动定义队列管理器上的两个集群通道:

- 集群发送方 (CLUSSDR) 到集群的完整存储库队列管理器的通道。

- 集群接收方 (CLUSRCVR) 通道。

您定义的 CLUSSDR 通道允许队列管理器与集群进行初始联系。 进行初始联系后，集群会在需要时自动定义更多 CLUSSDR 通道。

自动定义的 CLUSSDR 通道从接收队列管理器上相应的 CLUSRCVR 通道定义中获取其属性。 即使存在手动定义的 CLUSSDR 通道，也会使用自动定义的 CLUSSDR 通道中的属性。 例如，假设您定义 CLUSRCVR 通道而不在 **CONNAME** 参数中指定端口号，并手动定义指定端口号的 CLUSSDR 通道。 当自动定义的 CLUSSDR 通道替换手动定义的通道时，端口号 (取自 CLUSRCVR 通道) 将变为空白。 将使用缺省端口号，并且通道将发生故障。

如果手动定义的 CLUSSDR 通道与相应的 CLUSRCVR 通道定义之间存在配置差异，那么某些差异将立即生效 (例如，工作负载均衡参数) ，而某些差异仅在通道重新启动 (例如， TLS 配置) 时生效。

为避免混淆，请尽可能遵守以下准则:

- 仅手动定义 CLUSSDR 通道以指向完整存储库。
- 如果您已手动定义 CLUSSDR 通道，请将其配置为与接收队列管理器上相应的 CLUSRCVR 通道定义完全匹配。

另请参阅 使用自动定义的通道。

相关概念
使用自动定义的通道
使用集群传输队列和集群发送方通道
相关任务
设置新集群
将队列管理器添加至集群

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会推送给集群的所有成员，并与本地主题相结合以创建横跨多个队列管理器的主题空间的一些部分。 这支持将一个队列管理器中的某个主题上发布的消息传递给集群中其他队列管理器的预订。

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。 集群主题可供集群中的任何队列管理器使用，但是对集群主题的任何修改必须在定义该主题的队列管理器（主机）上执行，此时会通过完整存储库将修改传播到集群的所有成员。

有关配置集群主题以使用 直接路由 或 主题主机路由的信息，以及有关集群主题继承和通配符预订的信息，请参阅 定义集群主题。

有关用于显示集群主题的命令的信息，请参阅相关信息。

相关概念
处理管理主题
使用预订
相关参考
显示 TOPIC
显示主题状态
显示

缺省集群对象

Multi 在多平台上，缺省集群对象包含在定义队列管理器时自动创建的缺省对象集中。

z/OS 在 z/OS 上，可以在定制样本中找到缺省集群对象定义。

注: 您可以通过运行 MQSC 或 PCF 命令，以与任何其他通道定义相同的方式更改缺省通道定义。 请勿更改缺省队列定义， SYSTEM.CLUSTER.HISTORY.QUEUE 除外。

SYSTEM.CLUSTER.COMMAND.QUEUE
集群中的每个队列管理器都有一个名为 SYSTEM.CLUSTER.COMMAND.QUEUE 的本地队列，用于将消息传输到完整存储库。 此消息包含有关队列管理器的任何新信息或已更改的信息，或者有关其他队列管理器的任何信息请求。 SYSTEM.CLUSTER.COMMAND.QUEUE 通常为空。

SYSTEM.CLUSTER.HISTORY.QUEUE
集群中的每个队列管理器都有一个名为 SYSTEM.CLUSTER.HISTORY.QUEUE 的本地队列。 SYSTEM.CLUSTER.HISTORY.QUEUE 用于存储集群状态信息的历史记录以用于服务目的。

在缺省对象设置中， SYSTEM.CLUSTER.HISTORY.QUEUE 设置为 PUT (ENABLED)。 要禁止历史记录收集，请将设置更改为 PUT (DISABLED)。

SYSTEM.CLUSTER.REPOSITORY.QUEUE
集群中的每个队列管理器都有一个名为 SYSTEM.CLUSTER.REPOSITORY.QUEUE 的本地队列。 此队列用于存储所有完整存储库信息。 此队列通常不为空。

SYSTEM.CLUSTER.TRANSMIT.QUEUE
每个队列管理器都有一个名为 SYSTEM.CLUSTER.TRANSMIT.QUEUE 的本地队列的定义。 SYSTEM.CLUSTER.TRANSMIT.QUEUE 是所有消息到集群中所有队列和队列管理器的缺省传输队列。 您可以通过更改队列管理器属性 DEFCLXQ 将每个集群发送方通道的缺省传输队列更改为 SYSTEM.CLUSTER.TRANSMIT. *ChannelName*。 无法删除 SYSTEM.CLUSTER.TRANSMIT.QUEUE。 它还用于定义授权检查所使用的缺省传输队列是 SYSTEM.CLUSTER.TRANSMIT.QUEUE 还是 SYSTEM.CLUSTER.TRANSMIT. *ChannelName*。

SYSTEM.DEF.CLUSRCVR
每个集群都有一个名为 SYSTEM.DEF.CLUSRCVR 的缺省 CLUSRCVR 通道定义。 SYSTEM.DEF.CLUSRCVR 用于为您在集群中的队列管理器上创建集群接收方通道时未指定的任何属性提供缺省值。

SYSTEM.DEF.CLUSSDR
每个集群都有一个名为 SYSTEM.DEF.CLUSSDR 的缺省 CLUSSDR 通道定义。 SYSTEM.DEF.CLUSSDR 用于为您在集群中的队列管理器上创建集群发送方通道时未指定的任何属性提供缺省值。

相关概念
使用缺省集群对象

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。

在点到点 IBM 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 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。 发送应用程序必须执行以下操作:

- 放置 包含应用程序需要的信息的 IBM MQ 消息。
- 将消息分配给表示信息主题的主题。
- 让 IBM 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。

下图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。 预订由队列管理器上的订户进行，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给订户。

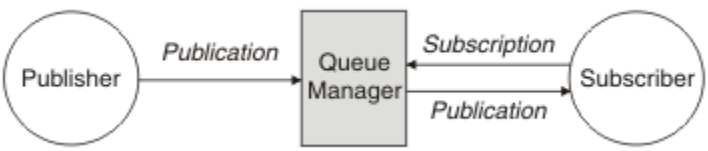


图 *17:* 简单发布/预订配置

典型的发布/预订系统在许多不同的主题上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。 应用程序可以是发布者和订户。

发布/预订消息传递与点到点之间的另一个显着差异是，发送到点到点队列的消息仅由单个使用应用程序处理。 发布到发布/预订主题的消息 (其中多个订户已注册兴趣) 由每个感兴趣的订户处理。

发布/预订组件

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

典型的发布/预订系统在许多不同的主题上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。 应用程序可以是发布者和订户。

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。

已发布的信息将在 IBM MQ 消息中发送，并且该信息的主题由其 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 IBM MQ 工具用于分发消息，因此应用程序可以使用可供现有 IBM 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。

出版商和出版物

在 IBM 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 IBM 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 IBM 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。

相关概念

发布/预订集群中保留的发布的设计注意事项

同步点下的出版物

在 IBM 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 IBM 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。

相关概念

克隆和共享的预订

相关参考

如何定义 sharedSubscription 属性的示例

受管队列和发布/预订

创建预订时，可以选择使用受管排队。 如果使用受管排队，那么在创建预订时将自动创建预订队列。 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。 这是因为 受管预订 是 IBM MQ 处理预订的受管预订。 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，预订名称 (**SubName**) 是必需的。 预订名称在队列管理器中必须唯一，以便可用于标识预订。 如果您有意关闭与预订的连接 (使用 MQCO_KEEP_SUB 选项) 或者已与队列管理器断开连接，那么在指定要恢复的预订时，必须使用此标识方法。 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。

您可以使用 **DURSUB** 主题属性来指定是否可以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。

相关概念

克隆和共享的预订

相关任务

使用 JMS 2.0 共享预订

相关参考

如何定义 sharedSubscription 属性的示例

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。

在修改消息以传递给每个订户之前，将根据发布程序放置的消息对选择字符串进行求值。 在选择字符串中使用可能在发布操作过程中修改的字段时，请小心。 例如， MQMD 字段 `UserIdentifier`， `MsgId` 和 `CorrelId`。

选择字符串不应引用队列管理器在发布操作中添加的任何消息属性字段 (请参阅 发布/预订消息属性) ，但包含发布主题字符串的消息属性 MQTopicString 除外。

相关概念

选择字符串规则和限制

主题

主题是发布/预订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 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 56 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 57 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 62 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 63 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，可以选择提供主题字符串或引用主题对象。 或者可以同时执行这两种操作，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 61 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。

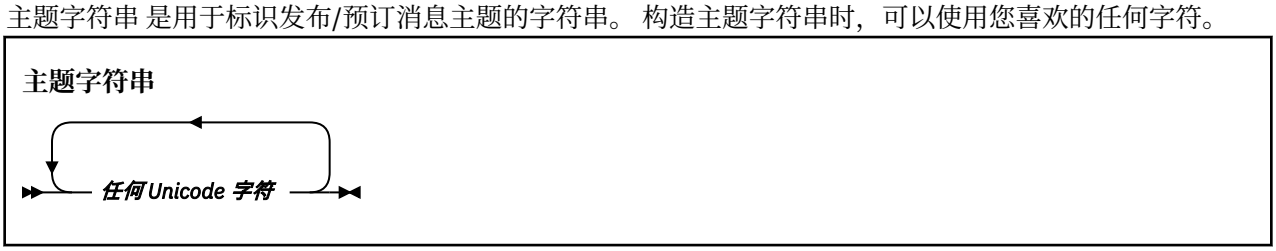
在基于内容的发布/预订中，您可以通过提供用于搜索每条消息内容的选择字符串来定义要接收的消息。 IBM MQ 提供了基于内容的发布/预订的中间形式，使用消息选择器来扫描消息属性而不是消息的完整内容，请参阅 选择器。 消息选择器的原型用途是预订主题，然后使用数字属性限定选择。 选择器使您能够指定您只对特定范围内的值感兴趣; 不能使用字符或基于主题的通配符来执行任何操作。 如果确实需要根据消息的完整内容进行过滤，那么需要使用 IBM Integration Bus。

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 IBM WebSphere MQ 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 57 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 57 页的『主题字符串示例』 显示了对 '#'的合理使用。

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
```

相关参考

TOPIC

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 IBM WebSphere MQ 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 IBM 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预订中的字符串是否必须与发布内容中的字符串完全匹配。 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

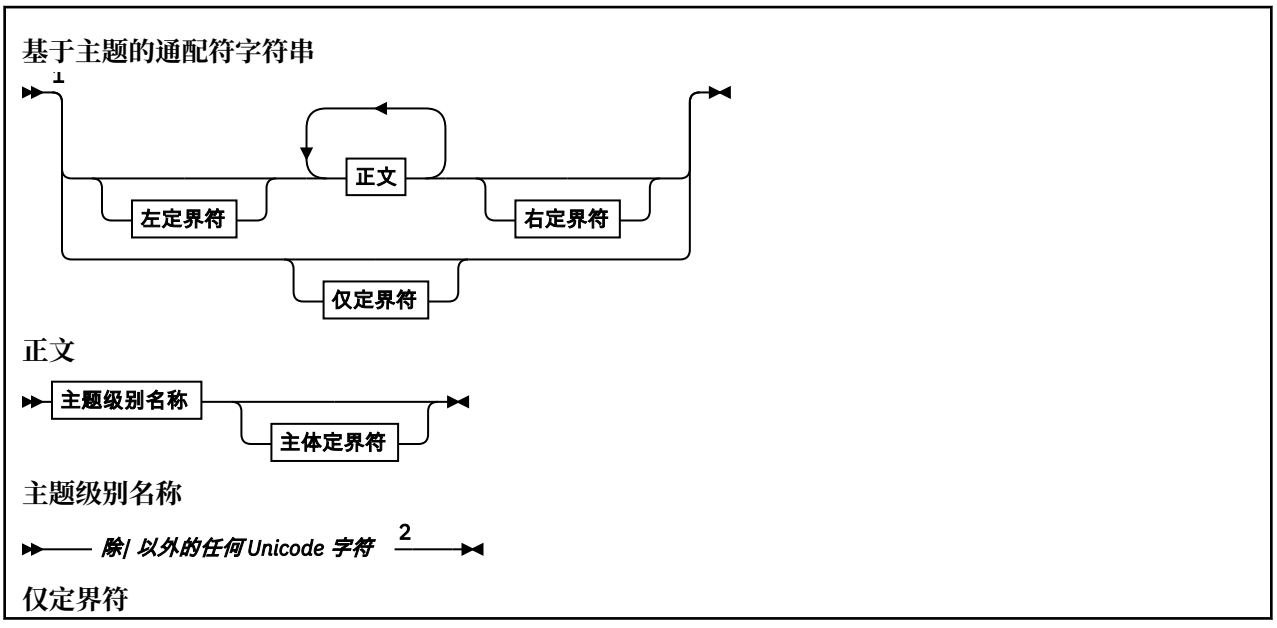
- 多级别通配符用于订阅中。 在发布中使用时，会将其视为文字。
- 多级通配符 '#' 用于匹配主题中的任意数量的级别。 例如，使用示例主题树，如果预订 'USA/Alaska/#'，那么将接收有关主题 'USA/Alaska' 和 'USA/Alaska/Juneau'的消息。
- 多级别通配符可以不表示任何级别，也可以表示多个级别。 因此， 'USA/#' 还可以与奇异 'USA'匹配，其中 '#' 表示零级别。 因为没有要分隔的级别，所以主题级别分隔符在此上下文中没有含义。
- 多级别通配符仅当单独指定或紧随主题级别分隔符指定时才有效。 因此， '#' 和 'USA/#' 是将 '#' 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 59 页的『当基于主题的通配符无效时』，以了解更多信息。

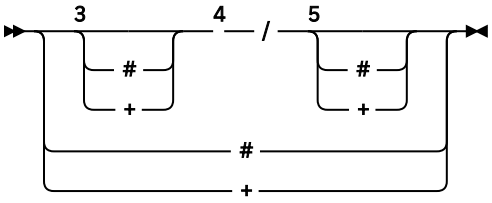
单一级别通配符

- 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 59 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。 有关更多信息，请参阅 第 59 页的『当基于主题的通配符无效时』。

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。

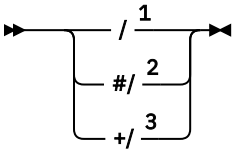




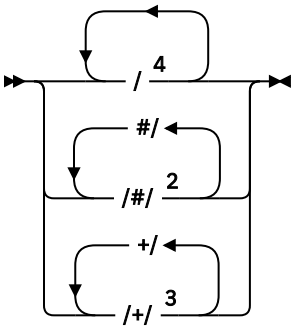
注:

[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。

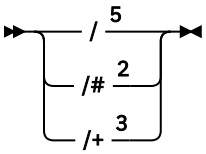
左定界符



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的字符匹配来选择主题。

您可以使用字符串 "'*'选择主题层次结构中多级的所有主题。 在基于字符的通配符方案中使用 "'*'相当于使用基于主题的通配符字符串 "'#'

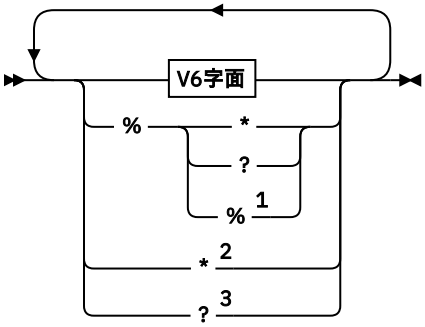
' *x*/*/*y* ' is equivalent to ' *x*/#/*y* ' in the topic-based scheme, and selects all topics in the topic hierarchy between levels ' *x* and *y* ', where ' *x* ' and ' *y* ' are topic names that are not in the set of levels returned by the wildcard.

在基于主题的方案中，'/+/'在基于字符的方案中没有完全对应的字符。 'IBM/*/Results'也会选择 "'IBM/Patents/Software/Results'。 只有当层次结构中每一级的主题名称集合都是唯一的时候，使用这两种方案构造的查询才能始终产生相同的匹配结果。

一般情况下，基于字符的方案中的 "'*'和 "'?'在基于主题的方案中并不对应。 基于主题的方案不使用通配符进行部分匹配。 基于字符的通配符订阅 "'IBM/*ware/Results'没有与主题对应的字符。

注: 使用字符通配符订阅的匹配速度比使用主题订阅的匹配速度慢。

基于字符的通配符字符串



V6 字面

任何 Unicode 字符，?* 和%'*

注：

[1] 指转义下面的字符，使其被视为一个字面意义。 '%'后必须有 "'*'、"'?'"或 "'%'"。 请参阅第 57 页的『主题字符串示例』。
[2] 指匹配订阅中的零个或多个字符。
[3] 表示与订阅中的一个字符完全匹配。

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。

下表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	ObjectString/TOPICSTR 单独使用。
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 62 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 62 页的图 18 显示了具有一个根主题的主题树的示例。

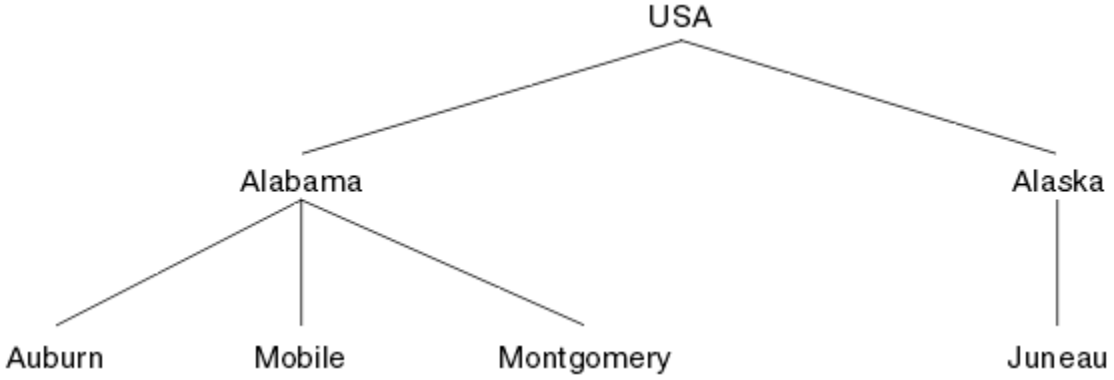


图 *18:* 主题树的示例

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 62 页的图 18 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。

相关任务
减少主题树中不需要的主题数

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 63 页的图 19 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 *19:* 主题树的可视化

第 63 页的图 20 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 *20:* 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:

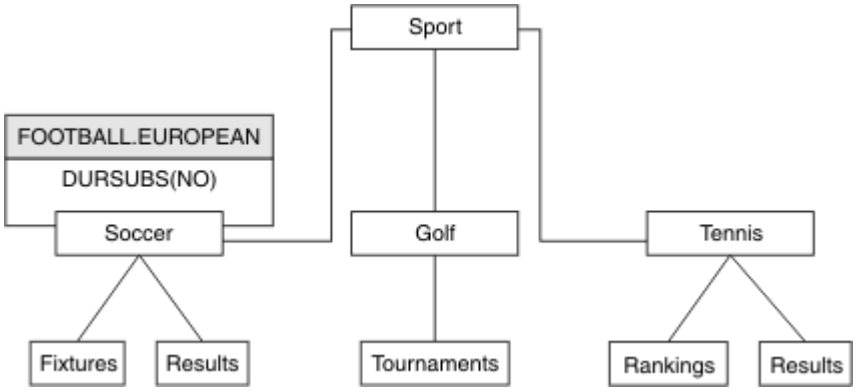


图 *21:* 与 *Sport/Soccer* 主题关联的管理主题对象的可视化

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 64 页的图 22 中扩展:

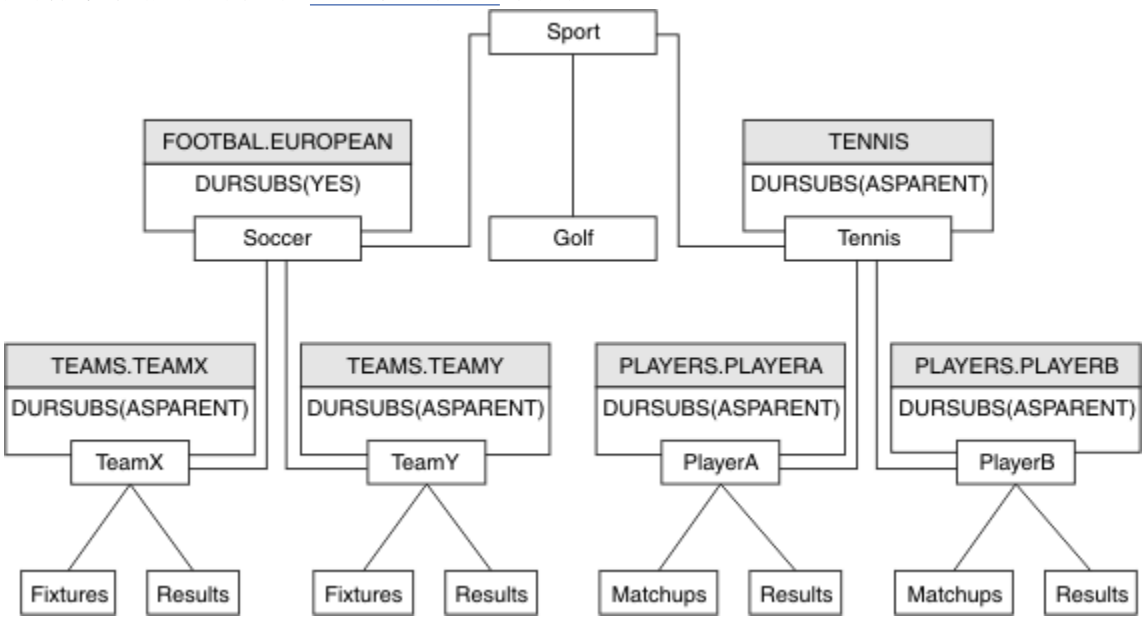


图 *22:* 具有多个管理主题对象的主题树

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 68 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 65 页的图 23 中显示的主题树结构。

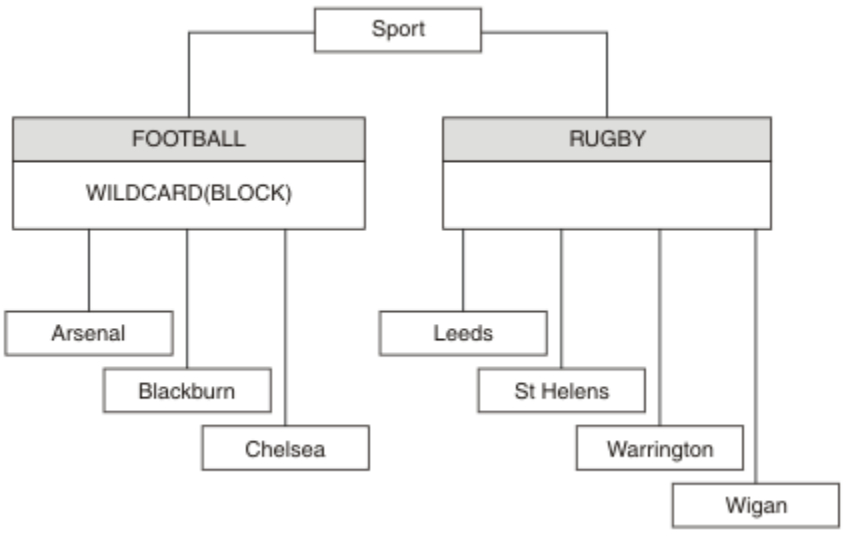


图 *23:* 使用 通配符 属性 *BLOCK* 的主题树

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 69 页的图 27。 在 第 70 页的图 30 中运行发布命令以创建一些发布。

```
pub QMA
```

图 *24:* 发布到 *QMA*

第 66 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	注意
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 65 页的图 23 中的示例中，预订 (例如 Sports/Football/#) 可以接收发布。

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 68 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 66 页的图 24。 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 以创建一批出版物; 请参阅 第 66 页的图 24。

第 67 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 67 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	注意
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 67 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	注意
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

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/预订代理迁移到更高版本的 IBM MQ 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

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/预订代理迁移到更高版本的 IBM MQ 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 65 页的图 23 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 65 页的图 23 中的主题对象相对应。 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*

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。 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。

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。 要使流可用于在启用了排队发布/预订的 IBM MQ 上运行的队列管理器，必须手动添加该流。

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。

添加流主题中描述了添加流的任务。 由于两个原因，您可能需要手动添加流。

1. 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。
2. 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

通过创建队列并为其提供与流相同的名称，将在 IBM MQ 中模拟已排队的发布/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，请为与流关联的主题选择另一主题字符串。 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 IBM MQ 中，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。

预订点和主题

指定的预订点由主题和主题对象进行仿真。

要手动添加预订点，请参阅 添加预订点。

IBM MQ 中的预订点

IBM MQ 将预订点映射到 IBM 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以未更改的方式映射到 IBM 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。

IBM MQ 应用程序不受预订点影响。 如果应用程序使用从其中一个匹配主题对象继承信息的主题，那么该应用程序将使用匹配的预订点与排队的应用程序进行互操作。

示例

现有 WebSphere Message Broker (现在称为 IBM Integration Bus) 迁移到 IBM MQ 的发布/预订应用程序创建了两个主题对象 GBP 和 USD 以及相应的主题字符串 'GBP' 和 'USD'。

已迁移到 IBM MQ 上运行的主题 NYSE/IBM/SPOT 的现有发布程序，这些发布程序使用预订点 USD 在主题 USD/NYSE/IBM/SPOT 上创建发布。 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 在 IBM MQ 发布/预订程序中预订美元现货价格。 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);
```

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 。

单个队列管理器发布/预订配置的示例

第 73 页的图 31 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。

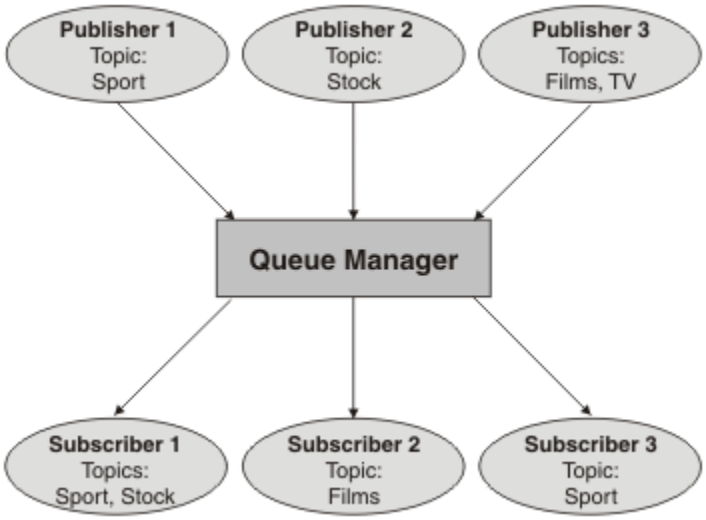


图 *31:* 单个队列管理器发布/预订示例

分布式发布/预订网络

每个队列管理器将发布到主题的消息与本地创建的预订 (已预订该主题) 相匹配。 您可以配置队列管理器网络，以便将连接到一个队列管理器的应用程序发布的消息传递到网络中其他队列管理器上创建的匹配预订。 这需要通过队列管理器之间的简单通道进行其他配置。

分布式发布/预订配置是一组连接在一起的队列管理器。 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。 将队列管理器连接在一起时，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。 为了说明这一点，下图向 第 72 页的『单个队列管理器发布/预订配置的示例』中描述的配置添加了第二个队列管理器。

- 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。

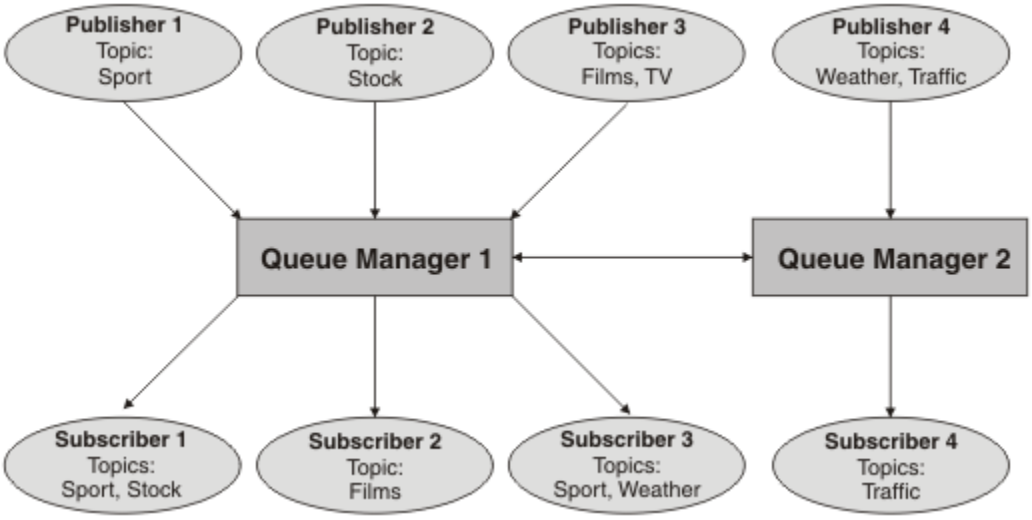


图 *32:* 具有两个队列管理器的发布/预订示例

您可以手动连接父层次结构和子层次结构中的队列管理器，也可以创建发布/预订集群，并让 IBM MQ 为您定义许多连接详细信息。 您还可以组合使用这两种拓扑，例如，通过在层次结构中将多个集群连接在一起。

发布/预订集群概述

发布/预订集群是向集群添加了一个或多个主题对象的标准集群。 当您在集群中的任何队列管理器上定义 管理主题对象，并通过指定集群名称使该主题对象成为集群对象时，该主题的发布者和订户可以连接到集群中的任何队列管理器，并且发布的消息将通过队列管理器之间的集群通道路由到订户。

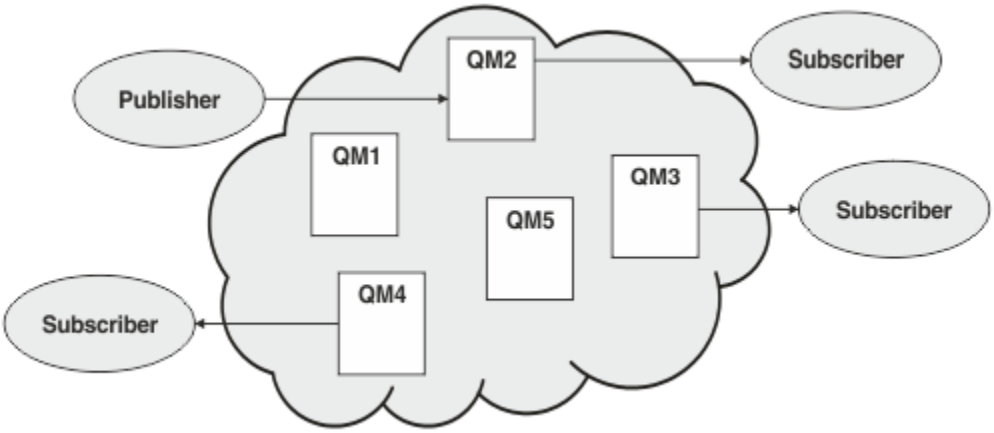


图 *33:* 发布/预订集群 *(publish/subscribe cluster)*

有两种方法可配置如何在集群中路由发布/预订消息:

- 直接路由 (direct routing)
- 主题主机路由 (topic host routing)

当您配置直接路由的集群主题时，在一个队列管理器上发布的消息将直接从该队列管理器发送到集群中任何其他队列管理器上的每个预订。 这可以为发布提供最直接的路径，但确实会导致集群中的所有队列管理器了解所有其他队列管理器，每个队列管理器都可能在它们之间建立集群通道。

使用主题主机路由时，在一个队列管理器上发布的消息将从该队列管理器发送到主管受管主题对象定义的队列管理器。 主题主机队列管理器会将消息路由至集群中其他任何队列管理器上的每个预订。 如果发布者或订户不在主题主机队列管理器上，那么这将导致发布的路径较长。 但是，好处是只有主题主机队列管理器才知道集群中的所有其他队列管理器，并且可能与它们一起建立了集群通道。

有关更多信息，请参阅 第 75 页的『发布/预订集群』。

发布/预订层次结构概述

发布/预订层次结构是由通道连接到分层结构中的一组队列管理器。 每个队列管理器标识其 父 队列管理器，如 将队列管理器连接到发布/预订层次结构中所述。

主题的发布者和订户可以连接到层次结构中的任何队列管理器，并且消息使用分层队列管理器连接在它们之间流动。

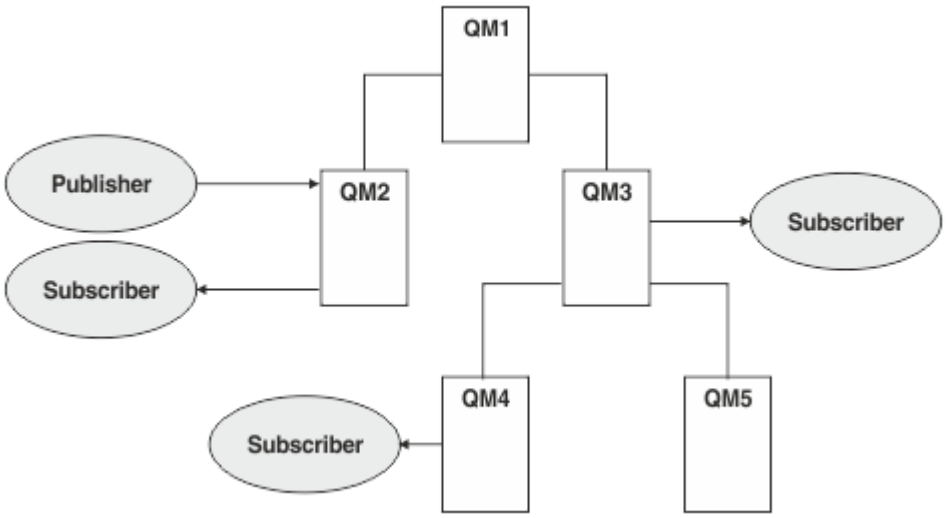


图 *34:* 发布/预订层次结构

在上图中，传递到 QM3 和 QM4 上的订户的出版物已从 QM2 路由到 QM1 ，然后传递到 QM3 ，最后传递到 QM4。

层次结构使您能够直接控制层次结构中每个队列管理器之间的关系。 这允许对从发布者到订户的消息路由进行细颗粒度控制，并且在具有受限连接的队列管理器网络之间进行路由时特别有用。 您应该仔细考虑每个队列管理器的可用性和功能，通过这些队列管理器将消息从发布者路由到订户。

有关更多信息，请参阅 第 77 页的『发布/预订层次结构』。

队列管理器之间的发布分布

除了路由选项外，还有两种方法可以在队列管理器网络中分发发布内容:

- 仅将发布从一个队列管理器发送到当前主管该发布的预订的队列管理器。
- 将每个发布发送到所有队列管理器，并使其与预订相匹配。

前者导致仅在必要时发送发布消息，但需要在队列管理器之间共享一定级别的预订知识。 后者不需要共享预订知识，但会导致在队列管理器之间发送不必要的发布消息。

缺省情况下， IBM MQ 使用前一种方法，在此方法中，发布仅发送到具有预订的队列管理器。 预订知识以代理预订形式在队列管理器之间传播。 这取决于预订的分布和生存期，以及发布的频率，因为在分布式发布/预订拓扑中使用的效率最高。 请参阅 发布/预订网络中的预订性能。

相关概念

第 62 页的『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。

发布/预订层次结构方案

相关任务

设计发布/预订集群

发布/预订集群

发布/预订集群是互连队列管理器的标准集群，在该集群上，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。 提供两个选项用于在发布/预订集群中路由发布：直接路由和主题主机路由。 您选择的路由取决于集群的大小和期望的活动模式。

用于发布/预订消息传递的集群与标准 IBM 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。 有关更多信息，请参阅 集群。

要配置标准的队列管理器集群以进行发布/预订消息传递，可在集群中的一个队列管理器上定义一个或多个受管主题对象。 要使主题成为集群主题，请使用集群的名称配置 **CLUSTER** 属性。 执行此操作时， 主题树 中的发布者或订户在该点或以下位置使用的任何主题将在集群中的所有队列管理器之间共享，并且发布到主题树的集群分支的消息将自动路由到集群中其他队列管理器上的预订。

在发布者队列管理器和其他队列管理器之间仅发送每条消息的一个副本，而不考虑目标队列管理器上该消息的订户数。 到达具有一个或多个预订的队列管理器时，将在所有预订之间复制消息。

任何加入集群的队列管理器都会自动了解集群主题，并且该队列管理器上的发布者和订户会自动参与集群。

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。

提供两个选项用于在发布/预订集群中路由发布：直接路由和主题主机路由。 要选择在集群中使用的消息路由，请将受管主题对象上的 **CLROUTE** 属性设置为以下值之一：

- **DIRECT**
- **TOPICHOST**

缺省情况下，主题路由为 **DIRECT**。 这是 IBM MQ 8.0 之前的唯一选项。 当在队列管理器上配置直接路由集群主题时，集群中的所有队列管理器都可识别集群中的所有其他队列管理器。 在执行发布和预订操作时，每个队列管理器都可以直接连接到集群中的任何其他队列管理器。

从 IBM MQ 8.0 开始，您可以改为将主题路由配置为 **TOPICHOST**。 在您使用主题主机路由时，集群中的所有队列管理器都会知晓托管了路由主题定义的集群队列管理器（即，已定义主题对象的队列管理器）。 在执行发布和预订操作时，集群中的队列管理器只会连接到这些主题主机队列管理器，而不会彼此直接连接。 主题主机队列管理器负责将发布从执行发布的队列管理器路由至具有匹配预订的队列管理器。

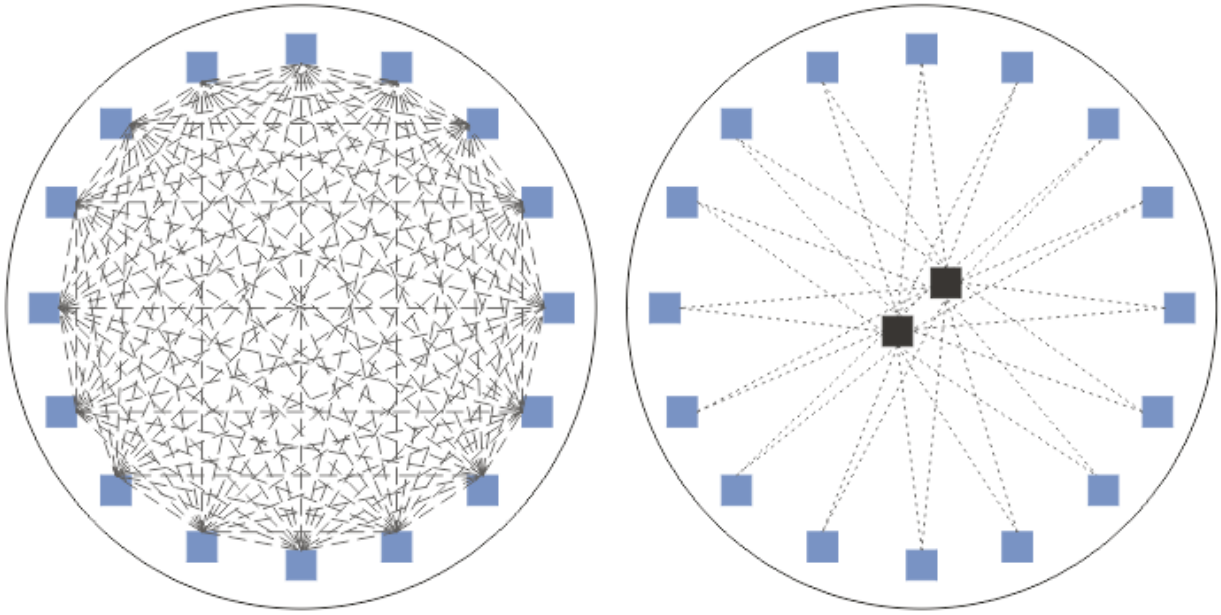


图 *35:* 直接路由和主题主机路由

直接路由概述

配置受管主题对象以进行直接路由时，只需要在集群中的其中一个队列管理器上定义主题对象，所有队列管理器才能了解该对象。 定义主题的队列管理器选项不会影响主题的发布/预订消息传递行为。

每条消息都直接从发布者队列管理器流向集群中其他队列管理器上的每个预订，而不是通过任何中间队列管理器传递。

缺省情况下，消息仅发送到集群中托管一个或多个预订的其他队列管理器。

- 这依赖于每个队列管理器直接向集群中的所有其他队列管理器通知当前具有一个或多个预订的所有主题。 这将导致集群中的所有队列管理器都知道正在预订的所有主题，以及托管预订的任何队列管理器都将建立到每个其他队列管理器的通道。 这与每个队列管理器是否具有发布程序无关。
- 可以通过更改为向集群中的所有队列管理器发送所有发布的模型 (无论它们是否具有预订) 来除去所有队列管理器上的每个单独预订主题的知识。 这将减少预订知识流量，但可能会增加发布流量以及每个队列管理器建立的通道数。 请参阅 发布/预订网络中的预订性能。

使用直接路由的集群主题的发布/预订消息流可以跨多个发布/预订集群，方法是将每个集群中的一个队列管理器添加到发布/预订层次结构中。 请参阅 组合多个集群的主题空间。

有关直接路由的更详细探索，请参阅 发布/预订集群中的直接路由。

主题主机路由概述

当为主题主机路由配置了受管主题对象时，来自集群中队列管理器的发布将通过配置了主题对象的队列管理器 ("主题主机") 进行路由，并从该队列管理器路由到存在预订的队列管理器。

- 这依赖于每个队列管理器向所有主题主机通知当前具有一个或多个预订的每个主题。 托管预订的任何队列管理器都会为与预订相关的主题建立到每个主题主机的通道。
- 出于发布/预订目的，不会使非主题托管队列管理器知道集群中的其他非主题托管队列管理器，并且不会为此目的在它们之间建立通道。
- 如果发布应用程序连接到托管主题的队列管理器，那么已发布的消息将直接路由到已创建匹配预订的队列管理器，而不需要额外的 "中继段"。 同样，如果在托管主题的唯一队列管理器上创建匹配的预订，那么发布到该主题的消息将直接路由到该队列管理器，而不需要额外的中继段。
- 将满足与发布程序相同的队列管理器上的预订，而不首先将发布路由到主题对象的主机。

对于集群队列，多个队列管理器可以配置相同的管理主题对象。 这将提供更高的消息路由可用性，并通过工作负载均衡进行水平缩放。 对于主题主机路由的主题对象，当多个队列管理器为主题树的同一分支配置同一指定主题时，托管预订的每个队列管理器将使每个主题主机了解预订的主题。

- 发布消息时，会将其发送到其中一个主题主机队列管理器，以转发到预订托管队列管理器。 主题主机队列管理器的选择遵循与集群点到点队列相同的缺省工作负载均衡规则。
- 如果发布队列管理器无法联系一个或多个主题主机队列管理器，那么会将消息路由到剩余的可用主题主管队列管理器。

主题树的路由分支中的主题的每个发布都将转发到其中一个主题主机，即使集群中的任何位置都没有该主题的预订也是如此。 缺省情况下，消息仅从此处发送到集群中托管一个或多个预订的其他队列管理器。

- 这依赖于向每个主题主机队列管理器通知集群中每个队列管理器上的所有预订主题字符串。
- 可以通过更改为将路由到主题主机的所有发布发送到集群中所有队列管理器的模型来除去每个单独预订的主题的知识，而不考虑它们是否具有预订。 这将减少预订知识流量，但可能会增加发布流量，并可能增加使用每个主题托管队列管理器建立的通道数。 请参阅 发布/预订网络中的预订性能。

使用主题主机路由的集群主题的发布/预订消息流 **不能** 通过使用发布/预订层次结构跨多个发布/预订集群。

有关主题主机路由的更详细探索，请参阅 发布/预订集群中的主题主机路由。

发布/预订层次结构

通过使用通道将队列管理器链接在一起，然后定义队列管理器对之间的子/父关系，构建发布/预订层次结构。 消息通过层次结构中的直接关系从发布者流向预订。 请注意，这可能意味着要到达多个 “中继段” 。

在任何一对队列管理器之间仅发送消息的一个副本，而不考虑目标队列管理器上消息的订户数。 到达具有一个或多个预订的队列管理器时，将在所有预订之间复制消息。

缺省情况下，仅将消息发送到层次结构中的其他队列管理器，这些队列管理器位于另一个队列管理器上的预订的路由上:

- 这依赖于每个队列管理器通知当前具有一个或多个预订的所有主题的每个直接关系 (在此队列管理器上或在其另一个关系上)。 这将导致层次结构中的所有队列管理器都知道要预订的所有主题。
- 可以将此行为更改为始终向层次结构中的所有队列管理器发送发布，而不考虑是否存在任何预订。 这样就不需要在层次结构中传播预订信息，但可以增加发布流量。

创建集群时，需要注意不要创建导致消息在网络中永久循环的循环。 不能在层次结构中创建此类循环。

每个队列管理器都必须具有唯一的队列管理器名称。

发布/预订消息流可以跨多个发布/预订集群。 为此，请将每个集群中的一个队列管理器添加到发布/预订层次结构中。

有关更详细的探索，请参阅 在发布/预订层次结构中路由。

发布/预订网络中的代理预订

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。 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。 您无需明确创建代理预订，队列管理器会替您执行此操作。

您可以将队列管理器一起连接到发布/预订集群或发布/预订层次结构中。 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 73 页的『分布式发布/预订网络』。

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快速变化的情况下，必须考虑代理预订传播的开销。 除了本主题的其余部分中描述的自动聚集外，您还可以进行手动配置更改，以进一步限制已连接队列管理器之间的代理预订和发布的流，并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 发布/预订网络中的预订性能。

代理预订不包含本地预订所使用的任何选择器，可以简化包含通配符的预订主题字符串。 这可能会导致发布与实际预订不匹配的代理预订相匹配，从而在队列管理器之间产生额外的发布流。 托管预订的队列管理器会过滤掉此类差异，以便不会将其他发布返回给预订。

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。

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。

相关概念

分布式发布/预订网络中进行回路检测

代理预订中的通配符

预订可以使用主题字符串中的通配符来匹配发布中的多个主题字符串。

预订可以使用两个通配符模式: 基于主题的 和 基于字符的。 请参阅第 57 页的『通配方案』。

在 IBM MQ 中，通配符预订的所有代理预订都将转换为使用基于主题的通配符。 如果找到基于字符的通配符，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 68 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 65 页的图 23 中显示的主题树结构。

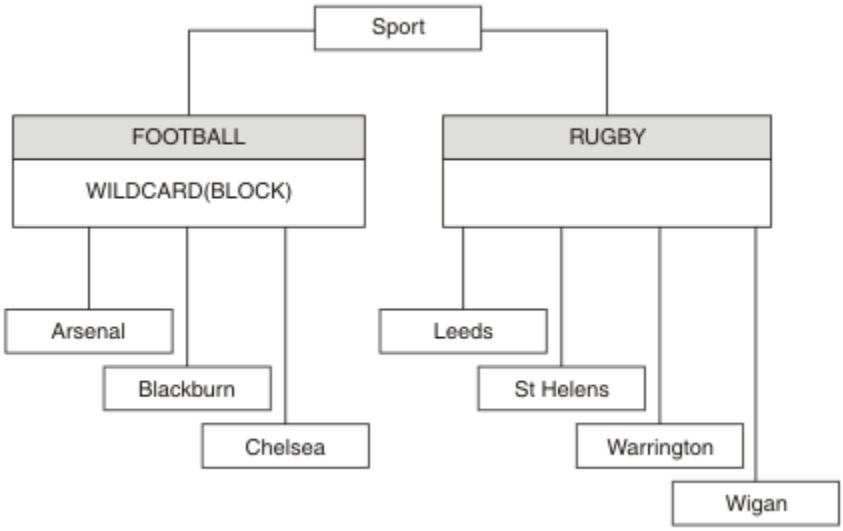


图 *36:* 使用 通配符 属性 *BLOCK* 的主题树

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 69 页的图 27。 在 第 70 页的图 30 中运行发布命令以创建一些发布。

```
pub QMA
```

图 *37:* 发布到 *QMA*

第 66 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	注意
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

表 6: QMA 上收到的出版物 (继续)			
预订	**主题字符串**	**收到的出版物**	**注意**
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 65 页的图 23 中的示例中，预订 (例如 Sports/Football/#) 可以接收发布。

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 68 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 66 页的图 24。 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 以创建一批出版物; 请参阅 第 66 页的图 24。

第 67 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 67 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**	**注意**
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

表 7: *QMA* 上收到的出版物 (继续)			
预订	**主题字符串**	**收到的出版物**	**注意**
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 67 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**	**注意**
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

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/预订代理迁移到更高版本的 IBM MQ 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

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/预订代理迁移到更高版本的 IBM MQ 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 65 页的图 23 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 *38:* 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 65 页的图 23 中的主题对象相对应。 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 *39:* 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 *40:* 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 *41:* 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 *42:* 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 *43:* 发布*: pub.bat*

相关概念

通配符预订和保留发布

发布作用域

配置发布/预订集群或层次结构时，发布的作用域将进一步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。

使用发布/预订集群时，发布范围主要由主题树中某些点的集群主题对象定义控制。 必须设置发布作用域以允许发布流至集群中的其他队列管理器。 仅当需要对特定队列管理器上的特定主题进行细粒度控制时，才应限制集群主题的发布范围。

使用发布/预订层次结构时，发布的作用域主要由此属性与 预订作用域 属性组合控制。

PUBSCOPE 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。

所有
发布将传递到本地订户以及连接到发布/预订集群或层次结构中的远程队列管理器的订户。 这些出版物称为 全局出版物。

ASPARENT
使用主题树中父主题的 **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。

相关概念
第 63 页的『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。

相关任务
配置分布式发布/预订网络

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。

使用发布/预订集群时，预订的作用域主要由主题树中某些点的集群主题对象的定义控制。 必须设置预订作用域以允许代理预订流向集群中的其他队列管理器。 仅当需要对特定队列管理器上的特定主题进行细粒度控制时，才应限制集群主题的预订范围。

使用发布/预订层次结构时，预订的作用域主要由此属性与 发布作用域 属性组合控制。

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。

所有
代理预订传播到发布/预订集群或层次结构中的远程队列管理器，并且订户接收本地和远程发布。

ASPARENT
使用主题树中父主题的 **SUBSCOPE** 设置。

当主题的预订作用域设置为 ALL(直接设置或通过 ASPARENT 解析) 时，针对该主题的各个预订可以通过在创建预订时指定 MQSO_SCOPE_QMGR 来将其作用域限制为 QMGR 。 对作用域为 QMGR 的主题的预订无法将作用域扩大到 ALL。

相关概念
第 63 页的『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。

相关任务
配置分布式发布/预订网络

主题空间

主题空间是您可以预订和发布的主题集。 分布式发布/预订拓扑中的队列管理器具有一个主题空间，该主题空间可能包含已在该拓扑中的已连接队列管理器上预订和发布的主题。

注: 有关队列管理器中的主题 (例如管理主题对象，主题字符串和主题树) 的概述，请参阅第 56 页的『主题』。 除非另有指定，否则当前文章中对 主题 的进一步引用将引用 主题字符串 。

主题最初是通过以下任一方式创建的:

- 以管理方式定义主题对象或持久预订时。
- 当应用程序动态地创建新主题的发布或预订时。

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。

还会为发布/预订集群中的集群主题的预订传播代理预订。 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。 当主题是 直接路由 集群主题时，会将代理预订转发到集群的所有成员。 当主题是 主题主机路由 集群主题时，仅会将代理预订转发到已定义集群主题对象的集群中的队列管理器。 有关更多信息，请参阅第 75 页的『发布/预订集群』。

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。

以下列表中描述了三个发布/预订拓扑的主题空间:

- 第 86 页的『案例 1。 发布/预订集群』.
- 第 87 页的『案例 2。 发布/预订层次结构』.

在单独的主题中，以下配置任务描述了如何组合主题空间。

- 在发布/预订集群中创建单个主题空间。
- 组合多个集群的主题空间。
- 组合和隔离多个集群中的主题空间。
- 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。

在多个队列管理器上需要集群主题对象的相同指定定义时 (例如，使用 主题主机路由时)，所有定义在必要时匹配很重要。 有关更多信息，请参阅 在发布/预订集群中创建单个主题空间。

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。

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。

可能会意外地在集群中的不同队列管理器上创建两个具有不同主题字符串的相同命名主题对象的定义。 这可能会导致混淆行为，因为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。 请参阅 同名的多个集群主题定义，以获取有关此重要点的更多信息。

案例 2。 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 IBM MQ 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。 没有将主题划分为局部主题和全局主题。

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。

相关概念

第 84 页的『发布作用域』
配置发布/预订集群或层次结构时，发布的作用域将进一步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。

第 85 页的『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。

相关任务

配置分布式发布/预订网络

IBM MQ 多点广播

IBM MQ 多点广播提供具有低等待时间和高扇出的可靠多点广播消息传递。

多点广播是一种高效的发布/预订消息传递形式，因为它可以扩展为大量订户，而不会对性能产生不利影响。 IBM MQ 使用应答、否定应答和序号来支持可靠的多点广播消息传递，以实现具有高扇出的低等待时间消息传递。

IBM MQ 多点广播的公平传递支持几乎同时的传递，从而确保不偏向任何接收方。 IBM MQ 多点广播使用网络来传递消息，因此无需发布/预订引擎对数据进行扇出。 将主题映射到组地址后，不需要队列管理器，因为发布程序和订户可以在对等方式下运行。 这就允许减少队列管理器服务器上的负载，而队列管理器服务器将不再是潜在的故障点。

初始多点广播概念

IBM MQ 通过使用 "通信信息" (COMMINFO) 对象，可以轻松地将多点广播集成到现有系统和应用程序中。 两个 TOPIC 对象字段支持快速配置现有 TOPIC 对象以支持或忽略多点广播流量。

多点广播所需的对象

以下信息是 IBM MQ 多点广播所需的两个对象的简要概述：

COMMINFO 对象

COMMINFO 对象包含与多点广播传输关联的属性。 有关 COMMINFO 对象参数的更多信息，请参阅 DEFINE COMMINFO。

必须设置的唯一 COMMINFO 字段是 COMMINFO 对象的名称。 然后，此名称用于向主题标识 COMMINFO 对象。 必须检查 COMMINFO 对象的 **GRPADDR** 字段，以确保该值是有效的多点广播组地址。

主题对象

主题是发布/预订消息中发布的信息的主题，通过创建 TOPIC 对象来定义主题。 有关 TOPIC 对象参数的更多信息，请参阅 DEFINE TOPIC。

通过更改以下 TOPIC 对象参数的值，可以将现有主题用于多点广播: **COMMINFO** 和 **MCAST**。

- **COMMINFO** 此参数指定多点广播通信信息对象的名称。
- **MCAST** 此参数指定在主题树中的此位置是否允许多点广播。 缺省情况下， **MCAST** 设置为 ASPARENT 表示从父代继承主题的多点广播属性。 将 **MCAST** 设置为 ENABLED 允许此节点上的多点广播流量。

多点广播网络和主题

以下信息概述了对具有不同类型的预订和主题定义的预订执行的操作。 这些示例都假定 TOPIC 对象 **COMMINFO** 参数设置为有效 COMMINFO 对象的名称:

主题集已启用多点广播
如果主题字符串 **MCAST** 参数设置为 ENABLED，那么允许来自支持多点广播的客户机的预订并进行多点广播预订，除非:

- 它是来自支持多点广播的客户机的持久预订。
- 它是来自支持多点广播的客户机的非受管预订。
- 它是来自不支持多点广播的客户机的预订。

在这些情况下，将进行非多点广播预订，并将 预订降级为正常发布/预订。

已禁用设置为多点广播的主题
如果主题字符串 **MCAST** 参数设置为 DISABLED，那么将始终进行非多点广播预订，并且 预订降级为正常发布/预订。

主题设置为仅多点广播
如果主题字符串 **MCAST** 参数设置为 ONLY，那么将允许来自支持多点广播的客户机的预订，并且将进行多点广播预订，除非:

- 这是持久预订: 已拒绝持久预订，原因码为 2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED
- 它是非受管预订: 非受管预订被拒绝，原因码为 2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR
- 它是来自不支持多点广播的客户机的预订: 这些预订被拒绝，原因码为 2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY
- 这是来自本地绑定应用程序的预订: 将拒绝这些预订，原因码为 2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY

Windows Linux AIX MQ Telemetry 概述

MQ Telemetry 包含作为队列管理器一部分的遥测 (MQXR) 服务，可自行编写或免费下载的遥测客户机以及命令行和资源管理器管理界面。 遥测指收集远程设备数据并管理各种远程设备。 通过 MQ Telemetry ， 您可以将数据收集和设备控制与 Web 应用程序集成。

MQ Telemetry 是 IBM MQ 的组件。 这些版本的升级基本上是安装更高版本的 IBM MQ。

可从 Eclipse Paho 和 MQTT.org 免费获取样本应用程序。 请参阅 IBM MQ Telemetry Transport 样本程序。

由于 MQ Telemetry 是 IBM MQ 的组件，因此可以随主产品一起安装 MQ Telemetry ， 也可以在安装主产品之后安装。 有关迁移信息，请参阅 在 Windows 上迁移 MQ Telemetry 和 在 Linux 上迁移 MQ Telemetry。

MQ Telemetry 中包含以下组件:

遥测通道
使用遥测通道来管理 MQTT 客户机与 IBM MQ 的连接。 遥测通道使用新的 IBM MQ 对象 (例如 SYSTEM.MQTT.TRANSMIT.QUEUE) 与 IBM MQ 进行交互。

遥测 (MQXR) 服务
MQTT 客户机使用 SYSTEM.MQXR.SERVICE 遥测服务来连接到遥测通道。

IBM MQ Explorer 支持 MQ Telemetry
可以使用 IBM MQ Explorer 来管理 MQ Telemetry 。

文档

MQ Telemetry 文档包含在标准 IBM MQ 产品文档中。 Java 和 C 客户机的 SDK 文档在产品文档中以 Javadoc 和 HTML 形式提供。

遥测概念

您从周围的环境收集信息以决定要执行的操作。 作为消费者，你先检查你在店里有什么，然后再决定买什么食物。 您想知道如果现在离开，在预订连接之前，一段旅程将需要多长时间。 你检查一下你的症状，然后再决定是否去看医生。 在决定是否等待之前，请检查公交车何时到达。 这些决策的信息直接来自计量表和设备，来自纸上的文字或屏幕，以及来自您的信息。 在任何时候，您都需要收集信息，汇集信息，对其进行分析，并对其采取行动。

如果信息来源广泛分散或无法获取，收集最准确的信息就会变得困难和昂贵。 如果您要进行的更改很多，或者很难进行更改，那么这些更改不会进行，或者在效果较差时进行。

如果通过将具有数字技术的设备连接到互联网来大大降低从广泛分散的设备收集信息和控制这些设备的成本，将会怎样? 可以利用互联网和企业的资源对信息进行分析。 您有更多机会做出明智的决策并采取行动。

技术趋势以及环境和经济压力正在推动这些变化发生:

1. 由于标准化和与低成本数字处理器的连接，连接和控制传感器和执行器的成本正在降低。
2. 互联网和互联网技术越来越多地用于连接设备。 在一些国家，在与互联网应用程序的连接数量上，移动电话超过了个人计算机。 其他设备肯定在关注。
3. 互联网和互联网技术使应用程序更容易获取数据。 轻松访问数据正在推动使用数据分析，将来自传感器的数据转化为更多解决方案中有用的信息。
4. 智能使用资源往往是减少碳排放和成本的更快和更便宜的方法。 替代办法: 寻找新的资源，或开发新的技术来利用现有资源，可能是长期的解决办法。 从短期来看，开发新技术或寻找新资源，通常比改进现有解决方案风险更高，速度更慢，成本更高。

示例

一个示例显示了这些趋势如何创造与环境智能交互的新机会。

The International Convention for the Safety of Life at Sea (SOLAS) requires Automatic Identification System (AIS) to be deployed on many ships. 300 吨以上的商船和客船都需要。 AIS 主要是沿海航运的避撞系统。 它被海洋当局用来监测和控制沿海水域。

世界各地的爱好者正在部署低成本的 AIS 跟踪站，并将沿海航运信息放到互联网上。 其他爱好者正在编写将来自 AIS 的信息与来自互联网的其他信息相结合的应用程序。 结果将放在 Web 站点上，并使用 Twitter 和 SMS 进行发布。

在一个应用程序中，来自南安普敦附近 AIS 站的信息与船舶所有权和地理信息相结合。 该应用程序将有关轮渡到达和离开的实时信息提供给 Twitter。 使用南安普敦和怀特岛之间的渡轮的普通通勤者使用推特或短信订阅新闻订阅。 如果饲料显示他们的轮渡运行较晚，通勤者可能会延迟他们的出发，并在其停靠时间晚于其预定到达时间时捕获轮渡。

要获取更多示例，请参阅第 91 页的『遥测用例』。

相关任务

安装 MQ Telemetry
管理 MQ Telemetry
在 Windows 上迁移 MQ Telemetry
在 Linux 上迁移 MQ Telemetry
为 MQ Telemetry 开发应用程序
对 MQ Telemetry 问题进行故障诊断

相关参考

MQ Telemetry 参考

Windows Linux AIX MQ Telemetry 简介

人们，企业和政府越来越希望使用 MQ Telemetry 与我们生活和工作的环境进行更智能的交互。 MQ Telemetry 将各种设备连接到因特网和企业，并降低为智能设备构建应用程序的成本。

什么是 MQ Telemetry?

- 它是 IBM MQ 的一个功能部件，用于将 IBM MQ 提供的通用消息传递主干扩展至各种远程传感器，执行器和遥测设备。 MQ Telemetry 扩展 IBM MQ ，使其能够将智能企业应用程序，服务和决策制定者与受检测设备网络互连。
- MQ Telemetry 的核心部件为:

  MQ Telemetry (MQXR) 服务。
  此服务在 IBM MQ 服务器中运行，并使用 IBM MQ Telemetry Transport (MQTT) 协议与遥测设备进行通信。

  您编写的 MQTT 应用程序。
  这些应用程序控制遥测设备与 IBM MQ 队列管理器之间传递的信息以及为响应该信息而执行的任何操作。 要帮助创建这些应用程序，请使用 MQTT 客户机库。

它具有什么功能?

- MQTT 是一种开放式消息传递传输，允许为各种设备创建 MQTT 实现。
- MQTT 客户机可以在资源有限的小型占用量设备上运行。
- MQTT 在带宽较低，发送数据的成本较高或可能很脆弱的网络上高效工作。
- 保证消息传递并使其与应用程序分离。
- 应用程序员不需要具备通信编程知识。
- 可以与其他消息传递应用程序交换消息。 这些应用程序可以是另一个遥测应用程序，也可以是 MQI , JMS 或企业消息传递应用程序。

如何使用?

- 提供了用于处理样本 IBM MQ Telemetry Transport v3 客户机应用程序 (`mqttv3app.jar`) 的样本脚本。请参阅 IBM MQ Telemetry Transport 样本程序。
- 使用 IBM MQ Explorer 及其关联工具来管理 IBM MQ 的遥测功能。
- 使用客户机库可帮助您创建连接到队列管理器并使用发布/预订消息传递的 MQTT 应用程序。
- 将应用程序和客户机库分发到要运行应用程序的设备。

它是如何工作的?

- MQTT 是发布预订协议。 MQTT 客户机应用程序可以将消息发布到 MQTT 服务器，或者预订由连接到 MQTT 服务器的应用程序发送的消息。
- MQTT 客户机应用程序使用实现 MQTT 消息传输的客户机库。
- 基本 MQTT 客户机应用程序类似于标准 MQ 客户机，但可以在更广泛的各种平台和网络上运行。
- MQ Telemetry (MQXR) 服务将 IBM MQ 队列管理器转换为 MQTT 服务器。
- 当 IBM MQ 队列管理器充当 MQTT 服务器时，连接到队列管理器的其他应用程序可以从 MQTT 客户机预订和接收消息。
- 队列管理器充当路由器将消息从发布应用程序分发到预订应用程序。
- 可以在不同类型的客户机应用程序之间分发消息。 例如，在 Telemetry 客户机与 JMS 客户机之间。

注: MQ Telemetry 替换在 WebSphere Message Broker V 7 中撤销的 SCADA 节点 (现在称为 IBM Integration Bus) 并在 Windows， Linux 和 AIX 上运行。

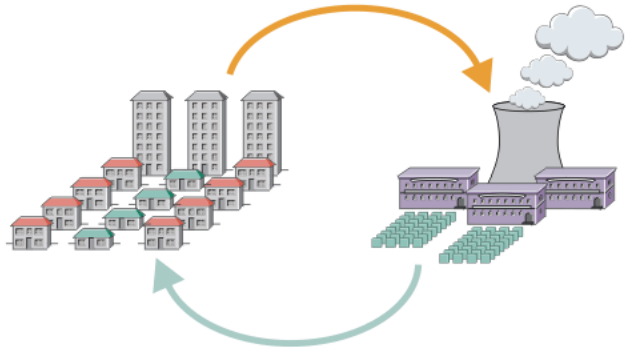
Windows Linux AIX 遥测用例

遥测是对数据的自动感应和度量以及对远程设备的自动控制。 重点在于数据从设备到中央控制点的传输。 遥测还包含向设备发送配置和控制信息。

MQ Telemetry 使用 MQTT protocol 连接小型设备，并使用 IBM MQ 将这些设备连接到其他应用程序。 MQ Telemetry 弥合了设备与互联网之间的差距，使其更容易构建 "智能解决方案"。 智能解决方案为监控和控制设备的应用程序解锁了互联网上和企业应用程序中可用的丰富信息。

下图演示了 MQ Telemetry 的一些典型用法:

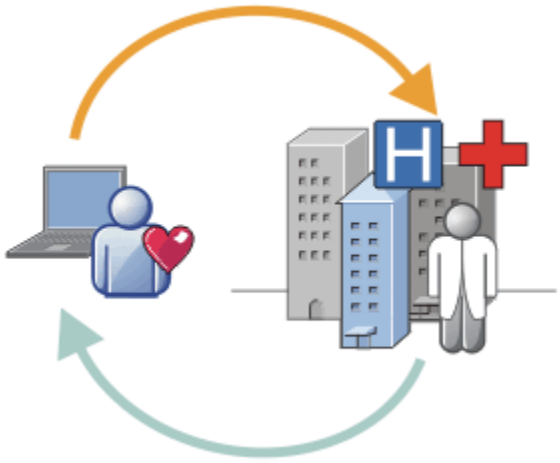
遥测: 智能电力



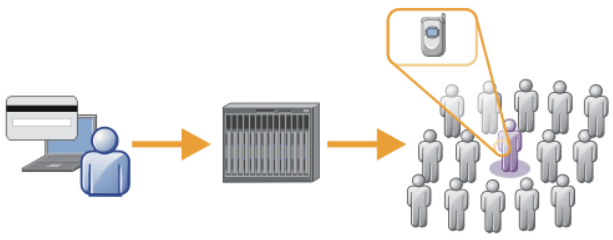
- 包含发送到服务提供者的能源使用数据的 MQTT 消息。
- MQ Telemetry 根据对能源使用情况数据的分析发送 CONTROL COMMANDS。
- 有关更多信息，请参阅以下用例: 第 93 页的『遥测用例: 家庭能源监测与控制』

遥测: 智慧健康服务

- MQ Telemetry 将健康数据发送到您的医院和医生。
- 可以根据运行状况数据分析来发送 MQTT 消息警报或反馈。
- 有关更多信息，请参阅以下用例: 第 92 页的『遥测用例: 家庭患者监控』



遥测: Crowd 中的一个



- 将简单的卡交易发送到银行的服务器。
- MQ Telemetry 识别千人中的一人，提醒客户已使用他们的卡。
- MQ Telemetry 可以使用最简单的信息输入，并找到该个人。

子主题中描述的用例来自实际示例。 它们说明了一些使用遥测的方法，以及遥测技术必须解决的一些常见问题。

Windows Linux AIX 遥测用例: 家庭患者监控

在 IBM 与心脏患者护理系统上的医疗保健提供者之间的协作中，植入的心内膜除颤器与医院进行通信。 使用 RF 遥测将有关患者和植入设备的数据传输到患者家中的 MQTT 设备。

通常，传输会在夜间进行到位于床头的发射器。 发射器通过电话系统将数据安全地传输到医院，在医院中分析数据。

该系统减少了患者必须看医生的次数。 它检测患者或设备何时需要关注，在发生紧急情况时，它会向待命医生发出警报。

IBM 与医疗保健提供者之间的协作具有许多遥测用例所共有的特征:

Invisibility

该设备无需用户干预，只需提供电源，电话线，并在一天中的部分时间与该设备接近。 其操作可靠，使用简单。

为了消除患者设置设备的需求，设备供应商对设备进行预配置。 患者必须将其插入。 患者消除了配置，简化了设备的操作，减少了设备配置错误的机会。

MQTT 客户机作为设备的一部分嵌入。 设备开发者将 MQTT 客户机实现嵌入到设备中，开发者或供应商将 MQTT 客户机配置为预配置的一部分。

MQTT 客户机作为 Java SE JAR 文件交付，开发者在其 Java 应用程序中包含该 JAR 文件。 对于非 Java 环境 (例如，此环境) ，设备开发者可以使用已发布的 MQTT 格式和协议以不同语言实现客户机。 或者，开发者可以使用其中一个作为 Windows， Linux 和 ARM 平台的共享库交付的 C 客户机。

不均衡的连接

除颤器与医院之间的通信具有不均匀的网络特性。 用两个不同的网络来解决从病人那里收集数据，把数据送到医院的不同问题。 在专利与 MQTT 设备之间，使用短程低功耗 RF 网络。 发送器使用 VPN TCP/IP 连接通过低带宽电话线连接到医院。

通常无法找到将每个设备直接连接到 Internet Protocol 网络的方法。 使用两个网络 (通过中心连接) 是常见的解决方案。 MQTT 设备是一个简单的集线器，用于存储来自患者的信息，并将其转发到医院。

安全性

医生必须能够信任患者数据的真实性，患者希望其数据的隐私得到尊重。

在某些情况下，它足以使用 VPN 或 TLS 对连接进行加密。 在其他情况下，即使已存储数据，也应确保数据安全。

有时遥测设备不安全。 例如，它可能位于共享住宅中。 必须对设备的用户进行认证，以确保数据来自正确的患者。 可以使用 TLS 向服务器认证设备本身，并向设备认证服务器。

设备与队列管理器之间的遥测通道支持 JAAS 进行用户认证，支持 TLS 进行通信加密和设备认证。 对发布的访问由 IBM MQ 中的对象权限管理器控制。

用于认证用户的标识可以映射到其他标识，例如公共患者标识。 公共标识可简化对 IBM MQ 中发布主题的授权配置。

连接

MQTT 设备与医院之间的连接使用拨号，并使用低至 300 波特的带宽。

为了在 300 波特处有效运行，除了 TCP/IP 头外， MQTT protocol 仅向消息添加几个额外的字节。

MQTT protocol 提供单次传输 触发并忘记 消息传递，这使等待时间保持较低。 它还可以使用多个传输来保证 至少一次 和 正好一次 传递 (如果保证传递比响应时间更重要)。 为了保证传递，消息将存储在设备上，直到成功传递为止。 如果设备以无线方式连接，那么保证交付特别有用。

扩充性

遥测设备通常大量部署，从数万到数百万。

将许多设备连接到系统会对解决方案产生很大的需求。 存在业务需求，例如设备及其软件的成本，以及管理许可证，设备和用户的管理需求。 技术需求包括网络和服务器上的负载。

与维护打开的连接相比，打开连接使用的服务器资源更多。 但是，在使用电话线之类的用例中，连接费用意味着连接处于打开状态的时间不会超过所需时间。 数据传输在很大程度上具有批处理性质。 可以安排通宵达旦的连接，避免在睡前出现突然的连接高峰。

在客户机上，客户机的可伸缩性由所需的最低客户机配置提供帮助。 MQTT 客户机嵌入在设备中。 不需要在向患者部署设备时内置配置或 MQTT 客户机许可证接受步骤。

在服务器上， MQ Telemetry 的初始目标是每个队列管理器打开 50,000 个连接。

使用 IBM MQ Explorer 来管理连接。 IBM MQ Explorer 会将要显示的连接过滤为可管理的数字。 通过适当选择的将标识分配给客户机的方案，您可以根据地理位置或按患者名称的字母顺序来过滤连接。

Windows Linux AIX 遥测用例: 家庭能源监测与控制

与传统计量表相比，智能计量表收集更多有关能耗的详细信息。

智能电表通常与本地遥测网络耦合在一起，以监控家庭中的各个设备。 还可以远程连接某些智能电表，以在远处进行监控。

远程连接可由个人，电源实用程序或中央控制点设置。 远程控制点可以读取电源使用情况并提供使用情况数据。 它可以提供数据来影响使用情况，例如持续定价和天气信息。 可限制负荷以提高整体发电效率。

智能计量表已开始广泛部署。 例如，英国政府正在就到 2020 年将智能电表部署到每个英国家庭进行协商。

家庭计量用例有以下几个共同特点:

Invisibility

除非用户希望通过使用计量表来参与节能，否则计量表不得要求用户干预。 它不得降低个别电器的能源供应的可靠性。

MQTT 客户机可以嵌入到随计量表一起部署的软件中，并且不需要单独的安装或配置。

不均衡的连接

设备与智能计量表之间的通信要求的连接标准与计量表与远程连接点之间的连接标准不同。

从智能计量表到设备的连接必须高度可用，并且符合家庭区域网络的网络标准。

远程网络可能使用各种物理连接。 其中一些，如细胞，其传播成本很高，可以是间歇性的。 MQTT v3 规范针对远程连接以及本地适配器与智能计量表之间的连接。

电源插座和应用程序与计量表之间的连接使用家庭区域网络，例如 Zigbee。 MQTT 用于传感器网络 (MQTT-S) ，旨在使用 Zigbee 和其他低带宽网络协议。 MQ Telemetry 不直接支持 MQTT-S。 它需要网关将 MQTT-S 连接到 MQTT v3。

与家庭病人监测一样，家庭能源监测和控制的解决方案需要多个网络，使用智能仪表作为枢纽进行连接。

安全性

存在许多与智能计量表相关联的安全问题。 这些问题包括交易的不可抵赖性，启动的任何控制操作的授权以及功耗数据的隐私。

为确保隐私，可以使用 TLS 对 MQTT 在计量表和远程控制点之间传输的数据进行加密。 为确保控制操作的授权，可以使用 TLS 对计量表与远程控制点之间的 MQTT 连接进行相互认证。

连接

远程网络的物理性质可能有很大差异。 它可能使用现有宽带连接，或者使用具有高呼叫成本和间歇性可用性的移动网络。 对于高成本，间歇性的连接， MQTT 是一种高效且可靠的协议; 请参阅 第 92 页的『遥测用例: 家庭患者监控』。

扩充性

最终电力公司或中央控制点，计划部署数千万台智能电表。 最初，每个部署的计量表数在数万到数十万。 此数目与每个队列管理器 50,000 个开放式客户机连接的初始 MQTT 目标相当。

家庭能源监测和控制的架构的一个关键方面是使用智能仪表作为网络集中器。 每个设备适配器都是一个单独的传感器。 通过使用 MQTT 将它们连接到本地集线器，集线器可以将数据流集中到具有中央控制点的单个 TCP/IP 会话上，还可以在短时间内存储消息以克服会话中断。

在家庭能源用例中，远程连接必须保持打开状态，原因有二。 首先，由于打开连接相对于发送请求需要很长时间。 在较短的时间间隔内打开多个连接以发送 “装入限制” 请求的时间太长。 其次，要接收来自电力公司的负载限制请求，必须首先由客户打开连接。 使用 MQTT 时，连接始终由客户机启动，要接收来自电力公司的负载限制请求，连接必须保持打开状态。

如果打开连接的速率是临界值，或者服务器启动时间临界请求，那么解决方案通常是维护许多打开的连接。

Windows Linux AIX 遥测用例: 射频识别 (RFID)

RFID 是使用嵌入式 RFID 标签来无线识别和跟踪物体。 RFID 标签可以被读取到几米的范围内，并且脱离 RFID 阅读器的视线。 无源标签由 RFID 阅读器激活。 在没有外部激活的情况下传输活动标记。 活动标记必须具有电源。 无源标签可以包含电源以增加其范围。

RFID 在很多应用中都有使用，用例的类型也差别很大。 RFID 用例，以及家庭患者监测和家庭能源监测与控制用例，都有一些相似之处和不同之处。

Invisibility

在许多用例中， RFID 阅读器是大量部署的，必须在没有用户干预的情况下工作。 阅读器包含用于与中央控制点通信的嵌入式 MQTT 客户机。

例如，在配送仓库中，阅读器使用运动传感器来检测托盘。 它激活托盘上项目的 RFID 标签，并将数据和请求发送到中央应用程序。 数据用于更新库存位置。 这些请求可控制下一个托盘发生的情况，例如将其移动到特定托架。 航空公司和机场行李系统都在以这种方式使用 RFID。

在某些 RFID 用例中，阅读器具有标准计算环境，例如 Java Platform, Micro Edition (Java ME)。 在这些情况下， MQTT 客户机可能在制造后部署在不同的配置步骤中。

不均衡的连接

RFID 阅读器可能与包含 MQTT 客户机的本地控制设备分离，或者每个阅读器都可能嵌入 MQTT 客户机。 通常，地理或通信因素指示拓扑的选择。

安全性

隐私和真实性是 RFID 标签附件中的安全问题。 RFID 标签具有不可侵扰性，可被秘密监控，欺骗或篡改。

RFID 安全问题的解决方案增加了部署新的 RFID 解决方案的机会。 尽管安全漏洞在 RFID 标签和本地阅读器中，但使用中央信息处理建议了应对不同威胁的方法。 例如，可以通过将库存级别与交付和分派动态关联来检测标记篡改。

连接

RFID 应用通常涉及批量存储和从 RFID 阅读器收集的信息以及即时查询。 在配送仓库用例中， RFID 阅读器一直连接。 读取标记时，会将其与有关读者的信息一起发布。 仓储应用程序将响应发布回阅读器。

在仓储应用程序中，网络通常是可靠的，即时请求可能会使用 触发和忘记 消息以实现低延迟性能。 批处理存储和转发数据可能使用 精确一次 消息传递，以最大限度降低与松散数据相关联的管理成本。

扩充性

如果 RFID 应用程序需要立即响应 (按秒或秒的顺序) ，那么 RFID 阅读器必须保持连接。

Windows Linux AIX 遥测用例: 环境传感

环境传感利用遥测收集有关河流水位和质量，大气污染物以及其他环境数据的信息。

传感器经常位于偏远的地方，无法进行有线通信。 无线带宽昂贵，可靠性低。 通常，在一个小的地理区域内的一些环境传感器被连接到一个安全位置的本地监控设备。 本地连接可以是有线连接或无线连接。

Invisibility

与中央监控设备相比，这些传感器设备的可访问性可能较低，供电能力较低，并且部署的数量更多。 传感器有时是 "哑" 的，本地监控设备包括用于转换和存储传感器数据的适配器。 监视设备可能包含支持 Java Platform, Standard Edition (Java SE) 或 Java Platform, Micro Edition (Java ME) 的通用计算机。 配置 MQTT 客户机时，不太可能主要需要隐蔽性。

不均衡的连接

传感器的功能以及远程连接的成本和带宽通常会导致本地监控中心连接到中央服务器。

安全性

除非在军事或防御用例中使用该解决方案，否则安全不是主要要求。

连接

许多用途不需要持续监控或即时提供数据。 需要立即转发异常数据 (例如，洪水级别警报)。 在本地监视器上聚集传感器数据以降低连接和通信成本，然后使用预定连接进行传输。 一旦在监视器上检测到异常数据，就会将其转发。

扩充性

传感器集中在本地集线器周围，传感器数据聚集到根据调度传输的包中。 这两个因素都降低了中央服务器上通过使用直接连接的传感器施加的负载。

Windows Linux AIX 遥测用例: 移动应用程序

移动应用程序是在无线设备上运行的应用程序。 这些设备是通用应用程序平台或定制设备。

一般平台包括手机和个人数据助手等手持设备，笔记本电脑等便携式设备。 定制设备使用针对特定应用程序定制的特殊用途硬件。 用于记录 "signed-for" 包裹递送的设备是定制移动设备的示例。 定制移动设备上的应用程序通常是在通用软件平台上构建的。

Invisibility

定制移动应用程序的部署是受管的，并且可以包含 MQTT 客户机应用程序的配置。 配置 MQTT 客户机时，不太可能主要需要隐蔽性。

不均衡的连接

与先前用例的本地中心拓扑不同，移动式客户机远程连接。 客户机应用程序层直接连接到中央集线器上的应用程序。

安全性

在物理安全性很小的情况下，必须对移动设备和移动用户进行认证。 TLS 用于确认设备的身份， JAAS 用于认证用户。

连接

如果移动应用程序依赖于无线覆盖，那么它必须能够脱机操作，并且能够高效地处理中断的连接。 在此环境中，目标是保持连接，但应用程序必须能够存储和转发消息。 通常，消息是订单或交货确认，并且具有重要的业务价值。 需要对它们进行可靠的存储和转发。

扩充性

可扩展性不是一个主要问题。 在定制移动应用程序用例中，应用程序客户机的数量可能不会超过数千或数万个。

Windows Linux AIX 将遥测设备连接至队列管理器

遥测设备使用 MQTT v3 客户机连接到队列管理器。 MQTT v3 客户机使用 TCP/IP 连接到称为遥测 (MQXR) 服务的 TCP/IP 侦听器。

将遥测设备连接到队列管理器时， MQTT 客户机将使用 `MqttClient.connect` 方法启动 TCP/IP 连接。 与 IBM MQ 客户机一样， MQTT 客户机必须连接到队列管理器以发送和接收消息。 使用随 MQ Telemetry 一起

安装的 TCP/IP 侦听器 (称为遥测 (MQXR) 服务) 在服务器上建立连接。 每个队列管理器最多运行一个遥测 (MQXR) 服务。

遥测 (MQXR) 服务使用每个客户机在 `MqttClient.connect` 方法中设置的远程套接字地址来分配与遥测通道的连接。 套接字地址是 TCP/IP 主机名和端口号的组合。 使用同一远程套接字地址的多个客户机通过遥测 (MQXR) 服务连接到同一遥测通道。

如果服务器上有多个队列管理器，请在队列管理器之间拆分遥测通道。 在队列管理器之间分配远程套接字地址。 使用唯一的远程套接字地址定义每个遥测通道。 两个遥测通道不得使用相同的套接字地址。

如果为多个队列管理器上的遥测通道配置了相同的远程套接字地址，那么要连接的第一个遥测通道将获胜。 在同一地址上连接的后续通道失败。

如果服务器上有多个网络适配器，请在遥测通道之间拆分远程套接字地址。 只要仅在一个遥测通道上配置任何特定套接字地址，那么套接字地址的分配完全是任意的。

使用 IBM MQ Explorer 的 MQ Telemetry 补充说明中提供的向导配置 IBM MQ 以连接 MQTT 客户机。 或者，遵循 在 Linux 和 AIX 上配置队列管理器以进行遥测 和 在 Windows 上配置队列管理器以进行遥测手动配置 中的指示信息。

相关参考

MQXR 属性

Windows Linux AIX 遥测连接协议

MQ Telemetry 支持 TCP/IP IPv4 和 IPv6 以及 TLS。

Windows Linux AIX 遥测 (MQXR) 服务

遥测 (MQXR) 服务是作为 IBM MQ 服务管理的 TCP/IP 侦听器。 使用 IBM MQ Explorer 向导或 **runmqsc** 命令创建服务。

MQ Telemetry (MQXR) 服务称为 SYSTEM.MQXR.SERVICE 。

IBM MQ Explorer 的 MQ Telemetry 函数中提供的 Telemetry 样本配置向导将创建遥测服务和样本遥测通道; 请参阅 使用 IBM MQ Explorer 验证 MQ Telemetry 的安装 。

从命令行创建样本配置; 请参阅 使用命令行验证 MQ Telemetry 的安装。

遥测 (MQXR) 服务会随队列管理器自动启动和停止。 使用 IBM MQ Explorer 中的 services 文件夹来控制服务。 要查看服务，必须单击该图标以停止 IBM MQ Explorer 从显示中过滤掉 SYSTEM 对象。

有关如何手动创建服务的示例，请参阅

- Linux AIX 在 Linux 上创建 `SYSTEM.MQXR.SERVICE`.
- Windows 在 Windows 上创建 `SYSTEM.MQXR.SERVICE`.

V 9.3.0 从 IBM MQ 9.3.0 开始， 在 Linux 上创建 `SYSTEM.MQXR.SERVICE` 和 在 Windows 上创建 `SYSTEM.MQXR.SERVICE` 将更新为指定缺省密钥，以要求对 MQTT TLS 通道的口令进行加密。 有关更多信息，请参阅 加密 MQTT TLS 通道的口令。

Windows Linux AIX 遥测通道

创建遥测通道以创建具有不同属性 (例如 Java 认证和授权服务 (JAAS) 或 TLS 认证) 的连接，或者管理客户机组。

使用 IBM MQ Explorer 的 MQ Telemetry 函数中提供的 **New Telemetry Channel** 向导创建遥测通道。 使用向导配置通道以接受来自特定 TCP/IP 端口上的 MQTT 客户机的连接。 从 IBM WebSphere MQ 7.1 开始，您可以使用命令行程序 **runmqsc** 来配置 MQ Telemetry 。

在不同端口上创建多个遥测通道，通过将客户机拆分为多个组，使大量客户机连接更易于管理。 每个遥测通道都有不同的名称。

您可以配置具有不同安全性属性的遥测通道，以创建不同类型的连接。 创建多个通道以接受不同 TCP/IP 地址上的客户机连接。 使用 TLS 对消息进行加密并认证遥测通道和客户机; 请参阅 MQTT 客户机和遥测通道的 TLS 配置。 指定用户标识以简化对 IBM MQ 对象的授权访问。 指定 JAAS 配置以使用 JAAS 认证 MQTT 用户; 请参阅 MQTT 客户机标识，授权和认证。

Windows Linux AIX IBM MQ Telemetry Transport 协议

IBM MQ Telemetry Transport (MQTT) v3 协议旨在以低带宽或昂贵的连接在小型设备之间交换消息，并可靠地发送消息。 它使用 TCP/IP。

MQTT protocol 已发布; 请参阅 IBM MQ Telemetry Transport 格式和协议。 协议版本 3 使用发布/预订，并支持三种服务质量: 触发和忘记， 至少一次和 正好一次。

协议头和字节数组消息有效内容的较小大小使消息保持较小。 这些头包含一个 2 字节的固定头，以及最多 12 个字节的其他变量头。 该协议使用 12 字节变量头来预订和连接，并且仅 2 字节变量头用于大多数发布。

通过三种服务质量，您可以在低延迟和可靠性之间进行权宜之法; 请参阅 MQTT 客户机提供的服务的程度。 触发和忘记 不使用持久设备存储器，仅使用一个传输来发送或接收发布。 至少一次，并且 正好一次 需要设备上的持久存储器以保持协议状态并保存消息，直到确认为止。

Windows Linux AIX MQTT Client

MQTT 客户机应用程序负责从遥测设备收集信息，连接到服务器以及将信息发布到服务器。 它还可以预订主题，接收出版物以及控制遥测设备。

与 IBM MQ 客户机应用程序不同， MQTT 客户机应用程序不是 IBM MQ 应用程序。 它们不指定要连接到的队列管理器。 它们不限于使用特定的 IBM MQ 编程接口。 相反， MQTT 客户机实现 MQTT 3 协议。 您可以编写自己的客户机库，以便在您选择的编程语言和平台上与 MQTT protocol 进行交互。 请参阅 IBM MQ Telemetry Transport 格式和协议。

要简化 MQTT 客户机应用程序的编写，请使用封装多个平台的 MQTT protocol 的 C， Java 和 JavaScript 客户机库。 如果将这些库合并到 MQTT 应用程序中，那么功能齐全的 MQTT 客户机可以短于 15 行代码。 MQTT 客户机库可从 Eclipse Paho 和 MQTT.org 免费获取。 请参阅 IBM MQ Telemetry Transport 样本程序。

MQTT 客户机应用程序始终负责启动与遥测通道的连接。 连接后， MQTT 客户机应用程序或 IBM MQ 应用程序可以启动消息交换。

MQTT 客户机应用程序和 IBM MQ 应用程序发布并预订同一组主题。 IBM MQ 应用程序还可以直接向 MQTT 客户机应用程序发送消息，而无需客户机应用程序首先创建预订。 请参阅 配置分布式排队以将消息发送到 MQTT 客户机。

MQTT 客户机应用程序使用遥测通道连接到 IBM MQ 。 遥测通道充当 MQTT 和 IBM MQ 所使用的不同类型消息之间的网桥。 它代表 MQTT 客户机应用程序在队列管理器中创建发布和预订。 遥测通道将与 MQTT 客户机应用程序预订匹配的发布从队列管理器发送到 MQTT 客户机应用程序。

Windows Linux AIX 向 MQTT 客户机发送消息

IBM MQ 应用程序可以通过发布到客户机创建的预订或通过直接发送消息来发送 MQTT v3 客户机消息。 MQTT 客户机可以通过发布到其他客户机预订的主题来相互发送消息。

MQTT 客户机预订从 IBM MQ 接收的发布内容

执行任务 第 99 页的『从 IBM MQ Explorer 将消息发布到 MQTT 客户机实用程序』 以将发布从 IBM MQ 发送到 MQTT 客户机。

MQTT v3 客户机接收消息的标准方法是它创建对主题或一组主题的预订。 在示例代码片段 第 98 页的图 44 中， MQTT 客户机使用主题字符串 “MQTT Examples”进行预订。 IBM MQ C 应用程序 第 98 页的图 45 使用主题字符串 “MQTT Examples”发布到主题。 在代码片段 第 99 页的图 46 中， MQTT 客户机接收回调方法 `messageArrived` 中的发布内容。

有关如何配置 IBM MQ 以发送发布以响应来自 MQTT 客户机的预订的更多信息，请参阅 发布消息以响应 MQTT 客户机预订。

IBM MQ 应用程序将消息直接发送到 MQTT 客户机

执行任务 第 103 页的『使用 IBM MQ Explorer 向 MQTT 客户机发送消息』 以将消息直接从 IBM MQ 发送到 MQTT 客户机。

以这种方式发送到 MQTT 客户机的消息称为非请求消息。 MQTT v3 客户机作为主题名称集的发布接收自发消息。 遥测 (MQXR) 服务将主题名称设置为远程队列名称。

有关如何配置 IBM MQ 以将消息直接发送到 MQTT 客户机的更多信息，请参阅 直接将消息发送到客户机。

MQTT 客户机发布消息

MQTT v3 客户机可以发布另一个 MQTT v3 客户机接收到的消息，但它无法发送未经请求的消息。 代码片段 第 99 页的图 47 显示了以 Java 编写的 MQTT v3 客户机如何发布消息。

用于将消息发送到一个特定 MQTT v3 客户机的典型模式是每个客户机创建对其自己的 `ClientIdentifier` 的预订。 执行任务 第 104 页的『将消息发布到特定 MQTT v3 客户机』 以使用 `ClientIdentifier` 作为主题字符串将消息从一个 MQTT 客户机发布到另一个 MQTT 客户机。

示例代码片段

第 98 页的图 44 中的代码片段显示了在 Java 中编写的 MQTT 客户机如何创建预订。 它还需要回调方法 `messageArrived` 来接收预订的发布。

```
String    clientId = String.format("%-23.23s",
                     System.getProperty("user.name") + "_" +
                     (UUID.randomUUID().toString())).trim()).replace('-', '_');
MqttClient  client = new MqttClient("localhost", clientId);
String topicString = "MQTT Examples";
int            QoS = 1;
client.subscribe(topicString, QoS);
```

图 *44: MQTT v3* 客户机订户

第 98 页的图 45 中的代码片段显示了以 C 编写的 IBM MQ 应用程序如何发送发布内容。 将从以下任务中抽取代码片段: Create a publisher to a variable topic

```
/* Define and set variables to.defaults */
/* Omitted lines declaring variables    */
char * topicName   = ""
char * topicString = "MQTT Examples"
char * publication = "Hello world!";
do {
  MQCONN(qMgrName, &Hconn, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  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);
  MQOPEN(Hconn, &td, MQOO_OUTPUT | MQOO_FAIL_IF_QUIESCING, &Hobj, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  pmo.Options = MQPMO_FAIL_IF_QUIESCING | MQPMO_RETAIN;
  MQPUT(Hconn, Hobj, &md, &pmo, (MQLONG)strlen(publication)+1, publication, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  MQCLOSE(Hconn, &Hobj, MQCO_NONE, &CompCode, &Reason);
  if (CompCode != MQCC_OK) break;
  MQDISC(&Hconn, &CompCode, &Reason);
} while (0);
```

图 *45: IBM MQ* 发布者

当发布到达时， MQTT 客户机将调用 MQTT application client `MqttCallback` 类的 `messageArrived` 方法。

```
public class CallBack implements MqttCallback {
  public void messageArrived(MqttTopic topic, MqttMessage message) {
    try {
      System.out.println("Message arrived: \"" + message.toString()
          + "\" on topic \"" + topic.toString() + "\"");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
// ... Other callback methods
}
```

图 *46: messageArrived 方法*

第 99 页的图 47 显示了 MQTT v3 将消息发布到 第 98 页的图 44 中创建的预订。

```
      String          address = "localhost";
      String         clientId = String.format("%-23.23s",
                                System.getProperty("user.name") + "_" +
                                (UUID.randomUUID().toString())).trim()).replace('-', '_');
      MqttClient       client = new MqttClient(address, clientId);
      String      topicString = "MQTT Examples";
      MqttTopic         topic = client.getTopic(Example.topicString);
      String     publication  = "Hello world";
      MqttMessage     message = new MqttMessage(publication.getBytes());
      MqttDeliveryToken token = topic.publish(message);
```

图 *47: MQTT v3 客户机发布程序*

Windows Linux AIX 从 IBM MQ Explorer 将消息发布到 MQTT 客户机实用程序

执行本任务中的步骤以使用 IBM MQ Explorer 发布消息，并使用 MQTT 客户机实用程序预订该消息。 其他任务显示如何配置队列管理器别名，而不是将缺省传输队列设置为 `SYSTEM.MQTT.TRANSMIT.QUEUE`。

开始之前

该任务假定您熟悉 IBM MQ 和 IBM MQ Explorer，并且已安装 IBM MQ 和 MQ Telemetry 功能部件。

为此任务创建队列管理器资源的用户必须具有足够的权限才能执行此操作。 出于演示目的，假定 IBM MQ Explorer 用户标识是 `mqm` 组的成员。

关于此任务

在该任务中，您将在 IBM MQ 中创建主题，并使用 MQTT 客户机实用程序预订该主题。 使用 IBM MQ Explorer 发布到主题时， MQTT 客户机将接收到该发布。

过程

请执行下列其中一项任务：

- 您已安装 MQ Telemetry，但尚未将其启动。 执行任务: 第 100 页的『在尚未定义遥测 (MQXR) 服务的情况下启动任务』。
- 您之前已运行 IBM MQ 遥测，但希望使用新的队列管理器来执行演示。 执行任务: 第 100 页的『在尚未定义遥测 (MQXR) 服务的情况下启动任务』。
- 您希望使用未定义遥测资源的现有队列管理器来执行该任务。 您不希望运行 " **定义样本配置** " 向导。

  a. 执行下列其中一项任务以设置遥测：

  – 在 Linux 和 AIX 上配置队列管理器以进行遥测

– 在 Windows 上配置队列管理器以进行遥测

b. 执行任务: 第 101 页的『使用正在运行的遥测 (MQXR) 服务启动任务』

- 如果要使用已定义遥测资源的现有队列管理器来执行任务，请执行以下任务: 第 101 页的『使用正在运行的遥测 (MQXR) 服务启动任务』。

下一步做什么

执行 第 103 页的『使用 IBM MQ Explorer 向 MQTT 客户机发送消息』 以将消息直接发送到客户机实用程序。

在尚未定义遥测 *(MQXR)* 服务的情况下启动任务

创建队列管理器并运行 **定义样本配置** 以定义队列管理器的样本遥测资源。 使用 IBM MQ Explorer 发布消息，并使用 MQTT 客户机实用程序预订该消息。

关于此任务

使用 **定义样本配置**设置样本遥测资源时，向导会设置访客用户标识许可权。 请仔细考虑是否希望以这种方式对访客用户标识进行授权。 guest on Windows 和 nobody on Linux 被授予发布和预订主题树的根以及将消息放入 SYSTEM.MQTT.TRANSMIT.QUEUE 的许可权。

向导还将缺省传输队列设置为 SYSTEM.MQTT.TRANSMIT.QUEUE，这可能会干扰在现有队列管理器上运行的应用程序。 可以但费力地配置遥测，而不使用缺省传输队列; 请执行以下任务: 第 102 页的『使用队列管理器别名』。 在此任务中，创建队列管理器以避免干扰任何现有缺省传输队列的可能性。

过程

1. 使用 IBM MQ Explorer 创建并启动新的队列管理器。
   a) 右键单击 Queue Managers 文件夹> **新建 > 队列管理器 ...**。 输入队列管理器名称> **完成**。

   组成队列管理器名称; 例如， MQTTQMGR。
2. 创建并启动遥测 (MQXR) 服务，并创建样本遥测通道。
   a) 打开 Queue Managers*QmgrName*\Telemetry 文件夹。
   b) 单击 **定义样本配置 ... > 完成**

   保持选中 **启动 MQTT 客户机实用程序** 复选框。
3. 使用 MQTT 客户机实用程序为 MQTT Example 创建预订。
   a) 单击**连接**。

   客户机历史记录 记录 Connected 事件。
   b) 在 **预订 \ 主题** 字段> **预订**中输入 MQTT Example 。

   客户机历史记录 记录 Subscribed 事件。
4. 在 IBM MQ 中创建 MQTTExampleTopic 。
   a) 右键单击 **MQ Explorer**> **新建** > **主题**中的 Queue Managers*QmgrName*\Topics 文件夹。
   b) 输入 MQTTExampleTopic 作为 **名称** > **下一步**。
   c) 输入 MQTT Example 作为 **主题字符串** > **完成**。
   d) 单击 **确定** 以关闭确认窗口。
5. 使用 IBM MQ Explorer 将 Hello World! 发布到主题 MQTT Example 。
   a) 单击 IBM MQ Explorer 中的 Queue Managers*QmgrName*\Topics 文件夹。
   b) 右键单击 MQTTExampleTopic > **测试发布 ...**
   c) 在 **消息数据** 字段> **发布消息** > 切换到 " MQTT 客户机实用程序" 窗口中输入 Hello World! 。

   客户机历史记录 记录 Received 事件。

使用正在运行的遥测 (MQXR) 服务启动任务

创建遥测通道和主题。 授权用户使用主题和遥测传输队列。 使用 IBM MQ Explorer 发布消息，并使用 MQTT 客户机实用程序预订该消息。

开始之前

在此版本的任务中，定义并运行了队列管理器 *QmgrName*。 定义并运行遥测 (MQXR) 服务。 遥测 (MQXR) 服务可能是手动创建的，也可能是通过运行 " **定义样本配置** " 向导创建的。

关于此任务

在此任务中，您将配置现有队列管理器以将发布发送到 MQTT 客户机实用程序。

任务的步骤 第 101 页的『1』 将缺省传输队列设置为 SYSTEM.MQTT.TRANSMIT.QUEUE，这可能会干扰在现有队列管理器上运行的应用程序。 可以但费力地配置遥测，而不使用缺省传输队列; 请执行以下任务: 第 102 页的『使用队列管理器别名』。

过程

1. 将 SYSTEM.MQTT.TRANSMIT.QUEUE 设置为缺省传输队列。
   a) 右键单击 Queue Managers*QmgrName* folder > **属性 ...**
   b) 在导航器中单击 **通信** 。
   c) 单击 **选择 ...** > 选择 SYSTEM.MQTT.TRANSMIT.QUEUE > **确定** > **确定**。
2. 创建遥测通道 MQTTExampleChannel 以将 MQTT 客户机实用程序连接到 IBM MQ，并启动 MQTT 客户机实用程序。
   a) 右键单击 **MQ Explorer**> **新建** > **遥测通道 ...**中的 Queue Managers*QmgrName* \Telemetry\Channels 文件夹。
   b) 在 **通道名称** 字段中输入 MQTTExampleChannel > **Next** > **Next**。
   c) 将客户机授权面板上的 **固定用户标识** 更改为要发布和预订 MQTTExample > **下一步**的用户标识。
   d) 保持选中 **启动客户机实用程序** > **完成**。
3. 使用 MQTT 客户机实用程序为 MQTT Example 创建预订。
   a) 单击**连接**。

      客户机历史记录 记录 Connected 事件。
   b) 在 **预订 \ 主题** 字段> **预订**中输入 MQTT Example 。

   客户机历史记录 记录 Subscribed 事件。
4. 在 IBM MQ 中创建 MQTTExampleTopic 。
   a) 右键单击 **MQ Explorer**> **新建** > **主题**中的 Queue Managers*QmgrName*\Topics 文件夹。
   b) 输入 MQTTExampleTopic 作为 **名称** > **下一步**。
   c) 输入 MQTT Example 作为 **主题字符串** > **完成**。
   d) 单击 **确定** 以关闭确认窗口。
5. 如果您希望用户 (不在 mqm 组中) 发布和预订 MQTTExample 主题，请执行以下操作:
   a) 授权用户发布和预订主题 MQTTExampleTopic:

      ```
      setmqaut -m qMgrName -t topic -n MQTTExampleTopic -p User ID -all +pub +sub
      ```

   b) 授权用户将消息放入 SYSTEM.MQTT.TRANSMIT.QUEUE:

      ```
      setmqaut -m qMgrName -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p User ID -all +put
      ```

6. 使用 IBM MQ Explorer 将 Hello World! 发布到主题 MQTT Example 。
   a) 单击 IBM MQ Explorer 中的 Queue Managers*QmgrName*\Topics 文件夹。
   b) 右键单击 MQTTExampleTopic > **测试发布 ...**

c) 在 **消息数据** 字段> **发布消息** > 切换到 " MQTT 客户机实用程序" 窗口中输入 Hello World! 。

客户机历史记录 记录 Received 事件。

使用队列管理器别名

使用 IBM MQ Explorer 将消息发布到 MQTT 客户机实用程序，而不将缺省传输队列设置为 SYSTEM.MQTT.TRANSMIT.QUEUE。

该任务是先前任务的延续，并使用队列管理器别名来避免将缺省传输队列设置为 SYSTEM.MQTT.TRANSMIT.QUEUE。

开始之前

完成任务 第 100 页的『在尚未定义遥测 (MQXR) 服务的情况下启动任务』 或任务 第 101 页的『使用正在运行的遥测 (MQXR) 服务启动任务』。

关于此任务

当 "MQTT 客户端创建订阅时，"IBM MQ 会使用 ClientIdentifier 作为远程队列管理器名称发送响应。在这项任务中，它使用了**客户标识符**"MyClient。

如果没有名为 MyClient 的传输队列或队列管理器别名，那么响应将放在缺省传输队列上。 通过将缺省传输队列设置为 SYSTEM.MQTT.TRANSMIT.QUEUE， MQTT 客户机将获取响应。

您可以避免使用队列管理器别名将缺省传输队列设置为 SYSTEM.MQTT.TRANSMIT.QUEUE 。 必须为每个 ClientIdentifier 设置队列管理器别名。 通常，客户机过多，以致无法实际使用队列管理器别名。 通常， ClientIdentifier 不可预测，因此无法以此方式配置遥测。

但是，在某些情况下，您可能必须将缺省传输队列配置为 SYSTEM.MQTT.TRANSMIT.QUEUE 以外的传输队列。 过程 中的步骤配置队列管理器别名，而不是将缺省传输队列设置为 SYSTEM.MQTT.TRANSMIT.QUEUE。

过程

1. 除去 SYSTEM.MQTT.TRANSMIT.QUEUE 作为缺省传输队列。
   a) 右键单击 Queue Managers*QmgrName* folder > **属性 ...**
   b) 在导航器中单击 **通信** 。
   c) 从 **缺省传输队列** 字段中除去 SYSTEM.MQTT.TRANSMIT.QUEUE > **确定**。
2. 检查您是否无法再使用 MQTT 客户机实用程序创建预订:
   a) 单击**连接**。

      客户机历史记录 记录 Connected 事件。
   b) 在 **预订 \ 主题** 字段> **预订**中输入 MQTT Example 。

   客户机历史记录 记录 Subscribe failed 和 Connection lost 事件。
3. 为**客户标识符**"MyClient 创建队列管理器别名。
   a) 右键单击 Queue Managers*QmgrName*\Queues 文件夹> **新建** > **远程队列定义**。
   b) 将定义命名为 MyClient > **Next**。
   c) 在 **远程队列管理器** 字段中输入 MyClient 。
   d) 在 **传输队列** 字段> **完成**中输入 SYSTEM.MQTT.TRANSMIT.QUEUE 。
4. 再次连接 MQTT 客户机实用程序。
   a) 检查 **客户机标识** 是否设置为 MyClient。
   b) **CONNECT**

   客户机历史记录 记录 Connected 事件。
5. 使用 MQTT 客户机实用程序为 MQTT Example 创建预订。

a) 单击**连接**。

客户机历史记录 记录 Connected 事件。

b) 在 **预订 \ 主题** 字段> **预订**中输入 MQTT Example 。

客户机历史记录 记录 Subscribed 事件。

6. 使用 IBM MQ Explorer 将 Hello World! 发布到主题 MQTT Example 。

a) 单击 IBM MQ Explorer 中的 Queue Managers*QmgrName*\Topics 文件夹。

b) 右键单击 MQTTExampleTopic > **测试发布 ...**

c) 在 **消息数据** 字段> **发布消息** > 切换到 " MQTT 客户机实用程序" 窗口中输入 Hello World! 。

客户机历史记录 记录 Received 事件。

Windows Linux AIX 使用 IBM MQ Explorer 向 MQTT 客户机发送消息

通过使用 IBM MQ Explorer 将消息放入 IBM MQ 队列，将消息发送到 MQTT 客户机实用程序。 此任务说明如何配置远程队列定义以将消息直接发送到 MQTT 客户机。

开始之前

执行任务 第 99 页的『从 IBM MQ Explorer 将消息发布到 MQTT 客户机实用程序』。 使 MQTT 客户机实用程序保持连接。

关于此任务

此任务演示如何使用队列将消息发送到 MQTT 客户机，而不是发布到主题。 您不会在客户机中创建预订。 该任务的步骤 第 103 页的『2』 演示已删除先前的预订。

过程

1. 通过断开连接并重新连接 MQTT 客户机实用程序来废弃任何现有预订。

将废弃预订，因为除非您更改缺省值，否则 MQTT 客户机实用程序将与清除会话连接; 请参阅 清除会话。

为了更容易执行该任务，请输入您自己的 ClientIdentifier，而不是使用由 MQTT 客户机实用程序创建的生成的 ClientIdentifier 。

a) 单击 **断开连接** 以断开 MQTT 客户机实用程序与遥测通道的连接。

客户机历史记录 记录 Disconnected 事件

b) 将 **客户机标识** 更改为 MyClient。

c) 单击**连接**。

客户机历史记录 记录 Connected 事件

2. 检查 MQTT 客户机实用程序是否不再接收 MQTTExampleTopic 的发布。

a) 单击 IBM MQ Explorer 中的 Queue Managers*QmgrName*\Topics 文件夹。

b) 右键单击 MQTTExampleTopic > **测试发布 ...**

c) 在 **消息数据** 字段> **发布消息** > 切换到 " MQTT 客户机实用程序" 窗口中输入 Hello World! 。

客户机历史记录中未记录任何事件。

3. 为客户机创建远程队列定义。

将 ClientIdentifier MyClient 设置为远程队列定义中的远程队列管理器名称。 使用您喜欢的任何名称作为远程队列名称。 远程队列名称将作为主题名称传递到 MQTT 客户机。

a) 右键单击 Queue Managers*QmgrName*\Queues 文件夹> **新建** > **远程队列定义**。

b) 将定义命名为 MyClientRemoteQueue > **Next**。

c) 在 **远程队列** 字段中输入 MQTTExampleQueue 。

d) 在 **远程队列管理器** 字段中输入 MyClient 。

e) 在 **传输队列** 字段> **完成**中输入 SYSTEM.MQTT.TRANSMIT.QUEUE 。

4. 将测试消息放在 MyClientRemoteQueue 上。

a) 右键单击 **MyClientRemoteQueue** > **放置测试消息 ...**

b) 在 "消息数据" 字段中输入 Hello queue! > **放入消息** > **关闭**

客户机历史记录 记录 Received 事件。

5. 除去 SYSTEM.MQTT.TRANSMIT.QUEUE 作为缺省传输队列。

a) 右键单击 Queue Managers*QmgrName* folder > **属性 ...**

b) 在导航器中单击 **通信** 。

c) 从 **缺省传输队列** 字段中除去 SYSTEM.MQTT.TRANSMIT.QUEUE > **确定**。

6. 重做步骤 第 104 页的『4』。

MyClientRemoteQueue 是明确指定传输队列的远程队列定义。 您不需要定义缺省传输队列以将消息发送到 MyClient。

下一步做什么

如果缺省传输队列不再设置为 SYSTEM.MQTT.TRANSMIT.QUEUE，那么除非为 ClientIdentifier MyClient 定义了队列管理器别名，否则 MQTT 客户机实用程序无法创建新的预订。 将缺省传输队列复原到 SYSTEM.MQTT.TRANSMIT.QUEUE。

Windows Linux AIX 将消息发布到特定 MQTT v3 客户机

将消息从一个 MQTT v3 客户机发布到另一个客户机，将 ClientIdentifier 用作主题名称，将 IBM MQ 用作发布/预订代理。

开始之前

执行任务 第 99 页的『从 IBM MQ Explorer 将消息发布到 MQTT 客户机实用程序』。 使 MQTT 客户机实用程序保持连接。

关于此任务

该任务演示了两件事:

1. 在一个 MQTT 客户机中预订主题，并从另一个 MQTT 客户机接收发布内容。
2. 通过使用 ClientIdentifier 作为主题字符串来设置 "点到点" 预订。

过程

1. 通过断开连接并重新连接 MQTT 客户机实用程序来废弃任何现有预订。

将废弃预订，因为除非您更改缺省值，否则 MQTT 客户机实用程序将与清除会话连接; 请参阅 清除会话。

为了更容易执行该任务，请输入您自己的 ClientIdentifier，而不是使用由 MQTT 客户机实用程序创建的生成的 ClientIdentifier 。

a) 单击 **断开连接** 以断开 MQTT 客户机实用程序与遥测通道的连接。

客户机历史记录 记录 Disconnected 事件

b) 将 **客户机标识** 更改为 MyClient。

c) 单击**连接**。

客户机历史记录 记录 Connected 事件

2. 创建主题 MyClient 的预订

MyClient 是此客户机的 ClientIdentifier 。

a) 在 **预订 \ 主题** 字段> **预订**中输入 MyClient 。

客户机历史记录 记录 Subscribed 事件。

3. 启动另一个 MQTT 客户机实用程序。
   a) 打开 Queue Managers*QmgrName*\Telemetry\channels 文件夹。
   b) 右键单击 **PlainText** 通道> **运行 MQTT 客户机实用程序 ...**
   c) 单击**连接**。

   客户机历史记录 记录 Connected 事件
4. 将 Hello MyClient! 发布到主题 MyClient。
   a) 从使用 ClientIdentifier MyClient 运行的 MQTT 客户机实用程序复制预订主题 MyClient。
   b) 将 MyClient 粘贴到每个 MQTT 客户机实用程序实例的 **发布 \ 主题** 字段中。
   c) 在 **Publication \ message** 字段中输入 Hello MyClient! 。
   d) 在这两个实例中单击 **发布** 。

结果

具有 ClientIdentifier MyClient 的 MQTT 客户机实用程序中的 **客户机历史记录** 记录了两个 **已接收** 事件和一个 **已发布** 事件。 另一个 MQTT 客户机实用程序实例记录一个 **已发布** 事件。

如果仅看到一个 **已接收** 事件，请检查以下可能的原因:

1. 队列管理器的缺省传输队列是否设置为 SYSTEM.MQTT.TRANSMIT.QUEUE ?
2. 是否在执行其他练习时创建了引用 MyClient 的队列管理器别名或远程队列定义? 如果存在配置问题，请删除引用 MyClient 的任何资源，例如队列管理器别名或传输队列。 断开客户机实用程序的连接，停止并重新启动遥测 (MQXR) 服务。

Windows Linux AIX 从 MQTT 客户机向 IBM MQ 应用程序发送消息

IBM MQ 应用程序可以通过预订主题从 MQTT v3 客户机接收消息。 MQTT 客户机使用遥测通道连接到 IBM MQ ，并通过发布到同一主题将消息发送到 IBM MQ 应用程序。

执行任务 第 105 页的『从 MQTT 客户机将消息发布到 IBM MQ』，以了解如何将发布从 MQTT 客户机发送到 IBM MQ 中定义的预订。

如果主题是集群主题，或者使用发布/预订层次结构进行分发，那么预订可以位于与 MQTT 客户机所连接的队列管理器不同的队列管理器上。

Windows Linux AIX 从 MQTT 客户机将消息发布到 IBM MQ

使用 IBM MQ Explorer 创建主题预订，并使用 MQTT 客户机实用程序发布到主题。

开始之前

执行任务 第 99 页的『从 IBM MQ Explorer 将消息发布到 MQTT 客户机实用程序』。 使 MQTT 客户机实用程序保持连接。

关于此任务

此任务演示使用 MQTT 客户机发布消息，并使用使用 IBM MQ Explorer 创建的非受管持久预订接收发布。

过程

1. 创建对主题字符串 MQTT Example 的持久预订。
   执行以下步骤以使用 IBM MQ Explorer 创建队列和预订。
   a) 右键单击 IBM MQ Explorer> **新建** > **本地队列 ...**中的 Queue Managers*QmgrName*\Queues 文件夹。
   b) 输入 MQTTExampleQueue 作为队列名称> **完成**。

c) 右键单击 IBM MQ Explorer> **新建** > **预订 ...**中的 Queue Managers*QmgrName*\Subscriptions 文件夹。

d) 输入 MQTTExampleSubscription 作为队列名称> **Next**。

e) 单击 **选择 ...** > MQTTExampleTopic > **确定**。

您已在 第 99 页的『从 IBM MQ Explorer 将消息发布到 MQTT 客户机实用程序』的步骤 第 100 页的『4』 中创建主题 MQTTExampleTopic 。

f) 输入 MQTTExampleQueue 作为目标名称> **完成**。

2. 作为可选步骤，在没有 mqm 权限的情况下设置队列以供其他用户使用。

如果要为权限低于 mqm 的用户设置配置，那么必须将 put 和 get 权限授予 MQTTExampleQueue。 在 第 99 页的『从 IBM MQ Explorer 将消息发布到 MQTT 客户机实用程序』中配置了对主题和传输队列的访问权。

a) 授权用户放入并进入队列 MQTTExampleQueue:

```
setmqaut -m qMgrName -t queue -n MQTTExampleQueue -p User ID -all +put +get
```

3. 使用 MQTT 客户机实用程序将 Hello IBM MQ! 发布到主题 MQTT Example 。

如果尚未使 MQTT 客户机实用程序保持连接，请右键单击 **PlainText** 通道> **运行 MQTT 客户机实用程序 ...** > **连接**。

a) 在 **发布 \ 主题** 字段中输入 MQTT Example 。

b) 在 **发布 \ 消息** 字段> **发布**中输入 Hello IBM MQ! 。

4. 打开 Queue Managers*QmgrName*\Queues 文件夹并查找 MQTTExampleQueue。

当前队列深度 字段为 1

5. 右键单击 MQTTExampleQueue > **浏览消息 ...** 并检查该出版物。

Windows Linux AIX MQTT 发布/预订应用程序

使用基于主题的发布/预订来编写 MQTT 应用程序。

当 MQTT 客户机连接时，发布在客户机与服务器之间的任一方向上流动。 在客户机上发布信息时，将从客户机发送这些发布。 当将消息发布到与客户机创建的预订匹配的主题时，将在客户机上接收到发布内容。

IBM MQ 发布/预订代理程序管理 MQTT 客户机创建的主题和预订。 MQTT 客户机创建的主题与 IBM MQ 应用程序创建的主题共享相同的主题空间。

将与 MQTT 客户机预订中的主题字符串匹配的发布放在 SYSTEM.MQTT.TRANSMIT.QUEUE 上，并将远程队列管理器名称设置为客户机的 ClientIdentifier 。 遥测 (MQXR) 服务将发布转发到创建预订的客户机。 它使用已设置为远程队列管理器名称的 ClientIdentifier 来标识客户机。

通常，必须将 SYSTEM.MQTT.TRANSMIT.QUEUE 定义为缺省传输队列。 可以将 MQTT 配置为不使用缺省传输队列，但这很繁重; 请参阅 配置分布式排队以将消息发送到 MQTT 客户机。

MQTT 客户机可以创建持久会话; 请参阅 第 109 页的『MQTT 无状态和有状态会话』。 在持久会话中创建的预订是持久的。 针对具有持久会话的客户机的发布将存储在 SYSTEM.MQTT.TRANSMIT.QUEUE 中，并在客户机重新连接时转发到该客户机。

MQTT 客户机还可以发布和预订保留的发布; 请参阅 保留的发布和 MQTT 客户机。 保留发布主题的订户接收到该主题的最新发布。 订户在创建预订时或在重新连接到其先前会话时接收保留的发布内容。

Windows Linux AIX 遥测应用程序

使用 IBM MQ 或 IBM Integration Bus 消息流编写遥测应用程序。

使用 JMS， MQI 或其他 IBM MQ 编程接口来编程 IBM MQ 中的遥测应用程序。

遥测 (MQXR) 服务在 MQTT v3 消息和 IBM MQ 消息之间进行转换。 它代表 MQTT 客户机创建预订和发布，并将发布转发到 MQTT 客户机。 发布是 MQTT v3 消息的有效内容。 有效内容包含消息头和 jms-bytes 格

式的字节数组。 遥测服务器在 MQTT v3 消息与 IBM MQ 消息之间映射头; 请参阅 第 107 页的『MQ Telemetry 与队列管理器的集成』。

使用 Publication， MQInput 和 JMSInput 节点在 IBM Integration Bus 和 MQTT 客户机之间发送和接收发布。

通过使用消息流，您可以使用 HTTP 将遥测与 Web 站点集成，并使用 IBM MQ 和 WebSphere Adapters 与其他应用程序集成。

Windows Linux AIX MQ Telemetry 与队列管理器的集成

MQTT 客户机作为发布/预订应用程序与 IBM MQ 集成。 它可以发布或预订 IBM MQ 中的主题，创建新主题或使用现有主题。 由于 MQTT 客户机 (包括其自身) 或其他发布到其预订主题的 IBM MQ 应用程序，它从 IBM MQ 接收发布。 将应用规则来决定发布的属性。

不支持与 IBM MQ 提供的主题，发布，预订和消息相关联的许多属性。 第 107 页的『MQTT 客户机到 IBM MQ 发布/预订代理程序』 和 第 108 页的『IBM MQ 到 MQTT 客户机』 描述如何设置发布属性。 这些设置取决于发布是要进入还是从 IBM MQ 发布/预订代理程序进行。

在 IBM MQ 中，发布/预订主题与管理主题对象相关联。 MQTT 客户机创建的主题没有任何不同。 当 MQTT 客户机为发布创建主题字符串时， IBM MQ 发布/预订代理程序会将其与管理主题对象相关联。 代理程序将出版物中的主题字符串映射到最近的管理主题对象父代。 此映射与 IBM MQ 应用程序的映射相同。 如果没有用户创建的主题，那么发布主题将映射到 SYSTEM.BASE.TOPIC。 应用于发布的属性派生自主题对象。

当 IBM MQ 应用程序或管理员创建预订时，将命名该预订。 使用 IBM MQ Explorer， 或使用 **runmqsc** 或 PCF 命令列出订阅。 已命名全部 MQTT 客户机预订。 它们被赋予一个形式名称:
ClientIdentifier:Topic name

MQTT 客户机到 IBM MQ 发布/预订代理程序

MQTT 客户机已将发布发送到 IBM MQ。 遥测 (MQXR) 服务会将发布内容转换为 IBM MQ 消息。 IBM MQ 消息包含三个部分:

1. MQMD
2. RFH2
3. 消息

MQMD 属性设置为其缺省值，但 第 107 页的表 9 中注明的属性除外。

表 *9: MQMD* 属性的设置

MQMD 字段	**类型**	**值**
Format	MQCHAR8	MQFMT_RF_HEADER_2
UserIdentifier	MQCHAR12	设置为下列其中一项: MqttClient.ClientIdentifier MqttConnectOptions.UserName IBM MQ 管理员为遥测通道设置的用户标识。
Priority	MQLONG	MQPRI_PRIORITY_AS_Q_DEF (IBM MQ 的缺省值，与缺省值为 4 的 JMS 不同。)
Persistence	MQLONG	QoS=0→MQPER_NOT_PERSISTENT QoS=1→MQPER_PERSISTENT QoS=2→MQPER_PERSISTENT

RFH2 头不包含用于定义 JMS 消息类型的 <msd> 文件夹。 遥测 (MQXR) 服务会将 IBM MQ 消息创建为缺省 JMS 消息。 缺省 JMS 消息类型为 jms-bytes 消息。 应用程序可以将其他头信息作为消息属性进行访问; 请参阅 消息属性。

RFH2 值设置为如 第 108 页的表 10 中所示。 "格式" 属性在 RFH2 固定头中设置，其他值在 RFH2 文件夹中设置。

表 *10: RFH2* 属性的设置		
RFH2 属性	**类型/文件夹**	**头**
FORMAT	MQCHAR8	MQFMT_NONE
ClientIdentifier	mqtt/clientId	复制长度为 1...23 字节的 MqttClient.ClientIdentifier 。
QoS	mqtt/qos	从入局 MQTT 消息复制 QoS 。
消息标识	mqtt/msgid	如果 QoS 为 1 或 2，请从入局 MQTT 消息复制 **消息标识** 。
MQIsRetained	mqps/Ret	如果原始 MQTT 发布是使用 RETAIN 属性集发送的，并且消息是作为保留发布接收的，请进行设置。
MQTopicString	mqps/Top	将 MQTT 消息发布到的主题。

MQTT 发布中的有效内容将映射到 IBM MQ 消息的内容:

表 *11: MQTT* 出版物中的有效内容如何映射到 *IBM MQ* 消息内容		
消息内容	**类型**	**IBM MQ 消息的内容**
缓冲期	MQBYTE *n*	从入局 MQTT 消息复制字节。 长度可以为零。

IBM MQ 到 MQTT 客户机

客户机已预订发布主题。 IBM MQ 应用程序已发布到主题，从而导致发布由 IBM MQ 发布/预订代理发送到 MQTT 订户。 或者， IBM MQ 应用程序已直接向 MQTT 客户机发送非请求消息。 第 108 页的表 12 描述了如何在发送到 MQTT 客户机的消息中设置固定消息头。 将废弃 IBM MQ 消息头或任何其他头中的任何其他数据。 IBM MQ 消息中的消息数据将作为 MQTT 消息中的消息有效内容发送，而不进行任何更改。 MQTT 消息由遥测 (MQXR) 服务发送到 MQTT 客户机。

表 *12:* 如何在发送到 *MQTT* 客户机的 *IBM MQ* 消息中设置固定消息头		
MQTT 字段	**类型**	**值**
DUP	BOOLEAN	如果设置为 QoS = 1 或 2，并且在先前的传输中已将消息发送到此客户机，并且在一段时间后未确认该消息。
QoS	int	从 IBM MQ 中的发布/预订代理设置出局发布中 QoS 的值的方式取决于传入发布。 这取决于是从 MQTT 客户机发送传入发布，还是从 IBM MQ 应用程序发送传入发布。 **MQTT** 入局发布以及订户所请求的 QoS 中 QoS 的较小值。 **IBM MQ** 从传入发布派生的 QoS 的较小值: MQPER_NOT_PERSISTENT→QoS=0 MQPER_PERSISTENT→QoS=2 以及订户请求的 QoS 。 如果在没有预订的情况下将消息发送到客户机，那么缺省情况下会将 QoS 设置为 2。 客户机可以通过预订具有不同 QoS 的 DEFAULT.QoS 来更改此值。

表 12: 如何在发送到 *MQTT* 客户机的 *IBM MQ* 消息中设置固定消息头 (继续)		
MQTT 字段	**类型**	**值**
RETAIN	BOOLEAN	如果传入发布具有保留属性集，请进行设置。

第 109 页的表 13 描述了如何在发送到 MQTT 客户机的 MQTT 消息中设置变量消息头。

表 13: 如何在发送到 *MQTT* 客户机的 *MQTT* 消息中设置 *MQTT* 变量头属性		
MQTT 字段	**类型**	**值**
Topic name	**字符串**	发布消息所使用的主题字符串。
Message ID	**字符串**	MQMD.MsgId 属性 (当它放置在 SYSTEM.MQTT.TRANSMIT.QUEUE 中时)。
Payload	byte[]	将字节从传入发布直接复制到发布/预订代理。 长度可以为零。

Windows Linux AIX MQTT 无状态和有状态会话

MQTT 客户机可以创建与队列管理器的有状态会话。 当有状态 MQTT 客户机断开连接时，队列管理器会维护客户机创建的预订以及未完成的消息。 当客户机重新连接时，它将解析正在使用的消息。 它会发送排队等候传递的所有消息，并接收断开连接时针对预订发布的所有消息。

当 MQTT 客户机连接到遥测通道时，它将启动新会话或恢复旧会话。 新会话没有未确认的未完成消息，没有预订，也没有等待传递的发布。 当客户机连接时，它指定是从干净会话开始，还是恢复现有会话; 请参阅 干净会话。

如果客户机恢复现有会话，那么它将继续执行，就像连接未中断一样。 等待传递的发布将发送到客户机，并且未落实的任何消息传输都将完成。 当持久会话中的客户机与遥测 (MQXR) 服务断开连接时，客户机创建的任何预订都将保留。 预订的发布将在客户机重新连接时发送到客户机。 如果在不恢复旧会话的情况下重新连接，那么遥测 (MQXR) 服务将废弃这些发布。

会话状态信息由队列管理器保存在 SYSTEM.MQTT.PERSISTENT.STATE 队列中。

IBM MQ 管理员可以断开连接并清除会话。

Windows Linux AIX 未连接 MQTT 客户机时

当客户机未连接时，队列管理器可以继续代表它接收发布。 它们将在客户机重新连接时转发给客户机。 如果客户机意外断开连接，那么客户机可以创建队列管理器代表客户机发布的 “Last will and testament”。

如果您希望在客户机意外断开连接时收到通知，那么可以注册最后一个遗嘱和遗嘱发布; 请参阅 最后一个遗嘱和遗嘱发布。 它由遥测 (MQXR) 服务发送，如果它检测到与客户机的连接已断开，而客户机未发出请求。

客户机可以随时发布保留的出版物; 请参阅 保留的出版物和 MQTT 客户机。 对主题的新预订可以请求发送与主题关联的任何保留发布。 如果创建最后一个遗嘱和遗嘱作为保留发布，那么可以使用它来监视客户机的状态。

例如，客户机在连接时发布保留发布，以公布其可用性。 同时，它创建了一个保留的最后遗嘱和遗嘱出版物，宣布其不可用。 此外，就在它进行计划中的断开连接之前，它将其不可用性作为保留出版物发布。 要了解客户机是否可用，您将预订保留发布的主题。 您将始终收到三个出版物中的一个。

如果客户机要在断开连接时接收发布的消息，请将该客户机重新连接到其先前会话; 请参阅 第 109 页的『MQTT 无状态和有状态会话』。 其预订处于活动状态，直到删除这些预订或客户机创建干净的会话为止。

Windows Linux AIX MQTT 客户机与 IBM MQ 应用程序之间的松耦合

MQTT 客户机与 IBM MQ 应用程序之间的发布流是松散耦合的。 出版物可能源自 MQTT 客户机或 IBM MQ 应用程序，并且没有设置顺序。 出版商和订阅者之间存在松散的联系。 它们通过出版物和订阅进行间接互动。 您还可以从 IBM MQ 应用程序直接将消息发送到 MQTT 客户机。

MQTT 客户机和 IBM MQ 应用程序以两种方式松散耦合:

1. 发布者和订阅者通过出版物和预订与主题的关联进行松散耦合。 发布者和订户通常不知道发布或订阅的其他来源的地址或身份。
2. MQTT 客户机在不同的线程上发布，预订，接收发布和处理传递应答。

MQTT 客户机应用程序不会等到发布已交付。 应用程序将消息传递到 MQTT 客户机，然后应用程序在其自己的线程上继续操作。 传递令牌用于将应用程序与发布的传递同步; 请参阅 传递令牌。

将消息传递到 MQTT 客户机后，应用程序可以选择等待 delivery-token。 客户机可以提供在将发布传递到 IBM MQ 时调用的回调方法，而不是等待。 它还可以忽略 delivery-token。

根据与消息关联的服务质量，会将 delivery-token 立即返回到回调方法，或者可能在一段相当长的时间后返回。 在客户机断开连接并重新连接后，甚至可能返回 delivery-token。 如果服务质量为 *fire and* 算了，那么将立即返回 delivery-token。 在其他两种情况下，仅当客户机接收到已向订户发送发布的确认时，才会返回传递令牌。

由于客户机预订而发送到 MQTT 客户机的发布将传递到 `messageArrived` 回调方法。 `messageArrived` 在与主应用程序不同的线程上运行。

将消息直接发送到 MQTT 客户机

您可以通过两种方法之一向特定 MQTT 客户机发送消息。

1. IBM MQ 应用程序可以在没有预订的情况下直接将消息发送到 MQTT 客户机; 请参阅 直接将消息发送到客户机。
2. 另一种方法是使用 `ClientIdentifier` 命名约定。 使所有 MQTT 订户使用其唯一 `ClientIdentifier` 作为主题来创建预订。 发布到 *ClientIdentifier*。 该出版物将发送到预订了主题 *ClientIdentifier* 的客户机。 通过使用此技术，您可以将发布发送到特定 MQTT 订户。

Windows Linux AIX MQ Telemetry 安全性

保护遥测设备的安全很重要，因为这些设备可能可移动，并且用于无法严格控制的场所。 您可以使用 VPN 来保护从 MQTT 设备到遥测 (MQXR) 服务的连接。 MQ Telemetry 提供了另外两种安全性机制: TLS 和 JAAS。

TLS 主要用于加密设备与遥测通道之间的通信，并认证设备正在连接到正确的服务器; 请参阅 使用 TLS 的遥测通道认证。 您还可以使用 TLS 来检查是否允许客户机设备连接到服务器; 请参阅 MQTT 使用 TLS 的客户机认证。

JAAS 主要用于检查是否允许设备的用户使用服务器应用程序; 请参阅 MQTT 使用密码进行客户机认证。 JAAS 可以与 LDAP 配合使用，以使用单点登录目录来检查密码。

TLS 和 JAAS 可以结合使用以提供双因子认证。 您可以将 TLS 使用的密码限制为符合 FIPS 标准的密码。

拥有至少数以万计的用户，提供个人安全概要文件并不总是切实可行的。 使用概要文件来授权个别用户访问 IBM MQ 对象也并非始终可行。 而是将用户分组到用于授权发布和预订主题以及向客户机发送发布的类中。

配置每个遥测通道以将客户机映射到公共客户机用户标识。 对在特定通道上连接的每个客户机使用公共用户标识; 请参阅 MQTT 客户机身份和授权。

授权用户组不会影响每个人的认证。 每个用户都可以在客户机或服务器上使用其 **用户名** 和 **密码**进行认证，然后在服务器上使用公共用户标识进行授权。

Windows Linux AIX MQ Telemetry 全球化

MQTT v3 协议中的消息有效内容将编码为字节数组。 通常，处理文本的应用程序会在 UTF-8 中创建消息有效内容。 遥测通道将消息有效内容描述为 UTF-8，但不执行任何代码页转换。 发布主题字符串必须为 UTF-8。

应用程序负责将字母数据转换为正确的代码页和数字数据转换为正确的数字编码。

MQTT Java 客户机具有方便的 `MqttMessage.toString` 方法。 此方法将消息有效内容视为以本地平台缺省字符集 (通常为 UTF-8) 进行编码。 它将有效内容转换为 Java 字符串。 Java 具有 String 方法

getBytes，用于将字符串转换为使用本地平台缺省字符集编码的字节数组。 两个 MQTT Java 程序在具有相同缺省字符集的平台之间交换消息有效内容中的文本，在 UTF-8 中轻松高效地执行此操作。

如果其中一个平台的缺省字符集不是 UTF-8，那么应用程序必须建立用于交换消息的约定。 例如，发布程序使用 getBytes("UTF8") 方法指定从字符串到 UTF-8 的转换。 要接收消息文本，订户假定消息以 UTF-8 字符集进行编码。

遥测 (MQXR) 服务将来自 MQTT 客户机消息的所有入局发布的编码描述为 UTF-8。 它设置 MQMD.CodedCharSetId 到 UTF-8， RFH2.CodedCharSetId 到 MQCCSI_INHERIT；请参阅 第 107 页的『MQ Telemetry 与队列管理器的集成』。 出版物的格式设置为 MQFMT_NONE，因此通道或 MQGET 无法执行任何转换。

MQ Telemetry 的性能和可伸缩性

在管理大量客户机和提高 MQ Telemetry 的可伸缩性时，请考虑以下因素。

容量规划

有关 MQ Telemetry 的性能报告的信息，请参阅 MQ 性能文档。

连接

连接所涉及的成本包括

- 在处理器使用情况和时间方面设置连接本身的成本。
- 网络成本。
- 保持连接打开但不使用连接时使用的内存。

当客户保持连接时，会产生额外的负载。 如果连接保持打开状态，那么 TCP/IP 流和 MQTT 消息将使用网络来检查该连接是否仍然存在。 此外，对于保持打开的每个客户机连接，将在服务器中使用内存。

如果每分钟发送多条消息，请保持连接处于打开状态，以避免启动新连接的成本。 如果每 10-15 分钟发送不到一条消息，请考虑断开连接以避免使其保持打开状态的成本。 您可能希望将 TLS 连接保持打开状态但处于空闲状态的时间更长，因为设置成本更高。

此外，请考虑客户的能力。 如果客户机上有存储转发工具，那么您可以对消息进行批处理，并断开发送批处理之间的连接。 但是，如果客户机已断开连接，那么客户机无法从服务器接收消息。 因此，应用程序的用途会影响决策。

如果系统有一个客户机发送许多消息 (例如文件传输)，请不要等待每条消息的服务器响应。 而是发送所有消息并在结束时检查是否已接收到所有消息。 或者，使用 服务质量 (QoS)。

您可以根据消息来改变 QoS，使用 QoS 0 来交付不重要的消息，并使用 QoS 2 来交付重要的消息。 如果 QoS 为 0，那么消息吞吐量可能是 QoS 为 2 的两倍左右。

命名约定

如果要为许多客户机设计应用程序，请实施有效的命名约定。 为了将每个客户端映射到正确的 ClientIdentifier 上，必须使 ClientIdentifier 具有意义。 良好的命名约定使管理员能够更轻松地确定哪些客户机正在运行。 命名约定可帮助管理员在 IBM MQ Explorer 中过滤一长串客户机，并帮助确定问题; 请参阅 客户机标识。

吞吐量

主题名称的长度会影响流经网络的字节数。 发布或预订时，消息中的字节数可能很重要。 因此，限制主题名称中的字符数。 当 MQTT 客户机预订主题 IBM MQ 时，为其提供格式的名称:

```
ClientIdentifier: TopicName
```

要查看 MQTT 客户机的所有预订，可以使用 IBM MQ MQSC **DISPLAY** 命令:

```
DISPLAY SUB(' ClientID1:*')
```

在 IBM MQ 中定义供 MQTT 客户机使用的资源

MQTT 客户机连接到 IBM MQ 远程队列管理器。 IBM MQ 应用程序有两种基本方法可将消息发送到 MQTT 客户机: 将缺省传输队列设置为 SYSTEM.MQTT.TRANSMIT.QUEUE 或使用队列管理器别名。 如果存在大量 MQTT 客户机，请定义队列管理器的缺省传输队列。 使用缺省传输队列设置可简化管理工作; 请参阅 配置分布式排队以向 MQTT 客户机发送消息。

通过避免预订来提高可伸缩性。

当 MQTT V3 客户机预订主题时，将通过 IBM MQ 中的遥测 (MQXR) 服务创建预订。 预订将客户机的发布路由到 SYSTEM.MQTT.TRANSMIT.QUEUE。 每个发布的传输头中的远程队列管理器名称都设置为进行预订的 MQTT 客户机的 ClientIdentifier 。 如果有许多客户机 (每个客户机都有自己的预订) ，那么这将导致在整个 IBM MQ 发布/预订集群或层次结构中维护许多代理预订。 有关不使用发布/预订，而是改为使用基于点到点的解决方案的信息，请参阅 直接向客户机发送消息。

管理大量客户机

要支持许多并发连接的客户机，请通过设置 JVM 参数 **-Xms** 和 **-Xmx** 来增加可用于遥测 (MQXR) 服务的内存。 请按照以下步骤操作:

1. 在遥测服务配置目录中找到 java.properties 文件; 请参阅 Windows 上的遥测 (MQXR) 服务配置目录 或 Linux 上的遥测服务配置目录 。
2. 遵循文件中的指示; 1 GB 的堆足以用于 50,000 个并发连接的客户机。

   ```
   # Heap sizing options - uncomment the following lines to set the heap to 1G
   #-Xmx1024m
   #-Xms1024m
   ```

3. 添加其他命令行参数以传递到 java.properties 文件中运行遥测 (MQXR) 服务的 JVM; 请参阅 将 JVM 参数传递到遥测 (MQXR) 服务。

要增加 Linux 上的打开文件描述符数，请将以下行添加到 /etc/security/limits.conf/，然后重新登录。

```
@mqm soft nofile 65000
@mqm hard nofile 65000
```

每个套接字都需要一个文件描述符。 遥测服务需要一些额外的文件描述符，因此此数目必须大于所需的打开套接字数。

队列管理器对每个非持久预订使用对象句柄。 要支持许多活动的非持久预订，请增加队列管理器中的最大活动句柄数; 例如:

```
echo ALTER QMGR MAXHANDS(999999999) | runmqsc qMgrName
```

图 *48:* 更改 *Windows* 上的最大句柄数

```
echo "ALTER QMGR MAXHANDS(999999999)" | runmqsc qMgrName
```

图 *49:* 更改 *Linux* 上的最大句柄数

其他注意事项

在规划系统需求时，请考虑重新启动系统所花费的时间长度。 计划的停机时间可能会影响排队等待处理的消息数。 配置系统，以便可以在可接受的时间内成功处理消息。 查看磁盘存储，内存和处理能力。 对于某些客户机应用程序，可能在客户机重新连接时废弃消息。 要废弃消息，请在客户机连接参数中设置 CleanSession ; 请参阅 清除会话。 或者，在 MQTT 客户机中使用尽力而为的服务质量 0 进行发布和预订;

请参阅 服务质量。 从 IBM MQ 发送消息时使用 **非持久** 消息。 当系统或连接重新启动时，不会恢复具有这些服务质量的消息。

Windows Linux AIX MQ Telemetry 支持的设备

MQTT 客户机可以在从传感器和执行器到手持设备和车辆系统等一系列设备上运行。

MQTT 客户机很小，并且在受小内存和低处理能力限制的设备上运行。 MQTT protocol 可靠且具有较小的头，这适用于受低带宽，高成本和间歇性可用性约束的网络。

MQ Telemetry 通过 MQTT 客户机应用程序与遥测设备通信。 这些应用程序使用以下资源，所有这些资源都实现 MQTT v3 协议:

- 以下客户机库:
  - *MQTT client for Java*，用于为 (例如) Android， OS X， Linux 或 Windows 设备构建本机应用程序。 使用此客户机库的应用程序可以在 Java 的所有变体上运行，从最小 CLDC (已连接的受限设备配置) /MIDP (移动信息设备概要文件) 到 CDC (已连接的设备配置) /Foundation， J2SE (Java Platform， Standard Edition) 和 J2EE (Java Platform， Enterprise Edition)。 还支持 IBM jclRM 定制类库。 Java ME 平台通常用于小型设备，如执行器，传感器，手机和其他嵌入式设备。 Java SE 平台通常安装在更高端的嵌入式设备上，例如台式计算机和服务器。
  - *MQTT client for C*，用于为 (例如) iOS， OS X， Linux 或 Windows 设备构建本机应用程序。 此客户机库提供 C 参考实现以及针对 Windows 和 Linux 系统的预构建本机客户机。 C 参考实现使 MQTT 能够移植到各种设备和平台。 Intel 上的某些 Windows 系统 (包括 Windows 7， RedHat， Ubuntu 以及 ARM 平台 (例如 Eurotech Viper) 上的某些 Linux 系统) 实现运行 C 客户机的 Linux 版本，但 IBM 不提供对这些平台的服务支持。 如果您打算致电 IBM 支持中心，那么必须在受支持的平台上重现客户机问题。
  - *MQTT client for Java*，用于构建基于浏览器的 Web 应用程序。

MQTT 客户机库可从 Eclipse Paho 和 MQTT.org 免费获取。 请参阅 IBM MQ Telemetry Transport 样本程序。

安全性 IBM MQ

在 IBM MQ 中，有几种提供安全性的方法: 授权服务接口; 用户编写的或第三方的通道出口; 使用传输层安全性 (TLS) 的通道安全性，通道认证记录 和消息安全性。

授权服务接口

对象权限管理器 (OAM) 提供了使用 MQI 调用，命令和访问对象的授权，缺省情况下已启用此授权。 通过 IBM MQ 用户组和 OAM 控制对 IBM MQ 实体的访问。 管理员可以根据需要使用命令行界面来授予或撤销权限。

有关创建授权服务组件的更多信息，请参阅 在 AIX, Linux, and Windows 系统上设置安全性。

用户编写的通道出口或第三方通道出口

通道可以使用用户编写的通道出口或第三方通道出口。 有关更多信息，请参阅 消息传递通道的通道出口程序。

使用 TLS 的通道安全性

传输层安全性 (TLS) 协议提供业界标准的通道安全性，防止窃听，篡改和冒充。

TLS 使用公用密钥和对称技术来提供消息机密性和完整性以及相互认证。

有关 IBM MQ 中安全性的全面复审，包括有关 TLS 的详细信息，请参阅 保护。 有关 TLS 的概述 (包括本节中描述的命令的指针)，请参阅 加密安全协议: TLS。

通道认证记录

使用通道认证记录对授予在通道级别连接系统的访问权进行精确控制。 有关更多信息，请参阅 通道认证记录。

消息安全性

使用 Advanced Message Security(这是 IBM MQ 的单独安装和许可组件) 对使用 IBM MQ 发送和接收的消息提供加密保护。 请参阅 Advanced Message Security。

相关任务

保护

规划安全需求

IBM MQ.NET 受管客户机 TLS 支持

IBM MQ.NET 完全受管客户机提供基于 Microsoft.NET SSLStreams 套件的传输层安全性 (TLS) 支持。 这与基于 IBM Global Security Kit (GSKit)的其他 IBM MQ 客户机不同。

您可以开发 IBM MQ.NET 应用程序以在受管方式或非受管方式下运行。

- 在受管方式下， .NET 应用程序在 .NET CLR (公共语言运行时) 中工作，而无需任何跨平台调用 (例如调用 C MQI)。
- 在非受管方式下，将针对底层 MQI 操作调用 C MQI。 基本上，非受管方式接口包含基于 C MQI 的 .NET 包装程序类。

受管 IBM MQ.NET 客户机使用 Microsoft.NET Framework 库来实现 TLS 安全套接字协议。 Microsoft 中的系统.NET.Security.SSLStream 类用于在 IBM MQ.NET 中实现安全性 (TLS)。

非受管 IBM MQ.NET 客户机方式已支持基于 C MQI (和 GSKit) 的 TLS 功能。 即， TLS 操作由 C MQI 处理。 在此情况下， GSKit 将实现 TLS 安全套接字协议。

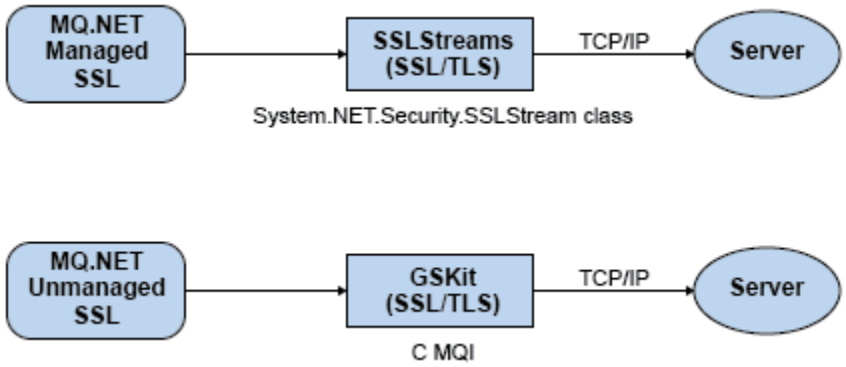


图 *50: IBM MQ.NET* 受管和非受管 *TLS* 比较

下表汇总了受管实现与非受管实现之间的差异:

表 *14:* 受管实施与非受管实施之间的差异

方式	协议	实现	注释
IBM MQ.NET 管理的 SSL	TLS	系统.NET.Security.SSLStream 类 SSLStream 类通过已连接的 TCP 套接字作为流运行	TLS 1.0 TLS 1.2 (仅适用于 Microsoft.NET Framework v4.5)

表 14: 受管实施与非受管实施之间的差异 (继续)			
方式	**协议**	**实现**	**注释**
IBM MQ.NET 非受管 SSL	TLS	GSKIT 和 C-MQI	TLS 安全套接字协议

相关概念

.NET 的安全套接字层 (SSL) 和传输层安全性 (TLS) 支持

IBM MQ MQI clients

IBM MQ MQI client 是 IBM MQ 产品的组件，可以安装在没有运行队列管理器的系统上。

IBM MQ MQI 客户机 是一个组件，允许在系统上运行的应用程序向在另一个系统上运行的队列管理器发出 MQI 调用。 来自调用的输出将发送回客户机，客户机会将其传递回应用程序。

通过使用 IBM MQ MQI client，在与客户机相同的系统上运行的应用程序可以连接到在另一个系统上运行的队列管理器。 应用程序可以向该队列管理器发出 MQI 调用。 此类应用程序称为 IBM MQ MQI client 应用程序，队列管理器称为 服务器队列管理器。

IBM MQ 服务器 是向一个或多个客户机提供排队服务的队列管理器。 所有 IBM MQ 对象 (例如，队列) 仅存在于队列管理器机器 (IBM MQ 服务器机器) 上，而不存在于客户机上。 IBM MQ 服务器还可以支持本地 IBM MQ 应用程序。

IBM MQ 服务器与普通队列管理器之间的区别在于，服务器与每个客户机都有专用通信链路。 有关为客户机和服务器创建通道的更多信息，请参阅 配置分布式排队。

IBM MQ MQI client 应用程序和服务器队列管理器使用 *MQI* 通道相互通信。 MQI 通道在客户机应用程序发出 **MQCONN** 或 **MQCONNX** 调用以连接到队列管理器时启动，并在客户机应用程序发出 **MQDISC** 调用以与队列管理器断开连接时结束。 MQI 调用流在 MQI 通道上的一个方向上的输入参数和输出参数在相反方向上的输入参数。

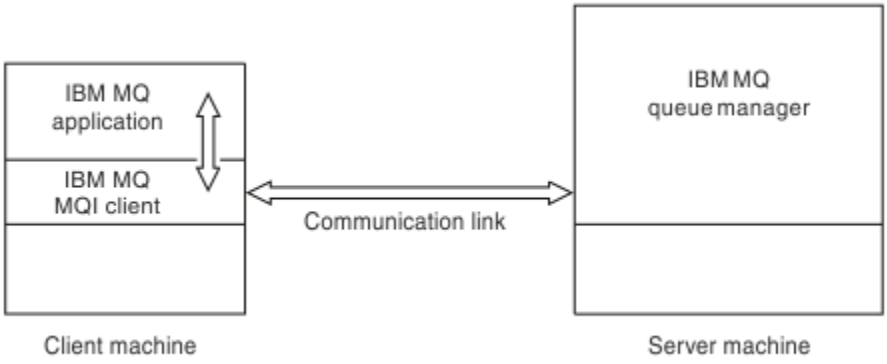


图 51: 客户机与服务器之间的链接

可以使用以下平台。 这些组合取决于您正在使用的 IBM MQ 产品，并在 第 116 页的『针对 IBM MQ 客户机的平台支持』中进行了描述。

IBM MQ MQI client	**IBM MQ 服务器**
AIX and Linux Windows IBM i	AIX and Linux Windows IBM i z/OS

MQI 可用于在客户机平台上运行的应用程序; 队列和其他 IBM MQ 对象保存在已安装在服务器上的队列管理器上。

要在 IBM MQ MQI client 环境中运行的应用程序必须首先与相关客户机库链接。 当应用程序发出 MQI 调用时， IBM MQ MQI client 会将请求定向到队列管理器，在该队列管理器中处理该请求，并从该队列管理器将应答发送回 IBM MQ MQI client。

应用程序与 IBM MQ MQI client 之间的链接是在运行时动态建立的。

您还可以使用 IBM MQ classes for .NET， IBM MQ classes for Java 或 IBM MQ classes for Java Message Service (JMS) 来开发客户机应用程序。 可以在以下平台上使用 Java 和 JMS 客户机:

- 

IBM i
- 

AIX
- 

Linux
- 

Windows

此处未描述 Java 和 JMS 的使用。 有关如何安装，配置和使用 IBM MQ classes for Java 和 IBM MQ classes for JMS 的完整详细信息，请参阅 使用 IBM MQ classes for Java 和 使用 IBM MQ classes for JMS。

客户机/服务器环境中的 IBM MQ 应用程序

当链接到服务器时，客户机 IBM MQ 应用程序可以采用与本地应用程序相同的方式发出大多数 MQI 调用。 客户机应用程序发出 MQCONN 调用以连接到指定的队列管理器。 然后，此队列管理器将处理指定从连接请求返回的连接句柄的任何其他 MQI 调用。

您必须将应用程序链接到相应的客户机库。 请参阅 为 IBM MQ MQI clients 构建应用程序。

相关概念

第 116 页的『为何使用 IBM MQ 客户机?』
使用 IBM MQ 客户机是实现 IBM MQ 消息传递和排队的有效方法。

第 118 页的『什么是扩展事务客户机?』
IBM MQ 扩展事务客户机可以在外部事务管理器的控制下更新由另一个资源管理器管理的资源。

第 119 页的『客户机如何连接到服务器』
客户机使用 MQCONN 或 MQCONNX 连接到服务器，并通过通道进行通信。

第 120 页的『事务管理和支持』
事务管理简介以及 IBM MQ 如何支持事务。

第 121 页的『扩展队列管理器设施』
您可以使用用户出口， API 出口或可安装服务来扩展队列管理器设施。

相关信息

如何设置 IBM MQ MQI client

为何使用 IBM MQ 客户机?

使用 IBM MQ 客户机是实现 IBM MQ 消息传递和排队的有效方法。

您可以使用在一台机器上运行的 MQI 和在另一台机器 (物理或虚拟) 上运行的队列管理器的应用程序。 这样做的好处是:

- 客户机上不需要完整的 IBM MQ 实现。
- 降低了客户机系统上的硬件需求。
- 减少了系统管理需求。
- 在客户机上运行的 IBM MQ 应用程序可以连接到不同系统上的多个队列管理器。
- 可以使用使用不同传输协议的备用通道。

针对 IBM MQ 客户机的平台支持

所有受支持的服务器平台上的 IBM MQ 都接受来自多个平台上的 IBM MQ MQI clients 的客户机连接。

在所有受支持的服务器平台上作为 基本产品和服务器 安装的 IBM MQ 可以接受来自以下平台上的 IBM MQ MQI clients 的连接:

- 

IBM i
- 

AIX
- 

Linux
- 

Windows

客户机连接在编码字符集标识 (CCSID) 和通信协议方面存在差异。

哪些应用程序在 IBM MQ MQI client 上运行?

客户机环境中支持完整 MQI。 这使几乎所有 IBM MQ 应用程序都能够配置为通过将 IBM MQ MQI client 上的应用程序链接到 MQIC 库 (而不是 MQI 库) 来在 IBM MQ MQI client 系统上运行。 例外情况如下所示:

- 带有信号的 MQGET
- 需要与其他资源管理器进行同步点协调的应用程序必须使用扩展事务客户机

如果启用预读，那么为了提高非持久消息传递性能，并非所有 MQGET 选项都可用。 该表显示了允许的选项以及是否可以在 MQGET 调用之间更改这些选项。

表 *15:* 启用预读时允许的 *MQGET* 选项

值	在启用预读并且可以在 MQGET 调用之间进行更改时允许	启用预读时允许，但不能在 MQGET 调用之间更改 [1]	启用预读时不允许使用的 MQGET 选项 [2]
MQGET MD 值	MsgId [3] CorrelId [3]	编码 CodedCharSetId	
MQGET MQGMO 选项	MQGMO_WAIT MQGMO_NO_WAIT MQGMO_FAIL_IF_QUIESCING MQGMO_BROWSE_FIRST [4] MQGMO_BROWSE_NEXT [4] MQGMO_BROWSE_MESSAGE_UNDER_CURSOR [4]	MQGMO_SYNCPOINT_IF_PERSISTENT MQGMO_NO_SYNCPOINT MQGMO_ACCEPT_TRUNCATED_MSG MQGMO_CONVERT MQGMO_LOGICAL_ORDER MQGMO_COMPLETE_MSG MQGMO_ALL_MSGS_AVAILABLE MQGMO_ALL_SEGMENTS_AVAILABLE MQGMO_MARK_BROWSE_HANDLE MQGMO_MARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_CO_OP MQGMO_UNMARK_BROWSE_HANDLE MQGMO_UNMARKED_BROWSE_MSG MQGMO_PROPERTIES_FORCE_MQRFH2 MQGMO_NO_PROPERTIES MQGMO_PROPERTIES_IN_HANDLE MQGMO_PROPERTIES_COMPATIBILITY	MQGMO_SET_SIGNAL MQGMO_SYNCPOINT MQGMO_MARK_SKIP 备份 (_BACKOUT) MQGMO_MSG_UNDER_CURSOR [4] MQGMO_LOCK MQGMO_UNLOCK
MQGMO 值		MsgHandle	

1. 如果在 MQGET 调用之间更改了这些选项，那么将返回 MQRC_OPTIONS_CHANGED 原因码。
2. 如果在第一个 MQGET 调用上指定这些选项，那么将禁用预读。 如果在后续 MQGET 调用上指定这些选项，那么将返回原因码 MQRC_OPTIONS_ERROR。
3. 客户机应用程序需要意识到，如果 MsgId 和 CorrelId 值在 MQGET 调用之间发生更改，那么具有先前值的消息可能已发送至客户机，并保留在客户机预读缓冲区中，直至被使用（或自动清除）为止。
4. 第一个 MQGET 调用确定在启用了预读时是否要从队列中浏览或获取消息。 如果应用程序尝试同时执行浏览与获取操作，将返回原因码 MQRC_OPTIONS_CHANGED。

5. MQGMO_MSG_UNDER_CURSOR 不能与预读配合使用。 在启用了预读时，可浏览或获取消息，但是不能同时执行这两个操作。

在 IBM MQ MQI client 上运行的应用程序可以同时连接到多个队列管理器，或者在 MQCONN 或 MQCONNX 调用上使用带有星号 (*) 的队列管理器名称 (请参阅 将 IBM MQ MQI client 应用程序连接到队列管理器 中的示例)。

什么是扩展事务客户机?

IBM MQ 扩展事务客户机可以在外部事务管理器的控制下更新由另一个资源管理器管理的资源。

如果您不熟悉事务管理的概念，请参阅 第 120 页的『事务管理和支持』。

请注意， XA 事务客户机现在作为 IBM MQ 的一部分提供。

客户机应用程序可以参与由它所连接的队列管理器管理的工作单元。 在工作单元中，客户机应用程序可以将消息放入该队列管理器所拥有的队列，并从中获取消息。 然后，客户机应用程序可以使用 **MQCMIT** 调用来落实工作单元，或者使用 **MQBACK** 调用来回退工作单元。 但是，在同一工作单元中，客户机应用程序无法更新其他资源管理器的资源，例如 Db2 数据库的表。 使用 IBM MQ 扩展事务客户机将除去此限制。

IBM MQ 扩展事务客户机是具有一些其他功能的 IBM MQ MQI client 。 使用此功能，客户机应用程序可以在同一工作单元中执行以下任务:

- 将消息放入它所连接的队列管理器所拥有的队列，并从中获取消息
- 更新 IBM MQ 队列管理器以外的资源管理器的资源

此工作单元必须由与客户机应用程序在同一系统上运行的外部事务管理器管理。 工作单元不能由客户机应用程序所连接到的队列管理器管理。 这意味着队列管理器只能充当资源管理器，而不能充当事务管理器。 这还意味着客户机应用程序只能使用外部事务管理器提供的应用程序编程接口 (API) 来落实或回退工作单元。 因此，客户机应用程序无法使用 MQI 调用 **MQBEGIN**， **MQCMIT** 和 **MQBACK**。

外部事务管理器使用连接到队列管理器的客户机应用程序所使用的相同 MQI 通道作为资源管理器与队列管理器进行通信。 但是，在发生故障后的恢复情况中，当没有应用程序在运行时，事务管理器可以使用专用 MQI 通道来恢复发生故障时队列管理器正在参与的任何不完整工作单元。

在此部分中，不具有扩展事务功能的 IBM MQ MQI client 称为 IBM MQ 基本客户机。 因此，您可以考虑 IBM MQ 扩展事务客户机由 IBM MQ 基本客户机组成，并添加扩展事务功能。

注: IBM i IBM i 上的 IBM MQ MQI client 不支持 IBM MQ 扩展事务功能。

针对扩展事务客户机的平台支持

Multi

扩展事务客户机可用于支持基本客户机的所有多平台。 客户机不可用于 z/OS。

使用扩展事务客户机的客户机应用程序只能连接到以下 IBM MQ 9.0 或更高版本产品的队列管理器:

- AIX IBM MQ for AIX
- IBM i IBM MQ for IBM i
- Linux IBM MQ for Linux
- Windows IBM MQ for Windows

z/OS 虽然没有在 z/OS 上运行的扩展事务客户机，但是使用扩展事务客户机的客户机应用程序可以连接到在 z/OS 上运行的队列管理器。

对于每个平台，扩展事务客户机的硬件和软件需求与 IBM MQ 基本客户机的硬件和软件需求相同。 扩展事务客户机支持编程语言 (如果它受 IBM MQ 基本客户机和您正在使用的事务管理器支持)。

有关所有平台的外部事务管理器的信息，请参阅 系统要求 IBM MQ。

客户机如何连接到服务器

客户机使用 MQCONN 或 MQCONNX 连接到服务器，并通过通道进行通信。

在 IBM MQ 客户机环境中运行的应用程序必须保持客户机与服务器之间的活动连接。

连接由发出 MQCONN 或 MQCONNX 调用的应用程序进行。 客户机和服务器通过 *MQI* 通道进行通信，或者在使用共享对话时，每个对话共享一个 MQI 通道实例。 当调用成功时， MQI 通道实例或对话将保持连接，直到应用程序发出 MQDISC 调用为止。 这是应用程序需要连接到的每个队列管理器的情况。

同一机器上的客户机和队列管理器

当您的机器还安装了队列管理器时，还可以在 IBM MQ MQI client 环境中运行应用程序。

在此情况下，您可以选择链接到队列管理器库或客户机库，但请记住，如果链接到客户机库，那么仍需要定义通道连接。 这在应用程序的开发阶段很有用。 您可以在不依赖他人的情况下在自己的机器上测试程序，并确信将其移至独立 IBM MQ MQI client 环境时仍有效。

不同平台上的客户

在此示例中，服务器在不同平台上与三个 IBM MQ MQI clients 通信。

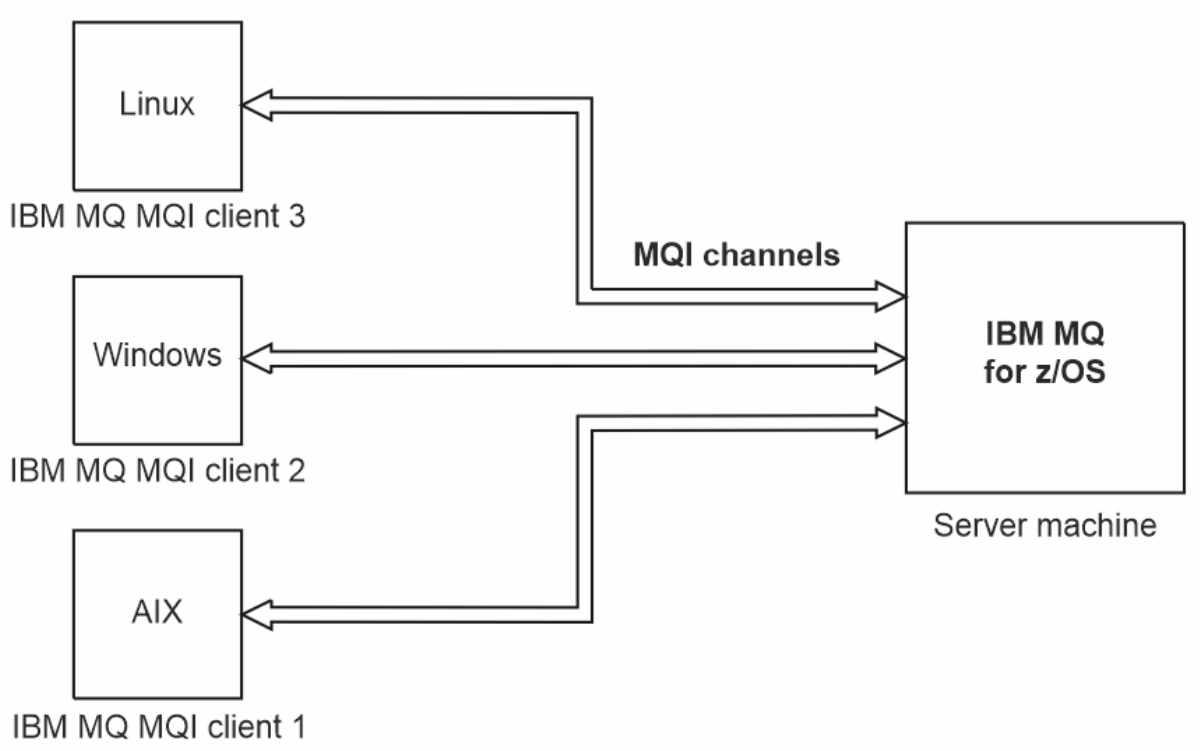


图 *52: IBM MQ* 服务器连接到不同平台上的客户机

其他更复杂的环境是可能的。 例如， IBM MQ 客户机可以连接到多个队列管理器，或者连接到作为队列共享组一部分的任意数量的队列管理器。

使用不同版本的客户机和服务器软件

如果您正在使用先前版本的 IBM MQ 产品，请确保服务器支持从客户机的 CCSID 进行代码转换。

IBM MQ 客户机可以连接到所有受支持的队列管理器版本。 如果要连接到较低版本的队列管理器，那么不能在客户机上的 IBM MQ 应用程序中使用来自较高版本的产品的功能部件和结构。

IBM MQ 队列管理器可以通过向下协商到相互支持的最高协议级别，与不同版本的客户机进行自身通信。 这意味着较旧的客户机可能与较旧的队列管理器级别配合使用。 建议客户机和服务器都是当前支持的 IBM MQ 版本，以促进问题诊断并通过 IBM 启用支持。

有关更多信息，请参阅 开发应用程序中支持的编程语言。

事务管理和支持

事务管理简介以及 IBM MQ 如何支持事务。

资源管理器 是一个计算机子系统，它拥有和管理可由应用程序访问和更新的资源。 以下是资源管理器的示例:

- IBM MQ 队列管理器，其资源是其队列
- Db2 数据库及其表中包含的资源

当应用程序更新一个或多个资源管理器的资源时，可能需要确保某些更新作为一个组成功完成，或者没有任何更新完成。 这种要求的原因是，如果其中一些更新成功完成，但其他更新未成功完成，那么业务数据将处于不一致状态。

以这种方式管理的资源的更新据说发生在 工作单元或 事务中。 应用程序可以将一组更新分组到工作单元中。

在工作单元期间，应用程序向资源管理器发出请求以更新其资源。 当应用程序发出落实所有更新的请求时，工作单元结束。 在落实更新之前，所有更新都不会对访问相同资源的其他应用程序可见。 或者，如果应用程序决定由于任何原因而无法完成工作单元，那么它可以发出请求，以回退其请求的所有更新，直至该点为止。 在这种情况下，任何更新都不会对其他应用程序可见。 这些更新通常在逻辑上相关，并且必须全部成功才能保留数据完整性。 如果一个更新成功，而另一个更新失败，那么数据完整性将丢失。

当工作单元成功完成时，会将其指定为 *commit*。 一旦落实，在该工作单元内进行的所有更新都将永久且不可逆。 但是，如果工作单元失败，那么将改为 回退所有更新。 此过程称为 同步点协调，在此过程中，将落实工作单元或以完整性回退工作单元。

落实或回退工作单元中的所有更新的时间点称为 同步点。 工作单元内的更新据说 在同步点控制内发生。 如果应用程序请求 在同步点控制之外的更新，那么资源管理器会立即落实该更新，即使存在正在进行的工作单元也是如此，并且以后无法回退该更新。

管理工作单元的计算机子系统称为 事务管理器或 点协调程序。

本地工作单元是其中更新的资源仅限于 IBM MQ 队列管理器资源的工作单元。 此处同步点协调由队列管理器本身使用单阶段落实过程提供。

全局 工作单元是其中属于其他资源管理器 (例如，符合 XA 的数据库) 的资源也会更新的工作单元。 在此，必须使用两阶段落实过程，并且工作单元可以由队列管理器本身进行协调，也可以由另一个符合 XA 的事务管理器 (例如 IBM TXSeries 或 BEA Tuxedo) 在外部进行协调。

事务管理器负责确保对工作单元中资源的所有更新都成功完成，或者没有任何更新完成。 应用程序向事务管理器发出落实或回退工作单元的请求。 事务管理器的示例为 CICS 和 WebSphere Application Server，尽管这两个事务管理器都具有其他功能。

某些资源管理器提供自己的事务管理功能。 例如， IBM MQ 队列管理器可以管理涉及对其自己的资源的更新和对 Db2 表的更新的工作单元。 队列管理器不需要单独的事务管理器来执行此功能，尽管如果是用户需求，可以使用此功能。 如果使用单独的事务管理器，那么会将其称为 外部事务管理器。

要让外部事务管理器管理工作单元，事务管理器与参与工作单元的每个资源管理器之间必须有标准接口。 此接口允许事务管理器和资源管理器相互通信。 其中一个接口是 *XA* 接口，它是许多事务管理器和资源管理器支持的标准接口。 XA 接口由开放式组在 分布式事务处理: *XA* 规范中发布。

当多个资源管理器参与工作单元时，事务管理器必须使用 两阶段落实 协议来确保工作单元中的所有更新都成功完成或都未完成，即使发生系统故障也是如此。 当应用程序向事务管理器发出请求以落实工作单元时，事务管理器将执行以下操作:

阶段 1 (准备落实)
事务管理器要求参与工作单元的每个资源管理器确保有关其资源的预期更新的所有信息都处于可恢复状态。 资源管理器通常通过将信息写入日志并确保将信息写入硬盘来执行此操作。 当事务管理器从每个资源管理器接收到有关其资源的预期更新的信息处于可恢复状态的通知时，阶段 1 将完成。

阶段 2 (落实)
当阶段 1 完成时，事务管理器将做出不可撤销的决策来落实工作单元。 它要求参与工作单元的每个资源管理器落实对其资源的更新。 当资源管理器接收到此请求时，它必须落实更新。 在此阶段，它无法选择将它们退出。 当事务管理器从每个资源管理器接收到其已将更新落实到其资源的通知时，阶段 2 完成。

XA 接口使用两阶段落实协议。

有关更多信息，请参阅 事务支持方案。

IBM MQ 还提供对 Microsoft Transaction Server (COM +) 的支持。 使用 Microsoft Transaction Server (COM +) 提供有关如何设置 IBM MQ 以利用 COM + 支持的信息。

扩展队列管理器设施

您可以使用用户出口， API 出口或可安装服务来扩展队列管理器设施。

用户出口

用户出口提供了一种机制，供您将自己的代码插入到队列管理器函数中。 支持的用户出口包括:

通道出口
这些出口会改变通道的运行方式。 通道出口-消息传递通道的出口程序中描述了通道出口。

数据转换出口
这些出口创建可放入应用程序中的源代码片段，以将数据从一种格式转换为另一种格式。 编写数据转换出口中描述了数据转换出口。

集群工作负载出口
此出口执行的函数由该出口的提供程序定义。 调用定义信息在 MQ_CLUSTER_WORKLOAD_EXIT-调用描述中提供。

API 出口

通过 API 出口，可以编写用于更改 IBM MQ API 调用（如 MQPUT 和 MQGET）行为的代码，然后直接在这些调用之前或之后插入该代码。 插入是自动的; 队列管理器在已注册的点处驱动退出代码。 有关 API 出口的更多信息，请参阅 使用和编写 API 出口。

可安装服务

可安装服务具有具有多个入口点的正规化接口 (API)。

可安装服务的实现称为 服务组件。 您可以使用 IBM MQ 随附的组件，也可以编写自己的组件以执行所需的功能。

目前，提供了以下可安装服务:

授权服务
授权服务允许您构建自己的安全设施。

实现服务的缺省服务组件是对象权限管理器 (OAM)。 缺省情况下， OAM 处于活动状态，您不必执行任何操作来对其进行配置。 您可以使用授权服务接口来创建其他组件以替换或扩充 OAM。 有关 OAM 的更多信息，请参阅 在 AIX, Linux, and Windows 系统上设置安全性。

名称服务
名称服务使应用程序能够通过识别远程队列 (就像它们是本地队列一样) 来共享队列。

您可以编写自己的名称服务组件。 例如，如果您打算将名称服务与 IBM MQ 配合使用，那么可能需要执行此操作。 要使用名称服务，您必须具有用户编写的组件或由其他软件供应商提供的组件。 缺省情况下，名称服务处于不活动状态。

相关概念

用户出口、API 出口和 IBM MQ 可安装服务

IBM MQ Java 语言接口

IBM MQ 提供了三个用于 Java 应用程序的应用程序编程接口 (API): IBM MQ classes for Jakarta Messaging， IBM MQ classes for JMS 和 IBM MQ classes for Java。

IBM 支持开放标准，并且是开放标准的积极参与者。

- 从 IBM MQ 8.0 开始，产品实现了 JMS 2.0 标准，该标准引入了新的简化 API 以及诸如共享预订之类的功能。
- V 9.3.0 V 9.3.0 从 IBM MQ 9.3.0 开始，还支持 Jakarta Messaging 3.0 。
- 此外， WebSphere Liberty 支持 JMS 2.0 和 Jakarta Messaging 3.0 与 IBM MQ 配合使用。

在 IBM MQ 中，有三个可在 Java 应用程序中使用的 API:

V 9.3.0 V 9.3.0 JM 3.0 **IBM MQ classes for Jakarta Messaging**

IBM MQ classes for Jakarta Messaging 是 Jakarta Messaging 提供程序，用于将 IBM MQ 的 Jakarta Messaging 接口实现为消息传递系统。 Jakarta Connectors Architecture 提供了将在 Jakarta EE 环境中运行的应用程序连接到企业信息系统 (EIS) (例如 IBM MQ 或 Db2) 的标准方法。

JMS 2.0 **IBM MQ classes for JMS**

IBM MQ classes for JMS 是 JMS 提供程序，用于将 IBM MQ 的 JMS 接口实现为消息传递系统。 Java Platform, Enterprise Edition Connector Architecture (JCA) 提供一种标准方式将运行在 Java EE 环境中的应用程序连接到企业信息系统 (EIS)，如 IBM MQ 或 Db2。

IBM MQ classes for Java

IBM MQ classes for Java 使您能够在 Java 环境中使用 IBM MQ 。 IBM MQ classes for Java 允许 Java 应用程序作为 IBM MQ 客户机连接到 IBM MQ，或直接连接到 IBM MQ 队列管理器。

注:

- V 9.3.0 V 9.3.0 JMS 2.0 已被 Jakarta Messaging 取代。 IBM MQ classes for JMS 继续支持 JMS 2.0 标准，但 Java 消息传递的未来增强功能将仅在 Jakarta Messaging 中出现，因此在 IBM MQ classes for Jakarta Messaging 中出现。 建议仅使用 IBM MQ classes for JMS 来维护和扩展现有 JMS 2.0 应用程序。 IBM MQ classes for Jakarta Messaging 应该是新开发的首选技术。
- Stabilized IBM MQ classes for Java 功能稳定在 IBM MQ 8.0 随附的级别上。 使用 IBM MQ classes for Java 的现有应用程序仍完全受支持，但此 API 是稳定的，因此将不会添加新的功能部件，并会拒绝针对增强功能的请求。 完全受支持意味着，在完成 IBM MQ 系统需求变化所需的任何更改的同时修复缺陷。

V 9.3.0 V 9.3.0 JM 3.0 从 IBM MQ 9.3 开始，使用 Java 8 构建 IBM MQ classes for Java， IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 。 必须使用这些级别或更高级别的 Java 运行时环境来运行使用这些接口的应用程序。

相关概念

从 Java 访问 IBM MQ -API 选项

V 9.3.0 V 9.3.0 为什么要将 IBM MQ 类用于 Jakarta Messaging?

为什么应该使用 IBM MQ classes for JMS?

为什么应该使用 IBM MQ classes for Java?

IBM MQ classes for JMS/Jakarta Messaging

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 是随 IBM MQ 提供的消息传递提供程序。 其中每个提供程序还提供了两组消息传递 API 的扩展。 Java Platform, Standard Edition (Java SE) 和 Java Platform, Enterprise Edition (Java EE) 应用程序都可以使用这些消息传递提供程序。

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0 支持 Jakarta Messaging 3.0. JMS 2.0 仍然完全支持。

JMS 和 Jakarta Messaging 规范定义了一组接口，应用程序可以使用这些接口来执行消息传递操作。 从 IBM MQ 8.0 开始，产品支持 JMS 2.0 版本的 JMS 标准。 此实现提供标准 API 的所有功能，但需要的接口更少，并且更便于使用。 有关更多信息，请参阅 第 125 页的『JMS 和 Jakarta Messaging 模型』 和 JMS 2.0 规范 (Java.net)。 JM 3.0 从 IBM MQ 9.3.0 开始，还支持 Jakarta Messaging 。

jakarta.jms (Jakarta Messaging 3.0) 或 javax.jms (JMS 2.0) 包指定消息传递接口的详细信息，消息传递提供程序为特定消息传递产品实现这些接口。 例如：

- IBM MQ classes for JMS 是 JMS 提供程序，用于实现 IBM MQ 的 JMS 接口，并且还为 JMS API 提供以下两组扩展：
  - IBM MQ JMS 扩展
  - IBM JMS 扩展
- 使用 javax.dims 或 jakarta.jms 创建的连接工厂，队列或主题对象，可以使用这些 API 中的任何一个 API 来寻址接口或一组 JMS 扩展; 即，可以将其强制转换为任何接口。 为保持最大程度的应用程序可移植性，请使用符合您需求的最通用的 API。

由于 JMS 和 Jakarta Messaging 有许多共同之处，因此本主题中对 JMS 的进一步提及可视为对两者的提及。 任何差异都会在必要时予以强调。

IBM MQ JMS 扩展

IBM MQ classes for JMS 还提供了针对 JMS API 的扩展。 IBM MQ classes for JMS 包含在 MQConnectionFactory， MQQueue 和 MQTopic 对象中实现的扩展。 这些对象包含特定于 IBM MQ 的属性和方法。 这些对象可以是受管对象，也可以由应用程序在运行时动态创建这些对象。 这些扩展称为 IBM MQ JMS 扩展。 请注意，在本文档中，应用程序在运行时动态创建的对象不会被视为受管对象。

IBM JMS 扩展

除 IBM MQ JMS 扩展外， IBM MQ classes for JMS 还提供了一组更通用的 JMS API 或 Java 扩展，作为所使用的编程语言。 这些扩展称为 IBM JMS 扩展，具有以下广泛目标：

- 在 IBM JMS 提供程序之间提供更高级别的一致性。
- 便于在两个 IBM 消息传递系统之间编写网桥应用程序。
- 为了更轻松地将应用程序从一个 IBM JMS 提供程序移植到另一个提供程序。

这些扩展的主要目标是在运行时动态创建和配置连接工厂和目标，但是这些扩展还提供了与消息传递不直接相关的功能，例如问题确定功能。

相关任务

使用 IBM MQ classes for JMS/Jakarta Messaging

配置 JMS 和 Jakarta Messaging 资源

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ classes for Jakarta Messaging: 概述

IBM MQ 9.3.0 引入了对 Jakarta Messaging 的支持。 对于 Jakarta Messaging 3.0， JMS 规范的控制从 Oracle 移至 Java 社区进程。 然而， Oracle 仍保留对“javax”名称的控制权，该名称在其他 Java 技术中也有使用。 因此，虽然 Jakarta Messaging 3.0 的功能与 JMS 2.0 相同，但在命名上还是有一些区别。 版本 3.0 的正式名称是 Jakarta Messaging ， 而不是 Java Message Service ， 而包和常量名称的前缀是 `jakarta` ， 而不是 `javax` 。

背景

多年来， Java 平台有两种形式: Standard Edition 和 Enterprise Edition。

Java Platform, Standard Edition (有时缩写为 Java SE) 是能够在独立上下文中运行的核心语言和类库。 Java SE 中的大多数 Java 包都具有以 "java." 开头的名称。

Java Platform, Enterprise Edition (Java EE) 对此进行了扩展，添加了诸如 "消息传递"， "各种 Bean" 和 "事务性" 等功能。 其中一些技术也可以在 Java SE 上下文中使用。 Java EE 中的大多数 Java 包历史上都具有以 "javax" 开头的名称。但是，存在一些交叉，因此某些 Java SE 包具有 "javax"。 作为其名称的前缀。

Java Message Service (JMS) 是 Java Platform, Enterprise Edition 的一部分。 Java EE 7 包含 JMS 2.0。

直到 Java EE 7，这些技术都由 Oracle 管理。

Java EE 技术最近已从 Oracle 的管理流程转变为由 Eclipse Foundation 监督的社区流程。

作为 "javax"。 无法将名称移至新项目，已采用新命名-所有包和属性名称现在都以 "j 惹" 作为前缀。 并且 Java Platform, Enterprise Edition 将在将来被称为 "Jakarta EE"。 版本编号仍在继续: 版本 8 是可在很大程度上被忽略的临时版本，而 Jakarta EE 9 是 "j 惹" 的点。 前缀生效。

在 IBM MQ 上下文中应用的主要 Jakarta EE 技术是 Jakarta Messaging 3.0 - Java Message Service (JMS) 2.0 的后继技术。 因此， Jakarta EE 9 合并 Jakarta Messaging 3.0。

IBM MQ 继续支持 Java EE 7 和 JMS 2.0，同时引入对 Jakarta EE 9 和 Jakarta Messaging 3.0 的支持。

交付的内容: Java SE

对于 Java Platform, Standard Edition，除了 IBM MQ classes for JMS (通过 IBM MQ 支持 JMS 2.0 操作) IBM MQ 9.3.0 提供 IBM MQ classes for Jakarta Messaging 之外，还提供了。 这些类提供了与 IBM MQ 集成的 Jakarta Messaging 3.0 提供程序，允许使用 IBM MQ 队列管理器来促进 Jakarta Messaging 操作。

这些文件作为标准 JAR 文件 `com.ibm.mq.jakarta.client.jar` 提供在 IBM MQ 安装的 `java/lib` 子目录中。

为了在 OSGi 容器 (例如 Apache Felix 或 Eclipse Equinox) 中使用， IBM MQ 还提供了一对 OSGi 捆绑软件:

- `com.ibm.mq.osgi.jms30.clientprereqs_`*`V.R.M.F`*`.jar`
- `com.ibm.mq.osgi.jms30.client_`*`V.R.M.F`*`.jar`

其中 *V.R.M.F* 表示 IBM MQ 的版本，例如 9.3.0.0。 可以在 IBM MQ 安装的 `java/lib/OSGi` 子目录中找到这些捆绑软件。

交付内容： Jakarta EE 9 ，以及 Jakarta EE 10

为了支持 Jakarta EE 9 和兼容的应用服务器中的 IBM MQ 消息， IBM MQ 为 Jakarta Messaging 提供了一个资源适配器： `wmq.jakarta.jmsra.rar`。 该文件位于 IBM MQ 安装目录下的 `java/lib/jca` 子目录中。

IBM MQ 继续在 安装的 子目录中提供 兼容的资源适配器 。 IBM MQ `java/lib/jca` Java EE 7 `wmq.jmsra.rar`

如何交付这些工件

这些 JAR 文件和资源适配器的 RAR 文件与预先存在的工件一起打包在常规的 IBM MQ 安装介质中——既有特定于平台的安装介质（如“ `.rpm` ”文件），也有可再分发介质（如自解压的可再分发客户端 JAR 文件）。

在 JMS 2.0 和 Jakarta Messaging 3.0 之间更改的内容

Jakarta EE 9 和 Jakarta Messaging 3.0 未引入任何新功能。 所有更改都是名称。 例如，在 JMS 2.0 中使用 "javax.jms.Connection" 时，在 Jakarta Messaging 3.0 中使用 "jakarta.jms.Connection"。

随着 Eclipse Foundation 推进 Jakarta EE 平台，它将在此基础上构建，并且此命名约定将用于将来引入的新功能。

在 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 之间更改的内容

摘要

IBM MQ classes for JMS(为 JMS 2.0 提供支持) 仍然可用，建议主要用于维护和扩展现有应用程序。 他们得到了充分的支持。

建议将为 Jakarta Messaging 3.0 提供支持的 IBM MQ classes for Jakarta Messaging 用于新开发。

在 IBM MQ 9.3.0 上，这两个产品在功能上等效。 仅命名不同。 但是，与 IBM MQ classes for JMS 相比，新的消息传递功能在 IBM MQ classes for Jakarta Messaging 中出现的可能性更大。

这两个产品可互操作。 IBM MQ classes for JMS 生成的消息可以由 IBM MQ classes for Jakarta Messaging 使用，反之亦然。 但是，这两个产品不得共存于单个应用程序中。

命名更改

表 *16:* 对软件包名称的更改

IBM MQ classes for JMS 包名	**IBM MQ classes for Jakarta Messaging 包名**
com.ibm.mq.jms[.*]	com.ibm.mq.jakarta.jms[.*]
com.ibm.jms	com.ibm.jakarta.jms
com.ibm.msg.client.jms.*	com.ibm.msg.client.jakarta.jms.*
com.ibm.msg.client.wmq.*	com.ibm.msg.client.jakarta.wmq.*

与公共服务 (跟踪，日志记录，本地语言支持等) 和 JMQI 实现 (本地和远程) 相关的包对于 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 都是公共的，因此在这些区域中不需要进行任何更改。

请注意，属性名称也已更改。 例如，用于在 IBM MQ classes for Jakarta Messaging 中启用 IBM MQ 扩展的属性为 **com.ibm.mq.jakarta.jms.SupportMQExtensions**。

独立于 IBM MQ classes for JMS 或 IBM MQ classes for Jakarta Messaging 的属性名称 (例如各种 **com.ibm.msg.client.commonservices.trace.*** 属性) 同样适用于这两个产品。

管理实用程序

crtmqenv 和 **setmqenv** 实用程序现在接受一个选项，以指定应该为 IBM MQ classes for JMS (`-j 2.0`) 还是 IBM MQ classes for Jakarta Messaging (`-j 3.0`) 配置类路径，并且存在称为 **runjms30** 的 **runjms** 实用程序的 IBM MQ classes for Jakarta Messaging 变体和类似名称。

当请求报告 Java 组件时， **dspmqver** 实用程序在其输出中包含 IBM MQ classes for Jakarta Messaging 。

要配置要通过 JNDI 检索的 IBM MQ classes for Jakarta Messaging 对象，新的 **JMS30Admin** 实用程序等同于 IBM MQ classes for JMS 的 **JMSAdmin** 实用程序。

请注意，由于底层对象来自不同的包。 由 **JMSAdmin** 创建的 JNDI 定义不能由 IBM MQ classes for Jakarta Messaging 使用，由 **JMS30Admin** 创建的 JNDI 定义也不能由 IBM MQ classes for JMS 使用。

注: 不支持 IBM MQ Explorer 提供的 IBM MQ classes for Jakarta Messaging 对象; 其 JNDI 集成仅适用于 IBM MQ classes for JMS 。

相关概念

为什么要将 IBM MQ 类用于 Jakarta Messaging?

JMS 和 Jakarta Messaging 模型

JMS 和 Jakarta Messaging 模型定义了一组接口， Java 应用程序可以使用这些接口来执行消息传递操作。 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 是定义 Java 消息传递对象如何与 IBM MQ 概念相关的消息传递提供程序。 JMS 和 Jakarta Messaging 规范期望将某些消息传递对象作为受管对象。

从 IBM MQ 8.0 开始，产品支持 JMS 标准的 JMS 2.0 版本，该版本引入了简化的 API ，同时还保留了 JMS 1.1 中的经典 API。

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0 支持 Jakarta Messaging 3.0. JMS 2.0 仍然完全支持。 由于 JMS 和 Jakarta Messaging 有许多共同之处，因此本主题中对 JMS 的进一步提及可视为对两者的提及。 任何差异都会在必要时予以强调。

简化的 API

JMS 2.0 引入了简化的 API，同时还保留了 JMS 1.1 中特定于域的接口和独立于域的接口。 简化的 API 减少了发送和接收消息所需的对象数，由以下接口构成：

ConnectionFactory
ConnectionFactory 是供 JMS 客户机用来创建连接的受管对象。 此接口也用于标准 API。

JMS 上下文
此对象结合了标准 API 的 Connection 和 Session 对象。 可以通过复制底层连接来从其他 JMSContext 对象创建 JMSContext 对象。

JMS 生产者
JMSProducer 由 JMSContext 创建，并用于向队列或主题发送消息。 JMSProducer 对象会导致创建发送消息所需的对象。

JMS 使用者
JMSConsumer 由 JMSContext 创建，并用于从主题或队列接收消息。

简化的 API 具有以下影响：

- JMSContext 对象始终自动启动底层连接。
- JMSProducer 和 JMSConsumer 现在可以直接处理消息体，而无需使用消息的 getBody 方法来获取整个消息对象。
- 在发送“消息体”（即消息内容）之前，可以使用方法链在 JMSProducer 对象上设置消息属性。 JMSProducer 将负责创建发送消息所需的所有对象。 通过使用 JMS 2.0，可以按如下所示设置属性和发送消息：

```
context.createProducer().
setProperty("foo", "bar").
setTimeToLive(10000).
setDeliveryMode(NON_PERSISTENT).
setDisableMessageTimestamp(true).
send(dataQueue, body);
```

JMS 2.0 还引入了可在多个使用者之间共享消息的共享预订。 所有 JMS 1.1 预订均视为非共享预订。

标准 API

以下列表汇总了标准 API 的主要 JMS 接口：

Destination
目标是应用程序发送消息的位置和/或应用程序从中接收消息的源。

ConnectionFactory
ConnectionFactory 对象封装了连接的一组配置属性。 应用程序使用连接工厂来创建连接。

Connection
Connection 对象将应用程序的活动连接封装到消息传递服务器中。 应用程序使用连接来创建会话。

Session
会话是用于发送和接收消息的单线程上下文。 应用程序使用会话来创建消息、消息生产者和消息使用者。 会话将进行事务处理或不进行事务处理。

消息
Message 对象封装了应用程序发送或接收的消息。

MessageProducer
应用程序使用消息生产者向目标发送消息。

MessageConsumer
应用程序使用消息使用者接收向目标发送的消息。

第 127 页的图 53 显示了这些对象及其关系。

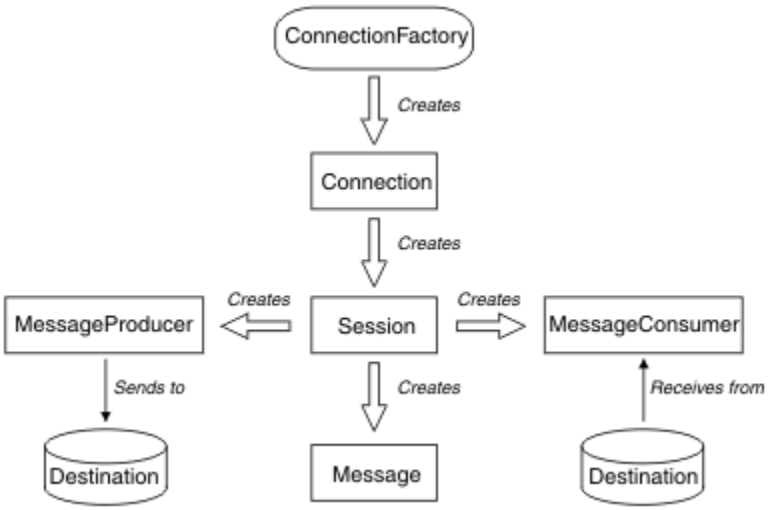


图 *53: JMS* 对象及其关系

此图显示了以下主要接口：ConnectionFactory、Connection、Session、MessageProducer、MessageConsumer、Message 和 Destination。 应用程序使用连接工厂来创建连接，并使用连接来创建会话。 然后，应用程序可以使用会话来创建消息、消息生产者和消息使用者。 应用程序使用消息生产者向目标发送消息，并使用消息使用者接收向目标发送的消息。

Destination、ConnectionFactory 或 Connection 对象可以由多线程应用程序的不同线程并行使用，但 Session、MessageProducer 或 MessageConsumer 对象不能由不同线程并行使用。 确保 Session、MessageProducer 或 MessageConsumer 对象不会被并行使用的最简单方法是为每个线程创建单独的 Session 对象。

JMS 支持以下两种类型的消息传递：

- 点到点消息传递
- 发布/预订消息传递

这两种类型的消息传递也称为消息传递域，您可以在应用程序中结合使用这两种类型的消息传递。 在点到点域中，目标是队列，在发布/预订域中，目标是主题。

对于 JMS 之前的 JMS 1.1 版本，点到点域的编程使用的是一组接口和方法，而发布/预订域的编程则使用的是另一组接口和方法。 二者相似，但相互独立。 从 JMS 1.1 开始，您可以使用一组同时支持这两种消息传递域的通用接口和方法。 通用接口为每个消息传递域提供独立于域的视图。 第 127 页的表 17 列出了独立于 JMS 域的接口及对应的域特定接口。

表 *17:* 独立于 *JMS* 域的接口及其相应的特定于域的接口

独立于域的接口	**点到点域的域特定接口**	**发布/预订域的域特定接口**
ConnectionFactory	QueueConnectionFactory	TopicConnectionFactory
Connection	QueueConnection	TopicConnection
Destination	队列	Topic
Session	QueueSession	TopicSession
MessageProducer	QueueSender	TopicPublisher
MessageConsumer	QueueReceiver QueueBrowser	TopicSubscriber

JMS 2.0 IBM MQ classes for JMS 2.0 支持较早的特定于 JMS 1.1 域的接口和 JMS 2.0 的简化 API。 因此， IBM MQ classes for JMS 2.0 可用于维护现有应用程序，包括在现有应用程序中开发新功能。

JM 3.0 IBM MQ classes for Jakarta Messaging 3.0 支持相同接口的 Jakarta Messaging 版本，建议用于新的应用程序开发。

在 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 中， JMS 对象通过以下方式与 IBM MQ 概念相关:

- Connection 对象具有派生自连接工厂（用于创建连接）属性的属性。 这些属性控制应用程序如何连接到队列管理器。 这些属性的示例包括队列管理器名称以及运行队列管理器的系统的主机名或 IP 地址（针对以 CLIENT 方式连接到队列管理器的应用程序）。
- Session 对象封装了 IBM MQ 连接句柄，因此定义了会话的事务范围。
- MessageProducer 对象和 MessageConsumer 对象各自封装了一个 IBM MQ 对象句柄。

使用 IBM MQ classes for JMS 或 IBM MQ classes for Jakarta Messaging 时，将应用 IBM MQ 的所有正常规则。 尤其需要注意，应用程序可以向远程队列发送消息，但只能从应用程序所连接到的队列管理器拥有的队列中接收消息。

JMS 规范认为 ConnectionFactory 和 Destination 对象都是受管对象。 管理员在中央存储库中创建和维护受管对象，而 JMS 应用程序可使用 Java 命名和目录接口 (JNDI) 来检索这些对象。

在 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 中， Destination 接口的实现是 Queue 和 Topic 的抽象超类，因此 Destination 实例是 Queue 对象或 Topic 对象。 独立于域的接口将队列或主题视为目标。 MessageProducer 或 MessageConsumer 对象的消息传递域由目标是队列还是主题来确定。

因此，在 IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 中，以下类型的对象可以是受管对象:

- ConnectionFactory
- QueueConnectionFactory
- TopicConnectionFactory
- 队列
- Topic
- XAConnectionFactory
- XAQueueConnectionFactory
- XATopicConnectionFactory

IBM MQ classes for JMS/Jakarta Messaging 体系结构

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 具有分层体系结构。 最上层代码是任何 IBM Java 消息传递提供程序都可以使用的公共层。

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0 支持 Jakarta Messaging 3.0. JMS 2.0 仍然完全支持。

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 具有分层体系结构，如图 第 129 页的图 54 所示。 最上层代码是可供任何 IBM JMS 或 Jakarta Messaging 提供程序使用的公共层。 当应用程序调用 JMS 或 Jakarta Messaging 方法时，不特定于消息传递系统的调用的任何处理都由公共层执行，这也会提供对调用的一致响应。 对特定于消息传递系统的调用的任何处理将委派到下一层。 在下图中，IBM MQ 消息传递提供者以及另外两个消息传递提供者（消息传递提供者 A 和消息传递提供者 B）都显示在下一层中。

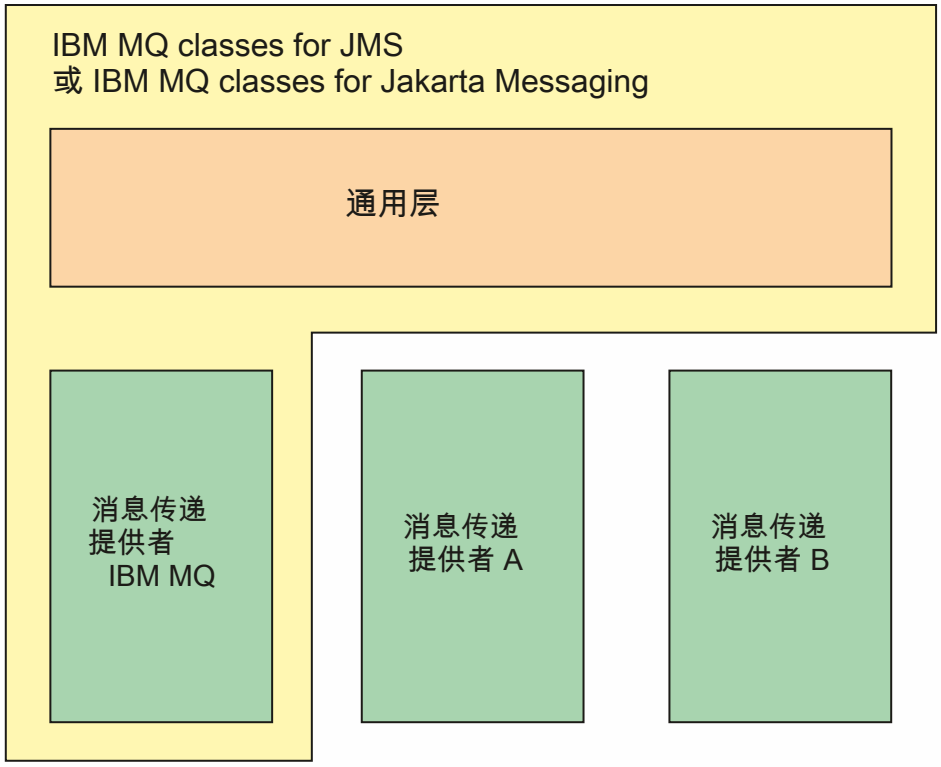


图 *54: IBM JMS* 和 *Jakarta Messaging* 提供程序的分层体系结构

分层体系结构实现了以下目标:

- 提高各种 IBM JMS 和 Jakarta Messaging 提供程序的行为一致性
- 更容易编写两个 IBM 消息传递系统之间的桥接应用程序
- 为了更轻松地将应用程序从一个 IBM JMS 或 Jakarta Messaging 提供程序移植到另一个提供程序

相关任务

使用 IBM MQ classes for JMS/Jakarta Messaging

受管对象支持

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 支持使用受管对象。

V 9.3.0 V 9.3.0 JM 3.0 从 IBM MQ 9.3.0 开始， Jakarta Messaging 3.0 ，支持开发新的应用程序。 IBM MQ 9.3.0 继续支持 现有应用程序。 JMS 2.0 不支持在同一应用程序中同时使用 Jakarta Messaging 3.0 API 和 JMS 2.0 API。 更多信息，请参阅为 JMS/Jakarta Messaging 使用 IBM MQ 类。

JMS 或 IBM MQ classes for Jakarta Messaging 应用程序中的逻辑流以 ConnectionFactory 和 Destination 对象开头。 应用程序使用 ConnectionFactory 对象来创建 Connection 对象，该对象表示从应用程序到消息传递服务器的活动连接。 应用程序使用 Connection 对象来创建 Session 对象，它是用于生成和使用消息的单线程上下文。 然后，应用程序可以使用 Session 对象和 Destination 对象来创建 MessageProducer 对象，应用程序使用该对象将消息发送到指定的目标。 目标是消息传递系统中的队列或主题，并由 Destination 对象封装。 应用程序还可以使用 Session 对象和 Destination 对象来创建 MessageConsumer 对象，应用程序使用该对象来接收已发送到指定目标的消息。

JMS 和 Jakarta Messaging 规范期望 ConnectionFactory 和 Destination 对象是受管对象。 管理员在中央存储库中创建和维护受管对象， JMS 或 Jakarta Messaging 应用程序使用 Java Naming Directory Interface (JNDI) 检索这些对象。 受管对象的存储库可以从简单文件到轻量级目录访问协议 (LDAP) 目录。

IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 支持使用受管对象。 应用程序可以使用通过 IBM MQ 公开的 IBM MQ classes for JMS 或 IBM MQ classes for Jakarta Messaging 的所有功能，而不

必将任何特定于 IBM MQ 的信息硬编码到应用程序本身中。 这种安排使应用程序在一定程度上独立于底层 IBM MQ 配置。

要实现此独立性，应用程序可以使用 JNDI 来检索存储为受管对象的连接工厂和目标，并仅使用 `javax.jms` (JMS 2.0) 或 `jakarta.jms` (Jakarta Messaging 3.0) 包中定义的接口来执行消息传递操作。

JMS 2.0 对于 "JMS 2.0，管理员可以使用 "IBM MQ、"JMS、"**JMSAdmin** 或 "IBM MQ Explorer"管理工具，在中央存储库中创建和维护管理对象。

JM 3.0 对于 Jakarta Messaging 3.0 ，不能使用 IBM MQ Explorer 管理 JNDI。 **JMSAdmin** 的 Jakarta Messaging 3.0 变体支持 JNDI 管理，即 **JMS30Admin**。

应用程序服务器通常为受管对象提供自己的存储库，并提供自己的用于创建和维护对象的工具。 因此，Java EE JM 3.0 或 Jakarta EE 应用程序可以使用 JNDI 从应用程序服务器存储库或中央存储库检索受管对象。

相关任务

配置 JMS 和 Jakarta Messaging 资源

Java EE 和 Jakarta EE 平台上受支持的通信类型

在 Java EE 和 Jakarta EE 平台上， IBM MQ classes for JMS 和 IBM MQ classes for Jakarta Messaging 支持应用程序组件与 IBM MQ 队列管理器之间的两种类型的通信。

V 9.3.0 V 9.3.0 JM 3.0 IBM MQ 9.3.0 支持 Jakarta Messaging 3.0. JMS 2.0 仍然完全支持。 由于 JMS 和 Jakarta Messaging 有许多共同之处，因此本主题中对 JMS 的进一步提及可视为对两者的提及。 任何差异都会在必要时予以强调。

在应用程序组件和 IBM MQ 队列管理器之间支持以下两种类型的通信：

- 出站通信
- 入站通信

出站通信

通过直接使用 JMS 或 Jakarta Messaging API ，应用程序组件将创建与队列管理器的连接，然后发送和接收消息。

例如，应用程序组件可以是应用程序客户机、servlet、Java Server Page (JSP)、企业 Java bean (EJB) 或消息驱动的 bean (MDB)。 在此类型的通信中，应用程序服务器容器在消息传递操作支持中仅提供低级功能，例如连接池和线程管理。

入站通信

如果使用的是入站通信，那么到达目标的消息将传递到 MDB，然后由 MDB 负责处理该消息。

Java EE JM 3.0 和 Jakarta EE 应用程序使用 MDB 异步处理消息。 MDB 用作 JMS 消息侦听器，并由用于定义消息处理方式的 onMessage() 方法实现。 已将 MDB 部署到应用程序服务器的 EJB 容器中。 MDB 的精确配置方式取决于所使用的应用程序服务器，但配置信息必须指定要连接到的队列管理器、如何连接到队列管理器、要监视哪个目标中的消息以及 MDB 的事务行为。 然后，MDB 容器将使用这些信息。 当满足 MDB 选择条件的消息到达指定目标时， EJB 容器使用 IBM MQ classes for JMS 或 IBM MQ classes for Jakarta Messaging 从队列管理器检索消息，然后通过调用其 onMessage() 方法将消息传递到 MDB。

与 IBM MQ classes for Java 的关系

IBM MQ classes for Java， IBM MQ classes for Jakarta Messaging 和 IBM MQ classes for JMS 是使用 MQI 的公共 Java 接口的同级。

第 131 页的图 55 显示了 IBM MQ classes for JMS， IBM MQ classes for Jakarta Messaging 和 IBM MQ classes for Java 之间的关系。

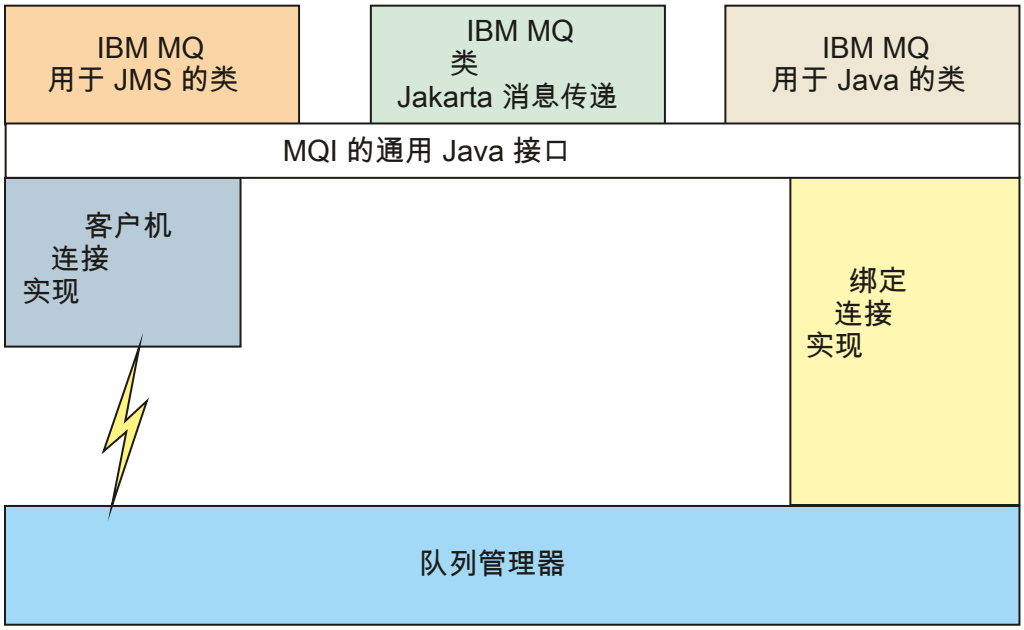


图 55: *IBM MQ classes for JMS*， *IBM MQ classes for Jakarta Messaging* 和 *IBM MQ classes for Java* 之间的关系

通常， Java 程序应仅使用一个接口与 IBM MQ - IBM MQ classes for Java， IBM MQ classes for Jakarta Messaging 或 IBM MQ classes for JMS 进行接口连接。 不支持混合接口，但有一个例外。 为保持与 IBM WebSphere MQ 7.0 之前的发行版的兼容性，用 Java 编写的通道出口类仍可以使用 IBM MQ classes for Java 接口，即使从 IBM MQ classes for JMS 调用这些通道出口类也是如此。 但是，使用 IBM MQ classes for Java 接口意味着应用程序仍依赖于下列其中一项:

- JMS 2.0 IBM MQ classes for Java JAR 文件 `com.ibm.mq.jar`。 如果不想在类路径中包含 `com.ibm.mq.jar`，可以改为使用 `com.ibm.mq.exits` 包中的接口集。
- V 9.3.0 V 9.3.0 JM 3.0 与 IBM MQ classes for Jakarta Messaging 进行互操作时使用 `com.ibm.mq.jakarta.client.jar`。

相关概念

V 9.3.0 V 9.3.0 为什么要将 IBM MQ 类用于 Jakarta Messaging?
为什么要使用 IBM MQ classes for JMS?
为什么应该将 IBM MQ 类用于 Java?

IBM MQ 消息传递提供程序

IBM MQ 消息传递提供者具有三种运行方式：正常方式、带有限制的正常方式和迁移方式。

IBM MQ 消息传递提供者具有以下三种运行方式:

- IBM MQ 消息传递提供者正常方式
- IBM MQ 消息传递提供者带有限制的正常方式
- IBM MQ 消息传递提供者迁移方式

IBM MQ 消息传递提供者正常方式使用 IBM MQ 队列管理器的所有功能来实现 JMS。 此方式已优化为使用 JMS 2.0 V 9.3.0 V 9.3.0 JM 3.0 或 Jakarta Messaging 3.0 API 和功能。

如果:

- 客户机在 **ConnectionFactory** 上指定 6 的提供者版本，客户机的行为方式与随 IBM WebSphere MQ 6.0 提供的客户机兼容。 仅支持 JMS 1.1 和 JMS 2 接口，但某些 JMS 2 功能 (例如共享预订，交付延迟和异步发送) 已禁用。 没有连接共享。
- 客户机在 **ConnectionFactory** 上指定 7 的提供程序版本，完全支持 JMS 1.1 和 JMS 2 接口。

- 未指定提供程序版本，尝试与提供程序版本 7 连接。 如果此操作失败，那么将对提供程序版本 6 进行进一步尝试。

如果要使用 IBM MQ Enterprise Transport 连接到 IBM Integration Bus ，请使用迁移方式。 如果使用 IBM MQ Real-Time Transport，那么会自动选择迁移方式，因为您已在连接工厂对象中显式选择了相应属性。 使用 IBM MQ Enterprise Transport 与 IBM Integration Bus 的连接遵循 配置 JMS **PROVIDERVERSION** 属性中描述的方式选择的一般规则。

相关任务

配置 JMS 资源

z/OS IBM MQ for z/OS 概念

IBM MQ for z/OS 使用的某些概念对于 z/OS 平台是唯一的。 例如，仅随 IBM MQ for z/OS 提供了日志记录机制，存储管理技术，恢复处置单元和队列共享组。 使用本主题作为有关这些概念的进一步信息的简介。

这些概念包括 IBM MQ for z/OS 使用的对象的概述，包括:

- 队列管理器
- 通道启动程序
- 共享队列和队列共享组
- 组内排队

以下主题还涵盖您需要的各种过程，包括:

- z/OS 上的系统定义
- 存储管理
- 恢复和重新启动
- IBM MQ for z/OS 中的安全概念

相关概念

第 133 页的『z/OS 上的队列管理器』
必须先安装 IBM MQ for z/OS 产品并启动队列管理器，然后才能让应用程序在 z/OS 系统上使用 IBM MQ 。 队列管理器拥有并管理 IBM MQ 所使用的资源集。

第 134 页的『z/OS 上的通道启动程序』
通道启动程序提供并管理用于启用 IBM MQ 分布式排队的资源。 IBM MQ 使用 消息通道代理程序 (MCA) 将消息从一个队列管理器发送到另一个队列管理器。

第 135 页的『用于管理 IBM MQ for z/OS 的术语和任务』
使用本主题作为术语以及特定于 IBM MQ for z/OS 的任务的简介。

第 137 页的『共享队列和队列共享组』
可以使用共享队列和队列共享组来实现 IBM MQ 资源的高可用性。 共享队列和队列共享组是 z/OS 平台上 IBM MQ for z/OS 独有的功能。

第 173 页的『组内排队』
本部分描述了组内排队，这是 z/OS 平台独有的 IBM MQ for z/OS 函数。 此函数仅对定义到队列共享组的队列管理器可用。

第 184 页的『z/OS 上的存储管理』
IBM MQ for z/OS 需要永久和临时数据结构，并使用页集和内存缓冲区来存储此数据。 这些主题提供了有关 IBM MQ 如何使用这些页集和缓冲区的更多详细信息。

第 188 页的『登录 IBM MQ for z/OS』
IBM MQ 维护发生数据更改和重要事件时的 日志 。 如果需要，可以使用这些日志将数据恢复到先前状态。

第 206 页的『在 z/OS 上恢复并重新启动』
使用本主题中的链接可了解 IBM MQ for z/OS 用于重新启动和恢复的功能。

第 219 页的『IBM MQ for z/OS 中的安全概念』
使用本主题来了解安全性对于 IBM MQ 的重要性，以及在系统上没有足够的安全性设置所产生的影响。

第 224 页的『z/OS 上的可用性』

IBM MQ for z/OS 具有许多用于实现高可用性的功能。 本主题描述了可用性的一些注意事项。

第 228 页的『z/OS 上的恢复单元处置』
某些事务应用程序在连接到队列共享组 (QSG) 中的队列管理器时，可以使用 GROUP (而不是 QMGR) 恢复处置单元，方法是在它们连接时指定 QSG 名称而不是队列管理器名称。 这将通过除去重新连接到 QSG 中的同一队列管理器的需求，使事务恢复更加灵活和稳健。

相关参考

第 197 页的『z/OS 上的系统定义』
IBM MQ for z/OS 使用许多缺省对象定义，并提供样本 JCL 来创建这些缺省对象。 使用本主题来了解这些缺省对象以及样本 JCL。

第 227 页的『IBM MQ for z/OS 上的监视和统计信息』
IBM MQ for z/OS 具有一组用于监视队列管理器和收集统计信息的工具。

z/OS 上的队列管理器

必须先安装 IBM MQ for z/OS 产品并启动队列管理器，然后才能让应用程序在 z/OS 系统上使用 IBM MQ 。 队列管理器拥有并管理 IBM MQ 所使用的资源集。

队列管理器

队列管理器 是向应用程序提供消息传递服务的程序。 使用消息队列接口（MQI）的应用程序可以将消息放置到队列并可从队列中获取消息。 队列管理器确保消息可以发送至正确的队列或传递至另一个队列管理器。 队列管理器处理向它发出的 MQI 调用以及提交给它的命令（无论从哪个源）。 队列管理器为每个调用或命令生成合适的完成代码。

队列管理器管理的资源包括:

- 用于保存 IBM MQ 对象定义和消息数据的页集
- 用于在队列管理器发生故障时恢复消息和对象的日志
- 处理器存储器
- 不同应用程序环境 (CICS， IMS 和批处理) 可通过其访问 IBM MQ API 的连接
- IBM MQ 通道启动程序，允许 z/OS 系统上的 IBM MQ 与其他系统之间进行通信

队列管理器具有名称，应用程序可以使用此名称连接到该队列管理器。

第 133 页的图 56 说明了队列管理器，其中显示了与不同应用程序环境的连接以及通道启动程序。

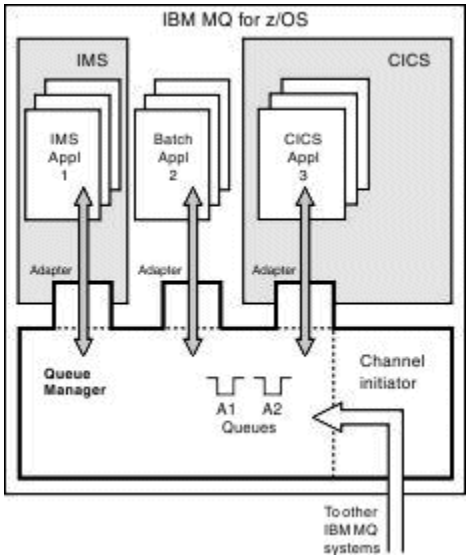


图 *56: IBM MQ for z/OS* 概述

z/OS 上的队列管理器子系统

在 z/OS 上， IBM MQ 作为在 IPL 时启动的 z/OS 子系统运行。 在子系统中，通过执行 JCL 过程来启动队列管理器，该过程指定 z/OS 数据集，这些数据集包含有关日志的信息，并保存对象定义和消息数据 (页集)。 子系统和队列管理器具有相同的名称，最多四个字符。 网络中的所有队列管理器都必须具有唯一名称，即使它们位于不同的系统，综合系统或平台上也是如此。

z/OS 上的通道启动程序

通道启动程序提供并管理用于启用 IBM MQ 分布式排队的资源。 IBM MQ 使用 消息通道代理程序 (MCA) 将消息从一个队列管理器发送到另一个队列管理器。

要将消息从队列管理器 A 发送到队列管理器 B ， 队列管理器 A 上的 发送 MCA 必须设置到队列管理器 B 的通信链路。 必须在队列管理器 B 上启动 接收 MCA ， 才能从通信链路接收消息。 此单向路径由发送 MCA ， 通信链路和接收 MCA 组成，称为 通道。 发送 MCA 从传输队列中获取消息并将它们从通道发送到接收 MCA。 接收 MCA 接收消息并将其放入目标队列。

在 IBM MQ for z/OS 中，发送和接收 MCA 都在通道启动程序 (通道启动程序也称为 移动者) 内运行。 通道启动程序在队列管理器的控制下作为 z/OS 地址空间运行。 只能有单个通道启动程序连接到队列管理器，并且它在与队列管理器相同的 z/OS 映像中运行。 可以同时在通道启动程序中运行数千个 MCA 进程。

第 134 页的图 57 显示了综合系统中的两个队列管理器。 每个队列管理器都有一个通道启动程序和一个本地队列。 由 AIX 和 Windows 上的队列管理器发送的消息将放置在本地队列上，由应用程序从中检索这些消息。 应答消息由类似的路由返回。

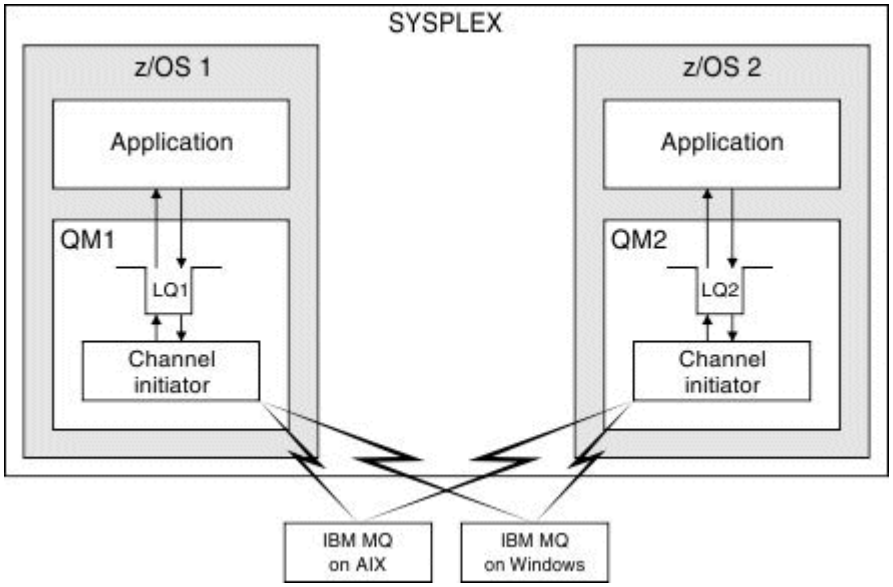


图 *57:* 队列管理器之间的通信

通道启动程序还包含与通道管理相关的其他进程。 这些过程包括:

侦听器
这些进程在诸如 TCP 之类的通信子系统上侦听入站通道请求，并在接收到入站请求时启动指定的 MCA。

监管者
这将管理通道启动程序地址空间，例如，它负责在发生故障后重新启动通道。

名称服务器
这用于将 TCP 名称解析为地址。

TLS 任务
这些用于执行加密和解密以及检查证书撤销列表。

z/OS 通道启动程序的 SMF 记录

通道启动程序 (CHINIT) 可以生成 SMF 统计信息记录和记帐记录以及有关任务和通道的信息。

CHINIT 可以生成具有以下类型信息的 SMF 统计信息记录和记帐记录:

- 任务: 分派器，适配器，域名服务器 (DNS) 和 SSL。 这些任务构成了所谓的 CHINIT 统计信息。
- 通道: 提供类似于 DIS CHSTATUS 命令提供的记帐信息。 这称为通道记帐。

IBM MQ for Multiplatforms 通过将 PCF 消息写入 SYSTEM.ADMIN.STATISTICS.QUEUE。 请参阅 通道统计信息消息数据，以获取有关如何在 IBM MQ for Multiplatforms 上记录统计信息的更多信息。

统计信息数据

您可以使用此信息来查找以下信息:

- 您是否需要更多的 CHINIT 任务，例如 SSL TCB 的数量以及这些任务使用的 CPU 数量。
- 针对这些任务的请求的平均时间。
- 时间间隔内针对 DNS 和 SSL 任务的最长持续时间请求以及发生此请求的时间。 您可以将此时间与通道可能迂到的问题相关联。

记帐数据

您可以使用此信息来监视通道使用情况并找出以下信息:

- 吞吐量最高的通道。
- 发送消息的速率以及发送数据的速率 (以 MB/ 秒计)。
- 实现的批处理大小。 如果实现的批处理大小接近为通道指定的批处理大小，那么通道可能接近其发送消息的限制。

使用 START TRACE 和 STOP TRACE 命令来控制记帐跟踪和统计信息跟踪的收集。 您可以在通道和队列管理器上使用 STATCHL 和 STATACLS 选项来控制通道是否生成 SMF 数据。

z/OS 用于管理 IBM MQ for z/OS 的术语和任务

使用本主题作为术语以及特定于 IBM MQ for z/OS 的任务的简介。

管理 IBM MQ for z/OS 所需的某些条款和任务特定于 z/OS 平台。 以下列表包含其中一些术语和任务。

- 共享队列
- 页集和缓冲池
- 记录
- 定制队列管理器环境
- 重新启动和恢复
- 安全性
- 可用性
- 处理对象
- 监控和统计信息
- 应用程序环境

共享队列

队列可以是 非共享队列，由 队列共享组拥有且只能由一个队列管理器访问，也可以是由队列共享组拥有的共享。 队列共享组由多个队列管理器组成，这些队列管理器在单个 z/OS 综合系统中运行，可以同时访问相同的 IBM MQ 对象定义和消息数据。 在队列共享组中，可共享对象定义存储在共享 Db2 数据库中。 共享队列消息保存在一个或多个耦合设施结构 (CF 结构) 中。 如果消息数据过大而无法直接存储在结构中 (大小超过

63 KB)，或者如果消息足够大，以至于安装定义的规则选择它进行卸载，那么消息控制信息仍存储在耦合设施条目中，但消息数据会卸载到共享消息数据集 (SMDS) 或共享 Db2 数据库中。 共享消息数据集，共享 Db2 数据库和耦合设施结构是由组中所有队列管理器共同管理的资源。

页集和缓冲池

将消息放入非共享队列时，队列管理器会将数据存储在页集上，以便在后续操作从同一队列获取消息时可以检索数据。 如果从队列中除去消息，那么保存数据的页集中的空间稍后将释放以供复用。 随着队列上保留的消息数增加，因此页集中使用的空间量增加，并且随着队列上的消息数减少，页集中使用的空间也会减少。

为了降低将数据写入页集和从页集读取数据的性能成本，队列管理器会将更新缓冲到处理器存储器中。 用于缓冲页集访问的存储量通过称为 缓冲池的 IBM MQ 对象进行控制。

有关页集和缓冲池的更多信息，请参阅 存储管理。

记录

对页集上保留的对象的任何更改以及对持久消息的操作都将记录为日志记录。 这些日志记录将写入称为 活动日志的日志数据集。 活动日志数据集的名称和大小保存在称为 引导数据集 (BSDS) 的数据集中。

当活动日志数据集填满时，队列管理器将切换到另一个日志数据集，以便日志记录可以继续，并将完整活动日志数据集的内容复制到 归档日志 数据集。 有关这些操作的信息 (包括归档日志数据集的名称) 保存在引导数据集中。 从概念上讲，队列管理器会循环使用一组活动日志数据集; 当填充活动日志时，会将日志数据卸载到归档日志中，并且活动日志数据集可供复用。

有关日志和引导数据集的更多信息，请参阅 第 188 页的『登录 IBM MQ for z/OS』。

定制队列管理器环境

启动队列管理器时，将读取一组控制队列管理器操作方式的初始化参数。 此外，将读取包含 IBM MQ 命令的数据集，并执行其包含的命令。 通常，这些数据集包含运行 IBM MQ 所需的系统对象的定义，您可以定制这些数据集以定义或初始化操作环境所需的 IBM MQ 对象。 读取这些数据集后，会将它们定义的任何对象存储在页集或 Db2 中。

有关初始化参数和系统对象的更多信息，请参阅 第 197 页的『z/OS 上的系统定义』。

恢复和重新启动

在 IBM MQ 操作期间的任何时候，处理器存储器中可能存在尚未写入页集的更改。 这些更改将写出到队列管理器中的后台任务最近最少使用的页集。

如果队列管理器异常终止，那么队列管理器重新启动的恢复阶段可以恢复丢失的页集更改，因为持久消息数据保存在日志记录中。 这意味着 IBM MQ 可以恢复持久消息数据和对象更改，直到发生故障为止。

如果作为队列共享组成员的队列管理器迁到耦合设施故障，那么仅当您已备份耦合设施结构时，才能恢复该队列上的持久消息。

有关恢复和重新启动的更多信息，请参阅 第 206 页的『在 z/OS 上恢复并重新启动』。

安全性

您可以使用外部安全性管理器，例如 Security Server (以前称为 RACF) 以保护 IBM MQ 拥有和管理的资源不受未经授权的用户访问。 您还可以使用传输层安全性 (TLS) 来实现通道安全性。 TLS 包含在 IBM MQ 产品中。

有关 IBM MQ 安全性的更多信息，请参阅 第 219 页的『IBM MQ for z/OS 中的安全概念』。

可用性

IBM MQ 的一些功能旨在提高队列管理器或通信子系统发生故障时的系统可用性。 有关这些功能部件的更多信息，请参阅第 224 页的『z/OS 上的可用性』。

处理对象

当队列管理器正在运行时，您可以通过 z/OS 控制台界面或通过使用 TSO 下的 ISPF 服务的管理实用程序来处理 IBM MQ 对象。 这两种机制都使您能够定义，变更或删除 IBM MQ 对象。 您还可以控制和显示各种 IBM MQ 和队列管理器功能的状态。

有关这些工具的更多信息，请参阅 可在 IBM MQ for z/OS 上发出 MQSC 和 PCF 命令的源。

您还可以使用 IBM MQ Explorer (图形用户界面) 来处理 IBM MQ 对象，该图形用户界面提供了使用队列，队列管理器和其他对象的可视方式。

监控和统计信息

有多个工具可用于监视队列管理器和通道启动程序。 您还可以收集统计信息以用于性能评估和记帐目的。

有关这些设施的更多信息，请参阅 第 227 页的『IBM MQ for z/OS 上的监视和统计信息』。

应用程序环境

当队列管理器已启动时，应用程序可以连接到该队列管理器并使用 IBM MQ API 启动该队列管理器。 这些应用程序可以是 CICS， IMS，批处理或 WebSphere Application Server 应用程序。 IBM MQ 应用程序还可以使用 CICS 和 IMS 网桥访问 CICS 和 IMS 系统上不知道 IBM MQ 的应用程序。

有关这些设施的更多信息，请参阅 第 230 页的『IBM MQ 和其他 z/OS 产品』。

有关编写 IBM MQ 应用程序的信息，请参阅以下文档:

- 开发应用程序
- 使用 C++
- 使用 IBM MQ classes for Java

z/OS 共享队列和队列共享组

可以使用共享队列和队列共享组来实现 IBM MQ 资源的高可用性。 共享队列和队列共享组是 z/OS 平台上 IBM MQ for z/OS 独有的功能。

本部分描述了属性和优点，并提供了有关多个队列管理器如何共享相同队列以及这些队列上的消息的信息。

什么是共享队列?

共享队列是本地队列的一种类型。 该队列上的消息可由综合系统中的一个或多个队列管理器访问。

队列共享组

可以访问同一组共享队列的队列管理器构成一个称为 队列共享组的组。

任何队列管理器都可以访问消息

队列共享组中的任何队列管理器都可以访问共享队列。 这意味着您可以将消息放入一个队列管理器上的共享队列，并从另一个队列管理器中获取同一消息。 这为队列共享组中不需要队列管理器之间的通道处于活动状态的通信提供了快速机制。

IBM MQ 支持将消息卸载到 Db2 或共享消息数据集 (SMDS)。 可配置任何大小的消息的卸载。

第 138 页的图 58 显示了三个队列管理器和一个耦合设施，构成队列共享组。 所有三个队列管理器都可以访问耦合设施中的共享队列。

应用程序可以连接到队列共享组中的任何队列管理器。 由于队列共享组中的所有队列管理器都可以访问所有共享队列，因此应用程序不依赖于特定队列管理器的可用性; 队列共享组中的任何队列管理器都可以为队列提供服务。

这将提供更大的可用性，因为队列共享组中的所有其他队列管理器都可以在其中一个队列管理器出现问题时继续处理该队列。

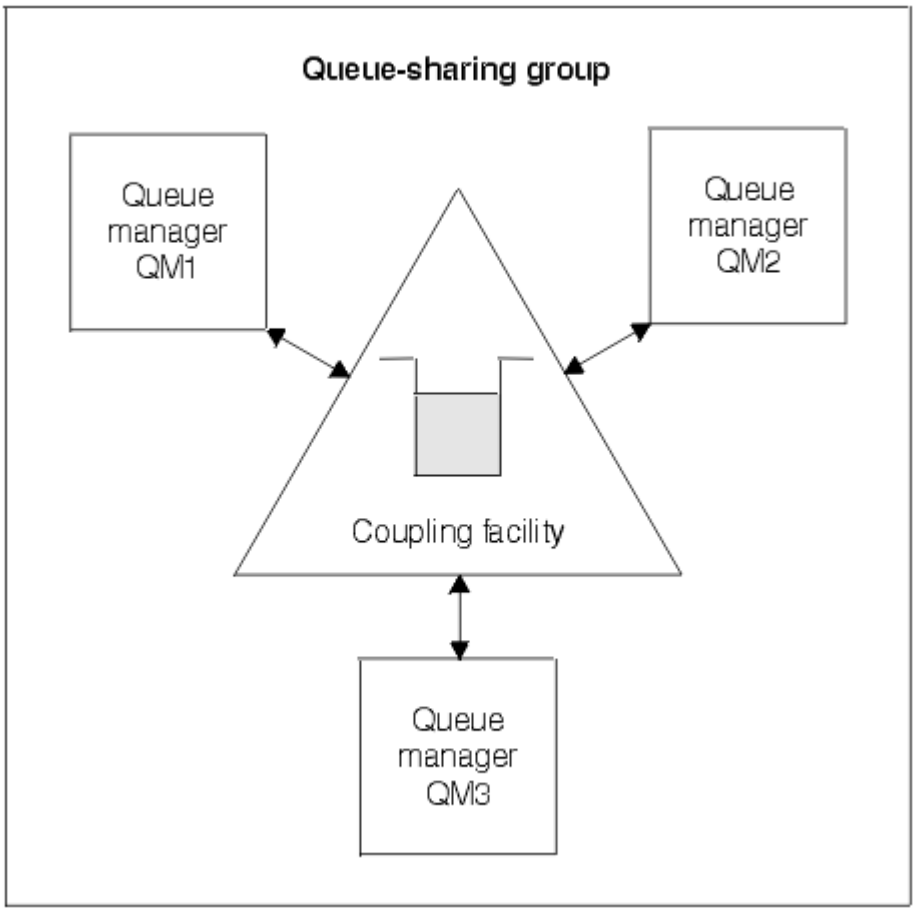


图 *58:* 队列共享组

队列定义由所有队列管理器共享

共享队列定义存储在 Db2 数据库表 OBJ_B_QUEUE 中。 因此，您只需要定义一次队列，然后队列共享组中的所有队列管理器都可以访问该队列。 这意味着要创建的定义较少。

相反，非共享队列的定义存储在拥有该队列的队列管理器的页集 0 上 (如 页集 中所述)。

如果已在定义队列管理器的页集中定义了具有该名称的队列，那么不能定义共享队列。 同样，如果存在同名的共享队列，那么不能在队列管理器页集上定义队列的本地版本。

什么是队列共享组?

可以访问相同共享队列的一组队列管理器称为队列共享组。 队列共享组的每个成员对同一组共享队列都具有访问权。

队列共享组具有最多为四个字符的名称。 该名称在网络中必须是唯一的，并且必须与所有队列管理器名称不同。

第 139 页的图 59 说明了包含两个队列管理器的队列共享组。 每个队列管理器都有一个通道启动程序及其自己的本地页集和日志数据集。

队列共享组的每个成员还必须连接到 Db2 系统。 Db2 系统必须全部位于同一个 Db2 数据共享组中，以便队列管理器可以访问用于保存共享对象定义的 Db2 共享存储库。 这些定义是任何类型的 IBM MQ 对象 (例如，队列和通道) 的定义，仅定义一次，然后组中的任何队列管理器都可以使用这些定义。 这些定义称为 全局 定义，在 专用定义和全局定义中进行了描述。

多个队列共享组可以引用特定数据共享组。 您可以指定 Db2 子系统的名称以及队列管理器在启动时在 IBM MQ 系统参数中使用的数据共享组。

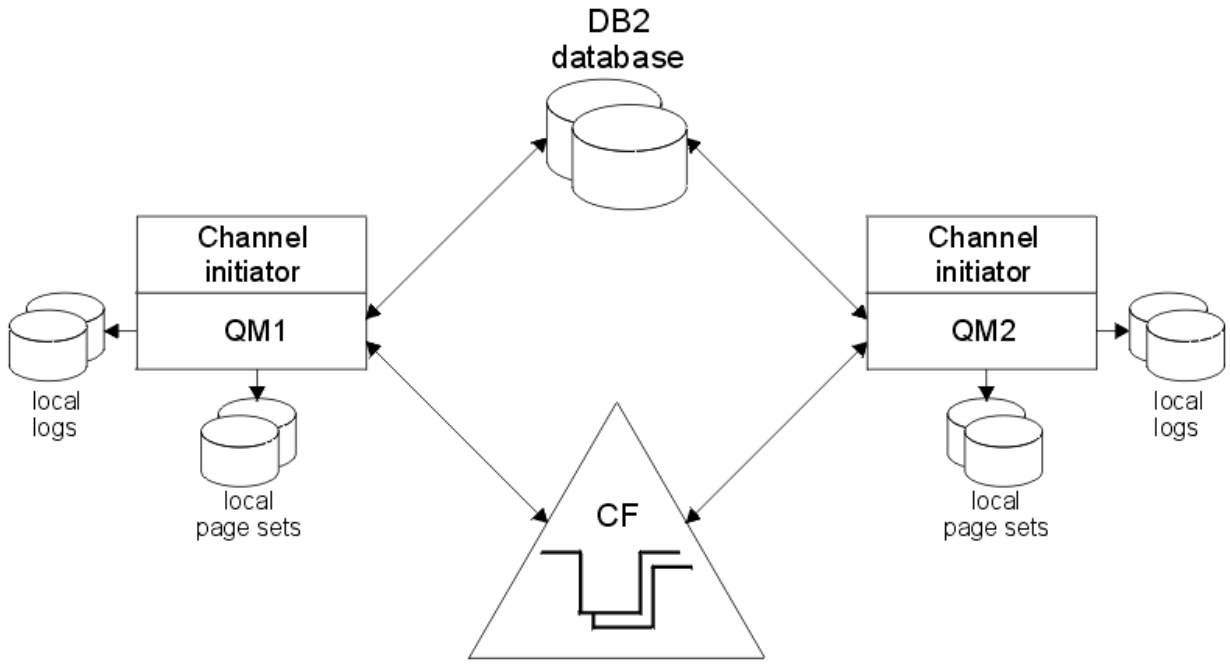


图 *59:* 队列共享组中队列管理器的组件

当队列管理器已加入队列共享组时，它有权访问为该组定义的共享对象，并且您可以使用该队列管理器在该组中定义新的共享对象。 如果在组中定义了共享队列，那么可以使用此队列管理器将消息放入这些共享队列并从中获取消息。 组中的任何队列管理器都可以检索共享队列上保留的消息。

您可以输入一次 MQSC 命令，并使其在队列共享组中的所有队列管理器上执行，就像在每个队列管理器上单独输入一样。 *command scope* 属性用于此目的。 将命令定向到不同的队列管理器中描述了此属性。

当队列管理器作为队列共享组的成员运行时，必须能够区分专用于该队列管理器的 IBM MQ 对象和全局定义的可供队列共享组中所有队列管理器使用的 IBM MQ 对象。 队列共享组处置 属性用于此目的。 此属性在 专用和全局定义中进行了描述。

您可以定义一组安全概要文件，用于控制对组中任何位置的 IBM MQ 对象的访问。 这意味着要定义的概要文件数量大大减少。

队列管理器只能属于一个队列共享组，并且该组中的所有队列管理器都必须位于同一综合系统中。 您可以在启动时在系统参数中指定队列管理器所属的队列共享组。

相关概念

第 140 页的『共享队列消息在何处挂起?』
共享队列上的每条消息都由 z/OS 耦合设施列表结构中的一个条目表示。 如果消息数据过大而无法容纳在同一条目中，那么会将其卸载到共享消息数据集 (SMDS) 或 Db2。

第 152 页的『使用共享队列的优点』
共享队列允许 IBM MQ 应用程序可伸缩，高度可用，并允许实现工作负载均衡。

第 169 页的『分布式排队和队列共享组』
分布式排队和队列共享组是可用于提高应用程序系统可用性的两种方法。 使用本主题来查找有关这些方法的更多信息。

第 171 页的『使用共享队列影响工作负载分布』

使用本主题来了解影响队列共享组中的共享队列的工作负载分布的因素。

相关参考

第 172 页的『在何处查找有关共享队列和队列共享组的更多信息』
使用本主题中的表来查找有关 IBM MQ for z/OS 如何使用共享队列和队列共享组的更多信息。

z/OS 共享队列消息在何处挂起?

共享队列上的每条消息都由 z/OS 耦合设施列表结构中的一个条目表示。 如果消息数据过大而无法容纳在同一条目中，那么会将其卸载到共享消息数据集 (SMDS) 或 Db2。

如果已将 CF 结构配置为使用系统类内存 (SCM)，那么 IBM MQ 可以在不进行其他配置的情况下使用此结构。

要点: IBM z16 计划是 IBM Z ® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

共享队列消息存储器

放入共享队列的消息不会存储在页集上，也不会使用缓冲池。

共享队列中的消息在 z/OS 耦合设施 (CF) 中的列表结构上具有条目。 同一综合系统中的许多队列管理器可以使用 CF 列表结构来访问这些消息。

小型共享队列消息的消息数据通常包含在耦合设施条目中。 对于较大的消息，消息数据可以存储在共享消息数据集 (SMDS) 中，也可以作为一个或多个二进制大对象 (BLOB) 存储在由 Db2 数据共享组共享的 Db2 表中。 超过 63 KB 的消息数据始终会卸载到 SMDS 或 Db2。 还可以选择以相同方式卸载较小的消息，以节省耦合设施结构中的空间。 请参阅第 141 页的『为共享消息指定卸载选项』以获取更多详细信息。

放置在共享队列上的消息将在耦合设施结构中引用，直到 MQGET 检索到这些消息为止。 耦合设施操作用于:

- 搜索下一条可检索的消息
- 锁定共享队列上未落实的消息
- 通知相关队列管理器已落实消息的到达

对持久消息执行的 MQPUT 和 MQGET 操作记录在执行该操作的队列管理器的日志中。 这将最大程度地降低耦合设施发生故障时数据丢失的风险。

耦合设施

共享队列上保存的消息在耦合设施中引用。 耦合设施位于综合系统中的任何 z/OS 映像外部，通常配置为在不同的电源上运行。 因此，耦合设施对软件故障具有弹性，您可以对其进行配置，以使其对硬件故障或电源中断具有弹性。 这意味着存储在耦合设施中的消息高度可用。

IBM MQ 使用的每个耦合设施列表结构都专用于特定队列共享组，但一个耦合设施可以保存多个队列共享组的结构。 不同队列共享组中的队列管理器无法共享数据。 队列共享组中最多 32 个队列管理器可以同时连接到耦合设施列表结构。

单个耦合设施列表结构最多可包含 512 个共享队列。 存储在结构中的消息数据总量受结构容量的限制。 但是，对于 **CFLEVEL(5)**，您可以使用 offload 参数来卸载小于 63 KB 的消息的数据，从而增加可以存储在结构中的消息数，尽管每条消息仍至少需要一个耦合设施条目以及至少 768 个字节的数据，由该条目的 256 个字节和头和描述符的两个元素的 512 个字节组成。

列表结构的大小受以下因素限制:

- 它必须位于单个耦合设施中。
- 它可能与 IBM MQ 和其他产品的其他结构共享可用的耦合设施存储器。

耦合设施列表结构可以具有与其关联的存储类内存。 在某些情况下，当与共享队列配合使用时，此存储类内存很有用。 请参阅 第 153 页的『将存储类内存与共享队列配合使用』 以获取更多信息。

规划 CF 结构大小

如果需要有关 CF 结构大小调整的指导，那么可以使用 MP16: IBM MQ for z/OS 容量规划和调整 supportpac。 您还可以使用基于 Web 的工具 CFSizer，该工具由 IBM 提供以帮助处理 CF 大小。

CF 结构对象

队列管理器对耦合设施结构的使用在 CF 结构 (CFSTRUCT) IBM MQ 对象中指定。

这些结构对象存储在 Db2 中。

使用与耦合设施结构相关的 z/OS 命令或定义时，需要队列共享组名称的前四个字符。 但是， IBM MQ CFSTRUCT 对象始终存在于单个队列共享组中，因此其名称不包含队列共享组名称的前四个字符。 例如，从 SQ03 开始在队列共享组中定义的 CFSTRUCT (MYDATA) 将使用耦合设施列表结构 SQ03MYDATA。

CF 结构具有用于确定其功能能力的 CFLEVEL 属性:

- 1 或 2-可用于小于 63 KB 的非持久消息
- 3-可用于小于 63 KB 的持久和非持久消息
- 4-可用于最大为 100 MB 的持久和非持久消息
- 5-可用于最大为 100 MB 的持久和非持久消息，并选择性地卸载到共享消息数据集 (SMDS) 或 Db2。

注: 使用 IBM MQ 时，可以加密耦合设施结构。 请参阅 加密耦合设施结构数据 以获取更多信息。

耦合设施的备份和恢复

可以使用 IBM MQ 命令 BACKUP CFSTRUCT 来备份耦合设施列表结构。 这会将当前在 CF 结构中的持久消息的副本放入进行备份的队列管理器的活动日志数据集中，并将备份的记录写入 Db2。

如果耦合设施发生故障，那么可以使用 IBM MQ 命令 RECOVER CFSTRUCT。 这将使用来自 Db2 的备份记录从 CF 结构的备份中查找和复原持久消息。 将使用队列共享组中所有队列管理器的日志来重放自上次备份以来的任何活动，然后将 CF 结构复原到故障前的点。

请参阅 BACKUP CFSTRUCT 和 RECOVER CFSTRUCT 命令以获取更多详细信息。

相关概念

第 141 页的『为共享消息指定卸载选项』
您可以选择共享队列消息的消息数据存储在 Db2 表或共享消息数据集 (SMDS) 中的位置。 您还可以根据消息的大小以及耦合设施结构 (CF) 的当前使用情况来选择卸载哪些消息。

第 143 页的『管理共享消息数据集 (SMDS) 环境』
如果选择共享消息数据集以卸载大型消息，那么还必须了解 IBM MQ 用于管理这些数据集的信息以及用于处理此信息的命令。 使用本主题来了解如何管理共享消息数据集。

z/OS 为共享消息指定卸载选项

您可以选择共享队列消息的消息数据存储在 Db2 表或共享消息数据集 (SMDS) 中的位置。 您还可以根据消息的大小以及耦合设施结构 (CF) 的当前使用情况来选择卸载哪些消息。

可以从耦合设施中卸载共享队列的消息数据，并将其存储在 Db2 表或称为 共享消息数据集 (SMDS) 的 IBM MQ 受管数据集中。

对于大于耦合设施条目大小 63 KB 的消息，将消息数据卸载到 SMDS 可能比卸载到 Db2 具有显着的性能改进。

仍使用耦合设施结构中的列表条目来管理每个共享队列消息，但当将消息数据卸载到 SMDS 时，该耦合设施条目仅包含一些控制信息以及对存储消息的相关磁盘块的引用列表。 使用此机制意味着每条消息所需的耦合设施元素存储量仅为消息的实际大小的一小部分。

选择共享队列消息的存储位置

通过 **CFSTRUCT** 定义上的 **OFFLOAD(SMDS|DB2)** 参数来控制 SMDS 或 Db2 共享消息存储器的选择。 **OFFLOAD(SMDS)** 是缺省值。

此参数还要求 **CFSTRUCT** 使用 **CFLEVEL(5)** 或更高版本。

OFFLOAD 参数仅在 **CFLEVEL(5)**中有效。 请参阅 DEFINE CFSTRUCT 以获取更多详细信息。

OFFLOAD(DB2) 主要用于迁移目的。

选择卸载哪些共享队列消息

根据消息数据的大小以及耦合设施结构的当前使用情况，将消息数据卸载到 SMDS 或 Db2 。 有三个规则，每个规则指定一对匹配的参数。 这些参数是对应的耦合设施结构使用率阈值百分比 (**OFFLDnTH**) 和消息大小限制 (**OFFLDnSZ**)。

这三个规则的当前实现是使用以下关键字对指定的:

- OFFLD1TH 和 OFFLD1SZ
- OFFLD2TH 和 OFFLD2SZ
- OFFLD3TH 和 OFFLD3SZ

规则对	缺省值	描述
规则对 1	OFFLD1TH(70) 和 OFFLD1SZ(32K)	如果耦合设施结构针对超过 32 KB 的消息超过 70% 的完全卸载数据
规则对 2	OFFLD2TH(80) 和 OFFLD2SZ(4K)	如果耦合设施结构对超过 4 KB 的消息的满卸载数据超过 80%
规则对 3	OFFLD3TH(90) 和 OFFLD3SZ(0K)	如果耦合设施结构对超过 0 KB (所有消息) 的消息的完全卸载数据超过 90%

如果卸载规则具有 OFFLD *x* SZ 值 64K ，那么表示该规则无效。 在这种情况下，仅当另一个卸载规则生效时，或者如果消息大于 63.75 KB ，那么将卸载消息，太大而无法存储在结构中。

卸载的每条消息仍需要耦合设施中的 0.75 KB 存储空间。

可以为每个结构指定的三个卸载规则旨在如下所示使用。

- 性能
  - 当应用程序结构中有大量空间时，仅当消息数据太大而无法存储在结构中时，或者如果它超过某个较低的消息大小阈值，那么应该卸载消息数据，以便将其存储在结构中的性能值不值得其需要的结构空间量。
  - 如果需要特定消息大小阈值，那么通常使用第一个卸载规则来指定该阈值。
- 容量
  - 当应用程序结构中的空间非常小时，应该卸载最大数量的消息数据，以便最佳地利用剩余空间。
  - 第三个卸载规则通常用于指示当结构接近已满时，应该卸载大多数消息，因此应用程序结构中的条目通常具有最小大小 (大约需要 0.75K 字节)。
  - 应该根据应用程序结构大小和最大预期任务列表数来选择使用阈值参数。 例如，如果最大预期待办事项是 1M 消息，那么此数量的消息所需的结构存储量约为 0.75G 字节。 这意味着例如，如果结构约为 10G 字节，那么卸载所有消息的使用阈值必须设置为 92% 或更低。
  - 结构空间分为元素和条目，尽管总体可能有足够的空间，但其中一个空间可能先于另一个空间耗尽。 系统提供 AUTOALTER 功能以在必要时调整比率，但这不是很敏感，因此实际可用的空间量可能略小一些。 因此，最好以不超过最大结构空间的 90% 为目标，因此在前一个示例中，用于卸载所有消息的使用阈值最好设置为 80% 左右。
- 缓冲转换:

- 随着耦合设施结构中剩余空间量的减少，性能特性发生较大突然变化是不可取的。 耦合设施管理在所使用的条目与元素的典型比率中突然发生阈值更改也是不可取的。
- 第二个卸载规则通常用于在性能和容量偏差卸载规则之间提供一些中间缓冲。 当耦合设施结构中使用的空间超过中间阈值时，可以将其设置为导致卸载活动显着增加。 这意味着剩余空间用得更慢，给耦合设施自动改变处理更多时间，以适应更高的使用水平。

如果不能扩展耦合设施结构，并且需要存储至少一些预定数量的消息，那么可以根据需要修改第三规则，以确保所有消息的数据卸载在适当的阈值处开始，以确保为该预定数量的消息保留空间。

例如，如果耦合设施结构大小为 4 GB，并且预先确定的消息数为 1 百万，那么需要 1,000,000 * 0.75 KB，即 768 MB，即 4 GB 的 18.75%。 在这种情况下，需要将卸载所有消息的阈值设置为 80% 左右，而不是 90%。 这将提供参数 OFFLD3TH(80) 和 OFFLD3SZ(0K)。 其他卸载参数也需要调整。

如果发现卸载非常小的消息具有显着的性能影响，但对于较大的消息，相对影响较小，那么可以降低其他规则的使用阈值，以更早卸载更大的消息，从而在结构中为需要卸载之前的小消息留出更多空间。

例如，如果频繁出现超过 32KB 的消息，但卸载这些消息所耗用的时间性能 (根据 RMF 统计信息或应用程序性能确定) 与将它们保留在耦合设施中所耗用的时间性能非常相似，那么第一个规则的阈值可设置为 0% 以卸载所有此类消息。 这将提供参数 OFFLD1TH(0) 和 OFFLD1SZ(32K)。 同样需要调整其他卸载参数。

如果存在许多围绕特定中间大小的消息 (例如 16 KB 和 6 KB)，那么更改第二个规则的消息大小选项可能有用，以便在相当低的使用阈值下卸载较大的消息，从而节省大量空间，但较小的消息仍然只存储在耦合设施中。

z/OS 管理共享消息数据集 *(SMDS)* 环境

如果选择共享消息数据集以卸载大型消息，那么还必须了解 IBM MQ 用于管理这些数据集的信息以及用于处理此信息的命令。 使用本主题来了解如何管理共享消息数据集。

SMDS 对象

在可通过队列共享组中的任何队列管理器更新的共享 SMDS 对象中跟踪每个共享消息数据集的属性和状态。

对于可访问每个耦合设施应用程序结构的每个队列管理器，都有一个共享消息数据集。 共享消息数据集由拥有队列管理器名称 (使用 SMDS 关键字指定) 和应用程序结构名称 (使用 CFSTRUCT 关键字指定) 标识。

注: 为结构定义 SMDS 数据集时，每个队列管理器必须有一个数据集。

SMDS 对象存储在数组中 (组中每个队列管理器有一个条目)，该数组构成 Db2 中存储的相应 CFSTRUCT 对象的扩展。

没有用于 DEFINE 或 DELETE SMDS 对象的命令，因为它是作为 CFSTRUCT 对象的一部分创建或删除的，但有一个用于 ALTER 它以更改单个拥有队列管理器的设置的命令。

有关 SMDS 命令的更多信息，请参阅 第 151 页的『SMDS 相关命令』

SMDSCONN 信息

共享消息数据集可能处于正常状态，但一个或多个队列管理器无法连接到该数据集，例如，由于安全性定义或直接访问设备连接存在问题。 因此，每个队列管理器都必须跟踪每个共享消息数据集的连接状态和可用性信息，例如，指示当前是否可以连接到该数据集，如果不是，那么说明原因。

SSMDSCONN 信息表示与共享消息数据集的队列管理器连接。 对于共享消息数据集本身，它由队列管理器标识，该队列管理器拥有与 CFSTRUCT 名称组合的共享消息数据集 (在共享对象本身的 SMDS 关键字上指定)。

没有用于标识连接队列管理器的参数，因为针对特定队列管理器的命令只能引用同一队列管理器的 SMDSCONN 信息。

SMDSCONN 信息条目保留在拥有队列管理器的主存储器中，并在重新启动队列管理器时重新创建。 但是，如果来自单个队列管理器的连接已显式停止，那么此信息也会作为标志存储在相应的 CFSTRUCT 或 SMDS 对象中的连接数组中，以便它在队列管理器重新启动期间持久存在。

状态和可用性信息

状态信息指示资源或连接的状态 (例如，是否尚未使用，正在正常使用或需要恢复)。 通常使用 STATUS 关键字进行描述。 可能的值取决于对象的类型。

通常会自动更新状态信息，例如，在使用资源或连接时检测到错误时。 但是，在某些情况下，还可以使用命令来更新状态，以允许队列管理器无法自动确定正确状态的情况。

可用性信息指示是否可以使用资源或连接，通常主要由状态信息确定。 对于共享消息数据集支持中使用的资源或连接类型，将实现三个级别的可用性:

可用
这意味着该资源可供正常使用。 这并不一定意味着它目前正在使用中 (可以根据 STATUS 值来确定)。 对于数据集，如果它需要重新启动处理，那么这将允许拥有队列管理器打开该数据集，但其他队列管理器必须等到该数据集重新处于 ACTIVE 状态。

由于错误而不可用
这意味着由于错误，资源已自动变为不可用，并且在执行某种形式的修复或恢复处理之前，预期不会再次可用。 但是，允许在没有操作员干预的情况下再次尝试使其可用。 此尝试也可以由用于将资源标记为 "已启用" 的命令或用于以指示恢复处理已完成的方式更改状态的命令触发。

使资源不可用的原因通常从相关 STATUS 值可见，但在某些情况下可能有其他原因使资源不可用，在这种情况下，会提供单独的 REASON 值来指示原因。

由于操作员命令而不可用
这意味着命令已显式禁用对资源的访问。 只能通过使用命令再次将其启用来使其可用。

SMDS 可用性

对于共享 SMDS 对象，可用性由 ACCESS 关键字描述，可能的值为 ENABLED， SUSPENDED 和 DISABLED。

可以对组中的任何队列管理器中的相关共享对象使用 **RESET SMDS** 命令来更新可用性，以设置 ACCESS (ENABLED) 或 ACCESS (DISABLED)。

如果可用性先前为 ACCESS (SUSPENDED)，那么将其更改为 ACCESS (ENABLED) 将触发新的尝试使用共享消息数据集，但如果先前的错误仍然存在，那么可用性将重置回 ACCESS (SUSPENDED)。

SMDSCONN 可用性

对于本地 SMDSCONN 信息条目，可用性由 AVAIL 关键字描述，可能的值为 NORMAL， ERROR 或 STOPPED。 可以使用针对特定队列管理器的 **START SMDSCONN** 或 **STOP SMDSCONN** 命令来更新可用性，以启用或禁用其连接。

如果可用性先前为 AVAIL (ERROR)，那么将其更改为 AVAIL (NORMAL) 将触发新的尝试使用共享消息数据集，但如果先前的错误仍然存在，那么可用性将重置为 AVAIL (ERROR)。

共享消息数据集共享状态和可用性

使用共享状态信息在组中管理每个共享消息数据集的可用性，可使用带有 TYPE (SMDS) 的 **DISPLAY CFSTATUS** 命令来显示这些信息。 这将显示已为每个结构激活数据集的每个队列管理器的状态信息。 每个数据集可以处于下列其中一种状态:

找不到
这意味着尚未激活相应的数据集。 仅当指定了特定队列管理器时，才会显示此状态，因为当选择了所有队列管理器时，将跳过尚未激活的数据集。

新建
正在首次打开并初始化数据集，准备使其处于活动状态。

活动的
这意味着数据集完全可用，应该由结构的所有活动队列管理器分配和打开。

失败
这意味着数据集完全不可用 (恢复处理除外)，并且必须由所有队列管理器关闭和取消分配。

恢复中
这意味着正在对此数据集进行介质恢复 (使用 RECOVER CFSTRUCT)。

已恢复
这指示已发出命令以将失败的数据集切换回活动状态，但需要进一步重新启动处理 (尚未完成) ，因此只有拥有此数据集的队列管理器才能打开此数据集以进行重新启动处理。

EMPTY
数据集不包含任何消息。 如果拥有队列管理器在数据集不包含任何消息时正常关闭该数据集，那么该数据集将处于此状态。 当要废弃先前数据集内容时，也可以将其置于 EMPTY 状态，因为应用程序结构已清空 (使用带有 TYPE PURGE 的 **RECOVER CFSTRUCT** ，或者仅对于不可恢复的结构，通过删除该结构的先前实例)。 下次数据集由其拥有的队列管理器打开时，空间映射将重置为空，并且状态将更改为 ACTIVE。 由于不再需要先前的数据集内容，因此可以将处于此状态的数据集替换为新分配的数据集，例如，更改空间分配或将其移动到另一个卷。

命令输出包括启用恢复日志记录的日期和时间 (如果有) 以及数据集失败的日期和时间 (如果当前未处于活动状态)。

可以通过 **RESET SMDS** 命令将共享消息数据集置于 FAILED 状态，也可以在检测到以下任何类型的错误时自动将其置于 FAILED 状态:

- 拥有的队列管理器无法分配或打开数据集。
- 在任何队列管理器成功打开数据集头之后，验证该数据集头失败。
- 当拥有队列管理器正在读取或写入数据时，将发生永久 I/O 错误。
- 当另一个队列管理器从已成功完成打开处理和验证的数据集中读取数据时，将发生永久 I/O 错误。

当数据集处于 FAILED 或 INRECOVER 状态时，它不可用于正常使用，因此如果可用性状态为 ACCESS (ENABLED) ，那么它将更改为 ACCESS (SUSPENDED)。

如果数据集已进入 FAILED 状态，但不需要介质恢复 (例如，由于数据仍然有效但存储设备暂时脱机) ，那么可以使用 **RESET SMDS** 命令来请求将状态直接更改为 RECOVERY 状态。

当数据集在完成恢复处理时或由于 **RESET SMDS** 命令而进入 RECOVERY 状态时，那么在完成重新启动处理后，即可再次使用该数据集。 如果它处于 ACCESS (SUSPENDED) 状态，那么它将自动切换回 ACCESS (ENABLED) 状态，这允许拥有队列管理器执行重新启动处理。 重新启动处理完成时，状态将更改为 ACTIVE ，然后所有其他队列管理器都可以再次连接到数据集。

共享消息数据集连接状态和可用性

每个队列管理器维护其与自身和组中其他队列管理器所拥有的每个共享消息数据集的连接的本地状态和可用性信息。 可以使用 **DISPLAY SMDSCONN** 命令显示此信息。

如果它无法访问处于 ACTIVE 状态的共享消息数据集 (属于另一个队列管理器) ，那么它会从其自己的角度将连接标记为不可用。

如果该错误明确指示数据集本身存在问题，那么队列管理器还会自动更改共享状态以指示数据集现在处于 FAILED 状态。 但是，如果该错误可能是由环境问题 (例如，未授权打开数据集) 引起的，那么队列管理器会发出错误消息并将数据集视为不可用，但不会修改共享数据集状态。 如果环境错误仍然是数据集的问题 (例如，它已分配在某些队列管理器无法访问的设备上) ，那么操作员可以使用指定 STATUS (FAILED) 的 RESET SMDS 命令来允许根据需要恢复或修复数据集。

如果无法建立与共享消息数据集的连接，但该数据集似乎有效，那么可以通过对拥有的队列管理器发出 **START SMDSCONN** 命令来触发使用该数据集的新尝试。

如果操作需要临时终止特定队列管理器与数据集之间的连接，但数据集本身未损坏，那么可以使用 **STOP SMDSCONN** 命令来关闭和取消分配数据集。 如果数据集正在使用中，那么队列管理器将正常关闭该数据集 (尽管该数据集中的任何数据请求都将被拒绝，并带有返回码)。 如果它是拥有的数据集，那么队列管理器将在 CLOSE 处理期间保存空间映射，从而避免需要重新启动处理。

如果需要从所有队列管理器中临时取出数据集 (例如移动该数据集) 但未损坏，那么最好将 **STOP SMDSCONN** 用于具有选项 CMDSCOPE (*) 的相关数据集，以首先停止使用该数据集的队列管理器，因为这将避免在数据集重新投入服务时需要重新启动处理。 相反，如果将数据集标记为 FAILED ，那么这将告知队列管理器必须立即停止使用该数据集，这意味着不会保存空间映射，并且需要通过重新启动处理重建空间映射。

如果重新启动队列管理器，那么将重试对先前处于 ACCESS (SUSPENDED) 状态的任何共享消息数据集的访问。

共享消息数据集恢复日志记录

记录持久共享消息以进行介质恢复。 这意味着可以在耦合设施结构或共享消息数据集发生任何故障后恢复消息，前提是恢复日志仍然完整。 出于灾难恢复目的，还可以从另一个站点的恢复日志中重新创建持久消息。

将消息数据写入共享消息数据集时，将单独记录写入数据集的每个块，后跟写入耦合设施的消息条目 (包括数据映射)。 恢复过程始终恢复耦合设施结构，但不需要恢复单个共享消息数据集，除非数据集状态为 FAILED，或者状态为 ACTIVE，但数据集头记录不再有效，这指示数据集已重新创建。 如果数据集的状态为 ACTIVE 并且数据集头仍然有效，或者如果数据集的状态为 EMPTY，那么不会选择恢复数据集，这指示发生故障时未在其中存储任何消息。

共享消息数据集备份

当使用 BACKUP CFSTRUCT 来备份应用程序结构中的共享消息时，将同时备份存储在共享消息数据集中的持久消息的任何数据，与先前存储在数据库中的持久共享消息一样。

共享消息数据集恢复

如果共享消息数据集已损坏或丢失，那么需要将其置于 FAILED 状态以阻止队列管理器使用该数据集，直到修复该数据集为止。 这通常会自动发生，但也可以使用指定 STATUS (FAILED) 的 **RESET SMDS** 命令来完成。

如果共享消息数据集包含任何持久消息，那么可以使用 RECOVER CFSTRUCT 命令来恢复这些消息。 此命令首先从最近的 BACKUP CFSTRUCT 命令复原该共享消息数据集的任何持久消息数据，然后应用自该时间以来记录的所有更改。 如果自首次激活数据集以来未执行任何 **BACKUP CFSTRUCT** 命令，那么会将其重置为空，然后应用自激活以来的所有更改。

如果 CFSTRUCT 内容和所有共享消息数据集都不可用 (例如，在灾难恢复情况下)，那么它们都可以通过单个 **RECOVER CFSTRUCT** 命令进行恢复。

如果共享消息数据集已损坏，但 CFSTRUCT 的恢复未处于活动状态，或者包含最新 BACKUP CFSTRUCT 的日志不可用或不可用，那么无法恢复卸载到该数据集的消息。 在这种情况下，可以使用带有参数 TYPE (PURGE) 的 **RECOVER CFSTRUCT** 命令将共享消息数据集标记为空，并从该数据集中存储数据的结构中删除任何消息。

发出 **RECOVER CFSTRUCT** 命令时，共享消息数据集状态将从 FAILED 更改为 INRECOVER。 如果恢复成功完成，那么状态会自动更改为 RECOVERY，否则会更改回 FAILED。

当数据集更改为恢复状态时，这将告知拥有队列管理器它现在可以尝试打开数据集并执行重新启动处理。

共享消息数据集恢复和同步点

共享消息数据集恢复过程将重新应用所有完整日志记录的更改，直到日志结束，而不考虑同步点。

如果在同步点内进行了更改，那么针对 CFSTRUCT 的重新启动或恢复处理可能会导致回退未落实的请求，因此某些已恢复的更改可能实际未使用，但无论如何都不会对恢复这些更改造成任何影响。

还可能是未落实的 MQPUT 消息可能已写入结构，但相应数据可能未写入数据集或日志 (因为只有在同步点处理开始时才会强制完成 I/O)。 这是无害的，因为重新启动处理将回退结构中的消息条目，因此它引用未恢复的数据这一事实无关紧要。

共享消息数据集重新启动处理

如果与 CFSTRUCT 的队列管理器连接正常终止，那么在数据集关闭之前，队列管理器会将每个共享消息数据集的可用块空间映射写入数据集内的检查点区域。 然后，可以在连接重新启动时再次读取空间映射，前提是 CFSTRUCT 和共享消息数据集都不需要在下一次重新启动之前进行任何恢复处理。

但是，如果队列管理器异常终止，或者结构或数据集需要任何恢复处理，那么在重新启动与该结构的队列管理器连接时，需要执行其他处理以动态重建空间映射。

如果数据集本身不需要恢复，那么队列管理器重新启动只需扫描结构的当前内容以查找对当前队列管理器拥有的消息数据的引用，并将相关数据块标记为空间映射中拥有的数据块。 在重建空间映射时，其他队列管理器可以继续使用该结构并读取重新启动队列管理器所拥有的数据。

恢复后重新启动共享消息数据集

如果必须从备份恢复共享消息数据集，那么存储在数据集中的所有非持久消息都将丢失，如果使用 TYPE (PURGE) 恢复数据集，那么存储在数据集中的所有消息都将丢失。 在恢复完成之前，数据集将标记为 FAILED 或 INRECOVER，因此任何尝试从另一个队列管理器读取其中一条受影响的消息都会返回错误代码，指示数据集暂时不可用。

当数据集已恢复时，状态将更改为已恢复，这允许拥有的队列管理器打开该数据集以进行重新启动处理，但该数据集仍对其他队列管理器不可用。 队列管理器重新启动会扫描结构以重新构建空间映射以获取任何剩余消息。 扫描还会检查已丢失数据的消息，并从结构中删除这些消息 (如果需要，将其标记为 "已丢失"，稍后将其删除)。

当此重新启动扫描完成时，数据集状态将自动从 "已恢复" 更改为 "活动"，此时其他队列管理器可以重新开始使用该数据集。

共享消息数据集使用信息

DISPLAY USAGE 命令现在还显示有关任何当前打开的共享消息数据集的共享消息数据集空间和缓冲池使用情况的信息。 如果指定了新选项 TYPE (SMDS) 或现有选项 TYPE (ALL)，那么将显示此信息。

共享消息数据性能和容量注意事项

监视数据集使用情况

DISPLAY USAGE 命令可以使用选项 **TYPE(SMDS)** 显示每个拥有的共享消息数据集的当前已满百分比。

当共享消息数据集达到 90% 满时，队列管理器通常会自动展开共享消息数据集，前提是选项 **DSEXPAND(YES)** 对 SMDS 定义有效。 当 SMDS 选项设置为 **DSEXPAND(YES)** 或 SMDS 选项设置为 **DSEXPAND(DEFAULT)** 且 CFSTRUCT 缺省选项设置为 **DSEXPAND(YES)** 时，这将适用。

如果由于创建数据集时未指定辅助分配大小而导致扩展尝试失败 (给出消息 IEC070I，原因码为 203) 队列管理器使用大约为当前大小的 20% 的覆盖辅助分配来重复扩展请求。

扩展数据集时，新的数据集扩展数据块将作为扩展处理的一部分进行格式化，对于非常大的扩展数据块，这可能需要数十秒甚至几分钟。 在格式化完成后，新空间可供使用，并且目录已更新为显示新的高使用率控制区间。

如果正在非常迅速地创建新消息，那么在扩展处理完成之前，现有数据集可能会变满。 在这种情况下，无法分配空间的任何请求都将临时暂挂，直到扩展尝试完成并且新空间可供使用为止。 如果扩展成功，那么将自动重试该请求。

如果由于缺少可用空间或由于已达到最大扩展数据块而导致扩展尝试失败，那么将发出一条消息，说明失败原因，然后将受影响的 SMDS 的覆盖选项自动更改为 **DSEXPAND(NO)** 以防止进一步的扩展尝试。 在这种情况下，存在数据集变满的风险，在这种情况下可能需要进一步的操作，如 数据集变满中所述。

监视应用程序结构使用情况

可以使用 MVS **DISPLAY XCF,STRUCTURE** 命令显示应用程序结构的使用级别，该命令指定应用程序结构的全名 (包括队列共享组前缀)。 IXC360I 响应消息显示元素和条目的当前使用情况。

当结构使用情况超过 CFRM 策略中指定的 **FULLTHRESHOLD** 值时，系统将发出消息 IXC585E，并可能执行自动 **ALTER** 操作 (如果指定)，这可能会改变条目与元素的比率或增大结构大小。

优化缓冲池大小

共享缓冲池中的每个缓冲区都用于读取或写入多达逻辑块大小的一条消息的连续页范围。 如果消息溢出到更多块中，那么单独块中的每个页面范围都需要单独的缓冲区。

在写或读操作之后包含消息数据的缓冲区将保留在存储器中，并使用最近最少使用的 (LRU) 高速缓存方案进行复用，以便稍后再次读取相同数据的请求不需要转至磁盘。 当在同一系统上运行的应用程序写入共享消息并在不久后读回共享消息时，这将提供重大优化。 如果出于选择目的浏览了另一个队列管理器拥有的消息，然后进行了检索，那么这也避免了从磁盘重新读取消息的需要。

这意味着每个应用程序结构所需的缓冲区数是每个并发 API 请求的一个，该请求为该应用程序结构读取或写入大量消息，外加一些额外的缓冲区，这些缓冲区将用于保存最近访问的数据，以优化后续读访问。

对于共享缓冲池，如果没有足够的缓冲区，那么如果没有立即可用的缓冲区，那么 API 请求将直接等待。 但是，应该避免这种情况，因为它可能导致性能显着下降。

来自共享缓冲池的 **DISPLAY USAGE** 命令的统计信息显示在当前统计信息时间间隔内是否有任何缓冲区等待，并且还显示最小可用缓冲区数 (或负值表示随时等待缓冲区的最大线程数) ，已保存数据的缓冲区数以及缓冲区请求在 LRU 链上成功找到已保存数据的次数的百分比 (“LRU 命中数”) 而不是必须读取它 (“LRU 未命中”)[1].

- 如果存在任何等待，那么应该增加缓冲区数。
- 如果有许多未使用的缓冲区，那么缓冲区的数目可能会减少，以便在区域中有更多的存储空间用于其他用途。
- 如果有许多缓冲区包含已保存的数据，但对该已保存的数据执行的读取操作所占的比例非常小，那么如果存储器可以更好地用于其他用途，那么缓冲区数量可能会减少。 但是，缓冲区的数目不应减少超过可用缓冲区的最低数目，因为这可能会触发等待，并且最好是足够高，使最低可用缓冲区计数通常远高于零。

删除共享消息数据集

DELETE CFSTRUCT 命令 (仅当结构中的所有共享队列都为空且已关闭时才允许) 不会删除共享消息数据集本身，但可以在此命令完成后以通常的方式将其删除。 如果要将同一数据集复用为共享消息数据集，那么必须首先对其进行重新格式化以将其重置为空状态。

共享消息数据集的异常情况

在正常使用期间可能会发生一些异常情况，即使不存在软件或硬件错误也是如此。

数据集变满

如果数据集变满但无法扩展，或者扩展尝试失败，那么使用相应队列管理器将大消息写入相应应用程序结构的应用程序将收到错误 2192，即 MQRC_STORAGE_MEDIUM_FULL (也称为 MQRC_PAGESET_FULL)。

由于应该处理数据的应用程序中发生故障，导致大量积压消息累积，因此数据集可能变满。 如果是这样，进一步扩展数据集只会是一个临时解决方案，让处理应用程序尽快再次运行很重要。

如果可以提供更多空间，那么可以使用 **ALTER SMDS** 命令来设置 **DSEXPAND(YES)** 或 **DSEXPAND(DEFAULT)** (假定已设置 YES 或假定为 CFSTRUCT 定义的 **DSEXPAND** 缺省值) 以触发重试。 但是，如果失败的原因是已达到最大扩展数据块数，那么将拒绝新的扩展尝试并显示一条消息，并且将再次设置 **DSEXPAND(NO)** 。 在这种情况下，进一步扩展它的唯一方法是重新分配它，这涉及使它暂时不可用，如下所述。

需要移动或重新分配数据集

如果需要移动或扩展数据集但在正常使用中，那么可以临时将其取出以允许移动或重新分配数据集。 尝试在数据集不可用时使用该数据集的任何 API 请求都将接收到原因码 MQRC_DATA_SET_NOT_AVAILABLE 。

1. 使用 **RESET SMDS** 命令将数据集标记为 **ACCESS(DISABLED)**。 这将导致它正常关闭并由所有当前连接的队列管理器取消分配。
2. 根据需要移动或重新分配数据集，将旧内容复制到新分配的数据集，例如使用 "访问方法服务" (AMS) **REPRO** 命令。

   在将旧数据复制到新数据集之前，请勿尝试对新数据集进行预格式化，因为这将导致将复制的数据追加到格式化数据集的末尾。
3. 使用 **RESET SMDS** 命令再次将数据集标记为 **ACCESS(ENABLED)** ，以使其重新投入使用。

[1] (Hits / (Hits+Misses))* 100

如果旧内容小于新数据集的大小，那么打开新数据集时将自动对其余空间进行预格式化。

如果旧内容大于新数据集的大小，那么队列管理器必须扫描耦合设施结构中的消息并重建空间映射，以确保没有任何活动数据丢失。 如果找到对位于新扩展数据块外部的数据块的任何引用，那么该数据集将标记为 **STATUS(FAILED)**，并且必须通过将该数据集替换为正确的大小之一并将旧数据集再次复制到该数据块中或者使用 **RECOVER CFSTRUCT** 来恢复任何持久消息来进行修复。

耦合设施结构空间小

如果耦合设施结构空间不足，导致消息 IXC585E，那么 值得检查是否已设置卸载规则以确保在此情况下卸载最大数据量。 否则，可以使用 **ALTER CFSTRUCT** 命令来修改卸载规则。

共享消息数据集的错误情境

有一些问题需要注意，这些问题只能由错误引起，不能在正常的操作情况下发生。

无法打开拥有的数据集

如果拥有共享消息数据集的队列管理器无法将其分配或打开，或者数据集属性不受支持，那么队列管理器将设置相应的 **SMDSCONN** 状态值 **ALLOCFAIL** 或 **OPENFAIL**，并将 **SMDSCONN** 可用性设置为 **AVAIL(ERROR)**。 它还将 SMDS 可用性设置为 **ACCESS(SUSPENDED)**。 更正错误后，使用 **RESET SMDS** 命令设置 **ACCESS(ENABLED)** 以触发重试，或者向拥有队列管理器发出 **START SMDSCONN** 命令。

无法打开只读数据集

如果队列管理器无法分配或打开另一个队列管理器拥有并标记为 **STATUS(ACTIVE)**的共享消息数据集，那么它假定这可能是由于其与数据集 (由 **SMDSCONN** 对象表示) 的连接存在特定问题，而不是数据集本身存在问题。

它将 **SMDSCONN** 标记为 **STATUS(ALLOCFAIL)** 或 **STATUS(OPENFAIL)** (视情况而定)，并将 **SMDSCONN** 可用性标记为 AVAIL (ERROR)，以防止进一步尝试使用它。

如果可以在不影响数据集本身状态的情况下更正问题，请使用 **START SMDSCONN** 命令来触发重试。

如果问题证明是数据集本身的问题，那么可以使用 **RESET SMDS** 命令将数据集标记为 **STATUS(FAILED)**，直到它已恢复为止。 恢复数据集后，将状态更改回 **STATUS(ACTIVE)** 的操作将导致通知其他队列管理器。 如果 **SMDSCONN** 标记为 **AVAIL(ERROR)**，那么它将自动更改回 **AVAIL(NORMAL)** 以触发打开数据集的新尝试。

数据集头已损坏

如果已成功打开数据集，但头信息的格式不正确，那么队列管理器将关闭并取消分配数据集，并将状态集设置为 **STATUS(FAILED)**，将可用性设置为 **ACCESS(SUSPENDED)**。 这允许使用 **RECOVER CFSTRUCT** 来恢复内容。

如果由于数据集包含来自另一个用途的残差数据并且未随后进行预格式化而产生错误，请对数据集进行预格式化，并使用 **RESET SMDS** 命令将状态更改为 **STATUS(RECOVERED)**。

否则，必须恢复数据集。

数据集意外为空

如果队列管理器打开标记为 **STATUS(ACTIVE)** 的数据集，但发现该数据集未初始化或新预格式化但以其他方式有效，那么队列管理器将关闭并取消分配共享消息数据集，然后将状态设置为 **STATUS(FAILED)**，并将可用性设置为 **ACCESS(SUSPENDED)**。

数据集存在永久 I/O 错误

如果数据集在成功 **OPEN** 处理后发生永久 I/O 错误，那么它可能需要恢复。 队列管理器会将数据集标记为 **STATUS(FAILED)**，以便所有当前连接的队列管理器都将关闭并取消分配该数据集。

数据集有可恢复的 I/O 错误

如果数据集存在硬件问题，那么这可能导致可恢复 I/O 错误，这些错误不会反映回队列管理器，但会导致性能显着下降，并且还会指示在不久的将来发生永久 I/O 错误的风险。

在这种情况下，可以通过使用 **RESET SMDS** 命令将数据集标记为 **STATUS(FAILED)** 来使其脱机以进行恢复。 这将导致所有队列管理器将其关闭并取消分配，例如，可以将其移至新卷，然后再使其再次可用。

以这种方式使数据集不可用时，不会保存空间映射，因此队列管理器连接重新启动处理将需要扫描耦合设施结构以查找数据集中的消息并重建空间映射，然后才能使数据集再次可用。 作为替代方法，如果共享消息数据集仍然可用，那么可以通过使用 **RESET SMDS** 命令将数据集标记为 **ACCESS(DISABLED)**，使其更轻轻地不可用，直到它准备好再次可用为止。

数据集内容不正确

队列管理器无法直接检测到数据集包含不正确的数据或不是最新的数据，例如，因为必须从备份复原包含该数据集的卷。 但是，它会执行完整性检查，这使得任何此类错误都不太可能导致应用程序看到不正确的消息数据。

为了完整性检查目的，在将消息数据传递给用户程序之前，数据集中的每个消息块都以相应的耦合设施条目标识的副本作为前缀，包括唯一的时间戳记，每当读取消息块时都会检查该时间戳记。 如果消息块前缀与条目标识不匹配 (并且在平均时间内未删除耦合设施条目)，那么假定消息块已损坏且不可用。

如果损坏的消息是持久的，那么数据集将标记为 **STATUS(FAILED)**，并且必须使用 **RECOVER CFSTRUCT** 命令来恢复结构内容。 如果损坏的消息是非持久消息，那么无法恢复该消息，因此将发出诊断消息并删除相应的耦合设施消息条目。

如果打开数据集时没有可用的已保存空间映射，那么将通过扫描耦合设施结构以获取对数据集中数据的引用来重建该映射。 在此扫描期间，队列管理器将执行多项操作:

1. 队列管理器确定当前保留在数据集中的最新消息 (如果有) 的位置。
2. 然后，队列管理器从数据集读取该消息，以确保块前缀与消息条目标识匹配

这些操作确保队列管理器检测数据集处于较低级别的任何情况，并将数据集标记为 FAILED。 但是，如果数据集是从先前副本复原的，并且自那时以来未添加任何新消息，或者自该副本之后添加的所有消息都已被读取和删除，那么此检查将容忍这种情况。

在数据集正常关闭的情况下，为了防止下级数据，队列管理器执行许多操作:

1. 当数据集正常关闭时，队列管理器会将空间映射时间戳记的副本保存在 Db2 内的 SMDS 对象中。
2. 然后，当再次打开数据集时，队列管理器会检查空间映射时间戳记是否相同

如果时间戳记不匹配，那么这表明可能已使用数据集的下级副本，因此队列管理器将忽略现有空间映射并对其进行重建，这将仅在未实际丢失消息数据的情况下成功。

注: 这些完整性检查不保证在所有理论上可能的情况下检测到低级别或损坏的数据集。 例如，如果消息块的启动有效，但其余数据已被部分覆盖，那么它们将不会检测到这种情况。

共享消息数据集的恢复方案

本部分描述了共享消息数据集恢复方案。

未丢失任何数据的数据集恢复

在某些情况下，可以复原失败数据集的正确内容，而无需实际恢复。 一个示例是，数据集包含来自先前使用的残差数据，并且未再次进行预格式化，可以通过对其进行预格式化来进行修正。 另一种情况是移动了数据集，但在复制数据的过程中发生了错误，可以通过正确再次复制数据来修正此错误。

在这种情况下，可以使用 **RESET SMDS** 命令设置 **STATUS(RECOVERED)** 来再次提供更正后的数据集。 如果可用性当前为 **ACCESS(SUSPENDED)**，那么将自动将其设置回 **ACCESS(ENABLED)**。

当通知拥有队列管理器已恢复数据集时，它会扫描结构内容以重构空间映射，然后将状态更改为 **STATUS(ACTIVE)**。 然后，其他队列管理器可以再次开始读取数据集。

具有 TYPE (NORMAL) 的数据集恢复

如果数据集的内容已丢失，但应用程序结构是使用 **RECOVER(YES)** 定义的，并且相应的恢复日志可用，那么可以使用 **RECOVER CFSTRUCT** 命令来恢复存储在该结构中的任何持久消息，包括卸载到共享消息数据集的持久消息数据。 此命令使用 **BACKUP CFSTRUCT** 命令记录的信息以及自备份时间以来记录的对持久消息的所有更改来复原当前状态。

RECOVER CFSTRUCT 命令始终将耦合设施结构中的所有持久消息与存储在 Db2 中的卸载消息数据一起恢复。 对于存储在共享消息数据集中的卸载数据，仅当每个数据集已标记为 **STATUS(FAILED)** 时，或者在恢复处理打开时发现它意外为空或无效时，才会选择对其进行恢复处理。 标记为活动且通过验证检查的任何共享消息数据集都不需要恢复，因为现有消息数据已正确，但头已更新以指示在恢复后将需要重建任何已保存的空间映射。

仅当结构已标记为失败时，才可以进行恢复处理，因为需要通过恢复处理来重构结构的完整内容。 但是，如果至少一个共享消息数据集已标记为失败，那么 **RECOVER CFSTRUCT** 命令将在必要时自动将结构标记为失败，以允许继续进行恢复处理。

可以从队列共享组中的任何队列管理器执行恢复，前提是已授予该队列管理器对相关数据集的写访问权。

仅备份和记录持久消息，因此正常恢复处理将复原所有持久消息，但将导致结构中的任何非持久消息丢失。

恢复完成后，为恢复选择的任何数据集将自动更改为 **STATUS(RECOVERED)**，如果可用性为 **ACCESS(SUSPENDED)**，那么将更改为 **ACCESS(ENABLED)**。 队列管理器通过扫描耦合设施中的消息来重建每个数据集的空间映射，然后将数据集标记为 **STATUS(ACTIVE)**，以便可以再次使用该数据集。

使用 TYPE (PURGE) 进行数据集恢复

对于可恢复结构，如果数据集内容已丢失，但由于某种原因无法恢复，例如，由于恢复日志不可用或恢复需要太长时间，那么可以将 **RECOVER CFSTRUCT** 命令与 **TYPE(PURGE)** 配合使用以将结构恢复到可用状态。 这会将结构重置为空状态，并将所有关联的数据集标记为 **STATUS(EMPTY)**。

删除应用程序结构

如果使用 MVS **SETXCF FORCE** 命令删除了不可恢复的应用程序结构，或者由于结构故障，那么在下次连接该结构时，将发出消息 CSQE028I 以声明已重置该结构，并且已废弃所有现有消息，并且任何现有数据集也将自动重置为 **STATUS(EMPTY)**。 此操作将使不可恢复的结构在该结构或任何关联数据集中丢失数据后再次可用。

如果删除了可恢复的应用程序结构，那么将以与该结构发生故障相同的方式处理该结构。

数据集恢复失败

如果 **RECOVER CFSTRUCT** 由于某些原因 (例如，由于日志数据集不再可用，或者由于正在进行恢复时队列管理器已终止) 而无法完成，那么将在头中标记至少已启动恢复的任何数据集，以显示已尝试部分恢复，并且该数据集将保持 **STATUS(FAILED)** 状态。

在这种情况下，选项是重复原始恢复请求，或者改为使用 **TYPE(PURGE)** 进行恢复，从而废弃现有数据。

如果尝试将数据集标记为 **STATUS(RECOVERED)** 而未实际恢复该数据集，那么下次打开该数据集时，队列管理器将看到该头指示未完成恢复，并将其再次标记为 **STATUS(FAILED)**。

场外灾难恢复

对于非现场灾难恢复，只能使用包含 CFSTRUCT 定义和关联 SMDS 状态信息的日志和 Db2 共享对象来重新创建持久共享消息。

设置包含定义的 Db2 表后，可以将应用程序结构和共享消息数据集设置为空。 当队列管理器连接到它们并发现它们意外为空时，它会将它们标记为失败，之后可以使用单个 **RECOVER CFSTRUCT** 命令来恢复所有受影响结构的所有持久消息。

z/OS *SMDS* 相关命令

本主题描述并提供对与共享消息数据集相关的命令的访问权。

显示和更改与大型消息卸载 (**OFFLOAD** 和卸载规则) 和共享消息数据集 (**DSGROUP, DSBLOCK, DSBUFS, DSEXPAND)**) 相关的 **CFSTRUCT** 选项:

- DISPLAY CFSTRUCT
- DEFINE CFSTRUCT

- 变更 CFSTRUCT
- 删除 CFSTRUCT

显示 **CFSTRUCT** 与大型消息卸载相关的状态 **(OFFLDUSE)**:

- 显示 CFSTATUS

显示和变更覆盖数据集选项 (**DSEXPAND** 和 **DSBUFS**) 对于个别队列管理器:

- 显示 SMDS
- 变更 SMDS

显示或修改队列共享组中数据集的状态和可用性:

- 显示 CFSTATUS TYPE (SMDS)
- 重置 SMDS

显示队列管理器的 SMDS 数据集空间使用情况和缓冲区使用情况信息:

- DISPLAY USAGE TYPE (SMDS)

显示或修改连接的状态和可用性 (**SMDSCONN**) 到单个队列管理器中的数据集:

- 显示 SMDSCONN
- 开始 SMDSCONN
- STOP SMDSCONN

备份和恢复共享消息，必要时包括 SMDS 中的大型消息数据:

- 备份 cfstruct
- RECOVER CFSTRUCT

z/OS 使用共享队列的优点

共享队列允许 IBM MQ 应用程序可伸缩，高度可用，并允许实现工作负载均衡。

共享队列的优点

共享队列体系结构 (克隆的服务器从单个共享队列中提取工作) 具有一些有用的属性:

- 通过添加服务器应用程序的新实例，或者甚至添加具有队列管理器 (在队列共享组中) 和应用程序副本的新 z/OS 映像，它是可扩展的。
- 高度可用。
- 它会根据队列共享组中每个队列管理器的可用处理容量自然执行 *pull* 工作负载均衡。

使用共享队列实现高可用性

以下示例说明如何使用共享队列来提高应用程序可用性。

请考虑以下 IBM MQ 场景: 在网络中运行的客户机应用程序想要发出在 z/OS 上运行的服务器应用程序的请求。 客户机应用程序构造请求消息并将其放在请求队列上。 然后，客户机等待来自服务器的应答，该应答发送到请求消息的消息描述符中指定的应答队列。

IBM MQ 管理请求消息从客户机到 z/OS 上的服务器输入队列的传输以及从服务器到客户机的响应的传输。 通过将服务器的输入队列定义为共享队列，可以在队列共享组中的任何队列管理器上检索放入队列的任何消息。 这意味着您可以在综合系统中的每个 z/OS 映像上配置队列管理器，并且通过将它们全部连接到同一个队列共享组，其中任何一个都可以访问服务器输入队列上的消息。

服务器的输入队列上的消息仍可用，即使其中一个队列管理器异常终止或由于管理原因必须将其停止也是如此。 您可以使整个 z/OS 映像脱机，并且消息仍可用。

要利用共享队列上消息的此可用性，请在综合系统中的每个 z/OS 映像上运行服务器应用程序实例，以提供更高的服务器应用程序容量和可用性，如 第 153 页的图 60 中所示。

服务器应用程序的一个实例从共享队列中检索请求消息，并根据内容执行其处理，生成作为 IBM MQ 消息发送回客户机的结果。 响应消息指定给在请求消息的消息描述符中指定的应答队列和应答队列管理器。

有许多选项可用于配置返回路径。 有关这些选项的更多信息，请参阅 第 169 页的『分布式排队和队列共享组』。

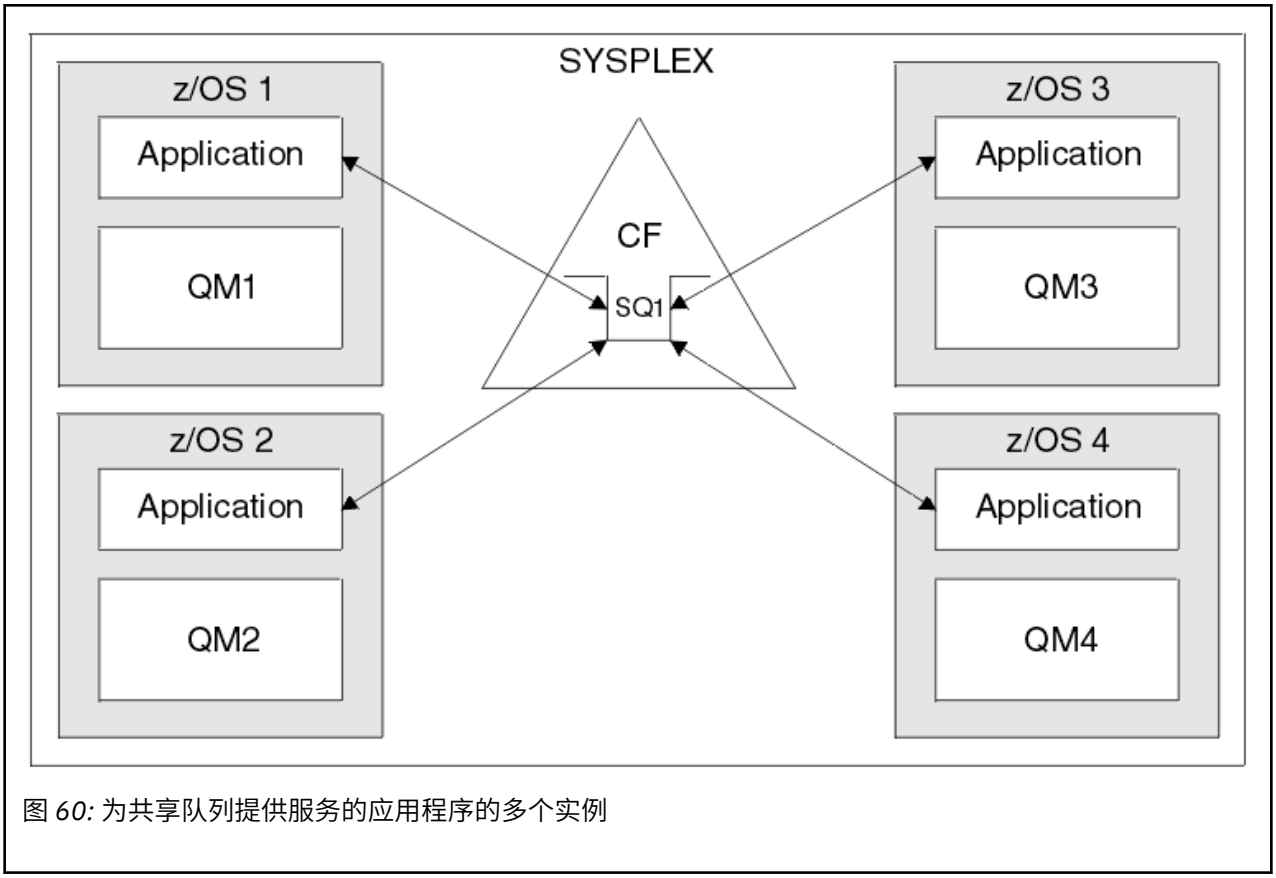


图 *60:* 为共享队列提供服务的应用程序的多个实例

对等恢复

为了进一步增强队列共享组中消息的可用性， IBM MQ 会检测组中的另一个队列管理器是否与耦合设施异常断开连接，并在可能的情况下完成仍处于暂挂状态的该队列管理器的工作单元。 此功能称为 对等恢复 。

假设队列管理器在应用程序已从同步点中的队列检索到请求消息但尚未放入响应消息或已落实工作单元的位置异常终止。 队列共享组中的另一个队列管理器检测到故障，并回退正在失败队列管理器上执行的未完成工作单元。 这意味着将请求消息放回到请求队列中，并可供其他某个服务器实例处理，而无需等待失败的队列管理器重新启动。

如果 IBM MQ 无法自动解析工作单元，那么您可以手动解析共享部分，以使队列共享组中的另一个队列管理器能够继续处理该工作。

z/OS 将存储类内存与共享队列配合使用

当与 IBM MQ for z/OS 共享队列配合使用时，使用存储类内存 (SCM) 可能是有利的。

要点: IBM z16 计划是 IBM Z ® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

z13， zEC12 和 zBC12 机器允许安装 Flash Express 卡。 这些卡包含闪存固态驱动器 (SSD)。 安装后，可将卡中的闪存分配给通常称为 SCM 的一个或多个 LPAR。

SCM 在 I/O 等待时间和成本方面介于实存储器和直接访问存储设备 (DASD) 之间。 由于 SCM 没有移动部件，因此与 DASD 相比，它的 I/O 延迟要低得多。

SCM 也比实存储器便宜得多。 因此，可以以相对较低的成本安装大量存储器; 例如，一对 Flash Express 卡包含 1424 GB 可用存储器。

这些特征意味着当必须在短时间内从实存储器中获取大量数据时， SCM 很有用，因为该数据可以比写入 DASD 更快地写入 SCM。 使用包含 IBM MQ 共享队列的耦合设施 (CF) 列表结构时，此特定点非常有用。

为什么列表结构会填充

定义 CF 结构时，将使用描述该结构的最大大小的 SIZE 属性对其进行配置。 由于 CF 结构始终永久驻留在实存储器中，因此在 CF 上定义的结构的 SIZE 属性总和应该小于分配给 CF 的实存储器量。

因此，有恒定的压力将任何给定结构的 SIZE 值保持在最小的可能值，以便更多的结构可以拟合到 CF 中。 但是，确保结构足够大以达到其目的可能会产生相互冲突的压力，因为使结构太小意味着它可能被填满，从而干扰使用它的应用程序或子系统。

强烈需要根据其预期使用情况来准确调整结构的大小。 但是，这项任务很难完成，因为工作负载会随着时间的推移而发生变化，并且要考虑它们的波动并不容易。

IBM MQ 共享队列使用 CF 列表结构来存储消息。 IBM MQ 调用 CF 结构，其中包含消息和应用程序结构。

使用存储在 IBM MQ CFSTRUCT 对象中的信息来引用应用程序结构。 将小于 63 KB 的消息放入共享队列时，该消息将完全作为单个列表条目以及零个或多个列表元素存储在应用程序结构中。

由于 IBM MQ 共享队列使用列表结构，因此所描述的压力也会影响共享队列。 在这种情况下，可以存储在共享队列中的最大消息数是下列各项的函数:

- 队列上消息的大小
- 结构的最大大小
- 结构中可用的条目数和元素数

因为多达 512 个共享队列可以使用相同的结构，并有效竞争条目和元素，这使得问题变得更加复杂

IBM MQ 队列用于在应用程序之间传输数据，因此常见情况是应用程序将消息放入队列，而应该获取这些消息的伙伴应用程序未在运行。

发生此情况时，队列上的消息数会随时间推移而增加，直到发生以下一种或多种情况:

- 放置应用程序停止放置消息。
- 获取应用程序开始获取消息。
- 队列上的现有消息将开始到期，并且将从队列中除去。
- 队列达到其最大深度，在此情况下， MQRC_Q_FULL 原因码将返回到放置应用程序。
- 包含共享队列的结构达到其最大大小，或者包含该结构的 CF 耗尽可用存储器。 在任一情况下，都会将 MQRC_STORAGE_MEDIUM_FULL 原因码返回到放置应用程序。

在最后三种情况下，队列已满。 此时，放置应用程序存在问题，因为其消息无处可去。 放置应用程序通常通过使用以下一个或多个解决方案来解决此问题:

- 重复重试放入消息 (可选) ，并在两次重试之间延迟。
- 将消息放在其他位置，例如数据库或文件。 稍后可访问这些消息，并将其正常放入队列中。
- 如果消息是非持久消息，请将其废弃。

但是，对于某些应用程序类 (例如，具有大量入局消息或无法访问文件系统的应用程序类) ，这些解决方案不实用。 确实需要确保队列永远不会或极不可能填满，这与共享队列特别相关。

SMDS 和卸载规则

IBM WebSphere MQ 7.1 中引入的卸载规则提供了一种降低应用程序结构填充可能性的方法。

每个应用程序结构都有三个与之关联的规则，使用三对关键字指定:

- OFFLD1SZ 和 OFFLD1TH
- OFFLD2SZ 和 OFFLD2TH
- OFFLD3SZ 和 OFFLD3TH

每个规则都指定必须满足的条件，才能将消息数据卸载到与应用程序结构关联的存储机制。 当前有两种类型的存储机制可用:

- Db2
- 虚拟存储器访问方法 (VSAM) 线性数据集组， IBM MQ 调用共享消息数据集 (SMDS)。

以下示例显示了 MQSC 命令，该命令使用 DEFINE CFSTRUCT 命令来创建名为 LIST1 的应用程序结构。

此结构具有缺省卸载规则，并使用 SMDS 作为卸载机制。 这意味着当结构满 70% (OFFLD1TH) 时，所有 32 KB 或更大 (OFFLD1SZ) 的消息都将卸载到 SMDS。

同样，当结构为 80% 已满 (OFFLD2TH) 时，将卸载所有 4 KB 或更大 (OFFLD2SZ) 的消息。 当结构为 90% 已满 (OFFLD3TH) 时，将卸载所有消息 (OFFLD3SZ)。

```
DEFINE CFSTRUCT(LIST1)
CFLEVEL(5)
OFFLOAD(SMDS)
OFFLD1SZ(32K) OFFLD1TH(70)
OFFLD2SZ(4K) OFFLD2TH(80)
OFFLD3SZ(0K) OFFLD3TH(90)
```

卸载消息存储在卸载介质中，而指向消息的指针存储在结构中。 虽然卸载规则会减少结构填充的机会，但由于在结构中的消息数据耗尽，因此某些数据仍会写入每个消息的结构中。 即，指向已卸载消息的指针。

此外，卸载规则会带来性能成本。 将消息写入结构比较快，主要由向 CF 发送写入请求所花费的时间支配。 实际写入结构的速度很快，以实际存储速度发生。

将消息写入 SMDS 要慢得多，因为它包括写入消息指针的结构，以及将消息数据写入 SMDS。 第二个写操作以 DASD 速度完成，并有可能增加等待时间。 如果使用 Db2 作为卸载机制，那么性能成本要高得多。

z/OS 存储类内存如何与 *IBM MQ for z/OS* 配合使用

将存储类内存 (SCM) 与 IBM MQ for z/OS 共享队列配合使用的概述。

要点: IBM z16 计划是 IBM Z ® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

处于 CFLEVEL 19 或更高级别的耦合设施 (CF) 可以将 SCM 分配给它。 然后，可以将该 CF 中定义的结构配置为使用 SCM 来减少结构填充的机会 (称为结构完全条件)。 当配置为使用 SCM 的结构在系统确定的点之后填满时， CF 开始将数据从该结构移动到 SCM 中，这将释放该结构中用于新数据的空间。

注: 由于 SCM 本身可以填充，因此将 SCM 分配给结构只会降低结构完全情况的可能性，但不会完全消除发生一次情况的可能性。

通过在耦合设施资源管理器 (CFRM) 策略中同时指定 **SCMALGORITHM** 和 **SCMMAXSIZE** 关键字 (包含该结构的定义) ，将结构配置为使用 SCM。

请注意，在指定这些关键字并应用 CFRM 策略之后，必须重建或取消分配结构以使其生效。

SCMALGORITHM 关键字

由于 SCM 的输入/输出速度比实存储器的输入/输出速度慢，因此 CF 使用一种算法，该算法针对结构的预期使用进行定制，以减少写入 SCM 或从中读取 SCM 的影响。

该算法由结构的 CFRM 策略中的 **SCMALGORITHM** 关键字使用 *KEYPRIORITY1* 值进行配置。 请注意，您应该仅将 *KEYPRIORITY1* 值用于 IBM MQ 共享队列所使用的列表结构。

KEYPRIORITY1 算法的工作原理是假定大多数应用程序将按优先级顺序从共享队列获取消息; 即，当应用程序获取消息时，它将获取具有最高优先级的最旧消息。

当结构开始填充超过系统定义的阈值 90% 时， CF 将开始异步迁移下一个可能性最小的消息。 这些是最近放入队列的优先级较低的消息。

将消息从结构异步迁移到 SCM 称为 "预登台"。

预登台可降低使用 SCM 的性能成本，因为这会降低在对 SCM 进行同步输入/输出期间应用程序被阻塞的机会。

除了预登台之外， *KEYPRIORITY1* 算法还会在有足够可用空间时异步将来自 SCM 的消息返回到结构中。 对于 *KEYPRIORITY1* 算法，这意味着当结构小于或等于满 70% 时。

将来自 SCM 的消息引入到结构中的操作称为 "预取"。

预取降低了应用程序尝试获取已预登台到 SCM 的消息的可能性，并且需要等待 CF 同步将消息返回到结构中。

SCMMAXSIZE 关键字

SCMMAXSIZE 关键字定义可由结构使用的 SCM 的最大数量。 由于需要时 SCM 由 CF 分配给结构，因此可以指定大于可用 SCM 总量的 **SCMMAXSIZE** 。 这被称为 "过度承诺"。

要点: 切勿过度落实 SCM。 如果这样做，依赖它的应用程序将不会获得它们期望的行为。 例如，使用共享队列的 IBM MQ 应用程序可能会获得意外的 MQRC_STORAGE_MEDIUM_FULL 原因码。

CF 使用各种数据结构来跟踪其对 SCM 的使用。 这些数据结构驻留在分配给 CF 的实存储器中，因此会减少结构可使用的实存储器量。 这些数据结构所使用的存储器被称为 "扩充空间"。

使用 SCM 配置结构时，会将少量实存储器从 CF 分配到称为固定扩充空间的结构。 即使结构从未实际使用任何 SCM ，也会分配此值。 随着来自该结构的数据存储到 SCM 中，将从 CF 中的备用实存储器分配额外的动态扩充空间。

从 SCM 中除去数据时，会将动态扩充空间返回到 CF。 扩充空间 (固定或动态) 从不从分配给结构的实存储器中获取。

除了扩充存储器外，当结构配置为使用 SCM 时，该结构使用的控制存储量也会增加。 这意味着在没有配置 SCM 的情况下，使用 SCM 配置的列表结构可以包含比相同大小的结构更少的条目和元素。

要了解 SCM 对新结构或现有结构的影响，请使用 CFSizer 工具。

需要注意的最后一个要点是，在将数据从结构移动到 SCM 中，并且已使用动态扩充空间之后，不能手动或自动更改结构。

即，分配给该结构的存储量不能增加或减少，该结构所使用的入元素比率不能改变，以此类推。 要使结构再次可更改，该结构不得具有任何存储在 SCM 中的数据，并且不得使用动态扩充存储器。

z/OS 为何使用 *SCM*

紧急存储和提高性能是将 SCM 与 IBM MQ for z/OS 配合使用的两个用例。

要点: IBM z16 计划是 IBM Z ® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

本部分介绍了两种可能的场景背后的理论。 有关如何设置方案的更多详细信息，请参阅:

- 第 159 页的『紧急存储器-基本配置』
- 第 164 页的『改进的性能-基本配置』

要点: 将 SCM 与 CF 结构配合使用不依赖于任何特定版本的 IBM MQ。 但是，紧急存储方案仅适用于 IBM WebSphere MQ 7.1 和更高版本，因为它需要 SMDS 和卸载规则。

紧急存储器

SMDS 和消息卸载可与 SCM 配合使用，以降低在扩展中断期间将 MQRC_STORAGE_MEDIUM_FULL 原因码返回到 IBM MQ 应用程序的可能性。

概述

在应用程序结构上配置单个共享队列。 放置应用程序将消息放入共享队列; 获取应用程序从共享队列获取消息。

在正常运行期间，队列深度预期接近零，但业务需求指示系统必须能够容忍获取应用程序中断两小时。 这意味着共享队列必须能够包含来自放置应用程序的两小时消息。

目前，通过使用缺省卸载规则和 SMDS 来实现此过程，从而使结构的大小最小化，同时降低与卸载相关联的性能成本。

预期发送到共享队列的消息速率在短期到中期将增加一倍。 虽然仍存在要求系统能够容忍两小时的中断，但 CF 中没有足够的实际存储空间来使结构大小增加一倍。

由于包含应用程序结构的 CF 位于 zEC12 机器上，因此存在将足够的 SCM 与该结构相关联以存储足够的消息的可能性，以便可以容忍两小时的中断。

考虑一段时间内发生的情况:

1. 最初，该系统处于稳定状态。 放入和获取应用程序都正常运行，并且队列深度接近或处于零。 结果是应用程序结构很大程度上是空的。
2. 在某个时间，获取应用程序发生意外故障并停止。 放置应用程序继续将消息放置到队列中，应用程序结构开始填充。
3. 在结构达到 70% 已满后，将满足第一个卸载规则的条件，并且将大小大于或等于 32 KB 的所有消息卸载到 SMDS。

   请参阅 第 154 页的『SMDS 和卸载规则』 以获取卸载规则的概述。
4. 当消息继续放入共享队列时，结构将继续填充 (因为消息数据存储在结构中，或者由于指向要存储在结构中的已卸载消息的指针)。

   当结构达到 80% 满时，第二个卸载规则将开始应用， 4 KB 或更大的消息将卸载到 SMDS。
5. 当结构超过 90% 已满时，所有消息都将卸载到 SMDS ，并且只有消息指针放置在结构中。

   大约此时，预登台算法开始运行，并开始将数据从结构移动到 SCM 中。 假定队列上的所有消息都具有相同的优先级，那么会预先暂存最新的消息。

   因为现在正在将所有消息卸载到 SMDS ，所以要移动到 SCM 中的数据不是实际的消息数据，而是指向 SMDS 上的消息的指针。

   因此，可以存储在该结构以及与该结构相关联的 SCM 和 SMDS 的组合上的消息数量非常大。

   性能: 在中断的此阶段，由于必须写入 SMDS ，因此放置应用程序可能会遭受一定程度的性能降级。 在这种情况下，在性能方面， SCM 的使用不应成为放置应用程序的限制因素。 SCM 提供了额外的空间来防止结构填充。
6. 最终获取应用程序再次可用，中断已结束。

   但是，该结构仍在使用 SCM。 获取应用程序开始从队列中读取消息，首先获取最旧的最高优先级消息。

   因为这些消息是在结构开始填满之前写出来的，所以它们完全是从结构的实存部分出来的。
7. 随着结构开始清空，它将低于预登台处于活动状态的阈值，因此预登台将停止。
8. 结构使用率低于卸载规则生效的点，因此除非消息超过 63 KB ，否则不再将其卸载到 SMDS。

   大约此时，预取算法开始将数据从 SCM 移动到结构中。 由于获取应用程序按 SCM 算法所期望的顺序从队列中获取消息，因此在获取应用程序需要消息之前会引入消息。

   结果是获取应用程序从不需要等待从 SCM 同步引入消息。
9. 当获取应用程序继续下移队列时，它将开始检索已卸载到 SMDS 的消息。
10. 最后，系统再次处于稳定状态。 SCM 或 SMDS 中未存储任何消息，并且队列深度接近零。

提高性能

此场景描述使用 SCM 来增加可以存储在共享队列上的消息数，而不会产生使用 SMDS 的性能成本。

描述

对于此场景，放置和获取应用程序通过存储在应用程序结构中的共享队列进行通信。

放置应用程序往往以突发方式运行，当它在短时间内放置大量消息时。 然后，在很长一段时间内，它根本不会生成任何消息。

获取应用程序按顺序处理每条消息，并对每条消息执行复杂处理。 因此，大多数情况下队列深度为零，但放置应用程序开始运行时除外，在此情况下，队列深度开始增加，因为放置消息的速度比获取消息的速度要快。

队列深度会增加，直到放入应用程序停止，并且获取应用程序有足够的时间处理队列上的所有消息。

注意:

1. 在此场景中，关键因素是性能。 发送到队列的消息始终小于 63 KB ，因此从不需要卸载到 SMDS。
2. 已调整应用程序结构的大小，使其足以包含将应用程序放入单个 "分类群发" 中的所有消息。
3. 必须全部禁用卸载规则，以便即使在结构开始填充时，也不会将消息卸载到 SMDS。 这是因为与向 SMDS 写入消息和从 SMDS 读取消息相关联的性能成本被视为不可接受。

随着时间的推移，将应用程序发送到分类群发的消息数必须增加几个数量级。 由于获取应用程序必须按顺序处理每条消息，因此队列上的消息数会增加到结构填充的点。

此时，放置应用程序在放置消息时接收到原因码 (MQRC_STORAGE_MEDIUM_FULL) ，而放置操作失败。 放入应用程序只能在无法将消息放入队列时短暂容许时间段。 如果周期太长，那么应用程序将结束。

假定您没有时间或技能来重写放置应用程序或获取应用程序，那么此问题有三种可能的解决方案:

1. 增大应用程序结构的大小。
2. 将卸载规则添加到应用程序结构，以便在队列开始填满时将消息卸载到 SMDS。
3. 使 SCM 与结构相关联。

第一个解决方案可以快速实施，但在 CF 上没有足够的实际存储可用。

第二个解决方案也可以快速实施，但卸载到 SMDS 对性能的影响被认为太大，无法使用此选项。

第三种解决方案 (将 SCM 与结构相关联) 提供了可接受的成本和性能平衡。

将 SCM 与结构相关联会导致 CF 中的实存储器的使用率更高，因为使用了操作的扩充存储器。 但是，实际存储量将小于第一个选项中使用的量。

另一个考虑因素是 SCM 的成本。 但是，这一成本比实际存储要便宜得多。 这些因素结合起来，使得第三种选择比第一种选择更便宜。

虽然第三个选项可能不如第一个选项，但 CF 使用的预取和预登台算法可以组合在一起，使性能方面的差异可以接受，或者在某些情况下可以忽略不计。

当然，性能比使用 SMDS 卸载消息要好得多。

考虑一段时间内发生的情况:

1. 最初，获取应用程序处于活动状态，正在等待消息传递到共享队列。 放置应用程序未处于活动状态，并且共享队列为空。
2. 在一定时间，放置应用程序变为活动状态，并开始将大量消息放入共享队列。 获取应用程序开始获取消息，但队列深度快速开始增加，因为获取应用程序比放置应用程序慢。

   因此，应用程序结构开始填充。
3. 随着时间的增加，放置应用程序仍处于活动状态。 应用程序结构填充了大约 90%。

   这是 SCM 预登台算法开始将消息从结构移动到 SCM 中，从而释放结构中的空间。

   因为获取应用程序首先从队列中获取最旧的，最高优先级的消息，所以它总是从结构中获取消息，而不需要等待消息从 SCM 同步引入到结构中。
4. 放置应用程序仍处于活动状态，并将消息放置到共享队列。 但是，应用程序从不接收 MQRC_STORAGE_MEDIUM_FULL 原因码，因为 SCM 中存在足够的空间来存储结构中不适合的所有消息。
5. 最终，放置应用程序将停止，因为它没有更多要放置的消息。

预登台算法停止，因为结构在使用中低于 90% ，并且获取应用程序继续处理队列中的消息。

6. 当获取应用程序开始释放结构中的空间时，预取算法开始将消息从 SCM 返回到结构中。

   由于获取应用程序按预取算法所期望的顺序处理消息，因此获取应用程序从不阻塞等待消息数据从 SCM 同步引入到结构中。

7. 最后，获取应用程序会处理共享队列上的所有消息，并等到下一条消息可用。 结构和 SCM 不包含消息。

z/OS 紧急存储器-基本配置

如何在 IBM MQ 上设置紧急存储器的基本方案。

关于此任务

要点: IBM z16 计划是 IBM Z ® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

SMDS 和消息卸载可与 SCM 配合使用，以降低在扩展中断期间将 MQRC_STORAGE_MEDIUM_FULL 原因码返回到 IBM MQ 应用程序的可能性。

例如，您的企业有一个将消息放入队列的应用程序和一个从队列中获取消息的应用程序。 在正常运行期间，您期望队列深度接近零，但业务需求指示系统能够容忍获取消息的应用程序中断两小时。

这意味着正在使用的共享队列必须能够包含来自放置应用程序的两小时消息。 目前，您可以使用缺省卸载规则和 SMDS 来实现此目标。

您期望将消息发送到共享队列的速率在短期到中期内增加一倍。 虽然您要求系统能够容忍两小时的中断仍然存在，但 CF 中没有足够的实存储器可用于将结构的大小增加一倍。 由于包含应用程序结构的 CF 位于 zEC12 机器上，因此您可以将足够的 SCM 与该结构相关联，以存储足够的消息，从而可以容忍两个小时的中断

此初始方案使用以下内容:

- 包含单个队列管理器 CSQ3 的队列共享组 IBM1。 除管理结构外，队列共享组还定义了单个应用程序结构 SCEN1。
- 耦合设施 (CF) CF01，其中 SCEN1 应用程序结构存储为 IBM1SCEN1 结构。 此结构的最大大小为 1 GB。
- 应用程序结构使用的单个共享队列 SCEN1.Q 。

此配置在 第 159 页的图 61 中说明。

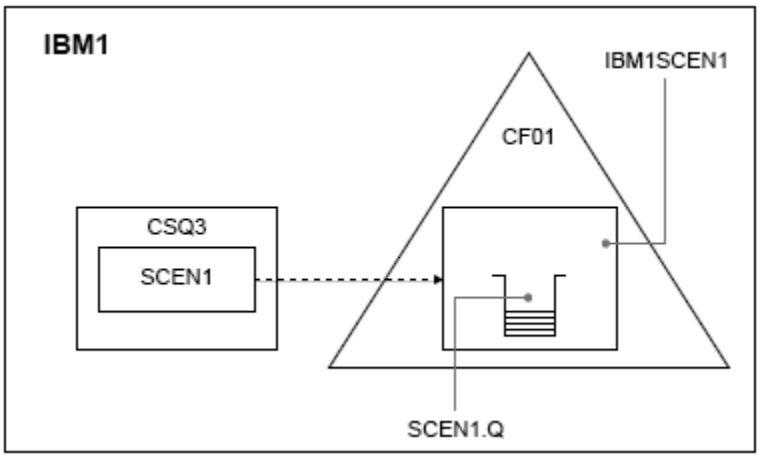


图 *61:* 基本配置

此外，假定队列管理器 CSQ3 已是队列共享组 IBM1 的唯一成员。

必须将结构 IBM1SCEN1 的定义添加到耦合设施资源管理器 (CFRM) 策略。 为了简单起见，定义了结构，以便只能通过指定 PREFLIST(CF01) 在单个耦合设施 CF01 中创建该结构。

注意: 要在生产系统中实现高可用性，对于 IBM MQ 所使用的任何结构，应该在 PREFLIST 中至少包含两个 CF。

过程

1. 使用以下命令刷新 CFRM 策略:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN1
```

结构 IBM1SCEN1 的样本 CFRM 策略:

```
STRUCTURE
NAME(IBM1SCEN1)
SIZE(1024M)
INITSIZE(512M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
```

2. 使用以下命令验证是否已正确创建结构:

```
D XCF,STR,STRNAME=IBM1SCEN1
```

此时，尚未将结构分配给队列共享组 (由 STATUS 行显示)。

3. 配置 IBM MQ 以使用 CFRM 策略中定义的结构。
   a. 使用结构名称为 SCEN1 的 DEFINE CFSTRUCT 命令来创建 IBM MQ CFSTRUCT 对象:

```
DEFINE CFSTRUCT(SCEN1)
CFCONLOS(TOLERATE)
CFLEVEL(5)
DESCR('Structure for SCM scenario 1')
RECOVER(NO)
RECAUTO(YES)
OFFLOAD(DB2)
OFFLD1SZ(64K) OFFLD1TH(70)
OFFLD2SZ(64K) OFFLD2TH(80)
OFFLD3SZ(64K) OFFLD3TH(90)
```

   .
   b. 使用 DISPLAY CFSTRUCT 命令验证结构。
   c. 使用以下 MQSC 命令定义 SCEN1.Q 共享队列以使用 SCEN1 结构:

```
DEFINE QLOCAL(SCEN1.Q) QSGDISP(SHARED) CFSTRUCT(SCEN1) MAXDEPTH(999999999)
```

4. 使用 IBM MQ Explorer 将单个消息放入队列 SCEN1.Q，然后再次关闭该消息。
5. 发出以下命令以检查现在是否已分配该结构:

```
D XCF,STR,STRNAME=IBM1SCEN1
```

检入命令的输出，即 STATUS 行显示 ALLOCATED。

结果

您已创建基本配置。 现在，您可以使用选择的任何方法来了解配置的基线性能。

下一步做什么

将 SMDS 和 SCM 添加到初始结构

相关概念

第 153 页的『将存储类内存与共享队列配合使用』

当与 IBM MQ for z/OS 共享队列配合使用时，使用存储类内存 (SCM) 可能是有利的。

z/OS 将 *SMDS* 和 *SCM* 添加到初始结构

如何在 IBM MQ 上为紧急存储器添加 SMDS 和 SCM。

关于此任务

要点: IBM z16 计划是 IBM Z ® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

此部分任务使用 第 159 页的『紧急存储器-基本配置』中描述的基本配置。 该方案描述了将共享消息数据集 (SMDS) 以及 SCM 添加到初始结构。

此最终配置在 第 161 页的图 62 中说明。

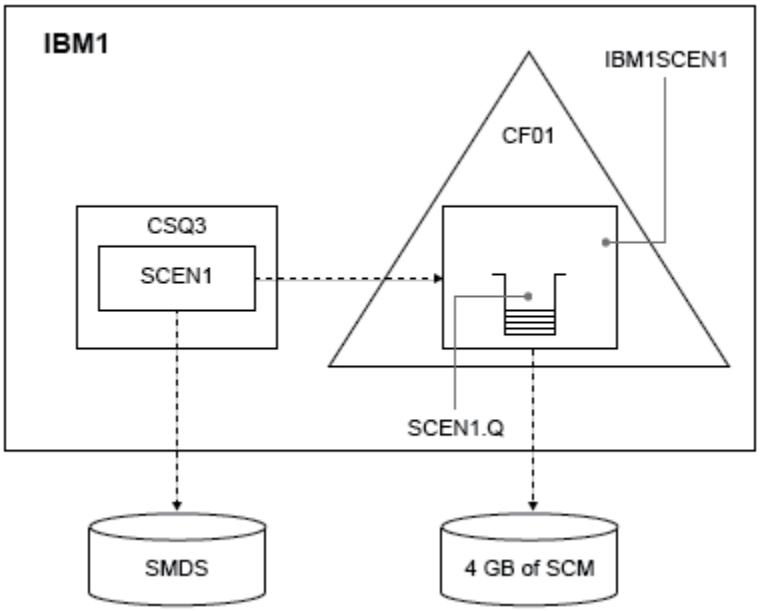


图 *62:* 为紧急存储器添加 *SMDS* 和 *SCM* 的配置

过程

1. 通过编辑 **CSQ4SMDS** 样本 JCL ， 创建 SCEN1 应用程序结构使用的 SMDS 数据集，如下所示:

```
//CSQ4SMDS JOB NOTIFY=&SYSUID
//*
//* Allocate SMDS
//*
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE CLUSTER           -
(NAME(CSQSMDS.SCEN1.CSQ3.SMDS) -
MEGABYTES(5000 3000)     -
LINEAR                -
SHAREOPTIONS(2 3) )      -
DATA                  -
(NAME(CSQSMDS.SCEN1.CSQ3.SMDS.DATA) )
/*
//*
//* Format the SMDS
//*
//FORM   EXEC PGM=CSQJUFMT,COND=(0,NE),REGION=0M
//STEPLIB DD DSN=MQ800.SCSQANLE,DISP=SHR
//     DD DSN=MQ800.SCSQAUTH,DISP=SHR
//SYSUT1  DD DISP=OLD,DSN=CSQSMDS.SCEN1.CSQ3.SMDS
//SYSPRINT DD SYSOUT=*
```

2. 发出 ALTER CFSTRUCT 命令以更改 SCEN1 应用程序结构，以使用 SMDS 进行卸载，并实现缺省卸载规则:

```
ALTER CFSTRUCT(SCEN1) OFFLOAD(SMDS) OFFLD1SZ(32K) OFFLD2SZ(4K) OFFLD3SZ(0K)
DSGROUP('CSQSMDS.SCEN1.*.SMDS') DSBLOCK(1M)
```

   请注意下列事项:

   - 由于 SCEN1.Q 是 SCEN1 应用程序结构上的唯一共享队列，因此 **DSBLOCK** 值已设置为 1M，这是可能的最大值。 这应该是我们场景最高效的设置。
   - 由于放置应用程序发送的消息为 30 KB，因此当结构已满 80% 时，直到满足第二个卸载规则时，才会启动对 SMDS 的卸载。

3. 再次运行测试应用程序。

   请注意增加队列上消息的存储量。

4. 通过执行以下过程将 4 GB SCM 添加到结构 IBM1SCEN1 中:

   a) 通过发出以下命令，检查已安装并分配给 CF01 的 SCM 数量:

   ```
   D CF,CFNAME=CF01
   ```

   b) 检查所显示输出的 STORAGE CONFIGURATION 部分中的 STORAGE-CLASS MEMORY 图以查看可用存储器。

   c) 使用 SCMMAXSIZE 和 SCMALGORITHM 关键字更新 CFRM 策略，如下所示:

   ```
   STRUCTURE
   NAME(IBM1SCEN1)
   SIZE(1024M)
   INITSIZE(512M)
   ALLOWAUTOALT(YES)
   FULLTHRESHOLD(85)
   PREFLIST(CF01)
   ALLOWREALLOCATE(YES)
   DUPLEX(DISABLED)
   ENFORCEORDER(NO)
   SCMMAXSIZE(4G)
   SCMALGORITHM(KEYPRIORITY1)
   ```

5. 通过发出以下命令来激活 CFRM 策略:

   ```
   SETXCF START,POLICY,TYPE=CFRM,POLNAME=polname
   ```

6. 重建 IBM1SCEN1 结构。

   必须执行此过程，因为结构是在进行先前更改时分配的。

   发出以下命令以重建结构:

   ```
   SETXCF START,REBUILD,STRNM=IBM1SCEN1
   ```

结果

您已成功将 SCM 添加到配置。

下一步做什么

优化系统性能。 请参阅 第 162 页的『优化存储类内存使用情况』 以获取更多信息。

z/OS 优化存储类内存使用情况

如何改进存储类内存 (SCM) 的使用。

要点: IBM z16 计划是 IBM Z® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

运行以下命令:

```
D XCF,STR,STRNAME=IBM1SCEN1
```

由于该结构已充满了消息数据，因此由于先前的测试，部分重建涉及将部分消息从该结构预登台到 SCM 中。 此进程是使用先前命令启动的。

此命令的输出生成，例如:

```
ACTIVE STRUCTURE
----------------
ALLOCATION TIME: 06/17/2014 09:28:50
CFNAME : CF01
COUPLING FACILITY: 002827.IBM.02.00000000B8D7
PARTITION: 3B CPCID: 00
STORAGE CONFIGURATION ALLOCATED MAXIMUM %
ACTUAL SIZE: 1024 M 1024 M 100
AUGMENTED SPACE: 3 M 142 M 2
STORAGE-CLASS MEMORY: 88 M 4096 M 2
ENTRIES: 120120 1089536 11
ELEMENTS: 240240 15664556 1
SPACE USAGE IN-USE TOTAL %
ENTRIES: 84921 219439 38
ELEMENTS: 2707678 3149050 85
EMCS: 2 282044 0
LOCKS: 1024
SCMHIGHTHRESHOLD : 90
SCMLOWTHRESHOLD : 70
ACTUAL SUBNOTIFYDELAY: 5000
PHYSICAL VERSION: CD5186A0 2BD8B85C
LOGICAL VERSION: CD515C50 CE2ED258
SYSTEM-MANAGED PROCESS LEVEL: 9
XCF GRPNAME : IXCLO053
DISPOSITION : KEEP
ACCESS TIME : NOLIMIT
MAX CONNECTIONS: 32
# CONNECTIONS : 1
CONNECTION NAME ID VERSION  SYSNAME JOBNAME  ASID STATE
--------------- -- -------  ------- --------  ---- ----------------
CSQEIBM1CSQ301  01 00010059 SC61    CSQ3MSTR 0091 ACTIVE
```

请注意命令输出中的以下内容:

- 该 STORAGE_CLASS MEMORY 确认已将 **MAXIMUM** 的 4096 MB SCM 添加到结构中。
- 用于预登台的 STORAGE-CLASS MEMORY 数量的 ALLOCATED 图。 现在，在添加 SCM 之前，结构中没有可用空间。
- 用于跟踪 SCM 使用情况的 AUGMENTED SPACE 数量。
- 预登台算法开始将数据从结构移动到 SCM 的点是当结构已满 90% 时。 这由不可配置的 **SCMHIGHTHRESHOLD** 属性指示。
- 预取算法开始将数据从 SCM 移动到结构中的以下点是在结构已满 70% 时。 这由不可配置的 **SCMLOWTHRESHOLD** 属性指示。

现在，您可以测试各种方法来优化 SCM 的使用。 请注意下列事项:

- 在使用 SCM 来存储消息之后，只有在从 SCM 中除去所有数据之后，才能更改结构。

  在这种情况下，这意味着输入/元素比率将冻结在首次使用 SCM 时的值。 在预登台算法开始将数据移动到 SCM 中之前，必须仔细确保结构处于所需的状态。
- 在使用 SCM 之前，当前结构大小是否正确?

  例如，您是否已将 **INITSIZE** 从 512 MB 增加到 1 GB 的 SIZE?

  如果不执行此操作，那么虽然您已启用自动更改结构，但预登台算法将在更改有机会启动之前开始将数据移动到 SCM 中。 因此，该结构使用 512 MB 的实存储器进行冻结。
- 在使用 SCM 之前，条目对元素的比率是否正确?

  此场景的目标是增加可作为整体存储在结构和 SCM 中的卸载消息指针数，以及尽可能将尽可能多的消息完全保存在结构存储器中。 访问这些消息比访问 SMDS 上的消息更快。

  因此，您需要具有以有利于存储消息的入口/元素比率开始的结构，然后在预登台算法首次启动之前转换为有利于存储消息指针的比率。 可以部分通过使用 IBM MQ 卸载规则来实现此转换。

通过发出以下命令来更改卸载规则:

```
ALTER CFSTRUCT(SCEN1) OFFLD1SZ(0K)
```

您可能必须执行几次运行以优化条目与元素的比率。

下表显示了在紧急存储方案的不同阶段中放入队列的消息数的可能改进。

表 *18:* 紧急存储方案的结果比较		
测试描述	**消息数**	**填充队列的时间 (秒)**
基本配置	27,850	3.2
具有缺省卸载规则的 SMDS	205,000	158
具有缺省卸载规则的 SCM	828,610	469
具有调整后的卸载规则的 SCM	1,135,775	679

表中的最后一行显示调整卸载规则具有必需的效果。

您需要检查您的系统，以查看是否可以以任何方式对这些数字进行改进。 例如，您可能会耗尽可用的 SMDS 存储器。 如果可以分配更多 SMDS 存储器，那么您应该能够显着增加队列上的消息数。

z/OS 改进的性能-基本配置

如何在 IBM MQ 上使用共享队列来设置基本方案以提高性能。

关于此任务

要点: IBM z16 计划是 IBM Z ® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

此场景描述了如何使用 SCM 来增加可以存储在共享队列上的消息数，而不会产生使用 SMDS 的性能成本。

此初始场景与用于紧急存储的场景非常相似，并使用:

- 包含单个队列管理器 CSQ3 的队列共享组 IBM1。 除管理结构外，队列共享组还定义了单个应用程序结构 SCEN2。
- 耦合设施 (CF) CF01，其中 SCEN2 应用程序结构存储为 IBM1SCEN2 结构。 此结构的最大大小为 2 GB。
- 配置为使用应用程序结构的单个共享队列 SCEN2.Q。

此配置在 第 164 页的图 63 中说明。

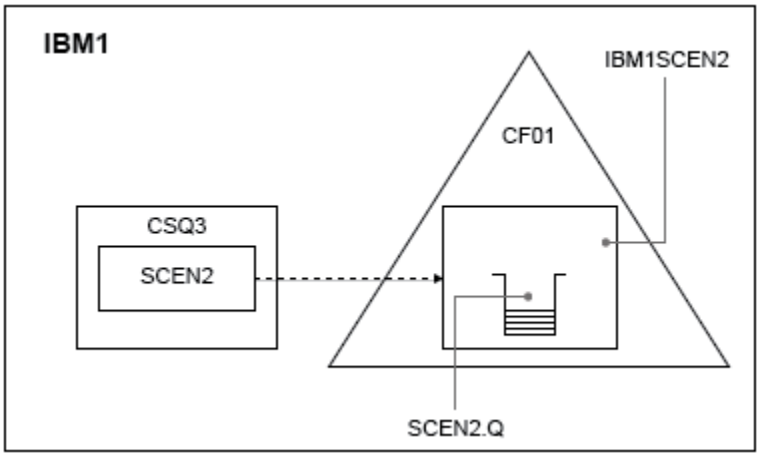


图 *63:* 基本配置

此外，假定队列管理器 CSQ3 已是队列共享组 IBM1 的唯一成员。

必须将结构 IBM1SCEN2 的定义添加到耦合设施资源管理器 (CFRM) 策略。 为了简单起见，定义了结构，以便只能通过指定 PREFLIST(CF01)在单个耦合设施 CF01 中创建该结构。

结构 IBM1SCEN2 的样本 CFRM 策略:

```
STRUCTURE
NAME(IBM1SCEN2)
SIZE(2048M)
INITSIZE(2048M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
```

INITSIZE 和 **SIZE** 关键字都具有值 2048M， 因此结构无法调整大小。

过程

1. 使用以下命令刷新 CFRM 策略:

   ```
   SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN2
   ```

2. 使用以下命令验证是否已正确创建结构:

   ```
   D XCF,STR,STRNAME=IBM1SCEN2
   ```

   发出上述命令将提供以下输出:

   ```
   RESPONSE=SC61
   IXC360I 07.58.51 DISPLAY XCF 581
   STRNAME: IBM1SCEN2
   STATUS: NOT ALLOCATED
   POLICY INFORMATION:
   POLICY SIZE : 2048 M
   POLICY INITSIZE: 2048 M
   POLICY MINSIZE : 1536 M
   FULLTHRESHOLD : 85
   ALLOWAUTOALT : YES
   REBUILD PERCENT: N/A
   DUPLEX : DISABLED
   ALLOWREALLOCATE: YES
   PREFERENCE LIST: CF01
   ENFORCEORDER : NO
   EXCLUSION LIST IS EMPTY

   EVENT MANAGEMENT: MESSAGE-BASED   MANAGER SYSTEM NAME: SC53
   MANAGEMENT LEVEL : 01050107
   ```

   此时，尚未将结构分配给队列共享组 (由 STATUS 行显示)。

3. 配置 IBM MQ 以使用 CFRM 策略中定义的结构。

   a. 使用具有结构名称 SCEN2 的 DEFINE CFSTRUCT 命令来创建 IBM MQ CFSTRUCT 对象。

   ```
   DEFINE CFSTRUCT(SCEN2)
   CFCONLOS(TOLERATE)
   CFLEVEL(5)
   DESCR('Structure for SCM scenario 2')
   RECOVER(NO)
   RECAUTO(YES)
   OFFLOAD(DB2)
   OFFLD1SZ(64K) OFFLD1TH(70)
   OFFLD2SZ(64K) OFFLD2TH(80)
   OFFLD3SZ(64K) OFFLD3TH(90)
   ```

   b. 使用 DISPLAY CFSTRUCT 命令检查结构。

   c. 使用以下 MQSC 命令定义 SCEN2.Q 共享队列以使用 SCEN2 结构:

   ```
   DEFINE QLOCAL(SCEN2.Q) QSGDISP(SHARED) CFSTRUCT(SCEN2) MAXDEPTH(999999999)
   ```

4. 使用 IBM MQ 资源管理器将单个消息放入队列 SCEN2.Q， 然后再次关闭该消息。

5. 发出以下命令以检查现在是否已分配该结构:

```
D XCF,STR,STRNAME=IBM1SCEN2
```

查看命令的输出 (显示部分) ，并确保 STATUS 行显示 ALLOCATED。

```
RESPONSE=SC61
IXC360I 08.31.27 DISPLAY XCF 703
STRNAME: IBM1SCEN2
STATUS: ALLOCATED
EVENT MANAGEMENT: MESSAGE-BASED
TYPE: SERIALIZED LIST
POLICY INFORMATION:
POLICY SIZE : 2048 M
POLICY INITSIZE: 2048 M
POLICY MINSIZE : 1536 M
FULLTHRESHOLD : 85
ALLOWAUTOALT : YES
REBUILD PERCENT: N/A
DUPLEX : DISABLED
ALLOWREALLOCATE: YES
PREFERENCE LIST: CF01
ENFORCEORDER : NO
EXCLUSION LIST IS EMPTY
```

此外，请注意 SPACE USAGE 部分中字段的值:

- 条目
- 元素
- EMCS
- 锁定

以下是值的示例:

```
SPACE USAGE   IN-USE      TOTAL    %
ENTRIES:      344686     345242   99
ELEMENTS:    6548455    6548467   99
EMCS:              2     780318    0
LOCKS:          1024
```

结果

您已创建基本配置。 现在，您可以使用选择的任何方法来了解配置的基线性能。

下一步做什么

您应该测试基本方案。 例如，您可以使用以下三个应用程序，按显示的顺序启动应用程序，并同时运行这些应用程序。

1. 使用 PCF 应用程序来请求当前深度 (**CURDEPTH**) SCEN2.Q 的值 (每 5 秒)。 输出可用于绘制一段时间内队列的深度。
2. 单线程获取应用程序通过使用具有无限等待的 get 从 SCEN2.Q 重复获取消息。 为模拟已除去的消息的处理，获取应用程序将对其除去的每 10 条消息暂停 4 毫秒。
3. 单线程放置应用程序将总共 100 万条 4 KB 非持久消息放入 SCEN2.Q。 此应用程序不会在放入每条消息之间暂停，因此放入消息的速度比获取消息的应用程序获取消息的速度更快 SCEN2.Q 。

   因此，当放置应用程序正在运行时， SCEN2.Q 的深度会增加。

   填充了结构 IBM1SCEN2 ，并且放置应用程序接收到 MQRC_STORAGE_MEDIUM_FULL 原因码时，放置应用程序会在尝试将下一条消息放入队列之前休眠 5 秒。

您可以在一段时间内绘制 CURDEPTH 应用程序的结果。 在放置应用程序暂停时获取某种形式的锯齿波输出，以允许队列部分清空。

转至 第 167 页的『将 SCM 添加到初始结构』 。

相关概念

第 153 页的『将存储类内存与共享队列配合使用』
当与 IBM MQ for z/OS 共享队列配合使用时，使用存储类内存 (SCM) 可能是有利的。

z/OS 将 *SCM* 添加到初始结构
如何添加 SCM 以提高 IBM MQ 上的性能。

关于此任务

要点: IBM z16 计划是 IBM Z ® 的最后一代，用于支持将虚拟闪存 (也称为存储类内存或 SCM) 用于耦合设施映像。 有关更多信息，请参阅: IBM Z 和 IBM LinuxONE 4Q 2023 Direction 语句。

作为替代方法，您应该使用更大的结构或将消息卸载到 SMDS。

此部分任务使用 第 164 页的『改进的性能-基本配置』中描述的基本配置。 本方案描述将 SCM 添加到初始结构。

此最终配置在 第 167 页的图 64 中说明。

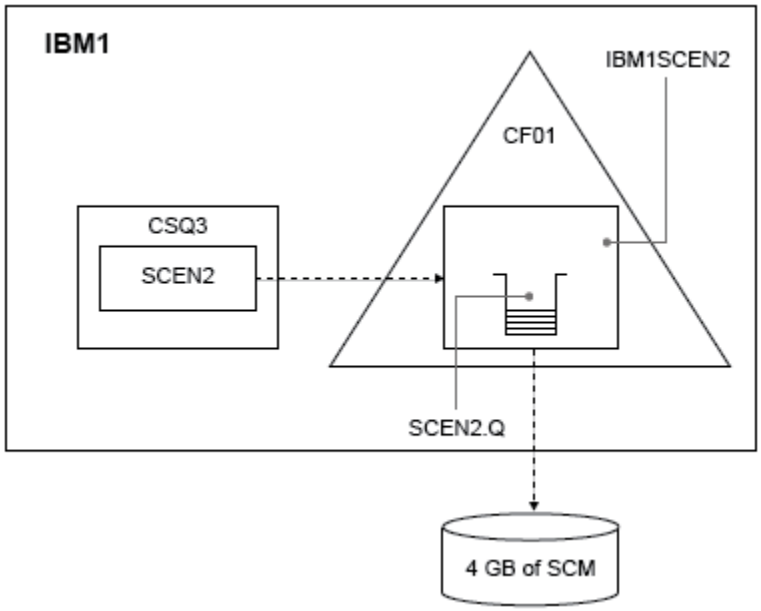


图 *64:* 配置添加 *SCM* 以提高性能

过程

1. 通过执行以下过程将 4 GB SCM 添加到结构 IBM1SCEN2 中:
   a) 通过发出以下命令，检查已安装并分配给 CF01 的 SCM 数量:

```
D CF,CFNAME=CF01
```

   b) 检查所显示输出的 STORAGE CONFIGURATION 部分中的 STORAGE-CLASS MEMORY 图以查看可用存储器。
   c) 使用 SCMMAXSIZE 和 SCMALGORITHM 关键字更新 CFRM 策略，如下所示:

```
STRUCTURE
NAME(IBM1SCEN2)
SIZE(2048M)
INITSIZE(2048M)
ALLOWAUTOALT(YES)
FULLTHRESHOLD(85)
PREFLIST(CF01)
ALLOWREALLOCATE(YES)
DUPLEX(DISABLED)
ENFORCEORDER(NO)
```

```
SCMMAXSIZE(4G)
SCMALGORITHM(KEYPRIORITY1)
```

2. 通过发出以下命令来激活 CFRM 策略:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=IBM1SCEN2
```

3. 重建 IBM1SCEN2 结构。
   必须执行此过程，因为结构是在进行先前更改时分配的。
   发出以下命令以重建结构:

```
SETXCF START,REBUILD,STRNM=IBM1SCEN2
```

4. 发出以下命令以确认结构的新配置:

```
D XCF,STR,STRNAME=IBM1SCEN2
```

   查看命令的输出，其部分如下:

```
SPACE USAGE     IN-USE       TOTAL      %
ENTRIES:          33         342684     0
ELEMENTS:         48        6503697     0
EMCS:              2         575600     0
LOCKS:                         1024
```

结果

通过增加使用 SCM 所需的控制存储器，计算实存储器使用方面的变化。

- 在将 SCM 添加到该结构之前，该结构具有以下总计，如 第 164 页的『改进的性能-基本配置』中所示:
  - 345,242 个条目
  - 6 548 467 个要素
  - 780,318 EMCS
- 将 SCM 添加到该结构后，该结构具有以下总计:
  - 342,684 个条目
  - 6 503 697 个要素
  - 575,600 EMCS

使用这些数字，在添加 SCM 后，将通过以下方法减小结构的大小:

- 2558 个条目
- 44 770 个元素
- 204,718 EMCS

对于已分配 4 GB SCM 的 2 GB 结构，用于管理 SCM 的结构存储量如下所示:

```
(2558 + 44,770 + 204,718) * 256 = 61.5 MB
```

请注意，添加更多 SCM 可能仅实现结构大小的边际缩减，因为用于跟踪 SCM 的控制存储量会随着结构大小和已分配 SCM 的数量增加而增加。

下一步做什么

重复 第 164 页的『改进的性能-基本配置』的最后部分中描述的测试。

您可以在一段时间内绘制修订后的应用程序的结果。 将图与先前获得的图进行比较，您现在获得一个没有锯齿波的输出，因为放置应用程序不再需要等待队列部分清空。

有关更多信息，请参阅 MP16: WebSphere MQ for z/OS -容量规划和调整。

z/OS 分布式排队和队列共享组

分布式排队和队列共享组是可用于提高应用程序系统可用性的两种方法。 使用本主题来查找有关这些方法的更多信息。

为了补充共享队列上消息的高可用性， IBM MQ 的分布式排队组件具有其他功能来提供以下功能:

- 更高的网络可用性。
- 增加了与队列共享组的入站网络连接的容量。

第 169 页的图 65 说明了分布式排队和队列共享组。 它显示综合系统中的两个队列管理器，这两个队列管理器都属于同一个队列共享组。 它们都可以访问共享队列 SQ1。 网络中的队列管理器 (例如，在 AIX 和 Windows 上) 可以通过任一队列管理器的通道启动程序将消息放入此队列。 两个队列管理器上的克隆应用程序都为队列提供服务。

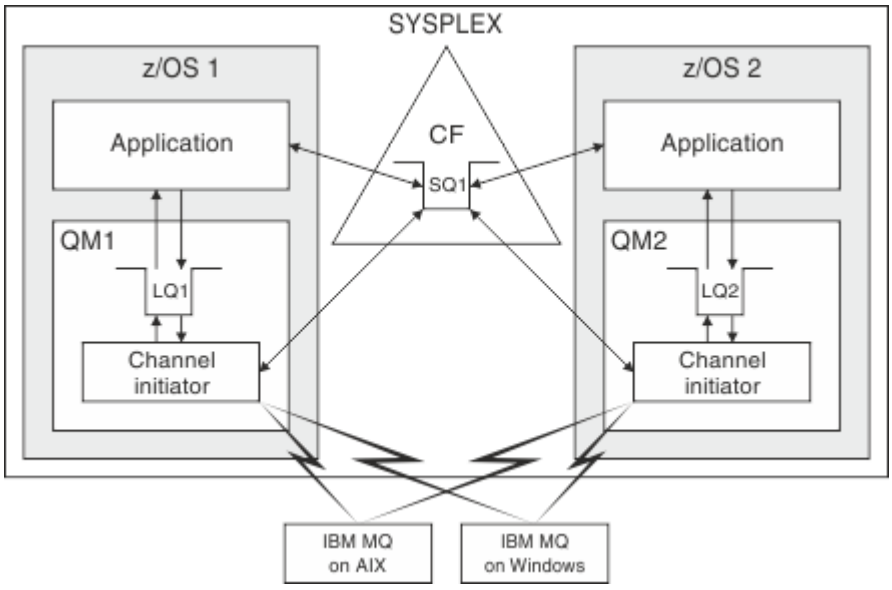


图 *65:* 分布式排队和队列共享组

相关概念

第 169 页的『共享通道』
使用本主题来了解共享通道的概念及其与 IBM MQ for z/OS 的使用。

第 173 页的『组内排队』
本部分描述了组内排队，这是 z/OS 平台独有的 IBM MQ for z/OS 函数。 此函数仅对定义到队列共享组的队列管理器可用。

第 171 页的『集群和队列共享组』
使用本主题来了解如何将队列共享组与集群配合使用。

z/OS 共享通道

使用本主题来了解共享通道的概念及其与 IBM MQ for z/OS 的使用。

许多网络产品提供了一种机制来隐藏网络中的服务器故障，或者在一组符合条件的服务器之间平衡入站网络请求。 网络产品使 通用端口 可用于入站网络连接请求，并且可以通过连接到其中一个合格服务器来满足入站请求。

这些联网产品包括:

- VTAM 通用资源
- SYSPLEX 分销商

通道启动程序利用这些产品来使用共享队列的功能

有两种类型的共享通道: 共享入站通道和 共享出站通道。

- 共享入站通道
- 共享出站通道

有关渠道的更多信息，请参阅

- 共享通道摘要
- 共享通道状态

共享入站通道

队列共享组中的每个通道启动程序都会启动一个额外的侦听器任务，以在 通用端口上进行侦听。 此通用端口由其中一种支持技术 (VTAM 和 TCP/IP) 提供给网络。 对通用端口的入站网络连接请求由网络技术分派给在通用端口上侦听的队列共享组 (QSG) 中的任何一个侦听器。

如果通道启动程序有权访问具有该名称的通道的通道定义，那么可以在入站连接定向到的通道启动程序上启动通道。 您可以将通道定义定义为队列管理器专用或存储在共享存储库中，以便在任何位置提供 (全局定义)。 这意味着您可以通过将通道定义定义为全局定义，使其在队列共享组中的任何通道启动程序上可用。

通过通用端口启动通道时存在其他差异; 通道同步与队列共享组而不是与单个队列管理器同步。 例如，考虑远程队列管理器通过通用端口启动通道。 当通道首次启动时，它可能在队列管理器 QM1 和消息流上启动。 如果通道停止并在队列管理器 QM2 上重新启动，那么有关已流动的消息数的信息仍然正确，因为与队列共享组同步。

您可以使用通过通用端口启动的入站通道将消息放入任何队列。 远程队列管理器不知道目标队列是否共享。 如果目标队列是共享队列，那么远程队列管理器将以负载均衡的方式通过任何可用通道启动程序进行连接，并将消息放入共享队列。

如果目标队列是专用队列，那么会将消息放入通道的当前实例所连接到的队列管理器所拥有的专用队列。 在此环境 (称为 复制的本地队列) 中，每个队列管理器必须定义相同的专用队列集。

为队列共享组配置 SVRCONN 通道

队列共享组中 SVRCONN 通道的最佳配置是在使用从点到点通道的不同端口号的每个 CHINIT 中设置专用侦听器。 然后，这些侦听器端口将用作新的工作负载分发机制 (例如，使用虚拟 IP 地址 (VIPA) 的综合系统分发器) 的 "后端" 资源。 然后，外部 VIPA 地址将用作网络中 CLNTCONN 定义的目标地址。 可以使用 QSGDISP (GROUP) 定义 SVRCONN 通道，因此 QSG 中的所有队列管理器都可以使用相同的定义。 此配置避免使用共享侦听器，因此会降低队列共享组维护共享通道状态的性能效果，客户机/服务器通道不需要此效果。

共享出站通道

如果出站通道从共享传输队列中获取消息，那么该出站通道被视为共享通道。 如果它是共享的，那么它将在队列共享组级别保存同步信息。 这意味着，如果通信子系统，通道启动程序或队列管理器发生故障，那么可以在队列共享组中的其他队列管理器和通道启动程序实例上重新启动通道。 以这种方式重新启动失败的通道是称为 对等通道恢复的共享通道的功能。

共享出站通道的工作负载均衡

如果您尚未指定要在特定通道启动程序上启动出站共享通道，那么该通道可以在队列共享组中的任何通道启动程序上启动。 IBM MQ 选择的通道启动程序是使用以下条件确定的:

- 当前需要的通信子系统对通道启动程序可用吗?
- Db2 连接是否可用于通道启动程序?
- 哪个通道启动程序的当前工作负载最低? 工作负载包括处于活动状态并正在重试的通道。

共享通道摘要

共享通道与专用通道在以下方面有所不同:

专用通道
与单个通道启动程序绑定。

- 出站通道使用本地传输队列。
- 已通过本地端口启动入站通道。
- 同步信息保存在 SYSTEM.CHANNEL.SYNCQ 队列。

共享通道
工作负载与高可用性平衡。

- 出站通道使用共享传输队列。
- 通过通用端口启动入站通道。
- 同步信息保存在 SYSTEM.QSG.CHANNEL.SYNCQ 队列。

通过将 CHLDISP 选项与 START CHANNEL 命令配合使用来启动通道时，可以指定通道是专用通道还是共享通道。 可以通过触发与专用通道相同的方式来启动共享通道。 但是，启动共享通道时， IBM MQ 会执行工作负载均衡，并在队列共享组中最合适的通道启动程序上启动该通道。 (如果需要，可以指定将在特定通道启动程序上启动共享通道。)

共享通道状态

队列共享组中的通道启动程序在 Db2 中维护共享通道状态表。 这将记录哪些通道在哪些通道启动器上处于活动状态。 如果存在通道启动程序或通信系统故障，那么将使用共享通道状态表。 它指示需要在队列共享组中的其他通道启动程序上重新启动哪些通道。

z/OS 集群和队列共享组

使用本主题来了解如何将队列共享组与集群配合使用。

您可以使共享队列可用于单个定义中的集群。 要执行此操作，请在定义共享队列时指定集群的名称。

网络中的用户将共享队列视为由队列共享组中的每个队列管理器托管 (未将共享队列通告为由队列共享组托管)。 客户机可以与队列共享组的任何成员启动会话，以将消息放入同一共享队列。

第 171 页的图 66 显示了集群成员如何通过队列共享组的任何成员访问共享队列。

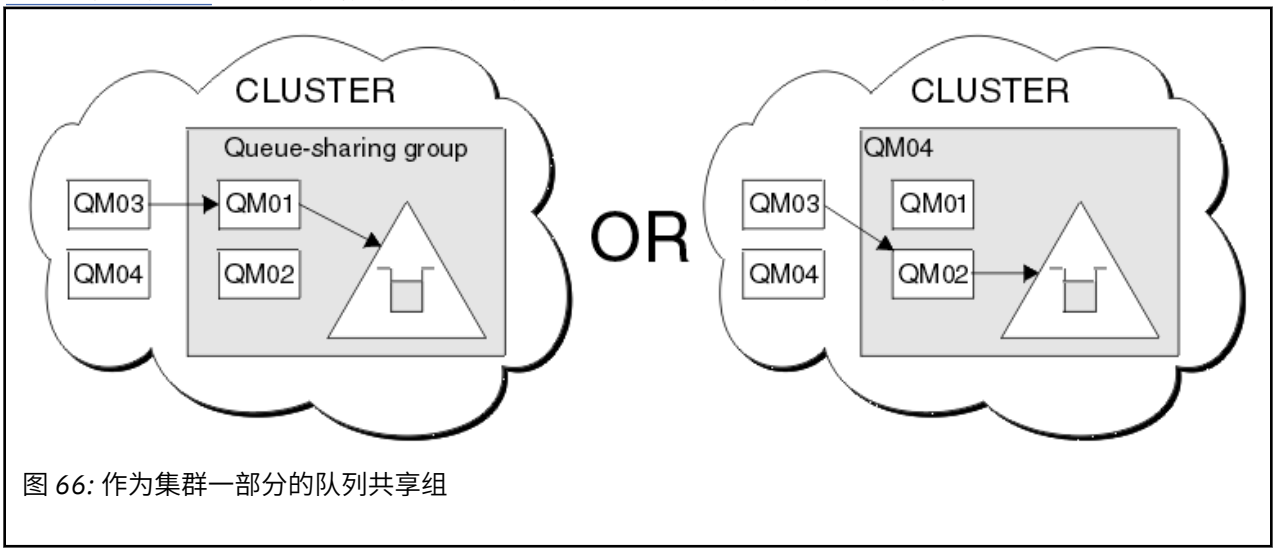


图 66: 作为集群一部分的队列共享组

z/OS 使用共享队列影响工作负载分布

使用本主题来了解影响队列共享组中的共享队列的工作负载分布的因素。

IBM MQ 未提供共享队列的工作负载均衡。 但是，队列共享组 (QSG) 中的工作负载分布可能会以 基于拉取的方式受到影响。 队列管理器服务 (接收写入共享队列的消息) 的选择受队列共享组中每个队列管理器的可用处理容量以及跨综合系统定义的工作负载管理目标的影响。

但是，很重要的一点是，执行消息的 MQPUT 的队列管理器在决定哪个队列管理器获取消息时也会产生很大影响。

本地队列管理器更有可能执行 MQGET

对于执行 MQPUT 的应用程序，本地队列管理器被认为是应用程序所连接的队列管理器。

以下注意事项会影响哪个队列管理器通过代表获取应用程序执行 MQGET 来为消息的 MQPUT 提供服务。

将消息放入空共享队列时，通常会先发布本地队列管理器，然后再通知队列共享组中的任何其他队列管理器。 如果本地队列管理器能够处理消息，那么它会在 QSG 中的任何其他队列管理器之前接收来自耦合设施 (CF) 的列表转换通知。 (列表转换通知是共享队列已将状态从 "空" 更改为 "非空" 的通知。)

在这种情况下，可能的情况如下:

1. 非持久消息的 MQPUT 不同步点， 快速放入正在等待的 *getter*。

   如果在队列的本地队列管理器上存在具有 *MQGET with wait* 的应用程序，那么消息的 MQPUT 将直接传递到获取应用程序的缓冲区，而不会写入队列。 这适用于共享队列和非共享队列。 此功能通常称为 快速放入等待的 *getter* 机制。 对于共享队列，不会通知 QSG 中的其他队列管理器，因为没有从空到非空的队列转换。 例如，这意味着，如果此队列管理器可以处理来自此应用程序的所有放入，并且假定没有其他应用程序将消息放入队列，那么队列共享组中的其他队列管理器都不会帮助排出此队列。 但是，如果本地队列管理器上没有等待的 MQGET，并且将消息放入共享队列，那么 CF 将根据其通知列表转换的规则来通知队列共享组中的其他队列管理器。
2. 持久或同步点消息的 MQPUT。

   在这种情况下，如果本地队列管理器上存在具有 *MQGET with wait* 的应用程序，那么会将消息放入共享队列，并且 CF 会根据其列表转换通知规则来通知队列共享组中的其他队列管理器。 但是，本地队列管理器不会等待来自 CF 的转换通知，而是首先采用任何本地 *MQGET with wait*，并且通常代表应用程序执行此消息的获取，然后队列共享组中的任何其他队列管理器才能响应 CF 通知。 这取决于本地队列管理器的繁忙程度。 否则， CF 由于消息到达空队列而通知的任何队列管理器都将尝试先处理 get。 响应的第一个队列管理器将处理新消息。
3. 最后，如果队列未耗尽消息，其中 CF 已针对队列发送状态从 "空" 更改为 "非空" 的通知，那么所有已连接的队列管理器都将有机会帮助处理队列。 在此情况下，工作负载据说是 基于拉取的。

此设计允许在纯基于拉取的工作负载分布上提高性能。 目的是利用 CF 中保留的队列提供的高可用性，同时允许队列管理器在可能的情况下执行 MQGET，而不需要引用 CF，从而尽可能高效地处理消息工作负载。

在强调工作负载平衡比先前描述的性能增强更重要的情况下，可以采用替代方法。 例如，确保没有任何获取应用程序连接到放置应用程序所连接到的同一队列管理器。 使用此设计时，将所有消息放入队列中，当队列从空移至非空时，将根据用于处理此类转换的 CF 算法通知 QSG 中的所有队列管理器。 此外， 快速放入等待 *getter* 机制不适用。

z/OS 在何处查找有关共享队列和队列共享组的更多信息

使用本主题中的表来查找有关 IBM MQ for z/OS 如何使用共享队列和队列共享组的更多信息。

表 *19:* 在何处查找有关共享队列和队列共享组的更多信息

Topic	**在哪里查找信息**
队列共享组恢复	第 206 页的『在 z/OS 上恢复并重新启动』
队列共享组安全性	第 219 页的『IBM MQ for z/OS 中的安全概念』

表 19: 在何处查找有关共享队列和队列共享组的更多信息 (继续)	
Topic	**在哪里查找信息**
专用和全局对象定义 将命令定向到不同的队列 :NONE.	可在 z/OS 上发出命令的源
规划耦合设施 环境	定义耦合设施资源
规划 SMDS 环境	规划共享消息数据集 (SMDS) 环境
规划您的 Db2 环境	规划 Db2 环境
设置共享队列 系统参数	第 137 页的『共享队列和队列共享组』
实用程序 迁移队列	z/OS 参考上的 IBM MQ 实用程序
控制台消息	IBM MQ for z/OS 的消息
MQSC 命令	MQSC 命令
IBM MQ 集群	配置队列管理器集群
IBM MQ 分布式排队 (distributed queuing) 通道名称	分布式队列管理简介
编写应用程序	应用程序设计概述
MQCONNX 调用	MQCONNX

z/OS 组内排队

本部分描述了组内排队，这是 z/OS 平台独有的 IBM MQ for z/OS 函数。 此函数仅对定义到队列共享组的队列管理器可用。

有关队列共享组的信息，请参阅 第 137 页的『共享队列和队列共享组』。

组内排队概念

您可以在队列共享组中的队列管理器之间执行快速消息传输，而无需定义通道。 这将使用称为 SYSTEM.QSG.TRANSMIT.QUEUE，这是共享传输队列。 队列共享组中的每个队列管理器都会启动一个名为 "组内排队代理" 的任务，该任务将等待消息到达此队列，这些消息将发送给它们的队列管理器。 检测到此类消息时，会将其从队列中除去并放置在正确的目标队列上。

将使用标准名称解析规则，但如果启用了组内排队 (IGQ) 并且目标队列管理器位于队列共享组 SYSTEM.QSG.TRANSMIT.QUEUE 用于将消息传输到正确的目标队列管理器，而不是使用传输队列和通道。

通过队列管理器属性启用组内排队。 组内排队将非持久消息移动到同步点外，并将持久消息移动到同步点内。 如果发现将消息传递到目标队列的问题，那么组内排队会尝试将这些消息放入死信队列。 如果死信队

列已满或未定义，那么将废弃非持久消息，但持久消息将回退并返回到 SYSTEM.QSG.TRANSMIT.QUEUE 和 IGQ 代理程序尝试传递消息，直到成功。

接收以队列共享组中另一队列管理器上的队列为目标的消息的入站共享通道可以使用组内排队将消息 中继到正确的目标。

有时，如果目标队列是共享队列，而不是首先将消息传输到目标队列管理器，那么您可能希望本地队列管理器将消息直接放入目标队列。 您可以使用队列管理器属性 SQQMNAME 来控制此操作。 如果将 SQQMNAME 的值设置为 USE，那么将在 **ObjectQMgrName** 指定的队列管理器上执行 MQOPEN 命令。

但是，如果目标队列是共享队列，并且您将 SQQMNAME 的值设置为 IGNORE，并且 **ObjectQMgrName** 是队列共享组中另一个队列管理器的值，那么将在本地队列管理器上打开共享队列。 如果本地队列管理器无法打开目标队列或将消息放入队列，那么消息将通过 IGQ 或 IBM MQ 通道传输到指定的 **ObjectQMgrName**。

组内排队可用于更有效地将小消息传递到队列共享组中远程队列管理器上的队列。 组内排队还支持大型消息，最大为 100 MB 减去 传输队列头的长度。

注: 如果使用此功能，那么用户必须对队列共享组中每个队列管理器上的队列具有相同的访问权。

下图显示了组内排队的典型示例。

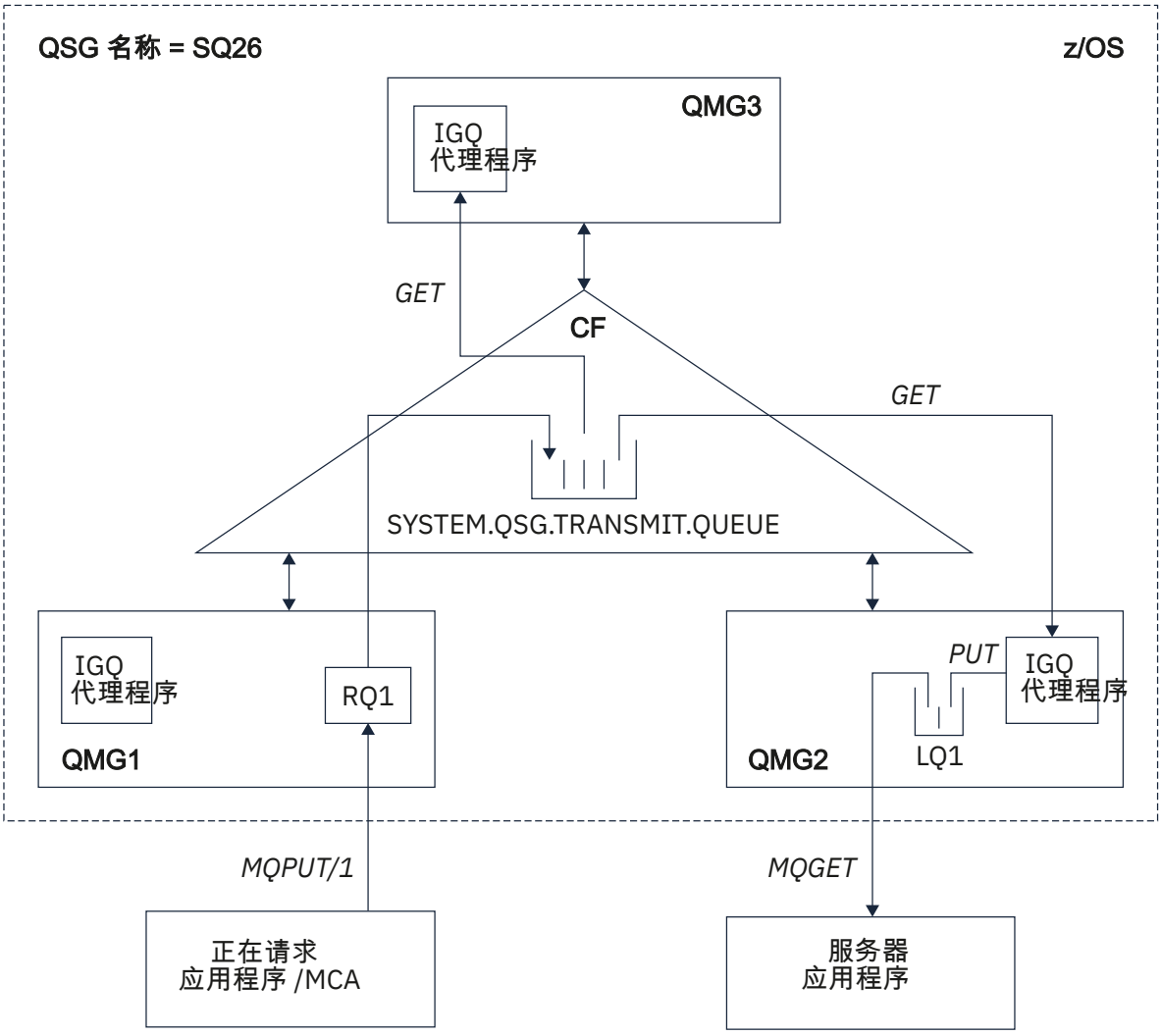


图 *67:* 组内排队的示例

图显示:

- 在定义到名为 SQ26 的队列共享组的三个队列管理器 (QMG1， QMG2 和 QMG3) 上运行的 IGQ 代理程序。

- 共享传输队列 SYSTEM.QSG.TRANSMIT.QUEUE 。
- 在队列管理器 QMG1 中定义的远程队列定义。
- 在队列管理器 QMG2 中定义的本地队列。
- 请求应用程序 (此应用程序可能是连接到队列管理器 QMG1 的消息通道代理程序 (MCA))。
- 连接到队列管理器 QMG2 的服务器应用程序。
- 放置到 SYSTEM.QSG.TRANSMIT.QUEUE。

组内排队和组内排队代理程序

在队列管理器初始化期间启动 IGQ 代理程序。 当应用程序打开消息并将其放入远程队列时，本地队列管理器确定是否将组内排队用于消息传输。 如果要使用组内排队，那么本地队列管理器会将消息放入 SYSTEM.QSG.TRANSMIT.QUEUE。 目标远程队列管理器上的 IGQ 代理程序检索消息并将其放入目标队列。

组内排队术语

术语的解释: 组内排队，供组内排队使用的共享传输队列，以及组内排队代理。

组内排队

组内排队可能会影响队列共享组中的队列管理器之间可能快速且成本较低的消息传输，而无需定义通道。

供组内排队使用的共享传输队列

每个队列共享组都有一个名为 SYSTEM.QSG.TRANSMIT.QUEUE ，供组内排队使用。 如果启用了组内排队，那么 SYSTEM.QSG.TRANSMIT.QUEUE 将显示在名称解析路径中。 当应用程序 (包括消息通道代理程序 (MCA)) 将消息放入远程队列时，本地队列管理器将确定消息的快速传输资格，并将其放在 SYSTEM.QSG.TRANSMIT.QUEUE。

组内排队代理程序

IGQ 代理程序是在队列管理器初始化时启动的任务，用于等待合适的消息到达 SYSTEM.QSG.TRANSMIT.QUEUE。 IGQ 代理程序从此队列中检索适当的消息，并将它们传递到目标队列。

每个队列管理器的 IGQ 代理程序始终启动，因为组内排队由队列管理器本身用于其自己的内部处理。

z/OS 组内排队的好处

组内排队的好处包括: 减少了系统定义，减少了系统管理，提高了性能，支持在队列共享组中的队列管理器之间进行多跳时迁移和传递消息。

组内排队的好处有:

减少的系统定义
组内排队无需在队列共享组中的队列管理器之间定义通道。

减少系统管理
由于队列共享组中的队列管理器之间未定义任何通道，因此不需要进行通道管理。

提高性能
因为只有一个 IGQ 代理需要将消息传递到目标队列 (而不是两个中间发送方和接收方代理) ，所以使用组内排队传递消息的成本可能低于使用通道传递消息的成本。 在组内排队中，只有一个接收组件，因为已除去对发送组件的需求。 此保存是因为消息可供目标队列管理器上的 IGQ 代理在本地队列管理器上的放置操作完成后传递到目标队列，并且在消息放入同步点作用域的情况下，已落实。

支持迁移

队列共享组外部的应用程序可以将消息传递到队列共享组中任何队列管理器上的队列，而仅连接到队列共享组中的特定队列管理器。 这是因为到达接收方通道 (针对远程队列管理器上的队列) 的消息可以使用组内排队以透明方式发送到目标队列。 此工具允许在队列共享组之间部署应用程序，而无需更改队列共享组外部的任何系统。

下图说明了典型配置，其中:

- 连接到队列管理器 QMG1 的请求应用程序需要将消息发送到队列管理器 QMG3 上的本地队列。
- 队列管理器 QMG1 仅连接到队列管理器 QMG2。
- 先前使用通道连接的队列管理器 QMG2 和 QMG3 现在是队列共享组 SQ26 的成员。

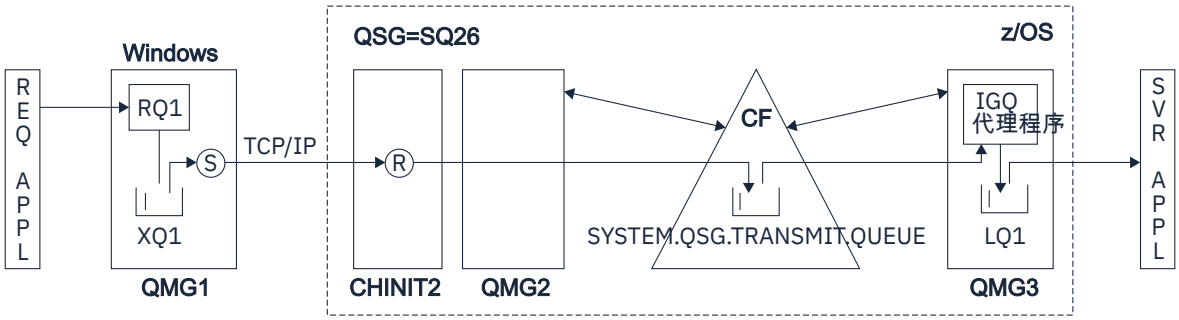


图 *68:* 迁移支持示例

操作流程如下所示:

1. 请求应用程序将以远程队列管理器 QMG3 上的本地队列 LQ1 为目标的消息放入远程队列定义 RQ1。
2. 在 Windows NT 工作站上运行的队列管理器 QMG1 将消息放入传输队列 XQ1。
3. QM1 上的发送方 MCA (S) 使用 TCP/IP 将消息传输到通道启动程序 CHINIT2 上的接收方 MCA (R)。
4. 通道启动程序 CHINIT2 上的接收方 MCA (R) 将消息放入共享传输队列 SYSTEM.QSG.TRANSMIT.QUEUE。
5. 队列管理器 QMG3 上的 IGQ 代理程序从 SYSTEM.QSG.TRANSMIT.QUEUE 并将其放入目标本地队列 LQ1。
6. 服务器应用程序从目标本地队列检索消息并对其进行处理。

在队列共享组中的队列管理器之间进行多跳时传递消息

支持迁移 中的上图还说明了在队列共享组中的队列管理器之间进行多跳时传递消息的情况。 通过使用组内排队，可以轻松将到达队列共享组中队列管理器的消息 (但指定用于队列共享组中另一个队列管理器上的队列) 传输到目标队列管理器上的目标队列。

z/OS 组内排队的限制

组内排队的限制包括: 有资格使用组内排队进行传输的消息，每个队列管理器的组内排队代理程序数，以及启动和停止组内排队代理程序。

本主题描述组内排队的限制。

有资格使用组内排队传输的消息

由于组内排队使用耦合设施 (CF) 中定义的共享传输队列，因此组内排队仅限于传递共享队列的最大受支持消息长度减去传输队列头 (MQXQH) 长度的消息。

每个队列管理器的组内排队代理程序数

对于队列共享组中的每个队列管理器，仅启动一个 IGQ 代理程序。

启动和停止组内排队代理程序

IGQ 代理程序在队列管理器初始化期间启动，并在队列管理器关闭期间终止。 它被设计为长时间运行，自我恢复 (在异常终止的情况下)，任务。 如果 SYSTEM.QSG.TRANSMIT.QUEUE (例如，如果此队列为 "禁止获取")，那么 IGQ 代理程序将继续重试。 如果 IGQ 代理程序迁到导致在队列管理器仍处于活动状态时代理程序正常终止的错误，那么可以通过发出 ALTER QMGR IGQ (ENABLED) 命令来重新启动该代理程序。 此命令可避免需要重新启动队列管理器。

将队列管理器属性 IGQ 设置为 ENABLED 或 DISABLED

如果队列管理器属性 IGQ 设置为 ENABLED 或 DISABLED，那么现有对象句柄可能会失效，原因码为 MQRC_OBJECT_CHANGED。 请参阅 组内排队入门 以获取更多信息。

z/OS 组内排队入门

您可以启用，禁用和使用组内排队，如本主题中所述。

启用组内排队

要在队列管理器上启用组内排队，需要执行以下操作:

- 定义名为 SYSTEM.QSG.TRANSMIT.QUEUE。 此队列的定义可以在 thlqual.SCSQPROC(CSQ4INSS) 中找到，即用于队列共享组的 SYSTEM 对象的 CSQINP2 样本。 必须使用正确的属性来定义此队列，如 thlqual.SCSQPROC(CSQ4INSS) 中所述，才能使组内排队正常工作。
- 由于 IGQ 代理程序始终在队列管理器初始化时启动，因此组内排队始终可用于入站消息处理。 IGQ 代理程序处理放在 SYSTEM.QSG.TRANSMIT.QUEUE。 但是，要对出站处理启用组内排队，必须将队列管理器属性 IGQ 设置为 ENABLED。

  要点: 如果队列管理器属性 IGQ 设置为 ENABLED，那么现有对象句柄可能会因原因码 MQRC_OBJECT_CHANGED 而失效。 请参阅 第 183 页的『组内排队的特定属性』 以获取更多信息。 如此原因码的 "程序员响应" 部分中所述，需要对应用程序进行编码以处理此情况 (请参阅 2041 (07F9) (RC2041): MQRC_OBJECT_CHANGED 以获取更多详细信息)。

此外，由于 IGQ 设计为长时间运行的自恢复任务，在初始化期间启动并在关闭时终止，请参阅 第 176 页的『组内排队的限制』 以获取更多信息。

禁用组内排队

要对出站消息传输禁用组内排队，请将队列管理器属性 IGQ 设置为 DISABLED。 如果对特定队列管理器禁用组内排队，那么该队列管理器上的 IGQ 代理程序仍可以处理已放置在 SYSTEM.QSG.TRANSMIT.QUEUE，该队列管理器已启用组内排队以进行出站传输。

要点: 如果队列管理器属性 IGQ 设置为 ENABLED，那么现有对象句柄可能会因原因码 MQRC_OBJECT_CHANGED 而失效。 请参阅 第 183 页的『组内排队的特定属性』 以获取更多信息。 如此原因码的 "程序员响应" 部分中所述，需要对应用程序进行编码以处理此情况 (请参阅 2041 (07F9) (RC2041): MQRC_OBJECT_CHANGED 以获取更多详细信息)。

此外，由于 IGQ 设计为长时间运行的自恢复任务，在初始化期间启动并在关闭时终止，请参阅 第 176 页的『组内排队的限制』 以获取更多信息。

使用组内排队

一旦启用了组内排队，就可以使用该队列，并且队列管理器将尽可能使用该队列。 即，当应用程序将消息放入远程队列定义，放入标准远程队列或放入集群队列时，队列管理器将确定消息是否符合使用组内排队传递的条件，如果符合，那么将消息放入 SYSTEM.QSG.TRANSMIT.QUEUE。 不需要更改用户应用程序或应用程序队列，因为对于符合条件的消息，队列管理器使用 SYSTEM.QSG.TRANSMIT.QUEUE，优先于任何其他传输队列。

z/OS 组内排队配置

除了典型的组内排队配置外，还可以进行其他配置。

第 173 页的『组内排队概念』 描述了典型配置。

相关概念

第 177 页的『分布式排队与组内排队 (多个交付路径)』
对于处理短消息的应用程序，可能可以配置组内排队，仅用于在队列共享组中的队列管理器之间传递消息。

第 179 页的『集群与组内排队 (多个交付路径)』
可以配置队列管理器，以便它们位于集群中以及队列共享组中。

第 181 页的『集群，组内排队和分布式排队』
可以配置作为集群成员以及队列共享组的队列管理器，并使用发送方/接收方通道对连接到分布式队列管理器。

z/OS 分布式排队与组内排队 (多个交付路径)

对于处理短消息的应用程序，可能可以配置组内排队，仅用于在队列共享组中的队列管理器之间传递消息。

通过通道通信进行组内排队的选择可由 CFSTRUCT 类型级别控制。 (3 而不是 4 或 5)。 在 SYSTEM.QSQ.TRANSMIT.QUEUE。

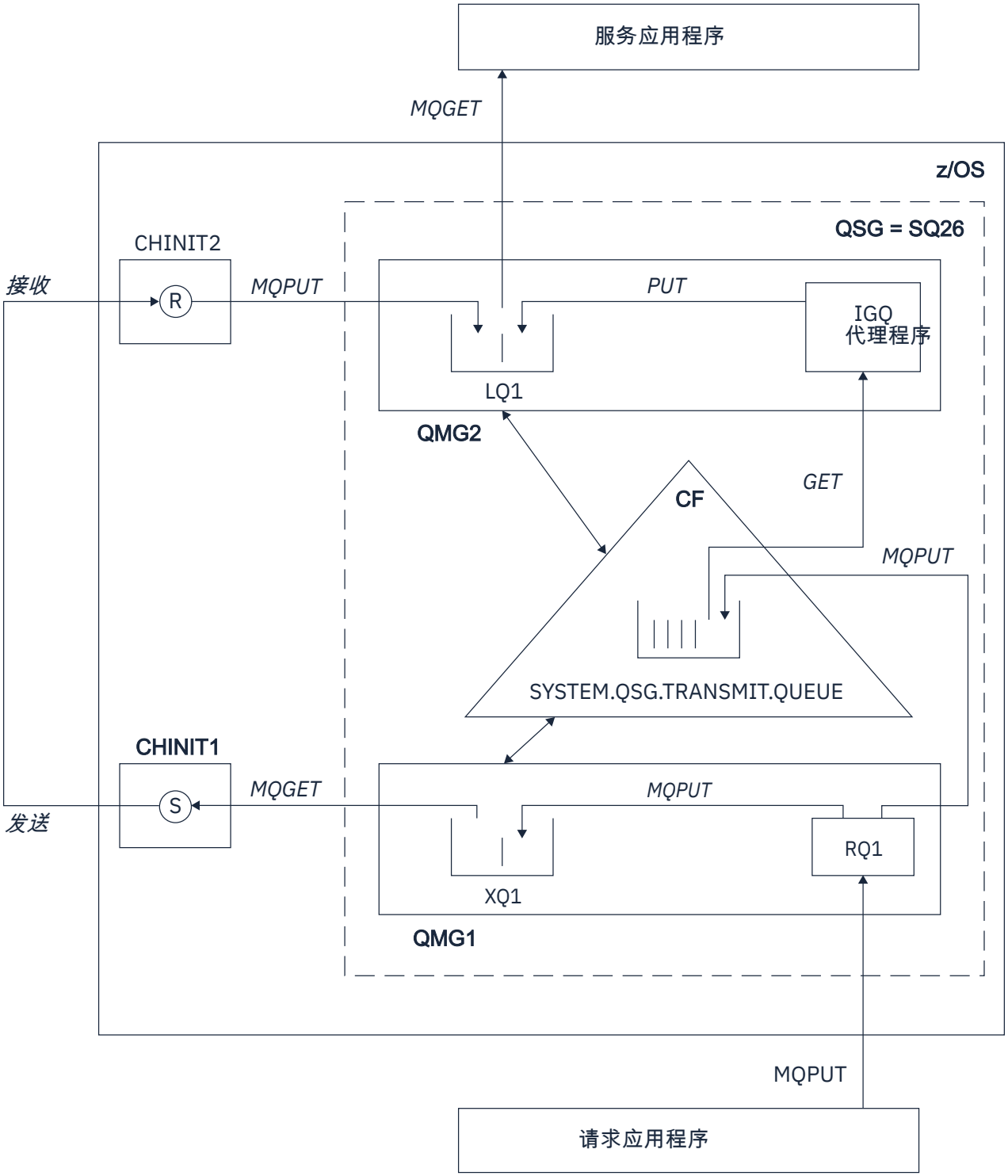


图 *69:* 示例配置

打开/放入处理

1. 请务必注意，当发出请求的应用程序打开远程队列 RQ1 时，将对非共享传输队列 XQ1 和共享传输队列 SYSTEM.QSG.TRANSMIT.QUEUE。

2. 当请求应用程序将消息放入远程队列时，根据是否对队列管理器上的出站传输启用组内排队以及消息特征，将消息放入传输队列 XQ1 或传输队列 SYSTEM.QSG.TRANSMIT.QUEUE。 队列管理器将所有大型消息放在传输队列 XQ1 上，并将所有小型消息放在传输队列 SYSTEM.QSG.TRANSMIT.QUEUE。
3. 如果传输队列 XQ1 已满或不可用，那么针对大型消息的放入请求将同步失败，并返回适当的返回码和原因码。 但是，针对小消息的放入请求将继续成功，并将其放入传输队列 SYSTEM.QSG.TRANSMIT.QUEUE。
4. 如果传输队列为 SYSTEM.QSG.TRANSMIT.QUEUE 已满，或者无法放入，针对小型消息的放入请求将同步失败，并返回适当的返回码和原因码。 但是，针对大型消息的放入请求将继续成功，并将其放入传输队列 XQ1。 在这种情况下，不会尝试将小消息放入传输队列。

大消息流

1. 请求应用程序将大型消息放入远程队列 RQ1。
2. 队列管理器 QMG1 将消息放入传输队列 XQ1。
3. 队列管理器 QMG1 上的发送方 MCA (S) 从传输队列 XQ1 检索消息，并将它们发送到队列管理器 QMG2。
4. 队列管理器 QMG2 上的接收方 MCA (R) 接收消息并将其放入目标队列 LQ1。
5. 服务应用程序从队列 LQ1 检索消息，然后进行处理。

小消息流

1. 请求应用程序将小消息放入远程队列 RQ1。
2. 队列管理器 QMG1 将消息放入传输队列 SYSTEM.QSG.TRANSMIT.QUEUE。
3. 队列管理器 QMG2 上的 IGQ 检索消息并将其放入目标队列 LQ1。
4. 服务应用程序从队列 LQ1 检索消息。

要考虑的要点

1. 请求应用程序不需要知道用于传递消息的底层机制。
2. 可以为小消息实现潜在的更快的消息传递机制。
3. 多个路径可用于消息传递 (即，正常通道路由和组内排队路由)。
4. 选择组内排队路径 (可能更快) 优先于正常通道路径。 根据消息特征，消息传递可能分为两条路径。 因此，消息可能会按顺序传递 (尽管如果仅使用正常通道路由传递消息，那么也可以进行此传递)。
5. 当选择了路由并且已将消息放入传输队列时，仅将所选路由用于消息传递。 SYSTEM.QSG.TRANSMIT.QUEUE 未转至传输队列 XQ1。

z/OS 集群与组内排队 (多个交付路径)

可以配置队列管理器，以便它们位于集群中以及队列共享组中。

当消息发送到集群队列并且本地和远程目标队列管理器位于同一队列共享组中时，组内排队用于传递小消息 (使用 SYSTEM.QSG.TRANSMIT.QUEUE) ，以及在组内排队支持消息大小的情况下传递大型消息。 另外，SYSTEM.CLUSTER.TRANSMIT.QUEUE 用于将消息传递到集群中但在队列共享组之外的任何队列管理器。 下图说明了此配置 (未显示通道启动程序)。

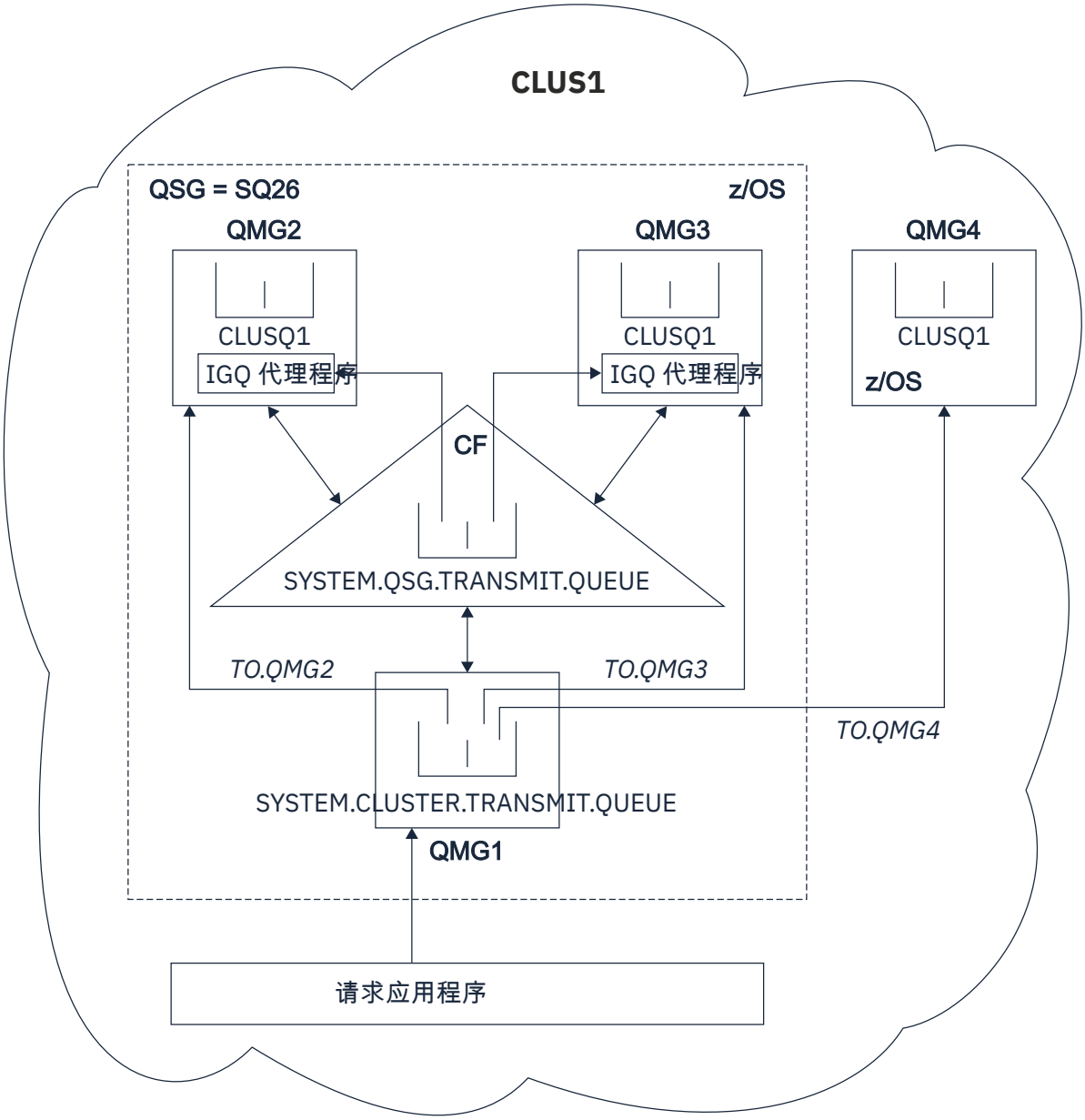


图 *70:* 使用组内排队的集群示例

图显示:

- 集群 CLUS1 中配置了四个 z/OS 队列管理器 QMG1， QMG2， QMG3 和 QMG4 。
- 队列共享组 SQ26 中配置的队列管理器 QMG1， QMG2 和 QMG3 。
- 在队列管理器 QMG2 和 QMG3 上运行的 IGQ 代理程序。
- 在 QMG1 中定义的本地 SYSTEM.CLUSTER.TRANSMIT.QUEUE 。

  注: 为了清晰起见，请参阅 SYSTEM.CLUSTER.TRANSMIT.QUEUE 。
- 共享 SYSTEM.QSG.TRANSMIT.QUEUE ， 它位于使用 CFLEVEL ($TAG3) RECOVER (YES) 属性配置的 IBM MQ 结构中。
- 集群通道 TO.QMG2 (将 QMG1 连接到 QMG2) ， TO.QMG3 (将 QMG1 连接到 QMG3) 和 TO.QMG4 (将 QMG1 连接到 QMG4)。
- 正在队列管理器 QMG2， QMG3 和 QMG4 上托管集群队列 CLUSQ1 。

假定发出请求的应用程序使用 MQOO_BIND_NOT_FIXED 选项打开集群队列，以便在放置时选择集群队列的目标队列管理器。

如果所选目标队列管理器为 QMG2:

- 请求应用程序放入的所有大型消息为:
  - 放入 SYSTEM.CLUSTER.TRANSMIT.QUEUE on QMG1，因为 SYSTEM.QSG.TRANSMIT.QUEUE 处于 CFLEVEL (3) 结构中; 因此仅支持最大大小为 63 KB 的消息。
  - 已使用集群通道 TO.QMG2 传输到 QMG2 上的集群队列 CLUSQ1
- 请求应用程序放入的所有小消息都是
  - 放入共享传输队列 SYSTEM.QSG.TRANSMIT.QUEUE。 此队列采用配置了 RECOVER (YES) 属性的结构，因此用于持久和非持久的小消息。
  - 由 QMG2 上的 IGQ 代理程序检索
  - 放入 QMG2 上的集群队列 CLUSQ1

如果所选目标队列管理器为 QMG4:

- 由于 QMG4 不是队列共享组 SQ26 的成员，因此请求应用程序放入的所有消息都是
  - 放入 QMG1 上的 SYSTEM.CLUSTER.TRANSMIT.QUEUE
  - 已使用集群通道 TO.QMG4 传输到 QMG4 上的集群队列 CLUSQ1

要考虑的要点

- 请求应用程序不需要知道用于传递消息的底层机制。
- 对于在队列共享组中的队列管理器之间传输小型非持久消息 (即使同一队列管理器位于集群中)，可能会实现更快的传递机制。
- 多个路径可用于消息传递 (即，集群路由和组内排队路由)。
- 选择组内排队路径 (可能更快) 优先于集群路径。 根据消息特征，消息传递可能分为两条路径。 因此，可能会按顺序传递消息。 请务必注意，此传递是可能的，而不考虑应用程序指定的 MQOO_BIND_ * 选项。 根据是在打开时指定 MQOO_BIND_NOT_FIXED， MQOO_BIND_ON_OPEN， MQOO_BIND_ON_GROUP 还是 MQOO_BIND_AS_Q_DEF，组内排队以集群相同的方式分发消息。
- 当选择了路由并且已将消息放入传输队列时，仅将所选路由用于消息传递。 SYSTEM.QSG.TRANSMIT.QUEUE 不会转移到 SYSTEM.CLUSTER.TRANSMIT.QUEUE。

z/OS 集群，组内排队和分布式排队

可以配置作为集群成员以及队列共享组的队列管理器，并使用发送方/接收方通道对连接到分布式队列管理器。

此配置是分布式排队与组内排队以及集群与组内排队的组合。

第 177 页的『分布式排队与组内排队 (多个交付路径)』中描述了组内排队。

第 179 页的『集群与组内排队 (多个交付路径)』中描述了使用组内排队的集群。

z/OS 组内排队消息

本部分描述了放入 SYSTEM.QSG.TRANSMIT.QUEUE。

消息结构

与放入传输队列的所有其他消息一样，放入 SYSTEM.QSG.TRANSMIT.QUEUE 以传输队列头 (MQXQH) 作为前缀。

消息持久性

在 IBM WebSphere MQ 5.3 及更高版本中，共享队列支持持久消息和非持久消息。

如果队列管理器在 IGQ 代理程序处理非持久消息时终止，或者如果 IGQ 代理程序在处理消息期间异常终止，那么正在处理的非持久消息可能会丢失。 如果需要恢复非持久消息，应用程序必须作出恢复安排。

如果 IGQ 代理程序发出的非持久消息的放置请求意外失败，那么正在处理的消息将丢失。

传递消息
IGQ 代理程序检索并传递同步点作用域外的所有非持久消息以及同步点作用域内的所有持久消息。 在这种情况下， IGQ 代理程序充当同步点协调程序。 因此， IGQ 代理程序会处理非持久消息，例如在消息通道上快速处理非持久消息的方式。 请参阅 快速非持久消息 (Fast， nonpersistent messages)。

消息批处理
IGQ 代理程序使用 50 条消息的固定批处理大小。 在批处理中检索到的任何持久消息将按 50 条消息的时间间隔落实。 当 SYSTEM.QSG.TRANSMIT.QUEUE。

消息大小
可以放入 SYSTEM.QSG.TRANSMIT.QUEUE 是共享队列支持的最大消息长度减去传输队列头 (MQXQH) 的长度。

缺省消息持久性和缺省消息优先级
如果是 SYSTEM.QSG.TRANSMIT.QUEUE 位于在打开时建立的队列名称解析路径中，然后对于具有缺省持久性和缺省优先级 (或者具有缺省持久性或缺省优先级) 的消息，将在选择具有所使用的缺省优先级和持久性值的队列时应用常规规则。 (请参阅 IBM MQ 消息 部分，以获取有关队列选择规则的更多信息)。

相关概念
第 182 页的『未传递/未处理的消息』
本主题描述在 SYSTEM.QSG.TRANSMIT.QUEUE。

第 182 页的『报告消息-组内排队』
本主题描述报告消息: 到达确认， 交付确认，到期报告和异常报告。

z/OS 未传递/未处理的消息

本主题描述在 SYSTEM.QSG.TRANSMIT.QUEUE。

如果 IGQ 代理程序无法将消息传递到目标队列，那么 IGQ 代理程序:

- 采用 MQRO_DISCARD_MSG 报告选项 (如果未传递的消息的 MQMD 的 "报告选项" 字段指示它必须) 并废弃未传递的消息。
- 尝试将未传递的消息放在目标队列管理器的死信队列上 (如果尚未废弃该消息)。 IGQ 代理程序以死信队列头 (MQDLH) 作为消息的前缀。

如果未定义死信队列，或者如果无法将未传递的消息放入死信队列，并且未传递的消息为:

- 持久， IGQ 代理程序将其正在处理的当前持久消息批处理回退，并进入重试状态。 有关更多信息，请参阅第 183 页的『组内排队的特定属性』。
- 非持久， IGQ 代理程序将废弃该消息并继续处理下一条消息。

如果队列共享组中的队列管理器在其关联的 IGQ 代理程序有时间处理其所有消息之前终止，那么未处理的消息将保留在 SYSTEM.QSG.TRANSMIT.QUEUE ， 直到下次启动队列管理器为止。 然后， IGQ 代理程序会检索消息并将其传递到目标队列。

如果耦合设施在 SYSTEM.QSG.TRANSMIT.QUEUE ， 任何未处理的非持久消息都将丢失。

IBM 建议应用程序不要将消息直接放入传输队列。 如果应用程序将消息直接放入 SYSTEM.QSG.TRANSMIT.QUEUE， IGQ 代理程序可能无法处理这些消息，它们将保留在 SYSTEM.QSG.TRANSMIT.QUEUE。 然后，用户必须使用自己的方法来处理这些未处理的消息。

z/OS 报告消息-组内排队

本主题描述报告消息: 到达确认， 交付确认，到期报告和异常报告。

确认到达 (COA)/确认交付 (COD) 报告消息
使用组内排队时，队列管理器会生成 COA 和 COD 消息。

到期报告消息
到期报告消息由队列管理器生成。

异常报告消息

根据未传递消息的消息描述符的 报告选项 字段中指定的 MQRO_EXCEPTION_ * 报告选项， IGQ 代理程序将生成所需的异常报告并将其放置在指定的应答队列上。 组内排队可用于将异常报告传递到目标应答队列。

报告消息的持久性与未传递消息的持久性相同。 如果 IGQ 代理未能解析目标应答队列的名称，或者如果它未能将应答消息放入传输队列 (用于后续传输至目标应答队列) ，那么它会尝试将异常报告放入生成报告消息的队列管理器的死信队列。 如果不可能，那么如果未传递的消息为:

- 持久， IGQ 代理程序会废弃异常报告，回退当前消息批次，并进入重试状态。 有关更多信息，请参阅第 183 页的『组内排队的特定属性』。
- 非持久性， IGQ 代理程序将废弃异常报告并继续处理 SYSTEM.QSG.TRANSMIT.QUEUE。

z/OS 组内排队的安全性

本主题描述组内排队的安全安排。

可以设置队列管理器属性 IGQAUT (IGQ 权限) 和 IGQUSER (IGQ 代理程序用户标识) ， 以控制当 IGQ 代理程序打开目标队列时执行的安全性检查级别。

组内排队权限 (IGQAUT)

可以设置 IGQAUT 属性以指示要执行的安全性检查的类型，从而确定 IGQ 代理程序在建立将消息放入目标队列的权限时要使用的用户标识。

IGQAUT 属性类似于通道定义上可用的 PUTAUT 属性。

组内排队用户标识 (IGQUSER)

IGQUSER 属性可用于指定 IGQ 代理程序在建立将消息放入目标队列的权限时要使用的用户标识。

IGQUSER 属性类似于通道定义上可用的 MCAUSER 属性。

z/OS 组内排队的特定属性

本节描述了组内排队的特定属性，包括对象句柄的失效，组内排队代理程序的自恢复和重试功能以及组内排队代理程序和序列化。

对象句柄失效 (MQRC_OBJECT_CHANGED)

如果在打开对象后发现对象的属性已更改，那么队列管理器将在下次使用 MQRC_OBJECT_CHANGED 时使对象句柄失效。

组内排队引入了对象句柄失效的以下规则:

- 如果是 SYSTEM.QSG.TRANSMIT.QUEUE 包含在名称解析路径中，因为组内排队在打开时为 ENABLED ，但在放置时发现组内排队为 DISABLED ，那么队列管理器会使对象句柄失效，并使使用 MQRC_OBJECT_CHANGED 的放置请求失败。
- 如果是 SYSTEM.QSG.TRANSMIT.QUEUE 包含在名称解析路径中，因为组内排队在打开时已禁用，但发现组内排队在放置时已启用，那么队列管理器将使对象句柄失效，并使带有 MQRC_OBJECT_CHANGED 的放置请求失败。
- 如果是 SYSTEM.QSG.TRANSMIT.QUEUE 在打开处理期间包含在名称解析路径中，因为组内排队是在打开时启用的，但 SYSTEM.QSG.TRANSMIT.QUEUE 定义已按放置时间进行更改，那么队列管理器将使对象句柄失效，并使带有 MQRC_OBJECT_CHANGED 的放置请求失败。

组内排队代理的自我恢复

如果 IGQ 代理程序异常终止，那么将发出消息 CSQM067E ，并且 IGQ 代理程序将再次启动。

组内排队代理程序的重试功能

如果 IGQ 代理程序在访问 SYSTEM.QSG.TRANSMIT.QUEUE (例如，由于未定义此代理程序，或者未使用不正确的属性定义此代理程序，或者由于某些其他原因而禁止此代理程序)，因此 IGQ 代理程序将进入重试状态。

IGQ 代理程序会观察短期和长期重试计数和时间间隔。 这些计数和时间间隔的值 (不可更改) 如下所示:

表 *20:* 短重试次数和长重试次数以及时间间隔值

常量	**值**
短重试次数	10
短重试时间间隔	60 秒 = 1 分钟
长重试次数	999,999,999
长重试时间间隔	1200 秒 = 20 分钟

组内排队代理程序和序列化

当对等恢复仍在进行时， IGQ 代理程序尝试序列化对共享队列的访问可能会失败。

如果在 IGQ 代理程序处理一个或多个共享队列上未落实的消息时，队列共享组中的队列管理器发生故障，那么 IGQ 代理程序将结束，并且将对发生故障的队列管理器进行共享队列对等恢复。 由于共享队列对等恢复是一项异步活动，因此在共享队列对等恢复完成之前，可能会使失败的队列管理器以及该队列管理器的 IGQ 代理程序重新启动。 这将使任何已落实的消息都有可能在仍在恢复的情况下提前和不按顺序进行处理。 为了确保不按顺序处理消息， IGQ 代理程序通过发出 MQCONNX API 调用来序列化对共享队列的访问。

当对等恢复仍在进行时， IGQ 代理程序尝试序列化对共享队列的访问可能会失败。 发出错误消息，并将 IGQ 代理程序置于重试状态。 当队列管理器对等恢复完成时 (例如，在下次重试时)， IGQ 代理程序可以启动。

z/OS z/OS 上的存储管理

IBM MQ for z/OS 需要永久和临时数据结构，并使用页集和内存缓冲区来存储此数据。 这些主题提供了有关 IBM MQ 如何使用这些页集和缓冲区的更多详细信息。

相关概念

第 184 页的『IBM MQ for z/OS 的页集』
使用本主题来了解 IBM MQ for z/OS 如何使用页面集来存储消息。

第 185 页的『IBM MQ for z/OS 的存储类』
存储类是一个 IBM MQ for z/OS 概念，允许队列管理器将队列映射到页集。 您可以使用存储类来控制哪些数据集由哪些队列使用。

第 186 页的『IBM MQ for z/OS 的缓冲区和缓冲池』
IBM MQ for z/OS 使用缓冲区和缓冲池来临时高速缓存数据。 使用本主题来进一步了解如何组织和使用缓冲区。

相关参考

第 187 页的『在何处查找有关 IBM MQ for z/OS 的存储管理的更多信息』
使用本主题作为参考，以查找有关 IBM MQ for z/OS 的存储管理的更多信息。

z/OS IBM MQ for z/OS 的页集

使用本主题来了解 IBM MQ for z/OS 如何使用页面集来存储消息。

页集 是经过特殊格式化以供 IBM MQ 使用的 VSAM 线性数据集。 页集用于存储大多数消息和对象定义。

这方面的例外是全局定义，这些定义存储在 Db2 上的共享存储库中，以及共享队列上的消息。 这些未存储在队列管理器页集上。 有关共享队列的更多信息，请参阅 第 137 页的『共享队列和队列共享组』，有关全局定义的更多信息，请参阅 专用和全局定义。

IBM MQ 页集的大小可达 64 GB。 每个页集都由页集标识 (PSID) 标识，该标识是 00 到 99 范围内的整数。 每个队列管理器都必须具有自己的页集。

IBM MQ 使用页集零 (PSID=00) 来存储与队列管理器相关的对象定义和其他重要信息。 对于 IBM MQ 的正常操作，至关重要的是页集零不会变满，因此请勿将其用于存储消息。

为了提高系统的性能，您还应该通过将短命消息与长命消息放在不同的页集中来将它们分开。

您必须格式化页集， IBM MQ 为此提供了 FORMAT 实用程序; 请参阅 格式化页集 (FORMAT)。 还必须向 IBM MQ 子系统定义页集。

可以将 IBM MQ for z/OS 配置为在页集变满时动态展开页集。 如果需要， IBM MQ 将继续扩展页集，直到存在 123 个逻辑扩展数据块 (如果有足够的可用磁盘存储空间) 为止。 如果以这种方式定义线性数据集，那么扩展数据块可以跨越卷，但是 IBM MQ 无法将页集扩展至超过 64 GB。

不能使用来自另一个 IBM MQ 队列管理器上的一个 IBM MQ 队列管理器的页集，也不能更改队列管理器名称。 如果要将数据从一个队列管理器传输到另一个队列管理器，那么必须从第一个队列管理器卸载所有对象和消息，然后将它们重新装入到另一个队列管理器。

在运行低于 V6 的发行版的队列管理器中，无法使用大于 4 GB 的页集。 在迁移期间，当您可能需要回退到先前发行版的代码时:

- 请勿将页集 0 更改为大于 4 GB。
- 在使用先前发行版重新启动队列管理器时，大于 4 GB 的其他页集将保持脱机状态。

有关迁移能够扩展至 4 GB 以外的现有页集的更多信息，请参阅 将页集定义为大于 4 GB。

管理员可以动态地将页集添加到正在运行的队列管理器，或者从正在运行的队列管理器中除去页集 (页集为零除外)。 仅当 DEFINE PSID 命令包含 DSN 关键字时，该命令才能在队列管理器重新启动完成后运行。

z/OS IBM MQ for z/OS 的存储类

存储类是一个 IBM MQ for z/OS 概念，允许队列管理器将队列映射到页集。 您可以使用存储类来控制哪些数据集由哪些队列使用。

存储类简介

存储类 将一个或多个队列映射到页集。 这意味着该队列的消息存储在该页集上。

存储类允许您控制存储非共享消息数据的位置，以用于管理，数据集空间和负载管理或应用程序隔离目的。 如果您正在使用 IMS 网桥 (如 第 231 页的『IBM MQ 和 IMS』 中所述)，那么还可以使用存储类来定义 IMS 区域的 XCF 组和成员名。

共享队列不使用存储类来获取页集映射，因为它们上的消息未存储在页集上。

存储类的工作方式

- 使用 DEFINE STGCLASS 命令定义存储类，并指定页集标识 (PSID)。
- 定义队列时，请在 STGCLASS 属性中指定存储类。

在以下示例中，本地队列 QE5 通过存储类 ARC2 映射到页集 21。

```
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE QLOCAL(QE5) STGCLASS(ARC2)
```

这意味着放置在队列 QE5 上的消息将存储在页集 21 上 (如果它们在队列上的长度足以写入 DASD)。

多个队列可以使用相同的存储类，您可以根据需要定义任意数量的存储类。 例如，您可以扩展先前示例以包含更多存储类和队列定义，如下所示:

```
DEFINE STGCLASS(ARC1) PSID(05)
DEFINE STGCLASS(ARC2) PSID(21)
DEFINE STGCLASS(MAXI) PSID(05)
DEFINE QLOCAL(QE1) STGCLASS(ARC1) ...
DEFINE QLOCAL(QE2) STGCLASS(ARC1) ...
DEFINE QLOCAL(QE3) STGCLASS(MAXI) ...
DEFINE QLOCAL(QE4) STGCLASS(ARC2) ...
DEFINE QLOCAL(QE5) STGCLASS(ARC2) ...
```

在 第 186 页的图 71 中，两个存储类 ARC1 和 MAXI 都与页集 05 相关联。 因此，队列 QE1， QE2 和 QE3 映射到页集 05。 同样，存储类 ARC2 将队列 QE4 和 QE5 与页集 21 相关联。

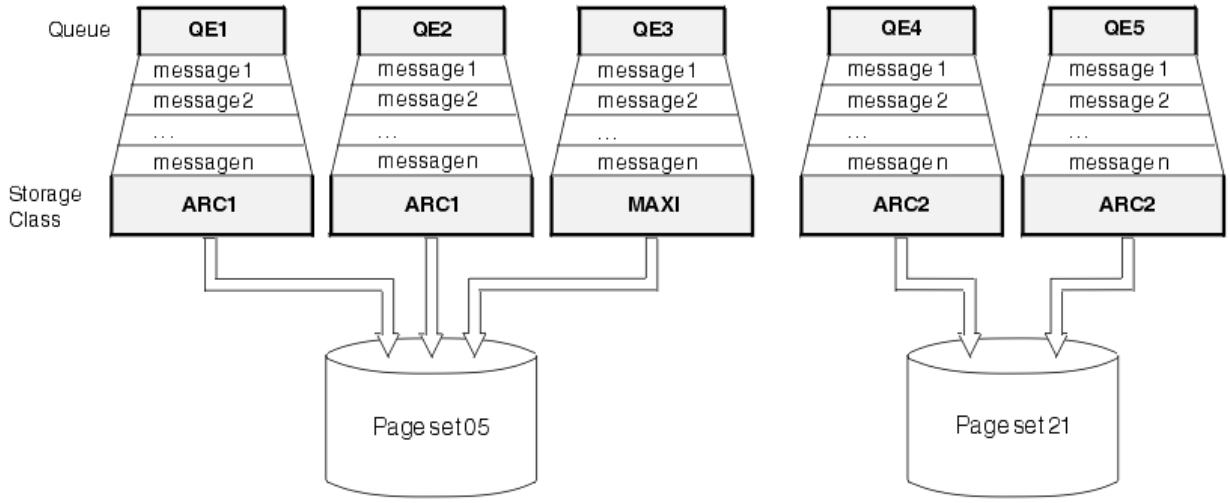


图 *71:* 通过存储类将队列映射到页集

如果在未指定存储类的情况下定义队列，那么 IBM MQ 将使用缺省存储类。

如果将消息放在指定了不存在的存储类的队列上，那么应用程序将接收到错误。 必须更改队列定义以向其提供现有存储类名，或者创建由队列命名的存储类。

仅当下列情况下，才能更改存储类:

- 使用此存储类的所有队列都为空，并且没有未落实的活动。
- 将关闭所有使用此存储类的队列。

z/OS IBM MQ for z/OS 的缓冲区和缓冲池

IBM MQ for z/OS 使用缓冲区和缓冲池来临时高速缓存数据。 使用本主题来进一步了解如何组织和使用缓冲区。

为了提高效率， IBM MQ 使用了一种高速缓存形式，即消息 (和对象定义) 临时存储在缓冲区中，然后存储在 DASD 上的页集中。 短命消息 (即，在接收消息后不久从队列中检索的消息) 可能只会存储在缓冲区中。 此高速缓存活动由作为 IBM MQ 组件的缓冲区管理器控制。

这些缓冲区组织成 缓冲池。 最多可以为每个队列管理器定义 100 个缓冲池 (0 到 99)。

建议您使用与 第 187 页的图 72 中概述的对象和消息类型分隔一致的最小缓冲池数，以及应用程序可能具有的任何数据隔离需求。 每个缓冲区的长度为 4 KB。 缺省情况下，缓冲池使用 31 位存储器，在此方式下，最大缓冲区数由队列管理器地址空间中可用的 31 位存储器量确定; 对于缓冲区，请勿使用超过约 70% 的存储空间。 或者，可以从 64 位存储器进行缓冲池存储器分配 (使用 **DEFINE BUFFPOOL** 命令的 LOCATION 属性)。 使用 LOCATION (上面) 以便使用 64 位存储器有两个好处。 首先，有更多的 64 位存储器可用，因此缓

冲池可以大得多，其次， 31 位存储器可供其他功能使用。 通常，您拥有的缓冲区越多，缓冲效率越高，IBM MQ 的性能越好。

第 187 页的图 72 显示了消息，缓冲区，缓冲池和页集之间的关系。 缓冲池与一个或多个页集相关联; 每个页集与单个缓冲池相关联。

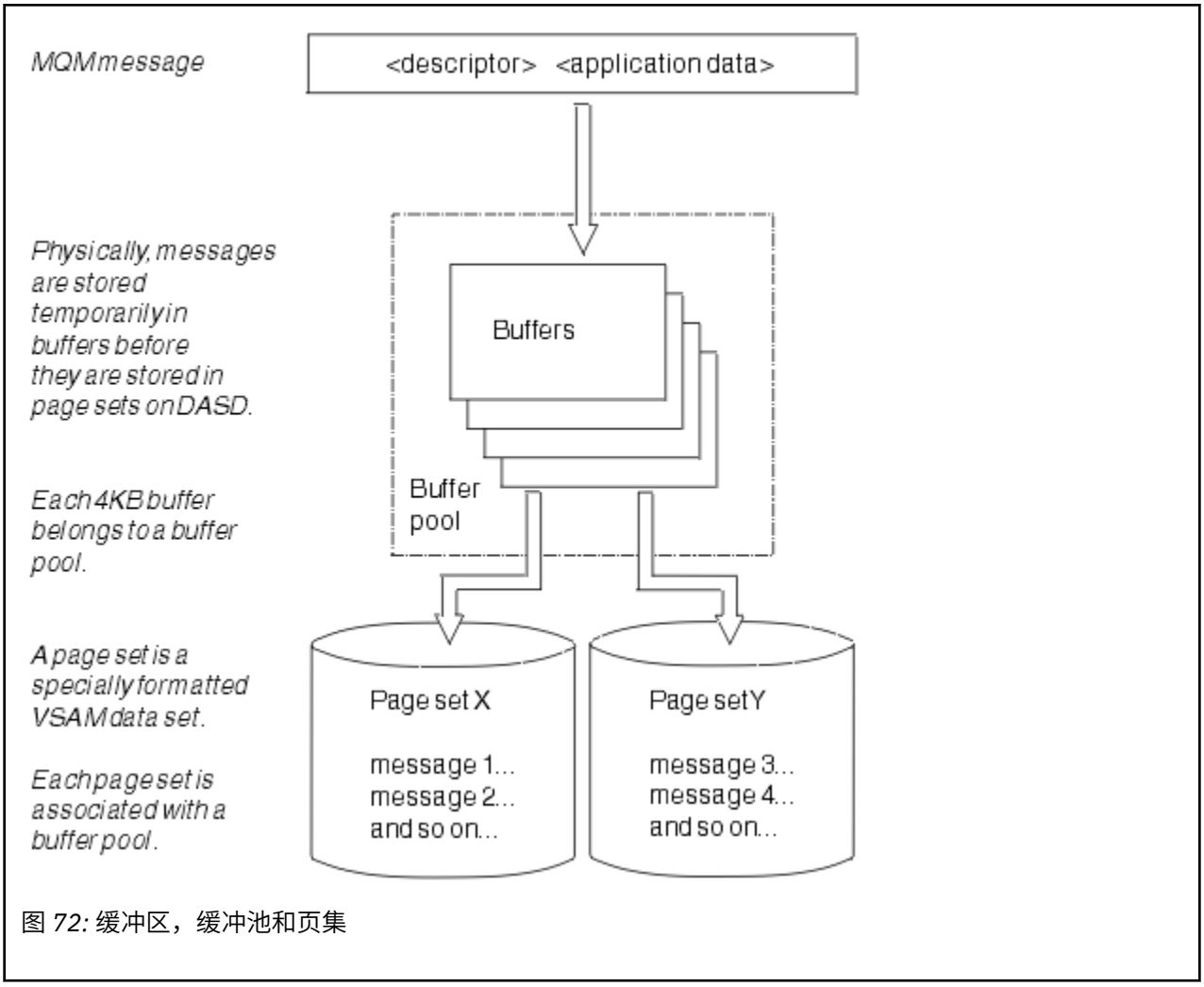


图 72: 缓冲区，缓冲池和页集

您可以使用 **ALTER BUFFPOOL** 命令动态发出命令以修改缓冲池大小和位置。 可以使用 **DEFINE PSID** 命令动态添加页集，也可以使用 **DELETE PSID** 命令删除页集。

如果缓冲池太小，那么 IBM MQ 会发出消息 CSQP020E。 然后，可以动态地向受影响的缓冲池添加更多缓冲区 (请注意，您可能必须从其他缓冲池中除去缓冲区才能执行此操作)。

您可以使用 **DEFINE BUFFPOOL** 命令指定池中的缓冲区数，并且可以使用 **ALTER BUFFPOOL** 命令动态调整缓冲池的大小。 通过使用 **DISPLAY USAGE** 命令显示使用缓冲池的页集，可动态确定池中的当前缓冲区数。

出于性能原因，请勿将消息和对象定义放在同一缓冲池中。 将一个缓冲池 (例如，数字零) 专门用于保留对象定义的页集零。 同样，将短命消息和长命消息保留在不同的缓冲池中，因此保留在不同的页集和不同的队列中。

重新启动后无法使用 **DEFINE BUFFPOOL** 命令来创建新的缓冲池。 相反，如果 **DEFINE PSID** 命令使用 DSN 关键字，那么它可以显式标识当前未定义的缓冲池。 然后将创建新的缓冲池。

z/OS 在何处查找有关 IBM MQ for z/OS 的存储管理的更多信息

使用本主题作为参考，以查找有关 IBM MQ for z/OS 的存储管理的更多信息。

您可以从以下源中找到有关此部分中主题的更多信息:

表 21: 在何处查找有关存储管理的更多信息	
Topic	**在哪里查找信息**
您需要多少存储空间	在 z/OS 上规划存储和性能需求
创建页面集的大小 和缓冲池	规划页集和缓冲池
管理页面集	管理页集
MQSC 命令	MQSC 命令

z/OS 登录 IBM MQ for z/OS

IBM MQ 维护发生数据更改和重要事件时的 日志 。 如果需要，可以使用这些日志将数据恢复到先前状态。

引导数据集 (BSDS) 存储有关包含日志的数据集的信息。

日志不包含有关统计信息，跟踪或性能评估的信息。 有关 IBM MQ 收集的统计信息和监视信息的更多详细信息，请参阅 监视和统计信息。

有关日志记录的更多信息，请参阅以下主题:

- 第 188 页的『IBM MQ for z/OS 中的日志文件』
- 第 191 页的『日志的构造方式』
- 第 192 页的『如何写入 IBM MQ for z/OS 日志』
- 第 195 页的『较大的日志相对字节地址』
- 第 196 页的『引导程序数据集』

相关任务

规划日志记录环境

使用系统参数模块设置日志

z/OS 管理 z/OS

相关参考

z/OS IBM MQ for z/OS 的消息

z/OS IBM MQ for z/OS 中的日志文件

日志文件包含事务恢复所需的信息。 可以归档活动日志文件，以便您可以长时间保留日志数据。

什么是日志文件

IBM MQ 记录在 活动日志中发生的所有重要事件。 日志包含恢复所需的信息:

- 持久消息
- IBM MQ 对象，例如队列
- IBM MQ 队列管理器

活动日志包括循环使用的数据集集合 (最多 310 个)。

您可以启用日志归档，以便在活动日志填满时在归档数据集中生成副本。 使用归档允许您在较长时间内保留日志数据。 如果不使用归档，那么将覆盖日志合并和较早的数据。 要恢复页集或恢复 CF 结构中的数据，您需要从生成页集或结构的备份时获取日志数据。 可以在磁盘或磁带上创建归档日志。

归档

由于活动日志具有固定大小，因此 IBM MQ 会定期将每个日志数据集的内容复制到 归档日志，该日志通常是直接访问存储设备 (DASD) 或磁带上的数据集。 如果存在子系统或事务故障，那么 IBM MQ 将使用活动日志并在必要时使用归档日志进行恢复。

归档日志最多可包含 1000 个连续数据集。 您可以使用 z/OS 集成目录工具 (ICF) 对每个数据集进行编目。

归档是 IBM MQ 恢复的基本组件。 如果恢复单元是长时间运行的恢复单元，那么可能会在归档日志中找到该恢复单元中的日志记录。 在这种情况下，恢复需要来自归档日志的数据。 但是，如果禁用归档，那么具有新日志记录的活动日志会合并，覆盖先前的日志记录。 这意味着 IBM MQ 可能无法回退恢复单元，并且消息可能丢失。 然后，队列管理器异常终止。

因此，在生产环境中， **从不关闭归档**。 如果这样做，那么在系统或事务发生故障后，您将面临丢失数据的风险。 只有在测试环境中运行时，才能考虑关闭归档。 如果需要执行此操作，请使用 CSQ6LOGP 宏，如 使用 CSQ6LOGP 中所述。

为了帮助防止意外长时间运行的工作单元出现问题， IBM MQ 发出一条消息 (CSQJ160I 或 CSQJ161I) 如果在活动日志卸载处理期间检测到长时间运行的工作单元。

双重日志记录

在双日志记录中，每个日志记录将写入两个不同的活动日志数据集，以最大程度地降低在重新启动期间发生数据丢失问题的可能性。

您可以将 IBM MQ 配置为与 单日志记录 或 双日志记录一起运行。 通过单一日志记录，日志记录将写入一次活动日志数据集。 每个活动日志数据集都是一个单扩展数据块 VSAM 线性数据集 (LDS)。 通过双重日志记录，每个日志记录将写入两个不同的活动日志数据集。 双日志记录可最大限度降低重新启动期间发生数据丢失问题的可能性。

日志分流

日志分流会导致某些工作单元的日志记录进一步向下写入日志。 这将减少在队列管理器重新启动或回退时，对于长时间运行或长期不确定的工作单元，必须读取的日志数据量。

当工作单元被视为长时，每个日志记录的表示将进一步写到日志下。 此方法称为 分流。 处理整个工作单元后，该工作单元处于 *shunted* 状态。 与已中断的工作单元相关的任何回退或重新启动活动都可以使用已中断的日志记录，而不是使用原始工作单元日志记录。

检测长时间运行的工作单元是检查点进程的功能。 在检查点时间，检查每个活动工作单元以确定是否需要对其进行 shunted。 如果工作单元自创建以来已通过两个先前的检查点，或者自上次被回避以来，那么该工作单元适合被回避。 这意味着单个工作单元可能被多次回避。 这称为 多舍入 工作单元。

每三个检查点都有一个工作单元。 但是，检查点以异步方式执行到日志开关 (或写入导致超过 LOGLOAD 的日志记录)。

一次只发生一个检查点，因此在检查点完成之前可能有多个日志开关。

这意味着，如果没有足够的活动日志，或者这些日志太小，那么在填充所有日志之前，可能无法完成大型工作单元的分流。

如果无法完成分流，那么会产生消息 CSQR027I 。

如果已关闭日志归档，那么当尝试回退其分流失败的工作单元时，将发生 ABEND 5C6 ，原因为 00D1032A 。 要避免此问题，应使用 OFFLOAD=YES。

日志分流始终处于活动状态，无论是否启用日志归档都将运行。

注: 虽然对工作单元的所有日志记录都进行了舍入，但不会对每条记录的整个内容进行舍入，而只会对回退所需的部分进行舍入。 这意味着写入的日志数据量将保持在最小范围内，如果发生页集故障，那么无法使用已中断的记录。 长时间运行的工作单元是已运行超过三个队列管理器检查点的工作单元。

有关日志分流的更多信息，请参阅 管理日志。

日志压缩

您可以配置 IBM MQ for z/OS 以在从日志数据集写入和读取日志记录时对其进行压缩和解压缩。

日志压缩可用于减少针对专用队列上的持久消息写入日志的数据量。 实现的压缩量取决于消息中包含的数据类型。 例如， "运行长度编码" (RLE) 通过压缩字节的重复实例来工作，这可以为结构化数据或面向记录的数据提供有效的结果。

注意: 要放入共享队列的持久消息不受日志压缩限制。

您可以使用系统管理设施 115 (SMF) 记录的 "日志管理器" 部分中的字段来监视实现的数据压缩量。 有关 SMF 的更多信息，请参阅 使用系统管理设施 和 记帐和统计信息消息。

日志压缩会提高系统的处理器利用率。 仅当队列管理器的吞吐量受写入日志数据集的 IO 带宽限制时，或者受保存日志数据集所需的磁盘存储器限制时，才应考虑使用压缩。 如果您正在使用共享队列，那么可以通过向队列共享组添加其他队列管理器并在更多队列管理器之间分配工作负载来缓解 IO 带宽约束。

可以根据需要启用和禁用日志压缩选项，而无需停止和重新启动队列管理器。 无论当前日志压缩设置如何，队列管理器都可以读取任何压缩日志记录。

队列管理器支持日志压缩的 3 设置。

无
不使用日志数据压缩。 这是缺省值。

RLE
使用运行长度编码 (RLE) 执行日志数据压缩。

ANY
启用队列管理器以选择用于提供最大程度的日志记录压缩的压缩算法。 此选项将导致 RLE 压缩。

您可以使用下列其中一项来控制日志记录的压缩:

- MQSC 中的 SET 和 DISPLAY LOG 命令; 请参阅 SET LOG 和 DISPLAY LOG
- PCF 接口中的 "设置日志" 和 "查询日志" 功能; 请参阅 设置日志 和 查询日志
- 系统参数模块中的 CSQ6LOGP 宏; 请参阅 使用 CSQ6LOGP

此外， "日志打印" 实用程序 CSQ1LOGP 支持扩展任何压缩日志记录。

日志数据

日志最多可包含 1800 万 ($1.8 * 10^{19}$) 字节。 每个字节可由其从日志开头的偏移量寻址，该偏移量称为其 相对字节地址 (RBA)。

RBA 由 6 字节或 8 字节字段引用，该字段提供 2^{48} 字节或 2^{64} 字节的总可寻址范围，具体取决于是否正在使用 6 字节或 8 字节的日志 RBA。

但是，当 IBM MQ 检测到所使用的范围超出 F00000000000 (如果正在使用 6 字节的 RBA) 或 FFFF800000000000 (如果正在使用 8 字节的日志 RBA) 时，会发出消息 CSQI045， CSQI046， CSQI047 和 CSQJ032 ，警告您重置日志 RBA。

如果 RBA 值达到 FFF800000000 (如果正在使用 6 字节日志 RBA) 或 FFFFFFC000000000 (如果正在使用 8 字节日志 RBA) ，那么队列管理器将终止，原因码为 00D10257。

发出有关已用日志范围的警告消息后，应规划队列管理器中断，在此期间可以将队列管理器转换为使用 8 字节的日志 RBA ，或者可以重置日志。 重置队列管理器的日志中记录了重置日志的过程。

如果您的队列管理器正在使用 6 字节的日志 RBA ，请考虑将队列管理器转换为使用 8 字节的日志 RBA ， 而不是按照 实现较大的日志相对字节地址中记录的过程来重置队列管理器的日志。

日志由 日志记录组成，每个日志记录都是作为单个单元处理的一组日志数据。 日志记录由其头的第一个字节的 RBA 或其日志记录序号 (LRSN) 标识。 RBA 或 LRSN 唯一地标识从日志中的特定点开始的记录。

是否使用 RBA 或 LRSN 来标识日志点取决于您是否正在使用队列共享组。 在队列共享环境中，不能使用相对字节地址来唯一标识日志点，因为多个队列管理器可以同时更新同一队列，并且每个队列都有自己的日志。 要解决此问题，日志记录序号派生自时间戳记值，并且不一定表示日志记录在日志中的物理位移。

每个日志记录都有一个标头，该标头提供其类型，创建记录的 IBM MQ 子组件以及恢复单元记录的恢复单元标识。

有四种类型的日志记录，在以下标题下描述:

- 恢复单元日志记录
- 检查点记录
- 页集控制记录
- CF 结构备份记录

恢复单元日志记录

大多数日志记录描述了对 IBM MQ 队列的更改。 所有此类更改都在恢复单元内进行。

IBM MQ 使用涉及 撤销/重做 和 补偿日志记录 的特殊日志记录方法来减少重新启动时间并提高系统可用性。

这样做的一个效果是，重启时间是有界的。 如果在重新启动期间发生故障，因此必须再次重新启动队列管理器，那么在第一次重新启动期间完成到故障点的所有恢复活动都不需要在第二次重新启动期间重新应用。 这意味着连续重新启动不会花费越来越长的时间来完成。

检查点记录

为缩短重新启动时间， IBM MQ 会在正常操作期间采用定期检查点。 这些情况如下所示:

- 写入预定义数量的日志记录时。 此数字由系统参数宏 CSQ6SYSP 的称为 LOGLOAD 的检查点频率操作数定义，如 使用 CSQ6SYSP 中所述。
- 在成功重新启动结束时。
- 正常终止。
- 每当 IBM MQ 切换到循环中的下一个活动日志数据集时。

在执行检查点时， IBM MQ 发出 DISPLAY CONN 命令 (如 DISPLAY CONN 中所述) 以便将当前处于不确定状态的连接的列表写入 z/OS 控制台日志。

页集控制记录

这些记录在每个检查点注册 IBM MQ 队列管理器已知的页集和缓冲池，并记录有关在检查点时执行页集的介质恢复所需的日志范围的信息。

对页集和缓冲池的某些动态更改也会作为页集控制记录写入，以便可以恢复这些更改并在下次队列管理器重新启动时自动恢复这些更改。

CF 结构备份记录

这些记录保存从耦合设施列表结构读取的数据，以响应 BACKUP CFSTRUCT 命令。 在不太可能发生耦合设施结构故障的情况下， RECOVER CFSTRUCT 命令使用这些记录以及恢复单元记录来执行耦合设施结构到故障点的介质恢复。

相关任务

实现较大的日志相对字节地址

z/OS 日志的构造方式

使用本主题来了解用于描述日志记录的术语。

每个活动日志数据集都必须是 VSAM 线性数据集 (LDS)。 写入活动日志数据集的物理输出单元是 4 KB 控制区间 (CI)。 每个 CI 都包含一个 VSAM 记录。

物理和逻辑日志记录

一个 VSAM CI 是 物理 记录。 在特定时间记录的信息构成 逻辑 记录，其长度因 CI 中的可用空间而异。 因此，一条物理记录可能包含:

- 多条逻辑记录
- 一个或多个逻辑记录以及另一个逻辑记录的一部分
- 仅一个逻辑记录的一部分

术语 日志记录 指的是 逻辑 记录，而不考虑需要多少 物理 记录来存储该记录。

z/OS 如何写入 IBM MQ for z/OS 日志

使用本主题来了解 IBM MQ 如何处理日志文件记录。

IBM MQ 将每个日志记录写入称为 活动日志的 DASD 数据集。 当活动日志已满时， IBM MQ 会将其内容复制到称为 归档日志的 DASD 或磁带数据集。 此过程称为 卸载。

第 193 页的图 73 说明了日志记录过程。 日志记录通常经历以下循环:

1. IBM MQ 注意到对恢复日志记录中的数据和重要事件的更改。
2. 如果需要， IBM MQ 将处理恢复日志记录并将其分为多个段。
3. 日志记录按顺序放置在 输出日志缓冲区中，这些缓冲区格式化为 VSAM 控制区间 (CI)。 每个日志记录都由零到 $2^{64}-1$ 范围内的相对字节地址标识。
4. CI 将写入一组预定义的 DASD 活动日志数据集，这些数据集按顺序使用并循环使用。
5. 如果归档处于活动状态，那么随着每个活动日志数据集变满，其内容将自动卸载到新的归档日志数据集。

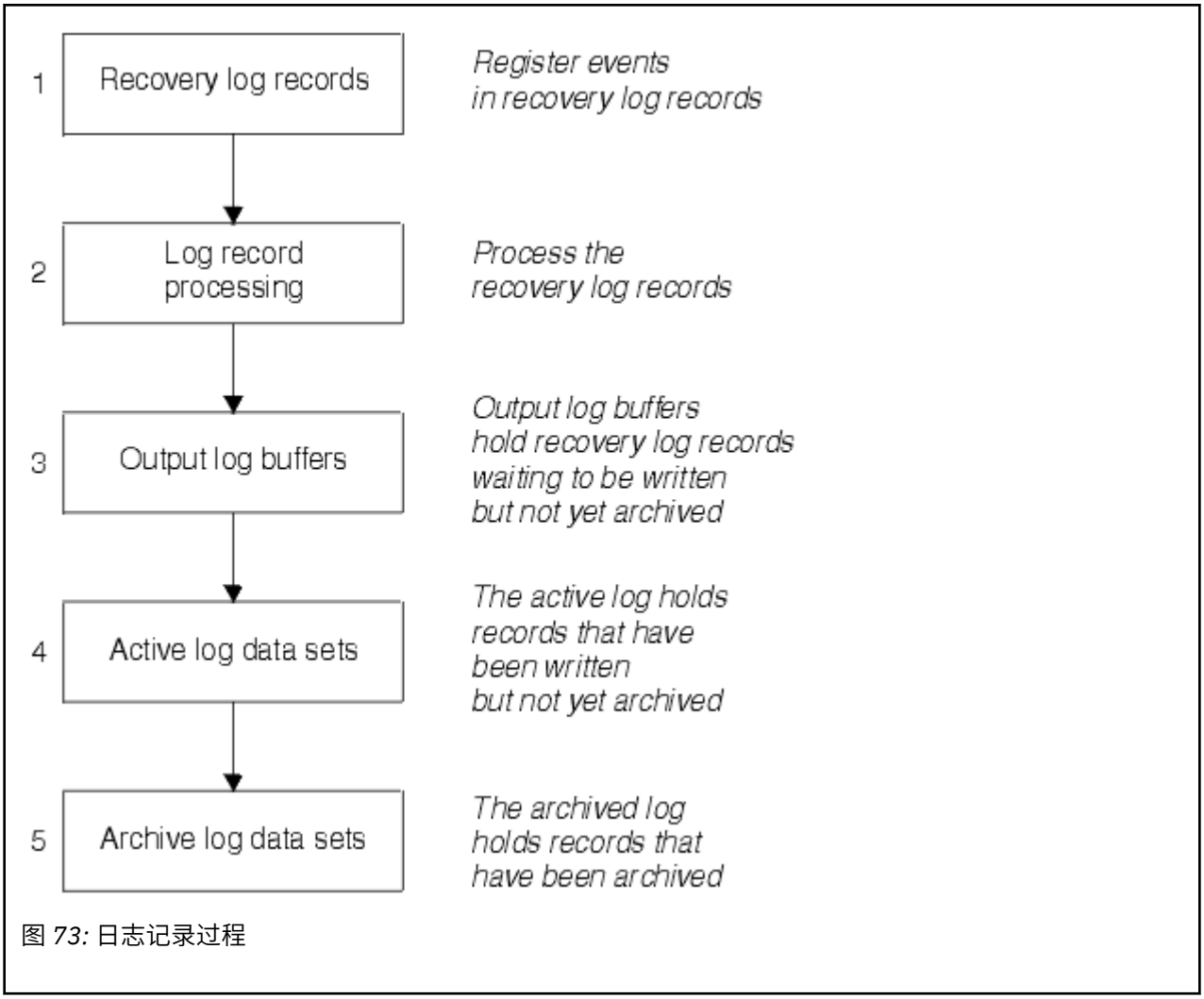


图 *73:* 日志记录过程

写入活动日志时

每当发生以下任何情况时，都会将存储中日志缓冲区写入活动日志数据集:

- 日志缓冲区变满。
- 达到写入阈值 (如 CSQ6LOGP 宏中所指定)。
- 发生某些重要事件，例如落实点，或者发出 IBM MQ BACKUP CFSTRUCT 命令时。

初始化队列管理器时，将动态分配 BSDS 中指定的活动日志数据集，以供队列管理器独占使用，并保持专门分配给 IBM MQ ，直到队列管理器终止为止。

动态添加日志数据集

可以在队列管理器运行时动态定义新的活动日志数据集。 此功能可减轻由于瞬态问题而导致归档无法卸载活动日志时队列管理器挂起的问题。 请参阅 DEFINE LOG 命令以获取更多信息。

注: 要重新定义或除去活动日志，必须终止并重新启动队列管理器。

IBM MQ 和存储管理子系统

IBM MQ 参数使您能够在动态分配 IBM MQ 归档日志数据集时指定存储管理子系统 (MVS/DFP SMS) 存储类。 IBM MQ 启动日志数据集的归档，但您可以使用 SMS 来执行归档数据集的分配。

相关参考

第 194 页的『写入 IBM MQ for z/OS 归档日志时』
使用本主题来了解将活动日志复制到归档日志的过程以及该过程发生的时间。

z/OS 写入 *IBM MQ for z/OS* 归档日志时

使用本主题来了解将活动日志复制到归档日志的过程以及该过程发生的时间。

将活动日志复制到归档日志的过程称为 卸载。 在 第 194 页的图 74 中以示意图方式显示了卸载到其他日志记录事件的关系。

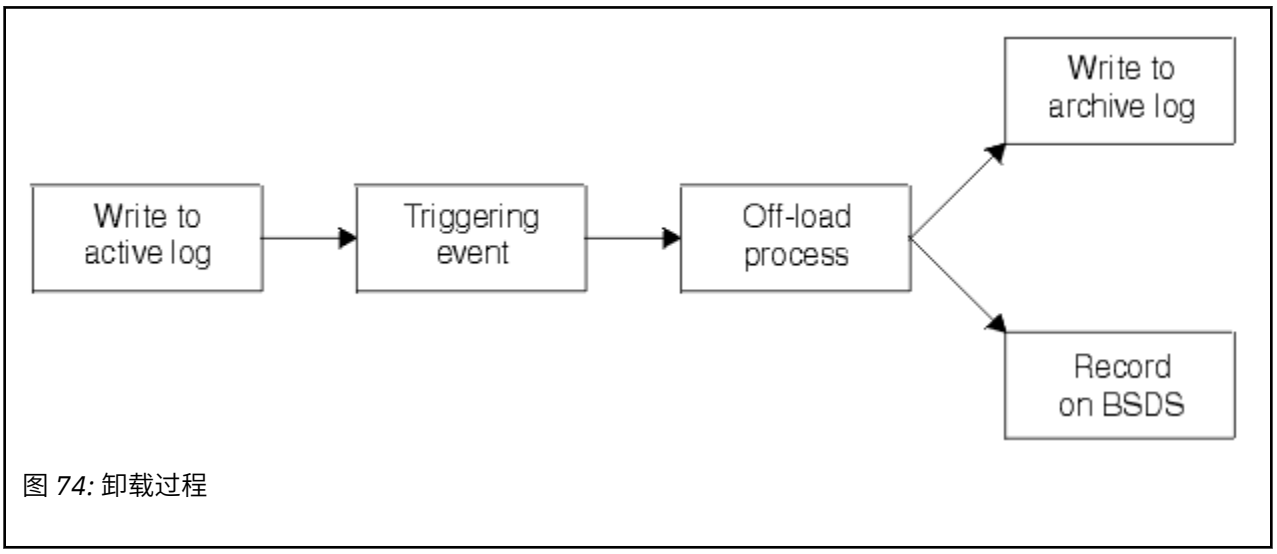


图 *74:* 卸载过程

触发卸载过程

活动日志到归档日志的卸载过程可由多个事件触发。 例如:

- 填充活动日志数据集。
- 使用 MQSC ARCHIVE LOG 命令。
- 写入活动日志数据集时发生错误。

数据集在故障点之前被截断，未写入的记录将成为新数据集的第一条记录。 对于截断的数据集，将触发卸载，如同对于正常的完整日志数据集一样。 如果存在双活动日志，那么将截断这两个副本，以使这两个副本保持同步。

消息 CSQJ110E 在最后一个可用活动日志为 5% 已满时发出，此后将以 5% 为增量发出，并说明正在使用的日志容量的百分比。 如果所有活动日志已满，那么 IBM MQ 将停止处理，直到发生卸载，并发出以下消息:

```
CSQJ111A +CSQ1 OUT OF SPACE IN ACTIVE LOG DATA SETS
```

卸载过程

当所有活动日志已满时， IBM MQ 将运行卸载过程并停止处理，直到卸载过程完成为止。 如果在活动日志已满时卸载处理失败，那么 IBM MQ 将异常终止。

当活动日志准备好卸载时，将向 z/OS 控制台操作员发送请求以安装磁带或准备 DASD 单元。 ARCWTOR 日志记录选项的值 (有关更多信息，请参阅 使用 CSQ6ARVP) 确定是否接收到请求。 如果使用磁带进行卸载，请指定 ARCWTOR=YES。 如果值为 YES ，那么请求前面有 WTOR (消息号 CSQJ008E) 指示操作员准备要分配的归档日志数据集。

操作员无需立即响应此消息。 但是，延迟响应会延迟卸载过程。 它不会影响 IBM MQ 性能，除非操作员将响应延迟很长时间以使 IBM MQ 耗尽活动日志。

操作员可以通过取消卸载过程进行响应。 在这种情况下，如果分配用于双归档数据集的第一个副本，那么卸载过程仅会延迟，直到下一个活动日志数据集变满为止。 如果分配给第二个副本，那么归档进程将切换到单副本方式，但仅针对此数据集。

卸载时的中断和错误

在卸载处理完成之前，停止队列管理器的请求不会生效。 如果 IBM MQ 在卸载过程中失败，那么卸载将在重新启动队列管理器时再次开始。

卸载处理期间的消息

卸载消息由 IBM MQ 和卸载过程发送到 z/OS 控制台。 您可以使用这些消息在各种日志数据集中查找 RBA 范围。

z/OS 较大的日志相对字节地址

此功能通过增加必须重置日志之前的时间段来提高队列管理器的可用性。

恢复数据将写入日志，以便在重新启动队列管理器时持久消息可用。 术语 "日志相对字节地址" (log RBA) 用于引用数据位置作为日志开头的偏移量。

在 IBM MQ 8.0 之前， 6 字节日志 RBA 最多可以处理 256 太字节的数据。 在写入此数量的日志记录之前，必须遵循 重置队列管理器的日志中记录的过程来重置队列管理器的日志。

重置队列管理器的日志不是一个快速过程，可能需要长时间中断，因为需要在此过程中重置页集。 对于高使用率队列管理器，此操作通常可能每年执行一次。

从 IBM MQ 8.0 开始，日志 RBA 可以是 8 字节长的，现在队列管理器可以在需要重置日志 RBA 之前处理 64,000 倍以上的数据 (16 EB)。 使用更大的日志 RBA 的影响是写入的日志数据大小增加了几个字节。

何时启用此功能?

V 9.3.0 在 IBM MQ 9.3.0 或更高版本上创建的队列管理器已启用此功能。

如果当前日志 RBA 接近日志 RBA 范围的末尾，请考虑将队列管理器转换为使用 8 字节日志 RBA ，而不是重置队列管理器的日志。 转换队列管理器以使用 8 字节日志 RBA 需要比重置日志更短的中断时间，并显着增加必须重置日志之前的时间段。

在队列管理器初始化期间发出的消息 CSQJ034I 指示队列管理器的日志 RBA 范围结束 (已配置) ，并且可用于确定是否正在使用 6 字节或 8 字节日志 RBA。

如何启用此功能?

通过启动版本为 2 格式的 BSDS 的队列管理器来启用 8 字节日志 RBA。 总之，实现这一目标的途径是:

1. 确保队列共享组中的所有队列管理器都满足启用 8 字节日志 RBA 的要求
2. 完全关闭队列管理器
3. 运行 BSDS 转换实用程序 以创建版本为 2 的 BSDS 副本。
4. 使用转换后的 BSDS 重新启动队列管理器。

一旦将队列管理器转换为使用 8 字节日志 RBA ，它就无法返回到使用 6 字节日志 RBA。

有关如何启用 8 字节日志 RBA 的详细过程，请参阅 实现较大的日志相对字节地址 。

相关任务

计划增大最大可寻址日志范围

相关参考

BSDS 转换实用程序 (CSQJUCNV)

z/OS 引导程序数据集

IBM MQ 需要引导数据集作为引用日志数据集和日志记录的机制。 在正常处理和重新启动恢复期间需要此信息。

引导程序数据集用于的内容

引导数据集 (BSDS) 是 VSAM 键序列数据集 (KSDS)，用于保存 IBM MQ 所需的信息。 它包含以下内容:

- IBM MQ 已知的所有活动日志数据集和归档日志数据集的清单。 IBM MQ 使用此库存来执行以下操作:
  - 跟踪活动日志数据集和归档日志数据集
  - 找到日志记录，以便它可以满足正常处理期间的日志读取请求
  - 找到日志记录，以便它可以处理重新启动处理

  每次定义归档日志数据集或复用活动日志数据集时， IBM MQ 都会将信息存储在库存中。 对于活动日志，清单会显示哪些已满，哪些可供复用。 库存保存该数据集中保存的日志的每个部分的相对字节地址 (RBA)。
- 所有最近 IBM MQ 活动的 回绕 清单。 如果必须重新启动队列管理器，那么需要执行此操作。

如果队列管理器有错误并且您必须将其重新启动，那么 BSDS 是必需的。 IBM MQ **必须** 具有 BSDS。 为了最大限度降低重新启动期间发生问题的可能性，您可以使用双 BSD 配置 IBM MQ，每个 BSD 都记录相同的信息。 使用双 BSD 称为以 双方式运行。 如果可能，请将副本放在不同的卷上。 这将降低在卷损坏或损坏时两者都丢失的风险。 使用双 BSD，而不是对 DASD 进行双写。

BSDS 是在定制 IBM MQ 时设置的，您可以使用更改日志库存实用程序 (CSQJU003) 来管理库存。 有关此实用程序的更多信息，请参阅管理 IBM MQ for z/OS。 它由队列管理器启动过程中的 DD 语句引用。

通常， IBM MQ 会保留 BSDS 的重复副本。 如果发生 I/O 错误，那么它将取消分配失败的副本并继续使用单个 BSDS。 您可以复原双模操作，如 管理 IBM MQ for z/OS 中所述。

安装 IBM MQ 时，将首先在 BSDS 中注册活动日志。 如果不终止并重新启动队列管理器，那么无法替换活动日志。

将动态分配归档日志数据集。 分配一个数据集时，将在 BSDS 中注册数据集名称。 当添加归档时，归档日志数据集的列表将展开，并在达到用户确定的条目数时回绕。 对于单个归档日志记录，最大条目数为 1000，对于双日志记录，最大条目数为 2000。

您可以使用磁带管理系统来删除归档日志数据集 (IBM MQ 没有自动化方法)。 因此，在系统管理员删除归档日志数据集之后，有关归档日志数据集的信息可以在 BSDS 中。

相反，可能已超过最大归档日志数据集数，并且 BSDS 中的数据在数据集到达到期日期之前很久就已删除。

您可以使用以下 MQSC 命令来确定日志的范围，以及保存各种类型的介质或队列管理器恢复所需的最早日志 RBA 的活动或归档日志数据集的名称:

```
DISPLAY USAGE TYPE(DATASET)
```

如果系统参数模块指定在分配时对归档日志数据集进行编目，那么 BSDS 将指向集成目录设施 (ICF) 目录以获取后续分配所需的信息。 否则，每个卷的 BSDS 条目将注册以后分配所需的卷序列号和单元信息。

BSDS 版本

BSDS 的格式因其版本而异。 增加 BSDS 的版本允许使用新功能。 IBM MQ 支持以下 BSDS 版本:

V1
受 IBM MQ 的所有发行版支持。 版本 1 BSDS 支持 6 字节日志 RBA 值。

V2
受 IBM MQ 8.0 和更高版本支持。 版本 2 BSDS 支持 8 字节的日志 RBA 值，并且每个活动日志副本中最多有 310 个数据集。

V 9.3.0 缺省情况下，针对在 IBM MQ 9.3.0 和更高版本上创建的队列管理器启用。

V3
受 IBM MQ 8.0 和更高版本支持。 当向任一活动日志副本添加了超过 31 个数据集时， BSDS 将自动从 V 2 转换为 V 3。

您可以通过运行打印日志映射实用程序 (CSQJU004)来确定 BSDS 的版本。 要将 BSDS 从 V 1 转换为 V 2 ，请运行 BSDS 转换实用程序 (CSQJUCNV)。

请参阅 第 195 页的『较大的日志相对字节地址』 ，以获取有关 6 字节和 8 字节日志 RBA 的更多信息。

归档日志数据集和 BSDS 副本

每次创建新的归档日志数据集时，还会创建 BSDS 的副本。 如果归档日志在磁带上，那么 BSDS 是第一个输出卷上的第一个数据集。 如果归档日志位于 DASD 上，那么 BSDS 是单独的数据集。

归档日志和 BSDS 副本的数据集名称相同，但归档日志名称的最低级别限定符以 A 开头， BSDS 副本以 B 开头，例如:

归档日志名称
CSQ.ARCHLOG1.E00186.T2336229. *A* 0000001

BSDS 副本名称
CSQ.ARCHLOG1.E00186.T2336229. *B* 0000001

如果在复制 BSDS 时发生读错误，那么不会创建副本，将发出消息 CSQJ125E ，并且将在没有 BSDS 副本的情况下继续卸载到新的归档日志数据集。

z/OS z/OS 上的系统定义

IBM MQ for z/OS 使用许多缺省对象定义，并提供样本 JCL 来创建这些缺省对象。 使用本主题来了解这些缺省对象以及样本 JCL。

设置系统参数

在 IBM MQ for z/OS 中，系统参数模块控制 IBM MQ 在其操作中使用的日志记录，归档，跟踪和连接环境。 系统参数由三个汇编程序宏指定，如下所示:

CSQ6SYSP
系统参数，包括设置连接和跟踪环境。

CSQ6LOGP
日志记录参数。

CSQ6ARVP
日志归档参数。

IBM MQ for z/OS 随附了缺省参数模块。 如果这些值不包含要使用的值，那么您可以使用随 IBM MQ 提供的样本来创建自己的参数模块。 样本为 `thlqual.SCSQPROC(CSQ4ZPRM)`。

您可以在队列管理器运行时更改某些系统参数。 请参阅 MQSC 命令中的 SET SYSTEM ， SET LOG 和 SET ARCHIVE 命令。

有关定义的更多信息，请参阅以下主题:

- 第 198 页的『定义 IBM MQ for z/OS 的系统对象』
- 第 201 页的『在 IBM MQ for z/OS 上调整队列管理器』
- 第 202 页的『随 IBM MQ for z/OS 提供的样本定义』

相关概念
定制样本初始化输入数据集
可在 IBM MQ for z/OS 上发出 MQSC 和 PCF 命令的源
相关任务
管理 z/OS
配置集群
监视 IBM MQ

z/OS 定义 IBM MQ for z/OS 的系统对象

IBM MQ for z/OS 需要其他预定义对象以用于发布/预订应用程序，集群和通道控制以及其他系统管理功能。

IBM MQ for z/OS 所需的系统对象可分为以下类别:

- 发布/预订对象
- 系统缺省对象
- 系统命令对象
- 系统管理对象
- 通道队列
- 集群队列
- 队列共享组队列
- 存储类
- 定义系统对象死信队列
- 缺省传输队列
- 内部队列
- 第 201 页的『通道认证队列』

发布/预订对象

在将发布/预订应用程序与 IBM MQ for z/OS 配合使用之前，需要定义多个系统对象。 样本定义随 IBM MQ 一起提供，以帮助您定义这些对象。 CSQ4INSG 中描述了这些样本。

要使用发布/预订，您需要定义以下对象:

- 名为 SYSTEM.RETAINED.PUB.QUEUE，用于保存队列管理器中每个保留发布的副本。 每个完整主题名称最多可以有一个保留发布存储在此队列中。 如果应用程序将在许多不同的主题上使用保留发布，或者如果保留发布消息是大型消息，那么应仔细规划此队列的存储需求，包括将其分配给其自己的页集 (如果其存储需求较大)。 为了提高性能，您应该使用索引类型 MSGID 来定义此队列 (如提供的样本队列定义中所示)。
- 名为 SYSTEM.DURABLE.SUBSCRIBER.QUEUE，用于保存队列管理器中持久预订的持久副本。 为了提高性能，您应该使用索引类型 CORRELID 来定义此队列 (如提供的样本队列定义中所示)。
- 名为 SYSTEM.DURABLE.MODEL.QUEUE，用作受管持久预订的模型。
- 名为 SYSTEM.NDURABLE.MODEL.QUEUE，用作受管非持久预订的模型。
- 名为 SYSTEM.QPUBSUB.QUEUE.NAMELIST，其中包含由排队的发布/预订接口监视的队列名称的列表。
- 名为 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST，其中包含由排队的发布/预订接口用于将主题对象与预订点匹配的主题对象的列表。
- 名为 SYSTEM.BASE.TOPIC，用作解析属性的基本主题。
- 名为 SYSTEM.BROKER.DEFAULT.STREAM，这是已排队的发布/预订接口所使用的缺省流。
- 名为 SYSTEM.BROKER.DEFAULT.SUBPOINT，这是已排队的发布/预订接口使用的缺省 RFH2 预订点。
- 名为 SYSTEM.BROKER.ADMIN.STREAM，这是已排队的发布/预订接口所使用的管理流。
- 名为 SYSTEM.DEFAULT.SUB，这是用于在 DEFINE SUB 命令上提供缺省值的缺省预订对象。

系统缺省对象

系统缺省对象用于在定义对象时提供缺省属性，并且不指定定义所基于的其他对象的名称。

缺省系统对象定义的名称以字符 "SYSTEM.DEFAULT"或"SYSTEM.DEF。" 例如，系统缺省本地队列名为 SYSTEM.DEFAULT.LOCAL.QUEUE。

这些对象定义这些 IBM MQ 对象的属性的系统缺省值:

- 本地队列
- 模型队列
- 别名队列
- 远程队列
- 进程
- 名称列表
- 通道
- 存储类
- 认证信息

共享队列是一种特殊类型的本地队列，因此当您定义共享队列时，定义基于 SYSTEM.DEFAULT.LOCAL.QUEUE。 您需要记住为耦合设施结构名称提供一个值，因为缺省定义中未指定一个值。 或者，您可以定义自己的缺省共享队列定义，以用作共享队列的基础，以便它们都继承必需的属性。 请记住，您只需要在队列共享组中的一个队列管理器上定义共享队列。

系统命令对象

系统命令对象的名称以字符 SYSTEM.COMMAND。 必须先定义这些对象，然后才能使用 IBM MQ 操作和控制面板向 IBM MQ 子系统发出命令。

有两个系统命令对象:

1. 系统命令输入队列是一个本地队列，在 IBM MQ 命令处理器处理命令之前，会将这些命令放在该队列上。 它必须称为 SYSTEM.COMMAND.INPUT。 名为 SYSTEM.ADMIN.COMMAND.QUEUE 以与 IBM MQ for Multiplatforms 兼容，并供 IBM MQ Console 和 administrative REST API 使用。
2. SYSTEM.COMMAND.REPLY.MODEL 是用于定义系统命令应答队列的模型队列。

有两个额外的对象可供 IBM MQ Explorer 使用:

- SYSTEM.MQEXPLORER.REPLY.MODEL 队列
- SYSTEM.ADMIN.SVRCONN 通道

SYSTEM.REST.REPLY.QUEUE 是 IBM MQ administrative REST API 使用的应答队列。

通常使用非持久消息发送命令，因此这两个系统命令对象都应该具有 DEFPSIST (NO) 属性，以便使用它们的应用程序 (包括提供的应用程序，例如实用程序以及操作和控制面板) 在缺省情况下获取非持久消息。 如果应用程序将持久消息用于命令，请为应答队列设置 DEFTYPE (PERMDYN) 属性，因为此类命令的应答消息是持久消息。

系统管理对象

系统管理对象的名称以字符 SYSTEM.ADMIN。

有七个系统管理对象:

- SYSTEM.ADMIN.CHANNEL.EVENT 队列
- SYSTEM.ADMIN.COMMAND.EVENT 队列
- SYSTEM.ADMIN.CONFIG.EVENT 队列
- SYSTEM.ADMIN.PERFM.EVENT 队列
- SYSTEM.ADMIN.QMGR.EVENT 队列
- SYSTEM.ADMIN.TRACE.ROUTE.QUEUE 队列
- SYSTEM.ADMIN.ACTIVITY.QUEUE 队列

通道队列

要使用分布式排队，您需要定义以下对象:

- 名为 SYSTEM.CHANNEL.SYNCQ，用于维护通道的序号和逻辑工作单元标识 (LUWID)。 要提高通道性能，您应该使用索引类型 MSGID 来定义此队列 (如提供的样本队列定义中所示)。
- 名为 SYSTEM.CHANNEL.INITQ，用于通道命令。

不能将这些队列定义为共享队列。

集群队列

要使用 IBM MQ 集群，您需要定义以下对象:

- 称为 SYSTEM.CLUSTER.COMMAND.QUEUE，用于在队列管理器之间通信存储库更改。 写入此队列的消息包含对要应用于存储库本地副本的存储库数据的更新或对存储库数据的请求。
- 名为 SYSTEM.CLUSTER.REPOSITORY.QUEUE，用于保存存储库的持久副本。
- 名为 SYSTEM.CLUSTER.TRANSMIT.QUEUE，这是集群中所有目标的传输队列。 出于性能原因，您应该使用索引类型 CORRELID 来定义此队列 (如样本队列定义中所示)。

这些队列通常包含大量消息。

不能将这些队列定义为共享队列。

队列共享组队列

要使用共享通道和组内排队，需要定义以下对象:

- 名为 SYSTEM.QSG.CHANNEL.SYNCQ，用于保存共享通道的同步信息。
- 名为 SYSTEM.QSG.TRANSMIT.QUEUE，用作组内排队的传输队列。 如果您正在队列共享组中运行，那么必须定义此队列，即使您未使用组内排队也是如此。

存储类

建议您定义以下六个存储类。 您必须定义其中四个值，因为 IBM MQ 需要这些值。 建议使用其他存储类定义，因为它们在样本队列定义中使用。

DEFAULT (必需)
此存储类用于所有性能不重要且不适合任何其他存储类的消息队列。 如果在定义队列时未指定缺省存储类，那么它也是提供的缺省存储类。

NODEFINE (必需)
如果未定义定义队列时指定的存储类，那么将使用此存储类。

REMOTE (必需)
此存储类主要用于传输队列，即具有短期性能关键消息的系统相关队列。

SYSLNGLV
此存储类用于长时间运行的性能关键型消息。

SYSTEM (必需)
此存储类用于性能关键型系统相关消息队列，例如 SYSTEM.CHANNEL.SYNQ 和 SYSTEM.CLUSTER.* 队列。

SYSVOLAT
此存储类用于短期的性能关键型消息。

您可以根据需要修改其属性并添加其他存储类定义。

定义系统对象死信队列

如果消息目标无效，那么将使用死信队列。 IBM MQ 将此类消息放在称为死信队列的本地队列上。 虽然具有死信队列不是必需的，但您应该将其视为必需的，尤其是在使用分布式排队或其中一个 IBM MQ 网桥时。

请 **不要** 将死信队列定义为共享队列。 放入一个队列管理器上的本地队列可能会放入死信队列。 如果死信队列是共享队列，那么另一个系统上的死信队列处理程序可以处理消息并将其放在同名的队列上，但因为这是在另一个队列管理器上，所以它将是错误的队列，或者具有不同的安全概要文件。 如果该队列不存在，那么将无法对其进行重新处理。

如果您决定定义死信队列，那么还必须将其名称告知队列管理器。 要执行此操作，请使用 ALTER QMGR DEADQ (*queue-name*) 命令。 有关更多信息，请参阅 显示和改变队列管理器属性。

缺省传输队列

当没有其他合适的传输队列可用于将消息发送到另一个队列管理器时，将使用缺省传输队列。 如果您定义了缺省传输队列，那么还必须定义通道，以便为该队列提供服务。 如果不执行此操作，那么放入缺省传输队列的消息不会传输至远程队列管理器，它们将保留在该队列上。

如果您决定定义缺省传输队列，那么还必须将其名称告知队列管理器。 要执行此操作，请使用 ALTER QMGR 命令。

内部队列

- **暂挂数据队列**
  - 为内部使用而定义的队列 SYSTEM.PENDING.DATA.QUEUE，支持在 JMS 发布/预订环境中使用持久预订。
- **JMS 2.0 交付延迟登台队列**
  - 如果使用 JMS 2.0 提供的交付延迟功能，那么将使用内部登台队列 SYSTEM.DDELAY.LOCAL.QUEUE。 队列管理器使用此队列来临时存储以非零交付延迟发送的消息，直到交付延迟完成，并将消息放入其目标目的地。 在 CSQ4INSG 中提供并注释掉了此队列的样本定义。
  - 定义 SYSTEM.DDELAY.LOCAL.QUEUE 队列，您必须设置 STGCLASS， MAXMSGL 和 MAXDEPTH 属性，以获取将以传递延迟方式发送的预期消息数。 此外，在定义 SYSTEM.DDELAY.LOCAL.QUEUE 队列确保只有队列管理器可以将消息放入此队列。 应注意确保没有用户标识具有将消息放入此队列的权限。

通道认证队列

要在内部使用通道认证，请使用 SYSTEM.CHLAUTH.DATA.QUEUE 队列是必需的。 样本定义随 IBM MQ 一起提供，以帮助您定义这些对象。 此样本在 CSQ4INSA 中进行了描述，其中还定义了一些缺省规则。

z/OS 在 IBM MQ for z/OS 上调整队列管理器

您可以执行几个简单步骤来确保调整队列管理器以避免基本性能问题。

您可以通过多种方法来提高队列管理器的性能，这些性能由 ALTER QMGR 命令设置的队列管理器属性控制。 此部分包含有关如何通过设置队列管理器上允许的最大消息数或通过对队列管理器执行 "清理" 来执行此操作的信息。 IBM MQ SupportPac MP16 -IBMMQ 为了 z/OS 容量规划与调整 提供了有关性能和调整的更多信息。

同步点

队列管理器的其中一个角色是应用程序中的同步点控制。 应用程序会构造一个工作单元，其中包含任意数量的 MQPUT 或 MQGET 调用，这些调用通过 MQCMIT 调用终止。

随着一个 MQCMIT 作用域内 MQPUT 或 MQGET 调用的数量增加，落实的性能成本显着增加。 通常，应用程序应设计为在单个同步点中不使用 MQPUT/MQGET 大量消息。

您可以使用 MAXUMSGS 队列管理器属性以管理方式限制任何单个同步点中的消息数。 如果应用程序超过此限制，那么它会在 MQPUT,MQPUT1 或 MQGET 调用上接收到 MQRC_SYNCPOINT_LIMIT_达到的 MQRC_SYNCPOINT_LIMIT_超出此限制。 然后，应用程序应根据需要发出 MQCMIT 或 MQBACK。

MAXUMSGS 的缺省值为 10000。 如果要实施下限，那么可以降低此值，这也可以帮助防止循环应用程序。 在减少 MAXUMSGS 之前，请确保您了解现有应用程序以确保它们不会超过限制，或者可以容许 MQRC_SYNCPOINT_LIMIT_已达到返回码

到期消息数

已到期的消息将由下一个相应的 MQGET 调用废弃。 但是，如果未发生此类调用，那么不会废弃已到期的消息，对于某些队列，尤其是那些通过 MessageId， CorrelId 或 GroupId 进行消息检索并且为队列建立索引以实现性能的队列，可能会累积许多已到期的消息。 队列管理器可以定期扫描任何队列以查找到期的消息，然后删除这些消息。 您可以选择执行此扫描的频率 (如果有的话)。 有两种方法可以执行此操作:

显式请求
您可以控制扫描哪些队列以及时间。 发出 REFRESH QMGR TYPE (到期) 命令，指定要扫描的一个或多个队列。

定期扫描
您可以使用 EXPRYINT 属性在队列管理器对象中指定到期时间间隔。 队列管理器在每个队列上维护有关到期消息的信息，并知道何时值得扫描到期消息。 每次达到 EXPRYINT 时间间隔时，队列管理器都会查找值得扫描到期消息的候选队列，并仅扫描它认为值得扫描的那些队列。 它不会扫描所有队列。 这可避免在不必要的扫描上浪费任何处理器时间。

共享队列仅由队列共享组中的一个队列管理器扫描。 通常，要重新启动的第一个队列管理器或要设置 EXPRYINT 的第一个队列管理器将执行扫描。

注: 必须为队列共享组中的所有队列管理器设置相同的 EXPRYINT 值。

z/OS 随 IBM MQ for z/OS 提供的样本定义

使用本主题作为样本 JCL 的参考，以及 IBM MQ for z/OS 随附的代码。

IBM MQ 在 thlqual.SCSQPROC 库中提供了以下样本定义。 您可以使用它们来定义系统对象以及定制您自己的对象。 您可以将其中一些数据包含在初始化输入数据集中（在初始化命令中进行了说明）。

表 *22:* 系统对象的 *IBM MQ* 样本定义

初始化输入数据集 (initialization input data set)	**样本名称**
CSQINP1	CSQ4INP1 CSQ4INPR
CSQINP2	CSQ4INSA CSQ4INYS [1] CSQ4INSX CSQ4INSG CSQ4INSR CSQ4INSS CSQ4INSJ CSQ4INSM CSQ4INYG CSQ4INYR CSQ4INYC CSQ4INYD CSQ4INSC

表 *22:* 系统对象的 *IBM MQ* 样本定义 *(*继续*)*	
初始化输入数据集 (initialization input data set)	**样本名称**
CSQINPT	CSQ4INST CSQ4INYT
其他	CSQ4DISP CSQ4INPX CSQ4IVPQ CSQ4IVPG CSQ4MSTR CSQ4MSRR CSQ4QMIN

注:

1. 这些样本定义的顺序很重要: 如果 INYS， INSX 和 INSG 顺序不正确，那么会发生错误。

CSQINP1 样本

当您对每个消息类使用一个页集时，请使用样本 CSQINP1 数据集 thlqual.SCSQPROC(CSQ4INP1)，或者在对主要消息类使用多个页集时使用 thlqual.SCSQPROC(CSQ4INPR)。 它包含缓冲池的定义，与缓冲池关联的页集以及 ALTER SECURITY 命令。 将样本包括在队列管理器启动式任务过程的 CSQINP1 并置中。

CSQINP2 样本

CSQ4INSG 系统对象样本

样本 CSQINP2 数据集 thlqual.SCSQPROC(CSQ4INSG) 包含以下系统对象的定义以供一般使用:

- 系统缺省对象
- 系统命令对象
- 系统管理对象
- 供系统使用的其他对象

必须定义此样本中的对象，但在子系统首次启动时只需执行一次。 在 CSQINP2 数据集中包含定义是执行此操作的最佳方法。 它们在队列管理器关闭和重新启动之间进行维护。 您不得更改对象名，但可以根据需要更改其属性。

满足以下条件时，将向 SYSTEM.DURABLE.SUBSCRIBER.QUEUE 队列 (即使发布预订未处于活动状态):

- QMGR 属性 PSMODE 设置为 DISABLED
- 存在样本对象 CSQ4INST 语句 `DEFINE SUB('SYSTEM.DEFAULT.SUB')`。

要避免此情况，请删除或注释掉 `DEFINE SUB('SYSTEM.DEFAULT.SUB')` 语句。

仅当使用 JMS 2.0 交付延迟时，才需要定义 JMS 2.0 交付延迟登台队列 SYSTEM.DDELAY.LOCAL.QUEUE。 缺省情况下，会将队列定义注释掉，如果需要，可以将其取消注释。

CSQ4INSA 系统对象和认证样本

样本 CSQINP2 数据集 thlqual.SCSQPROC(CSQ4INSA) 包含通道认证系统队列定义。 此队列保存通道认证记录。 它还包含缺省通道认证规则。

如果 CHLAUTH 在队列管理器上已启用并且要运行通道，或者要 SET 或 DISPLAY CHLAUTH 记录，那么必须定义此样本中的对象。 您只需在子系统首次启动时定义一次它们。 在 CSQINP2 数据集中包含定义是执行此操作的最佳方法。 它们在队列管理器关闭和重启时保持运行。 您不能更改队列名称。

CSQ4INSS 系统对象样本

如果您正在使用队列共享组，那么可以定义其他系统对象。

样本数据集 thlqual.SCSQPROC(CSQ4INSS) 包含用于 CF 结构的样本命令以及共享通道和组内排队所需的系统对象的一组定义。

不能按现在的方式使用此样本; 必须先对其进行定制，然后才能使用此样本。 然后，可以将此成员包括在队列管理器启动过程的 CSQINP2 DD 并置中，也可以将其用作 CSQUTIL 实用程序的 COMMAND 函数的输入以发出所需的命令。

在定义组或共享对象时，仅需要将它们包括在队列共享组中的一个队列管理器的 CSQINP2 DD 并置中。

CSQ4INSX 系统对象样本

如果您正在使用分布式排队和集群，那么必须定义其他系统对象。

样本数据集 thlqual.SCSQPROC(CSQ4INSX) 包含所需的队列定义。 可以将此成员包括在队列管理器启动过程的 CSQINP2 DD 并置中，也可以将其用作 CSQUTIL 实用程序中的 COMMAND 函数的输入，以发出必需的 DEFINE 命令。

有两种类型的对象定义:

- SYSTEM.CHANNEL.xx，任何分布式排队都需要
- SYSTEM.CLUSTER.xx，集群所需的

CSQ4INSJ 系统 JMS 对象样本

定义在 JMS 发布/预订域中使用的队列。

CSQ4INSM 系统对象样本

如果您正在使用高级消息安全性，那么必须定义其他系统对象。 样本数据集 thlqual.SCSQPROC(CSQ4INSM) 包含所需的队列定义。

CSQ4INSR 对象样本

定义 WebSphere Application Server 和代理使用的队列。

CSQ4INYD 对象样本

如果您正在使用分布式排队，并且需要设置自己的队列，进程和通道。

样本数据集 thlqual.SCSQPROC(CSQ4INYD) 包含可用于定制分布式排队对象的样本定义。 它包括:

- 发送端的一组定义
- 接收端的一组定义
- 一组用于使用客户机的定义

不能按样使用此样本-在使用前必须对其进行定制。 然后，可以将此成员包含在队列管理器启动过程的 CSQINP2 DD 并置中，也可以将其用作 CSQUTIL 实用程序的 COMMAND 函数的输入以发出必需的 DEFINE 命令。 (这更可取，因为这意味着您不必在每次重新启动队列管理器时重新定义这些对象)。

CSQ4INYC 对象样本

如果您正在使用集群，那么将在需要时自动创建等同于分布式排队的通道定义和远程队列定义的定义。 但是，需要一些手动通道定义-集群的集群接收方通道以及至少一个集群存储库队列管理器的集群发送方定义。

样本数据集: thlqual.SCSQPROC(CSQ4INYC) 包含可用于定制集群对象的以下样本定义:

- 队列管理器的定义

- 接收通道的定义
- 发送通道的定义
- 集群队列的定义
- 集群列表的定义

不能按样使用此样本-在使用前必须对其进行定制。 然后，可以将此成员包含在队列管理器启动过程的 CSQINP2 DD 并置中，也可以将其用作 CSQUTIL 实用程序的 COMMAND 函数的输入以发出必需的 DEFINE 命令。 这是首选项，因为这意味着您不必在每次重新启动 IBM MQ 时重新定义这些对象。

CSQ4INYG 对象样本

样本数据集: thlqual.SCSQPROC(CSQ4INYG) 包含可用于定制您自己的对象以供常规使用的以下样本定义:

- 死信队列
- 缺省传输队列
- CICS 适配器对象

不能按样使用此样本-在使用前必须对其进行定制。 然后，可以将此成员包含在队列管理器启动过程的 CSQINP2 DD 并置中，也可以将其用作 CSQUTIL 实用程序的 COMMAND 函数的输入以发出必需的 DEFINE 命令。 这是首选项，因为这意味着您不必在每次重新启动 IBM MQ 时重新定义这些对象。

除了此处的样本定义外，您还可以使用系统对象定义作为自己的资源定义的基础。 例如，您可以制作 SYSTEM.DEFAULT.LOCAL.QUEUE ，并将其命名为 MY.DEFAULT.LOCAL.QUEUE。 然后，您可以根据需要更改此副本中的任何参数。 然后，可以使用您选择的任何方法发出 DEFINE 命令，前提是您有权创建该类型的资源。

缺省传输队列
在确定是否要定义缺省传输队列之前，请先阅读 缺省传输队列 描述。

- 如果您决定要定义缺省传输队列，请记住还必须定义通道以提供服务。
- 如果您决定不想定义一个，请记住从样本中的 ALTER QMGR 命令中除去 DEFXMITQ 语句。

CICS 适配器对象
此样本定义名为 CICS01.INITQ。 此队列由 IBM MQ 提供的 CKTI 事务使用。 您可以更改此队列的名称; 但是，它必须与在 INITPARM 语句中的 CICS 系统初始化表 (SIT) 或 SYSIN 覆盖中指定的名称相匹配。

CSQ4INYS/CSQ4INYR 对象样本

用于使用的存储类定义:

- 为每类消息设置一个页面
- 主要消息类的多页集

例如， SYSTEM.COMMAND.INPUT 使用 STGCLASS ('SYSVOLAT') 和 SYSTEM.CLUSTER.TRANSMIT.QUEUE 使用 STGCLASS ('REMOTE')。 在 CSQ4INYS 中，这两个存储类都使用相同的页集。 在 CSQ4INYR 中，这些存储类使用不同的页集以减少传输队列填充的影响。

CSQINPT 样本

CSQ4INST
样本数据集 thlqual.SCSQPROC(CSQ4INST) 包含系统缺省预订的定义。

必须在此样本中定义对象，但只需在首次启动发布/预订引擎时执行一次。 在 CSQINPT 数据集中包含定义是执行此操作的最佳方法。 它在队列管理器关闭和重新启动之间进行维护。 您不得更改对象名，但可以根据需要更改其属性。

CSQ4INYT
样本数据集 thlqual.SCSQPROC(CSQ4INYT) 包含一组在启动发布/预订引擎时可能要运行的命令。 此样本显示主题和预订信息。

其他

CSQ4DISP 显示样本

样本数据集: thlqual.SCSQPROC(CSQ4DISP) 包含一组通用 DISPLAY 命令，这些命令显示队列管理器上的所有已定义资源。 这包括所有 IBM MQ 对象的定义以及诸如存储类和跟踪之类的定义。 这些命令可以生成大量输出。 可以在 CSQINP2 数据集中使用此样本，也可以将其用作 CSQUTIL 实用程序的 COMMAND 函数的输入。

CSQ4INPX 样本

样本数据集: thlqual.SCSQPROC(CSQ4INPX) 包含您每次启动通道启动程序时可能要执行的一组命令。 您必须先定制此样本，然后才能将其包含在通道启动程序的 CSQINPX 数据集中。

CSQ4IVPQ 和 CSQ4IVPG 样本

样本数据集 thlqual.SCSQPROC(CSQ4IVPQ) 和 thlqual.SCSQPROC(CSQ4IVPG) 包含运行安装验证程序 (IVP) 所需的 DEFINE 命令集。

可以在 CSQINP2 数据集中包含这些样本。 成功运行 IVP 后，每次重新启动队列管理器时都不需要再次运行这些 IVP。 因此，您不需要将这些样本永久保留在 CSQINP2 并置中。

CSQ4MSTR 和 CSQ4MSRR 样本

以下是队列管理器的样本启动式任务过程: thlqual.SCSQPROC(CSQ4MSTR) 和 thlqual.SCSQPROC(CSQ4MSRR)。

CSQ4MSRR 使用 CSQINP2 并置中的 CSQ4INYR ， 以便重要队列分布在不同页集之间。

您可以除去注释，以便可以根据需要将 CSQMINI 卡用于新创建的队列管理器。

CSQ4QMIN 样本

样本 QMINI 数据集 thlqual.SCSQPROC(CSQ4QMIN)。

请参阅 QMINI 数据集 ， 以获取 QMINI 数据集和 **TransportSecurity** 节的详细信息。

z/OS 在 z/OS 上恢复并重新启动

使用本主题中的链接可了解 IBM MQ for z/OS 用于重新启动和恢复的功能。

IBM MQ for z/OS 具有用于重新启动和恢复的强大功能。 有关队列管理器在停止后如何恢复以及重新启动时会发生什么的信息，请参阅以下子主题:

- 第 207 页的『如何在 IBM MQ for z/OS 中对数据进行更改』
- 第 208 页的『如何在 IBM MQ for z/OS 中保持一致性』
- 第 210 页的『在 IBM MQ for z/OS 中终止期间发生的情况』
- 第 211 页的『在 IBM MQ for z/OS 中重新启动和恢复期间发生的情况』
- 第 212 页的『如何解析不确定的恢复单元』
- 第 214 页的『共享队列恢复』

相关概念

z/OS IBM MQ for z/OS 恢复操作

相关任务

规划备份和恢复

z/OS 管理 z/OS

相关参考

z/OS IBM MQ for z/OS 的消息

z/OS 如何在 IBM MQ for z/OS 中对数据进行更改

IBM MQ for z/OS 必须与其他子系统交互以保持所有数据一致。 本主题包含有关 恢复单元的信息，它们是什么以及如何在 回退中使用它们。

恢复单元

恢复单元 是由应用程序的单个队列管理器完成的处理，用于将 IBM MQ 数据从一个一致性点更改为另一个一致性点。 一致性点 (也称为 同步点 或 落实点) 是应用程序访问的所有可恢复数据一致的时间点。

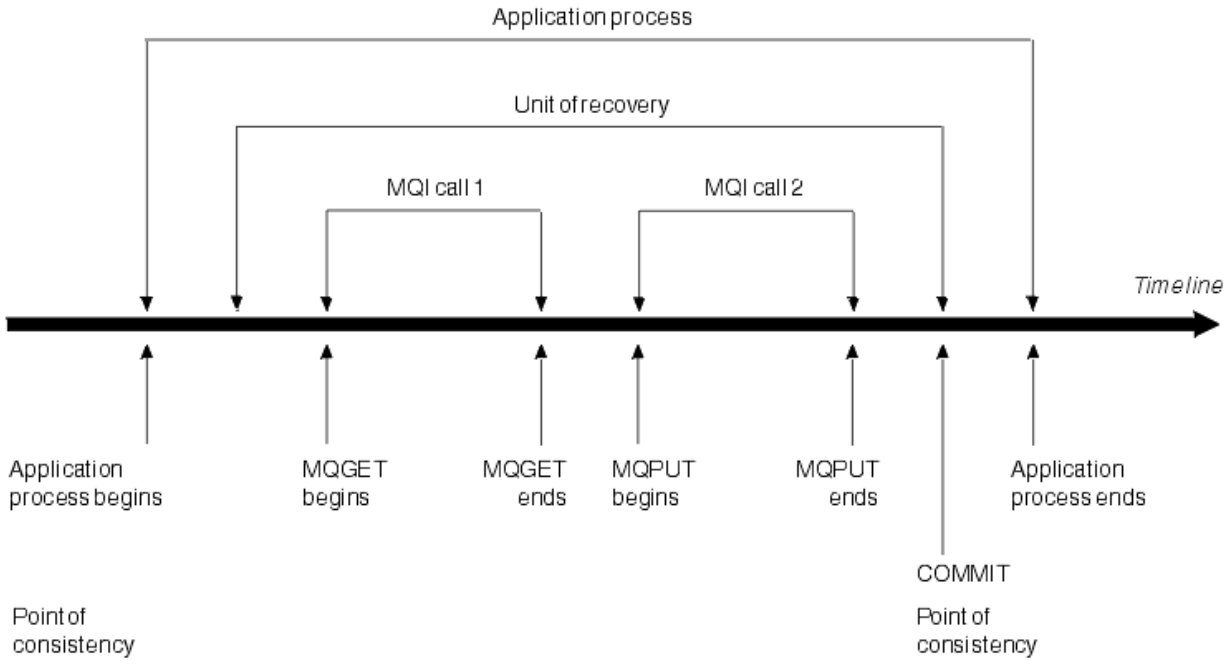


图 *75:* 应用程序中的恢复单元

恢复单元从程序开始后对数据的第一个更改开始，或者在前一个一致性点之后; 它以后一个一致性点结束。 第 207 页的图 75 显示了恢复单元，一致性点和应用程序之间的关系。 在此示例中，应用程序通过 MQI 调用 1 和 2 对队列进行更改。 应用程序可以包含多个恢复单元，也可以仅包含一个恢复单元。 但是，任何完整的恢复单元都将在落实点中结束。

例如，银行交易将资金从一个帐户转移到另一个帐户。 首先，程序从第一个科目 A 中减去金额。 然后，将金额添加到第二个帐户 B。 从 A 中减去金额后，这两个帐户不一致， IBM MQ 无法落实。 当将金额添加到科目 B 时，它们变为一致。 当这两个步骤都完成时，程序可以通过落实来声明一致性点，从而使更改对其他应用程序可见。

应用程序的正常终止会自动导致一致性点。 CICS 和 IMS 程序中的某些程序请求也会导致一致性点，例如 EXEC CICS SYNCPOINT。

回退工作

如果在恢复单元中发生错误，那么 IBM MQ 将除去对数据的任何更改，将数据返回到恢复单元启动时的状态; 即， IBM MQ 将回退工作。 这些事件显示在 第 208 页的图 76 中。

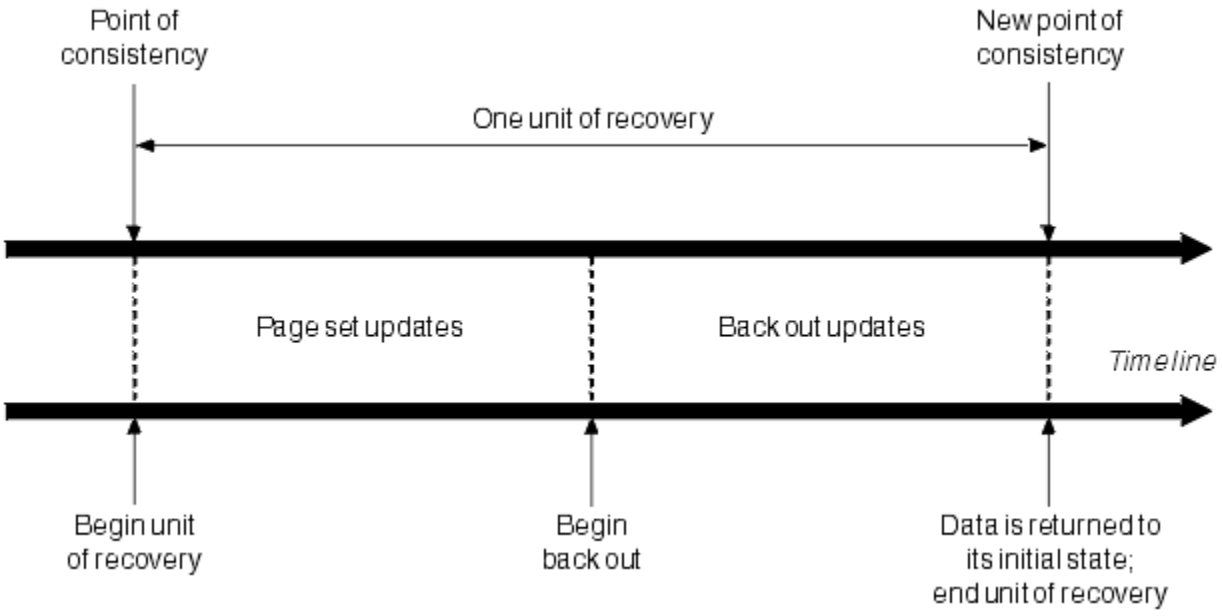


图 *76:* 显示回退的恢复单元

z/OS 如何在 IBM MQ for z/OS 中保持一致性

IBM MQ for z/OS 中的数据必须与批处理， CICS， IMS 或 TSO 一致。 在一个数据中更改的任何数据都必须与另一个数据中的更改相匹配。

在一个系统落实已更改的数据之前，它必须知道另一个系统可以进行相应的更改。 因此，系统必须进行通信。

在 两阶段落实 期间 (例如，在 CICS 下)，一个子系统协调进程。 该子系统称为 协调程序; 另一个是 参与者。 CICS 或 IMS 始终是与 IBM MQ 交互的协调程序，而 IBM MQ 始终是参与者。 在批处理或 TSO 环境中， IBM MQ 可以参与由 z/OS RRS 协调的两阶段落实协议。

在 单阶段落实 期间 (例如，在 TSO 或批处理下)， IBM MQ 始终是交互中的协调程序，并完全控制落实过程。

在 WebSphere Application Server 环境中， JMS 会话对象的语义确定是使用单阶段还是两阶段落实协调。

与 CICS 或 IMS 的一致性

IBM MQ 与 CICS 或 IMS 之间的连接支持以下同步点协议:

- 两阶段落实-用于更新由多个资源管理器拥有的资源的事务。

  这是标准的分布式同步点协议。 它涉及比单阶段落实更多的日志记录和消息流。
- 单阶段落实-用于更新单个资源管理器 (IBM MQ) 所拥有的资源的事务。

  此协议针对日志记录和消息流进行了优化。
- 绕过同步点-对于涉及 IBM MQ 的事务，但在需要同步点的队列管理器中不执行任何操作 (例如，浏览队列)。

在每种情况下， CICS 或 IMS 都充当同步点管理器。

IBM MQ 用于与 CICS 或 IMS 通信的两阶段落实的阶段如下所示:

1. 在阶段 1 中，每个系统独立确定其日志中是否记录了足够的恢复信息，并且可以落实其工作。

   在阶段结束时，系统进行通信。 如果他们同意，每个人都开始下一阶段。

2. 在阶段 2 中，更改是永久性的。 如果其中一个系统在阶段 2 期间异常终止，那么该操作由恢复过程在重新启动期间完成。

两阶段落实过程的说明

第 209 页的图 77 说明了两阶段落实过程。 CICS 或 IMS 协调程序中的事件显示在上行， IBM MQ 中的事件显示在下行。

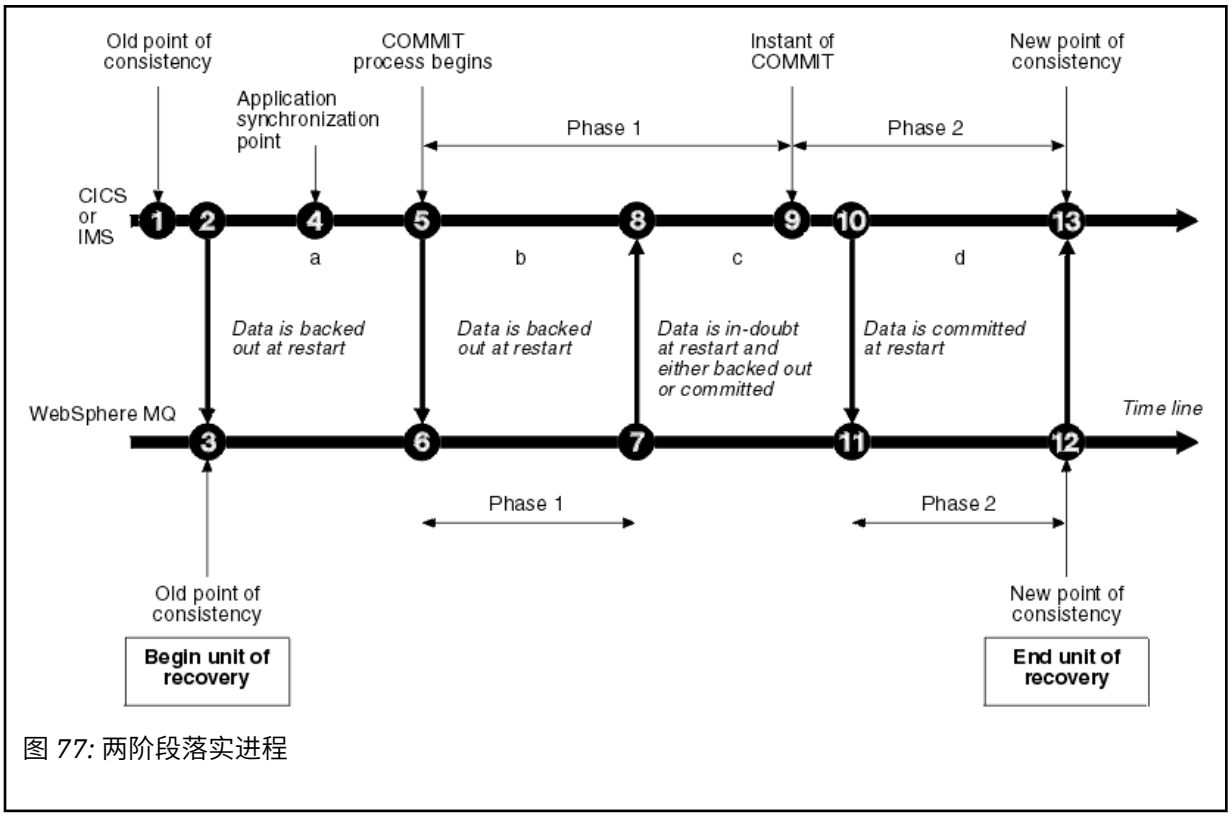


图 77: 两阶段落实进程

以下部分中的数字与图中显示的数字相链接。

1. 协调程序中的数据是一致的。
2. 协调程序中的应用程序通过添加消息来调用 IBM MQ 以更新队列。
3. 这将在 IBM MQ 中启动恢复单元。
4. 在协调程序中继续处理，直到到达应用程序同步点为止。
5. 然后，协调程序启动落实处理。 CICS 程序使用 SYNCPOINT 命令或正常应用程序终止来启动落实。 IMS 程序可以使用 CHKP 调用， SYNC 调用，对 IOPCB 的 GET UNIQUE 调用或正常应用程序终止来启动落实。 落实处理的阶段 1 开始。
6. 当协调程序开始阶段 1 处理时， IBM MQ 也是如此。
7. IBM MQ 成功完成阶段 1 ，将此事实写入其日志，并通知协调程序。
8. 协调程序接收通知。
9. 协调程序已成功完成其阶段 1 处理。 现在，两个子系统都同意落实数据更改，因为这两个子系统都已完成阶段 1 ，并且可以从任何错误中恢复。 协调程序在其日志中记录落实的瞬间-两个子系统进行更改的不可撤销决策。

   协调程序现在开始处理的阶段 2-实际落实。
10. 协调程序通知 IBM MQ 开始其阶段 2。
11. IBM MQ 记录阶段 2 的开始。
12. 阶段 2 已成功完成，现在这是 IBM MQ 的新一致性点。 IBM MQ 然后通知协调程序它已完成其阶段 2 处理。
13. 协调程序完成其阶段 2 处理。 由两个子系统控制的数据现在是一致的，可供其他应用程序使用。

异常终止后如何保持一致性

在异常终止后重新启动队列管理器时，它必须确定是落实还是回退在终止时处于活动状态的恢复单元。 对于某些恢复单元， IBM MQ 具有足够的信息来做出决策。 对于其他用户，它不会执行此操作，并且必须在重新建立连接时从协调程序获取信息。

第 209 页的图 77 显示两个阶段中的四个周期 :a， b， c 和 d。 恢复单元的状态取决于发生终止的时间段。 状态可以是以下项之一：

在飞行中
队列管理器在完成阶段 1 (时间段 a 或 b) 之前结束; 在重新启动期间， IBM MQ 会回退更新。

不确定
队列管理器在完成阶段 1 之后以及在开始阶段 2 之前结束 (周期 c); 只有协调程序知道是在落实之前还是之后发生了错误 (点 9)。 如果是之前发生的，那么 IBM MQ 必须回退其更改; 如果是之后发生的，那么 IBM MQ 必须进行其更改并落实这些更改。 重新启动时， IBM MQ 将在处理此恢复单元之前等待来自协调程序的信息。

在落实中
队列管理器在开始其自己的阶段 2 处理 (周期 d) 后结束; 它进行已落实的更改。

在回退中
队列管理器在恢复单元开始回退后结束，但在重新启动期间进程完成 (图中未显示) 之前， IBM MQ 继续回退更改。

z/OS 在 IBM MQ for z/OS 中终止期间发生的情况

队列管理器正常终止以响应 STOP QMGR 命令。 如果队列管理器由于任何其他原因而停止，那么终止会异常。

请注意，在队列管理器终止期间， IBM MQ 在内部发出命令

```
DISPLAY CONN(*) TYPE(CONN) ALL WHERE (APPLTYPE NE SYSTEMAL)
```

以便您了解哪些线程可能会阻止队列管理器完成关闭。

SYSTEMAL 与 SYSTEM 或 CHINIT 的 APPLTYPES 相匹配，因此 DISPLAY CONN 命令将过滤与 SYSTEMAL 不匹配的应用程序类型，返回到可能阻止正常关闭的线程的作业记录信息。

正常终止

在正常终止中， IBM MQ 以有序方式停止所有活动。 您可以使用停顿，强制或重新启动方式来停止 IBM MQ 。 效果在 第 210 页的表 23 中给出。

表 *23:* 使用 *QUIESCE* ， *FORCE* 和 *RESTART* 终止

线程类型	**QUIESCE**	**FORCE**	**RESTART**
活动线程数	运行到完成	回退	回退
新线程数	可以启动	不允许	不允许
新连接	不允许	不允许	不允许

如果在应用程序仍处于连接状态时发生终止，那么将通知批处理应用程序。

使用 CICS 时，当前线程仅运行到恢复单元的末尾。 对于 CICS，以停顿方式停止队列管理器将停止 CICS 适配器，因此如果活动任务包含多个恢复单元，那么该任务不一定运行到完成。

如果以强制或重新启动方式停止队列管理器，那么不会分配任何新线程，并且将回滚已连接的线程上的工作。 使用这些方式可以为处于落实处理阶段之间的线程创建不确定的恢复单元。 当 IBM MQ 与控制 CICS， IMS 或 RRS 子系统重新连接时，将解决这些问题。

在任何方式下停止队列管理器时，步骤如下：

1. 连接已结束。
2. IBM MQ 停止接受命令。
3. IBM MQ 确保完成对页集的任何未完成的更新。
4. IBM MQ 在内部发出 DISPLAY USAGE 命令，以便在 z/OS 控制台日志上记录重新启动 RBA。
5. 将执行关闭检查点并更新 BSDS。

指定停顿方式的终止不会影响不确定的恢复单元。 任何有疑问的单位都是有疑问的。

异常终止

异常终止可能会使数据处于不一致状态，例如:

- 恢复单元在到达一致性点之前已中断。
- 尚未将已落实的数据写入页集。
- 未落实的数据已写入页集。
- 应用程序已在落实进程的阶段 1 和阶段 2 之间中断，使恢复单元处于不确定状态。

IBM MQ 解决了在重新启动和恢复期间由于异常终止而产生的任何数据不一致。

z/OS 在 IBM MQ for z/OS 中重新启动和恢复期间发生的情况

IBM MQ 使用其恢复日志和引导数据集 (BSDS) 来确定重新启动时要恢复的内容。 BSDS 标识活动日志数据集和归档日志数据集，以及日志中最新 IBM MQ 检查点的位置。

重新启动和恢复简介

初始化 IBM MQ 后，将按如下所示执行队列管理器重新启动过程:

- 日志初始化
- 当前状态重建
- 转发日志恢复
- 反向日志恢复
- 队列索引重建

恢复完成后:

- 已落实的更改将反映在数据中。
- 不确定活动反映在数据中。 但是，在 IBM MQ 识别并处理不确定决策之前，数据已锁定并且无法使用。
- 已从队列中除去中断的正在进行的更改和正在中止的更改。 这些消息是一致的，可以使用。
- 已采用新的检查点。
- 已为包含持久消息的已建立索引的队列构建新索引 (如 第 212 页的『重建队列索引』 中所述)。

如果正在使用双 BSD ，那么 IBM MQ 会检查 BSDS 中时间戳记的一致性:

- 如果 BSDS 的两个副本都是最新的，那么 IBM MQ 会测试两个时间戳记是否相等。 如果不是， IBM MQ 将发出消息 CSQJ120E 并终止。 当 BSDS 的两个副本在单独的 DASD 卷上维护，并且在队列管理器停止时复原了其中一个卷时，可能会发生此情况。 IBM MQ 在重新启动时检测到情境。
- 如果取消分配 BSDS 的一个副本，并且继续使用单个 BSDS 进行日志记录，那么可能会出现问题。 如果两个 BSDS 副本都在单个卷上维护，并且卷已复原，或者如果两个 BSDS 副本都单独复原，那么 IBM MQ 可能不会检测到复原。 在这种情况下，系统将不知道 BSDS 中未记录的日志记录。

在应用程序请求连接 之后 发生重新启动时，不会通知批处理应用程序。

了解恢复所需的日志范围

在重新启动期间，必须读取的日志数据范围取决于许多因素:

- 在异常终止时，系统中通常有许多不完整的工作单元。 如前所述，重新启动处理将使系统处于一致性状态，这可能涉及回退正在运行的工作单元，或恢复对不确定工作单元的锁定。 工作单元恢复要求所有工作单元日志记录都可用于进行中工作单元，回退工作单元和不确定工作单元。 IBM MQ 将 "分流" 旧的工作单元，以便可以使用更小范围的日志数据来执行工作单元恢复。
- 在异常终止时，通常有许多持久更新仅保存在缓冲池高速缓存中。 它们尚未写入磁盘。 这些更改必须从日志中读取，并重新应用于页集中保存的数据。 检查点中的页集恢复 RBA 描述将页集更新为一致状态所需的最低日志 RBA。
- 如果已将旧页集引入到系统中，例如，已引入页集备份以从介质故障中恢复，那么必须从执行备份时的日志中读取所有更改。 这些更改将重新应用于正在恢复的页集中保存的数据。 页集的第 0 页中保存的页集恢复 RBA 描述了页集的介质恢复所需的最低日志 RBA。
- 如果在共享队列上使用持久消息，那么需要一系列日志数据来恢复保存持久消息的 CFSTRUCT。 执行 CFSTRUCT 恢复所需的最早日志数据来自旧 CFSTRUCT BACKUP 的时间。

在正常运行期间，可以使用 DISPLAY USAGE TYPE (DATASET) 命令来查看与这些因素相关联的恢复日志范围 (当然，由于重新引入旧页集，它无法提供信息)。 为避免在异常终止时可能延长队列管理器重新启动的任何问题，请定期监视 DISPLAY USAGE TYPE (DATASET) 输出的值。

此外，队列管理器还会发出与以下因素相关的参考消息:

- CSQJ160I 和 CSQJ161I 警告长时间运行的工作单元。
- CSQR026I 和 CSQR027I 提供有关这些长时间运行的工作单元是否已成功舍入的信息。
- CSQE040I 和 CSQE041E 警告结构备份越来越旧，因此 RECOVER CFSTRUCT 操作将需要很长时间。

确定哪个应用程序具有长时间运行的工作单元

可以确定具有长时间运行的工作单元的应用程序。 要执行此操作，请使用 DISPLAY CONN 命令。

DISPLAY CONN 命令返回连接到队列管理器的所有应用程序的连接信息以及其他信息，这些信息可帮助您确定哪些应用程序当前具有长时间运行的工作单元。 DISPLAY CONN 命令返回的信息与 DISPLAY QSTATUS 命令返回的信息类似，但主要区别在于 DISPLAY CONN 显示有关特定连接的对象和事务信息的信息，而不是与特定对象关联的连接的详细信息。

对于每个已连接的应用程序， DISPLAY CONN 命令会返回以下信息:

- 基本信息，包括连接标识和 PID。
- 该连接的事务信息，包括创建事务的时间和日期 (即，在同步点下执行第一个 MQGET/PUT 时) 以及事务首次写入日志时的时间和日期。
- 记录时间信息，指示哪个应用程序仍具有长时间运行的工作单元。
- 连接当前已打开的所有对象的列表。 每个对象的详细信息将作为单独的消息返回，并将 "连接标识" 用作键。 由于存在不同类型的对象 (例如队列和队列管理器) ，因此随对象显示的信息特定于其对象类型。

重建队列索引

要提高未按顺序检索消息的队列上的 MQGET 操作的速度，您可以指定希望 IBM MQ 维护该队列上所有消息的消息索引或相关标识或组标识。

重新启动队列管理器时，将为每个队列重建这些索引。 这仅适用于持久消息; 非持久消息将在重新启动时删除。 如果已建立索引的队列包含大量持久消息，那么这将增加重新启动队列管理器所花费的时间。

您可以选择使用 CSQ6SYSP 宏的 QINDXBLD 参数来异步重建索引以启动队列管理器。 如果设置 QINDXBLD=NOWAIT ，那么 IBM MQ 将重新启动而不等待重建索引。

z/OS 如何解析不确定的恢复单元

如果 IBM MQ 失去与另一个资源管理器的连接，那么它通常会尝试在重新启动时恢复所有不一致的对象。

如果 IBM MQ 失去了与 CICS， IMS 或 RRS 的连接，那么通常会尝试在重新启动时恢复所有不一致的对象。解析不确定的恢复单元所需的信息必须来自协调系统。 以下各部分描述了针对不同环境的解析过程。

- 如何从 CICS 解析不确定的恢复单元
- 如何从 IMS 解析不确定的恢复单元
- 如何从 RRS 解析不确定的恢复单元
- 如何解析具有 GROUP 恢复处置单元的不确定恢复单元

如何从 CICS 解析不确定的恢复单元

在某些情况下， CICS 无法运行 IBM MQ 进程来解析不确定的恢复单元。 发生此情况时， IBM MQ 将发送下列其中一条消息:

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E

后跟消息 CSQC408I。

有关这些消息的含义的详细信息，请参阅 IBM MQ for z/OS 消息、完成和原因代码 手册。

不确定单元的解析不会影响 CICS 资源。 CICS 控制恢复协调，并在重新启动时自动落实或回退每个单元，具体取决于是否存在标记落实开始的日志记录。 在重新连接 IBM MQ 时，不确定对象的存在不会锁定 CICS 资源。

CICS 适配器的其中一项功能是在 CICS 和 IBM MQ 之间保持数据同步。 如果队列管理器在连接到 CICS 时异常终止，那么 CICS 可能会在不知道 IBM MQ 的情况下落实或回退工作。 当队列管理器重新启动时，该工作称为 不确定。

IBM MQ 无法解析这些不确定的恢复单元 (即，落实或回退对 IBM MQ 资源所作的更改) ，直到重新启动或重新连接到 CICS 的连接为止。

在 CICS 适配器启动期间，将启动用于解析不确定恢复单元的过程。 当适配器请求不确定的恢复单元列表时，该进程将启动。 过去:

- 适配器从 IBM MQ 接收此连接标识的不确定恢复单元列表，并将其传递到 CICS 以进行解析。
- CICS 将此列表中的条目与其自己的日志中的条目进行比较。 CICS 从其自己的列表中确定它对每个不确定的恢复单元执行的操作。

对于所有已解析的单元， IBM MQ 会根据需要更新队列并释放相应的锁定。 未解析的单元可以在重新启动后保留。 通过 管理 IBM MQ for z/OS 中描述的方法来解决这些问题。

如何从 IMS 解析不确定的恢复单元

在 IMS 中解析不确定的恢复单元不会影响 DL/I 资源。 IMS 控制恢复协调，并且在重新启动时自动落实或回退不完整的 DL/I 工作。 是否存在 IMS 日志记录类型 X'3730 '和 X'3801' 的联机区域 (非快速路径) 的落实或回退决策。 存在不确定的恢复单元并不意味着在 IBM MQ 连接之前 DL/I 记录处于锁定状态。

在队列管理器重新启动期间， IBM MQ 会生成不确定恢复单元的列表。 IMS 会构建其自己的残差恢复条目 (RREs) 列表。 在解析所有条目之前，会在 IMS 检查点记录 RREs。

在将 IMS 区域重新连接到 IBM MQ 期间， IMS 向 IBM MQ 指示是落实还是回退 IBM MQ 中标记为不确定的工作单元。

解析不确定单元时:

1. 如果 IBM MQ 识别到它已标记要落实的条目，并且 IMS 已将其标记为要回退，那么 IBM MQ 会发出消息 CSQQ010E。 IBM MQ 针对 IBM MQ 与 IMS 之间此类型的所有不一致情况发出此消息。
2. 如果 IBM MQ 具有任何剩余的不确定单元，那么适配器将发出消息 CSQQ008I。

对于所有已解析的单元， IBM MQ 会根据需要更新队列并释放相应的锁定。

IBM MQ 维护对未解决的不确定工作的锁定。 如果挂起重要的锁定，那么这可能会导致系统中的积压。 该连接仍处于活动状态，因此您可以解析 IMS RREs。 通过 管理 IBM MQ for z/OS 中描述的方法恢复不确定线程。

除非存在软件或操作问题 (例如 IMS 冷启动) ，否则应解决所有不确定工作。 IMS 控制区域在两种情况下进行不确定解析:

1. 在与 IBM MQ 的连接开始时，将以同步方式完成解析。
2. 当程序异常终止时，在此期间将以异步方式完成解析。

如何从 RRS 解析不确定的恢复单元

RRS 适配器的其中一个功能是在 IBM MQ 和其他参与 RRS 的资源管理器之间保持数据同步。 如果 IBM MQ 已完成落实的第一阶段，并且正在等待来自 RRS (落实协调程序) 的决策时发生故障，那么恢复单元将进入不确定状态。

在 RRS 与 IBM MQ 之间重新建立通信时， RRS 会自动落实或回退每个恢复单元，这取决于是否存在标记落实开始的日志记录。 在重新建立与 RRS 的连接之前， IBM MQ 无法解析这些不确定的恢复单元 (即，落实或回退对 IBM MQ 资源所作的更改)。

在某些情况下， RRS 无法解决不确定的恢复单元。 发生此情况时， IBM MQ 会将下列其中一条消息发送到 z/OS 控制台:

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

有关这些消息的含义的详细信息，请参阅 IBM MQ for z/OS 消息、完成和原因代码 手册。

对于所有已解析的恢复单元， IBM MQ 会根据需要更新队列并释放相应的锁定。 未解析的恢复单元可以在重新启动后保留。 通过 管理 IBM MQ for z/OS 中描述的方法来解决这些问题。

如何解析具有 GROUP 恢复处置单元的不确定恢复单元

具有 GROUP 恢复单元处置的不确定事务可由事务协调程序由启用了 GROUPUR 队列管理器属性的队列共享组 (QSG) 中的任何队列管理器解析。 每当事务协调程序重新连接时，它通常会请求任何未完成的不确定事务的列表，然后使用其日志中的信息来解析这些事务。

当已连接到 GROUP 恢复处置单元的事务协调程序请求不确定事务列表时，返回的列表包含在队列共享组中存在的具有 GROUP 恢复处置单元的所有不确定事务。 此列表不依赖于那些不确定事务是在哪个队列管理器上启动的。 处理此类请求的队列管理器通过使用 SYSTEM.QSG.UR.RESOLUTION.QUEUE。 然后，队列管理器会从任何不活动队列管理器的最后一个检查点读取这些队列管理器的日志，以确定如果这些队列管理器处于活动状态，它们将报告的任何其他不确定事务。

当事务协调程序请求解析不确定事务时，它所连接的队列管理器会识别事务是否源自自身，如果是，那么会以与具有 QMGR 恢复处置单元的事务相同的方式解析该事务。 如果事务是在 QSG 中的另一个活动队列管理器上发起的，那么将使用 SYSTEM.QSG.UR.RESOLUTION.QUEUE。 如果事务是在 QSG 中的不活动队列管理器上发起的，那么将立即解析任何共享队列工作，并将解析任何剩余专用队列工作的请求放在 SYSTEM.QSG.UR.RESOLUTION.QUEUE。 在接受新工作之前，不活动的队列管理器会在启动时处理此请求。 在此场景中，原始队列管理器的日志仍反映恢复单元处于不确定状态，直到它重新启动并处理请求为止。

z/OS 共享队列恢复

使用本主题来了解队列共享组环境中各种组件的 IBM MQ 恢复和弹性。

- 第 215 页的『事务恢复』
- 第 215 页的『对等恢复』

事务恢复

当应用程序发出 MQBACK 调用或异常终止 (例如，由于 EXEC CICS ROLLBACK 或 IMS 异常终止) 队列管理器中存储的线程级别信息) 时，将确保回滚进行中的工作单元。 共享队列上同步点内的 MQPUT 和 MQGET 操作将以与对非共享队列的更新相同的方式回滚。

对等恢复

如果队列管理器发生故障，那么它将与当前连接到的耦合设施结构异常断开连接。 如果 z/OS 实例与耦合设施之间的连接失败 (例如，耦合设施或分区的物理链路故障或断电) ，那么也会检测到队列管理器与所涉及的耦合设施结构之间的连接异常终止。 保持连接到该结构的同一队列共享组中的其他队列管理器会检测到异常断开连接，并且所有尝试对该结构上失败的队列管理器启动 对等恢复 。 只有其中一个队列管理器成功启动对等恢复，但所有其他队列管理器都协作恢复失败的队列管理器所拥有的工作单元。

如果队列管理器在没有同级连接到结构时失败，那么当另一个队列管理器连接到该结构时，或者当失败的队列管理器重新启动时，将执行恢复。

对等恢复 (通常称为对等级别恢复 (Peer Level Recovery ， PLR)) 是按结构执行的，单个队列管理器可以同时参与多个结构的恢复。 但是，根据发生故障时连接到不同结构的队列管理器，合作恢复不同结构的同级的集合可能有所不同。

当失败的队列管理器重新启动时，它将重新连接到它在发生故障时所连接到的结构，并恢复未由同级恢复恢复的任何剩余未解决的工作单元。

对等恢复是一个多阶段过程。 在第一阶段中，将恢复在未完成阶段之后前进的工作单元; 这可能涉及落实正在落实的工作单元的消息，以及锁定不确定的工作单元的消息。 在第二阶段中，将检查在发生故障的队列管理器中具有活动线程的队列，回滚与未完成的工作单元相关的未落实消息，并重置有关发生故障的队列管理器中共享队列上的活动句柄的信息。 这意味着 IBM MQ 会重置任何指示符，这些指示符指示发生故障的队列管理器已打开共享队列以进行互斥输入，从而允许其他活动队列管理器打开队列以进行输入。

共享队列定义

表示共享队列属性的队列对象保存在队列共享组使用的共享 Db2 存储库中。 确保已执行足够的过程来备份和恢复用于存放 IBM MQ 对象的 Db2 表。 您还可以使用 IBM MQ CSQUTIL 实用程序来创建 MQSC 命令，以重放到队列管理器中，从而重新定义 IBM MQ 对象，包括存储在 Db2 中的共享队列和组定义。

记录

队列共享组可以支持持久消息，因为共享队列上的消息可以记录在队列管理器日志中。

耦合设施和结构故障

对于耦合设施 (CF) 结构，可以报告两种类型的故障: 结构故障和连接丢失。 用于数据共享 (XES) 的综合系统服务向 IBM MQ 通知 CF 结构故障或具有结构故障事件的 CF 故障。 如果 XES 创建了丢失连接事件，这不一定表示结构存在问题，可能是没有可用于与结构通信的连接。 可能并非所有队列管理器都接收到该结构的连

接丢失事件; 这取决于与 CF 的连接的配置。 由于操作员命令 (例如 VARY PATH OFFLINE 或 CONFIG CHP OFFLINE) ， 也可能接收到丢失连接事件。

可以将 IBM MQ 使用的 CF 结构配置为使用系统管理的双工。 这意味着如果发生单个故障，那么系统管理的故障转移处理将隐藏结构故障或连接中断，并且不会通知队列管理器该故障。 如果双工结构或连接的两个实例都发生故障，那么队列管理器将接收相应的事件，并以与单纯形法结构的故障事件相同的方式处理该事件。 方案中描述了队列管理器如何处理事件的详细信息。

在不太可能发生 CF 或结构故障的情况下，存储在受影响应用程序结构中的任何非持久消息都将丢失。 可以使用 RECOVER CFSTRUCT 命令来恢复持久消息。 如果可恢复的应用程序结构已失败，那么将阻止此结构的任何进一步应用程序活动，直到该结构已恢复为止。

要确保可以在合理的时间段内恢复 CF 结构，请使用 BACKUP CFSTRUCT 命令进行频繁备份。 您可以选择在队列共享组中的任何队列管理器上执行备份，或者指定一个队列管理器来执行所有备份。 自动执行备份过程，以确保定期执行备份。

每个备份都将写入执行备份的队列管理器的活动日志数据集。 共享队列 Db2 存储库记录正在备份的 CF 结构的名称，执行备份的队列管理器的名称，该队列管理器日志上此备份的 RBA 范围以及备份时间。

管理结构包含有关在任何应用程序结构发生故障时共享队列上的不完整工作单元的信息，因此管理结构必须在 RECOVER CFSTRUCT 处理期间可用。 如果管理结构已失败，那么队列共享组中的所有队列管理器必须先重建其管理结构条目，然后才能发出 RECOVER CFSTRUCT 命令。

队列管理器会自动重新构建其管理结构条目，而不会终止。 如果队列管理器在发生故障时未运行，那么其管理结构条目可由同一级别或更高级别运行的队列共享组中的另一个队列管理器重建。

要恢复应用程序结构，请向要执行恢复的队列管理器发出 RECOVER CFSTRUCT 命令。 您可以恢复单个 CF 结构，也可以同时恢复多个 CF 结构。 您可以使用队列共享组中的任何队列管理器进行恢复，它不必是执行备份的队列管理器，也不必是先前已连接到失败结构的队列管理器。

RECOVER CFSTRUCT 命令使用通过 Db2 存储库信息 (因此必须在执行恢复的队列管理器上提供 Db2) 找到的备份，并将此恢复到故障点。

RECOVER CFSTRUCT 命令通过将来自队列共享组中在备份启动和失败时间之间执行 MQPUT 或 MQGET 的每个队列管理器的日志记录应用于映射到 CF 结构的任何共享队列来执行此操作。 生成的日志合并可能需要读取大量日志数据，因为自读取备份以来参与队列管理器写入的所有日志数据。 强烈建议您进行频繁 (例如，每小时) 备份，尤其是在备份中存在大量消息时。

结构故障场景

方案

如果针对 CF 结构报告了故障，那么已连接的队列管理器所执行的操作取决于以下内容:

- z/OS 的 XES 组件向 IBM MQ 报告的故障类型。
- 结构类型 (应用程序或管理)
- 队列管理器级别
- IBM MQ CFSTRUCT 对象 (2 ， 3 ， 4 或 5) 的 CFLEVEL。 这不是 CFCC 微码的 CFLEVEL)
- IBM MQ CFSTRUCT 对象在 CFLEVEL 的 RECAUTO 属性 (5)

以下场景描述了针对管理结构报告故障时发生的情况:

- 如果接收到管理结构的结构故障事件，那么将自动重新分配并重建该结构，而不会终止队列管理器。 当队列管理器尝试连接到该结构的新实例时， XES 会分配该结构的新实例。

  当队列管理器已连接到结构的新实例时，队列管理器会将其自身的条目写入结构中。 此处理由队列管理器执行，并且不是 XES 重建处理的一部分。

  如果队列管理器在发生故障时未运行，或者在其管理结构部分的恢复完成之前终止，那么队列共享组中的另一个队列管理器可以重建其管理结构条目。

  队列管理器的管理结构条目只能由在同一级别或更高级别运行的另一个队列管理器重建。 如果队列管理器的管理结构条目无法由队列共享组中的另一个队列管理器重建，请重新启动队列管理器，以便它可以完成其部分结构的重建。

在重建所有队列管理器的管理结构条目之前，将暂挂某些操作。 暂挂的操作包括:

- 打开和关闭共享队列。
- 正在落实或回退恢复单元。
- 连接到队列管理器或与队列管理器断开连接的序列化应用程序。
- 备份或恢复应用程序结构。

任何已连接到队列管理器的序列化应用程序都可以继续处理。 任何尝试与 MQCNO_SERIALIZE_CONN_TAG_QSG 或 MQCNO_RESTRICT_CONN_TAG_QSG 参数连接的序列化应用程序都会收到 MQRC_CONN_TAG_NOT_USABLE 返回码。

重建队列管理器的管理结构条目后，将恢复暂挂的操作。

以下场景描述了针对应用程序结构报告故障时发生的情况:

- 如果接收到应用程序结构的结构故障事件，并且 CFLEVEL 为 1 或 2，那么队列管理器将终止。 重新启动队列管理器。 尝试再次连接到结构的第一个队列管理器会导致 XES 分配该结构的新实例。
- 如果接收到应用程序结构的结构故障事件，并且 CFLEVEL 为 3， 4 或 5，那么连接到该结构的队列管理器将继续运行。 不使用失败结构中的队列的应用程序可以继续正常处理。

  但是，尝试对失败结构中的队列执行操作的应用程序会收到 MQRC_CF_STRUC_FAILED 错误，直到成功重建该结构为止，此时应用程序可以再次打开队列。

  对于使用 RECAUTO (YES) 定义的 CFLEVEL (5) 应用程序结构，将自动启动结构重建。 否则，当发出 RECOVER CFSTRUCT 命令时将重建结构。

对耦合设施连接故障的弹性

什么是耦合设施连接故障的弹性?

对耦合设施连接失败的弹性是指队列共享组中的队列管理器能够容忍与耦合设施结构的连接丢失而不终止。 此功能还会尝试在另一个具有更好连接的耦合设施中重建结构，以便尽快重新获得对共享队列的访问权。

什么是部分丢失连接?

IBM MQ 将部分失去连接定义为以下情况: 综合系统中的一个或多个系统失去与耦合设施的连接，在该耦合设施中分配了系统所访问的结构，但综合系统中的至少一个系统维护与同一耦合设施的连接。

什么是完全失去连接?

IBM MQ 定义了完全失去连接的情况，即综合系统中没有任何系统具有与耦合设施的连接以及在其中分配的结构。

为什么要启用此功能?

对耦合设施连接故障的弹性可提高 IBM MQ 的可用性，允许非共享队列在队列管理器与一个或多个耦合设施结构的连接中断后保持可用。 此外，与耦合设施结构失去连接的队列管理器会自动尝试在另一个可用的耦合设施中重建结构，从而提高队列共享组中共享队列的可用性。

启用此功能时的注意事项

如果没有可用的备用耦合设施，那么允许在未终止的情况下断开与耦合设施结构的连接的队列管理器可能无法在一段时间内重新连接到耦合设施结构。 在失去连接的结构上定义的共享队列将保持不可用状态，直到恢复与该结构的连接为止。 在这种情况下，为了执行共享队列工作而连接到队列共享组成员的应用程序可能会发现他们需要访问的共享队列不可用。 为了避免这种情况，建议将队列管理器配置为在与耦合设施结构的连接中断时终止。 此终止会强制应用程序连接到队列共享组的另一个成员，该成员具有与耦合设施结构的连接，在该耦合设施结构中定义了应用程序所需的共享队列。

管理耦合设施连接故障的弹性

如何启用此功能?

必须执行以下步骤才能启用对耦合设施连接的弹性

1. 确保 CFRM 耦合数据集已格式化为支持系统管理的重建。 这允许队列管理器启动系统管理的重建，以在可用耦合设施中重新创建结构。 使用 **DISPLAY XCF,COUPLE,TYPE=CFRM** 命令可确定 CFRM 耦合数据集的格式。 要支持系统管理的重建，应通过指定以下内容来格式化 CFRM 耦合数据集:

   ```
   "ITEM NAME(SMREBLD) NUMBER(1)"
   ```

   请参阅 z/OS MVS 设置综合系统 文档，以获取有关格式化 CFRM 耦合数据集的更多信息。
2. 确保备用耦合设施可用并且位于所有 IBM MQ 耦合设施结构的 CFRM 首选项列表中。 这使队列管理器能够尝试将结构重建到备用可用耦合设施中，以尽快恢复对结构的访问权。

   必须在 CFRM 策略中使用 ENFORCEORDER (NO) 定义 IBM MQ 结构，以便如果 IBM MQ 需要重新分配结构，那么 XCF 能够在配置中选择最佳 CF。

   有关结构首选项列表的更多信息，请参阅 z/OS MVS 设置综合系统 文档。
3. 更改需要容许与 CFLEVEL (5) 的连接丢失的所有应用程序耦合设施结构。 这是可容许丢失连接的最低级别。
4. 确定 **QMGR CFCONLOS** 和 **CFSTRUCT CFCONLOS** 属性所需的值，并相应地更改这些值。 **QMGR CFCONLOS** 属性控制是否允许丢失与管理结构的连接， **CFSTRUCT CFCONLOS** 属性控制每个应用程序耦合设施结构是否允许丢失连接。 如果保留这些属性的缺省值，那么队列管理器将在与任何耦合设施结构的连接中断后终止。
5. 确定每个应用程序耦合设施结构的 **CFSTRUCT RECAUTO** 属性所需的值，并相应地更改这些值。 此属性控制在完全失去连接后，是否应该使用记录的数据自动恢复耦合设施结构。 如果保留此属性的缺省值，那么在完全失去连接后不会对应用程序结构执行自动恢复。

方案 1-与管理结构的连接中断

队列管理器可以容许与管理结构的连接中断而不终止。

当已配置为容许与管理结构的连接丢失的任何队列管理器失去与管理结构的连接时，队列共享组的所有成员都将断开与管理结构的连接。 然后，队列共享组中的所有活动队列管理器尝试重新连接到管理结构，使其在耦合设施中重新分配，并与综合系统中的所有系统建立最佳连接，然后重新构建管理结构数据。

注: 这不一定是与具有活动队列管理器的所有系统具有最佳连接的耦合设施。

如果队列管理器无法重新连接到管理结构，例如，由于管理结构的 CFRM 首选项列表中的耦合设施都不可用，那么某些共享队列操作将保持不可用状态，直到队列管理器可以成功重新连接到管理结构并重建其管理结构数据为止。 当合适的耦合设施在系统上变为可用时，将自动进行重新连接。

在队列管理器启动期间，由于缺少与耦合设施的连接，或者没有合适的耦合设施可用于分配该结构，因此无法连接到管理结构。 然后，队列共享组中的所有活动队列管理器尝试重新连接到管理结构，使其在另一个耦合设施 (如果有) 中重新分配，并重建管理结构数据。

方案 2-与应用程序结构的连接中断

在不终止队列管理器的情况下，可以容许与 **CFLEVEL(5)** 或更高版本的应用程序结构的连接中断。 连接到 **CFLEVEL(4)** 或更低版本的应用程序结构的队列管理器，或尚未配置为容许连接丢失的 **CFLEVEL(5)** 结构，当与该结构的连接丢失时，异常终止，原因码为 00C510AB 。

当与已配置为容许丢失连接的应用程序结构的连接丢失时，与该结构的连接断开的所有队列管理器都将断开连接。 队列管理器的后续行为取决于连接丢失是部分还是全部。

部分丢失与应用程序结构的连接

如果确定失去了部分连接，那么失去了与结构的连接的队列管理器会尝试启动系统管理的重建，以便将结构移至另一个具有改进的连接的耦合设施。 如果此重建成功，那么会将结构中的持久消息和非持久消息复制到其他耦合设施，并复原对结构上的队列的访问。 未断开连接的队列管理器仍连接到该结构，但是，在系统管理的重建过程中，访问该结构的操作会延迟。

如果应用程序结构无法重建到具有改进的连接的另一个耦合设施，或者在另一个耦合设施中重建该结构后，某些队列管理器仍然没有与该结构的连接，那么在恢复到该耦合设施的连接之前，该结构上定义的队列在没有与该结构的连接的队列管理器上仍然不可用。 当队列管理器变为可用时，它会自动重新连接到该结构，并且会恢复对该结构上定义的共享队列的访问权。

与应用程序结构的连接完全中断

如果综合系统中的所有 MVS 系统都与分配了应用程序结构的耦合设施失去了连接，那么只要尝试重新连接到该结构， z/OS 就会从该耦合设施取消分配该结构。 队列管理器可能由于多种原因而尝试重新连接到结构，例如应用程序尝试打开共享队列，或者系统通知新的耦合设施资源可能已变为可用。 因此，在完全失去与应用程序结构的连接之后，可能会丢失受影响结构中的所有非持久消息。

如果使用 **RECAUTO(YES)**定义了可恢复的应用程序结构，那么将在完全失去连接后自动恢复这些结构。 如果备用耦合设施可用于在其中分配结构，或者无论何时此类耦合设施变为可用，那么将立即开始恢复。 如果尚未使用 **RECAUTO(YES)**定义结构，那么可以通过发出 **RECOVER CFSTRUCT** 命令来启动恢复。 这将恢复结构中的所有持久消息，但所有非持久消息都将丢失。 由于此过程涉及读取队列管理器日志，因此可能需要一些时间才能完成，因此建议定期执行结构备份以缩短时间，直到恢复对结构上的共享队列的访问。

一旦应用程序尝试打开在该结构上定义的共享队列或从系统接收到新耦合设施资源变为可用的通知，队列管理器就会尝试重新连接到不可恢复的应用程序结构。 如果有合适的耦合设施可用于分配结构，那么将分配新结构并恢复对该结构上定义的共享队列的访问权。 由于无法将持久消息放入不可恢复结构中定义的队列，因此共享队列上的所有消息都将丢失。

操作行为

如果配置为容许与特定耦合设施结构的连接丢失的 IBM WebSphere MQ 7.1 或更高版本队列管理器失去连接，那么队列共享组的成员将尝试从故障中自动恢复并重新连接到该结构。 此活动可能涉及重新分配另一个耦合设施中的结构，如果一个耦合设施可用，那么该耦合设施具有更好的连接性。 但是，仍可能需要操作员干预以从失去连接的情况中恢复。

通常，必需的操作程序操作是:

1. 解决导致连接丢失的故障原因。
2. 确保可以分配 IBM MQ 结构的耦合设施在综合系统中的所有系统上都可用

在丢失连接事件后，已在另一个耦合设施中自动重新分配的任何结构都可以移动到耦合设施，并具有与队列共享组中所有队列管理器的最佳连接。 如果需要，可以通过启动系统管理的重建命令 **SETXCF START,REBUILD** 来完成此操作，如 z/OS 移动虚拟服务器系统命令参考中所述。

如果部分失去与应用程序结构的连接，那么失去与该结构的连接的队列管理器将尝试启动系统管理的重建。 仅当另一个耦合设施具有与当前连接到该结构的所有活动队列管理器的连接时，此进程才会在该耦合设施中分配该结构。 因此，当队列共享组中的大多数队列管理器已失去与应用程序结构的连接时，由于仍连接到原始结构的队列管理器，它们可能无法将该结构重建到另一个耦合设施中。 在这种情况下，可以关闭仍连接到原始结构的队列管理器以允许重建结构，也可以发出 **RESET CFSTRUCT ACTION(FAIL)** 命令以使结构失败。 可以通过发出 **RECOVER CFSTRUCT** 命令在适用的结构上启动恢复。

注: 发生故障并恢复结构时，该结构上的所有非持久消息都将丢失。

z/OS IBM MQ for z/OS 中的安全概念

使用本主题来了解安全性对于 IBM MQ 的重要性，以及在系统上没有足够的安全性设置所产生的影响。

为什么必须保护 IBM MQ 资源

IBM MQ 处理可能有价值的信息传输。 应用安全性可确保 IBM MQ 拥有和管理的资源不受未经授权的访问。 这种接触可能导致信息的丢失或泄露。

您应该确保下列任何资源都不会被任何未经授权的用户或进程访问或更改:

- 与 IBM MQ 的连接
- IBM MQ 对象，例如队列，进程和名称列表
- IBM MQ 传输链路，即 IBM MQ 通道
- IBM MQ 系统控制命令 (system control commands)
- IBM MQ 消息

- 与消息关联的上下文信息

为了提供必要的安全性， IBM MQ 使用 z/OS 系统授权工具 (SAF) 将授权请求路由到外部安全性管理器 (ESM) ，例如 Security Server (以前称为 RACF)。 IBM MQ 不会对其自身进行安全性验证。 在使用分布式排队或客户机的情况下，您可能需要其他安全措施， IBM MQ 为这些安全措施提供通道认证记录，通道出口，MCAUSER 通道属性和 TLS。

允许访问对象的决定由 ESM 作出， IBM MQ 遵循该决定。 如果 ESM 无法做出决策，那么 IBM MQ 会阻止对该对象的访问。

如果不保护 IBM MQ 资源会发生什么情况

如果对安全性不执行任何操作，那么最可能的影响是 所有 用户都可以访问和更改 每个 资源。 这不仅包括本地用户，还包括使用分布式排队或客户机的远程系统上的用户，其中登录安全性控制的严格程度可能低于 z/OS 的通常情况。

要启用安全性检查，必须执行以下操作:

- 安装并激活 ESM (例如， Security Server)。
- 如果您使用的 ESM 不是安全服务器，请定义 MQADMIN 类。
- 激活 MQADMIN 类。

您必须考虑使用混合大小写资源名称是否对您的企业有利。 如果在 ESM 概要文件中使用混合大小写资源名称，那么必须定义并激活 MXADMIN 类。

z/OS 数据集加密

"数据集加密" (DSE) 提供对 z/OS 数据集进行加密的功能，以便这些数据集包含的数据只能由授予特定许可权的用户标识查看或修改。 这将对文件系统中的静态数据进行加密，并防止无意中向具有合法业务需求和管理数据集本身的许可权的用户披露敏感信息。

在 IBM MQ for z/OS 9.1.4 之前， IBM MQ for z/OS 不支持将 DSE 与活动日志，页集和共享消息数据集 (SMDS) 配合使用，这些数据集为 IBM MQ 消息提供主持久性机制。

相反， Advanced Message Security 为 IBM MQ 消息传递提供了端到端加密解决方案，该解决方案包含整个 IBM MQ 网络，动态数据加密，静态数据加密，甚至运行时 IBM MQ 进程中的数据加密。

可以使用 DSE 对 IBM MQ 子系统中使用的其他 VSAM 和连续数据集进行加密。 例如:

- 引导数据集 (BSDS)
- 保存在启动时使用 CSQINPx DDNAMEs 读取的系统配置 (MQSC) 命令的顺序文件
- IBM MQ 归档日志，通常用于长期归档 IBM MQ 日志数据以进行审计。

您可以通过分配使用数据集密钥标签定义的数据类来使用 DSE 进行加密。 有关更多信息，请参阅 规划日志归档存储器。

从 IBM MQ for z/OS 9.1.4 开始，除了先前发行版中提供的支持外， IBM MQ for z/OS 还支持将 DSE 与活动日志和页集配合使用。

IBM MQ for z/OS 不支持将 DSE 用于共享消息数据集 (SMDS)。

请参阅 使用数据集加密在 IBM MQ for z/OS 上静态数据的机密性 部分。 for more information.

相关概念
安全概念
通道认证记录
在 z/OS 上使用 IBM MQ 对象的权限
加密安全性协议：TLS
相关任务
在 z/OS 上设置安全性
比较链接级别安全性和应用程序级别安全性

相关参考
IBM MQ for z/OS 的消息

z/OS IBM MQ for z/OS 中的安全性控制和选项

您可以指定是否对整个 IBM MQ 子系统开启安全性，以及是否要在队列管理器或队列共享组级别执行安全性检查。 您还可以控制为 API-资源安全性检查的用户标识数。

子系统安全性

子系统安全性是一个控件，用于指定是否对整个队列管理器执行任何安全性检查。 如果不需要安全性检查 (例如，在测试系统上)，或者如果您对可以连接到 IBM MQ 的所有资源 (包括客户机和通道) 的安全性级别感到满意，那么可以对队列管理器或队列共享组关闭安全性检查，以便不再进行安全性检查。

这是唯一可以完全关闭安全性并确定是否执行任何其他安全性检查的检查。 即，如果对队列管理器或队列共享组关闭检查，那么不会执行其他 IBM MQ 检查; 如果保持打开状态，那么 IBM MQ 将检查其他 IBM MQ 资源的安全性需求。

您还可以打开或关闭特定资源集 (例如命令) 的安全性。

队列管理器或队列共享组级别检查

您可以在队列管理器级别或队列共享组级别实现安全性。 如果在队列共享组级别实现安全性，那么该组中的所有队列管理器都共享相同的概要文件。 这意味着要定义和维护的概要文件更少，使安全管理更容易。 它还使您能够轻松地将新的队列管理器添加到队列共享组，因为它将继承现有安全概要文件。

如果您的安装需要这两者的组合 (例如，在迁移期间)，或者如果队列共享组中有一个队列管理器需要与该组中的其他队列管理器具有不同级别的安全性，那么也可以实现这两者的组合。

队列共享组级别安全性

将对整个队列共享组执行队列共享组级别安全性检查。 它使您能够简化安全管理，因为它要求您定义较少的安全概要文件。 用户标识使用特定资源的授权在队列共享组级别进行处理，并且独立于用户标识用于访问该资源的队列管理器。

例如，假设服务器应用程序在用户标识 SERVER 下运行，并希望访问名为 SERVER.REQUEST，您要在综合系统中的每个 z/OS 映像上运行 SERVER 实例。 而不是允许 SERVER 打开 SERVER.REQUEST (队列管理器级别安全性)，只能允许在队列共享组级别进行访问。

您可以使用队列共享组级别安全概要文件来保护所有类型的资源，无论是本地资源还是共享资源。

队列管理器级别安全性 (queue manager level security)
您可以使用队列管理器级别的安全概要文件来保护所有类型的资源，无论是本地资源还是共享资源。

两个级别的组合
您可以同时使用队列管理器和队列共享组级别安全性。

您可以覆盖作为该组成员的特定队列管理器的队列共享组级别安全设置。 这意味着您可以对单个队列管理器执行与对组中其他队列管理器执行的安全性检查级别不同的级别。

有关更多信息，请参阅 用于控制队列共享组或队列管理器级别安全性的概要文件。

控制检查的用户标识数

RESLEVEL 是一个安全服务器概要文件，用于控制为 IBM MQ 资源安全性检查的用户标识数。 通常，当用户尝试访问 IBM MQ 资源时， Security Server 会检查相关用户标识以查看是否允许访问该资源。 通过定义 RESLEVEL 概要文件，您可以控制是否检查零个，一个或两个用户标识 (如果适用)。

这些控件是在连接的基础上完成的，并且是在连接的生命周期内完成的。

每个队列管理器只有一个 RESLEVEL 概要文件。 控制由用户标识对此概要文件具有的访问权实现。

混合大小写或大写 IBM MQ RACF 类

现在，您可以使用混合案例 RACF 概要文件支持，这允许您使用混合案例资源名称并定义 IBM MQ RACF 概要文件以保护它们。

您可以选择:

- 继续仅使用先前发行版中的大写 IBM MQ RACF 类，或者
- 使用新的混合大小写 IBM MQ RACF 类。

如果不使用混合大小写 RACF 概要文件，那么您仍可以在 IBM MQ for z/OS 中使用混合大小写资源名称; 但是，这些资源名称只能由大写 IBM MQ 类中的通用 RACF 概要文件保护。 使用混合大小写 IBM MQ RACF 概要文件支持时，您可以通过在混合大小写 IBM MQ 类中定义 IBM MQ RACF 概要文件来提供更精细的保护级别。

z/OS 可以在 IBM MQ for z/OS 中保护的资源

当队列管理器启动时，或者当操作员命令指示时， IBM MQ for z/OS 确定要保护哪些资源。

您可以控制对每个单独的队列管理器执行哪些安全性检查。 例如，您可以在生产队列管理器上实现多项安全性检查，但在测试队列管理器上不实现任何安全性检查。

连接安全性

当应用程序尝试连接到队列管理器时，将执行连接安全性检查。 它通过发出 MQCONN 或 MQCONNX 请求来完成，或者在通道启动程序或 CICS 或 IMS 适配器发出连接请求时完成。

如果您正在使用队列管理器级别安全性，那么可以对特定队列管理器关闭连接安全性检查。 但是，如果执行此操作，那么任何用户都可以连接到该队列管理器。

对于 CICS 适配器，仅 CICS 地址空间用户标识用于连接安全性检查，而不是单个 CICS 终端用户标识。 对于 IMS 适配器，当 IMS 控件或从属区域连接到 IBM MQ 时，将检查 IMS 地址空间用户标识。 对于通道启动程序，将检查通道启动程序地址空间所使用的用户标识。

您可以在队列管理器或队列共享组级别开启或关闭连接安全性检查。

命令安全性

当用户从 可在 IBM MQ for z/OS 上发出 MQSC 和 PCF 命令的源中描述的任何源发出 MQSC 命令时，将执行命令安全性检查。 您可以对命令指定的资源进行单独检查，如 第 222 页的『命令资源安全性』中所述。

如果关闭命令检查，那么不会检查命令的签发者以查看他们是否有权发出命令。

如果从控制台输入 MQSC 命令，那么控制台必须具有 z/OS SYS 控制台权限属性。 从 CSQINP1 或 CSQINP2 数据集发出或由队列管理器在内部发出的命令将免除所有安全性检查，而针对 CSQINPX 的命令将使用通道启动程序地址空间的用户标识。 您必须控制允许谁通过正常数据集保护来更新这些数据集。

您可以在队列管理器或队列共享组级别打开或关闭命令安全性检查。

命令资源安全性

某些 MQSC 命令 (例如，定义本地队列) 涉及 IBM MQ 资源的操作。 当命令资源安全性处于活动状态时，每次发出涉及资源的命令时， IBM MQ 都会检查是否允许用户更改该资源的定义。

您可以使用命令资源安全性来帮助实施命名标准。 例如，可能允许工资单管理员仅删除和定义名称以 "PAYROLL" 开头的队列。 如果命令资源安全性处于不活动状态，那么不会对该命令正在处理的资源进行安全性检查。 不要将命令资源安全性与命令安全性混淆; 两者是独立的。

关闭命令资源安全性检查不会影响专门针对不涉及命令的其他处理类型执行的资源检查。

您可以在队列管理器或队列共享组级别开启或关闭命令资源安全性检查。

通道安全性注意事项

通道安全性

当您使用通道时，可用的安全功能取决于您要使用的通信协议。 如果使用 TCP，那么虽然可以使用 TLS，但通信协议未提供任何安全功能。 如果您正在使用 APPC，那么可以将用户标识信息从发送 MCA 通过网络传递到目标 MCA 以进行验证。

对于这两种协议，您可以指定要出于安全目的检查的用户标识以及数量。 同样，可供您使用的选项取决于您正在使用的协议，您在定义通道时指定的内容以及通道启动程序的 RESLEVEL 设置。

有关可用通道安全性类型的更多信息，请参阅 通道认证记录 和 安全性出口概述

相关参考

第 223 页的『IBM MQ for z/OS 中的 API-资源安全性』
当应用程序使用 MQOPEN 或 MQPUT1 调用打开对象时，将检查资源。 打开对象所需的访问权取决于打开队列时指定的打开选项。

z/OS IBM MQ for z/OS 中的 API-资源安全性

当应用程序使用 MQOPEN 或 MQPUT1 调用打开对象时，将检查资源。 打开对象所需的访问权取决于打开队列时指定的打开选项。

API-资源安全性细分为以下检查:

- 队列
- 流程
- 名称列表
- 备用用户
- 上下文

打开队列管理器对象或访问存储类对象时，不会执行安全性检查。

队列

队列安全性检查控制允许谁打开哪个队列，以及允许他们使用哪些选项来打开该队列。 例如，可能允许用户打开名为 PAYROLL.INCREASE.SALARY 用于浏览队列上的消息 (使用 MQOO_BROWSE 选项)，但不从队列中除去消息 (使用 MQOO_INPUT_ * 选项之一)。 如果关闭队列检查，那么任何用户都可以使用任何有效打开选项 (即， MQOPEN 或 MQPUT1 调用上的任何有效 MQOO_ * 选项) 打开任何队列。

可以在队列管理器或队列共享组级别打开或关闭队列安全性检查。

进程

当用户打开进程定义对象时，将执行进程安全性检查。 如果关闭进程检查，那么任何用户都可以打开任何进程。

您可以在队列管理器或队列共享组级别开启或关闭进程安全性检查。

名称列表

当用户打开名称列表时，将执行名称列表安全性检查。 如果关闭对名称列表的检查，那么任何用户都可以打开任何名称列表。

您可以在队列管理器或队列共享组级别打开或关闭名称列表安全性检查。

备用用户

备用用户安全性控制一个用户标识是否可以使用另一个用户标识的权限来打开 IBM MQ 对象。

例如：

- 在用户标识 PAYSERV 下运行的服务器程序从用户标识 USER1 放入队列的队列中检索请求消息。
- 当服务器程序获取请求消息时，它将处理请求并将应答重新放入与请求消息一起指定的应答队列中。
- 服务器可以指定其他用户标识 (在本例中为 USER1) ，而不是使用自己的用户标识 (PAYSERV) 来授权打开应答队列。 在此示例中，备用用户安全性将控制是否允许用户标识 PAYSERV 在打开应答队列时将用户标识 USER1 指定为备用用户标识。

在对象描述符 (MQOD) 的 *AlternateUserId* 字段中指定备用用户标识。

您可以在任何 IBM MQ 对象 (例如，流程或名称列表) 上使用备用用户标识。 它不会影响任何其他资源管理器 (例如，用于 CICS 安全性或用于 z/OS 数据集安全性) 所使用的用户标识。

如果备用用户安全性未处于活动状态，那么任何用户都可以使用任何其他用户标识作为备用用户标识。

您可以在队列管理器或队列共享组级别开启或关闭备用用户安全性检查。

Context

上下文是适用于特定消息的信息，包含在作为消息一部分的消息描述符 (MQMD) 中。 上下文信息有两个部分：

身份部分
第一次将消息放入队列的应用程序的用户。 它由以下字段组成：

- *UserIdentifier*
- *AccountingToken*
- *ApplIdentityData*

"源" 部分
将消息放入当前存储该消息的队列的应用程序。 它由以下字段组成：

- *PutApplType*
- *PutApplName*
- *PutDate*
- *PutTime*
- *ApplOriginData*

在进行 MQPUT 或 MQPUT1 调用时，应用程序可以指定上下文数据。 应用程序可能会生成数据，可能会从另一条消息传递数据，或者缺省情况下队列管理器可能会生成数据。 例如，服务器程序可以使用上下文数据来检查请求者的身份，即，此消息是否来自正确的应用程序? 通常， *UserIdentifier* 字段用于确定备用用户的用户标识。

您可以使用上下文安全性来控制用户是否可以在任何 MQOPEN 或 MQPUT 调用上指定任何上下文选项。 有关上下文选项的信息，请参阅 与消息上下文相关的 MQOPEN 选项。 有关与上下文相关的消息描述符字段的描述，请参阅 MQMD-消息 descriptorMQMD -消息描述符。

如果关闭上下文安全性检查，那么任何用户都可以使用队列安全性允许的任何上下文选项。

您可以在队列，队列管理器或队列共享组级别打开或关闭上下文安全性检查。

z/OS z/OS 上的可用性

IBM MQ for z/OS 具有许多用于实现高可用性的功能。 本主题描述了可用性的一些注意事项。

如果队列管理器或通道启动程序发生故障，那么 IBM MQ 的一些功能可以提高系统可用性。 有关这些功能的更多信息，请参阅以下部分：

- 综合系统注意事项
- 共享队列
- 共享通道
- IBM MQ 网络可用性

- 使用 z/OS 自动重新启动管理器 (ARM)
- 使用 z/OS Extended Recovery Facility (XRF)
- 使用 z/OS GROUPUR 属性在队列共享组中进行恢复
- 在何处查找有关可用性的更多信息

综合系统注意事项

在 综合系统中，许多 z/OS 操作系统映像在单个系统映像中进行协作，并使用耦合设施进行通信。 IBM MQ 可以使用综合系统环境的设施来增强可用性。

除去队列管理器与特定 z/OS 映像之间的亲缘关系允许在发生映像故障时在另一 z/OS 映像上重新启动队列管理器。 重新启动机制可以是手动，使用 ARM 或使用系统自动化，如果您确保执行以下操作:

- 所有页集，日志，引导数据集，代码库和队列管理器配置数据集都在共享卷上定义。
- 子系统定义具有综合系统作用域和综合系统中的唯一名称。
- 在 IPL 时，安装在每个 z/OS 映像上的 早期代码 的级别相同。
- TCP 虚拟 IP 地址 (VIPA) 在综合系统中的每个 TCP 堆栈上都可用，并且您已将 IBM MQ TCP 侦听器和入站连接配置为使用 VIPA 而不是缺省主机名。

有关在综合系统中使用 TCP 的更多信息，请参阅 IBM Redbooks 出版物 综合系统中的 *TCP/IP* SG24-5235。

您还可以配置在综合系统中的不同操作系统映像上运行的多个队列管理器，以作为队列共享组运行，这可以利用共享队列和共享通道来实现更高的可用性和工作负载均衡。

共享队列

在队列共享组环境中，应用程序可以连接到队列共享组中的任何队列管理器。 由于队列共享组中的所有队列管理器都可以访问同一组共享队列，因此应用程序不依赖于特定队列管理器的可用性; 队列共享组中的任何队列管理器都可以为任何队列提供服务。 如果队列管理器由于队列共享组中的所有其他队列管理器都可以继续处理队列而停止，那么这将提供更大的可用性。 有关共享队列的高可用性的信息，请参阅 第 152 页的『使用共享队列的优点』。

为了进一步增强队列共享组中消息的可用性， IBM MQ 会检测组中的另一个队列管理器是否与耦合设施异常断开连接，并在可能的情况下完成仍处于暂挂状态的该队列管理器的工作单元。 这称为 对等恢复 ，在 第 215 页的『对等恢复』中进行了描述。

对等恢复无法恢复在发生故障时处于不确定状态的工作单元。 您可以使用自动重新启动管理器 (ARM) 来重新启动故障中涉及的所有系统 (例如 CICS， Db2 和 IBM MQ) ，并确保它们都在同一新处理器上重新启动。 这意味着它们可以再同步，并快速恢复不确定的工作单元。 在第 226 页的『使用 z/OS Automatic Restart Manager (ARM)』中对此进行了描述。

共享通道

在队列共享组环境中， IBM MQ 提供了向网络提供高可用性的功能。 通道启动程序使您能够使用网络产品来平衡一组合格服务器之间的网络请求，并隐藏网络中的服务器故障 (例如， VTAM 通用资源)。 IBM MQ 将通用端口用于入站请求，以便可以将连接请求路由到队列共享组中的任何可用通道启动程序。 在第 169 页的『共享通道』中对此进行了描述。

共享出站通道采用它们从共享传输队列发送的消息。 有关共享通道状态的信息保存在整个队列共享组级别的一个位置。 这意味着，如果通道启动程序，队列管理器或通信子系统发生故障，那么可以在队列共享组中的其他通道启动程序上自动重新启动通道。 这称为 对等通道恢复 ，在 共享出站通道中进行了描述。

IBM MQ 网络可用性

使用通道将 IBM MQ 消息从队列管理器传输到 IBM MQ 网络中的队列管理器。 您可以在多个级别更改配置，以提高队列管理器的网络可用性以及 IBM MQ 通道检测网络问题和重新连接的能力。

TCP *Keepalive* 可用于 TCP/IP 通道。 它使 TCP 在会话之间定期发送包以检测网络故障。 KAINT 通道属性确定通道的这些包的频率。

AdoptMCA 允许终止由于网络中断而在接收处理中阻塞的通道，并将其替换为新的连接请求。 您可以使用带有 MQSC 实用程序的 ADOPTMCA 队列管理器属性或带有可编程命令格式接口的 AdoptNewMCAType 属性来控制 AdoptMCA 。

ReceiveTimeout 可防止通道在网络接收调用中被永久阻塞。 RCVTIME 和 RCVTMIN 通道启动程序参数确定通道的接收超时特征 (作为其脉动信号间隔的函数)。 有关更多详细信息，请参阅 队列管理器参数 。

使用 z/OS Automatic Restart Manager (ARM)

您可以将 IBM MQ for z/OS 与 z/OS 自动重新启动管理器 (ARM) 结合使用。 如果队列管理器或通道启动程序失败，那么 ARM 将在同一 z/OS 映像上重新启动该队列管理器或通道启动程序。 如果 z/OS 失败，那么整个相关子系统和应用程序组也会失败。 ARM 可以按预定义顺序在综合系统中的另一个 z/OS 映像上自动重新启动所有发生故障的系统。 这称为跨系统重新启动。

ARM 支持在共享队列环境中快速恢复不确定事务。 如果您未使用队列共享组，那么它还会提供更高的可用性。

在发生 z/OS 故障时，可以使用 ARM 在综合系统中的其他 z/OS 映像上重新启动队列管理器。

要启用自动重新启动，必须执行以下操作:

1. 设置 ARM 耦合数据集。
2. 定义您希望 z/OS 在 *ARM* 策略中执行的自动重新启动操作。
3. 启动 ARM 策略。

如果要自动重新启动不同 z/OS 映像中的队列管理器，那么必须使用综合系统范围的唯一 4 字符子系统名称来定义可能重新启动该队列管理器的每个 z/OS 映像中的每个队列管理器。

在 IBM MQ 网络中使用 ARM 中描述了将 ARM 与 IBM MQ 配合使用。

使用 z/OS Extended Recovery Facility (XRF)

您可以在扩展恢复设施 (XRF) 环境中使用 IBM MQ 。 所有 IBM MQ 拥有的数据集 (可执行代码， BSD ，日志和页集) 都必须在活动和备用 XRF 处理器之间共享的 DASD 上。

如果使用 XRF 进行恢复，那么必须在活动处理器上停止队列管理器并在备用处理器上启动该队列管理器。 对于 CICS，可以使用 CICS 提供的命令列表 (CLT) 来执行此操作，或者系统操作员可以手动执行此操作。 对于 IMS，这是手动操作，您必须在协调 IMS 系统完成处理器切换后执行此操作。

必须先完成或终止 IBM MQ 实用程序，然后才能将队列管理器切换到备用处理器。 在规划 XRF 恢复计划时，请仔细考虑此潜在中断的影响。

在活动处理器上的队列管理器终止之前，请注意防止队列管理器在备用处理器上启动。 过早启动可能会导致数据，目录和日志中出现严重的完整性问题。 使用全局资源序列化 (GRS) 可防止在两个系统上同时使用 IBM MQ ，从而帮助避免完整性问题。 必须包含 BSDS 作为受保护资源，并且必须在 GRS 环中包含活动和备用 XRF 处理器。

在队列共享组中使用 z/OS GROUPUR 属性进行恢复

队列共享组 (QSG) 允许使用本主题中描述的其他事务工具。 GROUPUR 属性允许 XA 客户机应用程序对 QSG 的任何成员执行任何可能需要的不确定事务恢复。

如果 XA 客户机应用程序通过综合系统连接到队列共享组 (QSG)，那么无法保证它连接到哪个特定队列管理器。 队列共享组中的队列管理器使用 GROUPUR 属性可以启用可能需要在 QSG 的任何成员上进行的任何不确定事务恢复。 即使应用程序最初连接到的队列管理器不可用，也可以进行事务恢复。

此功能部件将 XA 客户机应用程序从对 QSG 的特定成员的任何依赖关系中释放出来，从而扩展队列管理器的可用性。 队列共享组作为单个实体向事务应用程序显示，该实体提供所有 IBM MQ 功能部件，并且没有单个队列管理器故障点。

此功能对于事务应用程序并不明显。

在何处查找有关可用性的更多信息

您可以从以下来源找到有关这些主题的更多信息:

表 *24:* 在何处查找有关可用性的更多信息

Topic	**在哪里查找信息**
队列共享组	第 137 页的『共享队列和队列共享组』
系统参数	配置系统参数
使用自动重新启动管理器 实用程序	在 IBM MQ 网络中使用 ARM
MQSC 命令	MQSC 命令

z/OS IBM MQ for z/OS 上的监视和统计信息

IBM MQ for z/OS 具有一组用于监视队列管理器和收集统计信息的工具。

IBM MQ 提供了用于监视系统和收集统计信息的工具。 有关这些设施的更多信息，请参阅以下部分:

- 第 227 页的『联机监视』
- 第 227 页的『IBM MQ 跟踪』
- 第 228 页的『事件』

联机监视

IBM MQ 包含用于监视 IBM MQ 对象状态的以下命令:

- DISPLAY CHSTATUS 显示指定通道的状态。
- DISPLAY QSTATUS 显示指定队列的状态。
- DISPLAY CONN 显示指定连接的状态。

有关这些命令的更多信息，请参阅 MQSC 命令。

IBM MQ 跟踪

IBM MQ 提供了可用于在队列管理器运行时收集以下信息的跟踪工具:

性能统计信息
统计信息跟踪收集以下信息以帮助您监视性能并调整系统:

- 不同 MQI 请求的计数 (消息管理器统计信息)
- 不同对象请求的计数 (数据管理器统计信息)

- 有关 Db2 使用情况的信息 (Db2 管理器统计信息)
- 有关耦合设施使用情况的信息 (耦合设施管理器统计信息)
- 有关 SMDS 使用情况的信息 (共享消息数据集统计信息)
- 有关缓冲池使用情况的信息 (缓冲区管理器统计信息)
- 有关日志记录的信息 (日志管理器统计信息)
- 有关存储器使用情况的信息 (存储管理器统计信息)
- 有关锁定请求的信息 (锁定管理器统计信息)

记帐数据

- 记帐跟踪收集有关处理 MQI 调用所耗用的处理器时间以及特定用户发出的 MQPUT 和 MQGET 请求数的信息。
- IBM MQ 还可以使用 IBM MQ 收集有关每个任务的信息。 此数据作为线程级记帐记录收集。 对于每个线程， IBM MQ 还会收集有关该线程所使用的每个队列的信息。

跟踪生成的数据将发送到系统管理设施 (SMF) 或通用跟踪设施 (GTF)。

事件

IBM MQ 事件提供有关队列管理器中的错误，警告和其他重要事件的信息。 通过将这些事件合并到您自己的系统管理应用程序中，可以针对多个 IBM MQ 应用程序监视多个队列管理器中的活动。 尤其是，您可以从单个队列管理器监视系统中的所有队列管理器。

可以通过用户编写的报告机制向支持向操作员呈现事件的管理应用程序报告事件。 事件还使应用程序能够充当其他管理网络 (例如 NetView) 的代理程序，以监视报告并创建相应的警报。

相关任务

使用 IBM MQ 跟踪

使用 IBM MQ 事件

z/OS z/OS 上的恢复单元处置

某些事务应用程序在连接到队列共享组 (QSG) 中的队列管理器时，可以使用 GROUP (而不是 QMGR) 恢复处置单元，方法是在它们连接时指定 QSG 名称而不是队列管理器名称。 这将通过除去重新连接到 QSG 中的同一队列管理器的需求，使事务恢复更加灵活和稳健。

已使用队列共享组名连接的应用程序启动的事务也具有 GROUP 恢复处置单元。

当事务应用程序与 GROUP 恢复单元处置连接时，它在逻辑上连接到队列共享组，并且与任何特定队列管理器都没有亲缘关系。 在连接到 QSG 中的任何队列管理器时，可以查询并解决它已启动的已完成落实进程的 phase-1 阶段的任何 2 阶段落实事务 (即，它们处于不确定状态)。 在恢复方案中，这意味着事务协调程序不必重新连接到同一队列管理器，而此时该队列管理器可能不可用。

与 QMGR 恢复单元处置连接的应用程序与它们所连接的队列管理器具有直接亲缘关系。 在恢复方案中，事务协调程序必须重新连接到同一队列管理器以解决任何不确定事务，而不考虑队列管理器是否属于队列共享组。

当应用程序指定队列共享组名并因此连接到具有 GROUP 恢复处置单元的 QSG 中的队列管理器时，队列共享组在逻辑上是单独的资源管理器。 这意味着仅当应用程序与同一恢复处置单元重新连接时，不确定事务才对其可视。 具有 QMGR 恢复处置单元的不确定事务对于已与 GROUP 恢复处置单元连接的应用程序不可见，反之亦然。

相关概念

第 229 页的『启用 GROUP 恢复单元』
队列共享组可以配置并启用对 GROUP 恢复单元的支持。

第 229 页的『应用程序支持』
使用此页面来确定哪些应用程序可以与 GROUP 恢复处置单元连接。

z/OS 启用 GROUP 恢复单元

队列共享组可以配置并启用对 GROUP 恢复单元的支持。

要在 QSG 中的队列管理器上使用 GROUP 恢复单元，请启用 GROUPUR 队列管理器属性。 有关此概念的更多信息，请先参阅 第 228 页的『z/OS 上的恢复单元处置』，然后再阅读本主题的其余部分。

启用 GROUPUR 队列管理器属性后，队列管理器接受具有 GROUP 恢复处置单元的新连接。 如果禁用此属性，那么将不接受具有此处置的新连接，尽管已连接的应用程序在断开连接之前不受影响。

当应用程序与 GROUP 恢复处置单元连接，并询问哪些事务存在疑问或尝试解决在队列共享组 (QSG) 中其他位置启动的事务时，它现在所连接的队列管理器必须能够与队列共享组的其他成员通信，以便它可以处理请求。 为此，它使用名为 SYSTEM.QSG.UR.RESOLUTION.QUEUE。 此队列必须位于称为 CSQSYSAPPL 的可恢复应用程序结构上。 该结构必须可恢复，因为在处理解析请求时，持久消息存储在此队列中。

必须先确保已定义耦合设施结构和共享队列，然后才能启用 GROUP 恢复单元。 您可以使用 CSQ4INSS 样本中的定义。 在定义队列或在启动期间检测到队列时，队列共享组中的每个队列管理器都会打开该队列，以便它可以接收入局请求。 如果要删除或移动队列，因为未正确定义该队列，那么可以通过更新队列对象以禁止 MQGET 请求来请求队列管理器关闭其打开的句柄。 当您进行了必要的更正时，允许应用程序再次从队列中获取消息将指示每个队列管理器重新打开该队列。 使用 DISPLAY QSTATUS 命令可识别在队列上打开的句柄。

完成此设置后，您可以在希望事务应用程序能够通过 GROUP 恢复单元处置连接到的每个队列管理器上启用 GROUP 恢复单元。 这不需要是队列共享组中的所有队列管理器，但如果选择仅在队列共享组的子集上启用此功能，那么必须确保应用程序仅尝试连接到已启用此功能的队列管理器。 有关更多信息，请参阅第 229 页的『应用程序支持』。

当您尝试启用 GROUPUR 队列管理器属性时，将执行许多配置检查。 队列管理器将检查:

- 它属于队列共享组。
- 已根据 CSQ4INSS 中的定义定义了名为 SYSTEM.QSG.UR.RESOLUTION.QUEUE 的共享队列。
- SYSTEM.QSG.UR.RESOLUTION.QUEUE 位于名为 CSQSYSAPPL 的可恢复 CF 结构上。

如果上述任何检查失败，那么 GROUPUR 属性将保持禁用状态，并返回消息代码。

如果启用了队列管理器属性，那么还会在队列管理器启动时执行这些配置检查。 如果在启动 GROUP 恢复单元期间有任何检查失败，并且队列管理器发出一条消息，指示哪些检查失败。 执行必要的更正操作后，必须重新启用队列管理器属性。

z/OS 应用程序支持

使用此页面来确定哪些应用程序可以与 GROUP 恢复处置单元连接。

对 GROUP 恢复单元处置的支持仅限于某些类型的事务应用程序，对于这些应用程序， IBM MQ for z/OS 是资源管理器而不是事务协调程序。 当前支持的事务应用程序包括:

- IBM MQ 扩展事务客户机应用程序
- 在应用程序服务器中运行的 IBM MQ classes for JMS 应用程序，例如 WebSphere Application Server。
- 在 CICS Transaction Server 4.2 或更高版本中运行的 CICS 应用程序 (当使用 RESYNCMEMBER (GROUpresYNC) 配置了 CICS MQCONN 资源定义时)。

相关概念

第 229 页的『IBM MQ 扩展事务客户机应用程序』
使用此页面来确定 IBM MQ 扩展事务客户机应用程序如何使用 GROUP 恢复处置单元。

第 230 页的『CICS 应用程序』
使用此页面来确定 CICS 如何使用 GROUP 恢复处置单元。

z/OS *IBM MQ* 扩展事务客户机应用程序

使用此页面来确定 IBM MQ 扩展事务客户机应用程序如何使用 GROUP 恢复处置单元。

IBM MQ 扩展事务客户机应用程序的一个示例是使用 JMS 并在 WebSphere Application Server 中运行的应用程序，通过 TCP/IP 连接到 IBM MQ，而不是本地绑定。 这些客户机应用程序通过网络连接 (例如，通过

TCP/IP) 连接到 IBM MQ for z/OS 。 对于这些应用程序，它是为 xa_open 调用中传递的 xa_info 字符串的 QMNAME 参数指定的值，用于指定是使用 QMGR 还是 GROUP 恢复处置单元。 有关 xa_open 的更多信息，请参阅 xa_open 字符串的格式 和 xa_open 的其他错误处理。 对于 JMS 应用程序，通过在 ConnectionFactory 中指定队列共享组 (QSG) 的名称来完成此操作，而不是指定特定队列管理器的名称。

要使 XA 客户机应用程序能够利用 GROUP 恢复单元处置，必须配置 TCP/IP 设置，以允许客户机应用程序路由至已启用 GROUPUR 属性的队列共享组中的队列管理器，而不是特定队列管理器。 可用于执行此操作的动态虚拟 IP 地址技术之一是 z/OS SysPlex 分发器。 请参阅 z/OS 通讯服务器和 z/OS 基础技能：动态虚拟寻址 以了解更多详细信息。 如果要在队列共享组中的队列管理器子集上启用 GROUP 恢复单元，请确保无法将客户机应用程序路由到未启用它的那些应用程序。

您的客户机应用程序不必使用共享通道连接到队列共享组。

z/OS *CICS* 应用程序

使用此页面来确定 CICS 如何使用 GROUP 恢复处置单元。

CICS 4.2 和更高版本在 MQCONN 资源定义中提供组再同步选项 RESYNCMEMBER (GROUpresYNC)。 使用此选项配置的 CICS 可以连接到与该 CICS 区域在同一 LPAR 上运行的队列共享组中的任何合适的队列管理器。 要支持 CICS GROUpresYNC 选项，队列管理器必须在 MQ V7.1 或更高版本上运行，并且必须启用 GROUPUR 支持。

使用 GROUpresYNC 在连接到 MQ 的 CICS 区域中运行的事务将创建具有 GROUP 恢复处置单元的工作单元。

您可以使用 RESYNCMEMBER (GROUpresYNC) 在队列管理器发生故障后启用更快的恢复，因为它使 CICS 区域能够立即连接到在同一 LPAR 上运行的备用合格队列管理器，从而根据需要解决任何不确定事务，而无需等待队列管理器重新启动。

RESYNCMEMBER (GROUPRESYNC) 还会为 CICS 启用更灵活的重新启动选项。 具有配置为使用 GROUpresYNC 和 MQ 共享队列的 MQ 连接的 CICS 区域可以在作为同一队列共享组的成员运行队列管理器的任何 LPAR 上重新启动。

z/OS IBM MQ 和其他 z/OS 产品

使用本主题来了解 IBM MQ 如何使用其他 z/OS 产品。

相关概念

第 230 页的『IBM MQ 和 CICS』
IBM MQ 9.0.0 和更高版本支持的所有 CICS 版本都使用 CICS 提供的适配器和网桥版本。

第 236 页的『IBM MQ for z/OS 和 WebSphere Application Server』
使用本主题来了解 WebSphere Application Server 对 IBM MQ for z/OS 的使用。

相关参考

第 231 页的『IBM MQ 和 IMS』
使用本主题来了解 IBM MQ 如何使用 IMS。 IMS 适配器允许您将队列管理器连接到 IMS，并使 IMS 应用程序能够使用 MQI。

第 235 页的『IBM MQ 和 z/OS 批处理， TSO 和 RRS 适配器』
使用本主题来了解 IBM MQ 如何使用 z/OS Batch ， TSO 和 RRS 适配器。

z/OS IBM MQ 和 CICS

IBM MQ 9.0.0 和更高版本支持的所有 CICS 版本都使用 CICS 提供的适配器和网桥版本。

有关配置 IBM MQ CICS 适配器和 IBM MQ CICS bridge 组件的更多信息，请参阅 CICS 文档的 配置与 IBM MQ 的连接 部分。

相关任务

将 IBM MQ 与 CICS 结合使用

z/OS CICS 组连接

CICS 组连接使 CICS 区域能够连接到同一 LPAR 上 IBM MQ 队列共享组的任何活动成员，而不是指定单个队列管理器。 CICS 仍一次连接到单个队列管理器。

LPAR 上至少需要两个队列管理器来支持 CICS 组连接。 使用组连接可提供更高的可用性，因为您不需要特定队列管理器处于活动状态。 CICS 连接到 LPAR 上的队列共享组中的任何队列管理器。

有关更多信息，请参阅有关 MQCONN 资源的 CICS 文档。

CICS 尝试连接到传递的 MQNAME ，就像它是队列管理器一样:

- 如果队列管理器存在并且处于活动状态，那么连接将起作用。
- 如果连接失败， CICS 将查询组中队列管理器的状态，以确定哪些队列管理器在同一 LPAR 上处于活动状态。
- 如果多个队列管理器处于活动状态，那么 CICS 会检查 RESYNCMEMBER (YES) 和 UOW 状态，以确定 CICS 是需要连接到特定成员，还是应该连接到特定成员，或者如果未处于活动状态，请等待。
- 如果不需要连接到特定成员， CICS 将选择队列管理器 (使用随机算法)。
- CICS 尝试连接到所选队列管理器。
- 如果尝试失败，那么根据返回码， CICS 将选择下一个成员，然后再次执行选择循环。
- 如果没有任何队列管理器处于活动状态，那么 CICS 会向队列管理器列表发出多个连接，并在 ECBLIST 上等待，直到第一个队列管理器变为可用。

相关概念

第 231 页的『CICS 的组恢复单元 (GROUPUR)』

IBM MQ GROUPUR for CICS 为队列共享组 (QSG) 中的不确定工作单元提供对等恢复。 一个 IBM MQ 队列管理器可以代表队列共享组中的另一个队列管理器来解析不确定的工作单元。 这意味着如果 CICS 通过组连接重新连接到 QSG 中的另一个队列管理器，那么它可以解决来自先前 IBM MQ 连接的不确定事务。

相关信息

支持 IBM MQ 队列共享组

z/OS CICS 的组恢复单元 (GROUPUR)

IBM MQ GROUPUR for CICS 为队列共享组 (QSG) 中的不确定工作单元提供对等恢复。 一个 IBM MQ 队列管理器可以代表队列共享组中的另一个队列管理器来解析不确定的工作单元。 这意味着如果 CICS 通过组连接重新连接到 QSG 中的另一个队列管理器，那么它可以解决来自先前 IBM MQ 连接的不确定事务。

如果 CICS 区域正在使用队列管理器，并且队列管理器异常结束，那么将恢复任何不确定事务。 这样就不需要 CICS 区域等待它正在使用的队列管理器重新启动，然后解决任何不确定的工作单元。 这意味着在 LPAR 上至少需要两个队列管理器，以便在第一个队列管理器异常终止时， CICS 可以连接到另一个队列管理器。

CICS MQCONN 定义上的新 RESYNCMEMBER (GROUpresYNC) 设置:

- 使用 IBM MQ 组连接功能和对等恢复。
- 需要启用了 GROUPUR 属性的队列管理器。
- 仍然支持现有 CICS MQCONN RESYNCMEMBER 设置 (YES 和 NO):
  - 使用现有 CICS 组连接功能，而不进行对等恢复。
  - 更改 RESYNCMEMBER 设置将在 CICS 下次连接到 IBM MQ 时生效。

相关概念

第 229 页的『启用 GROUP 恢复单元』

队列共享组可以配置并启用对 GROUP 恢复单元的支持。

z/OS IBM MQ 和 IMS

使用本主题来了解 IBM MQ 如何使用 IMS。 IMS 适配器允许您将队列管理器连接到 IMS，并使 IMS 应用程序能够使用 MQI。

可选的其他 IBM MQ - IMS 网桥使应用程序能够运行不使用 MQI 的 IMS 应用程序。 这意味着您可以将旧应用程序与 IBM MQ 配合使用，而无需对其进行重写。

有关这些组件的更多信息，请参阅以下子主题:

相关概念

IBM MQ for z/OS 上的 IMS 和 IMS 网桥应用程序

相关任务

设置 IMS 适配器

设置 IMS 网桥

操作 IMS 适配器

相关参考

MQIIH - IMS 信息头

z/OS IMS 适配器

IMS 适配器是 IMS 应用程序与 IBM MQ 子系统之间的接口。

IBM MQ 适配器支持不同的应用程序环境通过消息排队网络发送和接收消息。 IMS 适配器是 IMS 应用程序与 IBM MQ 子系统之间的接口。 这使 IMS 应用程序可以使用 MQI。

IMS 适配器使用 IMS 提供的 外部子系统连接设施 (ESAF) 接收并解释对 IBM MQ 的访问请求。 通常， IMS 会自动连接到 IBM MQ ，而无需操作员干预。

IMS 适配器为以以下方式或状态运行的程序提供对 IBM MQ 资源的访问权:

- 任务 (TCB) 方式
- 问题状态
- 非交叉内存模式
- 非访问注册方式

适配器提供从应用程序任务控制块 (TCB) 到 IBM MQ 的连接线程。

The adapter supports a two-phase commit protocol for changes made to resources owned by IBM MQ with IMS acting as the syncpoint coordinator. IMS 不是同步点协调程序的对话 (例如 APPC 保护的 (SYNCLVL = SYNCPT) 对话) 不受 IMS 适配器支持。

适配器还提供触发器监视器事务 (CSQQTRMN)。 在第 233 页的『IMS 触发器监视器』中对此进行了描述。

您可以将 IBM MQ 与 IMS 扩展恢复设施 (XRF) 配合使用，以帮助从 IMS 错误进行恢复。

注: 从 IMS 15.2 起，不再支持扩展恢复设施 (XRF)。 请参阅 IMS 文档以获取更多信息。

使用适配器

应用程序和 IMS 适配器在同一地址空间中运行。 队列管理器在其自己的地址空间中是独立的。

必须链接编辑每个向适当的 IMS 语言接口模块发出一个或多个 MQI 调用的程序，并且除非它使用动态 MQI 调用，否则必须链接编辑 IBM MQ 提供的 API 存根程序 CSQQSTUB。 当应用程序发出 MQI 调用时，存根通过 IMS 外部子系统接口将控制权移交给适配器，该接口管理消息队列管理器对请求的处理。

使用 IMS 进行系统管理和操作

授权的 IMS 终端操作员可以发出 IMS 命令来控制和监视与 IBM MQ 的连接。 但是， IMS 终端操作员无法控制 IBM MQ 地址空间。 例如，操作员无法从 IMS 地址空间关闭 IBM MQ 。

限制

以下 IBM MQ API 调用在使用 IMS 适配器的应用程序中不受支持:

- MQCB
- MQCB_FUNCTION
- MQCTL

IMS 触发器监视器

IMS 触发器监视器 (**CSQQTRMN**) 是 IBM MQ 提供的 IMS 应用程序，用于在发生 IBM MQ 事件时 (例如，将消息放入特定队列时) 启动 IMS 事务。

运作方式

将消息放入应用程序消息队列时，如果满足触发器条件，那么将生成触发器。 然后，队列管理器将消息 (包含一些用户定义的数据) (称为 触发器消息) 写入为该消息队列指定的启动队列。 在 IMS 环境中，您可以启动 CSQQTRMN 实例以监视启动队列，并在触发消息到达时从中检索触发器消息。 通常，CSQQTRMN 通过 INSERT (ISRT) 将另一个 IMS 事务调度到 IMS 消息队列。 启动的 IMS 应用程序从应用程序消息队列中读取消息，然后对其进行处理。 CSQQTRMN 必须作为非消息 BMP 运行。

CSQQTRMN 的每个副本都服务一个启动队列。 启动后，触发器监视器将运行至 IBM MQ 或 IMS 结束。

CSQQTRMN 的 APPLCTN 宏必须指定 SCHDTYP=PARALLEL。

由于触发器监视器是面向批处理的 BMP ，因此触发器监视器启动的 IMS 事务包含以下内容:

- IOPCB 的 LTERM 字段中的空白
- IOPCB 的 "用户标识" 字段中触发器监视器 BMP 的 PSB 名称

如果目标 IMS 事务受安全服务器 (先前称为 RACF) 保护，那么您可能需要将 CSQQTRMN 定义为安全服务器的用户标识。

z/OS IBM MQ - IMS 网桥

IBM MQ - IMS 网桥是 IBM MQ for z/OS 的组件，允许从 IBM MQ 应用程序直接访问 IMS 系统上的应用程序。

IBM MQ - IMS 网桥启用 隐式 *MQI* 支持。 这意味着您可以重新设计由 3270 连接的终端控制的旧应用程序，使其由 IBM MQ 消息控制，而不必重写，重新编译或重新链接这些应用程序。 网桥是 IMS *Open Transaction Manager Access* (OTMA) 客户机。

在网桥应用程序中， IMS 应用程序中没有 IBM MQ 调用。 应用程序使用 GET UNIQUE (GU) 将其输入发送到 IOPCB ，并使用 ISRT 将其输出发送到 IOPCB。 IBM MQ 应用程序在消息数据中使用 IMS 头 (MQIIH 结构) 来确保应用程序可以像在由非可编程终端驱动时那样执行。 如果您正在使用处理多段消息的 IMS 应用程序，请注意，所有段都应包含在一条 IBM MQ 消息中。

第 234 页的图 78 中说明了 IMS 网桥。

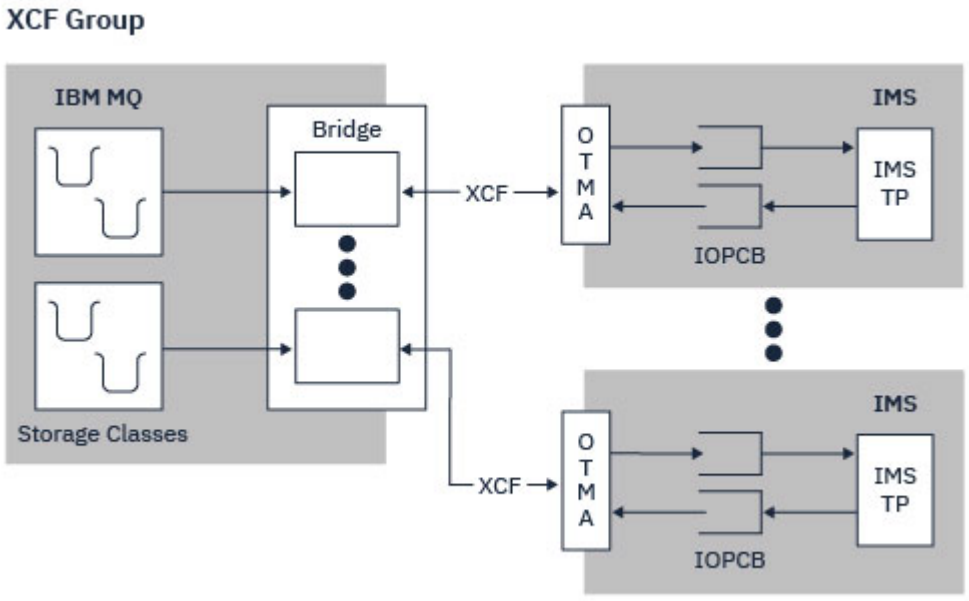


图 *78: IBM MQ - IMS* 网桥

队列管理器可以连接到一个或多个 IMS 系统，多个队列管理器可以连接到一个 IMS 系统。 唯一的限制是它们必须全部属于同一 XCF 组，并且必须全部位于同一综合系统中。

请参阅 设置 IMS 网桥，以获取有关设置 IMS 网桥以及将其他 IMS 连接添加到同一队列管理器的信息。

什么是 OTMA?

IMS OTMA 工具是在 IMS 上运行的基于事务的无连接客户机/服务器协议。 它充当通过 z/OS 跨系统耦合设施 (XCF)访问 IMS TM 应用程序的基于主机的通信服务器的接口。

OTMA 使客户机能够连接到 IMS，以便为客户机与 IMS 之间的交互提供高性能，用于大型网络或大量会话。 OTMA 在 z/OS 综合系统环境中实现。 因此， OTMA 的域仅限于 XCF 的域。

OTMA 资源监视

对 IMS v10 或更高版本中提供的 x "3C" OTMA 协议消息的支持包含在 IBM MQ for z/OS 中的 IBM MQ - IMS 网桥中。 这些消息由 IMS 发送到 OTMA 客户机以报告其运行状态。

如果 IMS 合作伙伴无法处理正在发送的事务请求量，那么它将通知 IBM MQ 已发生洪流警告。 作为响应，IBM MQ 将降低通过网桥发送请求的速率。

如果 IMS 仍无法处理事务请求，并且发生完全洪流情况，那么将暂挂所有到 IMS 合作伙伴的 TPIPE。 当 IMS 合作伙伴通知已解除洪水或洪水警告条件时， IBM MQ 将恢复所有暂挂的 TPIPE (如果适用)，并逐步提高发送事务请求的速率，直到达到最大速率。 控制台消息由 IBM MQ 发出，以响应 IMS 合作伙伴状态的更改。

如果正在使用 IMS v10 合作伙伴，那么应确保已应用 PTF UK45082 。

从 IBM MQ 提交 IMS 个事务

要提交使用网桥的 IMS 事务，应用程序会像往常一样将消息放入 IBM MQ 队列中。 这些消息包含 IMS 事务数据; 它们可以具有 IMS 头 (MQIIH 结构) 或允许 IBM MQ - IMS 网桥对消息中的数据进行假设。

IBM MQ 然后将消息放入 IMS 队列 (它首先在 IBM MQ 中排队，以允许使用同步点来确保数据完整性)。 IBM MQ 队列的存储类确定队列是否为 *OTMA* 队列 (即，用于将消息传输到 IBM MQ - IMS 网桥的队列) 以及将消息数据发送到的特定 IMS 伙伴。

远程队列管理器还可以通过写入 IBM MQ for z/OS 上的这些 OTMA 队列来启动 IMS 事务。

从 IMS 系统返回的数据将直接写入消息描述符结构 (MQMD) 中指定的 IBM MQ 应答队列。 (这可能是到 MQMD 的 **ReplyToQMgr** 字段中指定的队列管理器的传输队列。)

相关概念

IBM MQ for z/OS 上的 IMS 和 IMS 网桥应用程序

相关任务

定制 IMS 网桥

相关参考

第 231 页的『IBM MQ 和 IMS』

使用本主题来了解 IBM MQ 如何使用 IMS。 IMS 适配器允许您将队列管理器连接到 IMS，并使 IMS 应用程序能够使用 MQI。

z/OS IBM MQ 和 z/OS 批处理， TSO 和 RRS 适配器

使用本主题来了解 IBM MQ 如何使用 z/OS Batch， TSO 和 RRS 适配器。

批处理适配器简介

Batch/TSO 适配器是在 JES， TSO 或 z/OS UNIX System Services 下运行的 IBM MQ 和 z/OS 应用程序之间的接口。 这些适配器使 z/OS 应用程序能够使用 MQI。

适配器提供对以以下方式或状态运行的程序的 IBM MQ 资源的访问权:

- 任务 (TCB) 方式
- 问题或管理程序状态
- 非交叉内存模式
- 非访问注册方式

应用程序和 IBM MQ 之间的连接处于任务级别。 适配器提供从应用程序任务控制块 (TCB) 到 IBM MQ 的连接线程。

Batch/TSO 适配器支持对 IBM MQ 拥有的资源进行更改的单阶段落实协议。 它不支持多阶段落实协议。 RRS 适配器使 IBM MQ 应用程序能够与其他支持 RRS 的产品一起参与两阶段落实协议，由 z/OS 资源恢复服务 (RRS) 进行协调。

适配器使用 z/OS STIMERM 服务来调度每秒异步事件。 此事件运行不涉及批处理应用程序任务的任何等待的中断请求块 (IRB)。 此 IRB 检查是否已发布 IBM MQ 终止 ECB。 如果已发布终止 ECB，那么 IRB 将发布正在 IBM MQ 中等待事件 (例如，信号或等待) 的任何应用程序 ECB。

Batch/TSO 适配器

IBM MQ Batch/TSO 适配器为 z/OS 批处理和 TSO 应用程序提供 IBM MQ 支持。 在 z/OS Batch 或 TSO 下运行的所有应用程序都必须具有 API 存根程序 CSQBSTUB 链接-与它们一起编辑。 存根为应用程序提供对所有 MQI 调用的访问权。 通过发出 MQI 调用 **MQCMIT** 和 **MQBACK**，可以对应用程序使用单阶段落实和回退。

RRS 适配器

资源恢复服务 (RRS) 是 z/OS 的子组件，提供系统范围的服务，用于在 z/OS 产品之间协调两阶段落实。 IBM MQ Batch/TSO RRS 适配器 (RRS 适配器) 为要使用这些服务的 z/OS 批处理和 TSO 应用程序提供 IBM MQ 支持。 RRS 适配器使 IBM MQ 能够成为 RRS 协调的完整参与者。 应用程序可以与支持 RRS 的其他产品 (例如， Db2) 一起参与两阶段落实处理。

RRS 适配器提供了两个存根; 您必须链接编辑要将 RRS 与其中一个存根配合使用的应用程序。

CSQBRSTB
此存根允许您通过使用 RRS 可调用资源恢复服务 (而不是 MQI 调用 **MQCMIT** 和 **MQBACK**) 对应用程序使用两阶段落实和回退。

您还必须从库 SYS1.CSSLIB 与您的应用程序。 如果使用 MQI 调用 **MQCMIT** 和 **MQBACK**，那么将收到返回码 MQRC_ENVIRONMENT_ERROR。

CSQBRRSI
此存根允许您使用 MQI 调用 **MQCMIT** 和 **MQBACK** ; IBM MQ 实际将这些调用实现为 **SRRCMIT** 和 **SRRBACK** RRS 调用。

有关构建使用 RRS 适配器的应用程序的信息，请参阅 RRS 批处理适配器。

在何处查找有关 z/OS 批处理， TSO 和 RRS 适配器的更多信息

您可以在以下源中找到有关此部分中主题的更多信息:

表 *25:* 在何处查找有关将 *z/OS Batch* 与 *IBM MQ* 配合使用的更多信息

Topic	在哪里查找信息
设置批处理适配器	任务 19: 设置批处理， TSO 和 RRS 适配器
RRS 可调用资源恢复服务	*MVS Programming: Callable Services for High Level Languages*

z/OS IBM MQ for z/OS 和 WebSphere Application Server

使用本主题来了解 WebSphere Application Server 对 IBM MQ for z/OS 的使用。

在 WebSphere Application Server 下运行的以 Java 编写的应用程序可以使用 Java Message Service (JMS) 规范来执行消息传递。 此环境中的点到点消息传递可由 IBM MQ for z/OS 队列管理器提供。

使用 IBM MQ for z/OS 队列管理器来提供消息传递的好处是，连接 JMS 应用程序可以完全参与 IBM MQ 网络的功能。 例如，他们可以使用 IMS 网桥，或者与在其他平台上运行的队列管理器交换消息。

WebSphere Application Server 与队列管理器之间的连接

请参阅 同时使用 IBM MQ 和 WebSphere Application Server 以获取更多信息。

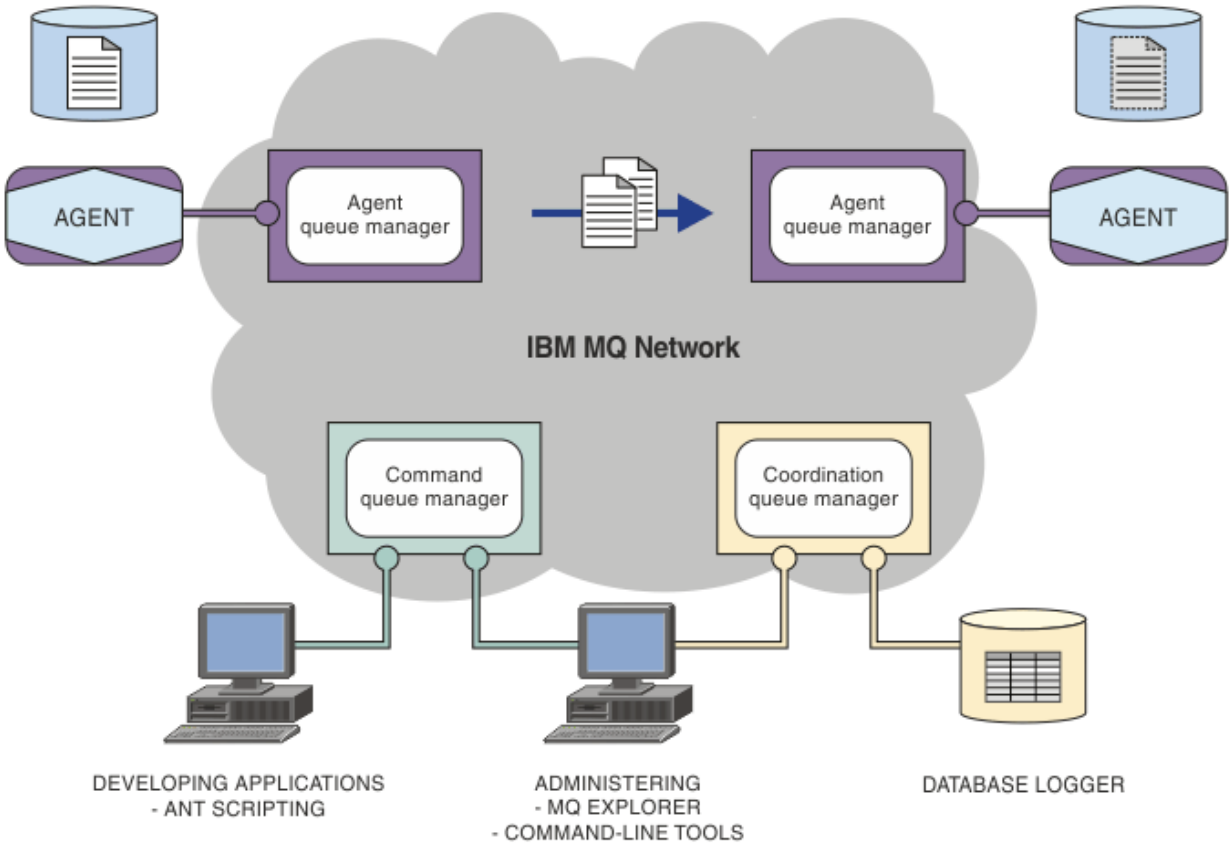
使用 JMS 应用程序中的 IBM MQ 函数

缺省情况下，保留在 IBM MQ 队列上的 JMS 消息使用 MQRFH2 头来保存某些 JMS 消息头信息。 很多旧 IBM MQ 应用程序不能处理带有这些头的消息，它们需要使用自己的特性头，例如 MQCIH 用于 CICS 桥接应用程序，或者 MQWIH 用于 IBM MQ 工作流程应用程序。 有关这些特殊注意事项的更多详细信息，请参阅 将 JMS 消息映射到 IBM MQ 消息。

Managed File Transfer

Managed File Transfer 将以受管和可审计方式在系统之间传输文件，而不用考虑文件大小或使用的操作系统。

您可以使用 Managed File Transfer 构建定制、可扩展且自动化的解决方案，使您能够管理和信任文件传输并保证文件传输的安全。 Managed File Transfer 可消除昂贵的冗余、降低维护成本并最大限度地利用现有的 IT 投资。



该图显示了简单的 Managed File Transfer 拓扑。 其中包含两个代理，分别与它们在 IBM MQ 网络中的代理队列管理器相连。 通过 IBM MQ 网络，将文件从位于图中一侧的代理传输到位于图中另一侧的代理。 在 IBM MQ 网络中还包含一个协调队列管理器和一个命令队列管理器。 应用程序和工具连接到这些队列管理器，以在 IBM MQ 网络中配置，管理，操作和记录 Managed File Transfer 活动。

根据您的操作系统和总体设置，Managed File Transfer 可以作为四个不同的选项进行安装。 这些选项为 Managed File Transfer Agent、Managed File Transfer Logger、Managed File Transfer Service 或 Managed File Transfer Tools。 有关更多信息，请参阅 Managed File Transfer 产品选项。

可以使用 Managed File Transfer 来执行下列任务：

- 创建受管文件传输
  - Windows Linux 在 Linux 或 Windows 平台上从 IBM MQ Explorer 创建新的文件传输。
  - 通过所有受支持平台上的命令行创建新的文件传输。
  - 将文件传输功能集成到 Apache Ant 工具中。
  - 通过将消息放置在代理命令队列上，编写控制 Managed File Transfer 的应用程序。
  - 安排稍后进行文件传输。 还可以根据一系列文件系统事件（例如，正在创建新文件）触发调度的文件传输。
  - 持续监视资源（例如，目录），并在该资源的内容满足某个预定义条件时启动任务。 此任务可以是文件传输， Ant 脚本或 JCL 作业。
  - 与 IBM MQ 队列进行文件传输。
  - 在 FTP、FTPS 或 SFTP 服务器之间进行文件传输。
  - 将文件传输至 Connect:Direct 节点和从这些节点传输文件。
  - 传输文本和二进制文件。 将在源和目标系统的代码页和行结束约定之间自动转换文本文件。
  - 可以使用针对基于安全套接字层 (SSL) 连接的行业标准保证传输安全。

- 查看正在进行的传输并记录有关网络中所有传输的信息。
  - Windows Linux 从 Linux 或 Windows 平台上的 IBM MQ Explorer 查看正在进行的传输的状态。
  - Windows Linux 在 Linux 或 Windows 平台上使用 IBM MQ Explorer 检查已完成传输的状态。
  - 使用 Managed File Transfer 数据库记录器功能将日志消息保存到 Db2 或 Oracle 数据库。

Managed File Transfer 基于 IBM MQ 构建，WebSphere MQ 在应用程序之间提供确定的仅一次性消息传递。 您可以利用 IBM MQ 的各种功能。 例如，您可以使用通道压缩来压缩通过 IBM MQ 通道在代理之间发送的数据，或使用 SSL 通道保证代理之间所发送数据的安全。 文件可靠地传输，并且可以容忍通过其执行文件传输的基础结构的故障。 如果经历网络中断，文件传输将在连接恢复时从断开的位置重新启动。

通过将文件传输与现有的 IBM MQ 网络进行整合，可以避免耗用维护两个独立的基础结构所需的资源。 如果您还不是 IBM MQ 客户，请创建 IBM MQ 网络以支持 Managed File Transfer，从而为未来的 SOA 实施构建主干。 如果您已是 IBM MQ 客户，那么 Managed File Transfer 可以利用现有 IBM MQ 基础结构，包括 IBM MQ Internet Pass-Thru 和 IBM Integration Bus。

您可以利用 IBM MQ 高可用性解决方案来提高 Managed File Transfer 配置的弹性。 如果代理程序使用复制的数据队列管理器 (RDQM)，那么必须将其配置为使用浮动 IP 地址功能。 这意味着代理程序使用相同的 IP 地址与当前正在运行的三个 RDQM 实例中的任何一个进行通信，并在故障转移时自动重新连接 (请参阅 RDQM 高可用性 和 创建和删除浮动 IP 地址)。 如果使用多实例队列管理器解决方案，那么应用程序将使用不同的 IP 地址与每个实例进行通信，由故障转移时的客户机重新连接处理 (请参阅 多实例队列管理器 和 通道和客户机重新连接)。

Managed File Transfer 与许多其他 IBM 产品集成：

IBM Integration Bus

已由 Managed File Transfer 作为 IBM Integration Bus 流的一部分传输的进程文件。 有关更多信息，请参阅 从 IBM Integration Bus 使用 MFT。

IBM Sterling Connect:Direct

使用 Managed File Transfer Connect:Direct 网桥将文件传输到现有 Connect:Direct 网络或从现有网络传输文件。 有关更多信息，请参阅 Connect:Direct 网桥。

IBM Tivoli Composite Application Manager

IBM Tivoli Composite Application Manager 提供可用于监视发布到协调队列管理器的信息的代理。

相关概念

Managed File Transfer 产品选项

第 239 页的『MFT 拓扑概述』
有关 Managed File Transfer 代理如何连接 IBM MQ 网络中的协调队列管理器的概述。

第 238 页的『MFT 如何使用 IBM MQ? 』
Managed File Transfer 可以使用多种方式与 IBM MQ 进行交互。

MFT 如何使用 IBM MQ?

Managed File Transfer 可以使用多种方式与 IBM MQ 进行交互。

- Managed File Transfer 通过将每个文件分割为一条或多条消息并通过 IBM MQ 网络传输这些消息，从而在代理进程之间传输文件。
- 代理进程通过使用非持久消息来移动文件数据，以在最大程度上降低对 IBM MQ 日志的影响。 通过彼此通信，代理进程可控制好传输包含文件数据的消息流。 这可防止包含文件数据的消息在 IBM MQ 传输队列上聚集，并确保当有任何非持久消息未交付时会重新发送文件数据。
- Managed File Transfer 代理会使用多个 IBM MQ 队列。 有关更多信息，请参阅 MFT 系统队列和系统主题。
- 虽然三个队列中的部分队列严格限制为内部使用，但是代理可接受形式为特殊格式的命令消息的请求，这些消息发送至特定队列以供该代理从中读取。 命令行命令和 IBM MQ Explorer 插件均可将 IBM MQ 消息发

送至代理以指示代理执行所需的操作。 您可以使用此方式来编写用于与代理进行交互的 IBM MQ 应用程序。 有关更多信息，请参阅 通过将消息放入代理命令队列来控制 MFT。

- Managed File Transfer 代理会将有关其状态以及传输进度和结果的信息发送至已指定为协调队列管理器的 MQ 队列管理器。 此信息由协调队列管理器发布，可供希望监视传输进度或保留发生的传输记录的应用程序来预订。 命令行命令和 IBM MQ Explorer 插件均可使用已发布的信息。 您可编写使用此信息的 IBM MQ 应用程序。 有关将信息发布到的主题的更多信息，请参阅 SYSTEM.FTE 主题。
- Managed File Transfer 的关键组件将会利用 IBM MQ 队列管理器的功能来存储和转发消息。 这意味着如果您遇到停机，那么基础结构中未受影响的部分仍可继续传输文件。 这可扩展至协调队列管理器，在此情况下，存储转发和持久预订的组合允许协调队列管理器承受变为不可用，而不会丢失有关发生的文件传输的关键信息。

MFT 拓扑概述

有关 Managed File Transfer 代理如何连接 IBM MQ 网络中的协调队列管理器的概述。

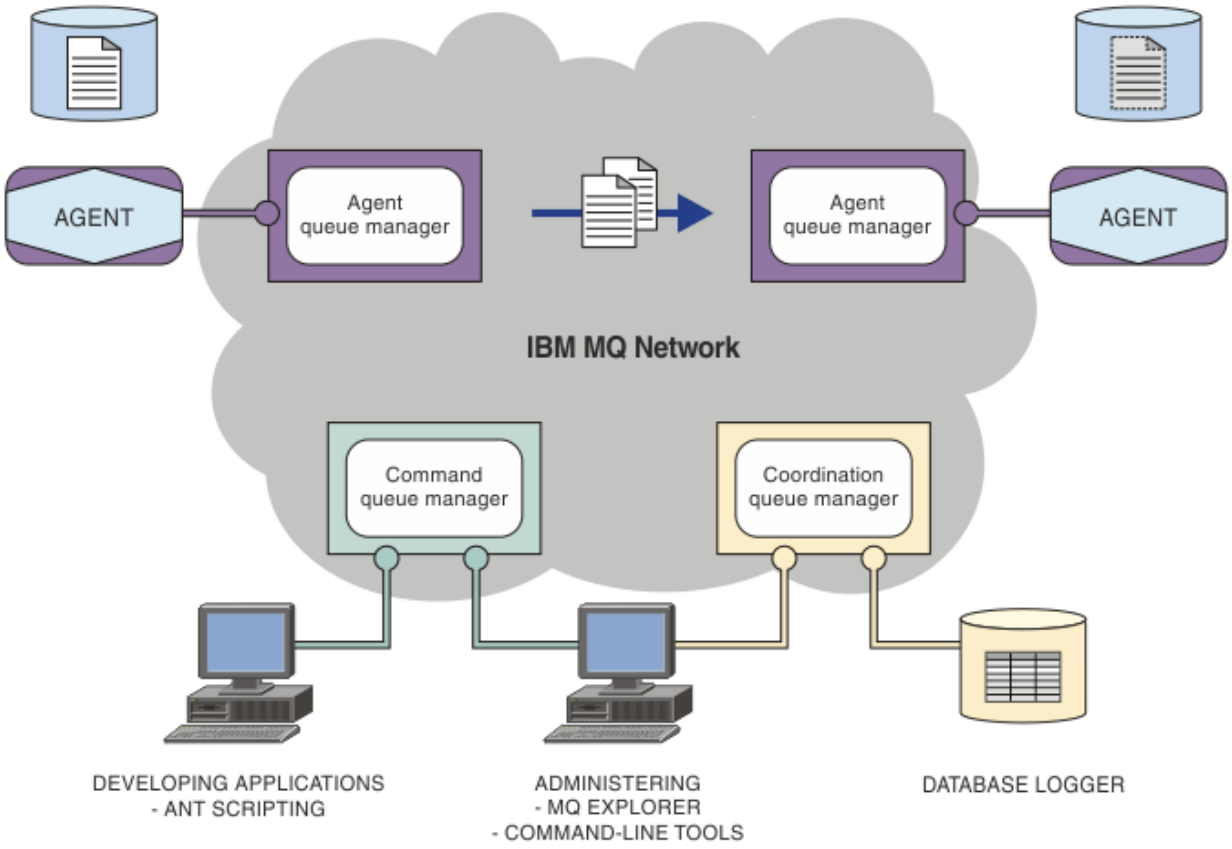
Managed File Transfer 代理发送并接收传输的文件。 每个代理在其关联队列管理器上都有自己的队列集，并且代理将以绑定或客户机方式连接到其队列管理器。 代理还可以使用协调队列管理器作为其队列管理器。

协调队列管理器广播审计和文件传输信息。 协调队列管理器表示代理集合、传输状态和传输审计信息的单个端点。 协调队列管理器并非执行传输所必需的。 如果协调队列管理器临时不可用，传输将照常继续。 审计和状态消息将会存储在代理队列管理器中，直至协调队列管理器可用，然后将照常处理。

代理注册协调队列管理器并将其详细信息发布到队列管理器。 此代理信息供 Managed File Transfer 插件用于支持从 IBM MQ Explorer 启动传输。 协调队列管理器上收集的代理信息还将供命令用来显示代理信息和代理状态。

传输状态和传输审计信息将在协调队列管理器上发布。 传输状态和传输审计信息供 Managed File Transfer 插件用于从 IBM MQ Explorer 监视传输的进度。 可以保留存储在协调队列管理器上的传输审计信息以提供可审计性。

命令队列管理器用于连接到 IBM MQ 网络，也是在您发出 Managed File Transfer 命令时要连接到的队列管理器。



相关概念

第 236 页的『Managed File Transfer』
Managed File Transfer 将以受管和可审计方式在系统之间传输文件，而不用考虑文件大小或使用的操作系统。

第 238 页的『MFT 如何使用 IBM MQ？』
Managed File Transfer 可以使用多种方式与 IBM MQ 进行交互。

Managed File Transfer 方案 (scenario)

MFT REST API 概述

REST API 支持某些 Managed File Transfer 命令，包括列出传输以及有关文件传输代理的详细信息。

从 IBM MQ 9.1.0 开始， REST API 包含用于列出所有当前 Managed File Transfer 传输和查询 Managed File Transfer 代理的状态的选项。 有关更多信息，请参阅 REST API MFT 入门。

IBM MQ Internet Pass-Thru

IBM MQ Internet Pass-Thru (MQIPT) 是 IBM MQ 的可选组件，可用于在跨因特网的远程站点之间实现消息传递解决方案。

从 IBM MQ 9.2.0 开始， MQIPT 是 IBM MQ 的可选组件。 要获取 IBM MQ 9.3.x 的 MQIPT 安装文件，请转至 IBM Fix Central for IBM MQ。 在 IBM MQ 9.2.0 之前， MQIPT 作为支持包提供。

您不必运行 IBM MQ 9.3 以在 IBM MQ 9.3 中使用 MQIPT 。 您可以使用 MQIPT 来连接任何受支持版本的 IBM MQ，并且不必安装与 MQIPT 版本相同的任何其他 IBM MQ 组件。

如果您已购买 IBM MQ 权利，那么可以安装所需数量的 MQIPT 副本。 MQIPT 安装不计入已购买的 IBM MQ 权利。 有关 IBM MQ 许可的更多信息，请参阅 IBM MQ 许可证信息。

注: 本文档与 IBM MQ 9.3 中的 MQIPT 相关。 有关 IBM Documentation 中的 MQIPT 支持包 (V 2.1) 文档，请参阅 IBM MQ 9.0 文档中的 MQIPT (SupportPac MS81) 。

注: 如果您正在使用 MQIPT 2.1 或更低版本，那么建议您升级到 MQIPT for IBM MQ 9.3，因为 MQIPT 支持包的支持结束日期为 2020 年 9 月 30th 。

IBM MQ Internet Pass-Thru 作为独立的服务运行，可以在两个 IBM MQ 队列管理器之间，或者在 IBM MQ 客户机和 IBM MQ 队列管理器之间接收和转发 IBM MQ 消息流。

当客户机和服务器没有位于同一物理网络时，MQIPT 启用此连接。

可以将 MQIPT 的一个或多个实例放在两个 IBM MQ 队列管理器之间的通信路径中，也可以放在 IBM MQ 客户机与 IBM MQ 队列管理器之间的通信路径中。 MQIPT 实例允许两个 IBM MQ 系统交换信息，两个系统之间不需要直接 TCP/IP 连接。 如果防火墙配置禁止两个系统之间的直接 TCP/IP 连接，那么这很有用。

MQIPT 在一个或多个 TCP/IP 端口侦听入站连接，这些入站连接可以传递正常 IBM MQ 消息、HTTP 内隧道传送的 IBM MQ 消息，或者使用传输层安全性 (TLS) 或安全套接字层 (SSL) 加密的消息。 MQIPT 可以处理多个并发连接。

发出初始 TCP/IP 连接请求的 IBM MQ 通道被称为调用者，其尝试连接到的通道被称为响应者，最终尝试联系的队列管理器称为目标队列管理器。

MQIPT 将数据从源转发至目标时将数据保留在内存中。 磁盘上不保存数据（除了操作系统分页到磁盘的内存）。 MQIPT 明确访问磁盘的唯一时间是读取配置文件并写入连接日志和跟踪记录。

可以通过 MQIPT 的一个或多个实例来生成完整范围的 IBM MQ 通道类型。 通信路径中存在 MQIPT 对所连接的 IBM MQ 组件的功能特征没有影响，但是对消息传输性能可能产生一些影响。

MQIPT 可与 IBM MQ 和 IBM Integration Bus 结合使用，如 第 244 页的『MQIPT 的可能配置』中所述。

要安装 MQIPT，请参阅 安装 MQIPT。

相关任务

配置 IBM MQ Internet Pass-Thru

管理和配置 IBM MQ Internet Pass-Thru

相关参考

IBM MQ Internet Pass-Thru 配置参考信息

MQIPT 的使用

IBM MQ Internet Pass-Thru (MQIPT) 有许多可能的用途。

MQIPT 可以用作通道集中器

通过这种方式使用 MQIPT，进出多个单独主机的通道似乎是进出防火墙，就好象所有通道都进出 MQIPT 主机。 这样可以简化防火墙过滤规则的定义和管理。

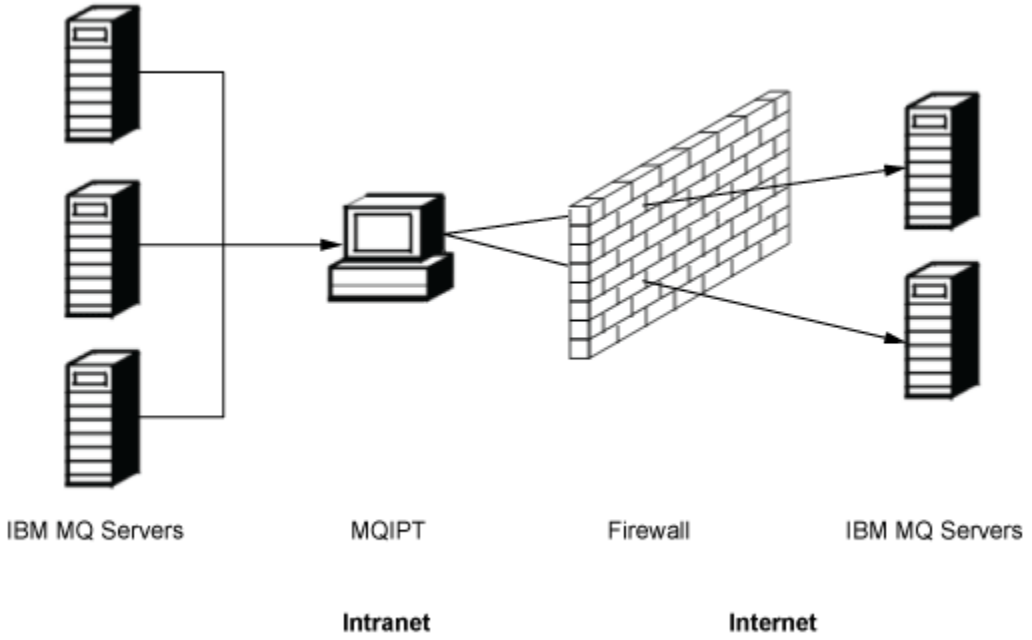


图 *79: MQIPT* 作为通道集中器的示例

可以将 MQIPT 放入 DMZ 以提供单点访问

DMZ 防火墙（有时称为“非军事区”防火墙）是一种用于保护局域网的防火墙配置。 它充当着不可信任的网络（通常是互联网）的暴露点。 如果将 MQIPT 置于 DMZ 防火墙中，并置于一个具有已知且可信的互联网协议（IP）地址的计算机上，则可以使用 MQIPT 侦听传入的 IBM MQ 通道连接，并将其转发至可信的内网；内网防火墙必须允许该可信计算机建立入站连接。 在此配置中，MQIPT 阻止外部访问请求接收受信内部网内计算机的真实 IP 地址。 MQIPT 以这种方式提供单点访问。 如果需要，可以将 MQIPT 配置为接受 TLS 连接，并使用单独的 TLS 连接将数据转发到目标，从而终止 DMZ 中的 TLS 会话。

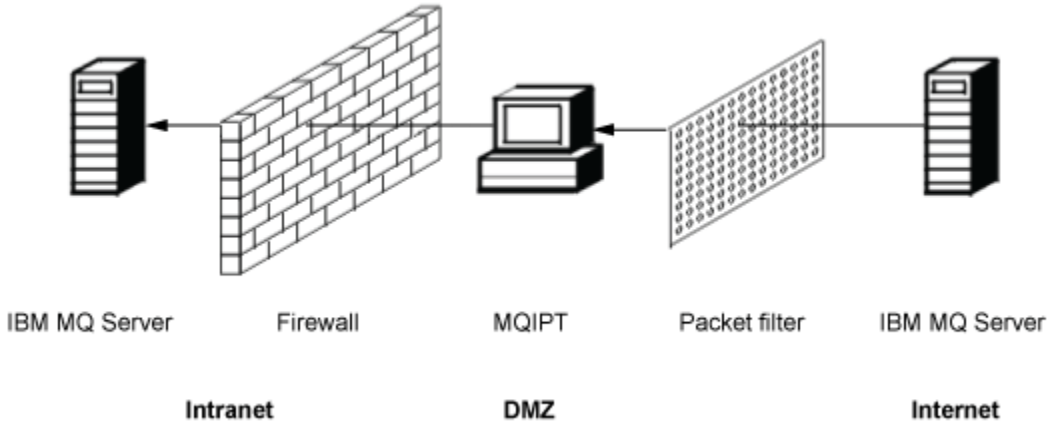


图 *80: DMZ* 防火墙中 *MQIPT* 示例

MQIPT 可以通过 HTTP 隧道传送通信

如果两个 MQIPT 实例部署一致，那么它们可以使用 HTTP 通信。 HTTP 隧道传送功能支持使用现有 HTTP 代理通过防火墙传递请求。 第一个 MQIPT 将 IBM MQ 协议插入 HTTP，第二个从 HTTP 包装器抽取 IBM MQ 协议，并将其转发到目标队列管理器。

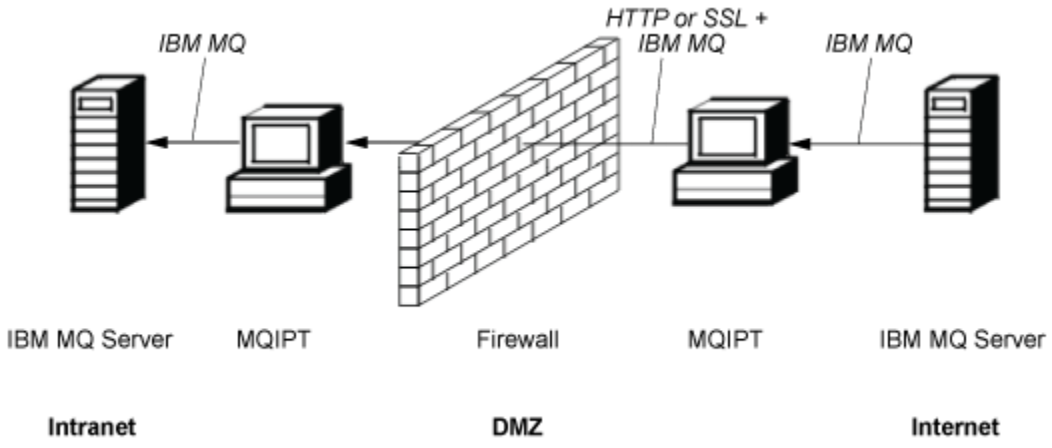


图 *81: MQIPT* 和 *HTTP* 隧道传送示例

MQIPT 可以加密消息

如果 MQIPT 如先前示例配置，通过防火墙传输请求之前可以加密请求。 第一个 MQIPT 对数据进行加密，第二个使用 SSL/TLS 对其进行解密，然后再将其发送到目标队列管理器。

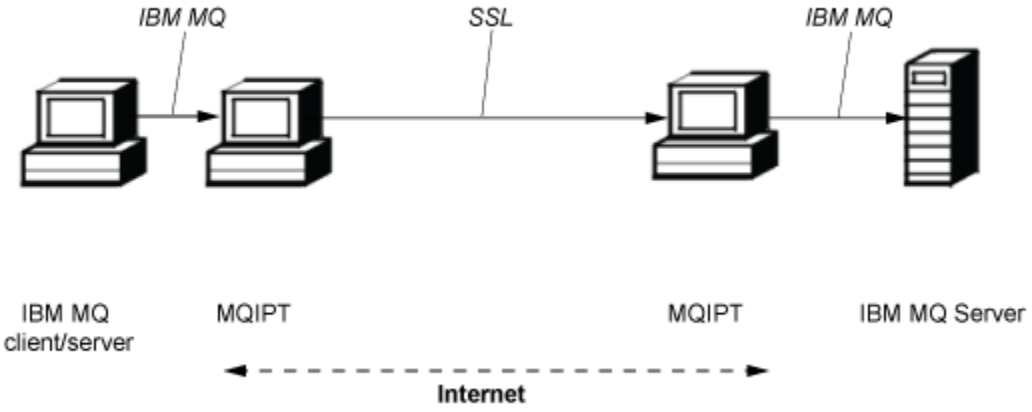


图 *82: MQIPT* 和 *SSL/TLS* 示例

MQIPT 的工作方式

在最简单的配置中，MQIPT 充当 IBM MQ 协议转发器。 它在 TCP/IP 端口上侦听并且接受来自 IBM MQ 通道的连接请求。

如果收到格式正确的请求，MQIPT 将在自身和目标 IBM MQ 队列管理器之间建立进一步的 TCP/IP 连接。 然后，将从入站连接接收到的所有协议包传递给目标队列管理器，并将目标队列管理器的协议包返回至原始的入站连接。

不涉及对 IBM MQ 协议（客户机/服务器或队列管理器到队列管理器）的更改，因为任何一端都不直接觉察到中介的存在。 不需要使用新版本的 IBM MQ 客户机或服务器代码。

要使用 MQIPT，必须将调用者通道配置为使用 MQIPT 主机名和端口，而不是目标队列管理器的主机名和端口。 这是使用 IBM MQ 通道的 **CONNAME** 属性定义的。 MQIPT 读取入站数据，并只将其传递给目标队列管理器。 其他配置字段（例如客户机/服务器通道的用户标识和密码）同样也传递到目标队列管理器。

多个队列管理器

可以使用 MQIPT 以允许访问多个目标队列管理器。 要实现此目的，必须存在一个机制来告知 MQIPT 连接到哪个队列管理器，因此， MQIPT 使用入站 TCP/IP 端口号来确定要连接到哪个队列管理器。

因此，您可以配置 MQIPT 在多个 TCP/IP 端口上侦听。 每一个侦听端口通过 MQIPT 路由映射到目标队列管理器。 您可以定义多达 100 个此类路由，从而将侦听的 TCP/IP 端口与目标队列管理器的主机名和端口关联。 这意味着目标队列管理器的主机名（IP 地址）从未对始发通道可视。 每个路由可以处理其侦听端口和目标之间的多个连接，每个连接独立运作。

MQIPT 配置文件

MQIPT 使用名为 `mqipt.conf` 的配置文件。 此文件包含所有路由及其关联属性的定义。 有关 `mqipt.conf` 的更多信息，请参阅 管理和配置 IBM MQ Internet Pass-Thru 。

启动 MQIPT 时，它将启动配置文件中列出的所有路由。 消息将写入系统控制台，显示每个路由的状态。 当对路由显示 MQCPI078 消息时，该路由准备好接受连接请求。

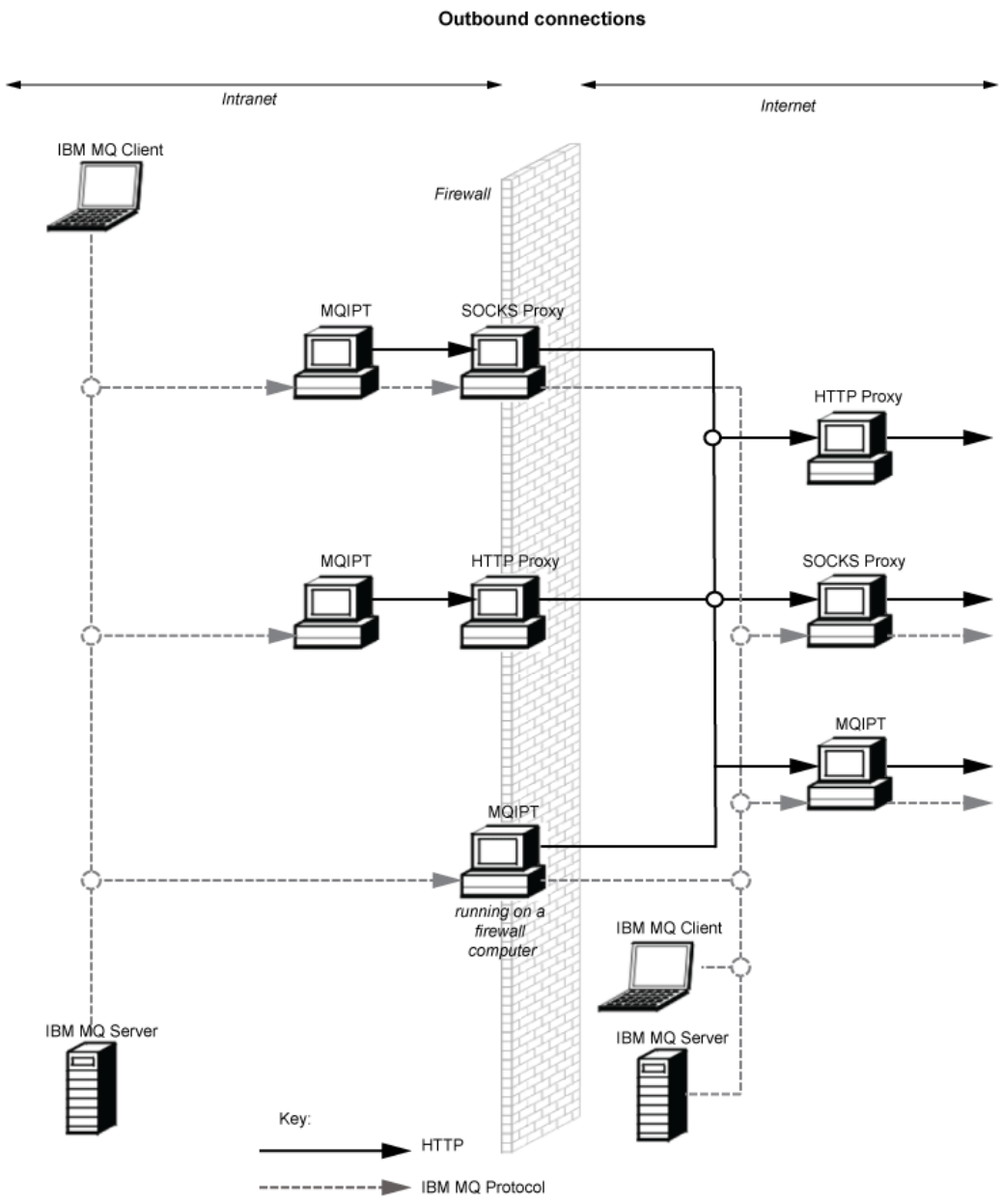
MQIPT 的可能配置

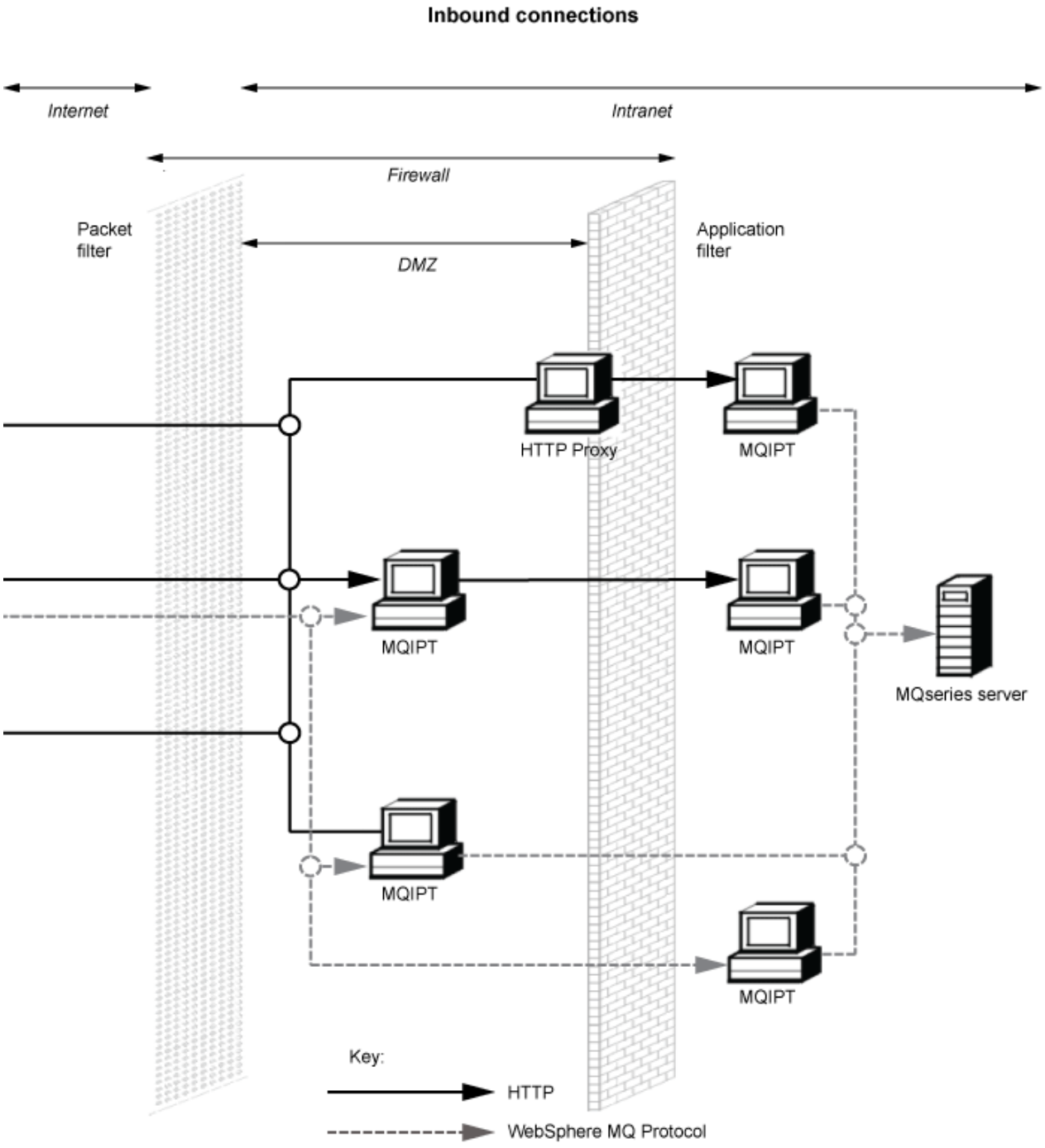
MQIPT 可与 IBM MQ 和 IBM Integration Bus 结合使用。

以下多部分图显示了 IBM MQ 拓扑中 MQIPT 的许多可能的配置。 该图演示了 MQIPT 发送消息的不同方式。 显示了内部网、防火墙内和防火墙外的互联网上的客户机和服务器，显示了向转发这些消息的 MQIPT、HTTP 代理或 SOCKS 代理传递消息。

消息通过入站防火墙传递到服务器之前，先由 DMZ 中的 MQIPT 代理或 HTTP 代理接收。

请注意，防火墙的内部网一侧上的 HTTP 代理、SOCKS 代理和 MQIPT 计算机代表互联网上多个计算机可能链接在一起。 例如， MQIPT 计算机在到达目标前可以通过一个或多个 SOCKS 或 HTTP 代理计算机，或更多 MQIPT 计算机进行通信。


Outbound connections
Intranet
Internet
IBM MQ Client
Firewall
MQIPT
SOCKS Proxy
HTTP Proxy
MQIPT
HTTP Proxy
SOCKS Proxy
MQIPT
MQIPT
running on a firewall computer
IBM MQ Client
IBM MQ Server
IBM MQ Server
Key:
HTTP
IBM MQ Protocol



兼容的配置

IBM MQ 客户机或队列管理器与 MQIPT 通信的兼容连接方案。 使用同一个或第二个 MQIPT 路由与目标队列管理器通信。

使用单个 MQIPT 路由的兼容配置

您可以使用单个 MQIPT 路由与 IBM MQ 通信。

第 247 页的表 26 中的列包含以下信息：

1. IBM MQ 和 MQIPT 路由之间使用的协议。 IBM MQ 客户机或队列管理器可以创建连接，该连接可以使用 IBM MQ 格式和协议 (FAP) 或 SSL/TLS 协议。
2. MQIPT 路由运行的方式。 MQIPT 和 IBM MQ 之间通过因特网通信的格式由 MQIPT 路由的配置确定。 请注意，表上列示 SSL 表示此情况下您还可以使用 TLS。
3. MQIPT 路由和目标队列管理器之间使用的协议。

表 *26:* 具有单个 *MQIPT* 实例的有效配置

1 IBM MQ 源协议	**2.MQIPT 路由的方式**	**3。 IBM MQ 目标协议**
FAP	FAP 代理（缺省）	FAP
	FAP 服务器和 SSL 客户机	SSL/TLS
SSL/TLS	SSL 代理	SSL/TLS
	SSL 服务器和 FAP 客户机	FAP
	SSL 服务器和 SSL 客户机	SSL/TLS

使用多个 MQIPT 路由的兼容配置

您可能在一个或多个 MQIPT 实例上选择使用多个路由，以与 IBM MQ 通信。

第 247 页的表 27 中的列包含以下信息：

1. IBM MQ 和第一个 MQIPT 路由之间使用的协议。 IBM MQ 客户机或队列管理器可以创建连接，该连接可以使用 IBM MQ 格式和协议 (FAP) 或 SSL/TLS 协议。
2. 第一个 MQIPT 路由运行的方式。 MQIPT 和 IBM MQ 之间通过因特网通信的格式由 MQIPT 路由的配置确定。 请注意，表上列示 SSL 表示此情况下您还可以使用 TLS。
3. 第二个 MQIPT 路由运行的方式。
4. 第二个 MQIPT 路由和目标队列管理器之间使用的协议。

表 *27:* 具有多个 *MQIPT* 实例的有效配置

1 IBM MQ 源协议	**2. 第一个 MQIPT 路由的方式**	**3. 第二个 MQIPT 路由的方式**	**4. IBM MQ 目标协议**
FAP（缺省）	FAP 代理（缺省）	FAP 代理（缺省）	FAP
	FAP 服务器和 SSL 客户机	SSL 代理	SSL/TLS
		SSL 服务器和 FAP 客户机	FAP
		SSL 服务器和 SSL 客户机	SSL/TLS
	HTTP 客户机	HTTP 服务器和 SSL 客户机	SSL/TLS
	HTTPS 客户机	HTTPS 服务器和 SSL 客户机	SSL/TLS
	HTTP 客户机	HTTP 服务器	FAP
	HTTPS 客户机	HTTPS 服务器	FAP

表 27: 具有多个 *MQIPT* 实例的有效配置 *(继续)*

1 IBM MQ 源协议	2. 第一个 MQIPT 路由的方式	3. 第二个 MQIPT 路由的方式	4. IBM MQ 目标协议
SSL/TLS	SSL 代理	SSL 代理	SSL/TLS
		SSL 服务器和 FAP 客户机	FAP
		SSL 服务器和 SSL 客户机	SSL/TLS
	HTTP 客户机	HTTP 服务器	FAP
	HTTPS 客户机	HTTPS 服务器	SSL/TLS
	HTTP 客户机	HTTP 服务器和 SSL 客户机	FAP
	HTTPS 客户机	HTTPS 服务器和 SSL 客户机	SSL/TLS

支持的通道配置

支持所有 IBM MQ 通道类型，但是配置限制为 TCP/IP 连接。 对于 IBM MQ 客户机或队列管理器，MQIPT 似乎是一个目标队列管理器。 通道配置需要目标主机和端口号时，指定 MQIPT 主机名和侦听器端口号。

客户机/服务器通道
MQIPT 侦听入局客户机连接请求，然后使用 HTTP 隧道， SSL/TLS 或作为标准 IBM MQ 协议包转发这些请求。 如果 MQIPT 正在使用 HTTP 隧道或 SSL/TLS ，那么它会将它们在连接上转发到第二个 MQIPT。 如果不使用 HTTP 隧道传送，那么将在连接上将其转发到它视为目标队列管理器的目标，（尽管这可能还是进一步 MQIPT）。 当目标队列管理器接受客户机连接时，将在客户机与服务器之间传送包。

集群发送方/接收方通道
如果 MQIPT 接收来自集群发送方通道的入站请求，它将假定队列管理器支持 SOCKS，SOCKS 握手过程期间将获取真正的目标地址。 它使用与客户机连接通道完全相同的方式，将请求转发到下一个 MQIPT 或目标队列管理器。 这还包括自动定义的集群发送方通道。

发送方/接收方
如果 MQIPT 接收来自发送方通道的入站请求，它将使用与客户机连接通道完全相同的方式，将请求转发到下一个 MQIPT 或目标队列管理器。 目标队列管理器验证入站请求并且在适当的情况下启动接收方通道。 发送发和接收方通道之间的所有通信（包括安全流）将被传送。

请求方/服务器
本组合的处理方式与前面的配置相同。 服务器通道在目标队列管理器执行连接请求的验证。

请求方/发送方
如果两个队列管理器之间不允许建立直接连接，但是允许连接到 MQIPT 并且接受它的连接，那么可以使用“回调”配置。

服务器/请求方和服务器/接收方
这些由 MQIPT 以处理 `Sender/Receiver` 配置的相同方式进行处理。

通道终止和故障条件

当 MQIPT 检测到 IBM MQ 通道关闭时（正常或异常），会传播通道关闭。 如果使用 MQIPT 关闭路由，那么经过此路由的所有通道都会关闭。

MQIPT 提供了可选空闲超时工具。 如果 MQIPT 检测到通道空闲时段超过超时，会在涉及的两个连接上立即执行关闭。

通道任何一端的 IBM MQ 系统会将这些异常关闭条件视为网络故障或者视为其合作伙伴终止了通道。 然后此通道能够重新启动并恢复（如果故障发生在协议不确定时段），好像未使用 MQIPT 一样。

消息的安全性

IBM MQ 分布式队列管理可确保正确传递消息。 当通道两端之间存在 MQIPT 时，仍然会发生此情况。 MQIPT 不会存储任何消息数据，也不会参与确保正确消息传递的同步点过程。

在使用快速非持久性 IBM MQ 消息时，如果 MQIPT 路由在传输 IBM MQ 消息时发生失败或重新启动，那么消息可能会丢失。 在重新启动路由之前，请确保所有使用 MQIPT 路由的 IBM MQ 通道都处于不活动状态。

有关 IBM MQ 中消息安全的更多信息，请参阅 消息安全。

多实例队列管理器和高可用性

MQIPT 可用于高可用性环境中的多实例队列管理器。

MQIPT 没有持久状态，故障转移 MQIPT 至其他系统没有任何优势。 而是在不同系统上运行多个具有相同 `mqipt.conf` 配置文件的 MQIPT 实例。 监控每一个 MQIPT 实例的可用性，并在必要时重新启动（在同一系统上）。 这样提供一组相同的 MQIPT 实例，可将其用于路由连接。 然后，您必须确保 IBM MQ 可以将连接路由到 MQIPT，并且该 MQIPT 可以将那些连接转发到目标队列管理器。

可以使用各种方式将出站 IBM MQ 通道定向到可用的 MQIPT 实例，例如：

- 使用 WebSphere Edge Components 产品中的负载均衡器或高可用性路由器，例如 IBM Network Dispatcher 。
- 使用逗号分隔列表在 IBM MQ 通道定义中指定多个连接名称。 然后，IBM MQ 尝试依次连接每个 MQIPT 地址，直到找到可用的 MQIPT 实例。

您还必须将连接从 MQIPT 定向到目标队列管理器。 如果高可用性配置确保 IP 地址使用目标队列管理器进行故障转移，那么不需要特别的 MQIPT 配置：在 **Destination** 路由属性中指定目标 IP 地址，允许故障转移操作移动队列管理器的 IP 地址。

但是，如果故障转移后，队列管理器的 IP 地址发生更改，那么您必须安排 MQIPT 将连接转发到正确的目标。 可以通过几种方式完成此任务：

- 写入一个路由出口来检查哪个 IP 地址和端口号是可以访问的，然后覆盖每个连接的路由目标。 MQIPT 随附了一些路由出口样本；可以对其进行改写以用于此目的。
- 使用高可用性负载均衡器重定向连接。
- 定义多个 MQIPT 路由，针对可能在运行队列管理器的每一个 IP 地址和端口定义一个路由。 然后，将 IBM MQ 连接定向到各种 MQIPT 路由，例如，在出站通道的连接名称中，以逗号分隔列表形式列出所有路由 IP 地址和端口号。

网络路径上所有端到端组件的调整也很重要：

1. 不可用系统的连接尝试必须及时断开，这样重新连接尝试将移动到第一个可用目标。

   对于 MQIPT SSL 路由，调整 **SSLClientConnectTimeout** 路由属性以确保不可用目标的及时连接断开。 有关 IBM MQ 调整参数的详细信息，请参阅 IBM MQ 文档。 此外，查看操作系统文档，以获取操作系统 TCP/IP 调整的详细信息。 在任何情况下，失败的连接尝试应快速返回网络故障（例如，TCP 复位分组），或者应该超时而不必过分延迟。
2. 故障系统的活动连接必须及时断开，这样可以建立新连接。

   您还应该考虑连接主动使用 MQIPT 时故障转移的影响。 网络连接很可能在故障转移期间断开。 对于客户机应用程序，您可以使用 IBM MQ 自动客户机重新连接功能重新建立已断开的连接。 对于消息通道，您可以指定较短的重试时间间隔，以便通道及时重新连接。 查看 IBM MQ 文档以获取有关自动客户机重新连接和消息通道重试配置的更多信息。

V 9.3.5 IBM MQ Console 和 REST API

您可以使用 IBM MQ Console 和 REST API 来管理 IBM MQ，并使用 HTTP 来执行消息传递操作。

- 您可以使用 IBM MQ Console 从 Web 浏览器执行基本管理任务。 有关更多信息，请参阅 使用 IBM MQ Console 进行管理。
- 您可以使用 administrative REST API 来管理 IBM MQ 对象，例如队列管理器和队列以及 Managed File Transfer 代理和传输。 有关更多信息，请参阅 使用 REST API 进行管理。

- 您可以使用 messaging REST API 来执行简单的点到点和发布消息传递。 有关更多信息，请参阅 使用 REST API 进行消息传递。

安装选项

IBM MQ Console 和 REST API 在名为 mqweb 的 WebSphere Liberty 服务器中运行。 从 IBM MQ 9.3.5 开始，您可以将 mqweb 服务器作为 IBM MQ 安装中的可选组件进行安装，也可以作为独立 IBM MQ Web Server 安装进行安装。

Linux V 9.3.5 独立安装的 IBM MQ Web Server

从 IBM MQ 9.3.5 开始， mqweb 服务器可以在 IBM MQ Web Server 的独立安装中运行。 独立 IBM MQ Web Server 安装使您能够在独立于 IBM MQ 安装的系统上安装和运行 mqweb 服务器。 通过安装独立 IBM MQ Web Server ，您可以更灵活地选择在哪些系统上运行 mqweb 服务器，以及选择在哪些系统上运行 mqweb 服务器的系统数量。 如果需要，可以在不同机器上运行 mqweb 服务器的多个实例，以提供所需的可伸缩性和可用性。

如果您已购买 IBM MQ 权利，那么可以安装所需数量的独立 IBM MQ Web Server 副本。 IBM MQ Web Server 安装不会计入已购买的 IBM MQ 权利。 有关 IBM MQ 许可的更多信息，请参阅 IBM MQ 许可证信息。

以下限制适用于独立 IBM MQ Web Server 安装:

- IBM MQ Console 只能用于管理远程队列管理器。
- messaging REST API 只能与远程队列管理器配合使用。
- administrative REST API 不可用。

仅在 Linux 平台上支持独立 IBM MQ Web Server 。

有关安装独立 IBM MQ Web Server 的更多信息，请参阅 安装独立 IBM MQ Web Server。

IBM MQ 安装的可选组件

您可以选择在 IBM MQ 安装过程中安装 IBM MQ Console 和 REST API 组件。

当 mqweb 服务器在 IBM MQ 安装中运行时，所有 IBM MQ Console 和 REST API 功能部件都可用。

- IBM MQ Console 可用于管理本地和远程队列管理器。
- messaging REST API 可与本地和远程队列管理器配合使用。
- administrative REST API 可用于管理本地和远程队列管理器。

要使用 IBM MQ Console 和 REST API 组件，请在 IBM MQ 安装过程中安装以下组件:

- AIX 在 AIX 上，安装 `mqm.web.rte` 文件集。
- IBM i 在 IBM i 上，安装 WEB 组件。
- Linux 在 Linux 上，安装 `MQSeriesWeb` 组件。
- Windows 在 Windows 上，安装 `Web Administration` 功能部件。
- z/OS 在 z/OS 上，安装 `IBM MQ for z/OS UNIX System Services Web Components` 功能部件。

声明

本信息是为在美国国内供应的产品和服务而编写的。

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暗示只能使用 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，则由用户自行负责。

IBM 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。 您可以用书面方式将许可查询寄往：

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：

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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 可以随时对本资料中描述的产品和/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。 所有这些名字都是虚构的，若现实生活中实际业务企业使用的名字和地址与此相似，纯属巧合。

版权许可：

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。 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 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 The 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 (https://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: