9.4

IBM MQ 的监视和性能

注

在使用本资料及其支持的产品之前，请阅读第 351 页的『声明』中的信息。

本版本适用于 IBM® MQ V 9 发行版 4 以及所有后续发行版和修订版，直到在新版本中另有声明为止。

内容

IBM MQ 监视和性能

使用本部分中的监视信息和指导以及特定调整提示来帮助提高队列管理器网络的性能。

关于此任务

根据队列管理器网络的大小和复杂性，您可以从监视网络中获取一系列信息。 您可以使用该信息以及特定调整提示中提供的信息来帮助您调整网络性能。

监视 IBM MQ 网络

IBM MQ 中提供了一些监视技术，用于获取有关队列管理器网络运行方式的统计信息和其他特定信息。 使用本节中的监视信息和指南来帮助提高队列管理器网络的性能。

以下列表提供了监视队列管理器网络的原因示例:

- 检测队列管理器网络中的问题。
- 帮助确定队列管理器网络中问题的原因。
- 提高队列管理器网络的效率。
- 熟悉队列管理器网络的运行。
- 确认队列管理器网络正在正确运行。
- 发生特定事件时生成消息。
- 记录消息活动。
- 确定消息的最后已知位置。
- 实时检查队列管理器网络的各种统计信息。
- 生成审计跟踪。
- 说明应用程序资源使用情况。
- 容量规划。

V 9.4.0 ALW OpenTelemetry 集成

您可以将 IBM MQ 与 OpenTelemetry 跟踪系统集成。

OpenTelemetry 跟踪

OpenTelemetry 跟踪使您能够观察应用程序在数据流中的行为。 数据流可以并通常会包含几个不同的应用程序。 跟踪可以向您展示整个旅程，并提供对每个应用程序行为的洞察。 IBM MQ 提供了跟踪服务，使您能够与 OpenTelemetry 跟踪系统集成。

IBM MQ OpenTelemetry 跟踪服务作为 IBM MQ API 出口实现。 它作为 IBM 支持程序提供，这意味着您有权使用跟踪服务并作为 IBM MQ 权利的一部分接收支持。 请注意， IBM MQ 权利仅允许使用 IBM Instana 支持程序的 IBM MQ 跟踪出口组件。

除非您将跟踪出口与 IBM Instana 监视系统和 IBM Instana 权利配合使用，否则必须向 IBM MQ 支持人员报告因使用该出口而产生的问题。

可以在以下位置下载出口: https://ibm.biz/mqinstanaexit。

安装和配置 IBM MQ Open Telemetry 跟踪服务

可以在以下位置找到有关如何安装和配置 IBM MQ 跟踪的完整详细信息: IBM MQ Tracing。

可在此处找到 IBM Instana IBM MQ 出口支持的平台的详细信息: 本地 IBM MQ 支持的平台。

Multi 队列管理器运行状况检查行为

队列管理器执行定期运行状况检查以确保稳定可靠的性能。 本主题描述了队列管理器进行的一些运行状况检查，并说明了如何根据环境需求配置这些检查。

在大多数环境中，缺省配置是合适的，不需要更改执行这些检查的频率。 即使使用缺省设置，了解在检测到问题以及可能导致检查失败的环境问题时队列管理器的行为方式也很有用。 本主题旨在解释其中的一些行为。

警告: 请勿对这些检查的频率进行任何更改，除非 IBM 支持人员建议您这样做。

队列管理器的不同组件使用各种方法来检测和解决不一致问题，本主题并非旨在描述所有此类机制。 例如，IBM MQ 进程使用各种机制来确保它们所依赖的其他进程仍在运行。 所描述的行为是由执行控制器定期执行以发现环境或其他意外情况的行为。 (执行控制器是启动和管理大多数其他队列管理器进程的主 IBM MQ 进程。) 因为它们是定期检查，所以它们是按特定时间间隔进行的，可以通过设置适当的调整参数在一定程度上进行修改。

描述的某些检查是由专用的运行状况检查线程进行的。 如果检测到运行状况检查线程本身存在问题，那么会将警告消息 AMQ5066 写入队列管理器错误日志。

本主题中描述的行为可能会在未来发行版中发生更改，例如，如果观察到不同的缺省值在特定平台或配置上更稳定。

常规运行状况检查

队列管理器定期执行各种检查。 缺省情况下，这些检查每 10 秒执行一次 (在某些情况下，检查允许在报告错误之前进行两个循环，从而导致此类检查的时间间隔为 20 秒)。 在检查期间，队列管理器确保作为队列管理器一部分运行的各种进程仍在运行。 对于本机 HA 队列管理器，它会检查队列管理器是否成功将数据复制到备用实例。

如果此时某个关键检查失败 (例如，如果 `amqzmuc0` 进程不再运行)，那么队列管理器将无法继续运行。 但是，大多数检查都是为了整理不再需要的系统资源，这些资源可能只是导致将消息写入队列管理器错误日志。

在大多数情况下，不需要改变这些常规健康检查的频率。 将立即检测队列管理器或操作环境中的大多数事件，而不需要常规运行状况检查过程来检测这些事件。 此过程充当对队列管理器中其他位置未检测到的任何内容的定期检查。 如果需要，可以使用 **ECHeartBeatLen** 调整参数来配置频率。 最小值为 10000 毫秒 (10 秒)。 最大值为 60000 毫秒 (60 秒)。 如果设置为其最大值 60000，那么这可能会导致某些检查延迟两分钟。

检查是否正在进行日志进度

队列管理器检查是否以合理的速率写入日志。 这并不是检查记录器的性能是否最佳，而是为了发现可能需要进一步关注的情况。 例如，如果存储日志文件的磁盘特别慢，或者如果队列管理器在容器化环境中没有接收到足够的 CPU 时间来执行其所有工作。

如果此检查失败，那么队列管理器所执行的操作取决于正在使用的队列管理器的类型:

- 在非 HA 队列管理器上:
  - 写入 xecL_W_PERFORMANCE_BOTTLENECK FDC。 这可用作指示系统的某些部分可能需要进一步关注。 队列管理器仍在运行。 如果在 errors 目录中看到 xecL_W_PERFORMANCE_BOTTLENECK FDC，那么可能需要与您的存储器或平台团队协作，以了解底层系统资源是否足以运行 IBM MQ 。 如果 IBM MQ 正在过度落实的节点上的容器中运行，那么 IBM MQ 可能接收不到足够的预定 CPU 时间来执行其所有消息传递工作负载。
  - 从 IBM MQ 9.3.0 开始，会将警告消息 AMQ5068W 写入队列管理器错误日志，并且不会写入 xecL_W_PERFORMANCE_BOTTLENECK FDC。 如果在日志中看到 AMQ5068W 消息，那么可能需要与存储器或平台团队协作，以了解底层系统资源是否足以使 IBM MQ 运行。 如果 IBM MQ 正在过度落实的节点上的容器中运行，那么 IBM MQ 可能接收不到足够的预定 CPU 时间来执行其所有消息传递工作负载。 如果连续写入五条 AMQ5068W 警告消息，那么将写入 xecL_W_PERFORMANCE_BOTTLENECK FDC。

- 在多实例队列管理器上:
  - 如果日志进度运行状况检查失败，那么主实例将结束。 如果备用实例可用，那么它将启动并成为主实例。
  - 从 IBM MQ 9.3.0 开始，主实例在结束前检查备用实例是否可用。 如果备用队列管理器可用于故障转移到主实例结束。 此外，会将警告消息 AMQ5068W 写入队列管理器错误日志。
- 在本机 HA 队列管理器上，此检查的行为方式与非 HA 队列管理器相同。
- 在 RDQM (复制的数据队列管理器) 上，此检查的行为方式与非 HA 队列管理器相同。

IBM MQ 日志的进度问题可能是由队列管理器本身的性能问题引起的。

缺省情况下，此检查每 60 秒进行一次，尽管队列管理器在执行操作之前等待两个周期的检查。 这意味着使用缺省设置时，必须在队列管理器写入错误消息 (或者在 HA 队列管理器情况下发生故障转移) 之前经过两分钟。

在大多数情况下，即使在文件系统缓慢或队列管理器分配了少量 CPU 时间的情况下，缺省行为也适用，因为其他检查 (例如文件锁定 (请参阅 第 7 页的『检查文件锁定是否仍挂起』) 和基本文件系统操作) 将导致主实例在执行此检查之前进行故障转移。 如果需要，可以使用 **LivenessHeartBeatLen** 调整参数来配置此检查的频率。 可将其配置为 600 秒 (10 分钟) 的最大值。 0 的最小值具有完全禁用检查的效果。 对于非 HA 队列管理器，检查的唯一影响是队列管理器错误日志中存在额外的警告消息。 对于多实例队列管理器，您可以配置 **LivenessHeartBeatLen** 以使队列管理器的主实例更快 (通过减小值) 或更慢 (通过增大值) 进行故障转移。 如果您的环境偶尔会迁到文件系统 IO 速度非常慢但您希望队列管理器的主实例保持运行的情况，那么增大该值以降低日志进度检查的频率可能会很有用。 如果您有未设计为自动重新连接到备用实例的应用程序，并且需要手动干预来重新启动这些应用程序，那么这可能很有用。

注: 如果 **ECHeartBeatLen** 已增大，那么这将影响 **LivenessHeartBeatLen** 检查的计时。 执行常规运行状况检查时将执行日志进度检查，因此减少常规运行状况检查 (**ECHeartBeatLen**) 的频率可能会导致日志进度检查在配置的 **LivenessHeartBeatLen** 之后长达 30 秒。

常规日志记录文件系统性能

V 9.4.0

从 IBM MQ 9.4.0 开始，在队列管理器错误日志中发出警告消息 AMQ6729W (如果对此存储器执行的常规读/写操作似乎花费的时间超过预期时间)。 您可以使用 **AMQ_IODELAY** 环境变量来微调诊断和计时，以帮助诊断存储器性能问题，或者增加对此类延迟的容错。 有关更多信息，请参阅 AMQ_IODELAY，AMQ_IODELAY_INMS 和 AMQ_IODELAY_FFST。

检查文件锁定是否仍挂起

对于多实例队列管理器，执行控制器会定期检查以确保它仍持有主多实例文件的互斥锁定。 在许多情况下，如果由于 NFS 服务器的问题而丢失了锁定，那么主实例几乎会立即故障转移 (在进行此检查之前)。 将执行其他定期文件锁定检查，以确保主队列管理器在发生异常文件系统问题时发生故障转移。

缺省情况下，这些文件锁定检查每 20 秒执行一次。 如果需要，可以通过设置 **FileLockHeartBeatLen** 调整参数来更改此值。 调整参数的缺省值为 10 秒 (队列管理器允许在执行操作之前进行两次循环检查，导致缺省行为为每 20 秒检查一次)。 调整参数的最小值为 10 秒，最大值为 600 秒 (10 分钟)。

注: 如果 **ECHeartBeatLen** 已增加，那么这会影响 **FileLockHeartBeatLen** 检查的计时。 执行常规运行状况检查时将执行文件锁定检查，因此减少常规运行状况检查 (**ECHeartBeatLen**) 的频率可能会导致在配置的 **FileLockHeartBeatLen** 之后最多 30 秒进行文件锁定检查。

检查用户应用程序运行状况

在终止之前，队列管理器会定期检查不再运行的任何本地绑定应用程序是否已执行 MQDISC MQI 调用。 这些检查与 第 6 页的『常规运行状况检查』中描述的常规运行状况检查同时执行。 因此，此类检查的缺省时间间隔为 10000 毫秒 (10 秒)，更改 **ECHeartBeatLen** 调整参数的值将更改执行这些检查的频率。 此检查主要是为了确保释放与已连接应用程序相关联的任何资源，这不会导致 HA 或非 HA 队列管理器结束或故障转移到备用实例。

代理进程单独检测已终止而未发出 MQDISC MQI 调用的 IBM MQ 客户机应用程序，并释放与连接关联的任何资源。

相关概念

高可用性配置

事件监视

事件监视是检测队列管理器网络中出现的 检测事件 的过程。 检测事件是队列管理器或通道实例检测到的事件的逻辑组合。 此类事件会导致队列管理器或通道实例将特殊消息 (称为 事件消息) 放在事件队列上。

IBM MQ 检测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。 使用这些事件来监视队列管理器网络中队列管理器的操作，以实现以下目标:

- 检测队列管理器网络中的问题。
- 帮助确定队列管理器网络中问题的原因。
- 生成审计跟踪。
- 对队列管理器状态更改作出反应

相关参考

第 11 页的『事件类型』
使用此页面来查看队列管理器或通道实例可以报告的检测事件类型

事件消息引用

事件消息格式

发布 IBM MQ 事件消息

如何准备 IBM MQ 以发布事件消息。

关于此任务

事件消息将写入名为 SYSTEM.ADMIN.<feature name>.EVENT 的特殊命名队列。

这些事件队列需要注意的重要事项是，它是重要的名称。 缺省情况下，在队列管理器上，所有事件队列都定义为本地队列。 但是，您可以删除这些队列并将其重新定义 (可能作为远程队列) ，以便将所有事件都提供给专用事件处理队列管理器。 或者，可以使用指向主题对象的别名队列。

在任何一种情况下，任何重定向方法都要求读取事件队列的应用程序没有对要从中读取的队列的名称进行硬编码。 因此，您必须能够配置应用程序正在从中读取的队列。

以下命令显示如何使用以下假定来重新定义事件队列，以便发布事件消息。 您已:

- 未使用事件启动，或者
- 已从现有事件队列中除去所有消息，并已在执行这些步骤之前删除本地队列。

这些步骤仅显示要重新定义的 QMGR 和 CHANNEL 事件队列，但可以对所有事件进行扩展。

注: 主题字符串设计为使应用程序可以使用通配符预订所有事件，或根据需要预订特定事件。

过程

发出下列命令:

```
DEFINE TOPIC(ADMIN.QMGR.EVENT)    TOPICSTR('Events/QMgr')
DEFINE TOPIC(ADMIN.CHANNEL.EVENT) TOPICSTR('Events/Channel')

DEFINE QALIAS(SYSTEM.ADMIN.QMGR.EVENT)    TARGTYPE(TOPIC) TARGET(ADMIN.QMGR.EVENT)
DEFINE QALIAS(SYSTEM.ADMIN.CHANNEL.EVENT) TARGTYPE(TOPIC) TARGET(ADMIN.CHANNEL.EVENT)

DEFINE QLOCAL(ADMIN.EVENT)
DEFINE QLOCAL(ADMIN.QMGR.EVENT)

DEFINE SUB(EVENTS.ALL)  TOPICSTR('Events/+')    PSPROP(NONE)
```

```
        DESTCLAS(PROVIDED) DEST(ADMIN.EVENT)
DEFINE SUB(EVENTS.QMGR) TOPICSTR('Events/QMgr') PSPROP(NONE)
        DESTCLAS(PROVIDED) DEST(ADMIN.QMGR.EVENT)
```

假定事件读取应用程序能够从任何队列中读取事件消息，那么可以根据需要将该应用程序重新配置为从上面定义的其中一个队列中读取。

DEFINE SUB 命令上的 PSPROP(NONE) 配置是为了确保由发布/预订引擎 (例如 MQTopicString) 添加的任何消息属性都不会添加到事件消息中，从而确保现有应用程序可以继续工作而保持不变。

此外，应用程序还可以使用 MQSUB 调用直接预订以接收信息，作为替代方法，而不是使用管理 DEFINE SUB 命令。

现在，多个应用程序能够使用队列管理器在事件中发出的信息。

检测事件

检测事件是队列管理器或通道实例检测特殊消息 (称为 事件消息) 并将其放入事件队列的条件的逻辑组合。

IBM MQ 检测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。 您可以使用这些事件来监视队列管理器的操作 (使用其他方法，例如 Tivoli NetView for z/OS)。

第 10 页的图 1 说明了检测事件的概念。

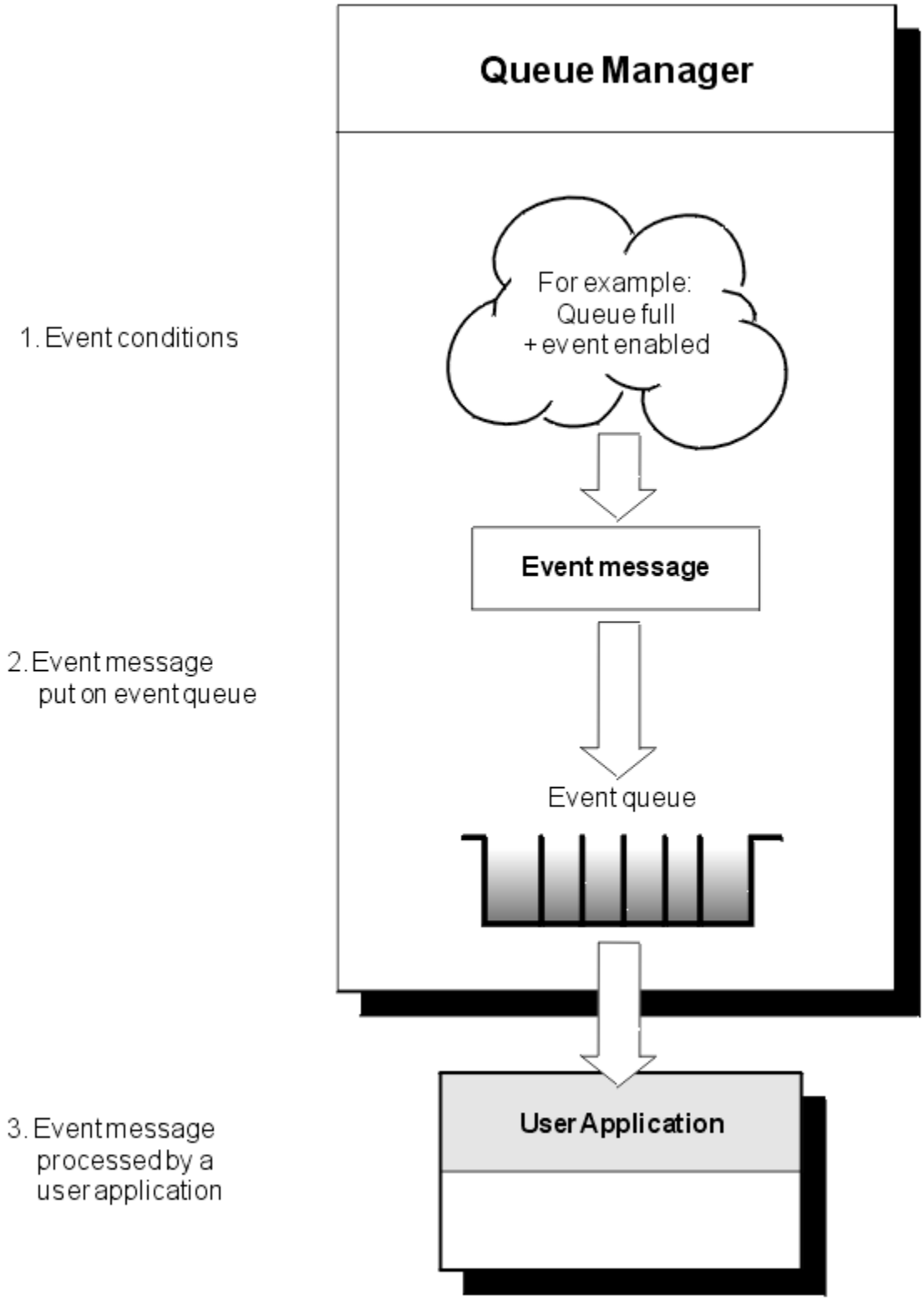


图 *1:* 了解检测事件

事件监视应用程序

使用事件来监视队列管理器的应用程序必须包含以下供应:

1. 在网络中的队列管理器之间设置通道。

2. 实现所需的数据转换。 数据转换的正常规则适用。 例如，如果要从 z/OS 队列管理器监视 UNIX 系统队列管理器上的事件，请确保将 EBCDIC 转换为 ASCII。

通过事件队列进行事件通知

发生事件时，队列管理器会将事件消息放入相应的事件队列 (如果已定义)。 事件消息包含有关可通过编写执行以下步骤的适当 MQI 应用程序来检索的事件的信息:

- 从队列获取消息。
- 处理消息以抽取事件数据。

相关信息描述了事件消息的格式。

导致事件的条件

以下列表提供了可导致检测事件的条件示例:

- 已达到队列上消息数的阈值限制。
- 通道实例已启动或停止。
- 队列管理器变为活动状态，或被请求停止。
- 应用程序尝试打开队列，指定在 IBM MQ for IBM i AIX, Linux®, and Windows 系统上未授权的用户标识。
- 将创建，删除，更改或刷新对象。
- MQSC 或 PCF 命令成功运行。
- 队列管理器开始写入新的日志扩展数据块。
- 如果满足事件条件，请将消息放在死信队列上。

相关概念

第 21 页的『性能事件』
性能事件与可能影响使用指定队列的应用程序的性能的条件相关。 性能事件的作用域是队列。 一个队列上的 **MQPUT** 调用和 **MQGET** 调用不会影响另一个队列上性能事件的生成。

第 49 页的『用于监视 Multiplatforms 版上的检测事件的样本程序』
amqsevt 格式化队列管理器可以创建的检测事件，并随 IBM MQ for Multiplatforms 一起提供。 程序从事件队列读取消息，并将其格式化为可读字符串。

事件类型

使用此页面来查看队列管理器或通道实例可以报告的检测事件类型

IBM MQ 检测事件具有以下类型:

- 队列管理器事件
- 通道和网桥事件
- 性能事件
- 配置事件
- 命令事件
- 记录器事件
- 本地事件

对于每个队列管理器，每个事件类别都有自己的事件队列。 该类别中的所有事件都会导致将事件消息放入同一队列。

此事件队列:	包含来自以下位置的消息:
SYSTEM.ADMIN.QMGR.EVENT	队列管理器事件
SYSTEM.ADMIN.CHANNEL.EVENT	通道事件
SYSTEM.ADMIN.PERFM.EVENT	性能事件

此事件队列:	包含来自以下位置的消息:
SYSTEM.ADMIN.CONFIG.EVENT	配置事件
SYSTEM.ADMIN.COMMAND.EVENT	命令事件
SYSTEM.ADMIN.LOGGER.EVENT	记录器事件
SYSTEM.ADMIN.PUBSUB.EVENT	获取与发布/预订相关的事件。 仅与多点广播配合使用。 有关更多信息，请参阅 多点广播应用程序监视。

通过将检测事件合并到您自己的系统管理应用程序中，可以跨多个队列管理器，跨多个不同节点以及针对多个 IBM MQ 应用程序监视活动。 特别是，您可以从单个节点 (对于支持 IBM MQ 事件的那些节点) 监视系统中的所有节点，如第 12 页的图 2 中所示。

可通过用户编写的报告机制向管理应用程序报告检测事件，该管理应用程序可将事件呈现给操作员。

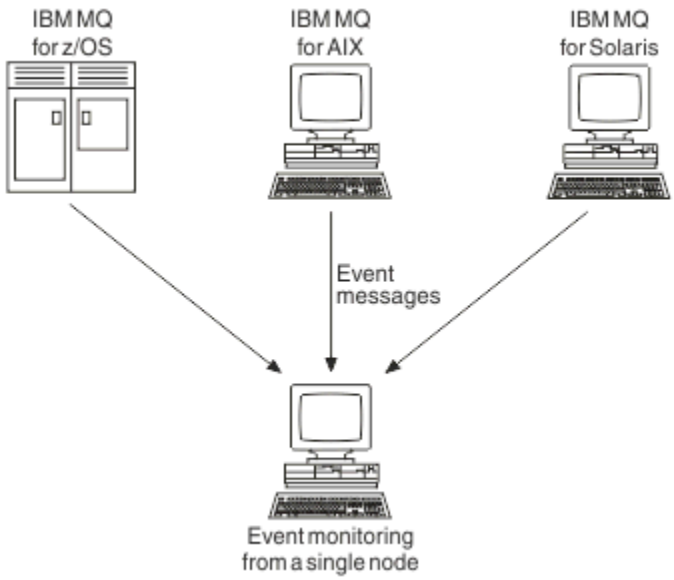


图 *2:* 在单个节点上跨不同平台监视队列管理器

检测事件还使充当其他管理网络 (例如 Tivoli NetView for z/OS) 的代理程序的应用程序能够监视报告并创建相应的警报。

队列管理器事件
队列管理器事件与队列管理器中资源的使用相关。 例如，如果应用程序尝试将消息放入不存在的队列中，那么将生成队列管理器事件。

以下示例显示可导致队列管理器事件的条件:

- 应用程序发出失败的 MQI 调用。 来自调用的原因码与事件消息中的原因码相同。

  在队列管理器的内部操作期间可能会发生类似的情况; 例如，生成报告消息时。 事件消息中的原因码可能与 MQI 原因码匹配，即使它未与任何应用程序关联也是如此。 请勿假定，因为事件消息原因码看起来像 MQI 原因码，所以事件必然是由来自应用程序的 MQI 调用失败导致的。
- 向队列管理器发出命令，处理此命令会导致事件。 例如:
  - 队列管理器已停止或启动。
  - 如果未授权关联的用户标识执行该命令，那么将发出该命令。

IBM MQ 将队列管理器事件的消息放在 SYSTEM.ADMIN.QMGR.EVENT 队列支持以下队列管理器事件类型:

ALW 权限 (仅在 AIX, Linux, and Windows 上)

权限事件报告授权，例如应用程序尝试打开其不具有必需权限的队列，或者从不具有必需权限的用户标识发出命令。 权限事件消息可以包含以下事件数据:

- 未授权 (类型 1)
- 未授权 (类型 2)
- 未授权 (类型 3)
- 未授权 (类型 4)
- 未授权 (类型 5)
- 未授权 (类型 6)

所有权限事件仅在 AIX, Linux, and Windows 上有效。

禁止

禁止事件指示已尝试对队列执行 MQPUT 或 MQGET 操作，该队列禁止放置或获取，或对主题禁止发布的主题执行 MQPUT 或 MQGET 操作。 禁止事件消息可以包含以下事件数据:

- 获取已禁止
- 放入已禁止

本地

当应用程序或队列管理器无法访问本地队列或其他本地对象 (例如，由于尚未定义该对象) 时，队列管理器可以生成本地事件消息。 本地事件消息可以包含以下事件数据:

- 别名基本队列类型错误
- 未知别名基本队列
- 未知对象名

远程

当应用程序或队列管理器无法访问另一个队列管理器上的远程队列 (例如，未正确定义传输队列) 时，队列管理器可以生成远程事件消息。 远程事件消息可以包含以下事件数据:

- 缺省传输队列类型错误
- 缺省传输队列使用错误
- 队列类型错误
- 远程队列名称错误
- 传输队列类型错误
- 传输队列用法错误
- 未知缺省传输队列
- 未知远程队列管理器
- 未知传输队列

启动和停止

启动和停止事件指示队列管理器已启动或已请求停止或停顿。

z/OS z/OS 仅支持启动事件。

除非 SYSTEM.ADMIN.QMGR.EVENT 队列定义为持久队列。 启动和停止事件消息可以包含以下事件数据:

- 队列管理器活动
- 队列管理器不活动

对于此列表中的每个事件类型，可以设置队列管理器属性以启用或禁用事件类型。

通道和网桥事件

由于在操作期间检测到的条件，通道会报告这些事件。 例如，当通道实例停止时。

在以下情况下生成通道事件:

- 当命令启动或停止通道时。
- 通道实例何时启动或停止。
- 当通道在获取消息时接收到转换错误警告。
- 尝试自动创建通道时; 无论尝试成功还是失败，都会生成事件。

注: 客户机连接不会导致 "通道已启动" 或 "通道已停止" 事件。

使用命令启动通道时，将生成事件。 通道实例启动时将生成另一个事件。 但是，通过侦听器， **runmqchl** 命令或队列管理器触发器消息启动通道不会生成事件。 在这些情况下，仅当通道实例启动时才会生成事件。

成功启动或停止通道命令至少生成两个事件。 将为通道连接的两个队列管理器生成这些事件 (前提是它们支持事件)。

如果将通道事件放在事件队列上，那么错误情况会导致队列管理器创建事件。

通道和网桥事件的事件消息放在 SYSTEM.ADMIN.CHANNEL.EVENT 队列。

通道事件消息可以包含以下事件数据:

- 通道已激活
- 通道自动定义错误
- 通道自动定义正常
- 通道转换错误
- 通道未激活
- 通道已启动
- 通道已停止
- 用户已停止通道
- 已阻塞通道

IMS 网桥事件 (仅限 z/OS)

z/OS

当 IMS 网桥启动或停止时，将报告这些事件。

IMS 网桥事件消息可以包含以下事件数据:

- 网桥已启动
- 网桥已停止

SSL 事件

唯一的 TLS 事件是 "通道 SSL 错误" 事件。 当使用 TLS 的通道无法建立 TLS 连接时，将报告此事件。

SSL 事件消息可以包含以下事件数据:

- 通道 SSL 错误
- 通道 SSL 警告

性能事件
性能事件是资源已达到阈值条件的通知。 例如，已达到队列深度限制。

性能事件与可能影响使用指定队列的应用程序的性能的条件相关。 不会为事件队列本身生成这些消息。

将在消息数据中的命令标识字段中返回事件类型。

如果队列管理器尝试将队列管理器事件或性能事件消息放在事件队列上，并且检测到通常会创建事件的错误，那么不会创建其他事件，并且不会执行任何操作。

工作单元中的 MQGET 和 MQPUT 调用可以生成性能事件，而无论工作单元是已落实还是已回退。

性能事件的事件消息将放在 SYSTEM.ADMIN.PERFM.EVENT 队列。

有两种类型的性能事件:

队列深度事件
队列深度事件与队列中的消息数相关; 即，队列已满或为空的程度。 共享队列支持这些事件。 队列深度事件消息可以包含以下事件数据:

- 队列深度过高
- 队列深度过低
- 队列已满

队列服务时间间隔事件
队列服务时间间隔事件与是否在用户指定的时间间隔内处理消息相关。 共享队列不支持这些事件。

z/OS IBM MQ for z/OS 支持 QSGDISP (SHARED) 队列的队列深度事件，但不支持服务时间间隔事件。 队列管理器和通道事件仍不受共享队列的影响。 队列服务事件消息可以包含以下事件数据:

- 队列服务时间间隔过长
- 队列服务时间间隔正常

配置事件
配置事件在显式请求配置事件时生成，或者在创建，修改或删除对象时自动生成。

配置事件消息包含有关对象属性的信息。 例如，如果创建了名称列表对象，那么将生成配置事件消息，并包含有关名称列表对象的属性的信息。

配置事件的事件消息将放在 SYSTEM.ADMIN.CONFIG.EVENT 队列。

配置事件按以下方式工作:

- 通过将事件消息写入 SYSTEM.ADMIN.CONFIG.EVENT 队列。 您可以通过 ALTER QMGR 命令上的 **CONFIGEV** 参数来写入这些事件。
- 当 DEFINE， ALTER 或 DELETE 命令对对象执行操作或使用 MQSET 调用时，将生成这些事件。
- 您可以使用 REFRESH QMGR TYPE (CONFIGEV) 命令创建当前队列管理器配置的基线图，这将为队列管理器中的每个对象创建事件消息。 请注意，由于这可能是一项耗时的操作，如果您有许多对象，那么可以在命令上使用 NAME 和 OBJECT 限定符将任务分解为更小的对象集。
- 事件消息记录四个可能的原因之一:
  - MQRC_CONFIG_CHANGE_OBJECT
  - MQRC_CONFIG_CREATE_OBJECT
  - MQRC_CONFIG_DELETE_OBJECT
  - MQRC_CONFIG_REFRESH_OBJECT

  MQRC_CONFIG_CHANGE_OBJECT， MQRC_CONFIG_CREATE_OBJECT 或 MQRC_CONFIG_DELETE_OBJECT 用于针对对象可能发出的相应 MQSC 或 PCF 命令。

  MQRC_CONFIG_REFRESH_OBJECT 用于在创建基本线图片时写入的事件消息。

有四种类型的配置事件:

创建对象事件
创建对象事件是在创建对象时生成的。 事件消息包含以下事件数据: Create object。

更改对象事件
更改对象事件是在更改对象时生成的。 事件消息包含以下事件数据: Change object。

删除对象事件
删除对象事件是在删除对象时生成的。 事件消息包含以下事件数据: Delete object。

刷新对象事件
刷新对象事件由要刷新的显式请求生成。 事件消息包含以下事件数据: 刷新对象。

命令事件
当 MQSC 或 PCF 命令成功运行时，将报告命令事件。

命令事件消息包含有关命令的源，上下文和内容的信息。 例如，如果 MQSC 命令 ALTER QLOCAL 成功运行，那么将生成带有此类信息的命令事件消息。

命令事件的事件消息放在 SYSTEM.ADMIN.COMMAND.EVENT 队列。

命令事件包含以下事件数据: Command。

Multi 记录器事件

当使用线性日志记录的队列管理器开始将日志记录写入新的日志扩展数据块 IBM i 或在 IBM i 上写入新的日志接收器时，将报告记录器事件。 z/OS 记录器事件不可用于 IBM MQ for z/OS。

记录器事件消息包含指定队列管理器重新启动队列管理器或介质恢复所需的日志扩展数据块的信息。

记录器事件的事件消息将放在 SYSTEM.ADMIN.LOGGER.EVENT 队列。

记录器事件消息包含以下事件数据: Logger。

事件消息数据摘要

使用此摘要可获取有关每种类型的事件消息可包含的事件数据的信息。

事件类型	**请参阅以下主题**
权限事件	未授权 (类型 1)
	未授权 (类型 2)
	未授权 (类型 3)
	未授权 (类型 4)
	未授权 (类型 5)
	未授权 (类型 6)
通道事件	通道已激活
	通道自动定义错误
	通道自动定义正常
	已阻塞通道
	通道转换错误
	通道未激活
	通道已启动
	通道已停止
	用户已停止通道
命令事件	命令
配置事件	创建对象
	更改对象
	删除对象
	刷新对象
IMS 网桥事件	网桥已启动
	网桥已停止

事件类型	请参阅以下主题
禁止事件	获取已禁止
	放入已禁止
本地事件	别名基本队列类型错误
	未知别名基本队列
	未知对象名
记录器事件	记录器
性能事件	队列深度过高
	队列深度过低
	队列已满
	队列服务时间间隔过长
	队列服务时间间隔正常
远程事件	缺省传输队列类型错误
	缺省传输队列使用错误
	队列类型错误
	远程队列名称错误
	传输队列类型错误
	传输队列用法错误
	未知缺省传输队列
	未知远程队列管理器
	未知传输队列
SSL 事件	通道 SSL 错误
启动和停止事件	队列管理器活动
	队列管理器不活动

控制事件

根据事件类型，通过为队列管理器和/或队列属性指定相应的值来启用和禁用事件。

您必须启用要生成的每个检测事件。 例如，导致 "队列已满" 事件的条件包括:

- 为指定的队列启用了 "队列已满" 事件，并且
- 应用程序发出 MQPUT 请求以将消息放入该队列，但请求失败，因为队列已满。

使用下列任何方法来启用和禁用事件:

- IBM MQ 脚本命令 (MQSC)。
- 相应的 IBM MQ PCF 命令。
- z/OS z/OS 上队列管理器的操作和控制面板。
- IBM MQ Explorer.

注: 只能通过命令为队列和队列管理器设置与事件相关的属性。 MQI 调用 MQSET 不支持与事件相关的属性。

相关概念

第 9 页的『检测事件』

检测事件是队列管理器或通道实例检测特殊消息 (称为 事件消息) 并将其放入事件队列的条件的逻辑组合。

在 z/OS 上使用操作和控制面板

相关任务

自动执行管理任务

使用可编程命令格式

相关参考

第 11 页的『事件类型』

使用此页面来查看队列管理器或通道实例可以报告的检测事件类型

MQSC 命令

控制队列管理器事件

您可以使用队列管理器属性来控制队列管理器事件。 要启用队列管理器事件，请将相应的队列管理器属性设置为 ENABLED。 要禁用队列管理器事件，请将相应的队列管理器属性设置为 DISABLED。

要启用或禁用队列管理器事件，请使用 MQSC 命令 **ALTER QMGR** 并指定相应的队列管理器属性。 第 18 页的表 1 概述了如何启用队列管理器事件。 要禁用队列管理器事件，请将相应的参数设置为 DISABLED。

表 *1:* 使用 *MQSC* 命令启用队列管理器事件

事件	**ALTER QMGR 参数**
权限 禁止 本地 远程 启动和停止	AUTHOREV (已启用) 抑制 (已启用) 本地 EV (已启用) REMOTEEV (已启用) STRSTPEV (已启用)

控制通道和网桥事件

您可以使用队列管理器属性来控制通道事件。 要启用通道事件，请将相应的队列管理器属性设置为 ENABLED。 要禁用通道事件，请将相应的队列管理器属性设置为 DISABLED。

要启用或禁用通道事件，请使用 MQSC 命令 **ALTER QMGR**，指定相应的队列管理器属性。 第 18 页的表 2 概述了如何启用通道和网桥事件。 要禁用队列管理器事件，请将相应的参数设置为 DISABLED。

限制: z/OS 通道自动定义事件在 IBM MQ for z/OS 上不可用。

表 *2:* 使用 *MQSC* 命令启用通道和网桥事件

事件	**ALTER QMGR 参数**
通道 仅与通道错误相关 IMS 网桥 SSL 通道自动定义	CHLEV (已启用) CHLEV (异常) BRIDGEEV (已启用) SSLEV (已启用) Chadev (已启用)

将 CHLEV 设置为异常时，将生成以下返回码和相应的原因限定符:

- MQRC_CHANNEL_ACTIVATED
- MQRC_CHANNEL_CONV_ERROR
- MQRC_CHANNEL_NOT_ACTIVATED
- MQRC_CHANNEL_STOPPED
  - 使用以下 ReasonQualifiers:
    - MQRQ_CHANNEL_STOPPED_ERROR
    - MQRQ_CHANNEL_STOPPED_RETRY

- MQRQ_CHANNEL_STOPPED_DISABLED

- MQRC_CHANNEL_STOPPED_BY_USER
- MQRC_CHANNEL_BLOCKED
  - 使用以下 ReasonQualifiers:
    - MQRQ_CHANNEL_BLOCKED_NOACCESS
    - MQRQ_CHANNEL_BLOCKED_USERID
    - MQRQ_CHANNEL_BLOCKED_ADDRESS

控制性能事件
您可以使用 PERFMEV 队列管理器属性来控制性能事件。 要启用性能事件，请将 PERFMEV 设置为 ENABLED。 要禁用性能事件，请将 PERFMEV 队列管理器属性设置为 DISABLED。

要将 PERFMEV 队列管理器属性设置为 ENABLED，请使用以下 MQSC 命令:

```
ALTER QMGR PERFMEV (ENABLED)
```

要启用特定性能事件，请设置相应的队列属性。 另外，请指定导致事件的条件。

队列深度事件
缺省情况下，将禁用所有队列深度事件。 要为任何队列深度事件配置队列，请执行以下操作:

1. 在队列管理器上启用性能事件。
2. 在所需队列上启用事件。
3. 如果需要，请将限制设置为相应的级别，以最大队列深度的百分比表示。

队列服务时间间隔事件
要为队列服务时间间隔事件配置队列，必须执行以下操作:

1. 在队列管理器上启用性能事件。
2. 根据需要在队列上设置 "队列服务时间间隔高" 或 "正常" 事件的控制属性。
3. 通过将队列的 QSVCINT 属性设置为适当的时间长度来指定服务时间间隔。

注: 启用后，可以在任何适当时间生成队列服务时间间隔事件，而不必等待发出队列的 MQI 调用。 但是，如果在队列上使用 MQI 调用来放置或除去消息，那么此时将生成任何适用的性能事件。 当耗用时间变为等于服务时间间隔时间时，不会生成事件。

控制配置，命令和记录器事件
您可以使用队列管理器属性 CONFIGEV， CMDEV 和 LOGGEREV 来控制配置，命令和记录器事件。 要启用这些事件，请将相应的队列管理器属性设置为 ENABLED。 要禁用这些事件，请将相应的队列管理器属性设置为 DISABLED。

配置事件
要启用配置事件，请将 CONFIGEV 设置为 ENABLED。 要禁用配置事件，请将 CONFIGEV 设置为 DISABLED。 例如，可以使用以下 MQSC 命令来启用配置事件:

```
ALTER QMGR CONFIGEV (ENABLED)
```

命令事件
要启用命令事件，请将 CMDEV 设置为 ENABLED。 要对除 DISPLAY MQSC 命令和 Inquire PCF 命令以外的命令启用命令事件，请将 CMDEV 设置为 NODISPLAY。 要禁用命令事件，请将 CMDEV 设置为 DISABLED。 例如，可以使用以下 MQSC 命令来启用命令事件:

```
ALTER QMGR CMDEV (ENABLED)
```

记录器事件

要启用记录器事件，请将 LOGGEREV 设置为 ENABLED。 要禁用记录器事件，请将 LOGGEREV 设置为 DISABLED。 例如，您可以使用以下 MQSC 命令来启用记录器事件:

```
ALTER QMGR LOGGEREV(ENABLED)
```

事件队列

发生事件时，队列管理器会将事件消息放在定义的事件队列上。 事件消息包含有关事件的信息。

您可以将事件队列定义为:

- 本地队列
- 别名队列
- 远程队列的本地定义，或作为
- 远程集群队列

如果将所有事件队列定义为一个队列管理器上同一远程队列的本地定义，那么可以集中监视活动。

不能将事件队列定义为传输队列，因为事件消息的格式与传输队列所需的消息格式不兼容。

共享事件队列是使用 QSGDISP (SHARED) 值定义的本地队列。

有关在 z/OS 上定义共享队列的更多信息，请参阅 使用共享队列进行应用程序编程。

当事件队列不可用时

如果在事件队列不可用时发生事件，那么事件消息将丢失。 例如，如果没有为某个事件类别定义事件队列，那么该类别的所有事件消息都将丢失。 例如，事件消息不会保存在死信 (undelivered-message) 队列上。

但是，您可以将事件队列定义为远程队列。 然后，如果远程系统上存在将消息放入已解析队列的问题，那么事件消息将到达远程系统的死信队列。

由于许多不同的原因，事件队列可能不可用，包括:

- 尚未定义队列。
- 已删除此队列。
- 队列已满。
- 已禁止放入队列。

缺少事件队列不会阻止事件发生。 例如，在性能事件之后，队列管理器会更改队列属性并重置队列统计信息。 无论是否将事件消息放入性能事件队列中，都会发生此更改。 在配置和命令事件的情况下也是如此。

使用触发的事件队列

您可以使用触发器来设置事件队列，以便在生成事件时，要放入事件队列中的事件消息将启动用户编写的监视应用程序。 此应用程序可以处理事件消息并执行相应的操作。 例如，某些事件可能需要通知操作员，其他事件可能启动自动执行某些管理任务的应用程序。

事件队列可以具有与其关联的触发器操作，并且可以创建触发器消息。 但是，如果这些触发器消息反过来导致通常会生成事件的条件，那么不会生成任何事件。 在此实例中不生成事件可确保不会发生循环。

相关概念

第 17 页的『控制事件』
根据事件类型，通过为队列管理器和/或队列属性指定相应的值来启用和禁用事件。

第 21 页的『事件消息的格式』
事件消息包含有关事件及其原因的信息。 与其他 IBM MQ 消息一样，事件消息具有两个部分: 消息描述符和消息数据。

使用共享队列的应用程序编程

触发器事件的条件

相关参考

QSGDisp (MQLONG)

事件消息的格式

事件消息包含有关事件及其原因的信息。 与其他 IBM MQ 消息一样，事件消息具有两个部分: 消息描述符和消息数据。

- 消息描述符基于 MQMD 结构。
- 消息数据由 事件头 和 事件数据组成。 事件头包含标识事件类型的原因码。 放置事件消息以及任何后续操作不会影响导致事件的 MQI 调用返回的原因码。 事件数据提供有关事件的更多信息。

通常，您使用定制的系统管理应用程序来处理事件消息，以满足其运行所在企业的需求。

当队列共享组中的队列管理器检测到生成事件消息的条件时，多个队列管理器可以为共享队列生成事件消息，从而生成多个事件消息。 为了确保系统可以关联来自不同队列管理器的多个事件消息，这些事件消息在消息描述符 (MQMD) 中设置了唯一的相关标识 *(CorrelId)* 。

相关参考

第 87 页的『活动报告 MQMD (消息描述符)』
使用此页面来查看活动报告的 MQMD 结构所包含的值

第 91 页的『活动报告 MQEPH (嵌入式 PCF 头)』
使用此页面来查看活动报告的 MQEPH 结构所包含的值

第 92 页的『活动报告 MQCFH (PCF 头)』
使用此页面来查看活动报告的 MQCFH 结构包含的 PCF 值

事件消息引用

事件消息格式

事件消息 MQMD (消息描述符)

事件消息 MQCFH (PCF 头)

事件消息描述

性能事件

性能事件与可能影响使用指定队列的应用程序的性能的条件相关。 性能事件的作用域是队列。 一个队列上的 **MQPUT** 调用和 **MQGET** 调用不会影响另一个队列上性能事件的生成。

可以在任何适当的时间生成性能事件消息，而不必等待发出队列的 MQI 调用。 但是，如果在队列上使用 MQI 调用来放置或除去消息，那么此时将生成任何相应的性能事件。

生成的每条性能事件消息都放置在队列 SYSTEM.ADMIN.PERFM.EVENT。

事件数据包含标识事件原因的原因码，一组性能事件统计信息和其他数据。 以下列表中描述了可以在性能事件消息中返回的事件数据类型:

- 队列深度过高
- 队列深度过低
- 队列已满
- 队列服务时间间隔过长
- 队列服务时间间隔正常

说明使用性能事件的示例假定您使用相应的 IBM MQ 命令 (MQSC) 来设置队列属性。 在 z/OS 上，您还可以使用队列管理器的操作和控制面板来设置队列属性。

相关参考

第 11 页的『事件类型』
使用此页面来查看队列管理器或通道实例可以报告的检测事件类型

性能事件统计信息

事件消息中的性能事件数据包含有关事件的统计信息。 使用统计信息来分析指定队列的行为。

事件消息中的事件数据包含有关系统管理程序的事件的信息。 对于所有性能事件，事件数据包含队列管理器的名称以及与事件关联的队列。 事件数据还包含与事件相关的统计信息。 第 22 页的表 3 汇总了可用于分析队列行为的事件统计信息。 所有统计信息都是指自上次重置统计信息以来发生的情况。

表 3: 性能事件统计信息

参数	描述
TimeSinceReset	自上次重置统计信息以来的耗用时间。
HighQDepth	自上次重置统计信息以来，队列中的最大消息数。
MsgEnqCount	自上次重置统计信息以来排队的消息数 (对队列的 MQPUT 调用数)。
MsgDeqCount	自上次重置统计信息以来，已出队的消息数 (对队列的 MQGET 调用数)。

发生以下任何更改时，将重置性能事件统计信息:

- 发生性能事件 (在所有活动队列管理器上重置统计信息)。
- 队列管理器停止并重新启动。
- 从应用程序发出 PCF 命令 "重置队列统计信息"。
- z/OS 仅在 z/OS 上，在控制台上发出 RESET QSTATS 命令。

相关概念

第 21 页的『性能事件』
性能事件与可能影响使用指定队列的应用程序的性能的条件相关。 性能事件的作用域是队列。 一个队列上的 **MQPUT** 调用和 **MQGET** 调用不会影响另一个队列上性能事件的生成。

第 23 页的『服务计时器』
队列服务时间间隔事件使用内部计时器 (称为 服务计时器)，该计时器由队列管理器控制。 仅当启用了队列服务时间间隔事件时，才会使用服务计时器。

第 24 页的『队列服务时间间隔事件的规则』
正式规则控制何时设置服务计时器并生成队列服务时间间隔事件。

相关任务

第 24 页的『启用队列服务时间间隔事件』
要为队列服务时间间隔事件配置队列，请设置相应的队列管理器和队列属性。

相关参考

队列深度过高
重置队列统计信息
重置 QSTATS

队列服务时间间隔事件

队列服务时间间隔事件指示是否在称为 服务时间间隔的用户定义的时间间隔内对队列执行了操作。 根据您的安装，您可以使用队列服务时间间隔事件来监视消息是否被足够快地从队列中取出。

共享队列上不支持队列服务时间间隔事件。

可能会发生以下类型的队列服务时间间隔事件，其中术语 *get operation* 指的是从队列中除去消息的 **MQGET** 调用或活动，例如使用 **CLEAR QLOCAL** 命令:

队列服务时间间隔正常

指示在执行下列其中一项操作之后:

- MQPUT 调用
- 保留非空队列的 get 操作

在用户定义的时间段 (称为 服务时间间隔) 内执行了获取操作。

只有 get 操作才能导致 "队列服务时间间隔正常" 事件消息。 "队列服务时间间隔正常" 事件有时被描述为 "正常" 事件。

队列服务时间间隔过长
指示在执行下列其中一项操作之后:

- MQPUT 调用
- 保留非空队列的 get 操作

未 在用户定义的服务时间间隔内执行获取操作。

get 操作或 MQPUT 调用可导致 "队列服务时间间隔高" 事件消息。 "队列服务时间间隔高" 事件有时被描述为 "高" 事件。

要同时启用 "队列服务时间间隔正常" 和 "队列服务时间间隔高" 事件，请将 `QServiceIntervalEvent` 控制属性设置为 "高"。 生成 "队列服务时间间隔高" 事件时，将自动启用 "队列服务时间间隔正常" 事件。 您不需要单独启用 "队列服务时间间隔正常" 事件。

"确定" 和 "高" 事件是互斥的，因此如果启用了一个事件，那么将禁用另一个事件。 但是，可以同时禁用这两个事件。

第 23 页的图 3 显示了队列深度与时间的图形。 在 P1 时，应用程序发出 MQPUT 以将消息放入队列。 在 G1 时，另一个应用程序发出 MQGET 以从队列中除去消息。

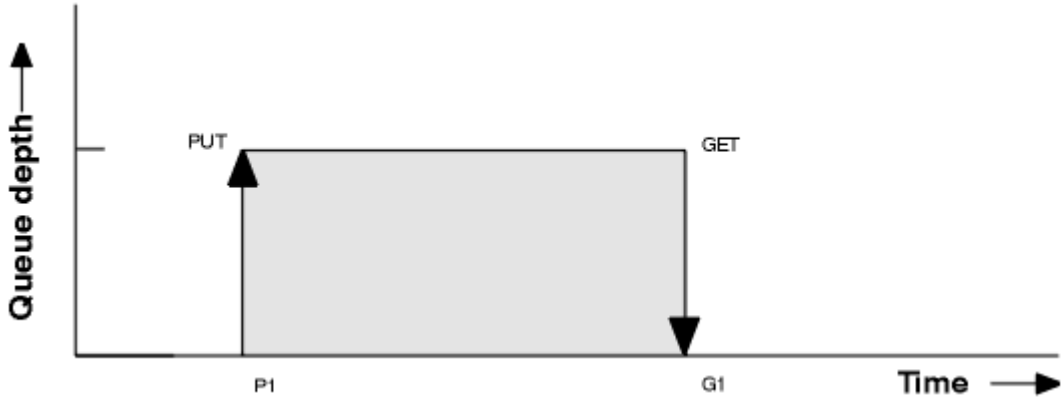


图 *3:* 了解队列服务时间间隔事件

队列服务时间间隔事件的可能结果如下:

- 如果 put 和 get 之间的耗用时间小于或等于服务时间间隔:
  - 如果启用了队列服务时间间隔事件，那么将在时间 G1 生成 队列服务时间间隔确定 事件
- 如果 put 和 get 之间的耗用时间大于服务时间间隔:
  - 如果启用了队列服务时间间隔事件，那么将在时间 G1 生成 队列服务时间间隔高 事件。

第 24 页的『队列服务时间间隔事件的规则』中描述了用于启动服务计时器和生成事件的算法。

相关参考
队列服务时间间隔正常
队列服务时间间隔过长
QServiceInterval 事件 (MQLONG)
QServiceInterval 事件 (10 位带符号整数)

服务计时器
队列服务时间间隔事件使用内部计时器 (称为 服务计时器) ，该计时器由队列管理器控制。 仅当启用了队列服务时间间隔事件时，才会使用服务计时器。

服务计时器的精确度量是什么?
服务计时器测量对空队列或 get 操作的 MQPUT 调用与下一个 put 或 get 操作之间的耗用时间，前提是这两个操作之间的队列深度非零。

服务计时器何时处于活动状态?
如果队列上有消息 (深度非零) 并且启用了队列服务时间间隔事件，那么服务计时器始终处于活动状态 (正在运行)。 如果队列变为空 (队列深度为零) ，那么计时器将进入 OFF 状态，以便在下一次放入时重新启动。

服务计时器何时重置?
在执行获取操作后，将始终重置服务计时器。 它还由对空队列的 MQPUT 调用重置。 但是，它不一定会在队列服务时间间隔事件上重置。

如何使用服务计时器?
在执行 get 操作或 MQPUT 调用之后，队列管理器将服务计时器测量的耗用时间与用户定义的服务时间间隔进行比较。 这种比较的结果是:

- 如果存在 get 操作并且耗用时间小于或等于服务时间间隔，那么将生成 OK 事件，并且将启用此事件。
- 如果耗用时间大于服务时间间隔，并且已启用此事件，那么将生成高事件。

应用程序可以读取服务计时器吗?
否，服务计时器是不可用于应用程序的内部计时器。

***TimeSinceReset* 参数如何?**
TimeSinceReset 参数作为事件数据中的事件统计信息的一部分返回。 它指定连续队列服务时间间隔事件之间的时间，除非重置事件统计信息。

队列服务时间间隔事件的规则
正式规则控制何时设置服务计时器并生成队列服务时间间隔事件。

服务计时器的规则

服务计时器重置为零并按如下所示重新启动:

- 在对空队列执行 MQPUT 调用之后。
- 在 MQGET 调用之后，如果队列在 MQGET 调用之后不为空。

计时器的重置不取决于是否已生成事件。

在队列管理器启动时，如果队列深度大于零，那么服务计时器将设置为启动时间。

如果执行 get 操作后队列为空，那么计时器将进入 OFF 状态。

队列服务时间间隔高事件数

必须启用 "队列服务时间间隔" 事件 (设置为 HIGH)。

生成 "队列服务时间间隔正常" 事件时，将自动启用 "队列服务时间间隔高" 事件。

如果服务时间大于服务时间间隔，那么将在下一个 MQPUT 或 get 操作上或之前生成事件。

队列服务时间间隔正常事件

生成 "队列服务时间间隔高" 事件时，将自动启用 "队列服务时间间隔正常" 事件。

如果服务时间 (耗用时间) 小于或等于服务时间间隔，那么将在下一个 get 操作上或之前生成事件。

相关任务
第 24 页的『启用队列服务时间间隔事件』
要为队列服务时间间隔事件配置队列，请设置相应的队列管理器和队列属性。

启用队列服务时间间隔事件
要为队列服务时间间隔事件配置队列，请设置相应的队列管理器和队列属性。

关于此任务

高事件和正常事件是互斥的; 即，启用一个事件时，会自动禁用另一个事件:

- 在队列上生成高事件时，队列管理器会自动禁用高事件，并为该队列启用 "确定" 事件。
- 在队列上生成 "确定" 事件时，队列管理器会自动禁用 "确定" 事件并为该队列启用高事件。

表 *4:* 使用 *MQSC* 启用队列服务时间间隔事件

队列服务时间间隔事件	**队列属性**
队列服务时间间隔过长 队列服务时间间隔正常 无队列服务时间间隔事件	Qsvciev (高) QSVCIEV (正常) QSVCIEV (无)
服务时间间隔	QSVCINT (*tt*) ，其中 *tt* 是服务 时间间隔 (以毫秒计)。

执行以下步骤以启用队列服务时间间隔事件:

过程

1. 将队列管理器属性 **PERFMEV** 设置为 ENABLED。
   在队列管理器上启用性能事件。
2. 根据需要，针对队列上的 "队列服务时间间隔高" 或 "确定" 事件设置控制属性 **QSVCIEV**。
3. 设置队列的 **QSVCINT** 属性以指定相应的服务时间间隔时间。

示例

要启用服务时间间隔为 10 秒 (10 000 毫秒) 的 "队列服务时间间隔高" 事件，请使用以下 MQSC 命令:

```
ALTER QMGR PERFMEV(ENABLED)
ALTER QLOCAL('MYQUEUE') QSVCINT(10000) QSVCIEV(HIGH)
```

.

队列服务时间间隔事件示例

使用本部分中的示例来了解可以从队列服务时间间隔事件获取的信息。

三个子主题示例提供了使用队列服务时间间隔事件的越来越复杂的插图。

每个子主题中的示例随附的图具有相同的结构:

- 图 1 是针对时间的队列深度图，显示各个 MQGET 调用和 MQPUT 调用。
- "注释" 部分显示时间约束的比较。 您必须考虑三个时间段:
  - 用户定义的服务时间间隔。
  - 服务计时器测量的时间。
  - 自上次重置事件统计信息以来的时间 (事件数据中的 TimeSince 重置)。
- "事件统计信息摘要" 部分显示在任何时刻启用的事件以及生成的事件。

这些示例说明队列服务时间间隔事件的以下方面:

- 队列深度随时间变化的方式。
- 服务计时器测量的耗用时间与服务时间间隔的比较。
- 已启用哪个事件。
- 生成哪些事件。

切记: 示例 1 显示了一个简单的情况，其中消息是间歇性的，并且在下一条消息到达之前从队列中除去了每条消息。 从事件数据中，您知道队列上的最大消息数为 1。 因此，您可以确定每条消息在队列中的时间长度。

但是，在一般情况下，如果队列中有多条消息，并且 MQGET 调用和 MQPUT 调用的序列不可预测，那么无法使用队列服务时间间隔事件来计算单个消息在队列中保留的时间长度。 在事件数据中返回的 TimeSinceReset 参数可以包含队列中没有消息的时间比例。 因此，将隐式地对从这些统计信息派生的任何结果进行平均值以包含这些时间。

相关概念

第 22 页的『队列服务时间间隔事件』
队列服务时间间隔事件指示是否在称为 服务时间间隔的用户定义的时间间隔内对队列执行了操作。 根据您的安装，您可以使用队列服务时间间隔事件来监视消息是否被足够快地从队列中取出。

第 23 页的『服务计时器』
队列服务时间间隔事件使用内部计时器 (称为 服务计时器)，该计时器由队列管理器控制。 仅当启用了队列服务时间间隔事件时，才会使用服务计时器。

队列服务时间间隔事件*: 示例 1*
MQGET 调用和 MQPUT 调用的基本序列，其中队列深度始终为 1 或 0。

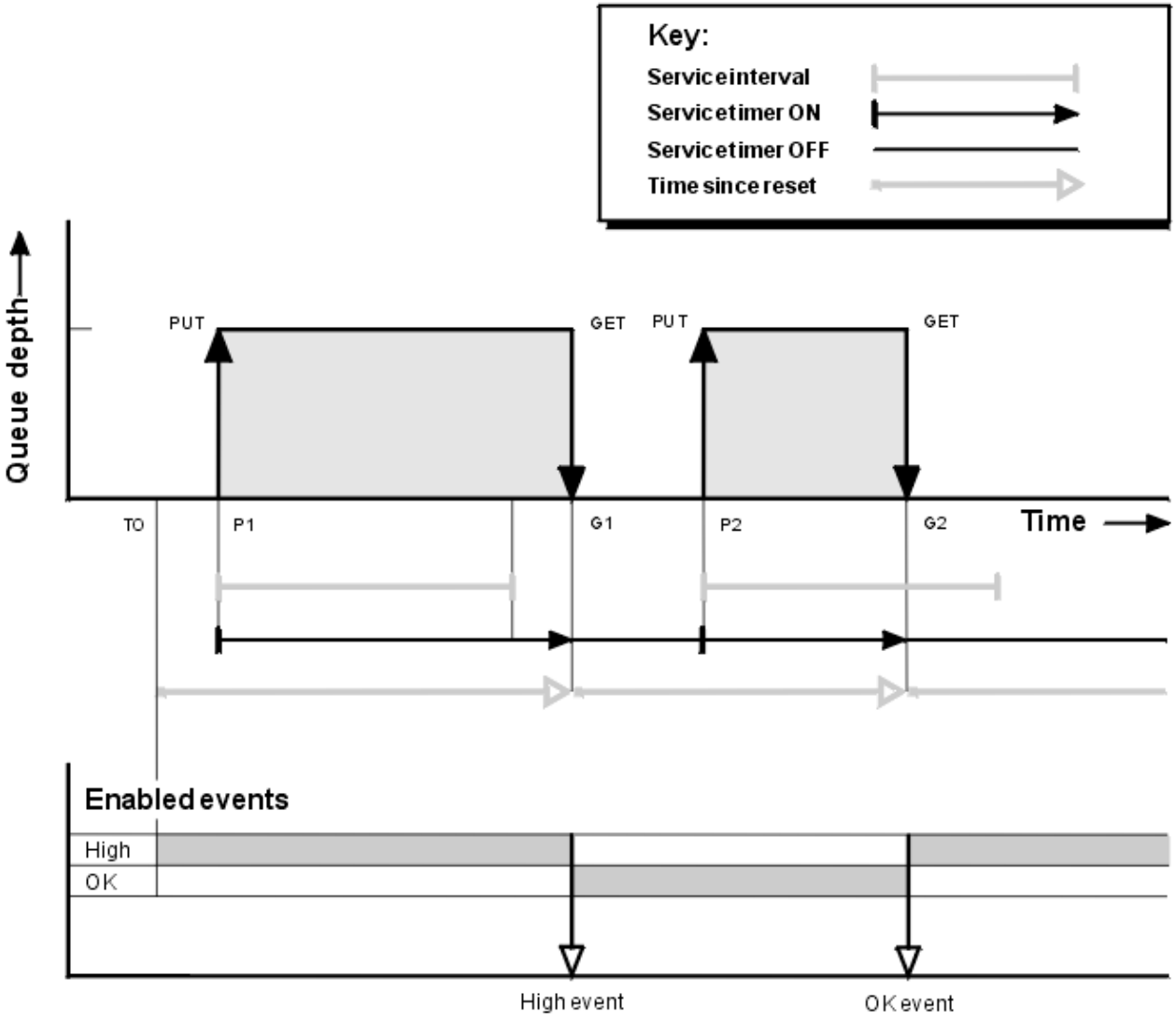


图 *4:* 队列服务时间间隔事件-示例 *1*

注释

1. 在 P1，应用程序将消息放入空队列中。 这将启动服务计时器。

   请注意， T0 可能是队列管理器启动时间。

2. 在 G1 上，另一个应用程序从队列中获取消息。 由于 P1 与 G1 之间的耗用时间大于服务时间间隔，因此将在 G1 上的 MQGET 调用上生成 "队列服务时间间隔高" 事件。 生成高事件时，队列管理器会重置事件控制属性，以便:

   a. 将自动启用 "确定" 事件。

   b. 已禁用高事件。

   由于队列现在为空，因此服务计时器切换为 OFF 状态。

3. 在 P2 处，将第二条消息放入队列中。 这将重新启动服务计时器。

4. 在 G2，将从队列中除去消息。 但是，由于 P2 与 G2 之间的耗用时间小于服务时间间隔，因此将在 G2 上的 MQGET 调用上生成 "队列服务时间间隔正常" 事件。 生成 OK 事件时，队列管理器会重置控制属性，以便:

   a. 将自动启用高事件。

   b. "确定" 事件已禁用。

   由于队列为空，因此服务计时器再次切换为 OFF 状态。

事件统计信息摘要

第 27 页的表 5 汇总了此示例的事件统计信息。

表 *5:* 事件统计信息摘要 *(*例如 *1)*

属性	**事件 1**	**事件 2**
事件时间	T (G1)	T (G2)
事件的类型	高	确定
TimeSinceReset	T (G1)-T (0)	T (G2)-T (G1)
HighQDepth	1	1
MsgEnqCount	1	1
MsgDeqCount	1	1

第 26 页的图 4 的中间部分显示了与该队列的服务时间间隔相比，由服务计时器测量的耗用时间。 要查看是否可能发生队列服务时间间隔事件，请将表示服务计时器 (带箭头) 的水平线的长度与表示服务时间间隔的线的长度进行比较。 如果服务计时器行较长，并且已启用 "队列服务时间间隔高" 事件，那么在下一次获取时将发生 "队列服务时间间隔高" 事件。 如果计时器行较短，并且已启用 "队列服务时间间隔确定" 事件，那么在下一次获取时将发生 "队列服务时间间隔确定" 事件。

队列服务时间间隔事件*:* 示例 *2*
MQPUT 调用和 MQGET 调用的序列，其中队列深度并非始终为 1 或 0。

此示例还显示在未生成事件 (例如，在时间 P2) 的情况下重置计时器的实例。

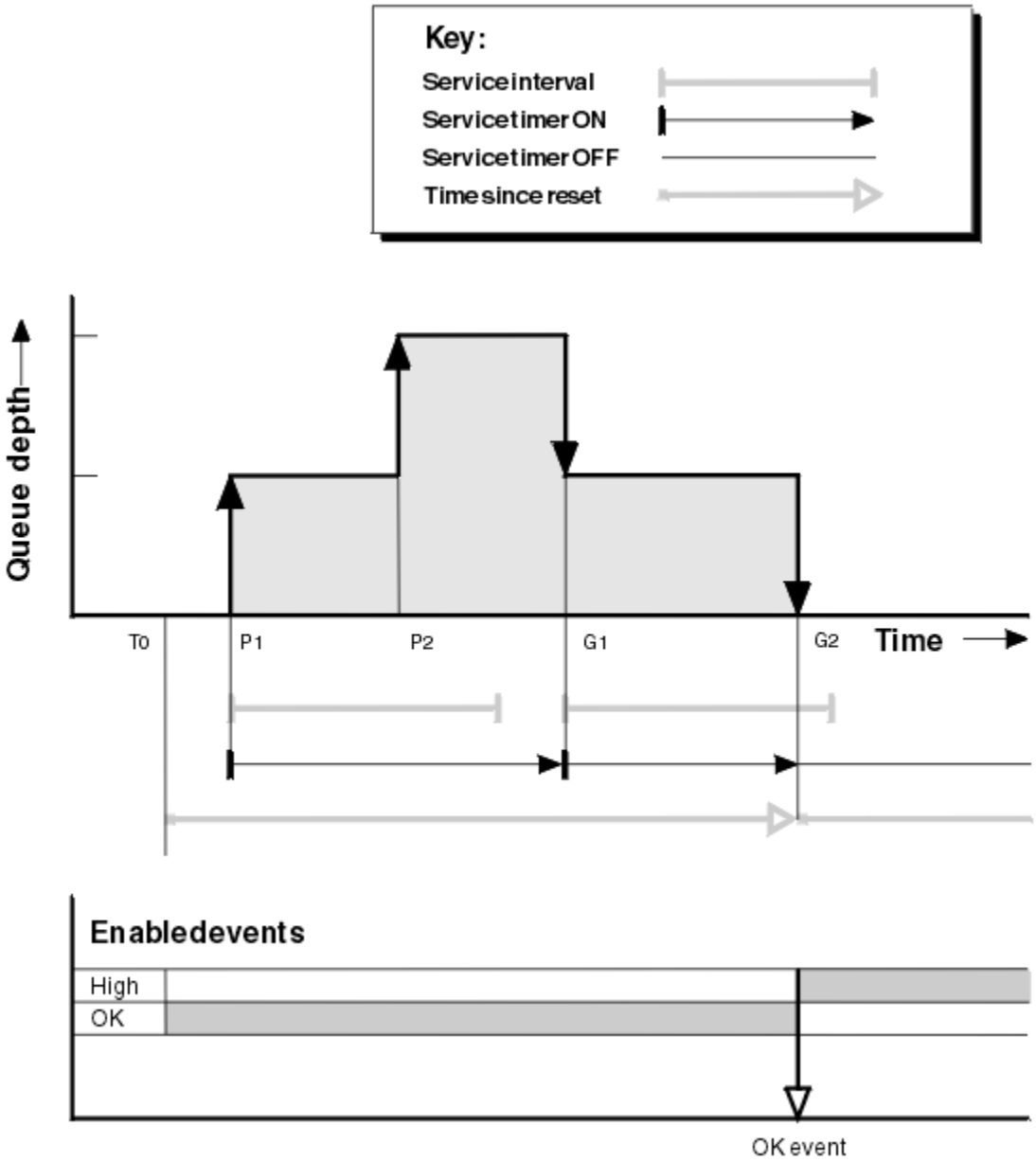


图 *5:* 队列服务时间间隔事件-示例 *2*

注释

在此示例中，最初启用了 OK 事件，并且在 T0 时重置了队列统计信息。

1. 在 P1 处，第一个放置会启动服务计时器。
2. 在 P2 处，第二个 put 不会生成事件，因为 put 无法导致 OK 事件。
3. 在 G1 上，现在已超过服务时间间隔，因此未生成 OK 事件。 但是， MQGET 调用会导致重置服务计时器。
4. 在 G2 上，第二次获取在服务时间间隔内发生，此时将生成 OK 事件。 队列管理器会重置事件控制属性，以便:

   a. 将自动启用高事件。

   b. "确定" 事件已禁用。

   由于队列现在为空，因此服务计时器切换为 OFF 状态。

事件统计信息摘要

第 29 页的表 6 汇总了此示例的事件统计信息。

表 *6:* 事件统计信息摘要 *(*例如 *2)*	
属性	**事件 2**
事件时间	T (G2)
事件的类型	确定
TimeSinceReset	T (G2)-T (0)
HighQDepth	2
MsgEnqCount	2
MsgDeqCount	2

队列服务时间间隔事件*:* 示例 *3*
MQGET 调用和 MQPUT 调用的序列，比先前示例更零星。

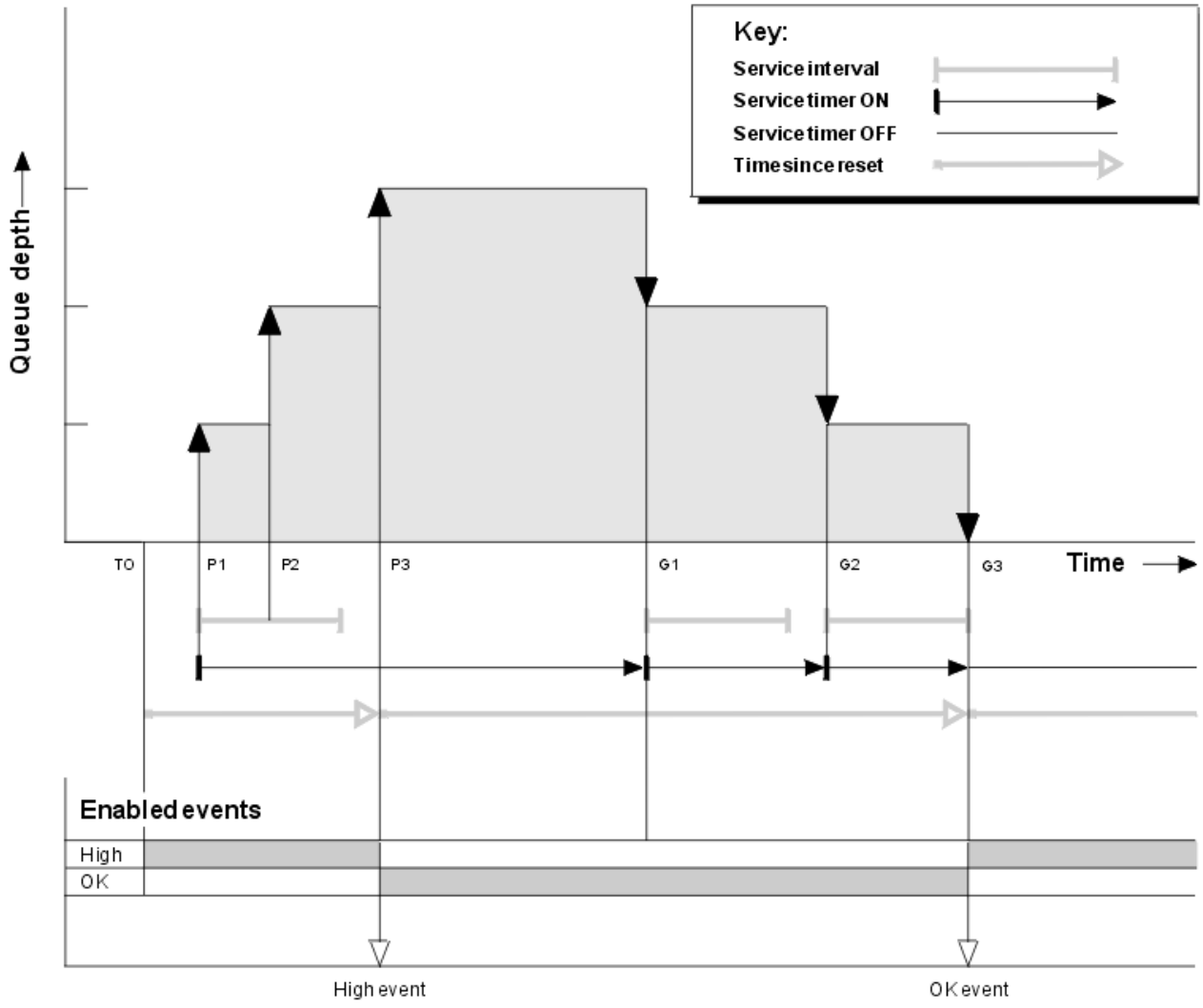


图 *6:* 队列服务时间间隔事件-示例 *3*

注释

1. 在 T (0) 时，将重置队列统计信息并启用 "队列服务时间间隔高" 事件。
2. 在 P1 处，第一个放置会启动服务计时器。
3. 在 P2，第二个放置会将队列深度增加到 2。 此处未生成高事件，因为未超过服务时间间隔时间。
4. 在 P3，第三个放置会导致生成高事件。 (计时器已超过服务时间间隔。) 未重置计时器，因为在放置之前队列深度不为零。 但是，已启用 "确定" 事件。
5. 在 G1 上， MQGET 调用不会生成事件，因为已超过服务时间间隔并且已启用 OK 事件。 但是， MQGET 调用会重置服务计时器。
6. 在 G2 上， MQGET 调用不会生成事件，因为已超过服务时间间隔并且已启用 OK 事件。 同样， MQGET 调用会重置服务计时器。
7. 在 G3 上，第三个获取将清空队列，并且服务计时器等于服务时间间隔。 因此，将生成 OK 事件。 服务计时器已重置，并且已启用高事件。 MQGET 调用会清空队列，这会使计时器处于 OFF 状态。

事件统计信息摘要

第 30 页的表 7 汇总了此示例的事件统计信息。

表 *7:* 事件统计信息摘要 *(*例如 *3)*

属性	**事件 1**	**事件 2**
事件时间	T (P3)	T (G3)
事件的类型	高	确定
TimeSinceReset	T (P3)-T (0)	T (G3)-T (P3)
HighQDepth	3	3
MsgEnqCount	3	0
MsgDeqCount	0	3

队列深度事件

队列深度事件与队列深度相关，即队列上的消息数。

在 IBM MQ 应用程序中，队列不得变满。 如果执行此操作，那么应用程序无法再将消息放入其指定的队列中。 虽然如果发生此情况，那么消息不会丢失，但完整队列可能会造成相当大的不便。 如果将消息放入队列的速度比处理它们的应用程序可以将它们关闭的速度更快，那么可以在队列上构建消息的数量。

此问题的解决方案取决于特定情况，但可能涉及:

- 将某些消息转移到另一个队列。
- 正在启动新应用程序以从队列中获取更多消息。
- 正在停止非必要消息流量。
- 增大队列深度以克服瞬态最大值。

提前警告问题可能正在发生，这样可以更轻松地采取预防行动。 为此， IBM MQ 提供了以下队列深度事件:

高队列深度事件
指示队列深度已增加到称为 "队列深度上限" 的预定义阈值。

低队列深度事件
指示队列深度已降低到称为 "队列深度下限" 的预定义阈值。

队列满事件
指示队列已达到其最大深度，即队列已满。

当应用程序尝试将消息放入已达到其最大深度的队列时，将生成 "队列已满事件"。 "队列深度高" 事件发出预先警告，指示队列正在填满。 这意味着在收到此事件后，系统管理员需要执行一些预防性操作。 您可以配置队列管理器，以便在预防性操作成功并且队列深度下降到更安全的级别时，队列管理器会生成 "队列深度低" 事件。

第一个队列深度事件示例说明了假定操作阻止队列变满的效果。

相关概念

第 33 页的『队列深度事件示例』
使用这些示例来了解可以从队列深度事件获取的信息

相关参考

队列已满
队列深度过高
队列深度过低

启用队列深度事件

要为任何队列深度事件配置队列，请设置相应的队列管理器和队列属性。

关于此任务

缺省情况下，将禁用所有队列深度事件。 启用后，将按如下所示生成队列深度事件:

- 将消息放入队列时，将生成 "队列深度过高" 事件，这将导致队列深度大于或等于为 **QDepthHighLimit** 设置的值。
  - 同一队列上的 "队列深度低" 事件会自动启用 "队列深度高" 事件。
  - "队列深度高" 事件会自动在同一队列上同时启用 "队列深度低" 和 "队列已满" 事件。
- Multi 当 GET 操作从队列中除去消息时，将生成 "队列深度下限" 事件，这将导致队列深度小于或等于为 **QDepthLowLimit** 设置的值。

  z/OS 当 GET 操作从队列中除去消息时，会生成 "队列深度下限" 事件，或者如果消息已被除去但已到期，那么会导致队列深度小于或等于为 **QDepthLowLimit** 设置的值。
  - "队列深度低" 事件由同一队列上的 "队列深度高" 事件或 "队列已满" 事件自动启用。
  - "队列深度低" 事件会自动在同一队列上同时启用 "队列深度高" 和 "队列已满" 事件。
- 当应用程序由于队列已满而无法将消息放入队列时，将生成 "队列已满" 事件。
  - 同一队列上的 "队列深度高" 或 "队列深度低" 事件会自动启用 "队列已满" 事件。
  - "队列已满" 事件会自动在同一队列上启用 "队列深度下限" 事件。

执行以下步骤为任何队列深度事件配置队列:

过程

1. 使用队列管理器属性 **PERFMEV** 在队列管理器上启用性能事件。
   这些事件将转至 SYSTEM.ADMIN.PERFM.EVENT 队列。
2. 设置下列其中一个属性以在所需队列上启用事件:
   - *QDepthHighEvent* (MQSC 中的 **QDPHIEV**)
   - *QDepthLow* 事件 (MQSC 中的 **QDPLOEV**)
   - *QDepthMax* 事件 (MQSC 中的 **QDPMAXEV**)
3. 可选： 要设置限制，请指定以下属性 (占最大队列深度的百分比):
   - *QDepthHighLimit* (MQSC 中的 **QDEPTHHI**)
   - *QDepthLow* 限制 (MQSC 中的 **QDEPTHLO**)

   限制: QDEPTHHI 不得小于 **QDEPTHLO**。

   如果 **QDEPTHHI** 等于 **QDEPTHLO** ，那么每次队列深度通过任一方向的值时都会生成一条事件消息，因为当队列深度低于该值时将启用高阈值，而当深度高于该值时将启用低阈值。

结果

注:

Multi 当 GET 操作将到期消息从队列中除去时，不会生成 "队列深度下限" 事件，这将导致队列深度小于或等于为 **QDepthLowLimit** 设置的值。 IBM MQ 仅在成功 GET 操作期间生成队列深度低事件消息。 因此，从队列中除去到期消息时，不会生成队列深度较低的事件消息。 此外，从队列中除去这些到期消息后，不会重置 *QDepthHighEvent* 和 *QDepthLowEvent* 。

z/OS IBM MQ 将在成功执行破坏性 GET 操作期间生成队列深度较低的事件消息，或者在匹配消息未到期的情况下生成已成功执行的破坏性 GET 操作。 否则，在常规后台处理期间从队列中除去到期消息时，不会生成队列深度较低的事件消息。 此外，在常规后台处理期间从队列中除去到期消息后，不会重置 *QDepthHighEvent* 和 *QDepthLowEvent* 。 有关到期消息处理的更多信息，请参阅 在 IBM MQ for z/OS 上调整队列管理器。

示例

要在限制设置为 80% 的队列 MYQUEUE 上启用 "队列深度高" 事件，请使用以下 MQSC 命令:

```
ALTER QMGR PERFMEV(ENABLED)
ALTER QLOCAL('MYQUEUE') QDEPTHHI(80) QDPHIEV(ENABLED)
```

要在限制设置为 20% 的队列 MYQUEUE 上启用 "队列深度低" 事件，请使用以下 MQSC 命令:

```
ALTER QMGR PERFMEV(ENABLED)
ALTER QLOCAL('MYQUEUE') QDEPTHLO(20) QDPLOEV(ENABLED)
```

要在队列 MYQUEUE 上启用 "队列已满" 事件，请使用以下 MQSC 命令:

```
ALTER QMGR PERFMEV(ENABLED)
ALTER QLOCAL('MYQUEUE') QDPMAXEV(ENABLED)
```

z/OS *Shared queues and queue depth events on z/OS*

On IBM MQ for z/OS, event monitoring is more straightforward for an application that uses shared queues if all the queue managers in the queue sharing group have the same setting for the PERFMEV attribute.

When a queue depth event occurs on a shared queue, and the queue manager attribute **PERFMEV** is set to ENABLED, the queue managers in the queue sharing group produce an event message. If **PERFMEV** is set to DISABLED on some of the queue managers, event messages are not produced by those queue managers, making event monitoring from an application more difficult. For more straightforward monitoring, give each queue manager the same setting for the **PERFMEV** attribute.

This event message that each queue manager generates represents its individual usage of the shared queue. If a queue manager performs no activity on the shared queue, various values in the event message are null or zero. You can use null event messages as follows:

- Ensure that each active queue manager in a queue sharing group generates one event message
- Highlight cases of no activity on a shared queue for the queue manager that produced the event message

Coordinating queue manager

When a queue manager issues a queue depth event, it updates the shared queue object definition to toggle the active performance event attributes. For example, depending on the definition of the queue attributes, a Queue Depth High event enables a Queue Depth Low and a Queue Full event. After updating the shared queue object successfully, the queue manager that detected the performance event initially becomes the *coordinating queue manager*.

If enabled for performance events, the coordinating queue manager performs the following actions:

1. Issues an event message that captures all shared queue performance data it has gathered since the last time an event message was created, or since the queue statistics were last reset. The message descriptor (MQMD) of this message contains a unique correlation identifier (*CorrelId*) created by the coordinating queue manager.
2. Broadcasts to all other *active* queue managers in the same queue sharing group to request the production of an event message for the shared queue. The broadcast contains the correlation identifier created by the coordinating queue manager for the set of event messages.

Having received a request from the coordinating queue manager, if there is an active queue manager in the queue sharing group that is enabled for performance events , that active queue manager issues an event message for the shared queue. The event message that is issued contains information about all the operations performed by the receiving (active) queue manager since the last time an event message was created, or since the statistics were last reset. The message descriptor (MQMD) of this event message contains the unique correlation identifier (*CorrelId*) specified by the coordinating queue manager.

When performance events occur on a shared queue, *n* event messages are produced, where *n* is a number from 1 to the number of active queue managers in the queue sharing group. Each event message contains data that relates to the shared queue activity for the queue manager that generated the event message.

Differences between shared and nonshared queues

Enabling queue depth events on shared queues differs from enabling them on nonshared queues. A key difference is that events are enabled for shared queues even if **PERFMEV** is DISABLED on the queue manager. This is not the case for nonshared queues.

Consider the following example, which illustrates this difference:

- QM1 is a queue manager with *PerformanceEvent* (**PERFMEV** in MQSC) set to DISABLED.
- SQ1 is a shared queue with **QSGDISP** set to（SHARED） QLOCAL in MQSC.
- LQ1 is a nonshared queue with **QSGDISP** set to（QMGR） QLOCAL in MQSC.

Both queues have the following attributes set on their definitions:

- **QDPHIEV (ENABLED)**
- **QDPLOEV (DISABLED)**
- **QDPMAXEV (DISABLED)**

If messages are placed on both queues so that the depth meets or exceeds the **QDEPTHHI** threshold, the **QDPHIEV** value on SQ1 switches to DISABLED. Also, **QDPLOEV** and **QDPMAXEV** are switched to ENABLED. SQ1's attributes are automatically switched for each performance event at the time the event criteria are met.

In contrast the attributes for LQ1 remain unchanged until **PERFMEV** on the queue manager is ENABLED. This means that if, for example, the queue manager's **PERFMEV** attribute is ENABLED, DISABLED and then set to ENABLED again, the performance event settings on shared queues might not be consistent with those of nonshared queues, even though they might have initially been the same.

队列深度事件示例

使用这些示例来了解可以从队列深度事件获取的信息

第一个示例提供队列深度事件的基本说明。 第二个例子更为广泛，但原理与第一个例子相同。 这两个示例都使用相同的队列定义，如下所示:

队列 MYQUEUE1 的最大深度为 1000 条消息。 高队列深度限制为 80% ，低队列深度限制为 20%。 最初，已启用 "队列深度高" 事件，而其他队列深度事件已禁用。

用于配置此队列的 IBM MQ 命令 (MQSC) 为:

```
ALTER QMGR PERFMEV(ENABLED)
```

```
DEFINE QLOCAL('MYQUEUE1') MAXDEPTH(1000) QDPMAXEV(DISABLED) QDEPTHHI(80)
QDPHIEV(ENABLED) QDEPTHLO(20) QDPLOEV(DISABLED)
```

相关概念

第 30 页的『队列深度事件』
队列深度事件与队列深度相关，即队列上的消息数。

相关任务

第 31 页的『启用队列深度事件』
要为任何队列深度事件配置队列，请设置相应的队列管理器和队列属性。

相关参考

MQSC 命令

队列深度事件*: 示例 1*
队列深度事件的基本序列。

第 34 页的图 7 显示了队列深度随时间变化的情况。

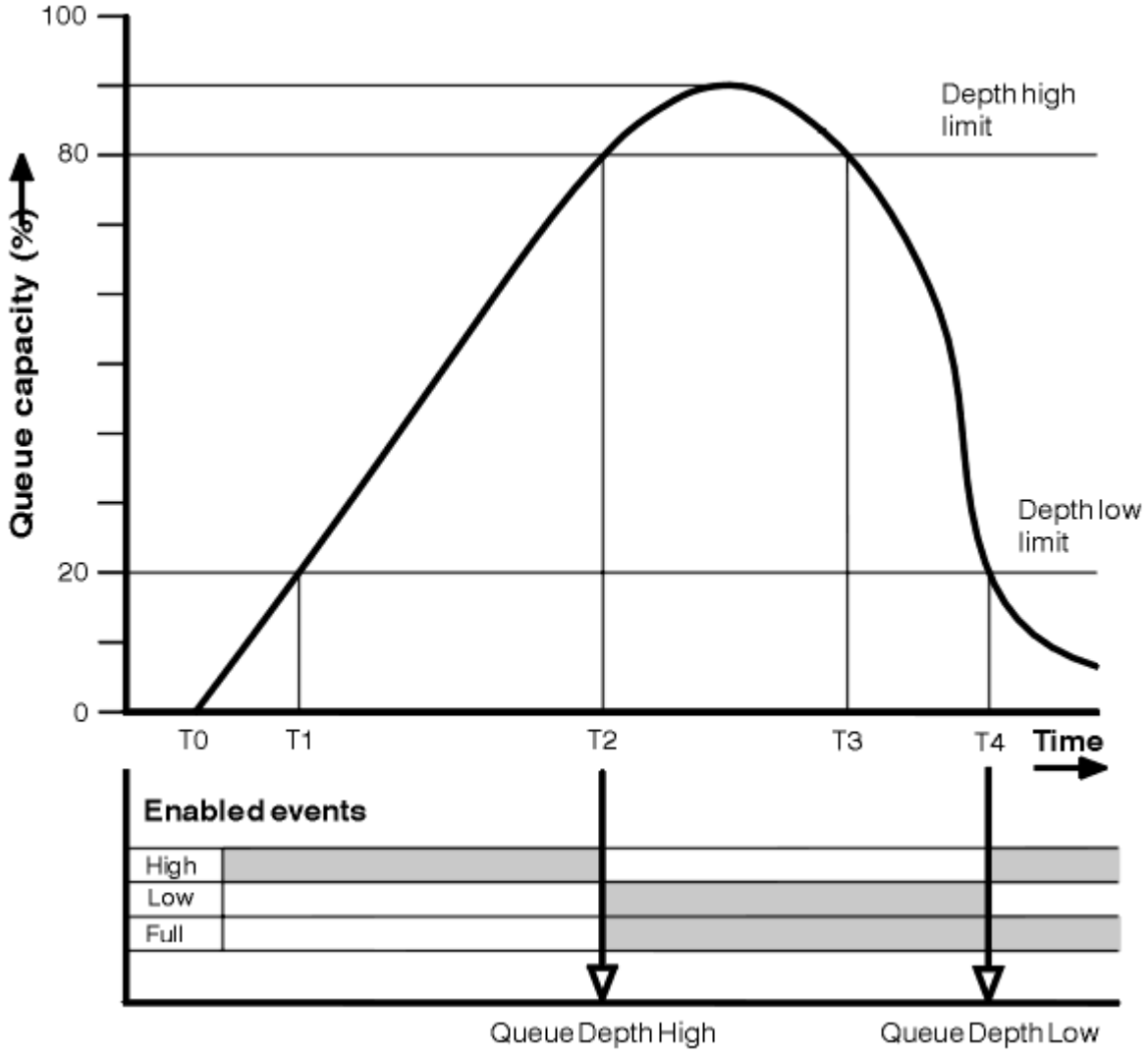


图 *7:* 队列深度事件 *(1)*

注释

1. 在 T (1) 处，队列深度正在增加 (MQPUT 调用数多于 MQGET 调用数)，并且超过了 "队列深度下限" 限制。 此时不会生成任何事件。

2. 当达到深度上限 (80%) 并生成 "队列深度上限" 事件时，队列深度将继续增大，直到 T (2) 为止。

   这将同时启用 "队列已满" 和 "队列深度低" 事件。
3. 事件发起的 (假定的) 预防性操作会阻止队列变满。 通过时间 T (3) ，再次达到队列深度上限，此时间已从以上时间开始。 此时不会生成任何事件。
4. 当队列深度达到深度下限 (20%) 并生成 "队列深度下限" 事件时，队列深度将继续下降，直到 T (4) 为止。

   这将同时启用 "队列已满" 和 "队列深度高" 事件。

事件统计信息摘要

第 35 页的表 8 汇总了队列事件统计信息， 第 35 页的表 9 汇总了已启用的事件。

表 *8:* 队列深度事件的事件统计信息摘要 *(示例 1)*

事件统计信息	**事件 2**	**事件 4**
事件时间	T (2)	T (4)
事件的类型	队列深度过高	队列深度过低
TimeSinceReset	T (2)-T (0)	T (4)-T (2)
HighQDepth (自重置以来的最大队列深度)	800	900
MsgEnqCount	1157	1220
MsgDeqCount	357	1820

表 *9:* 显示已启用哪些事件的摘要

时间段	**队列深度过高事件**	**队列深度下限事件**	**"队列已满" 事件**
在 T 之前 (1)	ENABLED	-	-
T (1) 到 T (2)	ENABLED	-	-
T (2) 到 T (3)	-	ENABLED	ENABLED
T (3) 到 T (4)	-	ENABLED	ENABLED
在 T 之后 (4)	ENABLED	-	ENABLED

队列深度事件*: 示例 2*
更广泛的队列深度事件序列。

第 36 页的图 8 显示了队列深度随时间变化的情况。

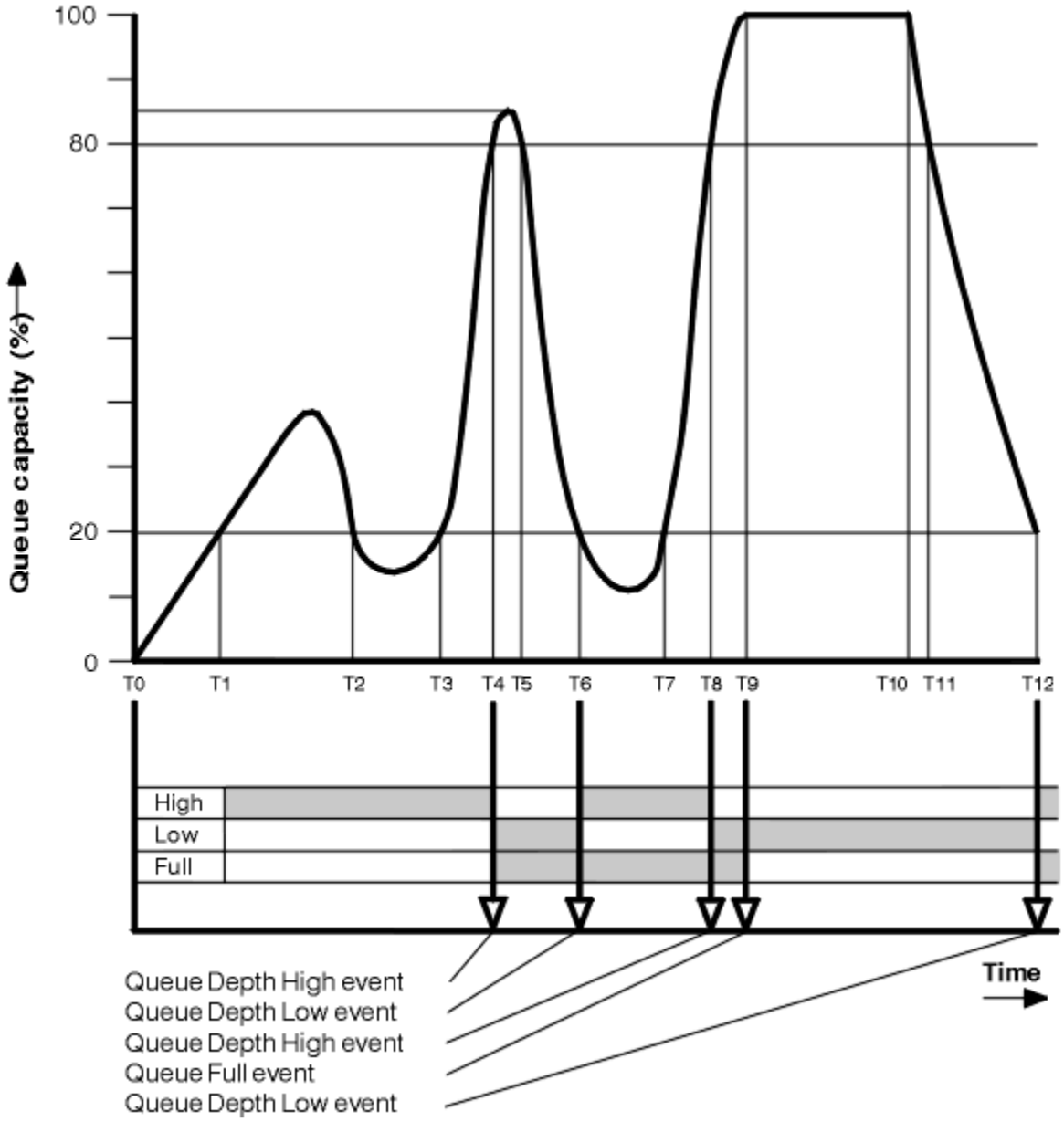


图 *8:* 队列深度事件 *(2)*

注释

1. 在以下时间不会生成 "队列深度下限" 事件:
   - T (1) (队列深度正在增大，但未启用)
   - T (2) (未启用)
   - T (3) (队列深度正在增大，但未启用)
2. 在 T (4) 处发生 "队列深度高" 事件。 这将同时启用 "队列已满" 和 "队列深度低" 事件。
3. 在 T (9) 处，由于队列已满而无法放入队列的第一条消息 **之后** 发生 "队列已满" 事件。
4. 在 T (12) 处发生 "队列深度低" 事件。

事件统计信息摘要

第 37 页的表 10 汇总了队列事件统计信息， 第 37 页的表 11 汇总了在不同时间为此示例启用的事件。

表 *10:* 队列深度事件的事件统计信息摘要 *(示例 2)*					
事件统计信息	**事件 4**	**事件 6**	**事件 8**	**事件 9**	**活动 12**
事件时间	T (4)	T (6)	T (8)	T (9)	T (12)
事件的类型	队列深度过高	队列深度过低	队列深度过高	队列已满	队列深度过低
TimeSinceReset	T (4)-T (0)	T (6)-T (4)	T (8)-T (6)	T (9)-T (8)	T (12)-T (9)
HighQDepth	800	855	800	1000	1000
MsgEnqCount	1645	311	1377	324	221
MsgDeqCount	845	911	777	124	1021

表 *11:* 显示已启用哪些事件的摘要			
时间段	**队列深度过高事件**	**队列深度下限事件**	**"队列已满" 事件**
T (0) 到 T (4)	ENABLED	-	-
T (4) 到 T (6)	-	ENABLED	ENABLED
T (6) 到 T (8)	ENABLED	-	ENABLED
T (8) 到 T (9)	-	ENABLED	ENABLED
T (9) 到 T (12)	-	ENABLED	-
在 T (12) 之后	ENABLED	-	ENABLED

注: 事件超出同步点。 因此，您可能有一个空队列，然后填充该队列会导致事件，然后在同步点管理器的控制下回滚所有消息。 但是，已自动设置事件启用，因此下次队列填满时不会生成任何事件。

配置事件

配置事件是在创建，更改或删除对象时生成的通知，也可以由显式请求生成。

请参阅 第 19 页的『控制配置，命令和记录器事件』，以获取有关如何开启事件的信息。

配置事件会通知您对对象属性的更改。 有四种类型的配置事件:

- 创建对象事件
- 更改对象事件
- 删除对象事件
- 刷新对象事件

事件数据包含以下信息:

源信息
包括从其中进行更改的队列管理器，进行更改的用户的标识以及更改的产生方式，例如通过控制台命令。

上下文信息
来自命令消息的消息数据中的上下文信息的副本。

仅当在 SYSTEM.COMMAND.INPUT 队列。

对象标识
包括对象的名称，类型和处置。

对象属性
包含对象中所有属性的值。

该事件是 PCF 格式的消息。 有关更多信息，请参阅:

- 更改对象
- 创建对象
- 删除对象
- 刷新对象

在更改对象事件的情况下，将生成 "前" 消息和 "后" 消息。 它们具有以下公共字段，以便您可以识别匹配的消息:

- 同一相关标识
- 在 PCF 头中:
  - 前消息头具有 `MsgSeqNumber': 1, 'Control': 'NOT_LAST'`
  - 后消息头具有 `MsgSeqNumber': 2, 'Control': 'LAST'`

生成的每条配置事件消息都放置在队列 SYSTEM.ADMIN.CONFIG.EVENT。

相关概念

第 15 页的『配置事件』
配置事件在显式请求配置事件时生成，或者在创建，修改或删除对象时自动生成。

相关参考

创建对象
更改对象
删除对象
刷新对象
第 11 页的『事件类型』
使用此页面来查看队列管理器或通道实例可以报告的检测事件类型

配置事件生成

使用此页面来查看导致生成配置事件的命令，并了解未生成配置事件的情况

当 CONFIGEV 队列管理器属性为 ENABLED 且

- 将发出下列任何命令或其 PCF 等效命令:
  - 删除授权信息
  - 删除 CFSTRUCT
  - 删除通道
  - 删除名称列表
  - 删除进程
  - 删除 QMODEL/QALIAS/QREMOTE
  - 删除 STGCLASS
  - 删除主题
  - 刷新队列管理器
- 即使对该对象没有任何更改，也会发出下列任何命令或它们的 PCF 等效命令:
  - 定义/变更 AUTHINFO
  - 定义/变更 CFSTRUCT
  - 定义/变更通道
  - DEFINE/ALTER NAMELIST
  - 定义/变更过程
  - DEFINE/ALTER QMODEL/QALIAS/QREMOTE
  - DEFINE/ALTER STGCLASS
  - 定义/变更 TOPIC

  - 定义 MAXSMSGS
  - SET CHLAUTH
  - ALTER QMGR，除非 CONFIGEV 属性为 DISABLED 并且未更改为 ENABLED
- 对于非临时动态的本地队列，将发出下列任何命令或其 PCF 等效命令，即使该队列没有更改也是如此。
  - 删除 QLOCAL
  - DEFINE/ALTER QLOCAL
- 将发出 MQSET 调用 (对于临时动态队列除外)，即使对象没有更改也是如此。

未生成配置事件时

在以下情况下，不会生成配置事件消息:

- 当命令或 MQSET 调用失败时
- 当队列管理器在尝试将配置事件放入事件队列时迁到错误，在这种情况下，命令或 MQSET 调用完成，但不会生成事件消息
- 对于临时动态队列
- 对 TRIGGER 队列属性进行内部更改时
- 对于配置事件队列 SYSTEM.ADMIN.CONFIG.EVENT， REFRESH QMGR 命令除外
- 对于导致集群更改的 REFRESH/RESET CLUSTER 和 RESUME/SUSPEND QMGR 命令
- 创建或删除队列管理器时

相关概念

可编程命令格式简介

第 37 页的『配置事件』

配置事件是在创建，更改或删除对象时生成的通知，也可以由显式请求生成。

相关参考

MQSC 命令

MQSET-设置对象属性

MQSET-设置对象属性

配置事件使用情况

使用此页面可查看如何使用配置事件来获取有关系统的信息，以及了解可能影响配置事件使用的因素 (例如 CMDSCOPE)。

您可以将配置事件用于以下目的:

1. 用于生成和维护中央配置存储库，可以从中生成报告并生成有关系统结构的信息。
2. 生成审计跟踪。 例如，如果对象意外更改，那么可以存储有关谁进行了更改以及何时进行了更改的信息。

   当还启用了命令事件时，这可能特别有用。 如果 MQSC 或 PCF 命令导致生成配置事件和命令事件，那么这两个事件消息将在其消息描述符中共享相同的相关标识。

对于 MQSET 调用或以下任何命令:

- DEFINE 对象
- ALTER 对象
- 删除对象

如果已启用队列管理器属性 CONFIGEV，但无法将配置事件消息放在配置事件队列上，例如未定义事件队列，那么将执行命令或 MQSET 调用而不考虑。

CMDSCOPE 的影响

对于使用 CMDSCOPE 的命令，将在执行命令的队列管理器上生成配置事件消息，而不是在输入命令的位置生成配置事件消息。 但是，事件数据中的所有源和上下文信息都将与输入的原始命令相关，即使使用 CMDSCOPE 的命令是源队列管理器生成的命令也是如此。

如果队列共享组包含非当前版本的队列管理器，那么将针对通过 CMDSCOPE 在当前版本的队列管理器 (而不是先前版本的队列管理器) 上执行的任何命令生成事件。 即使输入该命令的队列管理器处于先前版本，也会发生此情况，尽管在这种情况下，事件数据中不包含任何上下文信息。

相关概念

可编程命令格式简介

第 37 页的『配置事件』

配置事件是在创建，更改或删除对象时生成的通知，也可以由显式请求生成。

相关参考

MQSET-设置对象属性

MQSET-设置对象属性

刷新对象配置事件

"刷新对象" 配置事件与其他配置事件不同，因为仅当显式请求时才会发生此事件。

创建，更改和删除事件由 MQSET 调用或用于更改对象的命令生成，但仅当 MQSC 命令， REFRESH QMGR 或其 PCF 等效项明确请求时，才会发生刷新对象事件。

REFRESH QMGR 命令与生成配置事件的所有其他命令不同。 所有其他命令都适用于特定对象，并为该对象生成单个配置事件。 REFRESH QMGR 命令可以生成许多可能表示队列管理器所存储的每个对象定义的配置事件消息。 将为所选的每个对象生成一条事件消息。

REFRESH QMGR 命令使用三个选择标准的组合来过滤所涉及的对象数:

- 对象名称
- 对象类型
- 刷新时间间隔

如果在 REFRESH QMGR 命令上未指定任何选择条件，那么将对每个选择条件使用缺省值，并且将为队列管理器存储的每个对象定义生成刷新配置事件消息。 这可能会导致不可接受的处理时间和事件消息生成。 请考虑指定一些选择标准。

可在以下情况下使用生成刷新事件的 REFRESH QMGR 命令:

- 当需要有关系统中所有对象或某些对象的配置数据时，无论最近是否已处理这些对象，例如，首次启用配置事件时。

  请考虑使用多个命令，每个命令都具有不同的对象选择，但所有命令都包含在内。
- 如果 SYSTEM.ADMIN.CONFIG.EVENT 队列。 在此情况下，不会为 "创建" ， "更改" 或 "删除" 事件生成配置事件消息。 更正队列上的错误后，可以使用 "刷新队列管理器" 命令来请求生成事件消息，这些消息在队列中发生错误时已丢失。 在这种情况下，请考虑将刷新时间间隔设置为队列不可用的时间。

相关概念

第 37 页的『配置事件』

配置事件是在创建，更改或删除对象时生成的通知，也可以由显式请求生成。

相关参考

刷新队列管理器

刷新队列管理器

命令事件

命令事件是 MQSC 或 PCF 命令已成功运行的通知。

事件数据包含以下信息:

源信息
包括从其中发出命令的队列管理器，发出命令的用户的标识以及发出命令的方式 (例如控制台命令)。

上下文信息
来自命令消息的消息数据中的上下文信息的副本。 如果未使用消息输入命令，那么将省略上下文信息。

仅当在 SYSTEM.COMMAND.INPUT 队列。

命令信息
发出的命令的类型。

命令数据

- 对于 PCF 命令，这是命令数据的副本
- 对于 MQSC 命令，命令文本

命令数据格式不一定与原始命令的格式匹配。 例如，在 多平台 上，命令数据格式始终为 PCF 格式，即使原始请求是 MQSC 命令也是如此。

生成的每条命令事件消息都放置在命令事件队列 SYSTEM.ADMIN.COMMAND.EVENT。

相关参考
命令
第 11 页的『事件类型』
使用此页面来查看队列管理器或通道实例可以报告的检测事件类型

命令事件生成
使用此页面来查看导致生成命令事件的情境，并了解未生成命令事件的情境

在下列情况下，将生成命令事件消息:

- 当 CMDEV 队列管理器属性指定为 ENABLED 并且 MQSC 或 PCF 命令成功运行时。
- 当 CMDEV 队列管理器属性指定为 NODISPLAY 并且成功运行任何命令 (DISPLAY 命令 (MQSC) 和 Inquire 命令 (PCF) 除外) 时。
- 运行 MQSC 命令， ALTER QMGR 或 PCF 命令时， "更改队列管理器" 和 CMDEV 队列管理器属性满足以下任一条件:
  - 在更改后，未将 CMDEV 指定为 DISABLED
  - 在更改之前，未将 CMDEV 指定为 DISABLED

如果针对命令事件队列 SYSTEM.ADMIN.COMMAND.EVENT，如果队列仍然存在并且未禁止放入，那么将生成命令事件。

未生成命令事件时

在以下情况下，不会生成命令事件消息:

- 当命令失败时
- 当队列管理器在尝试将命令事件放在事件队列上时迂到错误，在这种情况下，无论命令是否运行，都不会生成事件消息
- 对于 MQSC 命令 REFRESH QMGR TYPE (早期)
- 对于 MQSC 命令 START QMGR MQSC
- 对于 MQSC 命令 SUSPEND QMGR , 如果指定了参数 LOG
- 对于 MQSC 命令 RESUME QMGR , 如果指定了参数 LOG

相关概念
第 40 页的『命令事件』
命令事件是 MQSC 或 PCF 命令已成功运行的通知。

相关参考
刷新队列管理器
开始 QMGR

已暂挂的队列管理器
恢复队列管理器
SUSPEND QMGR， RESUME QMGR 和集群

命令事件用法

使用此页面来查看如何使用命令事件来生成已运行的命令的审计跟踪

例如，如果对象意外更改，那么可以存储有关谁进行了更改以及何时进行了更改的信息。 当同时启用配置事件时，这可能特别有用。 如果 MQSC 或 PCF 命令导致生成命令事件和配置事件，那么这两个事件消息将在其消息描述符中共享相同的相关标识。

如果生成了命令事件消息，但无法将其放在命令事件队列上，例如，如果尚未定义命令事件队列，那么生成命令事件的命令仍将运行，而不考虑任何情况。

CMDSCOPE 的影响

对于使用 CMDSCOPE 的命令，将在运行命令的一个或多个队列管理器上生成命令事件消息，而不是在输入命令的位置生成命令事件消息。 但是，事件数据中的所有源和上下文信息都将与输入的原始命令相关，即使使用 CMDSCOPE 的命令是源队列管理器生成的命令也是如此。

相关概念

第 40 页的『命令事件』
命令事件是 MQSC 或 PCF 命令已成功运行的通知。

第 41 页的『命令事件生成』
使用此页面来查看导致生成命令事件的情境，并了解未生成命令事件的情境

相关参考

MQSC 命令
组中的 PCF 命令和响应

Multi 记录器事件

记录器事件是队列管理器已开始写入新日志扩展数据块 IBM i 或在 IBM i 上写入日志接收器的通知。 z/OS 记录器事件消息不可用于 IBM MQ for z/OS。

事件数据包含以下信息:

- 当前日志扩展数据块的名称。
- 重新启动恢复所需的最早日志扩展数据块的名称。
- 介质恢复所需的最早日志扩展数据块的名称。
- 日志扩展数据块所在的目录。
- 需要归档通知的最早日志扩展数据块的名称。

  将生成记录器事件以进行归档日志管理，即，当 **ARCHLOG** 发生更改时，将包括记录器事件消息中的 **ARCHLOG** 值。

  请参阅 DISPLAY QMSTATUS 以获取有关所有这些参数的更多信息。

  您可以使用 **CURRLOG** 和 **ARCHLOG** 值来确定应该归档的内容。 每当 **CURRLOG** 更改时，都可以发送数字小于 **CURRLOG** 的文件进行归档，并且在针对扩展数据块完成归档后，应调用 SET LOG 以通知队列管理器。

  ARCHLOG 是需要归档的最旧扩展数据块。 当您已归档该扩展数据块并调用 SET LOG 以通知队列管理器已归档该扩展数据块时，队列管理器将按顺序将 **ARCHLOG** 移至下一个扩展数据块。 队列管理器调度要删除或复用的 **ARCHLOG** 之前的扩展数据块。

  在为此扩展数据块调用 **SET LOG** 后，将发出新事件，因为 **ARCHLOG** 将已更改，并且您需要知道在新事件之后需要归档的内容。

如果由于某种原因导致归档过程失败，并且发生了通知的大构建，那么管理员可以手动发出命令 RESET QMGR TYPE (**ARCHLOG**)。 这将通知队列管理器，它可以复用或删除所有早于并包括您指定的扩展数据块的扩展数据块。

生成的每个记录器事件消息都放置在记录器事件队列 SYSTEM.ADMIN.LOGGER.EVENT。

相关参考

记录器

第 11 页的『事件类型』

使用此页面来查看队列管理器或通道实例可以报告的检测事件类型

Multi 记录器事件生成

使用此页面来查看导致生成记录器事件的情境，并了解未生成记录器事件的环境

在以下情况下会生成记录器事件消息:

- 当 LOGGEREV 队列管理器属性指定为 ENABLED 时，队列管理器将开始写入新的日志扩展数据块或在 IBM i 上写入日志接收器。
- 当 LOGGEREV 队列管理器属性指定为 ENABLED 并且队列管理器启动时。
- 当 LOGGEREV 队列管理器属性从 DISABLED 更改为 ENABLED 时。
- 当通知 LOGGEREV 队列管理器属性已归档队列管理器正在等待归档通知 (ARCHLOG) 的最旧日志扩展数据块的名称时。

提示: 您可以使用 RESET QMGR MQSC 命令来请求队列管理器开始写入新的日志扩展数据块。

未生成记录器事件时

在以下情况下，不会生成记录器事件消息:

- 当队列管理器配置为使用循环日志记录时。

  在这种情况下， LOGGEREV 队列管理器属性设置为 DISABLED ，不能改变。
- 当队列管理器在尝试将记录器事件放入事件队列时迁到错误，在这种情况下，导致事件的操作完成，但不会生成事件消息。

相关概念

第 42 页的『记录器事件』

记录器事件是队列管理器已开始写入新日志扩展数据块 IBM i 或在 IBM i 上写入日志接收器的通知。

z/OS 记录器事件消息不可用于 IBM MQ for z/OS。

相关参考

LoggerEvent (MQLONG)

LoggerEvent (10 位带符号整数)

重置队列管理器

Multi 记录器事件用法

使用此页面可查看如何使用记录器事件来确定队列管理器重新启动或介质恢复不再需要的日志扩展数据块。

您可以将多余的日志扩展数据块归档到介质 (例如磁带) 以进行灾难恢复，然后再将它们从活动日志目录中除去。 定期除去多余的日志扩展数据块可使磁盘空间使用率降至最低。

如果启用了 LOGGEREV 队列管理器属性，但记录器事件消息无法放在记录器事件队列上 (例如，因为尚未定义事件队列) ，那么导致事件的操作将继续进行而不考虑。

相关概念

第 42 页的『记录器事件』

记录器事件是队列管理器已开始写入新日志扩展数据块 IBM i 或在 IBM i 上写入日志接收器的通知。

z/OS 记录器事件消息不可用于 IBM MQ for z/OS。

相关参考

LoggerEvent (MQLONG)

LoggerEvent (10 位带符号整数)

第 43 页的『记录器事件生成』

使用此页面来查看导致生成记录器事件的情境，并了解未生成记录器事件的环境

Multi 用于监视记录器事件队列的样本 C 程序

使用此页面来查看样本 C 程序，该程序监视记录器事件队列以获取新事件消息，读取这些消息，并将消息内容放入 stdout。

```
/********************************************************************************/
/*                                                                              */
/* Program name: AMQSLOG0.C                                                     */
/*                                                                              */
/* Description:  Sample C program to monitor the logger event queue and         */
/*               display formatted messsage content to stdout when a logger     */
/*               event occurs                                                   */
/*   <copyright                                                                 */
/*   notice="lm-source-program"                                                 */
/*   pids="5724-H72,"                                                           */
/*   years="2005, 2024"                                                         */
/*   crc="186943832" >                                                          */
/*   Licensed Materials - Property of IBM                                       */
/*                                                                              */
/*   5724-H72,                                                                  */
/*                                                                              */
/*   (C) Copyright IBM Corp. 2005, 2024. All Rights Reserved.                   */
/*                                                                              */
/*   US Government Users Restricted Rights - Use, duplication or                */
/*   disclosure restricted by GSA ADP Schedule Contract with                    */
/*   IBM Corp.                                                                  */
/*   </copyright>                                                               */
/********************************************************************************/
/*                                                                              */
/* Function: AMQSLOG is a sample program which monitors the logger event        */
/* queue for new event messages, reads those messages, and displays the         */
/* formatted contents of the message to stdout.                                 */
/*                                                                              */
/********************************************************************************/
/*                                                                              */
/* AMQSLOG has 1 parameter - the queue manager name (optional, if not           */
/* specified then the default queue manager is implied)                         */
/*                                                                              */
/********************************************************************************/

/********************************************************************************/
/* Includes                                                                     */
/********************************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include <cmqc.h>        /* MQI constants*/
#include <cmqcfc.h>      /* PCF constants*/

/********************************************************************************/
/* Constants                                                                    */
/********************************************************************************/

#define   MAX_MESSAGE_LENGTH   8000

typedef struct _ParmTableEntry
{
  MQLONG  ConstVal;
  PMQCHAR Desc;
} ParmTableEntry;

ParmTableEntry ParmTable[] =
{
  {0                              ,""},
  {MQCA_Q_MGR_NAME                ,"Queue Manager Name"},
  {MQCMD_LOGGER_EVENT             ,"Logger Event Command"},
  {MQRC_LOGGER_STATUS             ,"Logger Status"},
  {MQCACF_ARCHIVE_LOG_EXTENT_NAME,"Archive Log Extent"},
  {MQCACF_CURRENT_LOG_EXTENT_NAME,"Current Log Extent"},
```

```
  {MQCACF_RESTART_LOG_EXTENT_NAME,"Restart Log Extent"},
  {MQCACF_MEDIA_LOG_EXTENT_NAME  ,"Media Log Extent"},
  {MQCACF_LOG_PATH               ,"Log Path"}
};

#if defined(MQ_64_BIT)
  #define Int32
#else
  #define Int32 "l"
#endif

/********************************************************************************/
/* Function prototypes                                                          */
/********************************************************************************/

static void ProcessPCF(MQHCONN    hConn,
                       MQHOBJ     hEventQueue,
                       PMQCHAR    pBuffer);

static PMQCHAR ParmToString(MQLONG Parameter);

/********************************************************************************/
/* Function: main                                                               */
/********************************************************************************/

int main(int argc, char * argv[])
{
  MQLONG    CompCode;
  MQLONG    Reason;
  MQHCONN   hConn = MQHC_UNUSABLE_HCONN;
  MQOD      ObjDesc = { MQOD_DEFAULT };
  MQCHAR    QMName[MQ_Q_MGR_NAME_LENGTH+1]  = "";
  MQCHAR    LogEvQ[MQ_Q_NAME_LENGTH]        = "SYSTEM.ADMIN.LOGGER.EVENT";
  MQHOBJ    hEventQueue = MQHO_UNUSABLE_HOBJ;
  PMQCHAR   pBuffer = NULL;

  printf("\n/************************************/\n");
  printf("/* Sample Logger Event Monitor start */\n");
  printf("/************************************/\n");

  /********************************************************************/
  /* Parse any command line options                                   */
  /********************************************************************/
  if (argc > 1)
  {
    strncpy(QMName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
  }

  pBuffer = (PMQCHAR)malloc(MAX_MESSAGE_LENGTH);
  if (pBuffer == NULL)
  {
    printf("Can't allocate %d bytes\n", MAX_MESSAGE_LENGTH);
    goto MOD_EXIT;
  }

  /********************************************************************/
  /* Connect to the specified (or default) queue manager              */
  /********************************************************************/
  MQCONN( QMName,
          &hConn,
          &CompCode,
          &Reason);

  if (Reason != MQRC_NONE)
  {
    printf("MQCONN ended with reason code %" Int32 "d\n", Reason);
    goto MOD_EXIT;
  }

  /********************************************************************/
  /* Open the logger event queue for input                            */
  /********************************************************************/
  strncpy(ObjDesc.ObjectQMgrName, QMName, MQ_Q_MGR_NAME_LENGTH);
  strncpy(ObjDesc.ObjectName, LogEvQ, MQ_Q_NAME_LENGTH);

  MQOPEN( hConn,
          &ObjDesc,
           MQOO_INPUT_EXCLUSIVE,
          &hEventQueue,
          &CompCode,
          &Reason );
```

```
  if (Reason != MQRC_NONE)
  {
    printf("MQOPEN failed for queue manager %.48s Queue %.48s Reason: %" Int32 "d\n",
                   ObjDesc.ObjectQMgrName,
           ObjDesc.ObjectName,
           Reason);
    goto MOD_EXIT;
  }
  else
  {
    /*****************************************************************/
    /* Start processing event messages                               */
    /*****************************************************************/
    ProcessPCF(hConn, hEventQueue, pBuffer);
  }

MOD_EXIT:
  if (pBuffer != NULL)
  {
    free(pBuffer);
  }

  /*****************************************************************/
  /* Close the logger event queue                                  */
  /*****************************************************************/
  if (hEventQueue != MQHO_UNUSABLE_HOBJ)
  {
    MQCLOSE(hConn, &hEventQueue, MQCO_NONE, &CompCode, &Reason);
  }


  /*****************************************************************/
  /* Disconnect                                                    */
  /*****************************************************************/
  if (hConn != MQHC_UNUSABLE_HCONN)
  {
    MQDISC(&hConn, &CompCode, &Reason);
  }

  return 0;
}

/******************************************************************************/
/* Function: ProcessPCF                                                       */
/******************************************************************************/
/*                                                                            */
/* Input Parameters:  Handle to queue manager connection                      */
/*                    Handle to the opened logger event queue object          */
/*                    Pointer to a memory buffer to store the incoming PCF    */
/*                      message                                               */
/*                                                                            */
/* Output Parameters: None                                                    */
/*                                                                            */
/* Logic: Wait for messages to appear on the logger event queue and display   */
/*        their formatted contents.                                           */
/*                                                                            */
/******************************************************************************/

static void ProcessPCF(MQHCONN    hConn,
                       MQHOBJ     hEventQueue,
                       PMQCHAR    pBuffer)
{
  MQCFH   * pCfh;
  MQCFST  * pCfst;
  MQGMO     Gmo      = { MQGMO_DEFAULT };
  MQMD      Mqmd     = { MQMD_DEFAULT };
  PMQCHAR   pPCFCmd;
  MQLONG    CompCode = MQCC_OK;
  MQLONG    Reason   = MQRC_NONE;
  MQLONG    MsgLen;
  PMQCHAR   Parm = NULL;

  Gmo.Options = MQGMO_WAIT +
                MQGMO_CONVERT +
                MQGMO_FAIL_IF_QUIESCING;
  Gmo.WaitInterval = MQWI_UNLIMITED;   /* Set timeout value          */

  /*****************************************************************/
  /* Process response Queue                                        */
  /*****************************************************************/
  while (Reason == MQRC_NONE)
  {
```

```
memcpy(&Mqmd.MsgId, MQMI_NONE, sizeof(Mqmd.MsgId));
memset(&Mqmd.CorrelId, 0, sizeof(Mqmd.CorrelId));

MQGET( hConn,
       hEventQueue,
      &Mqmd,
      &Gmo,
       MAX_MESSAGE_LENGTH,
       pBuffer,
      &MsgLen,
      &CompCode,
      &Reason );

if (Reason != MQRC_NONE)
{
  switch(Reason)
  {
    case MQRC_NO_MSG_AVAILABLE:
         printf("Timed out");
         break;

    default:
         printf("MQGET ended with reason code %" Int32 "d\n", Reason);
         break;
  }
  goto MOD_EXIT;
}

/*********************************************************************/
/* Only expect PCF event messages on this queue                      */
/*********************************************************************/
if (memcmp(Mqmd.Format, MQFMT_EVENT, MQ_FORMAT_LENGTH))
{
  printf("Unexpected message format '%8.8s' received\n", Mqmd.Format);
  continue;
}

/*********************************************************************/
/* Build the output by parsing the received PCF message, first the */
/* header, then each of the parameters                               */
/*********************************************************************/
pCfh = (MQCFH *)pBuffer;

if (pCfh->Reason != MQRC_NONE)
{
  printf("-----------------------------------------------------------------\n");
  printf("Event Message Received\n");

  Parm = ParmToString(pCfh->Command);
  if (Parm != NULL)
  {
    printf("Command  :%s \n",Parm);
  }
  else
  {
    printf("Command  :%" Int32 "d \n",pCfh->Command);
  }

  printf("CompCode :%" Int32 "d\n"   ,pCfh->CompCode);

  Parm = ParmToString(pCfh->Reason);
  if (Parm != NULL)
  {
    printf("Reason   :%s \n",Parm);
  }
  else
  {
    printf("Reason   :%" Int32 "d \n",pCfh->Reason);
  }
}

pPCFCmd  = (PMQCHAR)  (pCfh+1);
printf("-----------------------------------------------------------------\n");
while(pCfh->ParameterCount--)
{
  pCfst = (MQCFST *) pPCFCmd;
  switch(pCfst->Type)
  {
    case MQCFT_STRING:
      Parm = ParmToString(pCfst->Parameter);
      if (Parm != NULL)
      {
```

```
            printf("%-32s",Parm);
          }
          else
          {
            printf("%-32" Int32 "d",pCfst->Parameter);
          }

          fwrite(pCfst->String, pCfst->StringLength, 1, stdout);
          pPCFCmd += pCfst->StrucLength;
          break;

        default:
          printf("Unrecognised datatype %" Int32 "d returned\n", pCfst->Type);
          goto MOD_EXIT;
      }
      putchar('\n');
    }
```

样本输出

此应用程序生成以下形式的输出:

```
/************************************/
/* Sample Logger Event Monitor start */
/************************************/
-----------------------------------------------------------
Event Message Received
Command  :Logger Event Command
CompCode :0
Reason   :Logger Status
-----------------------------------------------------------
Queue Manager Name              CSIM

Current Log Extent              AMQA000001
Restart Log Extent              AMQA000001
Media Log Extent                AMQA000001
Log Path                        QMCSIM
-----------------------------------------------------------
```

相关概念

第 43 页的『记录器事件用法』
使用此页面可查看如何使用记录器事件来确定队列管理器重新启动或介质恢复不再需要的日志扩展数据块。

第 42 页的『命令事件用法』
使用此页面来查看如何使用命令事件来生成已运行的命令的审计跟踪

相关参考

第 43 页的『记录器事件生成』
使用此页面来查看导致生成记录器事件的情境，并了解未生成记录器事件的环境

权限配置事件

通过命令行， MQSC ， PCF 或相应 IBM i 命令从任何安全控制操作进行更改时，将输出权限配置事件。

事件数据包含以下信息:

源信息
包括从其中进行更改的队列管理器，进行更改的用户的标识以及更改的产生方式，例如通过控制台命令。

上下文信息
来自命令消息的消息数据中的上下文信息的副本。

当在 SYSTEM.ADMIN.COMMAND.QUEUE 队列。

权限记录身份
包括权限记录的概要文件名称和对象类型。

对象属性
包括权限记录中所有属性的值。

对于更改权限记录事件，将生成两条消息，一条消息包含更改前的信息，另一条消息包含更改后的信息。

生成的每条事件消息都放置在 SYSTEM.ADMIN.CONFIG.EVENT 队列。

相关参考

第 11 页的『事件类型』
使用此页面来查看队列管理器或通道实例可以报告的检测事件类型

权限配置事件生成

使用此页面可查看导致生成权限配置事件的情境，以及了解未生成权限配置事件的环境。

权限配置事件向您通知对权限记录的属性所作的更改。 有三种类型的权限配置事件:

- 更改权限记录
- 删除权限记录
- 刷新权限记录

当 **CONFIGEV** 队列管理器属性设置为 *ENABLED* 并且发出以下任何命令或其 MQSC 等效命令时，会将权限事件消息放入配置事件队列中，即使对权限记录没有实际更改也是如此:

- 删除权限记录 PCF 命令
- 设置权限记录 PCF 命令
- setmqaut 控制命令
- RVKMQMAUT CL 命令
- GRTMQMAUT CL 命令

未生成权限配置事件时

在以下情况下，不会生成权限配置事件消息:

- 当命令失败时
- 当队列管理器尝试将消息放入事件队列时迂到错误，在这种情况下，命令完成，但不会生成事件消息
- 创建或删除队列管理器时
- 删除对象时，无论删除命令上的 **AUTHREC** 选项如何。 相应的命令事件显示该操作，该操作不适用于个别用户的权限记录。

相关概念

第 40 页的『命令事件』
命令事件是 MQSC 或 PCF 命令已成功运行的通知。

相关参考

刷新队列管理器

Multi 用于监视 Multiplatforms 版上的检测事件的样本程序

amqsevt 格式化队列管理器可以创建的检测事件，并随 IBM MQ for Multiplatforms 一起提供。 程序从事件队列读取消息，并将其格式化为可读字符串。

作为样本程序，提供了源和二进制。 此样本在所有多平台上提供，包括 IBM i。

单个二进制文件 `amqsevt` (或 `amqsevt.exe`) 在样本文件集中提供，并安装在样本 bin (`tools\c\samples\bin` 或 `bin64`) 目录中。

源文件 `amqsevta.c` 也在样本文件集中提供，并安装在样本目录 (即 Windows 上的 `tools\c\samples`) 中。

请注意，通过使用 MQCB 来检索消息，程序可以从多个事件队列中读取并预订多个主题。

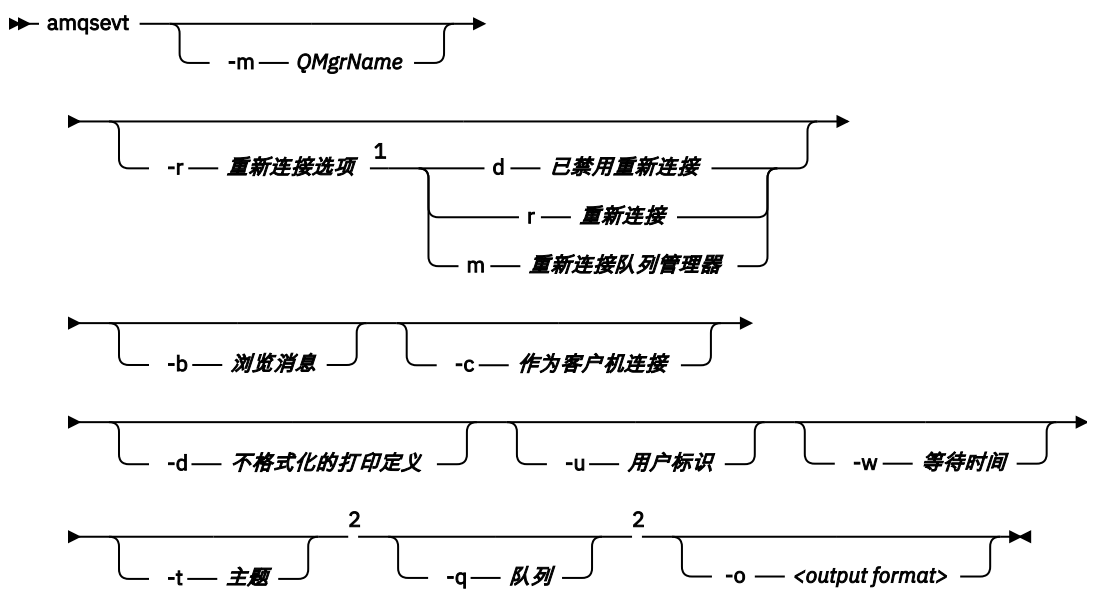
作为客户机运行时，样本可以连接到包括 z/OS 在内的任何队列管理器。

注意: 您可以在不指定任何参数的情况下使用程序，在这种情况下，程序会尝试连接到缺省队列管理器并从标准事件队列集 (SYSTEM.ADMIN. *。EVENT)。

在这种情况下，程序将永远等待消息，直到您按 Enter 键结束程序为止。

但是，您更有可能将该程序与所描述的各种选项配合使用。

语法



注:

[1] 仅当用作客户机时可用
[2] 队列和主题可以有多个条目

可选参数

-m *QueueManagerName*
指定用于读取事件的特定队列管理器。

-r *Reconnection Options*
用作客户机时的自动重新连接选项。 可能的值为:

d
已禁用重新连接客户机

r
重新连接客户机

m
重新连接队列管理器

-b
仅浏览记录，而不是破坏性地读取消息

-c
选择连接作为客户机。

-d
选择第二个示例中使用的打印方式。 MQI 常量与它们在头文件中显示的完全相同。

-U *User ID*
指定特定用户并导致出现请求密码的提示

-w *Wait*
如果在指定的秒数内未到达任何事件消息，那么将导致程序退出。

请注意，如果未指定时间，那么程序仅在按 Enter 键时正常结束。

-t *Topic* 和
-q *Queue*
可以在命令行上多次提供 **-q** 和 **-t** 选项。

因此，可以从某些标准队列以及从程序的单次运行中的主题 (如果要将事件发送到这些主题) 中进行读取。

如果在命令行上未指定任何队列或主题，那么将打开缺省事件队列。

注: 该程序检测是否已作为客户机连接到 z/OS 队列管理器，并相应地更改缺省事件队列集，因为 z/OS 没有 SYSTEM.ADMIN.LOGGER.EVENT 队列。

使用主题时，程序将非持久预订与受管队列配合使用，以便在退出时清除所有内容。

-o *<output format>*
输出的格式。 可能的值为:

文本
标准文本格式; 这是缺省值。

json
标准 JSON 格式; 任何了解 JSON 的应用程序都可以采用此输出并直接对其进行处理。

.

样本输出

以下三个示例显示了程序的输出。

第一个示例使用缺省格式化选项，其中程序采用字段的 MQI 定义并对输出进行格式化，以使输出更易读。

```
**** Message (320 Bytes) on Queue SYSTEM.ADMIN.QMGR.EVENT ****
Event Type                      : Queue Mgr Event
Reason                          : Unknown Alias Base Queue
Event created                   : 2015/06/17 13:47:07.02 GMT
  Queue Mgr Name                : V8003_A
  Queue Name                    : EVT.NO.BASE.QUEUE
  Base Object Name              : EVT.NOT.DEFINED
  Appl Type                     : Unix
  Appl Name                     : amqsput
  Base Type                     : Queue
```

第二个示例显示了不尝试转换 MQI 常量的备用格式 (使用 -d 选项)。 对于某些查找特定 MQI 值的脚本编制工具，这可能是首选。

```
**** Message (320 Bytes) on Queue SYSTEM.ADMIN.QMGR.EVENT ****
Event Type                      : MQCMD_Q_MGR_EVENT
Reason                          : MQRC_UNKNOWN_ALIAS_BASE_Q
Event created                   : 2015/06/17 13:52:48.18 GMT
  MQCA_Q_MGR_NAME               : V8003_A
  MQCA_Q_NAME                   : EVT.NO.BASE.QUEUE
  MQCA_BASE_OBJECT_NAME         : EVT.NOT.DEFINED
  MQIA_APPL_TYPE                : MQAT_UNIX
  MQCACF_APPL_NAME              : amqsput
  MQIA_BASE_TYPE                : MQOT_Q
```

第三个示例显示命令的 JSON 输出

```
amqsevt -m QM1 -q SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE -o json
```

```
{
"eventSource" : { "objectName": "SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE",
                  "objectType" : "Queue" },
"eventType" : {
    "name" : "Activity Trace",
    "value" : 209
  },
"eventReason" : {
    "name" : "None",
    "value" : 0
  },
"eventCreation" : {
    "timeStamp"  : "2018-07-10T12:44:26Z",
    "epoch"      : 1531226666
  },
"eventData" : {
  "queueMgrName" : "QM1",
  "hostName" : "<yourhostname>",
  "startDate" : "2018-07-10",
  "startTime" : "13:44:25",
  "endDate" : "2018-07-10",
  "endTime" : "13:44:26",
  "commandLevel" : 910,
...
```

用法示例

以下示例显示如何使用多个队列:

```
amqsevt -m QM1 -q SYSTEM.ADMIN.QMGR.EVENT -q SYSTEM.ADMIN.PERM.EVENT -w 1
```

相关概念

第 8 页的『事件监视』
事件监视是检测队列管理器网络中出现的 检测事件 的过程。 检测事件是队列管理器或通道实例检测到的事件的逻辑组合。 此类事件会导致队列管理器或通道实例将特殊消息 (称为 事件消息) 放在事件队列上。

第 9 页的『检测事件』
检测事件是队列管理器或通道实例检测特殊消息 (称为 事件消息) 并将其放入事件队列的条件的逻辑组合。

相关参考

C 编程

第 44 页的『用于监视记录器事件队列的样本 C 程序』
使用此页面来查看样本 C 程序，该程序监视记录器事件队列以获取新事件消息，读取这些消息，并将消息内容放入 stdout。

消息监控

消息监视是标识消息通过队列管理器网络所采用的路由的过程。 通过标识活动类型以及代表消息执行的活动序列，可以确定消息路由。

当消息通过队列管理器网络时，各种进程代表消息执行活动。 使用下列其中一种方法来确定消息路由:

- IBM MQ 显示路由应用程序 (dspmqrte)
- 活动记录
- 跟踪路由消息传递

这些方法都生成特殊消息，其中包含在消息通过队列管理器网络传递时对其执行的活动的相关信息。 使用这些特殊消息中返回的信息来实现以下目标:

- 记录消息活动。
- 确定消息的最后已知位置。
- 检测队列管理器网络中的路由问题。
- 帮助确定队列管理器网络中路由问题的原因。
- 确认队列管理器网络正在正确运行。

- 熟悉队列管理器网络的运行。
- 跟踪已发布的消息。

相关概念
消息类型

活动和业务

活动是应用程序代表消息执行的独立操作。 活动由操作组成，这些操作是应用程序执行的单个工作片段。

以下操作是活动的示例:

- 消息通道代理 (MCA) 将消息从传输队列发送到通道下
- MCA 从通道接收消息并将其放入其目标队列
- 应用程序从队列中获取消息，并将应答消息放入响应中。
- IBM MQ 发布/预订引擎处理消息。

活动由一个或多个 操作组成。 操作是应用程序执行的单个工作片段。 例如， MCA 从通道下的传输队列发送消息的活动由以下操作组成:

1. 从传输队列获取消息 (*Get* 操作)。
2. 将消息向下发送到通道 (发送 操作)。

在发布/预订网络中，处理消息的 IBM MQ 发布/预订引擎的活动可以包含以下多个操作:

1. 将消息放入主题字符串 (*Put* 操作)。
2. 对于为接收消息而考虑的每个订户，零个或多个操作 (发布 操作， 废弃发布 操作或 排除的发布 操作)。

来自活动的信息

您可以通过记录通过队列管理器网络路由消息时的信息来标识对消息执行的活动序列。 您可以根据对消息执行的活动序列确定通过队列管理器网络的消息路由，并可以获取以下信息:

消息的最后已知位置
如果消息未到达其预期目标，那么可以从完整或部分消息路由确定消息的最后已知位置。

队列管理器网络的配置问题
在通过队列管理器网络研究消息的路由时，您可能会看到消息未到达预期位置。 发生这种情况的原因有很多，例如，如果通道处于不活动状态，那么消息可能采用备用路由。

对于发布/预订应用程序，您还可以确定要发布到主题的消息以及由于发布到订户而在队列管理器网络中流动的任何消息的路径。

在这种情况下，系统管理员可以确定队列管理器网络中是否存在任何问题，并在适当情况下更正这些问题。

消息路由

根据确定消息路由的原因，您可以使用以下常规方法:

使用为跟踪路由消息记录的活动信息
跟踪路由消息记录特定用途的活动信息。 您可以使用它们来确定队列管理器网络的配置问题，或者确定消息的最后已知位置。 如果生成跟踪路由消息以确定未到达其预期目标的消息的最后已知位置，那么可以模拟原始消息。 这使跟踪路由消息最有可能遵循原始消息所采用的路由。

IBM MQ 显示路由应用程序可以生成跟踪路由消息。

使用为原始消息记录的活动信息
您可以为活动记录启用任何消息，并代表其记录活动信息。 如果消息未到达其预期目标，那么可以使用记录的活动信息来确定消息的最后已知位置。 通过使用原始消息中的活动信息，可以确定最准确的消息路由，从而导致最后一个已知位置。 要使用此方法，必须为活动记录启用原始消息。

警告: 避免启用队列管理器网络中的所有消息以进行活动记录。 针对活动记录启用的消息可以代表它们生成许多活动报告。 如果队列管理器网络中的每条消息都启用了活动记录，那么队列管理器网络流量可能会增加到不可接受的级别。

相关概念

第 52 页的『消息监控』
消息监视是标识消息通过队列管理器网络所采用的路由的过程。 通过标识活动类型以及代表消息执行的活动序列，可以确定消息路由。

第 54 页的『消息路由方法』
活动记录和跟踪路由消息传递是允许您在通过队列管理器网络路由消息时记录该消息的活动信息的技术。

第 59 页的『跟踪路由消息传递』
跟踪路由消息传递是一种使用 *trace-route messages* 来记录消息的活动信息的技术。 跟踪路由消息传递涉及将跟踪路由消息发送到队列管理器网络。

相关任务

编写您自己的消息通道代理程序

消息路由方法

活动记录和跟踪路由消息传递是允许您在通过队列管理器网络路由消息时记录该消息的活动信息的技术。

活动记录

如果消息指定了相应的报告选项，那么它会请求应用程序在通过队列管理器网络路由时生成 活动报告 。当应用程序代表消息执行活动时，可以生成活动报告，并将其传递到相应的位置。 活动报告包含有关对消息执行的活动的信息。

必须先按顺序排列使用活动报告收集的活动信息，然后才能确定消息路由。

跟踪路由消息传递

跟踪路由消息传递 是一种涉及将 跟踪路由消息 发送到队列管理器网络的技术。 当应用程序代表跟踪路由消息执行活动时，可以在跟踪路由消息的消息数据中累积活动信息，或者可以生成活动报告。 如果在跟踪路由消息的消息数据中累积了活动信息，那么当它到达其目标队列时，可以生成包含来自跟踪路由消息的所有信息的跟踪路由应答消息并将其传递到相应的位置。

由于跟踪路由消息专用于记录代表其执行的活动序列，因此与请求活动报告的常规消息相比，有更多处理选项可用。

比较活动记录和跟踪路由消息传递

活动记录和跟踪路由消息传递都可以提供活动信息，以确定消息通过队列管理器网络所采用的路由。 这两种方法都有各自的优势。

收益	活动记录	跟踪路由消息传递
可以确定消息的最后已知位置	Yes	Yes
可以确定队列管理器网络的配置问题	Yes	Yes
可由任何消息请求 (不限于与跟踪路由消息配合使用)	Yes	否
未修改消息数据	Yes	否
正常处理的消息	Yes	否
可以在消息数据中累积活动信息	否	Yes
可选消息传递到目标队列	否	Yes
如果在无限循环中捕获消息，那么可以对其进行检测和处理	否	Yes
可以可靠地按顺序放置活动信息	否	Yes

收益	活动记录	跟踪路由消息传递
为显示活动信息而提供的应用程序	否	Yes

消息路由完整性

在某些情况下，无法确定代表消息执行的完整活动序列，因此只能确定部分消息路由。 消息路由的完整性直接受路由消息的队列管理器网络的影响。

连接到队列管理器的 MCA 和用户编写的应用程序可以记录与代表消息执行的活动相关的信息。 活动信息的记录由队列管理器属性 ACTIVE 和 ROUTEREC 控制。 队列管理器网络可以确定完整的消息路由。

活动信息的存储方式

IBM MQ 将活动信息存储在活动报告，跟踪路由消息或跟踪路由应答消息中。 在每种情况下，信息都存储在称为 活动 PCF 组的结构中。 跟踪路由消息或跟踪路由应答消息可以包含许多活动 PCF 组，具体取决于对该消息执行的活动数。 活动报告包含一个活动 PCF 组，因为将为每个记录的活动生成单独的活动报告。

通过跟踪路由消息传递，可以记录其他信息。 此附加信息存储在称为 *TraceRoute* PCF 组的结构中。 TraceRoute PCF 组包含许多 PCF 结构，这些结构用于存储其他活动信息，并指定用于确定在通过队列管理器网络路由跟踪路由消息时如何处理该消息的选项。

相关概念

第 55 页的『活动记录』
活动记录是一种确定消息通过队列管理器网络所采用的路由的技术。 为了确定消息所采用的路径，将记录代表消息执行的活动。

第 59 页的『跟踪路由消息传递』
跟踪路由消息传递是一种使用 *trace-route messages* 来记录消息的活动信息的技术。 跟踪路由消息传递涉及将跟踪路由消息发送到队列管理器网络。

相关参考

第 63 页的『TraceRoute PCF 组』
TraceRoute PCF 组中的属性控制跟踪路由消息的行为。 *TraceRoute* PCF 组位于每个跟踪路由消息的消息数据中。

第 93 页的『活动报告消息数据』
使用此页面来查看活动报告消息中的 活动 PCF 组所包含的参数。 仅当已执行特定操作时，才会返回某些参数。

活动记录

活动记录是一种确定消息通过队列管理器网络所采用的路由的技术。 为了确定消息所采用的路径，将记录代表消息执行的活动。

使用活动记录时，可以在活动报告中记录代表消息执行的每个活动。 活动报告是一种报告消息类型。 每个活动报告都包含有关代表消息执行活动的应用程序的信息，活动发生的时间以及有关作为活动一部分执行的操作的信息。 通常会将活动报告传递到一起收集这些报告的应答队列。 通过研究与消息相关的活动报告，您可以确定消息通过队列管理器网络所采用的路由。

活动报告使用情况

通过队列管理器网络路由消息时，可以生成活动报告。 您可以通过以下方式使用活动报告信息:

确定消息的最后已知位置
如果针对活动记录启用的消息未到达其预期目标，那么可以研究通过队列管理器网络路由该消息时为其生成的活动报告，以确定消息的最后已知位置。

确定队列管理器网络的配置问题
可以将许多启用了活动记录的消息发送到队列管理器网络中。 通过研究与每条消息相关的活动报告，可以明显看出它们没有采用预期的路径。 发生这种情况的原因有很多，例如，通道可能已停止，从而强制

消息采用替代路由。 在这些情况下，系统管理员可以确定队列管理器网络中是否存在任何问题，如果存在，请更正这些问题。

注: 您可以使用 IBM MQ 显示路由应用程序将活动记录与跟踪路由消息结合使用。

活动报告格式

活动报告是由代表消息执行活动的应用程序生成的 PCF 消息。 活动报告是包含消息描述符和消息数据的标准 IBM MQ 报告消息，如下所示:

消息描述符

- MQMD 结构

消息数据

- 嵌入式 PCF 头 (MQEPH)
- 活动报告消息数据

活动报告消息数据由 *Activity* PCF 组组成，如果为跟踪路由消息生成，那么为 *TraceRoute* PCF 组。

相关参考

MQMD - 消息描述符

MQEPH-嵌入式 PCF 头

控制活动记录

在队列管理器级别启用活动记录。 要启用整个队列管理器网络，请单独启用网络中的每个队列管理器以进行活动记录。 如果启用更多队列管理器，那么将生成更多活动报告。

关于此任务

要在消息通过队列管理器路由时为其生成活动报告: 定义消息以请求活动报告; 启用队列管理器以进行活动记录; 并确保在消息上执行活动的应用程序能够生成活动报告。

如果您不希望在通过队列管理器路由消息时为消息生成活动报告，请 禁用 队列管理器以进行活动记录。

过程

1. 请求消息的活动报告

   a) 在消息的消息描述符中，在 报告 字段中指定 MQRO_ACTIVITY。

   b) 在消息的消息描述符中，在 *ReplyToQ* 字段中指定应答队列的名称。

   警告: 避免启用队列管理器网络中的所有消息以进行活动记录。 针对活动记录启用的消息可以代表它们生成许多活动报告。 如果队列管理器网络中的每条消息都启用了活动记录，那么队列管理器网络流量可能会增加到不可接受的级别。

2. 启用或禁用队列管理器以进行活动记录。

   使用 MQSC 命令 `ALTER QMGR`(指定参数 `ACTIVREC`) 来更改队列管理器属性的值。 值可以是:

   MSG
   已启用队列管理器以进行活动记录。 生成的任何活动报告都将传递到消息的消息描述符中指定的应答队列。 这是缺省值。

   队列
   已启用队列管理器以进行活动记录。 生成的任何活动报告都将传递到本地系统队列 SYSTEM.ADMIN.ACTIVITY.QUEUE。 系统队列还可用于将活动报告转发到公共队列。

   DISABLED
   已禁用队列管理器以进行活动记录。 在此队列管理器的作用域内不会生成任何活动报告。

例如，要启用队列管理器以进行活动记录，并指定将生成的任何活动报告传递到本地系统队列 SYSTEM.ADMIN.ACTIVITY.QUEUE，使用以下 MQSC 命令:

```
ALTER QMGR ACTIVREC(QUEUE)
```

切记: 修改 *ACTIVREC* 队列管理器属性时，正在运行的 MCA 直到通道重新启动后才会检测到更改。

3. 确保应用程序使用与 MCA 相同的算法来确定是否为消息生成活动报告:
   a) 验证消息是否已请求生成活动报告
   b) 验证消息当前所在的队列管理器是否已启用活动记录
   c) 将活动报告放在由 *ACTIVREC* 队列管理器属性确定的队列上

为活动报告设置公共队列

要在将报告传递到本地系统队列时确定与特定消息相关的活动报告的位置，在单个节点上使用公共队列更高效

开始之前

设置 **ACTIVREC** 参数以启用队列管理器进行活动记录，并指定将生成的任何活动报告传递到本地系统队列 SYSTEM.ADMIN.ACTIVITY.QUEUE。

关于此任务

如果将队列管理器网络中的多个队列管理器设置为将活动报告传递到本地系统队列，那么确定与特定消息相关的活动报告的位置可能很耗时。 或者，使用单个节点，该节点是托管公共队列的队列管理器。 队列管理器网络中的所有队列管理器都可以将活动报告交付到此公共队列。 使用公共队列的好处是，队列管理器不必将活动报告交付到消息中指定的应答队列，并且在确定与消息相关的活动报告的位置时，仅查询一个队列。

要设置公共队列，请执行以下步骤:

过程

1. 选择或定义队列管理器作为单个节点
2. 在单个节点上，选择或定义要用作公共队列的队列
3. 在要将活动报告传递到公共队列的所有队列管理器上，重新定义本地系统队列 SYSTEM.ADMIN.ACTIVITY.QUEUE 作为远程队列定义:
   a) 将单个节点的名称指定为远程队列管理器名称
   b) 将公共队列的名称指定为远程队列名称

确定消息路由信息

要确定消息路由，请从收集的活动报告中获取信息。 确定应答队列上是否有足够的活动报告，以使您能够确定所需信息并按顺序排列活动报告。

关于此任务

将活动报告放入应答队列的顺序不一定与执行活动的顺序相关。 您必须手动订购活动报告，除非为跟踪路由消息生成了活动报告，在这种情况下，您可以使用 IBM MQ 显示路由应用程序来订购活动报告。

确定应答队列上是否有足够的活动报告，以便您获取必需的信息:

过程

1. 通过比较活动报告和原始消息的标识，确定应答队列上的所有相关活动报告。 请确保设置原始消息的报告选项，以便活动报告可以与原始消息相关。
2. 对应答队列中标识的活动报告进行排序。
   您可以使用活动报告中的以下参数:

OperationType

执行的操作类型可能使您能够确定在当前活动报告之前或之后直接生成的活动报告。

例如，活动报告详细说明 MCA 从传输队列向通道发送消息。 活动报告中详细描述的最后一个操作具有 *OperationType* `send` 以及使用通道 CH1 将消息发送到目标队列管理器 QM1 的详细信息。 这意味着对消息执行的下一个活动将在队列管理器 QM1 上发生，并且它将以来自通道 CH1 的 `receive` 操作开始。 通过使用此信息，您可以识别下一个活动报告，前提是该报告存在并且已获取。

OperationDate* 和 *OperationTime

您可以根据每个活动报告中的操作日期和时间来确定活动的常规顺序。

警告: 除非队列管理器网络中的每个队列管理器的系统时钟同步，否则按日期和时间排序并不能保证活动报告的顺序正确。 您必须手动建立订单。

活动报告的顺序表示消息通过队列管理器网络所使用的路由或部分路由。

3. 从订购的活动报告中的活动信息获取所需的信息。

   如果有关该消息的信息不足，那么您可能能够获取更多活动报告。

检索更多活动报告

要确定消息路由，必须从收集的活动报告中提供足够的信息。 如果从消息指定的应答队列中检索与消息相关的活动报告，但您没有必需的信息，请查找进一步的活动报告。

关于此任务

要确定任何其他活动报告的位置，请执行以下步骤:

过程

1. 对于队列管理器网络中将活动报告交付到公共队列的任何队列管理器，请从具有与原始消息的 *MsgId* 相匹配的 *CorrelId* 的公共队列中检索活动报告。
2. 对于队列管理器网络中未向公共队列交付活动报告的任何队列管理器，请按如下所示检索活动报告:
   a) 检查现有活动报告以确定通过其路由消息的队列管理器。
   b) 对于这些队列管理器，请标识已启用活动记录的队列管理器。
   c) 对于这些队列管理器，标识未将活动报告返回到指定应答队列的任何队列管理器。
   d) 对于您标识的每个队列管理器，请检查系统队列 SYSTEM.ADMIN.ACTIVITY.QUEUE 并检索具有与原始消息的 *MsgId* 匹配的 *CorrelId* 的任何活动报告。
   e) 如果在系统队列上找不到任何活动报告，请检查队列管理器死信队列 (如果存在)。

      仅当设置了报告选项 MQRO_DEAD_LETTER_Q 时，才能将活动报告传递到死信队列。
3. 按顺序排列所有获取的活动报告。

   然后，活动报告的顺序表示消息所使用的路由或部分路由。
4. 从订购的活动报告中的活动信息获取所需的信息。

   在某些情况下，记录的活动信息无法到达指定的应答队列，公共队列或系统队列。

未获取活动信息的情况

要确定代表消息执行的完整活动序列，必须获取与每个活动相关的信息。 如果未记录或未获取与任何活动相关的信息，那么只能确定部分活动序列。

在以下情况下不记录活动信息:

- 消息由早于 IBM WebSphere MQ 6.0 的队列管理器处理。
- 消息由未启用活动记录的队列管理器处理。
- 期望处理消息的应用程序未在运行。

在以下情况下，记录的活动信息无法到达指定的应答队列:

- 没有定义通道将活动报告路由到应答队列。

- 用于将活动报告路由到应答队列的通道未在运行。
- 用于路由活动报告的远程队列定义未定义应答队列所在的队列管理器 (队列管理器别名)。
- 生成原始消息的用户对队列管理器别名没有打开或放置权限。
- 生成原始消息的用户没有打开或放入应答队列的权限。
- 禁止放入应答队列。

在以下情况下，记录的活动信息无法到达系统队列或公共队列:

- 如果要使用公共队列，并且没有定义通道将活动报告路由到公共队列。
- 如果要使用公共队列，并且用于将活动报告路由到公共队列的通道未在运行。
- 如果要使用公共队列并且未正确定义系统队列。
- 生成原始消息的用户没有打开或放入系统队列的权限。
- 禁止放入系统队列。
- 如果要使用公共队列，并且生成原始消息的用户对该公共队列没有打开或放置权限。
- 如果要使用公共队列并且禁止放置公共队列。

在这些情况下，如果未指定报告选项 MQRO_DISCARD_MSG ，那么如果在拒绝活动报告的队列管理器上定义了死信队列，那么可以从该队列中检索活动报告。 仅当生成活动报告的原始消息同时在消息描述符的 "报告" 字段中指定了 MQRO_PASS_DISCARD_AND_到期和 MQRO_DISCARD_MSG 时，活动报告才会指定此报告选项。

跟踪路由消息传递

跟踪路由消息传递是一种使用 *trace-route messages* 来记录消息的活动信息的技术。 跟踪路由消息传递涉及将跟踪路由消息发送到队列管理器网络。

由于跟踪路由消息是通过队列管理器网络路由的，因此会记录活动信息。 此活动信息包含有关执行活动的应用程序，执行这些活动的时间以及作为活动的一部分执行的操作的信息。 您可以将使用跟踪路由消息传递记录的信息用于以下目的:

确定消息的最后已知位置
如果消息未到达其预期目标，那么可以使用为跟踪路由消息记录的活动信息来确定消息的最后已知位置。 跟踪路由消息将发送到与原始消息具有相同目标的队列管理器网络中，并打算遵循相同的路由。 活动信息可以累积在跟踪路由消息的消息数据中，也可以使用活动报告进行记录。 要增加跟踪路由消息遵循与原始消息相同的路由的概率，您可以修改跟踪路由消息以模拟原始消息。

确定队列管理器网络的配置问题
跟踪路由消息将发送到队列管理器网络并记录活动信息。 通过研究为跟踪路由消息记录的活动信息，可以明显看出跟踪路由消息未遵循预期路由。 发生这种情况的原因有很多，例如，通道可能处于不活动状态，从而强制消息采用替代路由。 在这些情况下，系统管理员可以确定队列管理器网络中是否存在任何问题，如果存在，请更正这些问题。

您可以使用 IBM MQ 显示路由应用程序来配置，生成跟踪路由消息并将其放入队列管理器网络中。

警告: 如果将跟踪路由消息放入分发列表，那么结果将未定义。

相关概念
第 108 页的『跟踪路由消息引用』
使用此页面来获取跟踪路由消息格式的概述。 跟踪路由消息数据包含用于描述跟踪路由消息所导致的活动的参数

如何记录活动信息

通过跟踪路由消息传递，可以在跟踪路由消息的消息数据中记录活动信息，或者使用活动报告。 或者，可以使用这两种方法。

在跟踪路由消息的消息数据中累积活动信息

当通过队列管理器网络路由跟踪路由消息时，可以在跟踪路由消息的消息数据中累积有关代表跟踪路由消息执行的活动的信息。 活动信息存储在 *Activity* PCF 组中。 对于代表跟踪路由消息执行的每个活动，会将 活动 PCF 组写入跟踪路由消息的消息数据中的 PCF 块末尾。

其他活动信息记录在称为 *TraceRoute* PCF 组的 PCF 组中的跟踪路由消息传递中。 附加的活动信息存储在这个 PCF 组中，可以用来帮助确定记录的活动的顺序。 此方法由 *TraceRoute* PCF 组中的 累计 参数控制。

使用活动报告记录活动信息

通过队列管理器网络路由跟踪路由消息时，可以为代表跟踪路由消息执行的每个活动生成活动报告。 活动信息存储在 *Activity* PCF 组中。 对于代表跟踪路由消息执行的每个活动，将生成包含 活动 PCF 组的活动报告。 跟踪路由消息的活动记录与任何其他消息的工作方式相同。

与为任何其他消息生成的活动报告相比，为跟踪路由消息生成的活动报告包含其他活动信息。 其他信息将在 *TraceRoute* PCF 组中返回。 *TraceRoute* PCF 组中包含的信息仅从生成活动报告的时间开始准确。 您可以使用其他信息来帮助确定代表跟踪路由消息执行的活动的顺序。

获取记录的活动信息

当跟踪路由消息到达其预期目标或被废弃时，用于获取活动信息的方法取决于记录该信息的方式。

开始之前

如果您不熟悉活动信息，请参阅 第 59 页的『如何记录活动信息』。

关于此任务

使用以下方法在跟踪路由消息到达其预期目标或被废弃后获取活动信息:

过程

- 检索跟踪路由消息。

  TraceRoute PCF 组中的 *Deliver* 参数控制在到达时是将跟踪路由消息放在目标队列上，还是将其废弃。 如果将跟踪路由消息传递到目标队列，那么可以从此队列中检索跟踪路由消息。 然后，可以使用 IBM MQ 显示路由应用程序来显示活动信息。

  要请求在跟踪路由消息的消息数据中累积活动信息，请将 *TraceRoute* PCF 组中的 累计 参数设置为 `MQROUTE_ACCUMULATE_IN_MSG`。

- 使用跟踪路由应答消息。

  当跟踪路由消息到达其预期目标，或者无法在队列管理器网络中进一步路由跟踪路由消息时，可以生成跟踪路由应答消息。 跟踪路由应答消息包含来自跟踪路由消息的所有活动信息的重复项，并且传递到指定的应答队列或系统队列 SYSTEM.ADMIN.TRACE.ROUTE.QUEUE。 您可以使用 IBM MQ 显示路由应用程序来显示活动信息。

  要请求跟踪路由应答消息，请将 *TraceRoute* PCF 组中的 累计 参数设置为 `MQROUTE_ACCUMULATE_AND_REPLY`。

- 使用活动报告。

  如果为跟踪路由消息生成了活动报告，那么必须先找到活动报告，然后才能获取活动信息。 然后，要确定活动序列，必须对活动报告进行排序。

控制跟踪路由消息传递

在队列管理器级别启用跟踪路由消息传递，以便该队列管理器作用域中的应用程序可以将活动信息写入跟踪路由消息。 要启用整个队列管理器网络，请单独启用网络中的每个队列管理器以进行跟踪路由消息传递。 如果启用更多队列管理器，那么将生成更多活动报告。

开始之前

如果要使用活动报告来记录跟踪路由消息的活动信息，请参阅 第 56 页的『控制活动记录』。

关于此任务

要在通过队列管理器路由跟踪路由消息时记录该消息的活动信息，请执行以下步骤:

过程

- 定义如何记录跟踪路由消息的活动信息。

  请参阅 第 62 页的『生成和配置跟踪路由消息』
- 如果要在跟踪路由消息中累积活动信息，请确保队列管理器已启用跟踪路由消息传递
- 如果要在跟踪路由消息中累积活动信息，请确保在跟踪路由消息上执行活动的应用程序能够将活动信息写入跟踪路由消息的消息数据

相关概念

第 62 页的『生成和配置跟踪路由消息』

一种跟踪路由消息，包括特定的消息描述符和消息数据部分。 要生成跟踪路由消息，请手动创建消息或使用 IBM MQ 显示路由应用程序。

相关任务

第 56 页的『控制活动记录』

在队列管理器级别启用活动记录。 要启用整个队列管理器网络，请单独启用网络中的每个队列管理器以进行活动记录。 如果启用更多队列管理器，那么将生成更多活动报告。

启用队列管理器以进行跟踪路由消息传递

要控制是对跟踪路由消息传递启用还是禁用队列管理器，请使用队列管理器属性 ROUTEREC。

使用 MQSC 命令 `ALTER QMGR`，指定参数 `ROUTEREC` 以更改队列管理器属性的值。 值可以是以下任意值:

MSG
队列管理器已启用跟踪路由消息传递。 队列管理器作用域内的应用程序可以将活动信息写入跟踪路由消息。

如果 *TraceRoute* PCF 组中的 累计 参数设置为 `MQROUTE_ACCUMULATE_AND_REPLY`，并且要对 trace-route 消息执行下一个活动:

- 是废弃
- 是放入本地队列 (目标队列或死信队列)
- 将导致跟踪路由消息上执行的活动总数超过 *TraceRoute* PCF 组中 *MaxActivities* 参数的值。

生成跟踪路由应答消息，并将其传递到在跟踪路由消息的消息描述符中指定的应答队列。

队列
队列管理器已启用跟踪路由消息传递。 队列管理器作用域内的应用程序可以将活动信息写入跟踪路由消息。

如果 *TraceRoute* PCF 组中的 累计 参数设置为 `MQROUTE_ACCUMULATE_AND_REPLY`，并且要对 trace-route 消息执行下一个活动:

- 是废弃
- 是放入本地队列 (目标队列或死信队列)
- 将导致跟踪路由消息上执行的活动总数超过 *TraceRoute* PCF 组中 *MaxActivities* 参数的值。

将生成跟踪路由应答消息，并将其传递到本地系统队列 SYSTEM.ADMIN.TRACE.ROUTE.QUEUE。

DISABLED
已禁用队列管理器以进行跟踪路由消息传递。 跟踪路由消息中未累积活动信息，但是可以在此队列管理器的作用域内更新 *TraceRoute* PCF 组。

例如，要对跟踪路由消息传递禁用队列管理器，请使用以下 MQSC 命令:

```
ALTER QMGR ROUTEREC(DISABLED)
```

切记: 修改 *ROUTEREC* 队列管理器属性时，正在运行的 MCA 直到通道重新启动后才会检测到更改。

启用应用程序以进行跟踪路由消息传递
要对用户应用程序启用跟踪路由消息传递，请将算法基于消息通道代理程序 (MCA) 所使用的算法

开始之前

如果您不熟悉跟踪路由消息的格式，请参阅 第 108 页的『跟踪路由消息引用』。

关于此任务

为跟踪路由消息传递启用了消息通道代理程序 (MCA)。 要对用户应用程序启用跟踪路由消息传递，请使用 MCAs 使用的算法中的以下步骤:

过程

1. 确定正在处理的消息是否为跟踪路由消息。

   如果消息不符合跟踪路由消息的格式，那么不会将该消息作为跟踪路由消息进行处理。
2. 确定是否要记录活动信息。

   如果执行的活动的详细信息级别不低于 *Detail* 参数指定的详细信息级别，那么将在特定情况下记录活动信息。 仅当跟踪路由消息请求累积，并且队列管理器启用了跟踪路由消息传递，或者跟踪路由消息请求了活动报告并且队列管理器启用了活动记录时，才会记录此信息。

   - 如果要记录活动信息，请递增 *RecordedActivities* 参数。
   - 如果不记录活动信息，请递增 *UnrecordedActivities* 参数。
3. 确定对跟踪路由消息执行的活动总数是否超过 *MaxActivities* 参数的值。

   活动总数是 *RecordedActivities*， *UnrecordedActivities* 和 *DiscontinuityCount* 的总和。

   如果活动总数超过 *MaxActivities*，请拒绝具有反馈 MQFB_MAX_ACTIVactivities 的消息。
4. 如果 累积 的值设置为 MQROUTE_累计 _IN_MSG 或 MQROUTE_累计 _AND_REPLY ，并且队列管理器已启用跟踪路由消息传递，请将活动 PCF 组写入跟踪路由消息的消息数据中的 PCF 块末尾。
5. 将跟踪路由消息传递到本地队列。

   - 如果将参数 *Deliver* 指定为 MQROUTE_DELI_NO ，请拒绝带有反馈 MQFB_NOT_交付的 TRACE 路由消息。
   - 如果将参数 *Deliver* 指定为 MQROUTE_DELI_YES ，请将跟踪路由消息传递到本地队列。
6. 如果满足以下所有条件，那么生成跟踪路由应答消息:

   - 跟踪路由消息已传递到本地队列或被拒绝
   - 参数 累计的值为 MQROUTE_累计 _and_reply
   - 队列管理器已启用跟踪路由消息传递

   跟踪路由应答消息将放在由 ROUTEREC 队列管理器属性确定的队列上。
7. 如果跟踪路由消息请求了活动报告，并且队列管理器已启用活动记录，请生成活动报告。

   活动报告将放在由 ACTIVREC 队列管理器属性确定的队列上。

生成和配置跟踪路由消息

一种跟踪路由消息，包括特定的消息描述符和消息数据部分。 要生成跟踪路由消息，请手动创建消息或使用 IBM MQ 显示路由应用程序。

跟踪路由消息由以下部分组成:

消息描述符
MQMD 结构， 格式 字段设置为 MQFMT_ADMIN 或 MQFMT_EMBEDDED_PCF。

消息数据
下列其中一个组合:

- PCF 头 (MQCFH) 和跟踪路由消息数据 (如果 *Format* 设置为 MQFMT_ADMIN)
- 嵌入式 PCF 头 (MQEPH) ， 跟踪路由消息数据和其他用户指定的消息数据 (如果 格式 设置为 MQFMT_EMBEDDED_PCF)

跟踪路由消息数据由 *TraceRoute* PCF 组和一个或多个 活动 PCF 组组成。

手动生成

手动生成跟踪路由消息时，不需要 *Activity* PCF 组。 活动 PCF 组在 MCA 或用户编写的应用程序代表其执行活动时写入跟踪路由消息的消息数据。

IBM MQ 显示路由应用程序

使用 IBM MQ 显示路由应用程序 **dspmqrte** 来配置，生成跟踪路由消息并将其放入队列管理器网络中。 将消息描述符中的 *Format* 参数设置为 MQFMT_ADMIN。 不能将用户数据添加到 IBM MQ 显示路由应用程序生成的跟踪路由消息。

限制: 无法在 IBM MQ for z/OS 队列管理器上发出 `dspmqrte` 。 如果要将跟踪路由消息传递到的第一个队列管理器成为此类型的队列管理器，请使用可选参数 `-c` 作为客户机连接到队列管理器。

模仿原始消息

当使用跟踪路由消息来确定另一条消息通过队列管理器网络所采用的路由时，跟踪路由消息对原始消息的模仿越紧密，那么跟踪路由消息遵循与原始消息相同的路由的可能性就越大。

以下消息特征可能会影响消息在队列管理器网络中的转发位置:

Priority
可以在消息的消息描述符中指定优先级。

持久
可以在消息的消息描述符中指定持久性。

到期
可以在消息的消息描述符中指定到期。

报告选项
可以在消息的消息描述符中指定报告选项。

消息大小
为了模拟消息的大小，可以将其他数据写入消息的消息数据。 为此，额外的消息数据可能毫无意义。

提示: IBM MQ 显示路由应用程序无法指定消息大小。

消息数据
某些队列管理器网络使用基于内容的路由来确定转发消息的位置。 在这些情况下，需要写入跟踪路由消息的消息数据以模拟原始消息的消息数据。

提示: IBM MQ 显示路由应用程序无法指定消息数据。

TraceRoute PCF 组

TraceRoute PCF 组中的属性控制跟踪路由消息的行为。 *TraceRoute* PCF 组位于每个跟踪路由消息的消息数据中。

下表列出了 *TraceRoute* 组中 MCA 可识别的参数。 如果编写用户编写的应用程序以识别这些应用程序，那么可以添加更多参数，如 第 67 页的『其他活动信息』 中所述。

表 *12: TraceRoute PCF* 组

参数	类型
TraceRoute	MQCFGR
所需详情	MQCFIN
RecordedActivities	MQCFIN
UnrecordedActivities	MQCFIN
DiscontinuityCount	MQCFIN
MaxActivities	MQCFIN
累计	MQCFIN
向前	MQCFIN
交付	MQCFIN

TraceRoute PCF 组中每个参数的描述如下:

所需详情
指定要记录的活动信息的详细级别。 值可以是以下任意值:

MQROUTE_DETAIL_LOW
仅记录用户应用程序执行的活动。

MQROUTE_DETAIL_MEDIUM
应记录 MQROUTE_DETAIL_LOW 中指定的活动。 此外，还会记录 MCA 执行的活动。

MQROUTE_DETAIL_HIGH
应记录 MQROUTE_DETAIL_LOW 和 MQROUTE_DETAIL_MEDIUM 中指定的活动。 MCA 不会在此详细信息级别记录任何进一步的活动信息。 此选项仅对要记录进一步活动信息的用户应用程序可用。 例如，如果用户应用程序通过考虑某些消息特征来确定消息所采用的路由，那么可将有关路由逻辑的信息包含在此详细信息级别中。

RecordedActivities
指定代表跟踪路由消息执行的记录活动数。 如果有关活动的信息已写入跟踪路由消息，或者已生成活动报告，那么将视为记录该活动。 对于每个记录的活动， *RecordedActivities* 递增 1。

UnrecordedActivities
指定代表跟踪路由消息执行的未记录活动数。 如果启用了跟踪路由消息传递的应用程序既不累积也不将相关活动信息写入活动报告，那么会将该活动视为未记录。

在以下情况下，不会记录代表跟踪路由消息执行的活动:

- 执行的活动的详细信息级别小于参数 详细信息指定的详细信息级别。
- 跟踪路由消息请求活动报告但未累积，并且队列管理器未启用活动记录。
- 跟踪路由消息请求累积，但不是活动报告，并且未对队列管理器启用跟踪路由消息传递。
- 跟踪路由消息同时请求累积和活动报告，并且未对队列管理器启用活动记录和跟踪路由消息传递。
- 跟踪路由消息既不请求累积，也不请求活动报告。

对于每个未记录的活动，参数 *UnrecordedActivities* 递增 1。

DiscontinuityCount
指定在未启用跟踪路由消息传递的应用程序的情况下，通过队列管理器路由跟踪路由消息的次数。 此值由队列管理器递增。 如果此值大于 0 ，那么只能确定部分消息路由。

MaxActivities
指定可以代表跟踪路由消息执行的最大活动数。

活动总数是 *RecordedActivities*， *UnrecordedActivities* 和 *DiscontinuityCount* 的总和。 活动总数不得超过 *MaxActivities* 的值。

MaxActivities 的值可以是:

正整数
最大活动数。

如果超过最大活动数，那么将使用反馈 MQFB_MAX_ACTIVactivities 来拒绝跟踪路由消息。 这可以防止在无限循环中捕获到跟踪路由消息时无限期转发该消息。

MQROUTE_UNLIMITED_ACTIVITIES
可以代表跟踪路由消息执行无限数量的活动。

累计
指定用于累积活动信息的方法。 值可以是以下任意值:

MQROUTE_累加 _in_msg
如果为跟踪路由消息传递启用了队列管理器，那么将在跟踪路由消息的消息数据中累积活动信息。

如果指定了此值，那么跟踪路由消息数据由以下内容组成:

- *TraceRoute* PCF 组。
- 零个或多个 活动 PCF 组。

MQROUTE_蓄电池 _and_reply
如果为跟踪路由消息传递启用了队列管理器，那么将在跟踪路由消息的消息数据中累积活动信息，并且如果发生以下任何情况，将生成跟踪路由应答消息:

- IBM MQ 队列管理器将废弃跟踪路由消息。
- IBM MQ 队列管理器将跟踪路由消息放入本地队列 (目标队列或死信队列)。
- 对跟踪路由消息执行的活动数超过 *MaxActivities* 的值。

如果指定了此值，那么跟踪路由消息数据由以下内容组成:

- *TraceRoute* PCF 组。
- 零个或多个 活动 PCF 组。

MQROUTE_累计无
未在跟踪路由消息的消息数据中累积活动信息。

如果指定了此值，那么跟踪路由消息数据由以下内容组成:

- *TraceRoute* PCF 组。

向前
指定可以将跟踪路由消息转发到的位置。 值可以是:

MQROUTE_FORWARD_IF_SUPPORTED
跟踪路由消息仅转发给队列管理器，这些队列管理器将采用 *TraceRoute* 组中 *Deliver* 参数的值。

MQROUTE_FORWARD_ALL
无论是否将使用 *Deliver* 参数的值，都会将跟踪路由消息转发到任何队列管理器。

在确定是否将跟踪路由消息转发到远程队列管理器时，队列管理器使用以下算法:

1. 确定远程队列管理器是否能够支持跟踪路由消息传递。
   - 如果远程队列管理器能够支持跟踪路由消息传递，那么算法将继续执行步骤 第 65 页的『4』。
   - 如果远程队列管理器无法支持跟踪路由消息传递，那么算法将继续执行步骤 第 65 页的『2』
2. 确定 *TraceRoute* 组中的 *Deliver* 参数是否包含 MQROUTE_DELIVERY _rej_unsup_mask 位掩码中的任何无法识别的交付选项。
   - 如果找到任何无法识别的传递选项，那么将拒绝具有反馈 MQFB_UNSUPPORTED_DELIVERY 的跟踪路由消息。
   - 如果找不到无法识别的传递选项，那么算法将继续执行步骤 第 65 页的『3』。
3. 从跟踪路由消息中的 *TraceRoute* PCF 组确定参数 *Deliver* 的值。
   - 如果将 *Deliver* 指定为 MQROUTE_DELI_YES ，那么会将 trace-route 消息转发到远程队列管理器。
   - 如果将 *Deliver* 指定为 MQROUTE_DELI_NO ，那么算法将继续执行步骤 第 65 页的『4』。
4. 确定 *TraceRoute* 组中的 *Forward* 参数是否包含 MQROUTE_FORWARDING_REJ_UNSUP_MASK 位掩码中的任何无法识别的转发选项。
   - 如果找到任何无法识别的转发选项，那么将使用反馈 MQFB_UNSUPPORTED_转发来拒绝跟踪路由消息。
   - 如果找不到无法识别的转发选项，那么算法将继续执行步骤 第 65 页的『5』。
5. 从跟踪路由消息中的 *TraceRoute* PCF 组确定参数 *Forward* 的值。
   - 如果将 转发 指定为 MQROUTE_FORWARD_IF_SUPPORTED ，那么将拒绝具有反馈 MQFB_NOT_FORWARD 的跟踪路由消息。
   - 如果将 转发 指定为 MQROUTE_FORWARD_ALL ，那么可以将跟踪路由消息转发到远程队列管理器。

交付
指定当跟踪路由消息到达其预期目标时要执行的操作。 用户编写的应用程序必须先检查此属性，然后再将跟踪路由消息放在其目标队列上。 值可以是以下任意值:

MQROUTE_DELIVER_YES
到达时，会将跟踪路由消息放在目标队列上。 在目标队列上执行获取操作的任何应用程序都可以检索跟踪路由消息。

MQROUTE_DELIVER_NO
到达时，不会将跟踪路由消息传递到目标队列。 将根据消息的报告选项来处理该消息。

为跟踪路由应答消息设置公共队列

要在将报告传递到本地系统队列时确定与特定消息相关的跟踪路由应答消息的位置，在单个节点上使用公共队列更有效

开始之前

设置 **ROUTEREC** 参数以启用队列管理器进行跟踪路由消息传递，并指定将生成的任何跟踪路由应答消息传递到本地系统队列 SYSTEM.ADMIN.TRACE.ROUTE.QUEUE。

关于此任务

如果将队列管理器网络中的多个队列管理器设置为将跟踪路由应答消息传递到本地系统队列，那么确定与特定消息相关的跟踪路由应答消息的位置可能很耗时。 或者，使用单个节点，该节点是托管公共队列的队列管理器。 队列管理器网络中的所有队列管理器都可以将跟踪路由应答消息传递到此公共队列。 使用公共队列的好处是队列管理器不必将跟踪路由应答消息传递到消息中指定的应答队列，并且在确定与消息相关的跟踪路由应答消息的位置时，仅查询一个队列。

要设置公共队列，请执行以下步骤:

过程

1. 选择或定义队列管理器作为单个节点
2. 在单个节点上，选择或定义要用作公共队列的队列
3. 在将跟踪路由应答消息转发到公共队列的所有队列管理器上，重新定义本地系统队列 SYSTEM.ADMIN.TRACE.ROUTE.QUEUE 作为远程队列定义
   a) 将单个节点的名称指定为远程队列管理器名称
   b) 将公共队列的名称指定为远程队列名称

获取和使用记录的信息

使用下列任何方法来获取跟踪路由消息的记录活动信息

请注意，未获取活动信息的情况也适用于跟踪路由应答消息。

当针对活动记录和跟踪路由消息传递禁用的队列管理器处理跟踪路由消息时，不会记录活动信息。

从跟踪路由应答消息获取信息
要获取活动信息，请找到跟踪路由应答消息。 然后检索消息并分析活动信息。

关于此任务

仅当您知道跟踪路由应答消息的位置时，才能从跟踪路由应答消息获取活动信息。 找到消息并处理活动信息，如下所示:

过程

1. 检查在跟踪路由消息的消息描述符中指定的应答队列。 如果跟踪路由应答消息不在应答队列上，请检查以下位置:
   - 本地系统队列 SYSTEM.ADMIN.TRACE.ROUTE.QUEUE，在跟踪路由消息的目标队列管理器上
   - 公共队列 (如果已为跟踪路由应答消息设置公共队列)
   - 本地系统队列 SYSTEM.ADMIN.TRACE.ROUTE.QUEUE，在队列管理器网络中的任何其他队列管理器上，如果跟踪路由消息已放入死信队列，或者超过最大活动数，那么可能会发生此情况
2. 检索跟踪路由应答消息

3. 使用 IBM MQ 显示路径应用程序来显示记录的活动信息
4. 研究活动信息，获取您需要的信息

从跟踪路由消息获取信息
要获取活动信息，请找到跟踪路由消息，该消息必须在 *TraceRoute* PCF 组中具有相应的参数。 然后检索消息并分析活动信息。

关于此任务

仅当您知道跟踪路由消息的位置并且它在 *TraceRoute* PCF 组中具有指定为 MQROUTE_ACCUMULATE_IN_MSG 或 MQROUTE_ACCUMULATE_AND_REPLY 的参数 累计 时，才能从跟踪路由消息获取活动信息。

要将跟踪路由消息传递到目标队列，必须将 *TraceRoute* PCF 组中的 *Deliver* 参数指定为 MQROUTE_DELIVER_YES。

过程

1. 检查目标队列。 如果跟踪路由消息不在目标队列上，那么可以尝试使用启用了活动记录的跟踪路由消息来查找跟踪路由消息。 通过生成的活动报告，尝试确定跟踪路由消息的最后已知位置。
2. 检索跟踪路由消息
3. 使用 IBM MQ 显示路径应用程序来显示记录的活动信息
4. 研究活动信息，获取您需要的信息

从活动报告获取信息
要获取活动信息，请找到必须在消息描述符中指定报告选项的活动报告。 然后检索活动报告并分析活动信息。

关于此任务

仅当您知道活动报告的位置并且在跟踪路由消息的消息描述符中指定了报告选项 MQRO_ACTIVITY 时，才能从活动报告获取活动信息。

过程

1. 找到为跟踪路由消息生成的活动报告并对其进行排序。

   找到活动报告后，可以手动对其进行排序，也可以使用 IBM MQ 显示路径应用程序自动对活动信息进行排序和显示。
2. 研究活动信息，获取您需要的信息

其他活动信息

通过队列管理器网络路由跟踪路由消息时，用户应用程序可以通过在将 *Activity* 组写入跟踪路由消息或活动报告的消息数据时包含一个或多个额外的 PCF 参数来记录其他信息。

其他活动信息可帮助系统管理员识别跟踪路由消息所采用的路由或采用该路由的原因。

如果使用 IBM MQ 显示路由应用程序来显示跟踪路由消息的记录信息，那么除非 IBM MQ 显示路由应用程序识别每个参数的参数标识，否则只能使用数字标识显示任何其他 PCF 参数。 要识别参数标识，必须使用以下 PCF 参数记录其他信息。 将这些 PCF 参数包括在 活动 PCF 组中的相应位置。

GroupName

表 *13:* 组名

描述	**用于指定其他信息的分组参数。**
标识	MQGACF_VALUE_NAMING。
数据类型	MQCFGR

表 *13:* 组名 *(*继续*)*	
描述	**用于指定其他信息的分组参数。**
组中的参数	*ParameterName* *ParameterValue*

ParameterName

表 *14:* 参数名	
描述	**包含要由 IBM MQ 显示路由应用程序显示的名称，这会将 *ParameterValue* 的值放入上下文中。**
标识	MQCA_VALUE_NAME。
数据类型	MQCFST
包含在 PCF 组中:	*GroupName*.
值:	要显示的名称。

ParameterValue

表 *15:* 参数值	
描述	**包含要由 IBM MQ 显示路由应用程序显示的值。**
标识:	附加信息的 PCF 结构标识。
数据类型:	附加信息的 PCF 结构数据类型。
包含在 PCF 组中:	*GroupName*.
值:	要显示的值。

记录其他活动信息的示例

以下示例说明用户应用程序如何在代表跟踪路由消息执行活动时记录其他信息。 在这两个示例中， IBM MQ 显示路由应用程序用于生成跟踪路由消息，并显示返回给它的活动信息。

记录其他活动信息*:* 示例 *1*

其他活动信息由用户应用程序以格式记录，其中参数标识 不 由 IBM MQ 显示路由应用程序识别。

1. IBM MQ 显示路由应用程序用于生成跟踪路由消息并将其放入队列管理器网络中。 设置了必需的选项以请求以下内容:
   - 活动信息在跟踪路由消息的消息数据中累积。
   - 到达目标队列时，将废弃跟踪路由消息，并生成跟踪路由应答消息并将其传递到指定的应答队列。
   - 接收到跟踪路由应答消息时， IBM MQ 显示路由应用程序将显示累积的活动信息。

   跟踪路由消息将放入队列管理器网络中。
2. 当通过队列管理器网络路由跟踪路由消息时，为跟踪路由消息传递启用的用户应用程序将代表消息执行低详细信息活动。 除了将标准活动信息写入跟踪路由消息之外，用户应用程序还会将以下 PCF 参数写入 Activity 组的末尾:

   ColorValue

   标识
   65536

   数据类型
   MQCFST

值
"红色"

此附加 PCF 参数提供有关已执行的活动的进一步信息，但是将以 IBM MQ 显示路由应用程序 无法 识别参数标识的格式进行写入。

3. 跟踪路由消息到达目标队列，并且将向 IBM MQ 显示路由应用程序返回跟踪路由应答消息。 其他活动信息如下所示:

```
65536: 'Red'
```

IBM MQ 显示路由应用程序无法识别 PCF 参数的参数标识，并将其显示为数字值。 附加信息的上下文不清楚。

有关 IBM MQ 显示路由应用程序何时识别 PCF 参数的参数标识的示例，请参阅 第 69 页的『记录其他活动信息: 示例 2』。

记录其他活动信息: 示例 2

其他活动信息由用户应用程序以 IBM MQ 显示路径应用程序识别参数标识的格式记录。

1. IBM MQ 显示路由应用程序用于以与 第 68 页的『记录其他活动信息: 示例 1』中相同的方式生成跟踪路由消息并将其放入队列管理器网络中。
2. 当通过队列管理器网络路由跟踪路由消息时，为跟踪路由消息传递启用的用户应用程序将代表消息执行低详细信息活动。 除了将标准活动信息写入跟踪路由消息之外，用户应用程序还会将以下 PCF 参数写入 Activity 组的末尾:

ColorInfo

表 *16:* 颜色信息	
描述	**用于指定有关颜色的信息的分组参数。**
标识:	MQGACF_VALUE_NAMING。
数据类型:	MQCFGR。
组中的参数:	*ColorName* *ColorValue*

ColorName

表 *17:* 颜色名称	
描述	**包含要由 IBM MQ 显示路由应用程序显示的名称，该应用程序将 *ColorValue* 的值放入上下文中。**
标识:	MQCA_VALUE_NAME。
数据类型:	MQCFST。
包含在 PCF 组中:	*ColorInfo*.
值:	"颜色"

ColorValue

表 *18:* 颜色值	
描述	**包含要由 IBM MQ 显示路由应用程序显示的值。**
标识:	65536.
数据类型:	MQCFST。
包含在 PCF 组中:	*ColorInfo*.

表 18: 颜色值 (继续)	
描述	**包含要由 IBM MQ 显示路由应用程序显示的值。**
值:	"红色"

这些额外的 PCF 参数提供有关已执行的活动的进一步信息。 这些 PCF 参数以格式编写，其中参数标识由 IBM MQ 显示路由应用程序识别。

3. 跟踪路由消息到达目标队列，并且将向 IBM MQ 显示路由应用程序返回跟踪路由应答消息。 其他活动信息如下所示:

```
Color: 'Red'
```

IBM MQ 显示路由应用程序识别包含其他活动信息的值的 PCF 结构的参数标识具有相应的名称。 将显示相应的名称，而不是数字值。

IBM MQ 显示路由应用程序

使用 IBM MQ 显示路由应用程序 (**dspmqrte**) 使用命令行界面处理与跟踪路由消息相关的跟踪路由消息和活动信息。 z/OS IBM MQ 显示路由应用程序未在 IBM MQ for z/OS 上交付，但您可以从分布式安装运行该应用程序，并通过在发出 **dspmqrte** 命令时指定 **-c** 参数将其作为客户机连接到 IBM MQ for z/OS 队列管理器。

您可以将 IBM MQ dspmqrte 显示路由应用程序用于以下目的:

- 用于配置，生成跟踪路由消息并将其放入队列管理器网络中。

  通过将跟踪路由消息放入队列管理器网络中，可以收集并使用活动信息来确定跟踪路由消息所采用的路由。 您可以按如下所示指定跟踪路由消息的特征:

  - 跟踪路由消息的目标。
  - 跟踪路由消息如何模拟另一条消息。
  - 在通过队列管理器网络路由跟踪路由消息时应如何处理该消息。
  - 是使用活动记录还是跟踪路由消息传递来记录活动信息。
- 用于对与跟踪路由消息相关的活动信息进行排序和显示。

  如果 IBM MQ 显示路由应用程序已将跟踪路由消息放入队列管理器网络中，那么在返回相关活动信息后，可以立即对该信息进行排序和显示。 或者，可以使用 IBM MQ 显示路由应用程序来订购和显示与先前生成的跟踪路由消息相关的活动信息。

相关参考

长石

跟踪路由消息的参数

使用此页面来获取 IBM MQ 显示路由应用程序 **dspmqrte** 提供的参数的概述，以确定跟踪路由消息的特征，包括如何将其视为通过队列管理器网络进行路由。

相关参考

长石

队列管理器连接

使用此页面来指定 IBM MQ 显示路由应用程序连接到的队列管理器

-c

指定 IBM MQ 显示路由应用程序作为客户机应用程序进行连接。

如果未指定此参数，那么 IBM MQ 显示路由应用程序不会作为客户机应用程序进行连接。

-m *QMgrName*

IBM MQ 显示路由应用程序连接到的队列管理器的名称。 该名称最多可以包含 48 个字符。

如果未指定此参数，那么将使用缺省队列管理器。

目标目标
使用此页面来指定跟踪路由消息的目标

-q *TargetQName*
如果使用 IBM MQ 显示路由应用程序将跟踪路由消息发送到队列管理器网络，那么 *TargetQName* 指定目标队列的名称。

-ts *TargetTopic* 字符串
指定主题字符串。

-qm *TargetQMgr*
限定目标目标; 然后将应用正常队列管理器名称解析。 目标目标与 *-q TargetQName* 或 *-ts TargetTopicString* 一起指定。

如果未指定此参数，那么 IBM MQ 显示路由应用程序所连接的队列管理器将用作目标队列管理器。

-o
指定目标目标未绑定到特定目标。 通常，当要在集群中放置跟踪路由消息时，将使用此参数。 使用选项 MQOO_BIND_NOT_FIXED 打开目标目标。

如果未指定此参数，那么目标目标将绑定到特定目标。

发布主题
对于发布/预订应用程序，使用此页面来指定要发布的 IBM MQ 显示路由应用程序的跟踪路由消息的主题字符串

-ts *TopicName*
指定 IBM MQ 显示路由应用程序要将跟踪路由消息发布到的主题字符串，并将此应用程序置于主题方式。 在此方式下，应用程序将跟踪发布请求产生的所有消息。

您还可以使用 IBM MQ 显示路由应用程序来显示针对发布消息生成的活动报告的结果。

消息模拟
使用此页面来配置跟踪路由消息以模拟消息，例如，当原始消息未到达其预期目标时

跟踪路由消息传递的一个用途是帮助确定未到达其预期目标的消息的最后已知位置。 IBM MQ 显示路由应用程序提供了可帮助配置跟踪路由消息以模拟原始消息的参数。 模拟消息时，可以使用以下参数:

-l 持久性
指定生成的跟踪路由消息的持久性。 持久性 的可能值为:

yes
生成的跟踪路由消息是持久消息。 (MQPER_PERSISTENT)。

False
生成的跟踪路由消息 **不是** 持久消息。 (MQPER_NOT_PERSISTENT)。

q
生成的跟踪路由消息从 *-q TargetQName* 或 *-ts TargetTopicString* 指定的目标继承其持久性值。 (MQPER_PERSISTENCE_AS_Q_DEF)。

返回的跟踪路由应答消息或任何报告消息将与原始跟踪路由消息共享相同的持久性值。

如果 持久性 指定为 **yes**，那么必须指定参数 *-rq ReplyToQ*。 应答队列不能解析为临时动态队列。

如果未指定此参数，那么生成的跟踪路由消息 **不是** 持久消息。

-p 优先级
指定跟踪路由消息的优先级。 优先级 的值大于或等于 0 或 MQPRI_PRIORITY_AS_Q_DEF。 MQPRI_PRIORITY_AS_Q_DEF 指定从 *-q TargetQName* 或 *-ts TargetTopicString* 指定的目标获取优先级值。

如果未指定此参数，那么将从 *-q TargetQName* 或 *-ts TargetTopicString* 指定的目标获取优先级值。

-xs 到期
指定跟踪路由消息的到期时间 (以秒计)。

如果未指定此参数，那么到期时间将指定为 60 秒。

-ro 无 |*ReportOption*

none
指定不设置任何报告选项。

ReportOption
指定跟踪路由消息的报告选项。 可以使用逗号作为分隔符来指定多个报告选项。 *ReportOption* 的可能值为:

活动
设置了报告选项 MQRO_ACTIVITY。

COA
设置了报告选项 MQRO_COA_WITH_FULL_DATA。

COD
设置了报告选项 MQRO_COD_WITH_FULL_DATA。

异常
设置了报告选项 MQRO_EXCEPTION_WITH_FULL_DATA。

截止
设置了报告选项 MQRO_EXPIRATION_WITH_FULL_DATA。

丢弃
已设置报告选项 MQRO_DISCARD_MSG。

如果既未指定 *-ro ReportOption* 也未指定 *-ro none* ，那么将指定 MQRO_ACTIVITY 和 MQRO_DISCARD_MSG 报告选项。

IBM MQ 显示路由应用程序不允许您将用户数据添加到跟踪路由消息。 如果需要将用户数据添加到跟踪路由消息，那么必须手动生成跟踪路由消息。

记录的活动信息
使用此页面来指定用于返回记录的活动信息的方法，然后可以使用此方法来确定跟踪路由消息所采用的路由。

可以按如下所示返回记录的活动信息:

- 在活动报告中
- 在跟踪路由应答消息中
- 在跟踪路由消息本身中 (已放在目标队列上)

使用 **dspmqrte** 时，将使用以下参数来确定用于返回记录的活动信息的方法:

-ro 活动
指定使用活动报告返回活动信息。 缺省情况下，已启用活动记录。

-ac-ar
指定在跟踪路由消息中累积活动信息，并生成跟踪路由应答消息。

-ac
指定要在跟踪路由消息中累积活动信息。

如果未指定此参数，那么不会在跟踪路由消息中累积活动信息。

-阿尔
请求在以下情况下生成包含所有累积活动信息的跟踪路由应答消息:

- IBM MQ 队列管理器将废弃跟踪路由消息。
- IBM MQ 队列管理器将跟踪路由消息放入本地队列 (目标队列或死信队列)。
- 对跟踪路由消息执行的活动数超过 *-s* 活动中指定的值。

-ac -d 是
指定在跟踪路由消息中累积活动信息，并且在到达时将跟踪路由消息放在目标队列上。

-ac
指定要在跟踪路由消息中累积活动信息。

如果未指定此参数，那么不会在跟踪路由消息中累积活动信息。

-d 是
到达时，会将跟踪路由消息放入目标队列，即使队列管理器不支持跟踪路由消息传递也是如此。

如果未指定此参数，那么不会将跟踪路由消息放入目标队列。

然后，可以从目标队列中检索跟踪路由消息，并获取记录的活动信息。

您可以根据需要组合这些方法。

此外，可以使用以下参数指定记录的活动信息的详细级别:

-t 详细信息
指定记录的活动。 详细信息 的可能值为:

low
仅记录用户定义的应用程序执行的活动。

介质
将记录以低值指定的活动。 此外，还会记录由 MCA 执行的发布活动和活动。

high
记录以低值和中值指定的活动。 MCA 不会在此详细信息级别公开任何进一步的活动信息。 此选项可供仅公开进一步活动信息的用户定义应用程序使用。 例如，如果用户定义的应用程序通过考虑某些消息特征来确定消息所采用的路由，那么可以将路由逻辑包含在此详细信息级别中。

如果未指定此参数，那么将记录中等级别的活动。

缺省情况下， IBM MQ 显示路由应用程序使用临时动态队列来存储返回的消息。 当 IBM MQ 显示路由应用程序结束时，将关闭临时动态队列，并且将清除所有消息。 如果在当前执行的 IBM MQ 显示路由应用程序结束之后需要返回的消息，那么必须使用以下参数指定永久队列:

-rq *ReplyToQ*
指定要将对跟踪路由消息的所有响应发送到的应答队列的名称。 如果跟踪路由消息是持久的，或者如果指定了 *-n* 参数，那么必须指定不是临时动态队列的应答队列。

如果未指定此参数，那么将使用系统缺省模型队列 SYSTEM.DEFAULT.MODEL.QUEUE。

-rqm *ReplyTo* 队列管理器
指定应答队列所在的队列管理器的名称。 该名称最多可以包含 48 个字符。

如果未指定此参数，那么将使用 IBM MQ 显示路由应用程序所连接的队列管理器作为应答队列管理器。

如何处理跟踪路由消息
使用此页面来控制在通过队列管理器网络进行路由时如何处理跟踪路由消息。

以下参数可限制可在队列管理器网络中路由跟踪路由消息的位置:

-d 交付
指定是否在到达时将跟踪路由消息传递到目标队列。 交付 的可能值为:

yes
到达时，会将跟踪路由消息放入目标队列，即使队列管理器不支持跟踪路由消息传递也是如此。

False
到达时，不会将跟踪路由消息放入目标队列。

如果未指定此参数，那么不会将跟踪路由消息放入目标队列。

-f 转发
指定可将跟踪路由消息转发到的队列管理器的类型。 有关队列管理器用于确定是否将消息转发到远程队列管理器的算法的详细信息，请参阅 第 63 页的『TraceRoute PCF 组』。 正向 的可能值为:

全部
跟踪路由消息将转发到任何队列管理器。

警告: 如果转发到早于 IBM WebSphere MQ 6.0 的队列管理器，那么将无法识别跟踪路由消息，并且可以将该消息传递到本地队列，而不使用 *-d Deliver* 参数的值。

受支持
跟踪路由消息仅转发到将采用 *TraceRoute* PCF 组中的 *Deliver* 参数的队列管理器

如果未指定此参数，那么跟踪路由消息将仅转发到将使用 *Deliver* 参数的队列管理器。

以下参数可防止跟踪路由消息无限期地保留在队列管理器网络中:

-s 活动
指定在废弃跟踪路由消息之前可以代表该消息执行的已记录活动的最大数目。 这将防止在无限循环中捕获到跟踪路由消息时无限期转发该消息。 活动 的值大于或等于 1 或 MQROUTE_UNLIMITED_ACTIVITIES。 MQROUTE_UNLIMITED_ACTIVITIES 指定可以代表 trace-route 消息执行无限数量的活动。

如果未指定此参数，那么可以代表跟踪路由消息执行无限数量的活动。

-xs 到期
指定跟踪路由消息的到期时间 (以秒计)。

如果未指定此参数，那么到期时间将指定为 60 秒。

-xp *PassExpiry*
指定是否将来自跟踪路由消息的到期时间传递到跟踪路由应答消息。 *PassExpiry* 的可能值为:

yes
在 trace-route 消息的消息描述符中指定了报告选项 MQRO_PASS_DISCARD_AND_到期。

如果为跟踪路由消息生成了跟踪路由应答消息或活动报告，那么将传递 MQRO_DISCARD 报告选项 (如果已指定) 以及剩余的到期时间。

这是缺省值。

False
未指定报告选项 MQRO_PASS_DISCARD_AND_到期。

如果为跟踪路由消息生成了跟踪路由应答消息，那么 **不会** 传递来自跟踪路由消息的废弃选项和到期时间。

如果未指定此参数，那么不会指定 MQRO_PASS_DISCARD_AND_到期。

-ro 废弃
指定 MQRO_DISCARD_MSG 报告选项。 这可防止跟踪路由消息无限期地保留在队列管理器网络中。

显示活动信息

IBM MQ 显示路由应用程序可以显示其刚刚放入队列管理器网络的跟踪路由消息的活动信息，也可以显示先前生成的跟踪路由消息的活动信息。 它还可以显示用户编写的应用程序记录的其他信息。

要指定是否显示针对跟踪路由消息返回的活动信息，请指定以下参数:

-n
指定不显示针对跟踪路由消息返回的活动信息。

如果此参数随附对跟踪路由应答消息 (*-ar*) 或从 (*-ro ReportOption*) 生成选项的任何报告的请求，那么必须使用 *-rq ReplyToQ* 指定特定 (非模型) 应答队列。 缺省情况下，仅请求活动报告消息。

将跟踪路由消息放入指定的目标队列后，将显示包含跟踪路由消息的消息标识的 48 个字符的十六进制字符串。 IBM MQ 显示路由应用程序可以使用消息标识，以便稍后使用 *-i CorrelId* 参数显示跟踪路由消息的活动信息。

如果未指定此参数，那么将以 *-v* 参数指定的格式显示针对跟踪路由消息返回的活动信息。

显示刚刚放入队列管理器网络中的跟踪路由消息的活动信息时，可以指定以下参数:

-w *WaitTime*
指定 IBM MQ 显示路由应用程序将等待活动报告或跟踪路由应答消息返回到指定应答队列的时间 (以秒计)。

如果未指定此参数，那么会将等待时间指定为跟踪路由消息的到期时间加上 60 秒。

显示先前累积的活动信息时，必须设置以下参数:

-q *TargetQName*
如果正在使用 IBM MQ 显示路由应用程序来查看先前收集的活动信息，那么 *TargetQName* 指定存储活动信息的队列的名称。

-i *CorrelId*
当 IBM MQ 显示路由应用程序用于仅显示先前累积的活动信息时，将使用此参数。 在由 *-q TargetQName* 指定的队列上可以有许多活动报告和跟踪路由应答消息。 *CorrelId* 用于标识与跟踪路由消息相关的活动报告或跟踪路由应答消息。 在 *CorrelId* 中指定原始跟踪路由消息的消息标识。

CorrelId 的格式是 48 个字符的十六进制字符串。

当显示先前累积的活动信息或显示跟踪路由消息的当前活动信息时，可以使用以下参数:

-b
指定 IBM MQ 显示路由应用程序将仅浏览与消息相关的活动报告或跟踪路由应答消息。 这允许稍后再次显示活动信息。

如果未指定此参数，那么 IBM MQ 显示路由应用程序将以破坏性方式获取与消息相关的活动报告或跟踪路由应答消息。

-v 摘要 | all | none | 大纲 *DisplayOption*

摘要
将显示跟踪路由消息经过的队列。

全部
显示所有可用信息。

none
未显示任何信息。

大纲 *DisplayOption*
指定跟踪路由消息的显示选项。 可以使用逗号作为分隔符来指定多个显示选项。

如果未提供任何值，那么将显示以下内容:

- 应用程序名称
- 每个操作的类型
- 任何特定于操作的参数

DisplayOption 的可能值为:

活动
将显示 活动 PCF 组中的所有非 PCF 组参数。

标识
将显示具有参数标识 MQBACF_MSG_ID 或 MQBACF_CORREL_ID 的值。 这将覆盖 *msgdelta*。

消息
将显示 消息 PCF 组中的所有非 PCF 组参数。 指定此值时，不能指定 *msgdelta*。

消息增量
将显示 消息 PCF 组中自上次操作以来已更改的所有非 PCF 组参数。 指定此值时，不能指定 *message*。

操作
将显示 操作 PCF 组中的所有非 PCF 组参数。

跟踪路由
将显示 *TraceRoute* PCF 组中的所有非 PCF 组参数。

如果未指定此参数，那么将显示消息路由的摘要。

显示其他信息
当通过队列管理器网络路由跟踪路由消息时，用户编写的应用程序可以通过将一个或多个附加 PCF 参数写入跟踪路由消息的消息数据或活动报告的消息数据来记录附加信息。 要使 IBM MQ 显示路由应用程序以可读形式显示其他信息，必须以特定格式进行记录，如 第 67 页的『其他活动信息』中所述。

IBM MQ 显示路由应用程序示例

以下示例显示了如何使用 IBM MQ 显示路由应用程序。 在每个示例中，两个队列管理器 (QM1 和 QM2) 通过两个通道 (QM2.TO.QM1 和 QM1.TO.QM2)。

示例 *1*-请求活动报告
显示传递到目标队列的跟踪路由消息中的活动信息

在此示例中， IBM MQ 显示路由应用程序连接到队列管理器 QM1，并用于生成跟踪路由消息并将其传递到目标队列 TARGET.Q，在远程队列管理器 QM2 上。 指定了必需的报告选项，以便在路由跟踪路由应答消息时请求活动报告。 到达目标队列时，将废弃跟踪路由消息。 使用活动报告返回到 IBM MQ 显示路由应用程序的活动信息将按顺序放置并显示。

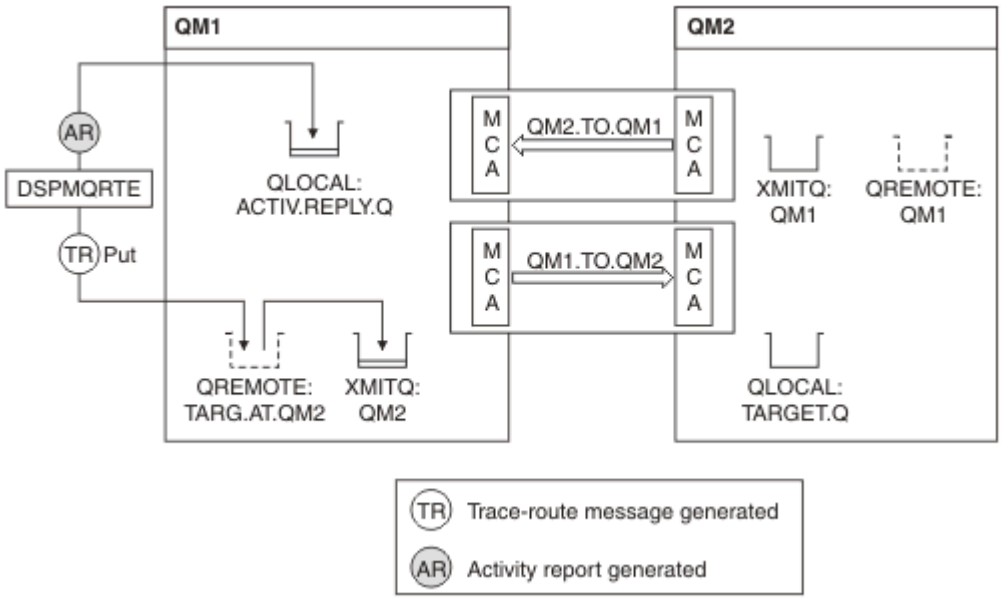


图 *9:* 请求活动报告，图 *1*

- 每个队列管理器 (QM1 和 QM2) 的 ACTIVREC 属性设置为 MSG。
- 发出以下命令:

```
dspmqrte -m QM1 -q TARG.AT.QM2 -rq ACTIV.REPLY.Q
```

  QM1 是 IBM MQ 显示路由应用程序连接到的队列管理器的名称 TARG.AT.QM2 是目标队列的名称，ACTIV.REPLY.Q 是请求将跟踪路由消息的所有响应发送到的队列的名称。

  对于未指定的所有选项，将采用缺省值，但请特别注意 -f 选项 (仅将 trace-route 消息转发到采用 TraceRoute PCF 组的 Deliver 参数的队列管理器)， -d 选项 (到达时，不会将 trace-route 消息放在目标队列上)， -ro 选项 (指定了 MQRO_ACTIVITY 和 MQRO_DISCARD_MSG 报告选项) 以及 -t 选项 (记录中等详细信息级别的活动)。
- DSPMQRTE 生成跟踪路由消息并将其放在远程队列 TARG.AT.QM2。
- 然后， DSPMQRTE 查看队列管理器 QM1 的 ACTIVREC 属性的值。 值为 MSG ，因此 DSPMQRTE 生成活动报告并将其放在应答队列 ACTIV.REPLY.Q。

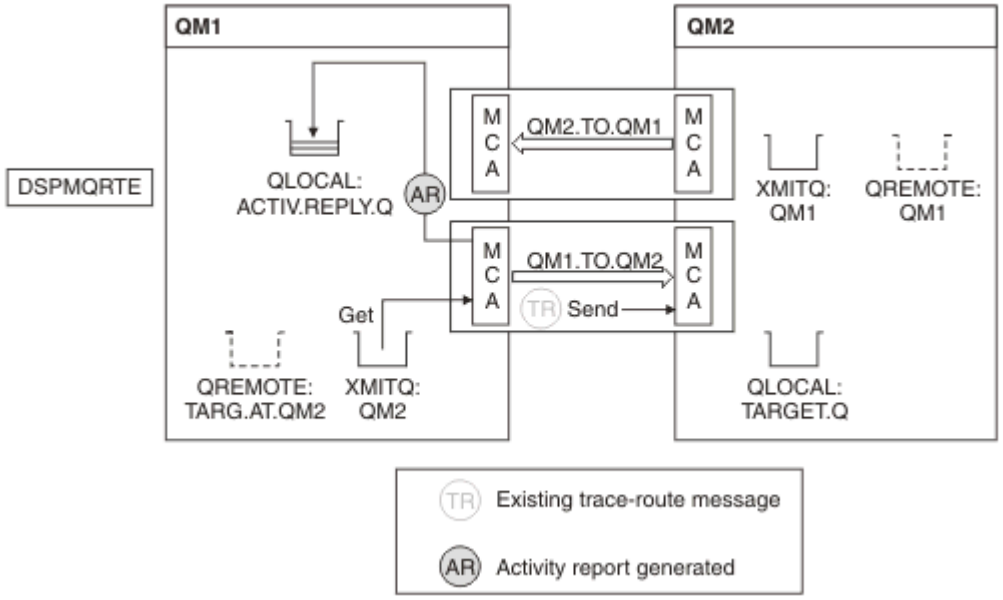


图 *10:* 请求活动报告，图 *2*

- 发送消息通道代理 (MCA) 从传输队列获取跟踪路由消息。 该消息是跟踪路由消息，因此 MCA 开始记录活动信息。
- 队列管理器 (QM1) 的 ACTIVREC 属性是 MSG ，并且在消息描述符的 "报告" 字段中指定了 MQRO_ACTIVITY 选项，因此 MCA 稍后将生成活动报告。 TraceRoute PCF 组中的 RecordedActivities 参数值将按 1 递增。
- MCA 会检查是否未超出 TraceRoute PCF 组中的 MaxActivities 值。
- 在将消息转发到 QM2 之前， MCA 遵循 转发 中描述的算法 (步骤 第 65 页的『1』， 第 65 页的『4』和 第 65 页的『5』) MCA 选择发送消息。
- 然后， MCA 生成活动报告并将其放入应答队列 (ACTIV.REPLY.Q)。

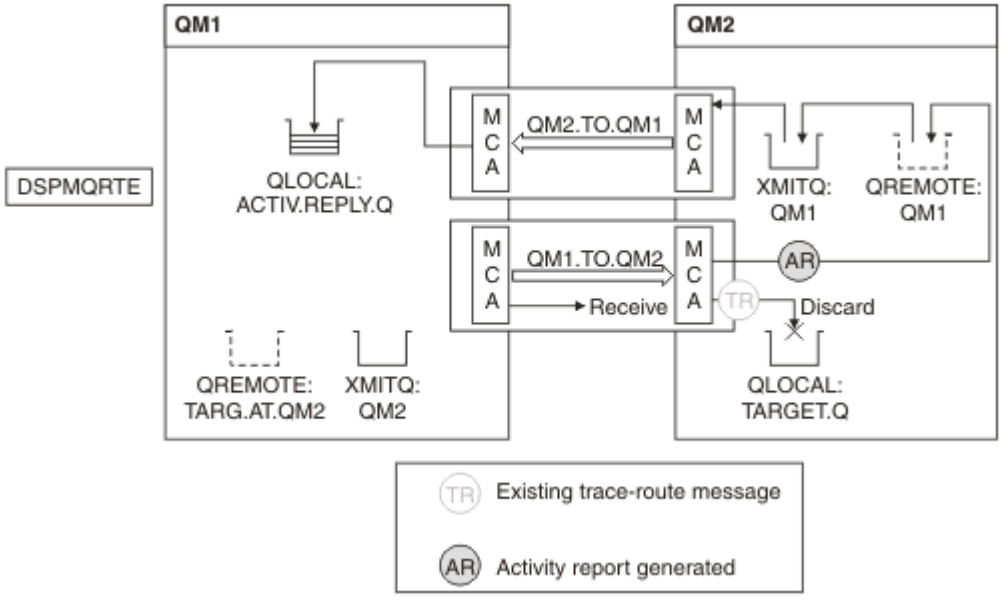


图 *11:* 请求活动报告，图 *3*

- 接收 MCA 从通道接收跟踪路由消息。 该消息是跟踪路由消息，因此 MCA 开始记录有关活动的信息。

- 如果跟踪路由消息所来自的队列管理器是 IBM WebSphere MQ 5.3.1 或更早版本，那么 MCA 会将 TraceRoute PCF 的 DiscontinuityCount 参数递增 1。 这里不是这样。
- 队列管理器 (QM2) 的 ACTIVREC 属性为 MSG ，并且指定了 MQRO_ACTIVITY 选项，因此 MCA 将生成活动报告。 RecordedActivities 参数值将按 1 递增。
- 目标队列是本地队列，因此将根据 TraceRoute PCF 组中的 "交付" 参数值随反馈 MQFB_NOT_DELIVER 一起废弃消息。
- 然后， MCA 生成最终活动报告并将其放入应答队列。 这将解析为与队列管理器 QM1 相关联的传输队列，并将活动报告返回到队列管理器 QM1 (ACTIV.REPLY.Q)。

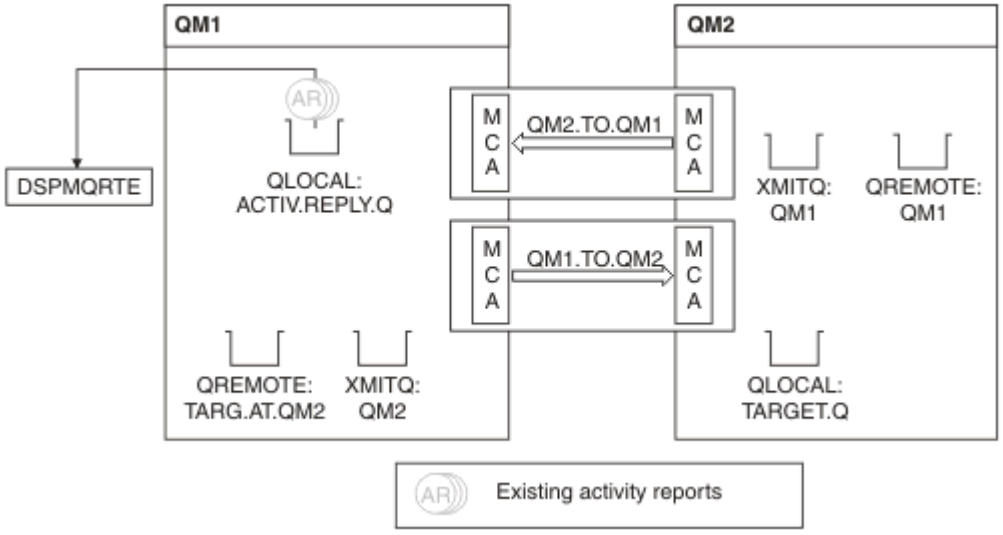


图 *12:* 请求活动报告，图 *4*

- 同时， DSPMQRTE 一直在应答队列 (ACTIV.REPLY.Q) ，正在等待活动报告。 它将等待最多 120 秒 (比跟踪路由消息的到期时间长 60 秒) ，因为当 DSPMQRTE 启动时未指定 -w。
- DSPMQRTE 从应答队列获取 3 活动报告。
- 对于每个活动，活动报告使用 TraceRoute PCF 组中的 RecordedActivities， UnrecordedActivities 和 DiscontinuityCount 参数进行排序。 此示例中非零的唯一值是 RecordedActivities，因此这是实际使用的唯一参数。
- 一旦显示废弃操作，程序就会立即结束。 尽管最终操作是废弃操作，但由于反馈是 MQFB_NOT_交付的，因此会将其视为已执行的放置操作。

  显示的输出如下:

```
AMQ8653: DSPMQRTE command started with options '-m QM1 -q TARG.AT.QM2
 -rq ACTIV.REPLY.Q'.
AMQ8659: DSPMQRTE command successfully put a message on queue 'QM2',
 queue manager 'QM1'.
AMQ8674: DSPMQRTE command is now waiting for information to display.
AMQ8666: Queue 'QM2' on queue manager 'QM1'.
AMQ8666: Queue 'TARGET.Q' on queue manager 'QM2'.
AMQ8652: DSPMQRTE command has finished.
```

示例 *2*-请求跟踪路由应答消息
生成跟踪路由消息并将其传递到目标队列

在此示例中， IBM MQ 显示路由应用程序连接到队列管理器 QM1，并用于生成跟踪路由消息并将其传递到目标队列 TARGET.Q，在远程队列管理器 QM2 上。 指定了必需的选项，以便在跟踪路由消息中累积活动信息。 到达目标队列时，将请求跟踪路由应答消息，并且将废弃跟踪路由消息。

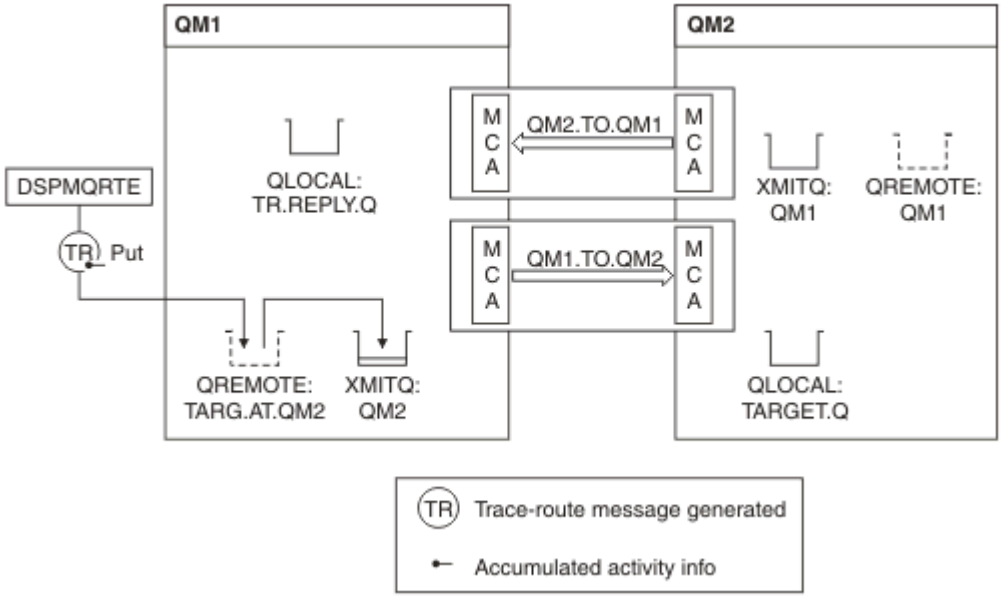


图 *13:* 请求跟踪路由应答消息，图 *1*

- 每个队列管理器 (QM1 和 QM2) 的 ROUTEREC 属性设置为 MSG。
- 发出以下命令:

```
dspmqrte -m QM1 -q TARG.AT.QM2 -rq TR.REPLY.Q -ac -ar -ro discard
```

  QM1 是 IBM MQ 显示路由应用程序连接到的队列管理器的名称 TARG.AT.QM2 是目标队列的名称，ACTIV.REPLY.Q 是请求将跟踪路由消息的所有响应发送到的队列的名称。 -ac 选项指定在跟踪路由消息中累积活动信息， -ar 选项指定将所有累积活动发送到由 -rq 选项 (即 TR.REPLY.Q)。 -ro 选项指定设置了报告选项 MQRO_DISCARD_MSG ，这意味着在此示例中不会生成活动报告。
- DSPMQRTE 在将消息放在目标路由上之前，在跟踪路由消息中累积活动信息。 队列管理器属性 ROUTEREC 不得为 DISABLED ，否则将发生此情况。

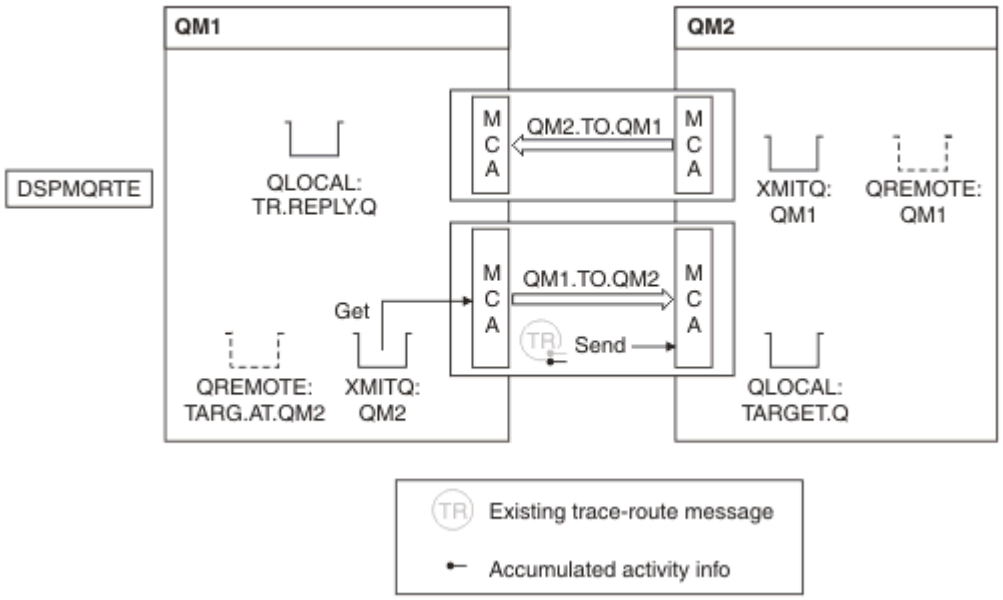


图 *14:* 请求跟踪路由应答消息，图 *2*

- 消息是跟踪路由消息，因此发送 MCA 开始记录有关活动的信息。

- QM1 上的队列管理器属性 ROUTEREC 未禁用，因此在将消息转发到队列管理器 QM2 之前， MCA 会累积消息中的活动信息。

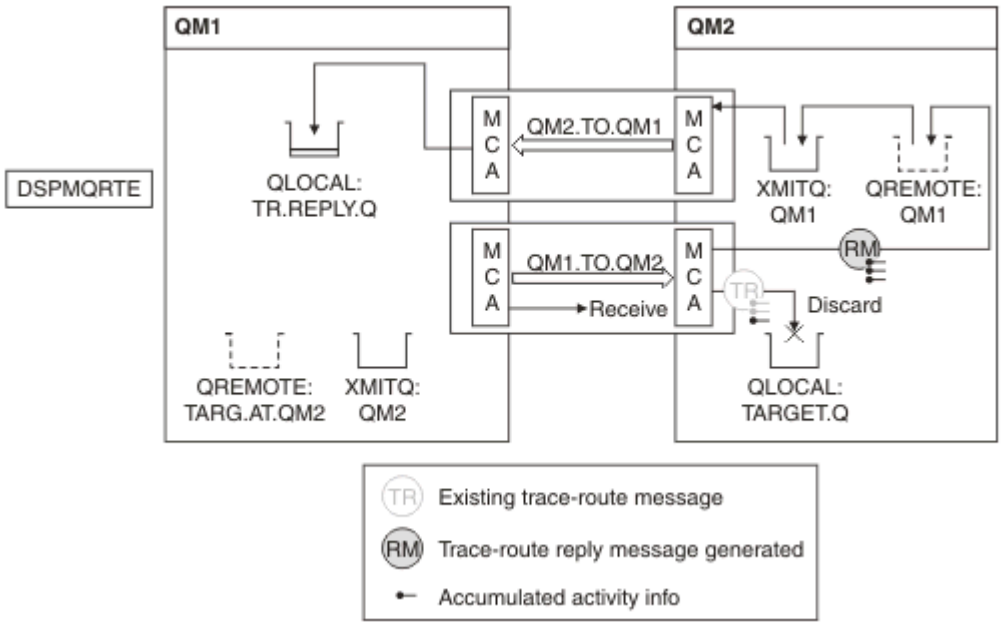


图 *15:* 请求跟踪路由应答消息，图 *3*

- 该消息是跟踪路由消息，因此接收 MCA 开始记录有关活动的信息。
- QM2 上的队列管理器属性 ROUTEREC 未处于 DISABLED 状态，因此 MCA 会累积消息中的信息。
- 目标队列是本地队列，因此将根据 TraceRoute PCF 组中的 "交付" 参数值随反馈 MQFB_NOT_DELIVER 一起废弃消息。
- 这是将在消息上执行的最后一个活动，并且由于 QM1 上的队列管理器属性 ROUTEREC 未处于 DISABLED 状态，因此 MCA 将根据 "累积" 值生成跟踪路由应答消息。 ROUTEREC 的值为 MSG ，因此应答消息放在应答队列上。 应答消息包含来自跟踪路由消息的所有累积活动信息。

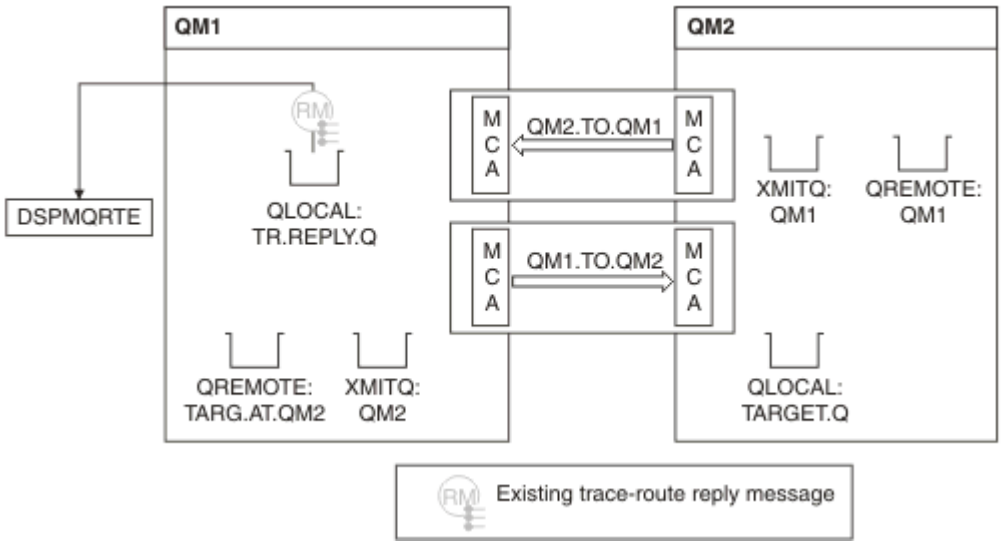


图 *16:* 请求跟踪路由应答消息，图 *4*

- 同时 DSPMQRTE 正在等待跟踪路由应答消息返回到应答队列。 当它返回时， DSPMQRTE 解析它包含的每个活动并将其打印出来。 最终操作是废弃操作。 DSPMQRTE 在打印后结束。

显示的输出如下:

```
AMQ8653: DSPMQRTE command started with options '-m QM1 -q TARG.AT.QM2 -rq
 TR.REPLY.Q'.
AMQ8659: DSPMQRTE command successfully put a message on queue 'QM2', queue
 manager 'QM1'.
AMQ8674: DSPMQRTE command is now waiting for information to display.
AMQ8666: Queue 'QM2' on queue manager 'QM1'.
AMQ8666: Queue 'TARGET.Q' on queue manager 'QM2'.
AMQ8652: DSPMQRTE command has finished.
```

示例 *3*-将活动报告交付到系统队列
检测何时将活动报告传递到除应答队列以外的队列，并使用 IBM MQ 显示路由应用程序从其他队列读取活动报告。

此示例与 第 76 页的『示例 1-请求活动报告』相同，但 QM2 现在将 ACTIVREC 队列管理属性的值设置为 QUEUE。 通道 QM1.TO.QM2 才能使其生效。

此示例演示如何检测何时将活动报告传递到除应答队列以外的队列。 一旦检测到， IBM MQ 显示路由应用程序将用于从另一个队列读取活动报告。

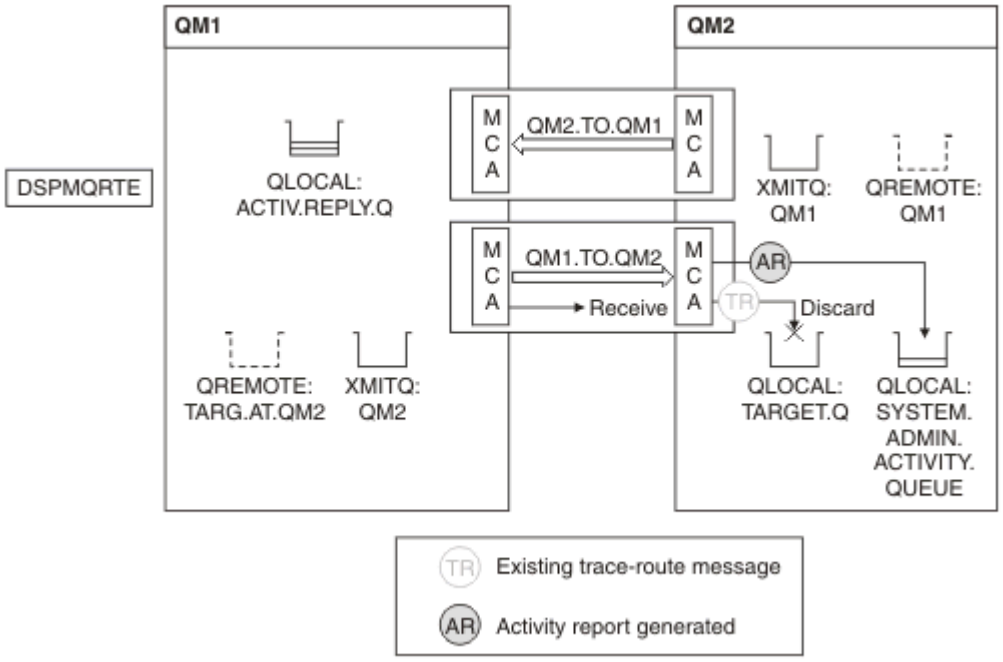


图 *17:* 正在将活动报告交付到系统队列，图 *1*

- 该消息是跟踪路由消息，因此接收 MCA 开始记录有关活动的信息。
- QM2 上 ACTIVREC 队列管理器属性的值现在为 QUEUE ，因此 MCA 会生成活动报告，但将其放在系统队列 (SYSTEM.ADMIN.ACTIVITY.QUEUE) ，不在应答队列 (ACTIV.REPLY.Q)。

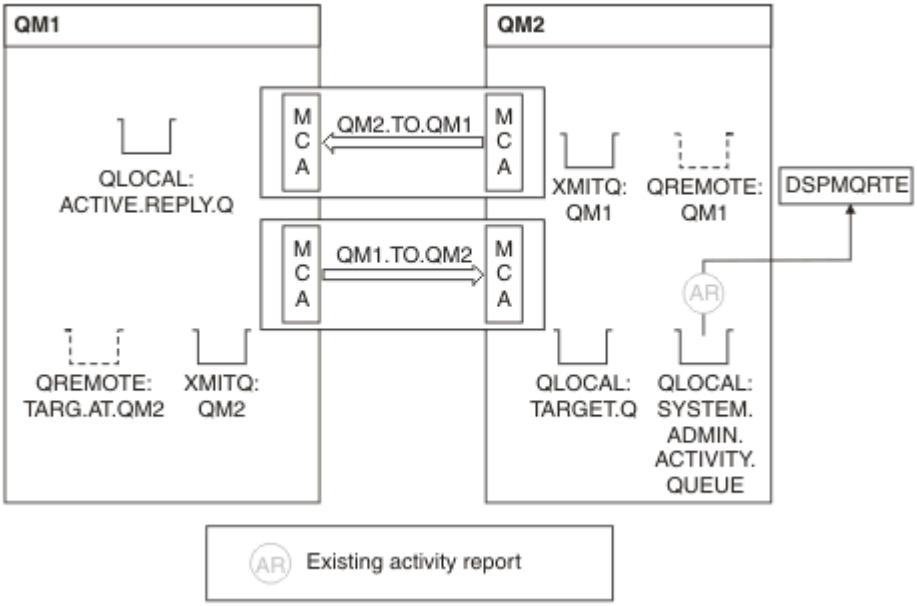


图 *18:* 正在将活动报告交付到系统队列，图 *2*

- 同时 DSPMQRTE 正在等待活动报告到达 ACTIV.REPLY.Q。 只有两个人来了 DSPMQRTE 继续等待 120 秒，因为似乎路由尚未完成。

显示的输出如下:

```
AMQ8653: DSPMQRTE command started with options '-m QM1 -q TARG.AT.QM2 -rq
          ACTIV.REPLY.Q -v outline identifiers'.
AMQ8659: DSPMQRTE command successfully put a message on queue 'QM2', queue
          manager 'QM1'.
AMQ8674: DSPMQRTE command is now waiting for information to display.
------------------------------------------------------------------------------
Activity:
 ApplName: 'cann\output\bin\dspmqrte.exe'

 Operation:
  OperationType: Put

  Message:

   MQMD:
    MsgId: X'414D51204C4152474551202020202020A3C9154220001502'
    CorrelId: X'414D51204C4152474551202020202020A3C9154220001503'
  QMgrName: 'QM1                                                '
  QName: 'TARG.AT.QM2                                          '
  ResolvedQName: 'QM2                                                  '
  RemoteQName: 'TARGET.Q                                              '
  RemoteQMgrName: 'QM2                                                 '
------------------------------------------------------------------------------
Activity:
 ApplName: 'cann\output\bin\runmqchl.EXE'

 Operation:
  OperationType: Get

  Message:

   MQMD:
    MsgId: X'414D51204C4152474551202020202020A3C9154220001505'
    CorrelId: X'414D51204C4152474551202020202020A3C9154220001502'

   EmbeddedMQMD:
    MsgId: X'414D51204C4152474551202020202020A3C9154220001502'
    CorrelId: X'414D51204C4152474551202020202020A3C9154220001503'
  QMgrName: 'QM1                                                '
  QName: 'QM2                                                  '
  ResolvedQName: 'QM2                                                  '

 Operation:
  OperationType: Send

  Message:

   MQMD:
    MsgId: X'414D51204C4152474551202020202020A3C9154220001502'
    CorrelId: X'414D51204C4152474551202020202020A3C9154220001503'
  QMgrName: 'QM1                                                '
  RemoteQMgrName: 'QM2                                                 '
  ChannelName: 'QM1.TO.QM2          '
  ChannelType: Sender
  XmitQName: 'QM2                                                  '
------------------------------------------------------------------------------
AMQ8652: DSPMQRTE command has finished.
```

- DSPMQRTE 观察到的最后一个操作是 "发送"，因此通道正在运行。 现在，我们必须确定为什么没有从队列管理器 QM2 (如 RemoteQMgr 名称中所标识) 接收任何其他活动报告。
- 要检查系统队列上是否有任何活动信息，请在 QM2 上启动 DSPMQRTE 以尝试收集更多活动报告。 使用以下命令来启动 DSPMQRTE:

```
dspmqrte -m QM2 -q SYSTEM.ADMIN.ACTIVITY.QUEUE
         -i 414D51204C4152474551202020202020A3C9154220001502 -v outline
```

  其中 414D51204C4152474551202020202020A3C9154220001502 是放入的跟踪路由消息的 MsgId 。
- 然后 DSPMQRTE 再次执行 MQGET 序列，等待与具有指定标识的跟踪路由消息相关的系统活动队列上的响应。
- DSPMQRTE 再获取一个它显示的活动报告。 DSPMQRTE 确定缺少先前的活动报告，并显示一条消息指出这一点。 不过我们已经知道这部分路线了。

显示的输出如下:

```
AMQ8653: DSPMQRTE command started with options '-m QM2
         -q SYSTEM.ADMIN.ACTIVITY.QUEUE
         -i 414D51204C4152474551202020202020A3C915420001502 -v outline'.
AMQ8674: DSPMQRTE command is now waiting for information to display.
--------------------------------------------------------------------------------

Activity:
 Activity information unavailable.

--------------------------------------------------------------------------------
Activity:
 ApplName: 'cann\output\bin\AMQRMPPA.EXE'

 Operation:
  OperationType: Receive
  QMgrName: 'QM2                                                '
  RemoteQMgrName: 'QM1                                                '
  ChannelName: 'QM1.TO.QM2          '
  ChannelType: Receiver

 Operation:
  OperationType: Discard
  QMgrName: 'QM2                                                '
  QName: 'TARGET.Q                                             '
  Feedback: NotDelivered

--------------------------------------------------------------------------------
AMQ8652: DSPMQRTE command has finished.
```

- 此活动报告指示路线信息现在已完成。 未发生问题。
- 仅仅因为路由信息不可用，或者由于 DSPMQRTE 无法显示所有路由，这并不意味着未传递消息。 例如，不同队列管理器的队列管理器属性可能不同，或者可能未定义应答队列以返回响应。

示例 *4*-诊断通道问题
诊断跟踪路由消息未到达目标队列的问题

在此示例中， IBM MQ 显示路由应用程序连接到队列管理器 QM1，生成跟踪路由消息，然后尝试将其传递到目标队列 TARGET.Q，在远程队列管理器 QM2 上。 在此示例中，跟踪路由消息未到达目标队列。 可用活动报告用于诊断问题。

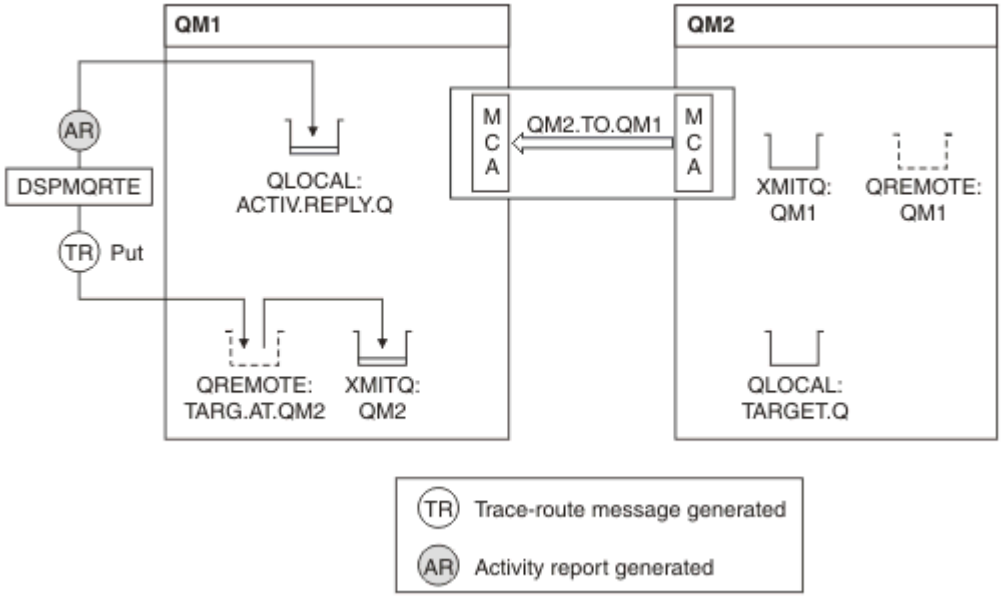


图 *19:* 诊断通道问题

- 在此示例中，通道为 QM1.TO.QM2 未在运行。
- DSPMQRTE 将跟踪路由消息 (例如 1) 放入目标队列并生成活动报告。

- 没有 MCA 从传输队列 (QM2) 获取消息，因此这是 DSPMQRTE 从应答队列返回的唯一活动报告。 这次路由未完成的事实表明存在问题。 管理员可以使用 ResolvedQName 中找到的传输队列来调查未提供传输队列的原因。

显示的输出如下:

```
AMQ8653: DSPMQRTE command started with options '-m QM1 -q TARG.AT.QM2
          -rq ACTIV.REPLY.Q -v outline'.
AMQ8659: DSPMQRTE command successfully put a message on queue 'QM2',
          queue manager 'QM1'.
AMQ8674: DSPMQRTE command is now waiting for information to display.
------------------------------------------------------------------------------
Activity:
 ApplName: 'cann\output\bin\dspmqrte.exe'

 Operation:
  OperationType: Put
  QMgrName: 'QM1                                                  '
  QName: 'TARG.AT.QM2                                            '
  ResolvedQName: 'QM2                                                    '
  RemoteQName: 'TARGET.Q                                                '
  RemoteQMgrName: 'QM2                                                      '

------------------------------------------------------------------------------
 AMQ8652: DSPMQRTE command has finished.
```

活动报告参考

使用此页面可获取活动报告消息格式的概述。 活动报告消息数据包含用于描述活动的参数。

活动报告格式

活动报告是包含消息描述符和消息数据的标准 IBM MQ 报告消息。 活动报告是应用程序生成的 PCF 消息，这些应用程序在通过队列管理器网络路由消息时代表消息执行活动。

活动报告包含以下信息:

消息描述符
MQMD 结构

消息数据
由以下内容组成:

- 嵌入式 PCF 头 (MQEPH)。
- 活动报告消息数据。

活动报告消息数据由 *Activity* PCF 组以及 *TraceRoute* PCF 组 (如果为跟踪路由消息生成) 组成。

第 86 页的表 19 显示了这些报告的结构，包括仅在特定条件下返回的参数。

表 19: 活动报告格式

MQMD 结构	嵌入式 PCF 头 MQEPH 结构	活动报告消息数据
结构标识 结构版本 报告选项 消息类型 到期时间 反馈 编码 编码字符集标识 消息格式 Priority 持久 消息标识 相关标识 回退计数 应答队列 应答队列管理器 用户标识 记帐标记 应用程序标识数据 应用程序类型 应用程序名称 放置日期 放置时间 应用程序原始数据 组标识 消息序号 偏移量 消息标志 原始长度	结构标识 结构版本 结构长度 编码 编码字符集标识 消息格式 标志 PCF 头 (MQCFH) 结构类型 结构长度 结构版本 命令标识符 消息序号 控制选项 完成代码 原因码 参数计数	活动 活动应用程序名称 活动应用程序类型 活动说明 操作 操作类型 操作日期 操作时间 消息 消息长度 MQMD [8] EmbeddedMQMD 队列管理器名称 队列共享组名 队列名称 [1] [2] [3] [7] 已解析的队列名称 [1] [3] [7] 远程队列名称 [3] [7] 远程队列管理器名称 [2] [3] [4] [5] [7] 预订级别 [9] 预订标识 [9] 反馈 [2] [10] 通道名称 [4] [5] 通道类型 [4] [5] 传输队列名称 [5] TraceRoute [6] 所需详情 记录的活动 未记录的活动 不连续计数 最大活动数 累计 交付

注意:

1. 针对 "获取" 和 "浏览" 操作返回。
2. 针对 "废弃" 操作返回。
3. 针对 "放置" , "放置应答" 和 "放置报告" 操作返回。
4. 针对 "接收" 操作返回。

5. 针对 "发送" 操作返回。
6. 针对跟踪路由消息返回。
7. 未针对发布活动中包含的主题的 Put 操作返回。
8. 未针对 "排除的发布" 操作返回。 对于 "发布" 和 "废弃的发布" 操作，返回的内容包含部分参数。
9. 针对 "发布"， "废弃发布" 和 "排除的发布" 操作返回。
10. 针对 "已废弃的发布" 和 "已排除的发布" 操作返回。

活动报告 *MQMD (消息描述符)*

使用此页面来查看活动报告的 MQMD 结构所包含的值

StrucId

结构标识:

数据类型

MQCHAR4

值

MQMD_STRUC_ID。

Version

结构版本号

数据类型

MQLONG

值

从原始消息描述符复制。 可能的值为:

MQMD_VERSION_1

Version-1 消息描述符结构，在所有环境中都受支持。

MQMD_VERSION_2

Version-2 消息描述符结构，在以下环境中受支持:

- AIX
- IBM i
- Linux
- Windows
- z/OS

以及连接到这些系统的所有 IBM MQ MQI clients 。

Report

用于进一步报告消息的选项

数据类型

MQLONG

值

如果在原始消息描述符的 报告 字段中指定了 MQRO_PASS_DISCARD_AND_到期或 MQRO_DISCARD_MSG:

MQRo_discard

如果无法将报告传递到目标队列，那么将废弃该报告。

否则:

MQRO_NONE

不需要报告。

MsgType

指示消息类型

数据类型
MQLONG

值
MQMT_REPORT

Expiry
报告消息生存期

数据类型
MQLONG

值
如果将原始消息描述符中的 报告 字段指定为 MQRO_PASS_DISCARD_AND_到期，那么将使用原始消息中的剩余到期时间。

否则：

MQEI_UNLIMITED
报告没有到期时间。

Feedback

描述： 反馈或原因码。

数据类型： MQLONG。

值： **MQFB_ACTIVITY**
活动报告。

Encoding

描述： 报告消息数据的数字编码。

数据类型： MQLONG。

值： MQENC_NATIVE。

CodedCharSetId

描述： 报告消息数据的字符集标识。

数据类型： MQLONG。

值： 根据需要设置。

Format

描述： 报告消息数据的格式名

数据类型： MQCHAR8.

值： **MQFMT_EMBEDDED_PCF**
嵌入式 PCF 消息。

Priority

描述： 报告消息优先级。

数据类型： MQLONG。

值： 从原始消息描述符复制。

Persistence

描述： 报告消息持久性。

数据类型： MQLONG。

值: 从原始消息描述符复制。

MsgId

描述: 消息标识。

数据类型: MQBYTE24.

值: 如果将原始消息描述符中的 报告 字段指定为 MQRO_PASS_MSG_ID，那么将使用原始消息中的消息标识。

否则，队列管理器将生成唯一值。

CorrelId

描述: 相关标识。

数据类型: MQBYTE24.

值: 如果将原始消息描述符中的 报告 字段指定为 MQRO_PASS_CORREL_ID，那么将使用原始消息中的相关标识。

否则，将从原始消息复制消息标识。

BackoutCount

描述: 回退计数器。

数据类型: MQLONG。

值: 0.

ReplyToQ

描述: 应答队列的名称。

数据类型: MQCHAR48.

值: 空白。

ReplyToQMgr

描述: 应答队列管理器的名称。

数据类型: MQCHAR48.

值: 生成报告消息的队列管理器名称。

UserIdentifier

描述: 生成报告消息的应用程序的用户标识。

数据类型: MQCHAR12.

值: 从原始消息描述符复制。

AccountingToken

描述: 允许应用程序对因消息而完成的工作收取费用的记帐令牌。

数据类型: MQBYTE32.

值: 从原始消息描述符复制。

ApplIdentityData

描述: 与身份相关的应用程序数据。

数据类型: MQCHAR32.

值: 从原始消息描述符复制。

PutApplType

描述: 放置报告消息的应用程序的类型。

数据类型: MQLONG。

值:

MQAT_QMGR
队列管理器生成的消息。

PutApplName

描述: 放置报告消息的应用程序的名称。

数据类型: MQCHAR28.

值: 队列管理器名称的前 28 个字节，或生成报告消息的 MCA 的名称。

PutDate

描述: 放入消息的日期。

数据类型: MQCHAR8.

值: 由队列管理器生成。

PutTime

描述: 放入消息的时间。

数据类型: MQCHAR8.

值: 由队列管理器生成。

ApplOriginData

描述: 与源相关的应用程序数据。

数据类型: MQCHAR4.

值: 空白。

如果 版本 为 MQMD_VERSION_2，那么存在以下其他字段:

GroupId

描述: 标识物理消息所属的消息组或逻辑消息。

数据类型: MQBYTE24.

值: 从原始消息描述符复制。

MsgSeqNumber

描述: 组中逻辑消息的序号。

数据类型: MQLONG。

值: 从原始消息描述符复制。

Offset

描述: 物理消息中的数据与逻辑消息开头的偏移量。

数据类型: MQLONG。

值: 从原始消息描述符复制。

MsgFlags

描述: 用于指定消息属性或控制其处理的消息标志。

数据类型: MQLONG。

值: 从原始消息描述符复制。

OriginalLength

描述: 原始消息的长度。

数据类型: MQLONG。

值: 从原始消息描述符复制。

活动报告 *MQEPH* (嵌入式 *PCF* 头)

使用此页面来查看活动报告的 MQEPH 结构所包含的值

MQEPH 结构包含随活动报告的消息数据提供的 PCF 信息以及随之而来的应用程序消息数据的描述。

对于活动报告， MQEPH 结构包含以下值:

StrucId

描述: 结构标识。

数据类型: MQCHAR4.

值: MQEPH_STRUC_ID。

Version

描述: 结构版本号。

数据类型: MQLONG。

值: MQEPH_VERSION_1.

StrucLength

描述: 结构长度。

数据类型: MQLONG。

值: 结构的总长度，包括跟随它的 PCF 参数结构。

Encoding

描述: 遵循最后一个 PCF 参数结构的消息数据的数字编码。

数据类型: MQLONG。

值: 如果报告消息中包含来自原始应用程序消息数据的任何数据，那么将从原始消息描述符的 编码 字段复制该值。

否则， 0。

CodedCharSetId

描述: 遵循最后一个 PCF 参数结构的消息数据的字符集标识。

数据类型: MQLONG。

值: 如果报告消息中包含来自原始应用程序消息数据的任何数据，那么将从原始消息描述符的 *CodedCharSetId* 字段复制该值。

否则， MQCCSI_UNDEFINED。

Format

描述: 遵循最后一个 PCF 参数结构的消息数据的格式名。

数据类型: MQCHAR8.

值: 如果报告消息中包含来自原始应用程序消息数据的任何数据，那么将从原始消息描述符的 格式 字段复制该值。

否则， MQFMT_NONE。

Flags

描述: 用于指定结构属性或控制其处理的标志。

数据类型: MQLONG。

值:

MQEPH_CCSID_EMBEDDED
指定在每个结构中的 *CodedCharSetId* 字段内单独指定包含字符数据的参数的字符集。

PCFHeader

描述: 可编程命令格式头

数据类型: MQCFH。

值: 请参阅第 92 页的『活动报告 MQCFH (PCF 头)』。

活动报告 *MQCFH (PCF 头)*

使用此页面来查看活动报告的 MQCFH 结构包含的 PCF 值

对于活动报告， MQCFH 结构包含以下值:

Type

描述: 用于标识报告消息内容的结构类型。

数据类型: MQLONG。

值:

MQCFT_REPORT
消息是报告。

StrucLength

描述: 结构长度。

数据类型: MQLONG。

值:

MQCFH_STRUC_LENGTH
MQCFH 结构的长度 (以字节为单位)。

Version

描述: 结构版本号。

数据类型: MQLONG。

值: MQCFH_VERSION_3

Command

描述: 命令标识。 这标识消息的类别。

数据类型: MQLONG。

值: **MQCMD_ACTIVITY_MSG**
消息活动。

MsgSeqNumber

描述: 消息序号。 这是一组相关消息中消息的序号。

数据类型: MQLONG。

值: 1

Control

描述: 控制选项。

数据类型: MQLONG。

值: MQCFC_LAST。

CompCode

描述: 完成代码。

数据类型: MQLONG。

值: MQCC_OK。

Reason

描述: 原因码限定完成代码。

数据类型: MQLONG。

值: MQRC_NONE。

ParameterCount

描述: 参数结构的计数。 这是遵循 MQCFH 结构的参数结构数。 组结构 (MQCFGR) 及其包含的参数结构仅计为一个结构。

数据类型: MQLONG。

值: 1 或更高版本。

活动报告消息数据

使用此页面来查看活动报告消息中的 活动 PCF 组所包含的参数。 仅当已执行特定操作时，才会返回某些参数。

活动报告消息数据由 *Activity* PCF 组以及 *TraceRoute* PCF 组 (如果为跟踪路由消息生成) 组成。 本主题中详细描述了 活动 PCF 组。

某些参数 (描述为 特定于操作的活动报告消息数据) 仅当已执行特定操作时才会返回。

对于活动报告，活动报告消息数据包含以下参数:

Activity

描述: 描述活动的分组参数。

标识: MQGACF_ACTIVITY。

数据类型: MQCFGR。

包含在 PCF 组中: 无。

PCF 组中的参数:

ActivityApplName
ActivityApplType
ActivityDescription
Operation
TraceRoute

已返回: 一直等在那里。

ActivityApplName

描述: 执行活动的应用程序的名称。

标识: MQCACF_APPL_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 活动。

最大长度: MQ_APPL_NAME_LENGTH。

已返回: 一直等在那里。

ActivityApplType

描述: 执行活动的应用程序的类型。

标识: MQIA_APPL_TYPE。

数据类型: MQCFIN。

包含在 PCF 组中: 活动。

已返回: 一直等在那里。

ActivityDescription

描述: 应用程序执行的活动的描述。

标识: MQCACF_ACTIVITY_DESCRIPTION。

数据类型: MQCFST。

包含在 PCF 组中: 活动。

最大长度: 64

已返回: 一直等在那里。

Operation

描述: 用于描述活动操作的分组参数。

标识: MQGACF_OPERATION。

数据类型: MQCFGR。

包含在 PCF 组中: 活动。

PCF 组中的参数:

OperationType
OperationDate
OperationTime
Message
QMgrName
QSGName

注: 根据操作类型，将在此组中返回其他参数。 这些其他参数描述为 特定于操作的活动报告消息数据。

已返回: 活动中每个操作一个 操作 PCF 组。

OperationType

描述: 执行的操作的类型。

标识: MQIACF_OPERATION_TYPE。

数据类型: MQCFIN。

包含在 PCF 组中: 操作。

值: MQOPER_ *。

已返回: 一直等在那里。

OperationDate

描述: 执行操作的日期。

标识: MQCACF_OPERATION_DATE。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_DATE_LENGTH。

已返回: 一直等在那里。

OperationTime

描述: 执行操作的时间。

标识: MQCACF_OPERATION_TIME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_TIME_LENGTH。

已返回: 一直等在那里。

Message

描述: 用于描述导致活动的消息的分组参数。

标识: MQGACF_MESSAGE。

数据类型: MQCFGR。

包含在 PCF 组中: 操作。

组中的参数:

MsgLength
MQMD
EmbeddedMQMD

已返回: 始终，但 "已排除的发布" 操作除外。

MsgLength

描述: 在发生活动之前导致该活动的消息的长度。

标识: MQIACF_MSG_LENGTH。

数据类型: MQCFIN。

包含在 PCF 组中: 消息。

已返回: 一直等在那里。

MQMD

描述: 与导致活动的消息的消息描述符相关的分组参数。

标识: MQGACF_MQMD。

数据类型: MQCFGR。

包含在 PCF 组中: 消息。

组中的参数:

StrucId
Version
Report
MsgType
Expiry
Feedback
Encoding
CodedCharSetId
Format
Priority
Persistence
MsgId
CorrelId
BackoutCount
ReplyToQ
ReplyToQMgr
UserIdentifier
AccountingToken
ApplIdentityData
PutApplType
PutApplName
PutDate
PutTime
ApplOriginData
GroupId
MsgSeqNumber
Offset
MsgFlags
OriginalLength

已返回: 始终，但 "已排除的发布" 操作除外。

EmbeddedMQMD

描述: 用于描述在传输队列上的消息中嵌入的消息描述符的分组参数。

标识: MQGACF_EMBEDDDED_MQMD。

数据类型: MQCFGR。

包含在 PCF 组中: 消息。

组中的参数:

StrucId
Version
Report
MsgType
Expiry
Feedback
Encoding
CodedCharSetId
Format
Priority
Persistence
MsgId
CorrelId
BackoutCount
ReplyToQ
ReplyToQMgr
UserIdentifier
AccountingToken
ApplIdentityData
PutApplType
PutApplName
PutDate
PutTime
ApplOriginData
GroupId
MsgSeqNumber
Offset
MsgFlags
OriginalLength

已返回: 对于将队列解析为传输队列的 Get 操作。

StrucId

描述: 结构标识

标识: MQCACF_STRUC_ID。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: 4.

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

Version

描述: 结构版本号。

标识: MQIACF_VERSION。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

Report

描述: 报告消息的选项。

标识: MQIACF_REPORT。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

MsgType

描述: 指示消息类型。

标识: MQIACF_MSG_TYPE。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

Expiry

描述: 消息生存期。

标识: MQIACF_EXPIRY。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

Feedback

描述: 反馈或原因码。

标识: MQIACF_FEEDBACK。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

Encoding

描述: 消息数据的数字编码。

标识: MQIACF_ENCODING。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

CodedCharSetId

描述: 消息数据的字符集标识

标识: MQIA_CODED_CHAR_SET_ID。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

Format

描述: 消息数据的格式名称

标识: MQCACH_FORMAT_NAME。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_FORMAT_LENGTH。

已返回: 始终，但 "已排除的发布" 操作除外。

Priority

描述: 这是消息优先级。

标识: MQIACF_PRIORITY。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，但 "已排除的发布" 操作除外。

Persistence

描述: 消息持久性。

标识: MQIACF_PERSISTENCE。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，但 "已排除的发布" 操作除外。

MsgId

描述: 消息标识。

标识: MQBACF_MSG_ID。

数据类型: MQCFBS。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_MSG_ID_LENGTH。

已返回: 始终，但 "已排除的发布" 操作除外。

CorrelId

描述: 相关标识。

标识: MQBACF_CORREL_ID。

数据类型: MQCFBS。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_CORREL_ID_LENGTH。

已返回: 始终，但 "已排除的发布" 操作除外。

BackoutCount

描述: 回退计数器。

标识: MQIACF_BACKOUT_COUNT。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，除了 "已排除的发布" 操作和 MQMD 中的 "发布" 和 "已废弃的发布" 操作。

ReplyToQ

描述: 应答队列的名称。

标识: MQCACF_REPLY_TO_Q。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_Q_NAME_LENGTH。

已返回: 始终，但 "已排除的发布操作" 和 MQMD 中的 "发布" 和 "已废弃的发布" 操作除外。

ReplyToQMgr

描述: 应答队列管理器的名称。

标识: MQCACF_REPLY_TO_Q_MGR。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_Q_MGR_NAME_LENGTH。

已返回: 始终，但 "已排除" 发布操作以及 MQMD 中的 "发布" 和 "已废弃" 发布操作除外。

UserIdentifier

描述: 发出消息的应用程序的用户标识。

标识: MQCACF_USER_IDENTIFIER。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_USER_ID_LENGTH。

已返回: 始终，但 "已排除的发布操作" 除外。

AccountingToken

描述: 允许应用程序对因消息而完成的工作收取费用的记帐令牌。

标识: MQBACF_ACCOUNTING_TOKEN。

数据类型: MQCFBS。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_ACCOUNTING_TOKEN_LENGTH。

已返回: 始终，但 "已排除的发布操作" 除外。

ApplIdentityData

描述: 与身份相关的应用程序数据。

标识: MQCACF_APPL_IDENTITY_DATA。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_APPL_IDENTITY_DATA_LENGTH。

已返回: 始终，但 "已排除的发布操作" 除外。

PutApplType

描述: 放置消息的应用程序的类型。

标识: MQIA_APPL_TYPE。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 始终，但 "已排除" 发布操作以及 MQMD 中的 "发布" 和 "已废弃" 发布操作除外。

PutApplName

描述: 放置消息的应用程序的名称。

标识: MQCACF_APPL_NAME。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_APPL_NAME_LENGTH。

已返回: 始终，但 "已排除" 发布操作以及 MQMD 中的 "发布" 和 "已废弃" 发布操作除外。

PutDate

描述: 放入消息的日期。

标识: MQCACF_PUT_DATE。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_PUT_DATE_LENGTH。

已返回: 始终，但 "已排除" 发布操作以及 MQMD 中的 "发布" 和 "已废弃" 发布操作除外。

PutTime

描述: 放入消息的时间。

标识: MQCACF_PUT_TIME。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_PUT_TIME_LENGTH。

已返回: 始终，但 "已排除" 发布操作以及 MQMD 中的 "发布" 和 "已废弃" 发布操作除外。

ApplOriginData

描述: 与源相关的应用程序数据。

标识: MQCACF_APPL_ORIGIN_DATA。

数据类型: MQCFST。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_APPL_ORIGIN_DATA_LENGTH。

已返回: 始终，但 "已排除" 发布操作以及 MQMD 中的 "发布" 和 "已废弃" 发布操作除外。

GroupId

描述: 标识物理消息所属的消息组或逻辑消息。

标识: MQBACF_GROUP_ID。

数据类型: MQCFBS。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

最大长度: MQ_GROUP_ID_LENGTH。

已返回: 如果 版本 指定为 MQMD_VERSION_2。 未在 "排除的发布操作" 和 MQMD 中针对 "发布" 和 "废弃的发布操作" 返回。

MsgSeqNumber

描述: 组中逻辑消息的序号。

标识: MQIACH_MSG_SEQUENCE_NUMBER。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 如果 *Version* 指定为 MQMD_VERSION_2。 未在 "排除的发布操作" 和 MQMD 中针对 "发布" 和 "废弃的发布操作" 返回。

Offset

描述: 物理消息中的数据与逻辑消息开头的偏移量。

标识: MQIACF_OFFSET。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 如果 *Version* 指定为 MQMD_VERSION_2。 未在 "排除的发布操作" 和 MQMD 中针对 "发布" 和 "废弃的发布操作" 返回。

MsgFlags

描述: 用于指定消息属性或控制其处理的消息标志。

标识: MQIACF_MSG_FLAGS。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 如果 *Version* 指定为 MQMD_VERSION_2。 未在 "排除的发布操作" 和 MQMD 中针对 "发布" 和 "废弃的发布操作" 返回。

OriginalLength

描述: 原始消息的长度。

标识: MQIACF_ORIGINAL_LENGTH。

数据类型: MQCFIN。

包含在 PCF 组中: *MQMD* 或 *EmbeddedMQMD*。

已返回: 如果 *Version* 指定为 MQMD_VERSION_2。 未在 "排除的发布操作" 和 MQMD 中针对 "发布" 和 "废弃的发布操作" 返回。

QMgrName

描述: 执行活动的队列管理器的名称。

标识: MQCA_Q_MGR_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_MGR_NAME_LENGTH

已返回: 一直等在那里。

QSGName

描述: 执行活动的队列管理器所属的队列共享组的名称。

标识: MQCA_QSG_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_QSG_NAME_LENGTH

已返回: 如果活动是在 IBM MQ for z/OS 队列管理器上执行的。

TraceRoute

描述: 用于指定 trace-route 消息的属性的分组参数。

标识: MQGACF_TRACE_ROUTE。

数据类型: MQCFGR。

包含在 PCF 组中: 活动。

组中的参数:

Detail
RecordedActivities
UnrecordedActivities
DiscontinuityCount
MaxActivities
Accumulate
Forward
Deliver

已返回: 如果活动是代表跟踪路由消息执行的。

TraceRoute PCF 组中的参数值是生成活动报告时来自跟踪路由消息的参数值。

特定于操作的活动报告消息数据

使用此页面来查看活动报告中 PCF 组 操作 中可能返回的其他 PCF 参数，具体取决于 *OperationType* 参数的值

其他参数根据以下操作类型而有所不同:

获取/浏览 *(MQOPER_GET/MQOPER_BROWSE)*
在 PCF 组 操作 中针对 "获取/浏览" (MQOPER_GET/MQOPER_BROWSE) 操作类型返回的其他活动报告消息数据参数 (已获取或浏览队列上的消息)。

QName

描述: 打开的队列的名称。

标识: MQCA_Q_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_NAME_LENGTH

已返回: 一直等在那里。

ResolvedQName

描述: 打开的队列解析为的名称。

标识: MQCACF_RESOLVED_Q_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_NAME_LENGTH

已返回: 一直等在那里。

废弃 *(MQOPER_DISCARD)*
在 "废弃" (MQOPER_DISCARD) 操作类型的 PCF 组 操作 中返回的其他活动报告消息数据参数 (已废弃消息)。

Feedback

描述: 废弃消息的原因。

标识: MQIACF_FEEDBACK。

数据类型: MQCFIN。

包含在 PCF 组中: 操作。

已返回: 一直等在那里。

QName

描述: 打开的队列的名称。

标识: MQCA_Q_NAME。

数据类型: MQCFST。

最大长度: MQ_Q_NAME_LENGTH

包含在 PCF 组中: 操作。

已返回: 如果由于未成功将消息放入队列而将其废弃。

RemoteQMgrName

描述: 将消息发送到的队列管理器的名称。

标识: MQCA_REMOTE_Q_MGR_NAME。

数据类型: MQCFST。

最大长度: MQ_Q_MGR_NAME_LENGTH

包含在 PCF 组中: 操作。

已返回: 如果 *Feedback* 的值为 MQFB_NOT_FORWARD。

发布/废弃发布/排除发布 *(MQOPER_PUBLISH/MQOPER_DISCARDED_PUBLISH/ MQOPER_EXCLUDED_PUBLISH)*
针对发布/废弃发布/排除的发布 (MQOPER_PUBLISH/MQOPER_DISCARDED_PUBLISH/ MQOPER_EXCLUDED_PUBLISH) 操作类型 (已传递，废弃或排除发布/预订消息) ，在 PCF 组 操作 中返回的其他活动报告消息数据参数。

SubId

描述: 预订标识。

标识: MQBACF_SUB_ID。

数据类型: MQCFBS。

包含在 PCF 组中: 操作。

已返回: 一直等在那里。

SubLevel

描述: 预订级别。

标识: MQIACF_SUB_LEVEL。

数据类型: MQCFIN。

包含在 PCF 组中: 操作。

已返回: 一直等在那里。

Feedback

描述: 废弃消息的原因。

标识: MQIACF_FEEDBACK。

数据类型: MQCFIN。

包含在 PCF 组中: 操作。

已返回: 如果由于未将消息传递给订户而废弃该消息，或者由于已排除该订户而未传递该消息。

发布操作 MQOPER_PUBLISH 提供有关传递到特定订户的消息的信息。 此操作描述可能已从关联的 Put 操作中描述的消息更改的继续消息元素。 与 Put 操作类似，它包含消息组 MQGACF_MESSAGE 以及其中的 MQMD 组 MQGACF_MQMD。 但是，此 MQMD 组仅包含以下字段，这些字段可由订户覆盖: *Format* , *Priority* , *Persistence* , *MsgId*, *CorrelId*, *UserIdentifier*, *AccountingToken* 和 *ApplIdentityData*。

操作信息中包含订户的 *SubId* 和 *SubLevel* 。 您可以将 *SubID* 与 MQCMD_INQUIRE_订户 PCF 命令配合使用，以检索订户的所有其他属性。

废弃的发布操作 MQOPER_DISCARDED_PUBLISH 类似于在点到点消息传递中未传递消息时使用的废弃操作。 如果明确请求不将消息传递至本地目标并且此订户指定本地目标，那么不会将消息传递至订户。 如果将消息获取到目标队列时出现问题 (例如，由于队列已满) ，那么也会将消息视为未传递。

"废弃发布" 操作中的信息与 "发布" 操作中的信息相同，添加了 反馈 字段以提供未传递消息的原因。 此反馈字段包含与 MQOPER_DISCARD 操作通用的 MQFB_ * 或 MQRC_ * 值。 废弃发布的原因 (而不是将其排除) 与废弃放置的原因相同。

排除的发布操作 MQOPER_EXCLUDED_PUBLISH 提供有关考虑传递消息的订户的信息，因为订户预订的主题与关联的 Put 操作的主题相匹配，但未将消息传递给订户，因为其他选择条件与放入主题的消息不匹配。 与 "废弃发布" 操作一样， 反馈 字段提供有关排除此预订的原因的信息。 但是，与 "废弃发布" 操作不同，未提供与消息相关的信息，因为没有为此订户生成消息。

放入/放入应答/放入报告 *(MQOPER_PUT/MQOPER_PUT_REPLY/MQOPER_PUT_REPORT)*
在 PCF 组 操作 中针对 "放入/放入应答/放入报告" (MQOPER_PUT/MQOPER_PUT_REPLY/ MQOPER_PUT_REPORT) 操作类型返回的其他活动报告消息数据参数 (消息，应答消息或报告消息已放入队列)。

QName

描述: 打开的队列的名称。

标识: MQCA_Q_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_NAME_LENGTH

已返回: 始终，除了一个例外: 如果 Put 操作是包含在发布活动中的主题，那么不会返回此异常。

ResolvedQName

描述: 打开的队列解析为的名称。

标识: MQCACF_RESOLVED_Q_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_NAME_LENGTH

已返回: 何时可以解析打开的队列。 如果 Put 操作是对包含在发布活动中的主题执行的操作，那么不会返回此值。

RemoteQName

描述: 在远程队列管理器上已知的已打开队列的名称。

标识: MQCA_REMOTE_Q_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_NAME_LENGTH

已返回: 如果打开的队列是远程队列。 如果 Put 操作是对包含在发布活动中的主题执行的操作，那么不会返回此值。

RemoteQMgrName

描述: 在其中定义远程队列的远程队列管理器的名称。

标识: MQCA_REMOTE_Q_MGR_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_MGR_NAME_LENGTH

已返回: 如果打开的队列是远程队列。 如果 Put 操作是对包含在发布活动中的主题执行的操作，那么不会返回此值。

TopicString

描述: 要将消息放入的完整主题字符串。

标识: MQCA_TOPIC_STRING。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

已返回: 如果 Put 操作是对发布活动中包含的主题执行的操作。

Feedback

描述: 将消息放入死信队列的原因。

标识: MQIACF_FEEDBACK。

数据类型: MQCFIN。

包含在 PCF 组中: 操作。

已返回: 如果消息已放入死信队列中。

接收 *(MQOPER_RECEIVE)*
针对 "接收" (MQOPER_RECEIVE) 操作类型 (在通道上接收到消息) , 在 PCF 组 操作 中返回的其他活动报告消息数据参数。

ChannelName

描述: 接收消息的通道的名称。

标识: MQCACH_CHANNEL_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_CHANNEL_NAME_LENGTH

已返回: 一直等在那里。

ChannelType

描述: 接收消息的通道类型。

标识: MQIACH_CHANNEL_TYPE。

数据类型: MQCFIN。

包含在 PCF 组中: 操作。

已返回: 一直等在那里。

RemoteQMgrName

描述: 从中接收消息的队列管理器的名称。

标识: MQCA_REMOTE_Q_MGR_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_MGR_NAME_LENGTH

已返回: 一直等在那里。

发送 *(MQOPER_SEND)*
针对 "发送" (MQOPER_SEND) 操作类型 (在通道上发送了消息) , 在 PCF 组 操作 中返回的其他活动报告消息数据参数。

ChannelName

描述: 发送消息的通道的名称。

标识: MQCACH_CHANNEL_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_CHANNEL_NAME_LENGTH。

已返回: 一直等在那里。

ChannelType

描述: 发送消息的通道的类型。

标识: MQIACH_CHANNEL_TYPE。

数据类型: MQCFIN。

包含在 PCF 组中: 操作。

已返回: 一直等在那里。

XmitQName

描述: 从中检索消息的传输队列。

标识: MQCACH_XMIT_Q_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_NAME_LENGTH。

已返回: 一直等在那里。

RemoteQMgrName

描述: 将消息发送到的远程队列管理器的名称。

标识: MQCA_REMOTE_Q_MGR_NAME。

数据类型: MQCFST。

包含在 PCF 组中: 操作。

最大长度: MQ_Q_MGR_NAME_LENGTH

已返回: 一直等在那里。

跟踪路由消息引用

使用此页面来获取跟踪路由消息格式的概述。 跟踪路由消息数据包含用于描述跟踪路由消息所导致的活动的参数

跟踪路由消息格式

跟踪路由消息是包含消息描述符和消息数据的标准 IBM MQ 消息。 消息数据包含有关通过队列管理器网络进行路由时在跟踪路由消息上执行的活动的信息。

跟踪路由消息包含以下信息:

消息描述符
MQMD 结构， 格式 字段设置为 MQFMT_ADMIN 或 MQFMT_EMBEDDED_PCF。

消息数据
由以下任一项组成:

- PCF 头 (MQCFH) 和跟踪路由消息数据 (如果 *Format* 设置为 MQFMT_ADMIN) ，或者
- 嵌入式 PCF 头 (MQEPH) ，跟踪路由消息数据和其他用户指定的消息数据 (如果 格式 设置为 MQFMT_EMBEDDED_PCF)。

使用 IBM MQ 显示路由应用程序生成跟踪路由消息时， 格式 设置为 MQFMT_ADMIN。

跟踪路由消息数据的内容由 *TraceRoute* PCF 组中的 累计 参数确定，如下所示:

- 如果 累计 设置为 MQROUTE_累计 _none ，那么跟踪路由消息数据将包含 *TraceRoute* PCF 组。
- 如果 累计 设置为 MQROUTE_累计 _IN_MSG 或 MQROUTE_累计 _AND_REPLY ，那么跟踪路由消息数据包含 *TraceRoute* PCF 组和零个或多个 活动 PCF 组。

第 109 页的表 20 显示了跟踪路由消息的结构。

表 *20:* 跟踪路由消息格式

MQMD 结构	**嵌入式 PCF 头 MQEPH 结构**	**跟踪路由消息数据**
结构标识 结构版本 报告选项 消息类型 到期时间 反馈 编码 编码字符集标识 消息格式 Priority 持久 消息标识 相关标识 回退计数 应答队列 应答队列管理器 用户标识 记帐标记 应用程序标识数据 应用程序类型 应用程序名称 放置日期 放置时间 应用程序原始数据 组标识 消息序号 偏移量 消息标志 原始长度	结构标识 结构版本 结构长度 编码 编码字符集标识 消息格式 标志 PCF 头 (MQCFH) 结构类型 结构长度 结构版本 命令标识符 消息序号 控制选项 完成代码 原因码 参数计数	TraceRoute 所需详情 记录的活动 未记录的活动 不连续计数 最大活动数 累计 交付

跟踪路由消息 *MQMD (消息描述符)*

使用此页面来查看跟踪路由消息的 MQMD 结构包含的值

StrucId

描述: 结构标识。

数据类型: MQCHAR4.

值: MQMD_STRUC_ID。

Version

描述: 结构版本号。

数据类型: MQLONG。

值: **MQMD_VERSION_1.**

Report

描述: 报告消息的选项。

数据类型: MQLONG。

值: 根据需求设置。 常见报告选项如下:

MQRO_DISCARD_MSG
消息到达本地队列时将被废弃。

MQRO_PASS_DISCARD_AND_EXPIRY
每个响应 (活动报告或跟踪路由应答消息) 都将设置报告选项 MQRO_DISCARD_MSG , 并传递剩余的到期时间。 这可确保响应不会无限期地保留在队列管理器网络中。

MsgType

描述: 消息类型。

数据类型: MQLONG。

值: 如果将 TraceRoute 组中的 累计 参数指定为 MQROUTE_蓄电池 _and_reply , 那么消息类型为 MQMT_REQUEST

否则:

MQMT_DATAGRAM。

Expiry

描述: 消息生存期。

数据类型: MQLONG。

值: 根据需求设置。 此参数可用于确保跟踪路由消息不会无限期地留在队列管理器网络中。

Feedback

描述: 反馈或原因码。

数据类型: MQLONG。

值: **MQFB_NONE**。

Encoding

描述: 消息数据的数字编码。

数据类型: MQLONG。

值: 根据需要设置。

CodedCharSetId

描述: 消息数据的字符集标识

数据类型: MQLONG。

值: 根据需要设置。

Format

描述: 消息数据的格式名称

数据类型: MQCHAR8.

值:

MQFMT_ADMIN
管理消息。 *TraceRoute* PCF 组之后没有用户数据。

MQFMT_EMBEDDED_PCF
嵌入式 PCF 消息。 用户数据遵循 *TraceRoute* PCF 组。

Priority

描述: 这是消息优先级。

数据类型: MQLONG。

值: 根据需求设置。

Persistence

描述: 消息持久性。

数据类型: MQLONG。

值: 根据需求设置。

MsgId

描述: 消息标识。

数据类型: MQBYTE24.

值: 根据需求设置。

CorrelId

描述: 相关标识。

数据类型: MQBYTE24.

值: 根据需求设置。

BackoutCount

描述: 回退计数器。

数据类型: MQLONG。

值: 0.

ReplyToQ

描述: 应答队列的名称。

数据类型: MQCHAR48.

值: 根据需求设置。

如果 *MsgType* 设置为 MQMT_REQUEST ， 或者如果 *Report* 设置了任何报告生成选项，那么此参数必须为非空白。

ReplyToQMgr

描述: 应答队列管理器的名称。

数据类型: MQCHAR48.

值: 根据需求设置。

UserIdentifier

描述: 发出消息的应用程序的用户标识。

数据类型: MQCHAR12.

值: 设置为正常。

AccountingToken

描述: 允许应用程序对因消息而完成的工作收取费用的记帐令牌。

数据类型: MQBYTE32.

值: 设置为正常。

ApplIdentityData

描述: 与身份相关的应用程序数据。

数据类型: MQCHAR32.

值: 设置为正常。

PutApplType

描述: 放置消息的应用程序的类型。

数据类型: MQLONG。

值: 设置为正常。

PutApplName

描述: 放置消息的应用程序的名称。

数据类型: MQCHAR28.

值: 设置为正常。

PutDate

描述: 放入消息的日期。

数据类型: MQCHAR8.

值: 设置为正常。

PutTime

描述: 放入消息的时间。

数据类型: MQCHAR8.

值: 设置为正常。

ApplOriginData

描述: 与源相关的应用程序数据。

数据类型: MQCHAR4.

值: 设置为正常 ...

跟踪路由消息 *MQEPH* (嵌入式 *PCF* 头)

使用此页面来查看跟踪路由消息的 MQEPH 结构所包含的值

MQEPH 结构包含对跟踪路由消息的消息数据及其后的应用程序消息数据的 PCF 信息的描述。 仅当其他用户消息数据跟在 TraceRoute PCF 组之后时，才会使用 MQEPH 结构。

对于跟踪路由消息， MQEPH 结构包含以下值:

StrucId

描述: 结构标识。

数据类型: MQCHAR4.

值: MQEPH_STRUC_ID。

Version

描述: 结构版本号。

数据类型: MQLONG。

值: MQEPH_VERSION_1.

StrucLength

描述: 结构长度。

数据类型: MQLONG。

值: 结构的总长度，包括跟随它的 PCF 参数结构。

Encoding

描述: 遵循最后一个 PCF 参数结构的消息数据的数字编码。

数据类型: MQLONG。

值: 消息数据的编码。

CodedCharSetId

描述: 遵循最后一个 PCF 参数结构的消息数据的字符集标识。

数据类型: MQLONG。

值: 消息数据的字符集。

Format

描述: 遵循最后一个 PCF 参数结构的消息数据的格式名。

数据类型: MQCHAR8.

值: 消息数据的格式名。

Flags

描述: 用于指定结构属性或控制其处理的标志。

数据类型: MQLONG。

值:

MQEPH_NONE
未指定任何标志。

MQEPH_CCSID_EMBEDDED
指定在每个结构中的 *CodedCharSetId* 字段内单独指定包含字符数据的参数的字符集。

PCFHeader

描述: 可编程命令格式头

数据类型: MQCFH。

值: 请参阅第 114 页的『跟踪路由消息 MQCFH (PCF 头)』。

跟踪路由消息 *MQCFH (PCF 头)*

使用此页面来查看跟踪路由消息的 MQCFH 结构包含的 PCF 值

对于跟踪路由消息， MQCFH 结构包含以下值:

Type

描述: 用于标识消息内容的结构类型。

数据类型: MQLONG。

值:

MQCFT_TRACE_ROUTE
消息是跟踪路由消息。

StrucLength

描述: 结构长度。

数据类型: MQLONG。

值:

MQCFH_STRUC_LENGTH
MQCFH 结构的长度 (以字节为单位)。

Version

描述: 结构版本号。

数据类型: MQLONG。

值: MQCFH_VERSION_3

Command

描述: 命令标识。 这标识消息的类别。

数据类型: MQLONG。

值:

MQCMD_TRACE_ROUTE
跟踪路由消息。

MsgSeqNumber

描述: 消息序号。 这是一组相关消息中消息的序号。

数据类型: MQLONG。

值: 1

Control

描述: 控制选项。

数据类型: MQLONG。

值: MQCFC_LAST。

CompCode

描述: 完成代码。

数据类型: MQLONG。

值: MQCC_OK。

Reason

描述: 原因码限定完成代码。

数据类型: MQLONG。

值: MQRC_NONE。

ParameterCount

描述: 参数结构的计数。 这是遵循 MQCFH 结构的参数结构数。 组结构 (MQCFGR) 及其包含的参数结构仅计为一个结构。

数据类型: MQLONG。

值: 1 或更高版本。

跟踪路由消息数据

使用此页面来查看构成跟踪路由消息数据的 *TraceRoute* PCF 组部分的参数

跟踪路由消息数据的内容取决于 *TraceRoute* PCF 组中的 累计 参数。 跟踪路由消息数据由 *TraceRoute* PCF 组和零个或多个 活动 PCF 组组成。 本主题中详细描述了 *TraceRoute* PCF 组。 有关 活动 PCF 组的详细信息，请参阅相关信息。

跟踪路由消息数据包含以下参数:

TraceRoute

描述: 用于指定 trace-route 消息的属性的分组参数。 对于跟踪路由消息，可以更改其中一些参数以控制其处理方式。

标识: MQGACF_TRACE_ROUTE。

数据类型: MQCFGR。

包含在 PCF 组中: 无。

组中的参数:

Detail
RecordedActivities
UnrecordedActivities
DiscontinuityCount
MaxActivities
Accumulate
Forward
Deliver

Detail

描述: 将为活动记录的详细信息级别。

标识: MQIACF_ROUTE_DETAIL。

数据类型: MQCFIN。

包含在 PCF 组中: *TraceRoute*.

值:

MQROUTE_DETAIL_LOW
记录用户编写的应用程序执行的活动。

MQROUTE_DETAIL_MEDIUM
记录 MQROUTE_DETAIL_LOW 中指定的活动。 此外，还会记录 MCA 执行的活动。

MQROUTE_DETAIL_HIGH
记录 MQROUTE_DETAIL_LOW 和 MQROUTE_DETAIL_MEDIUM 中指定的活动。 MCA 不会在此详细信息级别记录任何进一步的活动信息。 此选项仅适用于要记录进一步活动信息的用户编写的应用程序。

RecordedActivities

描述: 跟踪路由消息在其中记录信息的活动数。

标识: MQIACF_RECORDED_ACTIVactivities。

数据类型: MQCFIN。

包含在 PCF 组中: *TraceRoute*.

UnrecordedActivities

描述: 未记录信息的跟踪路由消息所导致的活动数。

标识: MQIACF_UNRECORDED_ACTIVactivities。

数据类型: MQCFIN。

包含在 PCF 组中: *TraceRoute*.

DiscontinuityCount

描述: 从不支持跟踪路由消息传递的队列管理器接收到跟踪路由消息的次数。

标识: MQIACF_DISCONTINUITY_COUNT。

数据类型: MQCFIN。

包含在 PCF 组中: *TraceRoute*.

MaxActivities

描述: 在停止处理跟踪路由消息之前，该消息可能涉及的最大活动数。

标识: MQIACF_MAX_ACTIVactivities。

数据类型: MQCFIN。

包含在 PCF 组中: *TraceRoute*.

值:

正整数
最大活动数。

MQROUTE_UNLIMITED_ACTIVITIES
无限数量的活动。

Accumulate

描述: 指定是否在跟踪路由消息中累积活动信息，以及是在废弃跟踪路由消息之前生成包含累积活动信息的应答消息，还是将其放在非传输队列上。

标识: MQIACF_ROUTE_累加。

数据类型: MQCFIN。

包含在 PCF 组中: *TraceRoute*.

值:

MQROUTE_累计无
未在跟踪路由消息的消息数据中累积活动信息。

MQROUTE_累加 _in_msg
活动信息在跟踪路由消息的消息数据中累积。

MQROUTE_蓄电池 _and_reply
活动信息在跟踪路由消息的消息数据中累积，将生成跟踪路由应答消息。

Forward

描述: 指定可将跟踪路由消息转发到的队列管理器。 确定是否将消息转发到远程队列管理器时，队列管理器使用 转发中描述的算法。

标识: MQIACF_ROUTE_转发。

数据类型: MQCFIN。

包含在 PCF 组中: *TraceRoute*.

值:

MQROUTE_FORWARD_IF_SUPPORTED
跟踪路由消息仅转发给队列管理器，这些队列管理器将采用 *TraceRoute* 组中 *Deliver* 参数的值。

MQROUTE_FORWARD_ALL
无论是否将使用 *Deliver* 参数的值，都会将跟踪路由消息转发到任何队列管理器。

Deliver

描述: 指定当跟踪路由消息成功到达目标队列时要执行的操作。

标识: MQIACF_ROUTE_DELIVERY。

数据类型: MQCFIN。

包含在 PCF 组中: *TraceRoute*.

值:

MQROUTE_DELIVER_YES
到达时，会将跟踪路由消息放在目标队列上。 在目标队列上执行破坏性获取的任何应用程序都可以接收跟踪路由消息。

MQROUTE_DELIVER_NO
到达时，将废弃跟踪路由消息。

跟踪路由应答消息引用

使用此页面来获取跟踪路由应答消息格式的概述。 跟踪路由应答消息数据与为其生成跟踪路由消息的跟踪路由消息数据重复

跟踪路由应答消息格式

跟踪路由应答消息是包含消息描述符和消息数据的标准 IBM MQ 消息。 消息数据包含有关通过队列管理器网络进行路由时在跟踪路由消息上执行的活动的信息。

跟踪路由应答消息包含以下信息:

消息描述符
MQMD 结构

消息数据
PCF 头 (MQCFH) 和跟踪路由应答消息数据

跟踪路由应答消息数据由一个或多个 活动 PCF 组组成。

当跟踪路由消息到达其目标队列时，可以生成包含来自跟踪路由消息的活动信息副本的跟踪路由应答消息。 跟踪路由应答消息将传递到应答队列或系统队列。

第 118 页的表 21 显示了跟踪路由应答消息的结构，包括仅在特定条件下返回的参数。

表 *21:* 跟踪路由应答消息格式

MQMD 结构	**PCF 头 MQCFH 结构**	**跟踪路由应答消息数据**
结构标识 结构版本 报告选项 消息类型 到期时间 反馈 编码 编码字符集标识 消息格式 Priority 持久 消息标识 相关标识 回退计数 应答队列 应答队列管理器 用户标识 记帐标记 应用程序标识数据 应用程序类型 应用程序名称 放置日期 放置时间 应用程序原始数据 组标识 消息序号 偏移量 消息标志 原始长度	PCF 头 (MQCFH) 结构类型 结构长度 结构版本 命令标识符 消息序号 控制选项 完成代码 原因码 参数计数	活动 活动应用程序名称 活动应用程序类型 活动说明 操作 操作类型 操作日期 操作时间 消息 消息长度 MQMD EmbeddedMQMD 队列管理器名称 队列共享组名 队列名称 [1] [2] [3] 已解析的队列名称 [1] [3] 远程队列名称 [3] 远程队列管理器- 名称 [2] [3] [4] [5] 反馈 [2] 通道名称 [4] [5] 通道类型 [4] [5] 传输队列名称 [5] TraceRoute 所需详情 记录的活动 未记录的活动 不连续计数 最大活动数 累计 交付

表 *21:* 跟踪路由应答消息格式 *(*继续*)*		
MQMD 结构	**PCF 头 MQCFH 结构**	**跟踪路由应答消息数据**
注: 1. 针对 "获取" 和 "浏览" 操作返回。 2. 针对 "废弃" 操作返回。 3. 针对 "放置" ， "放置应答" 和 "放置报告" 操作返回。 4. 针对 "接收" 操作返回。 5. 针对 "发送" 操作返回。		

跟踪路由应答消息 *MQMD (*消息描述符*)*

使用此页面来查看跟踪路由应答消息的 MQMD 结构所包含的值

对于跟踪路由应答消息， MQMD 结构包含 活动报告消息描述符中描述的参数。 跟踪路由应答消息描述符中的某些参数值与活动报告消息描述符中的参数值不同，如下所示:

MsgType

描述: 消息类型。

数据类型: MQLONG。

值: **MQMT_REPLY**

Feedback

描述: 反馈或原因码。

数据类型: MQLONG。

值: **MQFB_NONE**

Encoding

描述: 消息数据的数字编码。

数据类型: MQLONG。

值: 从跟踪路由消息描述符复制。

CodedCharSetId

描述: 消息数据的字符集标识

数据类型: MQLONG。

值: 从跟踪路由消息描述符复制。

Format

描述: 消息数据的格式名称

数据类型: MQCHAR8.

值: **MQFMT_ADMIN**
管理消息。

跟踪路由应答消息 *MQCFH (PCF* 头*)*

使用此页面来查看跟踪路由应答消息的 MQCFH 结构包含的 PCF 值

跟踪路由应答消息的 PCF 头 (MQCFH) 与跟踪路由消息的 PCF 头相同。

跟踪路由应答消息数据

跟踪路由应答消息数据与为其生成跟踪路由消息的跟踪路由消息数据重复

跟踪路由应答消息数据包含一个或多个 活动 组。 第 93 页的『活动报告消息数据』 中描述了这些参数。

记帐和统计信息消息

队列管理器生成记帐和统计信息消息以记录有关 IBM MQ 应用程序执行的 MQI 操作的信息，或记录有关 IBM MQ 系统中发生的活动的信息。

记帐消息
记帐消息用于记录有关 IBM MQ 应用程序执行的 MQI 操作的信息，请参阅 第 120 页的『记帐消息』。

统计信息消息
统计信息消息用于记录有关 IBM MQ 系统中发生的活动的信息，请参阅 第 123 页的『统计信息消息』。统计信息消息中记录的某些活动与内部队列管理器操作相关。

z/OS 此处描述的记帐消息和统计信息消息在 IBM MQ for z/OS 上不可用，但通过系统管理设施 (SMF) 提供了等效功能。

记帐和统计信息消息将传递到两个系统队列中的一个。 用户应用程序可以从这些系统队列中检索消息，并将记录的信息用于各种目的:

- 说明应用程序资源使用情况。
- 记录应用程序活动。
- 容量规划。
- 检测队列管理器网络中的问题。
- 帮助确定队列管理器网络中问题的原因。
- 提高队列管理器网络的效率。
- 熟悉队列管理器网络的运行。
- 确认队列管理器网络正在正确运行。

相关概念
第 288 页的『Using System Management Facility』
You can use SMF to collect statistics and accounting information. To use SMF, certain parameters must be set in z/OS and in IBM MQ.

记帐消息

记帐消息记录有关 IBM MQ 应用程序执行的 MQI 操作的信息。 记帐消息是包含多个 PCF 结构的 PCF 消息。

当应用程序与队列管理器断开连接时，将生成一条记帐消息并将其传递到系统记帐队列 (SYSTEM.ADMIN.ACCOUNTING.QUEUE)。 对于长时间运行的 IBM MQ 应用程序，将按如下所示生成中间记帐消息:

- 自建立连接以来的时间超过配置的时间间隔时。
- 自上次中间记帐消息以来经过的时间超过配置的时间间隔时。

记帐消息位于以下类别中:

MQI 记帐消息
MQI 记帐消息包含与使用与队列管理器的连接进行的 MQI 调用数相关的信息。

队列记帐消息
队列记帐消息包含与使用与队列管理器的连接 (按队列分组) 进行的 MQI 调用数相关的信息。

每条队列记帐消息最多可包含 100 条记录，每条记录都与应用程序针对特定队列执行的活动相关。

仅针对本地队列记录记帐消息。 如果应用程序针对别名队列进行 MQI 调用，那么将针对基本队列记录记帐数据，对于远程队列，将针对传输队列记录记帐数据。

注: 由于该信息与 IBM MQ 应用程序执行的 MQI 操作相关，因此该信息不包含与流式队列相关的操作，除非该操作由应用程序直接在该队列上。

相关参考

第 137 页的『MQI 记帐消息数据』
使用此页面来查看 MQI 记帐消息的结构

第 147 页的『队列记帐消息数据』
使用此页面来查看队列记帐消息的结构

记帐消息格式

记帐消息由一组由消息描述符和消息数据组成的 PCF 字段组成。

消息描述符

- 记帐消息 MQMD (消息描述符)

记帐消息数据

- 记帐消息 MQCFH (PCF 头)
- 始终返回的记帐消息数据
- 返回的记帐消息数据 (如果可用)

记帐消息 MQCFH (PCF 头) 包含有关应用程序的信息以及记录记帐数据的时间间隔。

记帐消息数据包括用于存储记帐信息的 PCF 参数。 记帐消息的内容取决于消息类别，如下所示:

MQI 记帐消息
MQI 记帐消息数据由多个 PCF 参数组成，但没有 PCF 组。

队列记帐消息
队列记帐消息数据由多个 PCF 参数组成，范围为 1 到 100 *QAccountingData* PCF 组。

对于收集了记帐数据的每个队列，都有一个 *QAccountingData* PCF 组。 如果应用程序访问超过 100 个队列，那么将生成多条记帐消息。 每条消息都相应地更新了 MQCFH (PCF 头) 中的 *SeqNumber* ，并且序列中的最后一条消息在 MQCFH 中具有指定为 MQCFC_LAST 的 *Control* 参数。

会计信息收集

使用队列和队列管理器属性来控制记帐信息的收集。 您还可以使用 MQCONNX 选项在连接级别控制集合。

控制 *MQI* 记帐信息的收集
使用队列管理器属性 ACCTMQI 来控制 MQI 记帐信息的收集。

要更改此属性的值，请使用 MQSC 命令 ALTER QMGR 并指定参数 ACCTMQI。 仅为启用记帐后开始的连接生成记帐消息。 **ACCTMQI** 参数可以具有以下值:

ON
针对到队列管理器的每个连接收集 MQI 记帐信息。

OFF
未收集 MQI 记帐信息。 这是缺省值。

例如，要启用 MQI 记帐信息收集，请使用以下 MQSC 命令:

```
ALTER QMGR ACCTMQI(ON)
```

队列记帐信息
使用队列属性 ACCTQ 和队列管理器属性 ACCTQ 来控制队列记帐信息的收集。

要更改队列属性的值，请使用 MQSC 命令 ALTER QLOCAL，并指定参数 ACCTQ。 仅为启用记帐后开始的连接生成记帐消息。 请注意，对该值所作的更改仅对在对该属性进行更改之后与队列管理器进行的连接有效。

队列属性 ACCTQ 可以具有以下值:

ON
将针对与打开队列的队列管理器的每个连接收集此队列的队列记帐信息。

OFF
未收集此队列的队列记帐信息。

QMGR
此队列的队列记帐信息集合根据队列管理器属性 ACCTQ 的值进行控制。 这是缺省值。

要更改队列管理器属性的值，请使用 MQSC 命令 ALTER QMGR 并指定参数 ACCTQ。 队列管理器属性 ACCTQ 可以具有以下值:

ON
将针对队列属性 ACCTQ 设置为 QMGR 的队列收集队列记帐信息。

OFF
对于将队列属性 ACCTQ 设置为 QMGR 的队列，不会收集队列记帐信息。 这是缺省值。

无
将对所有队列禁用队列记帐信息收集，而不考虑队列属性 ACCTQ。

如果队列管理器属性 ACCTQ 设置为 NONE ，那么将对所有队列禁用队列记帐信息收集，而不考虑队列属性 ACCTQ。

例如，要对队列 Q1 启用记帐信息收集，请使用以下 MQSC 命令:

```
ALTER QLOCAL(Q1) ACCTQ(ON)
```

要对将队列属性 ACCTQ 指定为 QMGR 的所有队列启用记帐信息收集，请使用以下 MQSC 命令:

```
ALTER QMGR ACCTQ(ON)
```

用于控制记帐信息收集的 *MQCONNX* 选项
在 MQCONNX 调用上使用 **ConnectOpts** 参数，通过覆盖队列管理器属性 ACCTMQI 和 ACCTQ 的有效值，在连接级别修改 MQI 和队列记帐信息的集合

ConnectOpts 参数可以具有以下值:

MQCNO_ACCOUNTING_MQI_ENABLED
如果将队列管理器属性 ACCTMQI 的值指定为 OFF ，那么将为此连接启用 MQI 记帐。 这相当于将队列管理器属性 ACCTMQI 指定为 ON。

如果未将队列管理器属性 ACCTMQI 的值指定为 OFF ，那么此属性无效。

MQCNO_ACCOUNTING_MQI_DISABLED
如果将队列管理器属性 ACCTMQI 的值指定为 ON ，那么将对此连接禁用 MQI 记帐。 这相当于将队列管理器属性 ACCTMQI 指定为 OFF。

如果未将队列管理器属性 ACCTMQI 的值指定为 ON ，那么此属性无效。

MQCNO_ACCOUNTING_Q_ENABLED
如果将队列管理器属性 ACCTQ 的值指定为 OFF ，那么将对此连接启用队列记帐。 将 ACCTQ 指定为 QMGR 的所有队列都启用了队列记帐。 这相当于将队列管理器属性 ACCTQ 指定为 ON。

如果未将队列管理器属性 ACCTQ 的值指定为 OFF ，那么此属性无效。

MQCNO_ACCOUNTING_Q_DISABLED
如果将队列管理器属性 ACCTQ 的值指定为 ON ，那么将对此连接禁用队列记帐。 这相当于将队列管理器属性 ACCTQ 指定为 OFF。

如果未将队列管理器属性 ACCTQ 的值指定为 ON ，那么此属性无效。

缺省情况下，已禁用这些覆盖。 要启用这些属性，请将队列管理器属性 ACCTCONO 设置为 ENABLED。 要对各个连接启用记帐覆盖，请使用以下 MQSC 命令:

```
ALTER QMGR ACCTCONO(ENABLED)
```

应用程序断开连接时的记帐消息生成

当应用程序与队列管理器断开连接时，将生成记帐消息。 还会针对长时间运行的 IBM MQ 应用程序编写中间记帐消息。

当应用程序断开连接时，将通过以下任一方式生成记帐消息:

- 应用程序发出 MQDISC 调用
- 队列管理器识别应用程序已终止

当自建立连接以来的时间间隔或自上次写入的中间记帐消息超过配置的时间间隔以来的时间间隔时，将为长时间运行的 IBM MQ 应用程序写入中间记帐消息。 队列管理器属性 ACCTINT 指定可以自动写入中间记帐消息的时间 (以秒计)。 仅当应用程序与队列管理器交互时，才会生成记帐消息，因此在未执行 MQI 请求的情况下长时间保持与队列管理器连接的应用程序不会生成记帐消息，直到在完成记帐时间间隔之后执行第一个 MQI 请求为止。

缺省记帐时间间隔为 1800 秒 (30 分钟)。 例如，要将记帐时间间隔更改为 900 秒 (15 分钟) ，请使用以下 MQSC 命令:

```
ALTER QMGR ACCTINT(900)
```

统计信息消息

统计信息消息记录有关 IBM MQ 系统中发生的活动的信息。 统计信息消息是包含多个 PCF 结构的 PCF 消息。

统计信息消息将传递到系统队列 (SYSTEM.ADMIN.STATISTICS.QUEUE) ，在配置的时间间隔内，只要存在某个活动。

统计信息消息位于以下类别中:

MQI 统计信息消息
MQI 统计信息消息包含与配置的时间间隔内执行的 MQI 调用数相关的信息。 例如，信息可以包括队列管理器发出的 MQI 调用数。

队列统计信息消息
队列统计信息消息包含与配置的时间间隔内队列活动相关的信息。 该信息包括放入队列和从队列中检索的消息数以及队列处理的总字节数。

每条队列统计信息消息最多可以包含 100 条记录，每条记录都与收集了统计信息的每个队列的活动相关。

仅针对本地队列记录统计信息消息。 如果应用程序对别名队列进行 MQI 调用，那么将针对基本队列记录统计数据，对于远程队列，将针对传输队列记录统计数据。

通道统计信息消息
通道统计信息消息包含与配置的时间间隔内通道活动相关的信息。 例如，信息可以是通道传输的消息数，也可以是通道传输的字节数。

每条通道统计信息消息最多包含 100 条记录，每条记录与收集了统计信息的每个通道的活动相关。

注: 由于该信息与 IBM MQ 系统中发生的活动相关，因此该信息包含与流式队列相关的操作。

相关参考

第 124 页的『控制 MQI 统计信息收集』
使用队列管理器属性 STATMQI 来控制 MQI 统计信息的收集。

第 124 页的『控制队列统计信息收集』
使用队列属性 STATQ 和队列管理器属性 STATQ 来控制队列统计信息的收集

第 125 页的『控制通道统计信息收集』
使用通道属性 STATCHL 来控制通道统计信息的收集。 您还可以设置队列管理器属性以控制信息收集。

统计信息消息格式

统计信息消息由一组由消息描述符和消息数据组成的 PCF 字段组成。

消息描述符

- 统计信息消息 MQMD (消息描述符)

记帐消息数据

- 统计信息消息 MQCFH (PCF 头)
- 始终返回的统计信息消息数据
- 返回的统计信息消息数据 (如果可用)

统计信息消息 MQCFH (PCF 头) 包含有关记录统计信息数据的时间间隔的信息。

统计信息消息数据包含用于存储统计信息的 PCF 参数。 统计信息消息的内容取决于消息类别，如下所示:

MQI 统计信息消息
MQI 统计信息消息数据由多个 PCF 参数组成，但没有 PCF 组。

队列统计信息消息
队列统计信息消息数据由多个 PCF 参数组成，范围为 1 到 100 *QStatisticsData* PCF 组。

对于时间间隔内处于活动状态的每个队列，都有一个 *QStatisticsData* PCF 组。 如果在时间间隔内有超过 100 个队列处于活动状态，那么将生成多条统计信息消息。 每条消息都相应地更新了 MQCFH (PCF 头) 中的 *SeqNumber* ，并且序列中的最后一条消息在 MQCFH 中具有指定为 MQCFC_LAST 的 *Control* 参数。

通道统计信息消息
通道统计信息消息数据由多个 PCF 参数组成，范围为 1 到 100 *ChlStatistics* 数据 PCF 组。

对于时间间隔内处于活动状态的每个通道，都有一个 *ChlStatistics* 数据 PCF 组。 如果在时间间隔内有超过 100 个通道处于活动状态，那么将生成多条统计信息消息。 每条消息都相应地更新了 MQCFH (PCF 头) 中的 *SeqNumber* ，并且序列中的最后一条消息在 MQCFH 中具有指定为 MQCFC_LAST 的 *Control* 参数。

统计信息收集
使用队列，队列管理器和通道属性来控制统计信息的收集

控制 *MQI* 统计信息收集
使用队列管理器属性 STATMQI 来控制 MQI 统计信息的收集。

要更改此属性的值，请使用 MQSC 命令 ALTER QMGR 并指定参数 **STATMQI**。 仅为启用统计信息收集后打开的队列生成统计信息消息。 **STATMQI** 参数可以具有以下值:

ON
针对与队列管理器的每个连接收集 MQI 统计信息。

OFF
未收集 MQI 统计信息。 这是缺省值。

例如，要启用 MQI 统计信息收集，请使用以下 MQSC 命令:

```
ALTER QMGR STATMQI(ON)
```

控制队列统计信息收集
使用队列属性 STATQ 和队列管理器属性 STATQ 来控制队列统计信息的收集

您可以对单个队列或多个队列启用或禁用队列统计信息收集。 要控制个别队列，请设置队列属性 STATQ。 您可以使用队列管理器属性 STATQ 在队列管理器级别启用或禁用队列统计信息收集。 对于使用值 QMGR 指定了队列属性 STATQ 的所有队列，将在队列管理器级别控制队列统计信息收集。

队列统计信息仅对使用 IBM MQ MQI 对象句柄的操作递增，这些操作是在启用统计信息收集后打开的。

仅针对先前时间段内已收集其统计数据的队列生成队列统计信息消息。

同一个队列可以有几个 put 操作，并通过几个 Object Handles 获取操作。 在启用统计信息收集之前，可能已打开某些对象句柄，但随后打开了其他对象句柄。 因此，队列统计信息可以记录一些 put 操作和 get 操作的活动，而不是所有操作。

要确保 "队列统计信息" 记录所有应用程序的活动，必须关闭并重新打开正在监视的一个或多个队列上的新对象句柄。 实现此目标的最佳方法是在启用统计信息收集后结束并重新启动所有应用程序。

要更改队列属性 STATQ 的值，请使用 MQSC 命令 ALTER QLOCAL 并指定参数 STATQ。 队列属性 STATQ 可以具有以下值:

ON
将针对与打开队列的队列管理器的每个连接收集队列统计信息。

OFF
未收集此队列的队列统计信息。

QMGR
此队列的队列统计信息收集根据队列管理器属性 STATQ 的值进行控制。 这是缺省值。

要更改队列管理器属性 STATQ 的值，请使用 MQSC 命令 ALTER QMGR 并指定参数 STATQ。 队列管理器属性 STATQ 可以具有以下值:

ON
针对将队列属性 STATQ 设置为 QMGR 的队列收集队列统计信息

OFF
对于将队列属性 STATQ 设置为 QMGR 的队列，不会收集队列统计信息。 这是缺省值。

无
将对所有队列禁用队列统计信息收集，而不考虑队列属性 STATQ。

如果队列管理器属性 STATQ 设置为 NONE ，那么将对所有队列禁用队列统计信息收集，而不考虑队列属性 STATQ。

例如，要对队列 Q1 启用统计信息收集，请使用以下 MQSC 命令:

```
ALTER QLOCAL(Q1) STATQ(ON)
```

要对将队列属性 STATQ 指定为 QMGR 的所有队列启用统计信息收集，请使用以下 MQSC 命令:

```
ALTER QMGR STATQ(ON)
```

控制通道统计信息收集
使用通道属性 STATCHL 来控制通道统计信息的收集。 您还可以设置队列管理器属性以控制信息收集。

您可以对各个通道或多个通道启用或禁用通道统计信息收集。 要控制各个通道，必须设置通道属性 STATCHL 以启用或禁用通道统计信息收集。 要同时控制多个通道，请使用队列管理器属性 STATCHL 在队列管理器级别启用或禁用通道统计信息收集。 对于使用值 QMGR 指定了通道属性 STATCHL 的所有通道，将在队列管理器级别控制通道统计信息收集。

自动定义的集群发送方通道不是 IBM MQ 对象，因此不具有与通道对象相同的属性。 要控制自动定义的集群发送方通道，请使用队列管理器属性 STATACLS。 此属性确定是针对通道统计信息收集启用还是禁用队列管理器中自动定义的集群发送方通道。

您可以将通道统计信息收集设置为以下三个监视级别之一: 低，中或高。 您可以在对象级别或队列管理器级别设置监视级别。 要使用的级别的选择取决于您的系统。 收集统计信息数据可能需要一些在计算上相对昂贵的指令，因此为了减少通道统计信息收集的影响，中，低监视选项会定期测量数据的样本，而不是一直收集数据。 第 125 页的表 22 汇总了可用于通道统计信息收集的级别:

表 *22:* 通道统计信息收集的详细级别

级别	描述	用法
低	定期测量一小部分数据样本。	用于处理大量消息的对象。
中等	定期测量数据样本。	对于大多数对象。

表 22: 通道统计信息收集的详细级别 (继续)		
级别	**描述**	**用法**
高	定期测量所有数据。	对于每秒仅处理几条消息的对象，其中最新的信息很重要。

要更改通道属性 STATCHL 的值，请使用 MQSC 命令 ALTER CHANNEL 并指定参数 STATCHL。

要更改队列管理器属性 STATCHL 的值，请使用 MQSC 命令 ALTER QMGR 并指定参数 STATCHL。

要更改队列管理器属性 STATACLS 的值，请使用 MQSC 命令 ALTER QMGR 并指定参数 STATACLS。

通道属性 STATCHL 可以具有以下值:

低
收集通道统计信息的详细信息级别较低。

中型
将使用中等级别的详细信息来收集通道统计信息。

高
将收集具有高级别详细信息的通道统计信息。

OFF
未收集此通道的通道统计信息。

QMGR
通道属性设置为 QMGR。 此通道的统计信息收集由队列管理器属性 STATCHL 的值控制。

这是缺省值。

z/OS 在 z/OS 系统上，启用此参数会直接开启统计信息数据收集，而不考虑您选择的值。 指定 LOW、MEDIUM 或 HIGH 对您的结果没有差别。必须启用此参数以收集通道记帐记录。

队列管理器属性 STATCHL 可以具有以下值:

低
对于将通道属性 STATCHL 设置为 QMGR 的所有通道，将使用较低级别的详细信息来收集通道统计信息。

中型
对于将通道属性 STATCHL 设置为 QMGR 的所有通道，将使用中等级别的详细信息来收集通道统计信息。

高
对于将通道属性 STATCHL 设置为 QMGR 的所有通道，将使用高级别详细信息来收集通道统计信息。

OFF
对于将通道属性 STATCHL 设置为 QMGR 的所有通道，不会收集通道统计信息。

这是缺省值。

无
将对所有通道禁用通道统计信息收集，而不考虑通道属性 STATCHL。

z/OS 在 z/OS 系统上，启用此参数会直接开启统计信息数据收集，而不考虑您选择的值。 指定 LOW、MEDIUM 或 HIGH 对您的结果没有差别。必须启用此参数以收集通道记帐记录。

队列管理器属性 STATACLS 可以具有以下值:

低
对于自动定义的集群发送方通道，收集统计信息的详细信息级别较低。

中型
将使用自动定义的集群发送方通道的中等详细信息级别来收集统计信息。

高
将使用自动定义的集群发送方通道的高级别详细信息来收集统计信息。

OFF
不会收集自动定义的集群发送方通道的统计信息。

QMGR
自动定义的集群发送方通道的统计信息收集由队列管理器属性 STATCHL 的值控制。

这是缺省值。

z/OS 在 z/OS 系统上，启用此参数会直接开启统计信息数据收集，而不考虑您选择的值。 指定 LOW、MEDIUM 或 HIGH 对您的结果没有差别。必须启用此参数以收集通道记帐记录。

例如，要对发送方通道 QM1.TO.QM2 启用具有中等详细信息级别的统计信息收集，请使用以下 MQSC 命令:

```
ALTER CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) STATCHL(MEDIUM)
```

要在中等详细级别对所有将通道属性 STATCHL 指定为 QMGR 的通道启用统计信息收集，请使用以下 MQSC 命令:

```
ALTER QMGR STATCHL(MEDIUM)
```

要在中等详细级别对所有自动定义的集群发送方通道启用统计信息信息收集，请使用以下 MQSC 命令:

```
ALTER QMGR STATACLS(MEDIUM)
```

统计信息消息生成
将按配置的时间间隔生成统计信息消息，并在队列管理器以受控方式关闭时生成统计信息消息。

配置的时间间隔由 STATINT 队列管理器属性控制，该属性指定统计信息消息生成之间的时间间隔 (以秒计)。缺省统计时间间隔为 1800 秒 (30 分钟)。 要更改统计信息时间间隔，请使用 MQSC 命令 ALTER QMGR 并指定 **STATINT** 参数。 例如，要将统计信息时间间隔更改为 900 秒 (15 分钟) ，请使用以下 MQSC 命令:

```
ALTER QMGR STATINT(900)
```

要在统计信息收集时间间隔到期之前将当前收集的统计信息数据写入统计信息队列，请使用 MQSC 命令 RESET QMGR TYPE(STATISTICS)。 发出此命令会将收集的统计信息数据写入统计信息队列并开始新的统计信息数据收集时间间隔。

显示记帐和统计信息

要使用记帐和统计信息消息中记录的信息，请运行应用程序 (例如 **amqsmon** 样本程序) 以将记录的信息转换为适当的格式

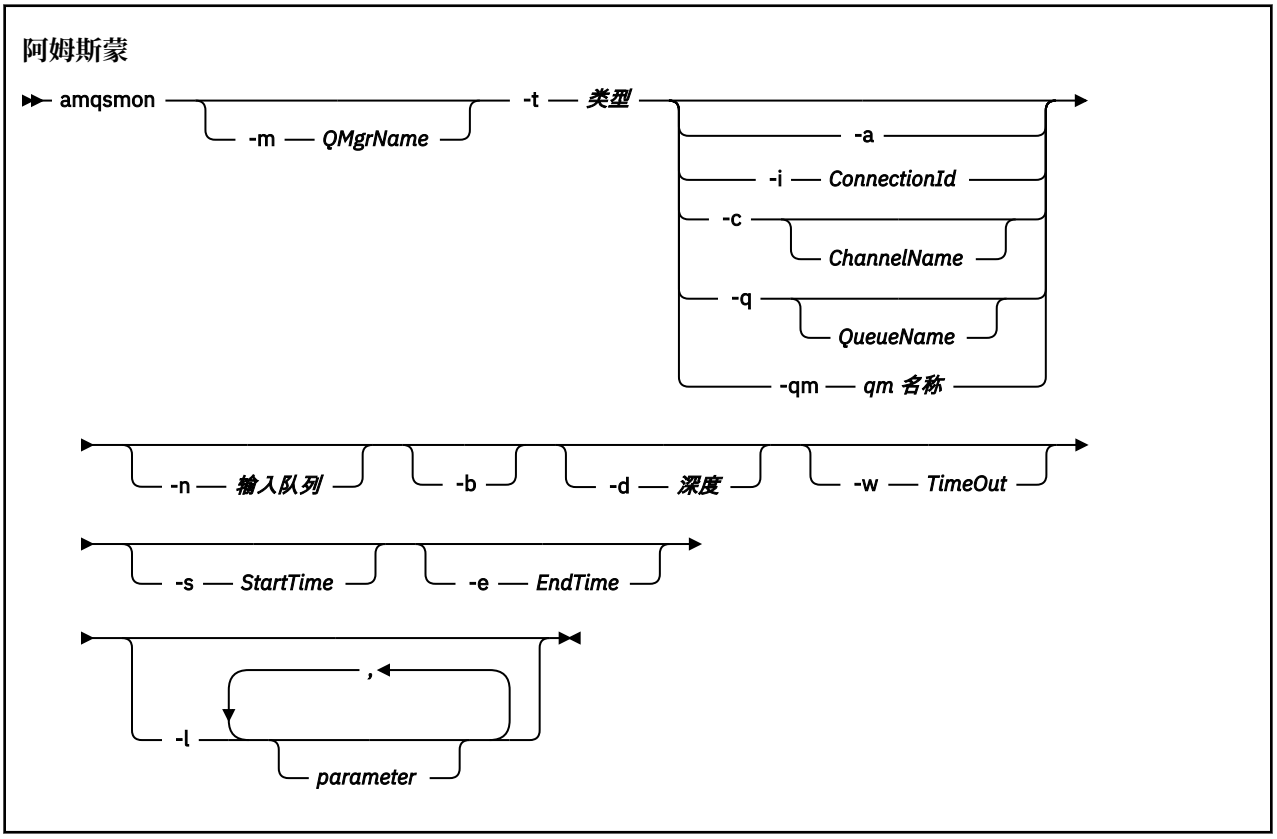
记帐和统计信息消息将写入系统记帐和统计信息队列。 **amqsmon** 是 IBM MQ 随附的样本程序，用于处理来自记帐和统计信息队列的消息，并以可读格式将信息显示到屏幕上。

由于 **amqsmon** 是样本程序，因此您可以使用提供的源代码作为模板来编写自己的应用程序以处理记帐或统计信息消息，或者修改 **amqsmon** 源代码以满足您自己的特定需求。

amqsmon (显示格式化的监视信息)

使用 **amqsmon** 样本程序以可读格式显示记帐和统计信息消息中包含的信息。 **amqsmon** 程序从记帐队列 SYSTEM.ADMIN.ACCOUNTING.QUEUE。 并从统计信息队列 SYSTEM.ADMIN.STATISTICS.QUEUE。

语法



必需参数

-t *type*
要处理的消息的类型。 将 类型 指定为下列其中一项:

记账
将处理记帐记录。 从系统队列 SYSTEM.ADMIN.ACCOUNTING.QUEUE ， 除非您已使用 -n 参数来选择要从中读取的特定队列。

statistics

处理统计信息记录。 从系统队列 SYSTEM.ADMIN.STATISTICS.QUEUE V 9.4.0 ， 除非您已使用 -n 参数来选择要从中读取的特定队列。

可选参数

-m *QMgrName*
要从中处理记帐或统计信息消息的队列管理器的名称。

如果未指定此参数，那么将使用缺省队列管理器。

-a
仅处理包含 MQI 记录的消息。

仅显示 MQI 记录。 不包含 MQI 记录的消息将始终保留在从中读取消息的队列中。

-q *QueueName*
QueueName 是可选参数。

如果未提供 *QueueName* : 仅显示队列记帐和队列统计信息记录。

如果提供了 *QueueName* :

仅显示由 *QueueName* 指定的队列的队列记帐和队列统计信息记录。

如果未指定 *-b* ，那么将废弃从中生成记录的记帐和统计信息消息。 由于记帐和统计信息消息还可以包含来自其他队列的记录，因此如果未指定 *-b* ，那么可以废弃不可见的记录。

-c *ChannelName*

ChannelName 是可选参数。

如果未提供 *ChannelName* :

仅显示通道统计信息记录。

如果提供了 *ChannelName* :

仅显示由 *ChannelName* 指定的通道的通道统计信息记录。

如果未指定 *-b* ，那么将废弃从中获取记录的统计信息消息。 由于统计信息消息还可以包含来自其他通道的记录，因此如果未指定 *-b* ，那么可以废弃不可见的记录。

仅当显示统计信息消息 (*-t statistics*) 时，此参数才可用。

V 9.4.0 **-qm *qm* 名称**

此参数是可选的，但如果指定此参数，那么必须提供队列管理器名称

指定只应从队列中读取由给定队列管理器生成的记帐或统计信息消息。 通常，仅当来自多个队列管理器的消息已转发到中央队列管理器时才使用。

-i *ConnectionId*

仅显示与 *ConnectionId* 指定的连接标识相关的记录。

仅当显示记帐消息 (*-t accounting*) 时，此参数才可用。

如果未指定 *-b* ，那么将废弃从中获取记录的统计信息消息。 由于统计信息消息还可以包含来自其他通道的记录，因此如果未指定 *-b* ，那么可以废弃不可见的记录。

V 9.4.0 **-n 输入队列**

要从中读取记帐或统计信息消息的队列。 如果未指定此值，那么将使用所选 -t *<type>* 的缺省队列。

-b

浏览消息。

以非破坏性方式检索消息。

-d 深度

可处理的最大消息数。

如果未指定此参数，那么可以处理数量不限的消息。

-w *timeout*

等待消息变为可用的最长时间 (以秒为单位)。

如果未指定此参数，那么一旦没有更多要处理的消息， `amqsmon` 将结束。

-s *StartTime*

仅处理放在指定 *StartTime* 之后的消息。

StartTime 以 `yyyy-mm-dd hh.mm.ss` 格式指定。 如果指定了没有时间的日期，那么时间将缺省为指定日期的 `00.00.00` 。 时间为 GMT。

有关不指定此参数的影响，请参阅 注意 1。

-e *EndTime*

仅将处理消息放在指定的 *EndTime* 之前。

EndTime 以 `yyyy-mm-dd hh.mm.ss` 格式指定。 如果指定了没有时间的日期，那么时间将缺省为指定日期的 `23.59.59` 。 时间为 GMT。

有关不指定此参数的影响，请参阅 注意 1。

-l 参数

仅显示所处理记录中的所选字段。 *Parameter* 是整数值的逗号分隔列表，每个整数值都映射到字段的数字常量，请参阅 amqsmon 示例 5。

如果未指定此参数，那么将显示所有可用字段。

注:

1. 如果未指定 *-s StartTime* 或 *-e EndTime*，那么可以处理的消息不受放置时间限制。

amqsmon (显示格式化监视信息) 示例

使用此页面来查看运行 amqsmon (显示格式化监视信息) 样本程序的示例

1. 请参阅 第 168 页的『队列统计信息消息数据』 以获取属性的说明。

   以下命令显示来自队列管理器 `saturn.queue.manager` 的所有 MQI 统计信息消息:

   ```
   amqsmon -m saturn.queue.manager -t statistics -a
   ```

   此命令的输出如下所示:

   ```
   RecordType: MQIStatistics
   QueueManager: 'saturn.queue.manager'
   IntervalStartDate: '2005-04-30'
   IntervalStartTime: '15.09.02'
   IntervalEndDate: '2005-04-30'
   IntervalEndTime: '15.39.02'
   CommandLevel: 600
   ConnCount: 23
   ConnFailCount: 0
   ConnsMax: 8
   DiscCount: [17, 0, 0]
   OpenCount: [0, 80, 1, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0]
   OpenFailCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   CloseCount: [0, 73, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
   CloseFailCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   InqCount: [4, 2102, 0, 0, 0, 46, 0, 0, 0, 0, 0, 0, 0]
   InqFailCount: [0, 31, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   SetCount: [0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
   SetFailCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
   PutCount: [26, 1]
   PutFailCount: 0
   Put1Count: [40, 0]
   Put1FailCount: 0
   PutBytes: [57064, 12320]
   GetCount: [18, 1]
   GetBytes: [52, 12320]
   GetFailCount: 2254
   BrowseCount: [18, 60]
   BrowseBytes: [23784, 30760]
   BrowseFailCount: 9
   CommitCount: 0
   CommitFailCount: 0
   BackCount: 0
   ExpiredMsgCount: 0
   PurgeCount: 0
   ```

2. 以下命令显示队列管理器 `saturn.queue.manager` 上队列 `LOCALQ` 的所有队列统计信息消息:

   ```
   amqsmon -m saturn.queue.manager -t statistics -q LOCALQ
   ```

   此命令的输出如下所示:

   ```
   RecordType: QueueStatistics
   QueueManager: 'saturn.queue.manager'
   IntervalStartDate: '2005-04-30'
   IntervalStartTime: '15.09.02'
   IntervalEndDate: '2005-04-30'
   IntervalEndTime: '15.39.02'
   ```

```
    CommandLevel: 600
    ObjectCount: 3
    QueueStatistics:
      QueueName: 'LOCALQ'
      CreateDate: '2005-03-08'
      CreateTime: '17.07.02'
      QueueType: Predefined
      QueueDefinitionType: Local
      QMinDepth: 0
      QMaxDepth: 18
      AverageQueueTime: [29827281, 0]
      PutCount: [26, 0]
      PutFailCount: 0
      Put1Count: [0, 0]
      Put1FailCount: 0
      PutBytes: [88, 0]
      GetCount: [18, 0]
      GetBytes: [52, 0]
      GetFailCount: 0
      BrowseCount: [0, 0]
      BrowseBytes: [0, 0]
      BrowseFailCount: 1
      NonQueuedMsgCount: 0
      ExpiredMsgCount: 0
      PurgedMsgCount: 0
```

3. 以下命令显示自 2005 年 4 月 30 日 15:30 以来从队列管理器 saturn.queue.manager 记录的所有统计信息消息。

```
amqsmon -m saturn.queue.manager -t statistics -s "2005-04-30 15.30.00"
```

此命令的输出如下所示:

```
    RecordType: MQIStatistics
    QueueManager: 'saturn.queue.manager'
    IntervalStartDate: '2005-04-30'
    IntervalStartTime: '15.09.02'
    IntervalEndDate: '2005-04-30'
    IntervalEndTime: '15.39.02'
    CommandLevel: 600
    ConnCount: 23
    ConnFailCount: 0
    ConnsMax: 8
    DiscCount: [17, 0, 0]
    OpenCount: [0, 80, 1, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0]
      ...
    RecordType: QueueStatistics
    QueueManager: 'saturn.queue.manager'
    IntervalStartDate: '2005-04-30'
    IntervalStartTime: '15.09.02'
    IntervalEndDate: '2005-04-30'
    IntervalEndTime: '15.39.02'
    CommandLevel: 600
    ObjectCount: 3
    QueueStatistics: 0
      QueueName: 'LOCALQ'
      CreateDate: '2005-03-08'
      CreateTime: '17.07.02'
      QueueType: Predefined
        ...
    QueueStatistics: 1
      QueueName: 'SAMPLEQ'
      CreateDate: '2005-03-08'
      CreateTime: '17.07.02'
      QueueType: Predefined
        ...
```

4. 请参阅 第 147 页的『队列记帐消息数据』 以获取属性的说明。

以下命令显示 2005 年 4 月 30 日从队列管理器 saturn.queue.manager 记录的所有记帐消息:

```
amqsmon -m saturn.queue.manager -t accounting -s "2005-04-30" -e "2005-04-30"
```

此命令的输出如下所示:

```
RecordType: MQIAccounting
QueueManager: 'saturn.queue.manager'
IntervalStartDate: '2005-04-30'
IntervalStartTime: '15.09.29'
IntervalEndDate: '2005-04-30'
IntervalEndTime: '15.09.30'
CommandLevel: 600
ConnectionId: x'414d514354524556312020202020208d0b3742010a0020'
SeqNumber: 0
ApplicationName: 'amqsput'
ApplicationPid: 8572
ApplicationTid: 1
UserId: 'admin'
ConnDate: '2005-03-16'
ConnTime: '15.09.29'
DiscDate: '2005-03-16'
DiscTime: '15.09.30'
DiscType: Normal
OpenCount: [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
OpenFailCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
CloseCount: [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
CloseFailCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
PutCount: [1, 0]
PutFailCount: 0
PutBytes: [4, 0]
GetCount: [0, 0]
GetFailCount: 0
GetBytes: [0, 0]
BrowseCount: [0, 0]
BrowseFailCount: 0
BrowseBytes: [0, 0]
CommitCount: 0
CommitFailCount: 0
BackCount: 0
InqCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
InqFailCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
SetCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
SetFailCount: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

RecordType: MQIAccounting
QueueManager: 'saturn.queue.manager'
IntervalStartDate: '2005-03-16'
IntervalStartTime: '15.16.22'
IntervalEndDate: '2005-03-16'
IntervalEndTime: '15.16.22'
CommandLevel: 600
ConnectionId: x'414d514354524556312020202020208d0b3742010c0020'
SeqNumber: 0
ApplicationName: 'runmqsc'
ApplicationPid: 8615
ApplicationTid: 1
    ...
```

5. 以下命令将浏览记帐队列并显示 MQI 记帐信息可用的每个应用程序的应用程序名称和连接标识:

```
amqsmon -m saturn.queue.manager -t accounting -b -l 7006,3024
```

此命令的输出如下所示:

```
MonitoringType: QueueAccounting
ConnectionId: x'414d5143514d393035202020202020fcf1855e01e80322'
ApplicationName: 'WebSphere MQ\bin\amqsput.exe'
QueueAccounting: 0

MonitoringType: QueueAccounting
ConnectionId: x'414d5143514d393035202020202020fcf1855e01ea0322'
ApplicationName: 'BM\MQ_4\bin64\MQExplorer.exe'
QueueAccounting: 0
QueueAccounting: 1
QueueAccounting: 2
QueueAccounting: 3
QueueAccounting: 4
QueueAccounting: 5
QueueAccounting: 6
QueueAccounting: 7
QueueAccounting: 8
```

```
QueueAccounting: 9

MonitoringType: QueueAccounting
ConnectionId: x'414d5143514d393035202020202020fcf1855e01e90322'
ApplicationName: 's\IBM\MQ_4\bin64\amqsput.exe'
QueueAccounting: 0

MonitoringType: QueueAccounting
ConnectionId: x'414d5143514d393035202020202020fcf1855e01ef0322'
ApplicationName: 'BM\MQ_4\bin64\MQExplorer.exe'
QueueAccounting: 0
QueueAccounting: 1
QueueAccounting: 2
QueueAccounting: 3
QueueAccounting: 4
QueueAccounting: 5
QueueAccounting: 6
QueueAccounting: 7
QueueAccounting: 8
QueueAccounting: 9

MonitoringType: QueueAccounting
ConnectionId: x'414d5143514d393035202020202020fcf1855e01e60322'
ApplicationName: 's\IBM\MQ_4\bin64\runmqsc.exe'
QueueAccounting: 0

5 Records Processed
```

有关如何查找本主题中使用的变量的详细信息，请参阅 第 133 页的『查找 ApplicationName (3024) 和 ConnectionId (7006) 的映射』 。

如果在 **amqsmon** 命令中为 **Type** 参数选择了 统计信息 ，请参阅 第 133 页的『查找 AvgTimeOnQ (703) 和 QmaxDepth (739) 的映射』 以获取有关如何查找变量的详细信息。

查找 ***ApplicationName*** *(3024)* 和 ***ConnectionId*** *(7006)* 的映射

使用本主题来说明如何找到示例中使用的变量，以监视记帐数据。

摘要:

- 3024 表示 MQCACF_APPL_NAME ，即 **ApplicationName**
- 7006 表示 MQBACF_CONNECTION_ID ，它是 **ConnectionId**

要找出映射需要执行两个过程:

1. 当您在 **amqsmon** 命令中为 **Type** 参数选择了 记帐 时，请访问 第 147 页的『队列记帐消息数据』 以获取属性 第 149 页的『ApplicationName』 和 第 149 页的『ConnectionId』 的说明。

   在每种情况下，查找属性 **Identifier**。

   ApplicationName 的 **Identifier** 是 MQCACF_APPL_NAME ，而 **ConnectionId** 的是 MQBACF_CONNECTION_ID
2. 搜索在步骤 第 133 页的『1』 中找到的标识。

   转至 常量 部分并向下滚动，直到找到 MQCACF_ * (命令格式字符参数类型) 列表。 找到 MQCACF_APPL_NAME 并看到值 3024。

   同样，查找 MQBACF_ * (命令格式字节参数类型)列表。 找到 MQBACF_CONNECTION_ID 并看到值 7006。

查找 *AvgTimeOnQ (703)* 和 *QmaxDepth (739)* 的映射

使用本主题来说明如何找到示例中使用的变量以监视统计信息数据。

摘要:

- 703 表示 MQIAMO64_AVG_Q_TIME ，即 **AvgTimeOnQ**
- 739 表示 MQIAMO_Q_MAX_DEPTH ，即 **QMaxDepth**

要找出映射需要执行两个过程:

1. 在 **amqsmon** 命令中为 **Type** 参数选择 *statistics* 时，请访问 第 168 页的『队列统计信息消息数据』 以获取属性 第 171 页的『AvgTimeOnQ』 和 第 171 页的『QMaxDepth』 的说明。

在每种情况下，查找属性 **Identifier**。

Identifier for **AvgTimeOnQ** 是 MQIAMO64_AVG_Q_TIME 和 for。 **QMaxDepth** 是 MQIAMO_Q_MAX_DEPTH。

2. 搜索在步骤 第 133 页的『1』中找到的标识。

   转至 常量 部分并向下滚动，直到找到 MQIAMO_ * (命令格式整数监视参数类型) 列表。 找到 MQIAMO_Q_MAX_DEPTH 并看到值 739。

   同样，查找 MQIAMO64_* (命令格式 64 位整数监视参数类型) 列表。 找到 MQIAMO64_AVG_Q_TIME , 您将看到值 703。

记帐和统计信息消息引用

使用此页面来获取记帐和统计信息消息的格式以及这些消息中返回的信息的概述

记帐和统计信息消息是包含消息描述符和消息数据的标准 IBM MQ 消息。 消息数据包含有关 IBM MQ 应用程序执行的 MQI 操作的信息，或者有关 IBM MQ 系统中发生的活动的信息。

消息描述符

- MQMD 结构

消息数据

- PCF 头 (MQCFH)
- 始终返回的记帐或统计信息消息数据
- 返回的记帐或统计信息消息数据 (如果可用)

记帐和统计信息消息格式

使用此页面作为 MQI 记帐消息结构的示例

表 *23: MQI* 记帐消息结构

MQMD 结构	**记帐消息头 MQCFH 结构**	**MQI 记帐消息数据** [1]
结构标识 结构版本 报告选项 消息类型 到期时间 反馈代码 编码 编码字符集标识 消息格式 消息优先级 持久 消息标识 相关标识 回退计数 应答队列 应答队列管理器 用户标识 记帐标记 应用程序标识数据 应用程序类型 应用程序名称 放置日期 放置时间 应用程序原始数据 组标识 消息序号 偏移量 消息标志 原始长度	结构类型 结构长度 结构版本 命令标识符 消息序号 控制选项 完成代码 原因码 参数计数	队列管理器 时间间隔启动日期 时间间隔开始时间 时间间隔结束日期 时间间隔结束时间 命令级别 连接标识 序列号 应用程序名称 应用程序进程标识 应用程序线程标识 用户标识 连接日期 连接时间 连接名称 通道名称 断开连接日期 断开连接时间 断开连接类型 打开计数 打开失败计数 关闭计数 关闭失败计数 放入计数 PUT 失败计数 Put1 计数 PUT1 失败计数 放入字节数 获取计数 获取失败计数 获取字节数 浏览计数 浏览失败计数 浏览字节数 落实计数 落实失败计数 回退计数 查询计数 查询失败计数 设置计数 设置失败计数

注:

1. 显示的参数是针对 MQI 记帐消息返回的参数。 实际记帐或统计信息消息数据取决于消息类别。

记帐和统计信息消息 *MQMD (*消息描述符*)*

使用此页面来了解记帐消息和统计信息消息的消息描述符与事件消息的消息描述符之间的差异

记帐和统计信息消息的消息描述符中的参数和值与事件消息的消息描述符中的参数和值相同，但有以下异常:

Format

描述: 消息数据的格式名。

数据类型: MQCHAR8.

值:

MQFMT_ADMIN
管理消息。

记帐和统计信息消息的消息描述符中包含的某些参数包含由生成消息的队列管理器提供的固定数据。

MQMD 还指定放入消息的队列管理器的名称 (截断为 28 个字符) , 以及将消息放入记帐或统计信息队列的日期和时间。

记帐和统计信息消息中的消息数据

记帐和统计信息消息中的消息数据基于 PCF 命令查询和响应中使用的可编程命令格式 (PCF)。 记帐和统计信息消息中的消息数据由 PCF 头 (MQCFH) 和记帐或统计信息报告组成。

记帐和统计信息消息 MQCFH (PCF 头)

记帐和统计信息消息的消息头是 MQCFH 结构。 记帐和统计信息消息的消息头中的参数和值与事件消息的消息头中的参数和值相同，但有以下例外:

Command

描述: 命令标识。 这将标识记帐或统计信息消息类别。

数据类型: MQLONG。

值:

MQCMD_ACCOUNTING_MQI
MQI 记帐消息。

MQCMD_ACCOUNTING_Q
队列记帐消息。

MQCMD_STATISTICS_MQI
MQI 统计信息消息。

MQCMD_STATISTICS_Q
队列统计信息消息。

MQCMD_STATISTICS_CHANNEL
通道统计信息消息。

Version

描述: 结构版本号。

数据类型: MQLONG。

值:

MQCFH_VERSION_3
Version-3 表示记帐和统计信息消息。

记帐和统计信息消息数据

记帐和统计信息数据的内容取决于记帐或统计信息的类别，如下所示:

MQI 记帐消息
MQI 记帐消息数据由多个 PCF 参数组成，但没有 PCF 组。

队列记帐消息
队列记帐消息数据由多个 PCF 参数组成，范围为 1 到 100 *QAccountingData* PCF 组。

MQI 统计信息消息
MQI 统计信息消息数据由多个 PCF 参数组成，但没有 PCF 组。

队列统计信息消息
队列统计信息消息数据由多个 PCF 参数组成，范围为 1 到 100 *QStatisticsData* PCF 组。

通道统计信息消息
通道统计信息消息数据由多个 PCF 参数组成，范围为 1 到 100 *ChlStatistics* 数据 PCF 组。

MQI 记帐消息数据

使用此页面来查看 MQI 记帐消息的结构

消息名称:	MQI 记帐消息。
平台模式:	全部， IBM MQ for z/OS 除外。
系统队列:	SYSTEM.ADMIN.ACCOUNTING.QUEUE.

QueueManager

描述:	队列管理器的名称
标识:	MQCA_Q_MGR_NAME
数据类型:	MQCFST
最大长度:	MQ_Q_MGR_NAME_LENGTH
已返回:	始终

IntervalStartDate

描述:	监视时间段开始的日期
标识:	MQCAMO_START_DATE
数据类型:	MQCFST
最大长度:	MQ_DATE_LENGTH
已返回:	始终

IntervalStartTime

描述:	监控周期开始的时间
标识:	MQCAMO_START_TIME
数据类型:	MQCFST
最大长度:	MQ_TIME_LENGTH
已返回:	始终

IntervalEndDate

描述:	监测期结束日期
标识:	MQCAMO_END_DATE
数据类型:	MQCFST
最大长度:	MQ_DATE_LENGTH
已返回:	始终

IntervalEndTime

描述:	监控周期结束的时间
标识:	MQCAMO_END_TIME

数据类型: MQCFST

最大长度: MQ_TIME_LENGTH

已返回: 始终

CommandLevel

描述: 队列管理器命令级别

标识: MQIA_COMMAND_LEVEL

数据类型: MQCFIN

已返回: 始终

ConnectionId

描述: IBM MQ 连接的连接标识

标识: MQBACF_CONNECTION_ID

数据类型: MQCFBS

最大长度: MQ_CONNECTION_ID_LENGTH

已返回: 始终

SeqNumber

描述: 序号。 对于长时间运行的连接的每个后续记录，此值将递增。

标识: MQIACF_SEQUENCE_NUMBER

数据类型: MQCFIN

已返回: 始终

ApplicationName

描述: 应用程序的名称。 此字段的内容等同于消息描述符中 *PutApplName* 字段的内容。

标识: MQCACF_APPL_NAME

数据类型: MQCFST

最大长度: MQ_APPL_NAME_LENGTH

已返回: 始终

ApplicationPid

描述: 应用程序的操作系统进程标识

标识: MQIACF_PROCESS_ID

数据类型: MQCFIN

已返回: 始终

ApplicationTid

描述: 应用程序中连接的 IBM MQ 线程标识

标识: MQIACF_THREAD_ID

数据类型: MQCFIN

已返回: 始终

UserId

描述: 应用程序的用户标识上下文
标识: MQCACF_USER_IDENTIFIER
数据类型: MQCFST
最大长度: MQ_USER_ID_LENGTH
已返回: 始终

ConnDate

描述: MQCONN 操作的日期
标识: MQCAMO_CONN_DATE
数据类型: MQCFST
最大长度: MQ_TIME_LENGTH
已返回: 可用时

ConnTime

描述: MQCONN 操作的时间
标识: MQCAMO_CONN_TIME
数据类型: MQCFST
最大长度: MQ_TIME_LENGTH
已返回: 可用时

ConnName

描述: 客户机连接的连接名称
标识: MQCACH_CONNECTION_NAME
数据类型: MQCFST
最大长度: MQ_CONN_NAME_LENGTH
已返回: 可用时

ChannelName

描述: 客户机连接的通道名称
标识: MQCACH_CHANNEL_NAME
数据类型: MQCFST
最大长度: MQ_CHANNEL_NAME_LENGTH
已返回: 可用时

RemoteProduct

描述: 客户机连接的远程产品标识，如 DISPLAY CHSATUS 的 **RPRODUCT** 字段中所示
标识: MQCACH_REMOTE_PRODUCT
数据类型: MQCFST
最大长度: MQ_REMOTE_PRODUCT_LENGTH
已返回: 可用时

RemoteVersion

描述: 客户机连接的远程产品版本，如 DISPLAY CHSTATUS 的 **RVERSION** 字段中所示

标识: MQCACH_REMOTE_VERSION

数据类型: MQCFST

最大长度: MQ_REMOTE_VERSION_LENGTH

已返回: 可用时

DiscDate

描述: MQDISC 操作的日期

标识: MQCAMO_DISC_DATE

数据类型: MQCFST

最大长度: MQ_DATE_LENGTH

已返回: 可用时

DiscTime

描述: MQDISC 操作的时间

标识: MQCAMO_DISC_TIME

数据类型: MQCFST

最大长度: MQ_TIME_LENGTH

已返回: 可用时

DiscType

描述: 断开连接的类型

标识: MQIAMO_DISC_TYPE

数据类型: MQCFIN

值: 可能的值为:

MQDISCONNECT_NORMAL
由应用程序请求

MQDISCONNECT_IMPLICIT
异常应用程序终止

MQDISCONNECT_Q_MGR
队列管理器中断的连接

已返回: 可用时

OpenCount

描述: 通过直接对 MQOPEN 发出调用或使用 MQPUT1 动词成功打开的对象数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_OPENS

数据类型: MQCFIL

已返回: 可用时

OpenFailCount

描述: 尝试打开对象失败的次数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_OPENS_FAILED

数据类型: MQCFIL

已返回: 可用时

CloseCount

描述: 已关闭的对象数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_CLOSES

数据类型: MQCFIL

已返回: 可用时

CloseFailCount

描述: 尝试关闭对象失败的次数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_CLOSES_FAILED

数据类型: MQCFIL

已返回: 可用时

PutCount

描述: 成功放入队列的持久和非持久消息数，但使用 MQPUT1 调用放入的消息除外。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_PUTS

数据类型: MQCFIL

已返回: 可用时

PutFailCount

描述: 尝试放入消息失败的次数

标识: MQIAMO_PUTS_FAILED

数据类型: MQCFIN

已返回: 可用时

Put1Count

描述: 使用 MQPUT1 调用成功放入队列的持久和非持久消息数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_PUT1S

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

Put1FailCount

描述: 尝试使用 MQPUT1 调用放入消息失败的次数

标识: MQIAMO_PUT1S_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

PutBytes

描述: 使用持久和非持久消息的 put 调用写入的字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_PUT_BYTES

数据类型: MQCFIL64

已返回: 可用时

GetCount

描述: 针对持久消息和非持久消息的成功破坏性 MQGET 调用数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_GETS

数据类型: MQCFIL

已返回: 可用时

GetFailCount

描述: 失败的破坏性 MQGET 调用数

标识: MQIAMO_GETS_FAILED

数据类型: MQCFIN

已返回: 可用时

GetBytes

描述: 针对持久和非持久消息检索的总字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_GET_BYTES

数据类型: MQCFIL64

已返回: 可用时

BrowseCount

描述: 针对持久和非持久消息的成功非破坏性 MQGET 调用数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_BROWSES

数据类型: MQCFIL

已返回: 可用时

BrowseFailCount

描述: 不成功的非破坏性 MQGET 调用数

标识: MQIAMO_BROWSES_FAILED

数据类型: MQCFIN

已返回: 可用时

BrowseBytes

描述: 为持久和非持久消息浏览的总字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_BROWSE_BYTES

数据类型: MQCFIL64

已返回: 可用时

CommitCount

描述: 成功事务数。 此数目包括已连接的应用程序隐式落实的那些事务。 没有未完成工作的落实请求将包括在此计数中。

标识: MQIAMO_COMMITS

数据类型: MQCFIN

已返回: 可用时

CommitFailCount

描述: 尝试完成事务失败的次数

标识: MQIAMO_COMMITS_FAILED

数据类型: MQCFIN

已返回: 可用时

BackCount

描述: 已处理的回退数，包括由于异常断开连接而导致的隐式回退

标识: MQIAMO_BACKOUTS

数据类型: MQCFIN

已返回: 可用时

InqCount

描述: 查询的成功对象数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_INQS

数据类型: MQCFIL

已返回: 可用时

InqFailCount

描述: 对象查询尝试失败的次数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_INQS_FAILED

数据类型: MQCFIL

已返回: 可用时

SetCount

描述: 成功 MQSET 调用的次数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_SETS

数据类型: MQCFIL

已返回: 可用时

SetFailCount

描述: 失败的 MQSET 调用数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_SETS_FAILED

数据类型: MQCFIL

已返回: 可用时

SubCountDur

描述: 创建，更改或恢复持久预订的成功预订请求数。 这是按操作类型建立索引的值的数组

0 = 创建的预订数

1 = 已变更的预订数

2 = 已恢复的预订数

标识: MQIAMO_SUBS_DUR

数据类型: MQCFIL

已返回: 可用时。

SubCountNDur

描述: 创建，变更或恢复非持久预订的成功预订请求数。 这是按操作类型建立索引的值的数组

0 = 创建的预订数

1 = 已变更的预订数

2 = 已恢复的预订数

标识: MQIAMO_SUBS_NDUR

数据类型: MQCFIL

已返回: 可用时。

SubFailCount

描述: 失败的预订请求数。

标识: MQIAMO_SUBS_FAILED

数据类型: MQCFIN

已返回: 可用时。

UnsubCountDur

描述: 持久预订的成功取消预订请求数。 这是按操作类型建立索引的值的数组
0-预订已关闭但未除去
1-已关闭并除去预订

标识: MQIAMO_UNSUBS_DUR

数据类型: MQCFIL

已返回: 可用时。

UnsubCountNDur

描述: 持久预订的成功取消预订请求数。 这是按操作类型建立索引的值的数组
0-预订已关闭但未除去
1-已关闭并除去预订

标识: MQIAMO_UNSUBS_NDUR

数据类型: MQCFIL

已返回: 可用时。

UnsubFailCount

描述: 不成功的取消预订请求数。

标识: MQIAMO_UNSUBS_FAILED

数据类型: MQCFIN

已返回: 可用时。

SubRqCount

描述: 成功的 MQSUBRQ 请求数。

标识: MQIAMO_SUBRQS

数据类型: MQCFIN

已返回: 可用时。

SubRqFailCount

描述: 失败的 MQSUB 请求数。

标识: MQIAMO_SUBRQS_FAILED

数据类型: MQCFIN

已返回: 可用时。

CBCount

描述: 成功的 MQCB 请求数。 这是按操作类型建立索引的值的数组
0-已创建或变更回调
1-已除去回调
2-已恢复回调
3-已暂挂回调

标识: MQIAMO_CBS

数据类型: MQCFIN

已返回: 可用时。

CBFailCount

描述: 失败的 MQCB 请求数。

标识: MQIAMO_CBS_FAILED

数据类型: MQCFIN

已返回: 可用时。

CtlCount

描述: 成功的 MQCTL 请求数。 这是按操作类型建立索引的值的数组

0-连接已启动

1-连接已停止

2-连接已恢复

3-连接已暂挂

标识: MQIAMO_CTLS

数据类型: MQCFIL

已返回: 可用时。

CtlFailCount

描述: 失败的 MQCTL 请求数。

标识: MQIAMO_CTLS_FAILED

数据类型: MQCFIN

已返回: 可用时。

StatCount

描述: 成功的 MQSTAT 请求数。

标识: MQIAMO_STATS。

数据类型: MQCFIN

已返回: 可用时。

StatFailCount

描述: 失败的 MQSTAT 请求数。

标识: MQIAMO_STATS_FAILED

数据类型: MQCFIN

已返回: 可用时。

PutTopicCount

描述: 成功放入主题的持久和非持久消息数，使用 MQPUT1 调用放入的消息除外。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

注: 使用解析为主题的队列别名放入的消息包含在此值中。

标识: MQIAMO_TOPIC_PUTS

数据类型: MQCFIL

已返回: 可用时。

PutTopicFailCount

描述: 尝试将消息放入主题失败的次数。

标识: MQIAMO_TOPIC_PUTS_FAILED

数据类型: MQCFIN

已返回: 可用时。

Put1TopicCount

描述: 使用 MQPUT1 调用成功放入主题的持久和非持久消息数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

注: 使用解析为主题的队列别名放入的消息包含在此值中。

标识: MQIAMO_TOPIC_PUT1S

数据类型: MQCFIL

已返回: 可用时。

Put1TopicFailCount

描述: 尝试使用 MQPUT1 调用将消息放入主题失败的次数。

标识: MQIAMO_TOPIC_PUT1S_FAILED

数据类型: MQCFIN

已返回: 可用时。

PutTopicBytes

描述: 对解析为发布操作的持久和非持久消息使用 put 调用写入的字节数。 这是应用程序放置的字节数，而不是传递给订户的结果字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_TOPIC_PUT_BYTES

数据类型: MQCFIL64

已返回: 可用时。

队列记帐消息数据

使用此页面来查看队列记帐消息的结构

消息名称:	队列记帐消息。
平台模式:	全部， IBM MQ for z/OS 除外。
系统队列:	SYSTEM.ADMIN.ACCOUNTING.QUEUE.

QueueManager

描述: 队列管理器的名称
标识: MQCA_Q_MGR_NAME
数据类型: MQCFST
最大长度: MQ_Q_MGR_NAME_LENGTH
已返回: 始终

IntervalStartDate

描述: 监视时间段开始的日期
标识: MQCAMO_START_DATE
数据类型: MQCFST
最大长度: MQ_DATE_LENGTH
已返回: 始终

IntervalStartTime

描述: 监控周期开始的时间
标识: MQCAMO_START_TIME
数据类型: MQCFST
最大长度: MQ_TIME_LENGTH
已返回: 始终

IntervalEndDate

描述: 监测期结束日期
标识: MQCAMO_END_DATE
数据类型: MQCFST
最大长度: MQ_DATE_LENGTH
已返回: 始终

IntervalEndTime

描述: 监控周期结束的时间
标识: MQCAMO_END_TIME
数据类型: MQCFST
最大长度: MQ_TIME_LENGTH
已返回: 始终

CommandLevel

描述: 队列管理器命令级别
标识: MQIA_COMMAND_LEVEL
数据类型: MQCFIN
已返回: 始终

ConnectionId

描述: IBM MQ 连接的连接标识

标识: MQBACF_CONNECTION_ID

数据类型: MQCFBS

最大长度: MQ_CONNECTION_ID_LENGTH

已返回: 始终

SeqNumber

描述: 序号。 对于长时间运行的连接的每个后续记录，此值将递增。

标识: MQIACF_SEQUENCE_NUMBER

数据类型: MQCFIN

已返回: 始终

ApplicationName

描述: 应用程序的名称。 此字段的内容等同于消息描述符中 PutApplName 字段的内容。

标识: MQCACF_APPL_NAME

数据类型: MQCFST

最大长度: MQ_APPL_NAME_LENGTH

已返回: 始终

ApplicationPid

描述: 应用程序的操作系统进程标识

标识: MQIACF_PROCESS_ID

数据类型: MQCFIN

已返回: 始终

ApplicationTid

描述: 应用程序中连接的 IBM MQ 线程标识

标识: MQIACF_THREAD_ID

数据类型: MQCFIN

已返回: 始终

UserId

描述: 应用程序的用户标识上下文

标识: MQCACF_USER_IDENTIFIER

数据类型: MQCFST

最大长度: MQ_USER_ID_LENGTH

已返回: 始终

ChannelName

描述: 客户机连接的通道名称

标识: MQCACH_CHANNEL_NAME

数据类型: MQCFST

最大长度: MQ_CHANNEL_NAME_LENGTH

已返回: 可用时

ConnName

描述: 客户机连接的连接名称

标识: MQCACH_CONNECTION_NAME

数据类型: MQCFST

最大长度: MQ_CONN_NAME_LENGTH

已返回: 可用时

ObjectCount

描述: 在已记录记帐数据的时间间隔内访问的队列数。 此值设置为消息中包含的 *QAccountingData* PCF 组数。

标识: MQIAMO_OBJECT_COUNT

数据类型: MQCFIN

已返回: 始终

QAccountingData

描述: 用于指定队列的记帐详细信息的分组参数

标识: MQGACF_Q_ACCOUNTING_DATA

数据类型: MQCFGR

组中的参数:

QName
CreateDate
CreateTime
QType
QDefinitionType
OpenCount
OpenDate
OpenTime
CloseDate
CloseTime
PutCount
PutFailCount
Put1Count
Put1FailCount
PutBytes
PutMinBytes
PutMaxBytes
GetCount
GetFailCount
GetBytes
GetMinBytes
GetMaxBytes
BrowseCount
BrowseFailCount
BrowseBytes
BrowseMinBytes
BrowseMaxBytes
TimeOnQMin
TimeOnQAvg
TimeOnQMax

已返回: 始终

QName

描述: 队列的名称
标识: MQCA_Q_NAME
数据类型: MQCFST
包含在 PCF 组中: *QAccountingData*
最大长度: MQ_Q_NAME_LENGTH
已返回: 可用时

CreateDate

描述: 创建队列的日期
标识: MQCA_CREATION_DATE
数据类型: MQCFST
包含在 PCF 组中: *QAccountingData*
最大长度: MQ_DATE_LENGTH

已返回: 可用时

CreateTime

描述: 创建队列的时间

标识: MQCA_CREATION_TIME

数据类型: MQCFST

包含在 PCF 组中: *QAccountingData*

最大长度: MQ_TIME_LENGTH

已返回: 可用时

QType

描述: 队列的类型

标识: MQIA_Q_TYPE

数据类型: MQCFIN

包含在 PCF 组中: *QAccountingData*

值: MQQT_LOCAL

已返回: 可用时

QDefinitionType

描述: 队列定义类型

标识: MQIA_DEFINITION_TYPE

数据类型: MQCFIN

包含在 PCF 组中: *QAccountingData*

值: 可能的值为:

MQQDT_PREDEFINED
MQQDT_PERMANENT_DYNAMIC
MQQDT_TEMPORARY_DYNAMIC

已返回: 可用时

OpenCount

描述: 在此时间间隔内，应用程序通过直接对 MQOPEN 发出调用或使用 MQPUT1 动词打开此队列的次数。

标识: MQIAMO_OPENS

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

OpenDate

描述: 在此记录时间间隔内首次打开队列的日期。 如果队列已在此时间间隔开始时打开，那么此值反映最初打开队列的日期。

标识: MQCAMO_OPEN_DATE

数据类型: MQCFST

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

OpenTime

描述: 在此记录时间间隔内首次打开队列的时间。 如果在此时间间隔开始时队列已打开，那么此值反映最初打开队列的时间。

标识: MQCAMO_OPEN_TIME

数据类型: MQCFST

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

CloseDate

描述: 在此记录时间间隔内最终关闭队列的日期。 如果队列仍处于打开状态，那么不会返回值。

标识: MQCAMO_CLOSE_DATE

数据类型: MQCFST

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

CloseTime

描述: 在此记录时间间隔内最终关闭队列的时间。 如果队列仍处于打开状态，那么不会返回值。

标识: MQCAMO_CLOSE_TIME

数据类型: MQCFST

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

PutCount

描述: 成功放入队列的持久和非持久消息数， MQPUT1 调用除外。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_PUTS

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

PutFailCount

描述: 尝试放入消息失败的次数 (MQPUT1 调用除外)

标识: MQIAMO_PUTS_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

Put1Count

描述: 使用 MQPUT1 调用成功放入队列的持久和非持久消息数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_PUT1S

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

Put1FailCount

描述: 尝试使用 MQPUT1 调用放入消息失败的次数

标识: MQIAMO_PUT1S_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

PutBytes

描述: 持久消息和非持久消息的总放入字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_PUT_BYTES

数据类型: MQCFIL64

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

PutMinBytes

描述: 放入队列中的最小持久和非持久消息大小。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_PUT_MIN_BYTES

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

PutMaxBytes

描述: 放置在队列上的最大持久和非持久消息大小。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_PUT_MAX_BYTES

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

GeneratedMsgCount

描述: 生成的消息数。 生成的消息为

- 队列深度高事件
- 队列深度低事件

标识: MQIAMO_GENERATED_MSGS

数据类型: MQCFIN

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

GetCount

描述: 针对持久消息和非持久消息的成功破坏性 MQGET 调用数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_GETS

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

GetFailCount

描述: 失败的破坏性 MQGET 调用数

标识: MQIAMO_GETS_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

GetBytes

描述: 在针对持久和非持久消息的破坏性 MQGET 调用中读取的字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_GET_BYTES

数据类型: MQCFIL64

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

GetMinBytes

描述: 在队列中检索到的最小持久消息和非持久消息的大小。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_GET_MIN_BYTES

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

GetMaxBytes

描述: 在队列中检索到的最大持久和非持久消息的大小。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_GET_MAX_BYTES

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

BrowseCount

描述: 针对持久和非持久消息的成功非破坏性 MQGET 调用数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_BROWSES

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

BrowseFailCount

描述: 不成功的非破坏性 MQGET 调用数

标识: MQIAMO_BROWSES_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

BrowseBytes

描述: 在返回持久消息的非破坏性 MQGET 调用中读取的字节数

标识: MQIAMO64_BROWSE_BYTES

数据类型: MQCFIL64

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

BrowseMinBytes

描述: 从队列中浏览的最小持久消息和非持久消息的大小。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_BROWSE_MIN_BYTES

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

BrowseMaxBytes

描述: 从队列浏览的最大持久和非持久消息的大小。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_BROWSE_MAX_BYTES

数据类型: MQCFIL

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

TimeOnQMin

描述: 以破坏性方式检索持久和非持久消息之前保留在队列中的最短时间 (以微秒为单位)。 对于在同步点下检索的消息，此值不包括落实 get 操作之前的时间。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_Q_TIME_MIN

数据类型: MQCFIL64

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

TimeOnQAvg

描述: 以破坏性方式检索持久和非持久消息之前保留在队列中的平均时间 (以微秒为单位)。 对于在同步点下检索的消息，此值不包括落实 get 操作之前的时间。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_Q_TIME_AVG

数据类型: MQCFIL64

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

TimeOnQMax

描述: 以破坏性方式检索持久和非持久消息之前保留在队列中的最长时间 (以微秒为单位)。 对于在同步点下检索的消息，此值不包括落实 get 操作之前的时间。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_Q_TIME_MAX

数据类型: MQCFIL64

包含在 PCF 组中: *QAccountingData*

已返回: 可用时

CBCount

描述: 成功的 MQCB 请求数。 这是按操作类型建立索引的值的数组

0-已创建或变更回调

1-已除去回调

2-已恢复回调

3-已暂挂回调

标识: MQIAMO_CBS

数据类型: MQCFIN

已返回: 可用时。

CBFailCount

描述: 失败的 MQCB 请求数。

标识: MQIAMO_CBS_FAILED

数据类型: MQCFIN

已返回: 可用时。

MQI 统计信息消息数据

使用此页面来查看 MQI 统计信息消息的结构

消息名称:	MQI 统计信息消息。
平台模式:	全部， IBM MQ for z/OS 除外。
系统队列:	SYSTEM.ADMIN.STATISTICS.QUEUE.

QueueManager

描述: 队列管理器的名称。

标识: MQCA_Q_MGR_NAME。

数据类型: MQCFST。

最大长度: MQ_Q_MGR_NAME_LENGTH。

已返回: 一直等在那里。

IntervalStartDate

描述: 监视时间段开始时的日期。

标识: MQCAMO_START_DATE。

数据类型: MQCFST。

最大长度: MQ_DATE_LENGTH

已返回: 一直等在那里。

IntervalStartTime

描述: 监视时间段开始时的时间。

标识: MQCAMO_START_TIME。

数据类型: MQCFST。

最大长度: MQ_TIME_LENGTH

已返回: 一直等在那里。

IntervalEndDate

描述: 监视时间段结束时的日期。

标识: MQCAMO_END_DATE。

数据类型: MQCFST。

最大长度: MQ_DATE_LENGTH

已返回: 一直等在那里。

IntervalEndTime

描述: 监视周期结束时的时间。

标识: MQCAMO_END_TIME。

数据类型: MQCFST。

最大长度: MQ_TIME_LENGTH

已返回: 一直等在那里。

CommandLevel

描述: 队列管理器命令级别。

标识: MQIA_COMMAND_LEVEL。

数据类型: MQCFIN。

已返回: 一直等在那里。

ConnCount

描述: 与队列管理器的成功连接数。

标识: MQIAMO_CONNS。

数据类型: MQCFIN。

已返回: 可用时。

ConnFailCount

描述: 失败的连接尝试次数。

标识: MQIAMO_CONNS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

ConnsMax

描述: 记录时间间隔内的最大并行连接数。

标识: MQIAMO_CONNS_MAX。

数据类型: MQCFIN。

已返回: 可用时。

DiscCount

描述: 与队列管理器断开连接的次数。 这是一个整数数组，由以下常量建立索引:

- MQDISCONNECT_NORMAL
- MQDISCONNECT_IMPLICIT
- MQDISCONNECT_Q_MGR

标识: MQIAMO_DISCS。

数据类型: MQCFIL。

已返回: 可用时。

OpenCount

描述: 通过直接对 MQOPEN 发出调用或使用 MQPUT1 动词成功打开的对象数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_OPENS。

数据类型: MQCFIL。

已返回: 可用时。

OpenFailCount

描述: 打开对象尝试失败的次数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_OPENS_FAILED。

数据类型: MQCFIL。

已返回: 可用时。

CloseCount

描述: 已成功关闭的对象数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_CLOSES。

数据类型: MQCFIL。

已返回: 可用时。

CloseFailCount

描述: 尝试关闭对象失败的次数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_CLOSES_FAILED。

数据类型: MQCFIL。

已返回: 可用时。

InqCount

描述: 成功查询的对象数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_INQS。

数据类型: MQCFIL。

已返回: 可用时。

InqFailCount

描述: 对象查询尝试失败的次数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_INQS_FAILED。

数据类型: MQCFIL。

已返回: 可用时。

SetCount

描述: 已成功更新的对象数 (SET)。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_SETS。

数据类型: MQCFIL。

已返回: 可用时。

SetFailCount

描述: 不成功的 SET 尝试次数。 此参数是按对象类型建立索引的整数列表，请参阅 参考说明 1。

标识: MQIAMO_SETS_FAILED。

数据类型: MQCFIL。

已返回: 可用时。

PutCount

描述: 成功放入队列的持久和非持久消息数， MQPUT1 请求除外。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO_PUTS。

数据类型: MQCFIL。

已返回: 可用时。

PutFailCount

描述: 失败的放入消息尝试次数。

标识: MQIAMO_PUTS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

Put1Count

描述: 使用 MQPUT1 请求成功放入队列的持久和非持久消息数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2

标识: MQIAMO_PUT1S.

数据类型: MQCFIL。

已返回: 可用时。

Put1FailCount

描述: 尝试使用 MQPUT1 请求将持久和非持久消息放入队列失败的次数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2

标识: MQIAMO_PUT1S_FAILED.

数据类型: MQCFIL。

已返回: 可用时。

PutBytes

描述: 使用 put 请求写入的持久和非持久消息的字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2

标识: MQIAMO64_PUT_BYTES.

数据类型: MQCFIL64.

已返回: 可用时。

GetCount

描述: 针对持久和非持久消息的成功破坏性获取请求数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2

标识: MQIAMO_GETS。

数据类型: MQCFIL。

已返回: 可用时。

GetFailCount

描述: 失败的破坏性获取请求数。

标识: MQIAMO_GETS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

GetBytes

描述: 在针对持久和非持久消息的破坏性获取请求中读取的字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2

标识: MQIAMO64_GET_BYTES.

数据类型: MQCFIL64.

已返回: 可用时。

BrowseCount

描述: 针对持久和非持久消息的成功非破坏性获取请求数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2

标识: MQIAMO_BROWSES。

数据类型: MQCFIL。

已返回: 可用时。

BrowseFailCount

描述: 失败的非破坏性获取请求数。

标识: MQIAMO_BROWSES_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

BrowseBytes

描述: 在针对持久和非持久消息的非破坏性获取请求中读取的字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2

标识: MQIAMO64_BROWSE_BYTES.

数据类型: MQCFIL64.

已返回: 可用时。

CommitCount

描述: 已成功完成的事务数。 此数目包括由应用程序断开连接以隐式方式落实的事务，以及没有未完成工作的落实请求。

标识: MQIAMO_COMMITS。

数据类型: MQCFIN。

已返回: 可用时。

CommitFailCount

描述: 尝试完成事务失败的次数。

标识: MQIAMO_COMMITS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

BackCount

描述: 处理的回退数，包括异常断开连接时的隐式回退。

标识: MQIAMO_BACKOUTS。

数据类型: MQCFIN。

已返回: 可用时。

ExpiredMsgCount

描述: 在可以检索持久和非持久消息之前，由于到期而丢弃的消息数。

标识: MQIAMO_MSGS_EXPIRED。

数据类型: MQCFIN。

已返回: 可用时。

PurgeCount

描述: 已清除队列的次数。

标识: MQIAMO_MSGS_PURGED。

数据类型: MQCFIN。

已返回: 可用时。

SubCountDur

描述: 创建，变更或恢复持久预订的成功预订请求数。 这是按操作类型建立索引的值的数组

0 = 创建的预订数

1 = 已变更的预订数

2 = 已恢复的预订数

标识: MQIAMO_SUBS_DUR。

数据类型: MQCFIL

已返回: 可用时。

SubCountNDur

描述: 创建，变更或恢复非持久预订的成功预订请求数。 这是按操作类型建立索引的值的数组

0 = 创建的预订数

1 = 已变更的预订数

2 = 已恢复的预订数

标识: MQIAMO_SUBS_NDUR。

数据类型: MQCFIL。

已返回: 可用时。

SubFailCount

描述: 失败的预订请求数。

标识: MQIAMO_SUBS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

UnsubCountDur

描述: 持久预订的成功取消预订请求数。 这是按操作类型建立索引的值的数组

0-预订已关闭但未除去

1-已关闭并除去预订

标识: MQIAMO_UNSUBS_DUR。

数据类型: MQCFIL。

已返回: 可用时。

UnsubCountNDur

描述: 非持久预订的成功取消预订请求数。 这是按操作类型建立索引的值的数组

0-预订已关闭但未除去

1-已关闭并除去预订

标识: MQIAMO_UNSUBS_NDUR。

数据类型: MQCFIL。

已返回: 可用时。

UnsubFailCount

描述: 失败的取消预订请求数。

标识: MQIAMO_UNSUBS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

SubRqCount

描述: 成功的 MQSUBRQ 请求数。

标识: MQIAMO_SUBRQS

数据类型: MQCFIN

已返回: 可用时。

SubRqFailCount

描述: 失败的 MQSUBRQ 请求数。

标识: MQIAMO_SUBRQS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

CBCount

描述: 成功的 MQCB 请求数。 这是按操作类型建立索引的值的数组

0-已创建或变更回调

1-已除去回调

2-已恢复回调

3-已暂挂回调

标识: MQIAMO_CBS。

数据类型: MQCFIL。

已返回: 可用时。

CBFailCount

描述: 失败的 MQCB 请求数。

标识: MQIAMO_CBS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

CtlCount

描述: 成功的 MQCTL 请求数。 这是按操作类型建立索引的值的数组:

0-连接已启动

1-连接已停止

2-连接已恢复

3-连接已暂挂

标识: MQIAMO_CTLS。

数据类型: MQCFIL。

已返回: 可用时。

CtlFailCount

描述: 失败的 MQCTL 请求数。

标识: MQIAMO_CTLS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

StatCount

描述: 成功的 MQSTAT 请求数。

标识: MQIAMO_STATS。

数据类型: MQCFIN。

已返回: 可用时。

StatFailCount

描述: 失败的 MQSTAT 请求数。

标识: MQIAMO_STATS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

SubCountDurHighWater

描述: 时间间隔内持久预订数的高水位标记。 这是由 SUBTYPE 建立索引的值数组

0-系统中所有持久预订的高水位标记

1-持久应用程序预订的高水位标记 (MQSUBTYPE_API)

2-持久管理预订的高水位标记 (MQSUBTYPE_ADMIN)

3-持久代理预订的高水位标记 (MQSUBTYPE_PROXY)

标识: MQIAMO_SUB_DUR_HIGHWATER

数据类型: MQCFIL。

已返回: 可用时。

SubCountDurLowWater

描述: 时间间隔内持久预订数的低水位标记。 这是按 SUBTYPE 建立索引的值的数组。

0-系统中所有持久预订的低水位标记

1-持久应用程序预订的低水位标记 (MQSUBTYPE_API)

2-持久管理预订的低水位标记 (MQSUBTYPE_ADMIN)

3-持久代理预订 (MQSUBTYPE_PROXY) 的低水位标记

标识: MQIAMO_SUB_DUR_LOWWATER

数据类型: MQCFIL。

已返回: 可用时。

SubCountNDurHighWater

描述: 时间间隔内非持久预订数的高水位标记。 这是由 SUBTYPE 建立索引的值数组

0-系统中所有非持久预订的高水位标记

1-非持久应用程序预订的高水位标记 (MQSUBTYPE_API)

2-非持久管理预订的高水位标记 (MQSUBTYPE_ADMIN)

3-非持久代理预订的高水位标记 (MQSUBTYPE_PROXY)

标识: Mqiamo_sub_ndur_highwater

数据类型: MQCFIL。

已返回: 可用时。

SubCountNDurLowWater

描述: 时间间隔内非持久预订数的低水位标记。 这是按 SUBTYPE 建立索引的值的数组。

0-系统中所有非持久预订的低水位标记

1-非持久应用程序预订的低水位标记 (MQSUBTYPE_API)

2-非持久管理预订的低水位标记 (MQSUBTYPE_ADMIN)

3-非持久代理预订 (MQSUBTYPE_PROXY) 的低水位标记

标识: MQIAMO_SUB_NDUR_LOWWATER

数据类型: MQCFIL。

已返回: 可用时。

PutTopicCount

描述: 成功放入主题的持久和非持久消息数，使用 MQPUT1 调用放入的消息除外。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

注: 使用解析为主题的队列别名放入的消息包含在此值中。

标识: MQIAMO_TOPIC_PUTS。

数据类型: MQCFIL。

已返回: 可用时。

PutTopicFailCount

描述: 尝试将消息放入主题失败的次数。

标识: MQIAMO_TOPIC_PUTS_FAILED。

数据类型: MQCFIN。

已返回: 可用时。

Put1TopicCount

描述: 使用 MQPUT1 调用成功放入主题的持久和非持久消息数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

注: 使用解析为主题的队列别名放入的消息包含在此值中。

标识: MQIAMO_TOPIC_PUT1S.

数据类型: MQCFIL。

已返回: 可用时。

Put1TopicFailCount

描述: 尝试使用 MQPUT1 调用将消息放入主题失败的次数。

标识: MQIAMO_TOPIC_PUT1S_FAILED.

数据类型: MQCFIN。

已返回: 可用时。

PutTopicBytes

描述: 对解析为发布操作的持久和非持久消息使用 put 调用写入的字节数。 这是应用程序放入的字节数，而不是传递给订户的结果字节数，请参阅 PublishMsg 字节以获取此值。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_TOPIC_PUT_BYTES.

数据类型: MQCFIL64.

已返回: 可用时。

PublishMsgCount

描述: 在时间间隔内传递到预订的消息数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_PUBLISH_MSG_COUNT

数据类型: MQCFIL。

已返回: 可用时。

PublishMsgBytes

描述: 在时间间隔内传递到预订的字节数。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_PUBLISH_MSG_BYTES

数据类型: MQCFIL64.

已返回: 可用时。

队列统计信息消息数据

使用此页面来查看队列统计信息消息的结构

消息名称:	队列统计信息消息。
平台模式:	全部， IBM MQ for z/OS 除外。
系统队列:	SYSTEM.ADMIN.STATISTICS.QUEUE.

QueueManager

描述: 队列管理器的名称

标识: MQCA_Q_MGR_NAME

数据类型: MQCFST

最大长度: MQ_Q_MGR_NAME_LENGTH

已返回: 始终

IntervalStartDate

描述: 监视周期开始的日期

标识: MQCAMO_START_DATE

数据类型: MQCFST

最大长度: MQ_DATE_LENGTH

已返回: 始终

IntervalStartTime

描述: 监视周期开始时的时间

标识: MQCAMO_START_TIME

数据类型: MQCFST

最大长度: MQ_TIME_LENGTH

已返回: 始终

IntervalEndDate

描述: 监视时间段结束时的日期

标识: MQCAMO_END_DATE

数据类型: MQCFST

最大长度: MQ_DATE_LENGTH

已返回: 始终

IntervalEndTime

描述: 监控周期结束时的时间

标识: MQCAMO_END_TIME

数据类型: MQCFST

最大长度: MQ_TIME_LENGTH

已返回: 始终

CommandLevel

描述: 队列管理器命令级别

标识: MQIA_COMMAND_LEVEL

数据类型: MQCFIN

已返回: 始终

ObjectCount

描述: 在已记录统计信息数据的时间间隔内访问的队列对象数。 此值设置为消息中包含的 QStatisticsData PCF 组数。

标识: MQIAMO_OBJECT_COUNT

数据类型: MQCFIN

已返回: 始终

QStatisticsData

描述: 用于指定队列统计信息详细信息的分组参数

标识: MQGACF_Q_STATISTICS_DATA

数据类型: MQCFGR

组中的参数:

QName
CreateDate
CreateTime
QType
QDefinitionType
QMinDepth
QMaxDepth
AvgTimeOnQ
PutCount
PutFailCount
Put1Count
Put1FailCount
PutBytes
GetCount
GetFailCount
GetBytes
BrowseCount
BrowseFailCount
BrowseBytes
NonQueuedMsgCount
ExpiredMsgCount
PurgeCount

已返回: 始终

QName

描述: 队列的名称
标识: MQCA_Q_NAME
数据类型: MQCFST
最大长度: MQ_Q_NAME_LENGTH
已返回: 始终

CreateDate

描述: 创建队列的日期
标识: MQCA_CREATION_DATE
数据类型: MQCFST
最大长度: MQ_DATE_LENGTH
已返回: 始终

CreateTime

描述: 创建队列的时间
标识: MQCA_CREATION_TIME
数据类型: MQCFST
最大长度: MQ_TIME_LENGTH
已返回: 始终

QType

描述: 队列的类型

标识: MQIA_Q_TYPE

数据类型: MQCFIN

值: MQOT_LOCAL

已返回: 始终

QDefinitionType

描述: 队列定义类型

标识: MQIA_DEFINITION_TYPE

数据类型: MQCFIN

值: 可能的值包括:

- MQQDT_PREDEFINED
- MQQDT_PERMANENT_DYNAMIC
- MQQDT_TEMPORARY_DYNAMIC

已返回: 可用时

QMinDepth

描述: 监视时间段内的最小队列深度

标识: MQIAMO_Q_MIN_DEPTH

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

QMaxDepth

描述: 监视时间段内的最大队列深度

标识: MQIAMO_Q_MAX_DEPTH

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

AvgTimeOnQ

描述: 在监视时间段内以破坏性方式从队列中检索的消息的平均等待时间 (以微秒为单位)。 此参数是按持久性值建立索引的整数列表，请参阅 参考说明 2。

标识: MQIAMO64_AVG_Q_TIME

数据类型: MQCFIL64

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

PutCount

描述: 成功放入队列的持久和非持久消息数， MQPUT1 请求除外。 此参数是按持久性值建立索引的整数列表。 请参阅 参考注释 2。

标识: MQIAMO_PUTS

数据类型: MQCFIL

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

PutFailCount

描述: 尝试将消息放入队列失败的次数

标识: MQIAMO_PUTS_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

Put1Count

描述: 使用 MQPUT1 调用成功放入队列的持久和非持久消息数。 此参数是按持久性值建立索引的整数列表。 请参阅 参考注释 2。

标识: MQIAMO_PUT1S

数据类型: MQCFIL

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

Put1FailCount

描述: 尝试使用 MQPUT1 调用放入消息失败的次数

标识: MQIAMO_PUT1S_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

PutBytes

描述: 在放入队列的请求中写入的字节数

标识: MQIAMO64_PUT_BYTES

数据类型: MQCFIL64

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

GetCount

描述: 针对持久和非持久消息的成功破坏性获取请求数。 此参数是按持久性值建立索引的整数列表。 请参阅 参考注释 2。

标识: MQIAMO_GETS

数据类型: MQCFIL

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

GetFailCount

描述: 失败的破坏性获取请求数

标识: MQIAMO_GETS_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

GetBytes

描述: 在针对持久和非持久消息的破坏性放入请求中读取的字节数。 此参数是按持久性值建立索引的整数列表。 请参阅 参考注释 2。

标识: MQIAMO64_GET_BYTES

数据类型: MQCFIL64

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

BrowseCount

描述: 针对持久和非持久消息的成功非破坏性获取请求数。 此参数是按持久性值建立索引的整数列表。 请参阅 参考注释 2。

标识: MQIAMO_BROWSES

数据类型: MQCFIL

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

BrowseFailCount

描述: 不成功的非破坏性获取请求数

标识: MQIAMO_BROWSES_FAILED

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

BrowseBytes

描述: 在针对持久和非持久消息的非破坏性获取请求中读取的字节数。 此参数是按持久性值建立索引的整数列表。 请参阅 参考注释 2。

标识: MQIAMO64_BROWSE_BYTES

数据类型: MQCFIL64

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

NonQueuedMsgCount

描述: 绕过队列并直接传输到正在等待的应用程序的消息数。

只有在某些情况下才能绕过队列。 此数字表示 IBM MQ 能够绕过队列的次数，而不是应用程序等待的次数。

标识: MQIAMO_MSGS_NOT_QUEUED

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

ExpiredMsgCount

描述: 由于持久和非持久消息在检索之前已到期而废弃的持久和非持久消息数。

标识: MQIAMO_MSGS_EXPIRED

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

PurgeCount

描述: 清除的消息数。

标识: MQIAMO_MSGS_PURGED

数据类型: MQCFIN

包含在 PCF 组中: *QStatisticsData*

已返回: 可用时

通道统计信息消息数据

使用此页面来查看通道统计信息消息的结构

消息名称:	通道统计信息消息。
平台模式:	全部， IBM MQ for z/OS 除外。
系统队列:	SYSTEM.ADMIN.STATISTICS.QUEUE.

QueueManager

描述: 队列管理器的名称。

标识: MQCA_Q_MGR_NAME。

数据类型: MQCFST。

最大长度: MQ_Q_MGR_NAME_LENGTH。

已返回: 一直等在那里。

IntervalStartDate

描述: 监视时间段开始时的日期。

标识: MQCAMO_START_DATE。

数据类型: MQCFST。

最大长度: MQ_DATE_LENGTH。

已返回: 一直等在那里。

IntervalStartTime

描述: 监视时间段开始时的时间。

标识: MQCAMO_START_TIME。

数据类型: MQCFST。

最大长度: MQ_TIME_LENGTH。

已返回: 一直等在那里。

IntervalEndDate

描述: 监视时间段结束时的日期

标识: MQCAMO_END_DATE。

数据类型: MQCFST。

最大长度: MQ_DATE_LENGTH。

已返回: 一直等在那里。

IntervalEndTime

描述: 监控周期结束时的时间

标识: MQCAMO_END_TIME。

数据类型: MQCFST。

最大长度: MQ_TIME_LENGTH

已返回: 一直等在那里。

CommandLevel

描述: 队列管理器命令级别。

标识: MQIA_COMMAND_LEVEL。

数据类型: MQCFIN。

已返回: 一直等在那里。

ObjectCount

描述: 在已记录统计信息数据的时间间隔内访问的通道对象数。 此值设置为消息中包含的 ChlStatistics 数据 PCF 组数。

标识: MQIAMO_OBJECT_COUNT

数据类型: MQCFIN。

已返回: 一直等在那里。

ChlStatisticsData

描述: 用于指定通道的统计信息详细信息的分组参数。

标识: MQGACF_CHL_STATISTICS_DATA。

数据类型: MQCFGR。

组中的参数:

ChannelName
ChannelType
RemoteQmgr
ConnectionName
MsgCount
TotalBytes
NetTimeMin
NetTimeAvg
NetTimeMax
ExitTimeMin
ExitTimeAvg
ExitTimeMax
FullBatchCount
IncmplBatchCount
AverageBatchSize
PutRetryCount

已返回: 一直等在那里。

ChannelName

描述: 通道的名称。

标识: MQCACH_CHANNEL_NAME。

数据类型: MQCFST。

最大长度: MQ_CHANNEL_NAME_LENGTH。

已返回: 一直等在那里。

ChannelType

描述: 通道类型。

标识: MQIACH_CHANNEL_TYPE。

数据类型: MQCFIN。

值: 可能的值为:

MQCHT_SENDER
发送方通道。

MQCHT_SERVER
服务器通道。

MQCHT_RECEIVER
接收方通道。

MQCHT_REQUESTER
请求方通道

MQCHT_CLUSRCVR
集群接收方通道。

MQCHT_CLUSSDR
集群发送方通道。

已返回: 一直等在那里。

RemoteQmgr

描述: 远程队列管理器的名称。

标识: MQCA_REMOTE_Q_MGR_NAME。

数据类型: MQCFST。

最大长度: MQ_Q_MGR_NAME_LENGTH

已返回: 可用时。

ConnectionName

描述: 远程队列管理器的连接名称。

标识: MQCACH_CONNECTION_NAME。

数据类型: MQCFST

最大长度: MQ_CONN_NAME_LENGTH

已返回: 可用时。

MsgCount

描述: 发送或接收的持久和非持久消息数。

标识: MQIAMO_MSGS。

数据类型: MQCFIN

已返回: 可用时。

TotalBytes

描述: 针对持久和非持久消息发送或接收的字节数。

标识: MQIAMO64_BYTES.

数据类型: MQCFIN64.

已返回: 可用时。

NetTimeMin

描述: 在记录时间间隔内测量的最短记录通道往返时间 (以微秒为单位)。

标识: MQIAMO_NET_TIME_MIN。

数据类型: MQCFIN。

已返回: 可用时。

NetTimeAvg

描述: 在记录时间间隔内测量的平均记录通道往返时间 (以微秒为单位)。

标识: MQIAMO_NET_TIME_AVG。

数据类型: MQCFIN。

已返回: 可用时。

NetTimeMax

描述: 在记录时间间隔内测量的最长记录通道往返时间 (以微秒为单位)。

标识: MQIAMO_NET_TIME_MAX。

数据类型: MQCFIN。

已返回: 可用时。

ExitTimeMin

描述: 在记录时间间隔内执行用户出口所耗用的最短记录时间 (以微秒为单位) ,

标识: MQIAMO_EXIT_TIME_MIN。

数据类型: MQCFIN。

已返回: 可用时。

ExitTimeAvg

描述: 在记录时间间隔内执行用户出口所耗用的平均记录时间 (以微秒为单位)。 以微秒为单位进行测量。

标识: MQIAMO_EXIT_TIME_AVG。

数据类型: MQCFIN。

已返回: 可用时。

ExitTimeMax

描述: 在记录时间间隔内执行用户出口所耗用的最长记录时间 (以微秒为单位)。 以微秒为单位进行测量。

标识: MQIAMO_EXIT_TIME_MAX。

数据类型: MQCFIN。

已返回: 可用时。

FullBatchCount

描述: 由于达到通道属性 BATCHSZ 或 BATCHLIM 的值而发送的通道处理的批处理数。

标识: MQIAMO_FULL_批处理。

数据类型: MQCFIN。

已返回: 可用时。

IncmplBatchCount

描述: 通道处理的未达到通道属性 BATCHSZ 或 BATCHLIM 的值而发送的批处理数。

标识: MQIAMO_INCOMPLETE_批处理。

数据类型: MQCFIN。

已返回: 可用时。

AverageBatchSize

描述: 通道处理的批处理的平均批处理大小。

标识: MQIAMO_AVG_BATCH_SIZE。

数据类型: MQCFIN。

已返回: 可用时。

PutRetryCount

描述: 时间间隔内未能放入消息并进入重试循环的次数。

标识: MQIAMO_PUT_RETRIES。

数据类型: MQCFIN。

已返回: 可用时。

参考注释

使用此页面来查看记帐和统计信息消息结构的描述所引用的注释

以下消息数据描述引用了这些说明:

- 第 137 页的『MQI 记帐消息数据』
- 第 147 页的『队列记帐消息数据』
- 第 158 页的『MQI 统计信息消息数据』
- 第 168 页的『队列统计信息消息数据』
- 第 174 页的『通道统计信息消息数据』

1. 此参数与 IBM MQ 对象相关。 此参数是由以下常量建立索引的值数组 (MQCFIL 或 MQCFIL64):

表 *24:* 按对象类型建立索引的数组	
对象类型	**值上下文**
MQOT_Q (1)	包含与队列对象相关的值。
MQOT_NAMELIST (2)	包含与名称列表对象相关的值。
MQOT_PROCESS (3)	包含与流程对象相关的值。
MQOT_Q_MGR (5)	包含与队列管理器对象相关的值。
MQOT_CHANNEL (6)	包含与通道对象相关的值。
MQOT_AUTH_INFO (7)	包含与认证信息对象相关的值。
MQOT_TOPIC (8)	包含与主题对象相关的值。

注: 返回由 13 个 MQCFIL 或 MQCFIL64 值组成的数组，但仅列出的值有意义。

2. 此参数与 IBM MQ 消息相关。 此参数是由以下常量建立索引的值数组 (MQCFIL 或 MQCFIL64):

表 *25:* 按持久性值建立索引的数组	
常量	**值**
1	包含非持久消息的值。
2	包含持久消息的值。

注: 其中每个数组的索引从零开始，因此 1 的索引引用数组的第二行。 这些表中未列出的这些数组的元素不包含记帐或统计信息。

应用程序活动跟踪

应用程序活动跟踪生成有关连接到队列管理器的应用程序的行为的详细信息。 它跟踪应用程序的行为，并提供应用程序在与 IBM MQ 资源交互时使用的参数的详细视图。 它还显示了应用程序发出的 MQI 调用的序列。

如果需要的信息多于事件监视，消息监视，记帐和统计信息消息以及实时监视提供的信息，请使用应用程序活动跟踪。

注: 将在每个应用程序的 IBM MQ 连接上生成活动跟踪; 因此，如果启用了记帐消息，那么活动跟踪操作将计入每个应用程序的 MQI 记帐信息。

IBM MQ 支持两种收集应用程序活动跟踪数据的方法。

- 集中收集应用程序活动跟踪信息，其中通过将活动跟踪 PCF 消息写入系统队列 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE。
- 预订活动跟踪数据，已写入特殊 IBM MQ 系统主题。

请注意， IBM MQ for z/OS 不支持活动跟踪。

ALW 现在，您可以在 IBM MQ 支持的大多数编程语言上指定应用程序名称，请参阅 以受支持的编程语言指定应用程序名称 以获取更多信息。

配置应用程序活动跟踪信息的集中收集

应用程序活动跟踪消息是 PCF 消息。 您可以使用配置文件来配置活动跟踪。 要配置应用程序活动跟踪信息的集中收集，请设置 ACTVTRC 队列管理器属性。 您可以使用 MQCONNX 选项在连接级别覆盖此设置，也可以使用活动跟踪配置文件在应用程序节级别覆盖此设置。

关于此任务

活动跟踪消息由 MQMD 结构组成 :PCF (MQCFH) 头结构，后跟多个 PCF 参数。 ApplicationTrace 数据 PCF 组的序列遵循 PCF 参数。 这些 PCF 组收集有关应用程序在连接到队列管理器时执行的 MQI 操作的信息。 您可以使用名为 `mqat.ini` 的配置文件来配置活动跟踪。

要控制是否收集应用程序活动跟踪信息，请配置以下一个或多个设置:

1. ACTVTRC 队列管理器属性。
2. ACTVCONO 设置 (在 MQCONNX 中传递的 MQCNO 结构中)。
3. 活动跟踪配置文件 `mqat.ini` 中应用程序的匹配节。

前一个序列很重要。 ACTVTRC 属性被 ACTVCONO 设置覆盖，这些设置被 `mqat.ini` 文件中的设置覆盖。

除非另有说明，否则将在每个操作完成后写入跟踪条目。 这些条目首先写入系统队列 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE，然后在应用程序与队列管理器断开连接时写入应用程序活动跟踪消息。 对于长时间运行的应用程序，如果发生以下任何事件，那么将写入中间消息:

- 连接的生存期达到定义的超时值。
- 操作数达到指定的数目。
- 内存中收集的数据量达到队列允许的最大消息长度。

使用 **ActivityInterval** 参数设置超时值。 使用 **ActivityCount** 参数设置操作数。 这两个参数都在活动跟踪配置文件 `mqat.ini` 中指定。

启用应用程序活动跟踪可能会影响性能。 可以通过调整 **ActivityCount** 和 **ActivityInterval** 设置来减少开销。 请参阅第 186 页的『调整应用程序活动跟踪的性能影响』。

查看应用程序活动跟踪消息内容的最简单方法是使用 第 186 页的『amqsact 样本程序』。

过程

1. 第 180 页的『设置 ACTVTRC 以控制活动跟踪信息的收集』.
2. 第 181 页的『设置 MQCONNX 选项以控制活动跟踪信息的收集』.
3. 第 181 页的『使用 mqat.ini 配置活动跟踪行为』.
4. 第 186 页的『调整应用程序活动跟踪的性能影响』.

设置 *ACTVTRC* 以控制活动跟踪信息的收集

使用队列管理器属性 ACTVTRC 来控制 MQI 应用程序活动跟踪信息的收集

关于此任务

仅针对在启用应用程序活动跟踪之后开始的连接生成应用程序活动跟踪消息。 **ACTVTRC** 参数可以具有以下值:

ON
已启用 API 活动跟踪收集

OFF
已禁用 API 活动跟踪收集

注: ACTVTRC 设置可由队列管理器 **ACTVCONO** 参数覆盖。 如果将 **ACTVCONO** 参数设置为 ENABLED，那么可以使用 MQCNO 结构中的 **Options** 字段来覆盖给定连接的 **ACTVTRC** 设置。 请参阅第 181 页的『设置 MQCONNX 选项以控制活动跟踪信息的收集』。

示例

要更改 **ACTVTRC** 参数的值，请使用 MQSC 命令 ALTER QMGR。 例如，要启用 MQI 应用程序活动跟踪信息收集，请使用以下 MQSC 命令:

```
ALTER QMGR ACTVTRC(ON)
```

下一步做什么

查看应用程序活动跟踪消息内容的最简单方法是使用第 186 页的『amqsact 样本程序』。

启用应用程序活动跟踪可能会影响性能。 可以通过调整 **ActivityCount** 和 **ActivityInterval** 设置来减少开销。 请参阅第 186 页的『调整应用程序活动跟踪的性能影响』。

设置 *MQCONNX* 选项以控制活动跟踪信息的收集

如果队列管理器属性 **ACTVCONO** 设置为 ENABLED，那么您可以在 MQCONNX 调用上使用 **ConnectOpts** 参数来逐个连接地启用或禁用应用程序活动报告。 这些选项覆盖队列管理器属性 **ACTVTRC** 定义的活动跟踪行为，并且可以被活动跟踪配置文件 mqat.ini 中的设置覆盖。

过程

1. 将队列管理器属性 **ACTVCONO** 设置为 ENABLED。

   注: 如果应用程序尝试使用 **ConnectOpts** 参数修改应用程序的记帐行为，并且 QMGR 属性 **ACTVCONO** 设置为 DISABLED，那么不会向应用程序返回任何错误，并且活动跟踪集合由队列管理器属性或活动跟踪配置文件 mqat.ini 定义。
2. 将 MQCONNX 调用上的 **ConnectOpts** 参数设置为 MQCNO_ACTIVITY_TRACE_ENABLED。

   MQCONNX 调用上的 **ConnectOpts** 参数可以具有以下值:

   MQCNO_ACTIVITY_TRACE_DISABLED
   已针对连接禁用活动跟踪。

   MQCNO_ACTIVITY_TRACE_ENABLED
   为连接启用了活动跟踪。

   注: 如果应用程序针对 MQCONNX 同时选择 MQCNO_ACTIVITY_TRACE_ENABLED 和 MQCNO_ACTIVITY_TRACE_DISABLED，那么调用将失败，原因码为 MQRC_OPTIONS_ERROR。
3. 检查这些活动跟踪设置是否未被活动跟踪配置文件 mqat.ini 中的设置覆盖。

   请参阅第 181 页的『使用 mqat.ini 配置活动跟踪行为』。

下一步做什么

查看应用程序活动跟踪消息内容的最简单方法是使用第 186 页的『amqsact 样本程序』。

启用应用程序活动跟踪可能会影响性能。 可以通过调整 **ActivityCount** 和 **ActivityInterval** 设置来减少开销。 请参阅第 186 页的『调整应用程序活动跟踪的性能影响』。

使用 *mqat.ini* 配置活动跟踪行为

活动跟踪行为是使用名为 mqat.ini 的配置文件配置的。 此文件用于定义报告活动跟踪数据的级别和频率。 该文件还提供了一种方法来定义规则，以根据应用程序的名称来启用和禁用活动跟踪。

关于此任务

Linux AIX 在 AIX and Linux 系统上， `mqat.ini` 位于队列管理器数据目录中，该目录与 `qm.ini` 文件位于同一位置。

Windows 在 Windows 系统上， `mqat.ini` 位于队列管理器数据目录 `C:\Program Files\IBM\WebSphere MQ\qmgrs\queue_manager_name` 中。 运行要跟踪的应用程序的用户需要具有读取此文件的许可权。

注: 从 IBM WebSphere MQ 7.1 或更低版本迁移的队列管理器将缺少 `mqat.ini` 文件。 在这种情况下，需要手动创建 `mqat.ini` 文件，并且需要对该文件设置 660 个许可权。

修改 `mqat.ini` 文件时，将根据修改后的版本处理新创建的 IBM MQ 连接。 除非更改了队列管理器参数 (例如，遵循 ALTER QMGR 命令)，否则现有连接将继续使用先前版本。

此文件遵循与 `mqs.ini` 和 `qm.ini` 文件相同的节键/参数/值对格式。

该文件由单个节 **AllActivityTrace** 组成，用于配置缺省情况下针对所有活动跟踪报告活动跟踪数据的级别和频率。

该文件还可以包含多个 **ApplicationTrace** 节。 其中每个连接都根据与规则的连接的应用程序名称匹配，定义一个或多个连接的跟踪行为规则。

AllActivity 跟踪节

单个 **AllActivityTrace** 节定义应用于所有 IBM MQ 连接的活动跟踪的设置，除非被覆盖。

AllActivityTrace 节中的各个值可以被 **ApplicationTrace 节** 节中的更具体的信息覆盖。

如果指定了多个 **AllActivityTrace** 节，那么将使用最后一节中的值。 所选 **AllActivity 跟踪** 中缺少的参数采用缺省值。 将忽略先前 **AllActivity 跟踪** 节中的参数和值。

可以在 **AllActivityTrace** 节下指定以下参数:

表 *26:* 可在活动跟踪配置文件中使用的参数/值对

名称	**值 (缺省为粗体类型)**	**描述**
ActivityInterval	0-99999999 (**1**)	跟踪消息之间的近似时间间隔 (以秒为单位)。 连接在该时间间隔内执行的所有活动都将写入一条消息。 如果此值为 0，那么将在连接断开时 (或达到活动计数时) 写入跟踪消息。
ActivityCount	0-99999999 (**100**)	跟踪消息之间的 MQI 或 XA 操作数。 如果此值为 0，那么将在连接断开时 (或活动时间间隔已过去时) 写入跟踪消息。
TraceLevel	LOW/ **MEDIUM** /HIGH	针对每个操作跟踪的参数详细信息量。 各个操作的描述详细说明了针对每个跟踪级别包含哪些参数。
TraceMessage 数据	**0** -104 857 600 (最大 100 MB)	针对 MQGET， MQPUT， MQPUT1 和回调操作跟踪的消息数据量 (以字节计)
StopOnGetTrace 消息	**ON** /OFF	由于可能发生循环，因此建议不要使用活动跟踪来跟踪也在处理活动跟踪消息的应用程序。
SubscriptionDelivery	**批处理** /立即执行	确定当存在一个或多个活动跟踪预订时是否使用 ActivityInterval 和 ActivityCount 参数。 如果将此参数设置为 IMMEDIATE，那么当跟踪数据具有匹配的预订时， ActivityInterval 和 ActivityCount 值将被有效值 1 覆盖。 不会将每个活动跟踪记录与来自相同连接的其他记录一起批处理，而是立即交付到预订而不会延迟。

ApplicationTrace 节

ApplicationTrace 节包含一条规则，用于定义将根据应用程序名称跟踪或不跟踪哪些 IBM MQ 连接。 (可选) 在覆盖全局跟踪级别和频率设置的 Allsettings 下定义的缺省行为。

此节可以包含 ApplName， ApplFunction 和 ApplClass 参数，这些参数根据 "连接匹配规则" 中定义的匹配规则来使用，以确定此节是否适用于特定连接。

该节必须包含跟踪参数，以确定此规则是打开还是关闭活动跟踪以用于匹配的连接。

可以使用 off 规则来显式禁用对更特定的应用程序名称的跟踪，并覆盖队列管理器或活动跟踪连接选项的 ACTVTRC 设置。

可以在 **ApplicationTrace** 节下指定以下参数:

表 *27:* 可在应用程序跟踪配置文件中使用的参数/值对

名称	**值 (缺省为粗体类型)**	**描述**
跟踪	ON/OFF (必需参数-无缺省值)	活动跟踪开关。 可以在特定于应用程序的节中使用此开关来确定活动跟踪对于当前应用程序节的作用域是否处于活动状态。 请注意，此值将覆盖队列管理器的 ACTVTRC 和 ACTVCONO 设置。
ApplName	字符串 (必需参数-无缺省值)	此值用于确定 ApplicationTrace 节应用于哪些应用程序。 它与 API 出口上下文结构 (相当于 MQMD.PutApplName) 中的 ApplName 值匹配。 ApplName 值的内容因应用程序环境而异。 对于 z/OS 以外的平台，仅包含 MQAXC.ApplName 与节中的值匹配。 进行比较时，将忽略最右边路径分隔符左边的字符。 z/OS 对于 z/OS 应用程序，整个 MQAXC.ApplName 与节中的值匹配。 可以在 ApplName 值的末尾使用单个通配符 (*) 来匹配该点之后的任意数目的字符。 如果 ApplName 值设置为单个通配符 (*)，那么 ApplName 值将与所有应用程序匹配。
IBM i IBM i ApplFunction	字符串 (缺省值 *****)	此值用于限定 **ApplicationTrace** 节和 **ApplName** 值应用于的应用程序。 此节是可选的，但仅对 IBM i 队列管理器有效。 可以在 **ApplName** 值的末尾使用单个通配符 (*) 来匹配任意数量的字符。 例如，指定 **ApplName** = * 和 **ApplFunction** = *AMQSPUT0* 的 **ApplicationTrace** 节适用于来自任何作业的 AMQSPUT0 程序的所有调用。
ApplClass	用户/MCA/ **全部**	应用程序的类。 请参阅下表，以获取有关 **AppType** 值如何对应于 IBM MQ 连接的说明。

下表显示 *AppClass* 值如何对应于连接 API 出口上下文结构中的 **APICallerType** 和 **APIEnvironment** 字段。

表 *28: Appclass* 值及其与 *APICallerType* 和 *APIEnvironment* 字段的对应方式

APPLCLASS	**API 调用者类型:**	**API 环境:**	**描述**
用户	MQXACT_EXTERNAL	MQXE_OTHER	仅跟踪用户应用程序

表 28: *Appclass* 值及其与 *APICallerType* 和 *APIEnvironment* 字段的对应方式 (继续)			
APPLCLASS	**API 调用者类型:**	**API 环境:**	**描述**
MCA	(任何值)	MQXE_MCA MQXE_MCA_CLNTCONN MQXE_MCA_SVRCONN	客户机和通道 (amqrmppa)
所有	(任何值)	(任何值)	跟踪所有连接

注意: 必须将 *MCA* 的 **APPLCLASS** 用于客户机用户应用程序，因为 *USER* 的类与这些类不匹配。

例如，要跟踪 **amqsputc** 样本应用程序，可以使用以下代码:

```
ApplicationTrace:
ApplClass=MCA                           # Application type
                                        #   Values:  (USER | MCA | INTERNAL | ALL)
                                        #   Default: USER
ApplName=amqsputc     # Application name (may be wildcarded)
                                        #   (matched to app name without path)
                                        #   Default: *
Trace=ON                                # Activity trace switch for application
                                        #   Values:  ( ON | OFF )
                                        #   Default: OFF
ActivityInterval=30                      # Time interval between trace messages
                                        #   Values: 0-99999999 (0=off)
                                        #   Default: 0
ActivityCount=1                         # Number of operations between trace msgs
                                        #   Values: 0-99999999  (0=off)
                                        #   Default: 0
TraceLevel=MEDIUM                       # Amount of data traced for each operation
                                        #   Values: LOW | MEDIUM | HIGH
                                        #   Default: MEDIUM
TraceMessageData=1000                   # Amount of message data traced
                                        #   Values: 0-100000000
                                        #   Default: 0
```

创建队列管理器时生成的缺省 `mqat.ini` 包含用于显式禁用所提供活动跟踪样本 **amqsact** 的活动跟踪的单个规则。

连接匹配规则

队列管理器将应用以下规则来确定要用于连接的节设置。

1. **AllActivityTrace** 节中指定的值用于连接，除非该值也出现在 **ApplicationTrace** 节中，并且该节满足点 2， 3 和 4 中描述的连接的匹配条件。
2. **ApplClass** 与 IBM MQ 连接的类型匹配。 如果 **ApplClass** 与连接类型不匹配，那么将忽略此连接的节。
3. 节中的 *ApplName* 值与连接的 API 出口上下文结构 (MQAXC) 中 **ApplName** 字段的文件名部分相匹配。

   文件名部分派生自最终路径分隔符 (/或 \) 右边的字符。 如果节 **ApplName** 包含通配符 (*)，那么仅将通配符左侧的字符与连接的 **ApplName** 中的等效字符数进行比较。

   例如，如果指定节值 "FRE*"，那么在比较中仅使用前三个字符，因此 "path/FREEDOM" 和 "path\FREDDY" 匹配，但 "path/FRIEND" 不匹配。 如果节的 *ApplName* 值与连接 **ApplName** 不匹配，那么将忽略此连接的节。
4. 如果多个节与连接的 **ApplName** 和 **ApplClass** 匹配，那么将使用具有最特定 **ApplName** 的节。

   最具体的 **ApplName** 定义为使用最多字符来匹配连接的 **ApplName** 的。

   例如，如果 `ini` 文件包含具有 **ApplName** = *"FRE*"* 的节以及具有 **ApplName** = *"FREE*"* 的另一节，那么将选择具有 **ApplName** = *"FREE*"* 的节作为具有 **ApplName** = *"path/FREEDOM"* 的连接的最佳匹配项，因为它与四个字符匹配 (而 **ApplName** = *"FRE*"* 仅与三个字符匹配)。
5. 如果在点 2， 3 和 4 中应用规则后，存在多个与连接 **ApplName** 和 **ApplClass** 的连接相匹配的节，那么将使用上次匹配的值，并且将忽略所有其他节。

覆盖每个规则的缺省设置

(可选) 对于与 **ApplicationTrace** 节匹配的那些连接，可以覆盖 **AllActivityTrace** 节下的全局跟踪级别和频率设置。

可以在 **ApplicationTrace** 节下设置以下参数。 如果未设置这些值，那么将从 **AllActivityTrace** 节设置继承该值:

- **ActivityInterval**
- **ActivityCount**
- **TraceLevel**
- **TraceMessageData**
- **StopOnTraceMsg**

mqat.ini 语法

mqat.ini 文件格式的语法规则为:

- 以散列或分号开头的文本被视为延伸到行尾的注释。
- 第一个重要 (非注释) 行必须是节键。
- 节键由节的名称后跟冒号组成。
- "参数/值" 对由后跟等号的参数的名称以及随后的值组成。
- 只能在一行上显示单个 "参数/值" 对。 (参数值不得回绕到另一行)。
- 将忽略前导和尾部空格。 节名称，参数名称和值或参数/值对之间的空格数量没有限制。 换行符很重要，不可忽略
- 任何行的最大长度为 2048 个字符
- 节键，参数名称和常量参数值不区分大小写，但变量参数值 (*ApplName* 和 *DebugPath*) 区分大小写。

应用程序活动跟踪文件示例

以下示例显示如何在 Activity Trace ini 文件中指定配置数据。

```
AllActivityTrace:
ActivityInterval=1
ActivityCount=100
TraceLevel=MEDIUM
TraceMessageData=0
StopOnGetTraceMsg=ON

ApplicationTrace:
ApplName=amqs*
Trace=ON
TraceLevel=HIGH
TraceMessageData=1000

ApplicationTrace:
ApplName=amqsact*
Trace=OFF
```

上述 **AllActivityTrace** 节定义缺省情况下，通过 ApplicationTrace 规则或通过队列管理器 ACTVTRC 属性或通过应用程序以编程方式启用活动跟踪时将如何执行活动跟踪。

第一个 **ApplicationTrace** 节定义一个规则，该规则将导致跟踪名称以 "amqs" 开头的应用程序的任何 MQI 活动。 为这些应用程序生成的跟踪将非常详细，最多包含 1000 字节的消息数据。 将继承活动时间间隔和计数参数

第二个 **ApplicationTrace** 节定义对名称以 "amqsact" (活动跟踪样本) 开头的应用程序关闭跟踪的规则。 此规则将覆盖 amqsact 应用程序的先前 "on" 规则，从而不会对该应用程序进行跟踪。

在 C 样本目录 (与 amqsact.c 文件相同的目录) 中，还提供了一个示例作为名为 mqat.ini 的样本。 对于已从较早发行版 IBM MQ 迁移的队列管理器，可以将此文件复制到队列管理器数据目录。

下一步做什么

启用应用程序活动跟踪可能会影响性能。 可以通过调整 **ActivityCount** 和 **ActivityInterval** 设置来减少开销。 请参阅第 186 页的『调整应用程序活动跟踪的性能影响』。

调整应用程序活动跟踪的性能影响

启用应用程序活动跟踪可能会导致性能下降。 通过仅跟踪您需要的应用程序，增加排入队列的应用程序数量，以及调整 mqat.ini 中的 **ActivityInterval**， **ActivityCount** 和 **TraceLevel**，可以减少此情况。

关于此任务

选择性地对应用程序或所有队列管理器应用程序启用应用程序活动跟踪可能会导致额外的消息传递活动，并且在队列管理器中需要额外的存储空间。 在消息传递性能至关重要的环境中，例如，在高工作负载应用程序中，或者服务级别协议 (SLA) 需要来自消息传递提供程序的最短响应时间时，可能不适合收集应用程序活动跟踪，或者可能需要调整生成的跟踪活动消息的详细信息或频率。 mqat.ini 文件中的 **ActivityInterval**， **ActivityCount** 和 **TraceLevel** 的预设值提供了缺省的详细信息和性能平衡。 但是，您可以调整这些值以满足系统的精确功能和性能要求。

过程

- 仅跟踪所需的应用程序。

  要执行此操作，请在 mqat.ini 中创建特定于应用程序的 ApplicationTrace 节，或者更改应用程序以在 MQCONNX 调用的 **MQCNO** 结构的选项字段中指定 MQCNO_ACTIVITY_TRACE_ENABLED。 请参阅第 181 页的『使用 mqat.ini 配置活动跟踪行为』和第 181 页的『设置 MQCONNX 选项以控制活动跟踪信息的收集』。
- 在启动跟踪之前，请检查是否至少有一个应用程序正在运行，并准备好从 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE 检索活动跟踪消息数据。
- 通过增加排入队列的应用程序数，使队列深度尽可能低。
- 在 mqat.ini 文件中设置 **TraceLevel** 值以收集所需的最小数据量。

  TraceLevel=LOW 对消息传递性能的影响最小。 请参阅第 181 页的『使用 mqat.ini 配置活动跟踪行为』。
- 调整 mqat.ini 中的 **ActivityCount** 和 **ActivityInterval** 值，以调整生成活动跟踪消息的频率。

  如果要跟踪多个应用程序，那么生成活动跟踪消息的速度可能比从 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE 中除去这些消息的速度更快。 但是，当您降低生成活动跟踪消息的频率时，也会增加队列管理器所需的存储空间以及将消息写入队列时的消息大小。

下一步做什么

查看应用程序活动跟踪消息内容的最简单方法是使用 第 186 页的『amqsact 样本程序』。

amqsact 样本程序

amqsact 为您格式化应用程序活动跟踪消息，并随 IBM MQ 一起提供。

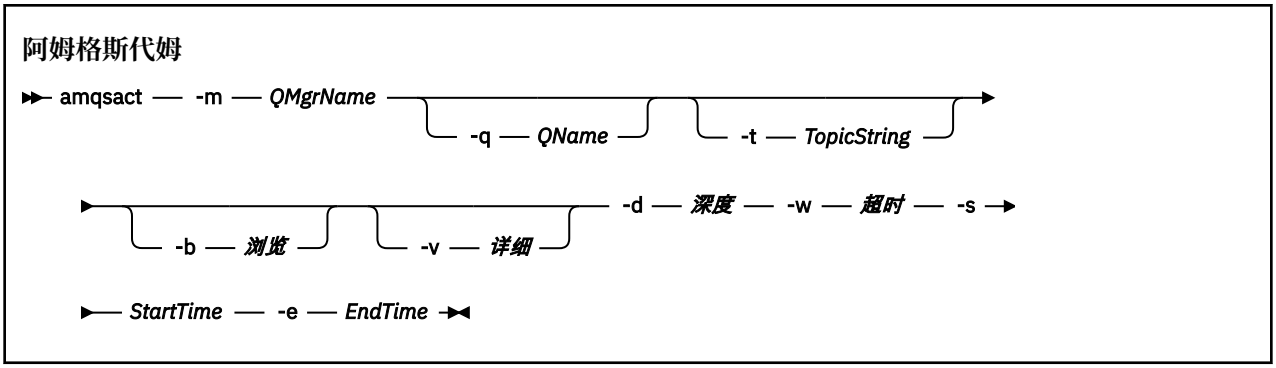
编译的程序位于样本目录中:

- Linux AIX 在 AIX and Linux MQ_INSTALLATION_PATH/samp/bin 上
- Windows 在 Windows MQ_INSTALLATION_PATH\tools\c\Samples\Bin 上

显示方式

缺省情况下，处于显示方式的 **amqsact** 在 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE。 您可以通过指定队列名称或主题字符串来覆盖此行为。

您还可以控制显示的跟踪周期，并指定在显示后是除去还是保留活动跟踪消息。



显示方式的必需参数

-m *QMgrName*
队列管理器的名称。

-d 深度
要显示的记录数。

-w *timeout*
等待的时间 (以秒计)。 如果指定时间段内未显示任何跟踪消息，那么 **amqsact** 将退出。

-s *StartTime*
要处理的记录的开始时间。

-e *EndTime*
要处理的记录的结束时间。

显示方式的可选参数

-q *QName*
指定特定队列以覆盖缺省队列名称

-t *TopicString*
预订事件主题

-b
仅浏览记录

-v
Verbose 输出

显示方式的示例输出

在 MQCONN API 调用上使用具有详细输出的队列管理器 *TESTQM* 上的 **amqsact** :

```
amqsact -m TESTQM -v
```

前面的命令给出了以下示例输出:

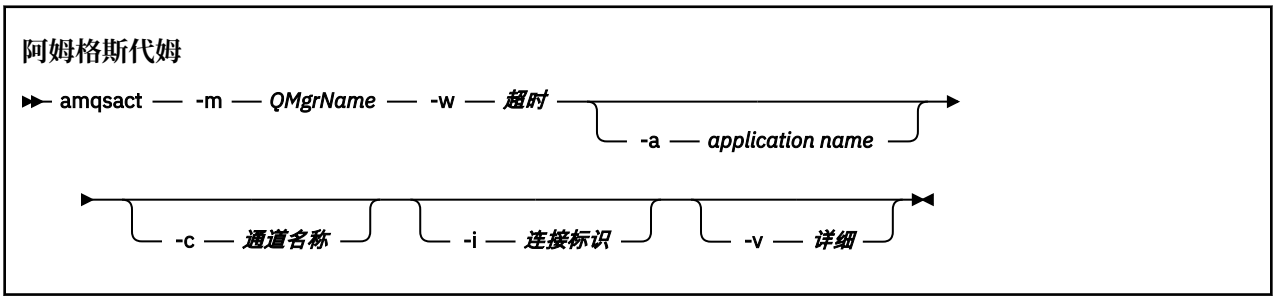
```
MonitoringType: MQI Activity Trace
Correl_id:
00000000: 414D 5143 5445 5354 514D 2020 2020 2020 'AMQCTESTQM   '
00000010: B5F6 4251 2000 E601           '       '
QueueManager: 'TESTQM'
Host Name: 'ADMINIB-1VTJ6N1'
IntervalStartDate: '2014-03-15'
IntervalStartTime: '12:08:10'
IntervalEndDate: '2014-03-15'
IntervalEndTime: '12:08:10'
CommandLevel: 750
SeqNumber: 0
ApplicationName: 'IBM MQ_1\bin\amqsput.exe'
Application Type: MQAT_WINDOWS_7
ApplicationPid: 14076
UserId: 'Emma_Bushby'
API Caller Type: MQXACT_EXTERNAL
```

```
API Environment: MQXE_OTHER
Application Function: ''
Appl Function Type: MQFUN_TYPE_UNKNOWN
Trace Detail Level: 2
Trace Data Length: 0
Pointer size: 4
Platform: MQPL_WINDOWS_7
MQI Operation: 0
Operation Id: MQXF_CONN
ApplicationTid: 1
OperationDate: '2014-03-15'
OperationTime: '12:08:10'
ConnectionId:
00000000: 414D 5143 5445 5354 514D 2020 2020 2020 'AMQCTESTQM   '
00000010: FFFFFFB5FFFFFFF6 4251 2000 FFFFFFE601  '        '
QueueManager: 'TESTQM'
Completion Code: MQCC_OK
Reason Code: 0
```

动态方式

您可以通过指定应用程序名称，通道名称或连接标识作为 **amqsact** 的自变量来启用动态方式。 请注意，可以在名称中使用通配符。

在动态方式下，通过使用对系统主题的非持久预订，在样本开始时启用活动跟踪数据。 当 **amqsact** 停止时，收集活动跟踪数据将停止。 必须在动态方式下为 **amqsact** 指定超时。 您可以同时运行 **amqsact** 的多个副本，每个实例接收任何活动跟踪数据的副本。



动态方式的必需参数

-m *QMgrName*
队列管理器的名称。

-w *timeout*
等待的时间 (以秒计)。 如果指定时间段内未显示任何跟踪消息，那么 **amqsact** 将退出。

动态方式的可选参数

-a 应用程序名称
指定要收集消息的应用程序名称

-c 通道名称
指定要为其收集消息的通道

-i 连接标识
指定要为其收集消息的连接。

-v
Verbose 输出

动态方式的示例输出

以下命令将生成并显示以文本 “amqs”开头的应用程序所建立的任何连接的活动跟踪消息。 在不活动 30 秒后， **amqsact** 程序结束，并且不会生成新的活动跟踪数据。

```
amqsactc -m QMGR1 -w 30 -a amqs*
```

以下命令生成并显示 QMGR1.TO.QMGR2 通道。 在处于不活动状态 10 秒后， **amqsact** 程序结束，并且不会生成新的活动跟踪数据。

```
amqsactc -m QMGR1 -w 10 -c QMGR1.TO.QMGR2
```

以下命令生成并显示现有 IBM MQ 连接上具有 CONN "6B576B5420000701" 和 EXTCONN "414D5143514D47523120202020202020" 的任何活动的详细活动跟踪消息。 在不活动一分钟后，**amqsact** 程序结束，并且不会生成新的活动跟踪数据。

```
amqsactc -m QMGR1 -w 60 -i 414D5143514D4752312020202020206B576B5420000701 -v
```

预订应用程序活动跟踪信息

您可以动态预订应用程序活动跟踪信息，作为通过队列管理器级别配置收集信息的替代方法。

关于此任务

应用程序活动跟踪应用程序的行为，并提供应用程序在与 IBM MQ 资源交互时使用的参数的详细视图。 它还显示了应用程序发出的 MQI 调用的序列。

除了将跟踪数据写入系统队列之外， 产品还提供了动态预订活动跟踪数据 (写入特殊 IBM MQ 系统主题) 的能力，而不是通过队列管理器级别配置收集信息。

请注意，从 IBM MQ 9.0 开始，产品不会将出口用于此目的。 如果先前已使用出口来跟踪应用程序活动，那么必须切换到使用替代方法来收集应用程序活动跟踪。

创建预订将启用活动跟踪。 您不必像集中收集跟踪数据一样设置队列管理器或应用程序属性。 但是，通过在队列管理器或应用程序级别禁用跟踪来显式阻止活动跟踪，也会阻止将活动跟踪传递到任何匹配的预订。

过程

- 第 189 页的『应用程序活动跟踪的预订』
- 第 190 页的『创建应用程序活动跟踪的预订』
- 第 191 页的『使用 amqsact 来查看跟踪消息』
- 第 193 页的『使用 mqat.ini 配置跟踪级别』

应用程序活动跟踪的预订

您可以预订 IBM MQ 系统主题以收集应用程序活动跟踪信息。

预订表示要跟踪的活动的特殊 IBM MQ 系统主题字符串。 预订将自动生成活动跟踪数据消息并将其发布到预订目标队列。 如果删除预订，那么将停止生成该预订的活动跟踪数据。

预订可以跟踪下列其中一个资源上的活动:

- 指定的应用程序
- 指定的 IBM MQ 通道
- 现有 IBM MQ 连接

您可以创建具有不同主题字符串或相同主题字符串的多个预订。 如果您创建多个具有相同系统活动跟踪主题字符串的预订，那么每个预订都会接收活动跟踪数据的副本，这可能会产生不利的性能影响。

启用任何级别的活动跟踪可能会产生不利的性能影响。 预订越多或预订的资源越多，潜在的性能开销就越大。 为了最大限度减少收集活动跟踪的开销，数据将写入消息并从应用程序活动本身异步传递到预订。 通常，多个操作会写入单个活动跟踪数据消息。 异步操作可以在应用程序操作与接收记录该操作的跟踪数据之间引入延迟。

创建应用程序活动跟踪的预订

您可以创建特定主题的预订，以收集应用程序活动跟踪数据。

针对特定系统主题字符串创建预订时，将自动向该预订发布相应的活动跟踪 PCF 数据消息。 有关预订主题的详细信息，请参阅 发布/预订消息传递。

主题字符串具有以下格式:

```
$SYS/MQ/INFO/QMGR/qmgr_name/ActivityTrace/resource_type/resource_identifier
```

其中:

- *qmgr_name* 指定被跟踪应用程序所连接到的队列管理器。 *qmgr_name* is the name of the queue manager with all trailing blank characters removed and any forward slash (/) characters replaced by an ampersand (&) character.
- *resource_type* 指定要收集的资源数据的类型，它是下列其中一个字符串:
  - `ApplName` 以指定应用程序。 请求预订具有与 *resource_identifier* 指定的应用程序名称相匹配的应用程序名称的所有 IBM MQ 连接。
  - `ChannelName` 以指定 IBM MQ 通道。
  - `ConnectionId` ，用于指定 IBM MQ 连接。
- *resource_identifier* 标识实际资源。 格式取决于资源类型:
  - 对于资源类型 `ApplName`， *resource_identifier* 是队列管理器看到的应用程序名称的尾部部分 (跟在最后/或 \ 后面的值) ，并且除去了任何尾部空白字符。 该值与 API 出口上下文结构 (MQAXC) 中的 ApplName 值匹配。 使用 MQSC 命令 **DISPLAY CONN** 时，连接的 ApplName 将作为 APPLTAG 值返回。
  - 对于资源类型 ChannelName， *resource_identifier* 是要跟踪的通道的名称。 如果通道名称标识 SVRCONN 通道，那么将跟踪已连接客户机的所有应用程序活动。 如果通道名称向队列管理器通道标识队列管理器，那么将跟踪入局和出局消息。 The *resource_identifier* is the channel name with all trailing blank characters removed and any '/' characters replaced by a '&' character.
  - 对于资源类型 `ConnectionId`， *resource_identifier* 是分配给每个连接的唯一连接标识。 主题字符串中的连接标识是作为十六进制字符串写入的完整 24 字节值。 此值是 EXTCONN 的并置，后跟从 MQSC 命令 **DISPLAY CONN** 返回的 CONN 值。

  您可以在 *resource_identifier* 中使用通配符来匹配单个预订中的多个资源标识。 通配符可以是缺省主题样式 ("#" 或 "+") ，也可以是字符样式 ("*" 或 "?")。 使用主题样式通配符时，无法将其与部分资源名称组合，只能将其用于匹配所有可能的应用程序，通道或连接。 使用任何通配符都会增加生成的跟踪数据级别，这可能会影响性能。

要预订这些主题字符串，您必须具有 “预订” 权限。 系统主题不会从队列管理器主题树的根继承权限。 必须向用户授予对主题树中 $SYS/MQ 点以上的受管主题对象的访问权。 如果您有权访问 SYSTEM.ADMIN.TOPIC，尽管这将授予对所有 $SYS/MQ 主题字符串 (而不仅仅是活动跟踪) 的访问权。 为了更具体地控制访问权，可以为树中更深层次的点定义新的受管主题对象，例如针对所有活动跟踪，或者针对特定应用程序名称或通道名称。

示例

以下示例显示在 Windows 系统上运行的名为 `amqsput` 的应用程序的主题字符串:

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ApplName/amqsputc.exe
```

以下示例显示通道的主题字符串:

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ChannelName/SYSTEM.DEF.SVRCONN
```

以下示例显示连接的主题字符串:

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ConnectionId/
414D5143514D4752312020202020206B576B5420000701
```

以下示例显示了用于创建预订以跟踪队列管理器 QMGR1:

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ChannelName/#
```

以下示例显示了用于为名称以 “amqs” 开头的应用程序创建跟踪数据的预订的主题字符串 (请注意，要使用 “*” 通配符，必须使用字符通配符模型创建预订):

```
$SYS/MQ/INFO/QMGR/QMGR1/ActivityTrace/ApplName/amqs*
```

相关概念

第 260 页的『用于监视和活动跟踪的系统主题』
队列管理器主题树中的系统主题用于资源监视 (其中一些主题类似于统计信息消息的内容) ，并用作使用应用程序活动跟踪的方法。

使用 *amqsact* 来查看跟踪消息

您可以使用 **amqsact** 程序 来生成和查看跟踪消息。

amqsact 程序是 IBM MQ 样本。 要使用此样本，必须使用客户机连接的可执行文件 **amqsactc**。 可执行文件位于样本目录中:

- 在 Linux 和 UNIX 平台上， *MQ_INSTALLATION_PATH*/samp/bin64
- 在 Windows 平台上， *MQ_INSTALLATION_PATH*\tools\c\Samples\Bin64

您可以通过两种方式使用 **amqsact** :

显示方式
格式化并显示要传递到 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE。

动态方式
创建对一组资源的预订，并通过运行 **amqsact** 来显示生成的活动跟踪。

显示方式

缺省情况下，处于显示方式的 **amqsact** 在 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE。 您可以通过指定队列名称或主题字符串来覆盖此行为。 必须使用 收集应用程序活动跟踪信息中描述的其中一种方法来启用活动跟踪。 您可以控制所显示的跟踪时间段，并指定在显示后是除去还是保留活动跟踪消息。 在显示方式下， **amqsact** 采用以下参数:

-m 队列管理器名称
必需。 指定要为其收集跟踪消息的队列管理器。

-q 队列名称
仅显示与指定队列相关的跟踪消息。

-t 主题字符串
仅显示与指定主题相关的跟踪消息。

-b
指定在显示后保留跟踪消息。

-v
以详细方式显示跟踪消息。

-d 深度
要显示的消息数。

-w *timeout*
指定超时。 如果在该时间段内未显示任何跟踪消息，那么 **amqsact** 将退出。

-s 启动时间
将此自变量与 -e 自变量配合使用以指定时间段。 将显示来自指定时间段的跟踪消息。

-e *end_time*
将此参数与 -s 参数配合使用以指定时间段。 将显示来自指定时间段的跟踪消息。

例如，以下命令显示在 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE，并在显示后删除消息:

```
amqsact -m QMGR1
```

以下命令显示指定队列 SUB.QUEUE，并在显示后删除消息。 将继续显示消息，直到 30 秒后才会显示新消息。 例如，此命令可与活动跟踪系统主题字符串的预订配合使用。

```
amqact -m QMGR1 -q SUB.QUEUE.1 -w 30
```

以下命令以详细格式显示当前保存在 SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE 。 消息将在显示后保留在队列上。

```
amqsact -m QMGR1 -b -v -s 2014-12-31 23.50.00 -e 2015-01-01 00.10.00
```

动态方式

您可以通过指定应用程序名称，通道名称或连接标识作为 **amqsact** 的自变量来启用动态方式。 可以在名称中使用通配符。 在动态方式下，通过使用对系统主题的非持久预订，在样本开始时启用活动跟踪数据。 当 **amqsact** 停止时，收集活动跟踪数据将停止。 必须在动态方式下为 **amqsact** 指定超时。 您可以同时运行 **amqsact** 的多个副本，并且每个实例接收任何活动跟踪数据的副本。 在动态方式下， **amqsact** 采用以下参数:

-m 队列管理器名称
必需。 指定要为其收集跟踪消息的队列管理器。

-w *timeout*
必需。 指定超时。 如果在该时间段内未显示任何跟踪消息，那么 **amqsact** 将退出。

-a 应用程序名称
指定要为其收集消息的应用程序。

-c 通道名称
指定要为其收集消息的通道。

-i 连接标识
指定要为其收集消息的连接。

-v
以详细方式显示跟踪消息。

例如，以下命令生成并显示名为 “amqsget.exe”的应用程序所建立的任何连接的活动跟踪消息。 在不活动 30 秒后， **amqsact** 程序结束，并且不会生成新的活动跟踪数据。

```
amqsactc -m QMGR1 -w 30 -a amqsget.exe
```

以下命令将生成并显示以文本 “amqs”开头的应用程序所建立的任何连接的活动跟踪消息。 在不活动 30 秒后， **amqsact** 程序结束，并且不会生成新的活动跟踪数据。

```
amqsactc -m QMGR1 -w 30 -a amqs*
```

以下命令生成并显示 QMGR1.TO.QMGR2 通道。 在处于不活动状态 10 秒后， **amqsact** 程序结束，并且不会生成新的活动跟踪数据。

```
amqsactc -m QMGR1 -w 10 -c QMGR1.TO.QMGR2
```

以下命令生成并显示任何通道上任何活动的活动跟踪消息。 在处于不活动状态 10 秒后， **amqsact** 程序结束，并且不会生成新的活动跟踪数据。

```
amqsactc -m QMGR1 -w 10 -c #
```

以下命令生成并显示现有 IBM MQ 连接上具有 CONN "6B576B5420000701" 和 EXTCONN "414D5143514D47523120202020202020" 的任何活动的详细活动跟踪消息。 在不活动一分钟后，**amqsact** 程序结束，并且不会生成新的活动跟踪数据。

```
amqsactc -m QMGR1 -w 60 -i 414D5143514D4752312020202020206B576B5420000701 -v
```

使用 *mqat.ini* 配置跟踪级别

通过设置 mqat.ini 配置文件的 AllActivityTrace 节的值来配置队列管理器的跟踪级别。

可以为 AllActivityTrace 节设置以下值:

ActivityInterval
跟踪消息之间的时间间隔 (以秒为单位)。 活动跟踪不使用计时器线程，因此跟踪消息不会在经过时间的确切时刻写入，而是在经过时间间隔之后执行第一个 MQI 操作时写入。 如果此值为 0 ，那么将在连接断开时 (或达到活动计数时) 写入跟踪消息。 缺省值为 1。

ActivityCount
跟踪消息之间的 MQI 操作数。 如果此值为 0 ，那么将在连接断开 (或活动时间间隔已过) 时写入跟踪消息。 缺省值为 100。

TraceLevel
针对每个操作跟踪的参数详细信息量。 各个操作的描述详细说明了针对每个跟踪级别包含哪些参数。 设置为 LOW ， MEDIUM 或 HIGH。 缺省为 MEDIUM。

TraceMessage 数据
针对 MQGET ， MQPUT ， MQPUT1 和回调操作跟踪的消息数据量 (以字节计)。 缺省值为 0。

StopOnGetTrace 消息
可以设置为 ON 或 OFF。 缺省为 ON。

SubscriptionDelivery
可以设置为 BATCHED 或 IMMEDIATE。 确定当存在一个或多个活动跟踪预订时是否使用 ActivityInterval 和 ActivityCount 参数。 如果将此参数设置为 IMMEDIATE ，那么当跟踪数据具有匹配的预订时，将使用有效值 1 覆盖 ActivityInterval 和 ActivityCount 值。 不会将每个活动跟踪记录与来自同一连接的其他记录一起批处理，而是立即交付到预订而不会延迟。 IMMEDIATE 设置将增加收集活动跟踪数据的性能开销。 缺省设置为 BATCHED。

应用程序活动跟踪消息引用

使用此页面来获取应用程序活动跟踪消息的格式以及这些消息中返回的信息的概述

应用程序活动跟踪消息是包含消息描述符和消息数据的标准 IBM MQ 消息。 消息数据包含有关 IBM MQ 应用程序执行的 MQI 操作的信息，或者有关 IBM MQ 系统中发生的活动的信息。

消息描述符

- MQMD 结构

消息数据

- PCF 头 (MQCFH)
- 始终返回的应用程序活动跟踪消息数据
- 特定于操作的应用程序活动跟踪消息数据

应用程序活动跟踪消息 *MQMD (消息描述符)*

使用此页面来了解应用程序活动跟踪消息的消息描述符与事件消息的消息描述符之间的差异

应用程序活动跟踪消息的消息描述符中的参数和值与事件消息的消息描述符中的参数和值相同，但存在以下异常:

Format

描述: 消息数据的格式名。

值: **MQFMT_ADMIN**
管理消息。

CorrelId

描述: 相关标识。

值: 已使用应用程序的 ConnectionId 初始化

MQCFH (PCF 头)

使用此页面来查看活动跟踪消息的 MQCFH 结构包含的 PCF 值

对于活动跟踪消息， MQCFH 结构包含以下值:

Type

描述: 用于标识消息内容的结构类型。

数据类型: MQLONG。

值: MQCFT_APP_ACTIVITY

StrucLength

描述: MQCFH 结构的长度 (以字节为单位)。

数据类型: MQLONG。

值: MQCFH_STRUC_LENGTH

Version

描述: 结构版本号。

数据类型: MQLONG。

值: MQCFH_VERSION_3

Command

描述: 命令标识。 此字段标识消息的类别。

数据类型: MQLONG。

值: MQCMD_ACTIVITY_TRACE

MsgSeqNumber

描述: 消息序号。 此字段是一组相关消息中消息的序号。

数据类型: MQLONG。

值: 1

Control

描述: 控制选项。

数据类型: MQLONG。

值: MQCFC_LAST。

CompCode

描述: 完成代码。

数据类型: MQLONG。

值: MQCC_OK。

Reason

描述: 原因码限定完成代码。

数据类型: MQLONG。

值: MQRC_NONE。

ParameterCount

描述: 参数结构的计数。 此字段是遵循 MQCFH 结构的参数结构数。 组结构 (MQCFGR) 及其包含的参数结构仅计为一个结构。

数据类型: MQLONG。

值: 1 或更高版本

应用程序活动跟踪消息数据

紧跟在 PCF 头之后的是一组参数，用于描述活动跟踪的时间间隔。 这些参数还指示在写入消息时的消息序列。 不保证标题后面的字段顺序和数量，允许将来添加其他信息。

消息名称:	活动跟踪消息。
系统队列:	SYSTEM.ADMIN.TRACE.ACTIVITY.QUEUE.

QueueManager

描述: 队列管理器的名称。

标识: MQCA_Q_MGR_NAME

数据类型: MQCFST

最大长度: MQ_Q_MGR_NAME_LENGTH

QSGName

z/OS

描述: 队列管理器所属的队列共享组的名称 (仅限 z/OS)。

标识: MQCA_QSG_NAME

数据类型: MQCFST

最大长度: MQ_Q_MGR_NAME_LENGTH

HostName

描述: 运行队列管理器的机器的主机名。

标识: MQCACF_HOST_NAME

数据类型: MQCFST

IntervalStartDate

描述: 监视时间段开始的日期。

标识: MQCAMO_START_DATE

数据类型: MQCFST

最大长度: MQ_DATE_LENGTH

IntervalStartTime

描述: 监视时间段开始的时间。

标识: MQCAMO_START_TIME

数据类型: MQCFST

最大长度: MQ_TIME_LENGTH

IntervalEndDate

描述: 监视时间段结束的日期。

标识: MQCAMO_END_DATE

数据类型: MQCFST

最大长度: MQ_DATE_LENGTH

IntervalEndTime

描述: 监视时间段结束的时间。

标识: MQCAMO_END_TIME

数据类型: MQCFST

最大长度: MQ_TIME_LENGTH

CommandLevel

描述: IBM MQ 命令级别。

标识: MQIA_COMMAND_LEVEL

数据类型: MQCFIN

SeqNumber

描述: 序号通常为零。 对于长时间运行的连接的每个后续记录，此值将递增。

标识: MQIACF_SEQUENCE_NUMBER

数据类型: MQCFIN

ApplicationName

描述: 应用程序的名称 (程序名)。

标识: MQCACF_APPL_NAME

数据类型: MQCFST

最大长度: MQ_APPL_NAME_LENGTH

ApplClass

描述: 执行活动的应用程序的类型。 可能的值 :MQAT_ *

标识: MQIA_APPL_TYPE

数据类型: MQCFIN

ApplicationPid

描述: 应用程序的操作系统进程标识。

标识: MQIACF_PROCESS_ID

数据类型: MQCFIN

UserId

描述: 应用程序的用户标识上下文。

标识: MQCACF_USER_IDENTIFIER

数据类型: MQCFST

最大长度: MQ_USER_ID_LENGTH

APICallerType

描述: 应用程序的类型。 可能的值 :MQXACT_EXTERNAL 或 MQXACT_INTERNAL。

标识: MQIACF_API_CALLER_TYPE

数据类型: MQCFIN

Environment

描述: 应用程序的运行时环境。 可能的值 :MQXE_ *

标识: MQIACF_API_environment

数据类型: MQCFIN

ChannelName

描述: 与连接关联的通道名称。 仅当 Environment 参数的值为 MQXE_MCA 或 MQXE_MCA_SVRCONN 时，才会返回此参数。

标识: MQCACH_CHANNEL_NAME

数据类型: MQCFST

最大长度: MQ_CHANNEL_NAME_LENGTH

ConnectionName

描述: 与连接关联的网络连接名称。 仅当 Environment 参数的值为 MQXE_MCA 或 MQXE_MCA_SVRCONN 时，才会返回此参数。

标识: MQCACH_CONNECTION_NAME

数据类型: MQCFST

最大长度: MQ_CONN_NAME_LENGTH

ChannelType

描述: 与连接关联的通道的类型。 仅当 Environment 参数的值为 MQXE_MCA 或 MQXE_MCA_SVRCONN 时，才会返回此参数。 可能的值 :MQCHT_ *

标识: MQIACH_CHANNEL_TYPE

数据类型: MQCFIN

RemoteProduct

描述: 与连接关联的远程产品标识。 仅当 Environment 参数的值为 MQXE_MCA 或 MQXE_MCA_SVRCONN 时，才会返回此参数。

标识: MQCACH_REMOTE_PRODUCT

数据类型: MQCFST

最大长度: MQ_REMOTE_PRODUCT_LENGTH

RemoteVersion

描述: 与连接关联的远程产品版本。 仅当 Environment 参数的值为 MQXE_MCA 或 MQXE_MCA_SVRCONN 时，才会返回此参数。

标识: MQCACH_REMOTE_VERSION

数据类型: MQCFST

最大长度: MQ_REMOTE_VERSION_LENGTH

FunctionName

描述: 由初始线程启动的最后一个高级函数的名称。

标识: MQCACF_APPL_FUNCTION

数据类型: MQCFST

FunctionType

描述: 由初始线程启动的最后一个高级函数的类型。 可能的值 :MQFUN_ *

标识: MQIACF_APPL_FUNCTION_TYPE

数据类型: MQCFIN

Detail

描述: 为连接记录的详细信息级别。 可能的值: 1=LOW 2=MEDIUM 3=HIGH

标识: MQIACF_TRACE_DETAIL

数据类型: MQCFIN

TraceDataLength

描述: 针对此连接跟踪的消息数据的长度 (以字节计)。

标识: MQIACF_TRACE_DATA_LENGTH

数据类型: MQCFIN

PointerSize

描述: 应用程序正在运行的平台上指针的长度 (以字节计) (用于帮助解释二进制结构)。

标识: MQIACF_POINTER_SIZE

数据类型: MQCFIN

Platform

描述: 运行队列管理器的平台。 可能的值 :MQPL_ *

标识: MQIA_PLATFORM

数据类型: MQCFIN

应用程序活动 *MQI* 操作的变量参数

应用程序活动数据 MQCFGR 结构后跟与正在执行的操作相对应的一组 PCF 参数。 以下部分中定义了每个操作的参数。

跟踪级别指示要包含在跟踪中的参数所需的跟踪详细程度级别。 可能的跟踪级别值为:

1. 低

   当为应用程序配置了 “low”， “medium” 或 “high” 活动跟踪时，将包含此参数。 此设置表示参数始终包含在操作的 AppActivityData 组中。 这组参数足以跟踪应用程序进行的 MQI 调用，并查看它们是否成功。

2. 中等

   仅当为应用程序配置了 “medium” 或 “high” 活动跟踪时，该参数才包含在操作的 AppActivityData 组中。 此参数集添加有关资源的信息，例如，应用程序使用的队列和主题名称。

3. 高

   仅当为应用程序配置了 “高” 活动跟踪时，该参数才包含在操作的 AppActivityData 组中。 这组参数包括传递到 MQI 和 XA 函数的结构的内存转储。 因此，它包含有关 MQI 和 XA 调用中使用的参数的更多信息。 结构内存转储是结构的浅副本。 为避免错误尝试取消引用指针，将结构中的指针值设置为 NULL。

   注: 转储的结构版本不一定与应用程序使用的版本相同。 该结构可由 API 交叉出口，活动跟踪代码或队列管理器修改。 队列管理器可以将结构修改为更高版本，但队列管理器从不将其更改为该结构的较低版本。 这样做会有丢失数据的风险。

MQBACK
应用程序已启动 MQBACK MQI 函数

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型 MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型 MQCFIN

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQBEGIN
应用程序已启动 MQBEGIN MQI 函数

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型 MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型 MQCFIN

MQBO

描述: MQBEGIN 选项结构。 如果在 MQBEGIN 调用上使用 NULL 指针，那么不包含此参数。

PCF 参数: MQBACF_MQBO_STRUCT

跟踪级别: 3

类型 MQCFBS

长度: MQBO 结构的长度 (以字节计)。

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQCALLBACK
应用程序已启动 MQCALLBACK 函数

ObjectHandle

描述: 对象句柄

PCF 参数: MQIACF_HOBJ

跟踪级别: 1

类型 MQCFIN

CallType

描述: 为什么调用了函数。 其中一个 MQCBCT_ * 值

PCF 参数: MQIACF_CALL_TYPE

跟踪级别: 1

类型 MQCFIN

MsgBuffer

描述: 消息数据。

PCF 参数: MQBACF_MESSAGE_DATA

跟踪级别: 1

类型 MQCFBS

长度: 长度由 APPTRACE 配置中设置的 TRACEDATA () 参数控制。 如果 TRACEDATA=NONE , 那么将省略此参数。

MsgLength

描述: 消息的长度。 (取自 MQCBC 结构中的 DataLength 字段)。

PCF 参数: MQIACF_MSG_LENGTH

跟踪级别: 1

类型 MQCFIN

HighResTime

描述: 自 1970 年 1 月 1st 午夜 (UTC) 以来的操作时间 (以微秒为单位)

注: 此计时器的准确性根据平台对高分辨率计时器的支持而有所不同

PCF 参数: MQIAMO64_HIGHRES_TIME

跟踪级别: 2

类型 MQCFIN64

ReportOptions

描述: 报告消息的选项

PCF 参数: MQIACF_REPORT

跟踪级别: 2

类型 MQCFIN

MsgType

描述: 消息类型

PCF 参数: MQIACF_MSG_TYPE

跟踪级别: 2

类型 MQCFIN

Expiry

描述: 消息生命周期

PCF 参数: MQIACF_EXPIRY

跟踪级别: 2

类型 MQCFIN

Format

描述: 消息数据的格式名称

PCF 参数: MQCACH_FORMAT_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_FORMAT_LENGTH

Priority

描述: 消息优先级

PCF 参数: Mqiacf_priority

跟踪级别: 2

类型 MQCFIN

Persistence

描述: 消息持久性

PCF 参数: Mqiacf_persistence

跟踪级别: 2

类型 MQCFIN

MsgId

描述: 消息标识

PCF 参数: MQBACF_MSG_ID

跟踪级别: 2

类型 MQCFBS

长度: MQ_MSG_ID_LENGTH

CorrelId

描述: 相关标识

PCF 参数: MQBACF_CORREL_ID

跟踪级别: 2

类型 MQCFBS

长度: MQ_CORREL_ID_LENGTH

ObjectName

描述: 打开的对象的名称。

PCF 参数: MQCACF_OBJECT_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH

ResolvedQName

描述: 从中检索消息的队列的本地名称。

PCF 参数: MQCACF_RESOLVED_Q_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH

ReplyToQueue

描述: MQ_Q_NAME_LENGTH

PCF 参数: MQCACF_REPLY_TO_Q

跟踪级别: 2

类型 MQCFST

ReplyToQMgr

描述: MQ_Q_MGR_NAME_LENGTH

PCF 参数: MQCACF_REPLY_TO_Q_MGR

跟踪级别: 2

类型 MQCFST

CodedCharSetId

描述: 消息数据的字符集标识

PCF 参数: MQIA_CODED_CHAR_SET_ID

跟踪级别: 2

类型 MQCFIN

Encoding

描述: 消息数据的数字编码。

PCF 参数: Mqiacf_encoding

跟踪级别: 2

类型 MQCFIN

PutDate

描述: MQ_PUT_DATE_LENGTH

PCF 参数: MQCACF_PUT_DATE

跟踪级别: 2

类型 MQCFST

PutTime

描述: MQ_PUT_TIME_LENGTH

PCF 参数: MQCACF_PUT_TIME

跟踪级别: 2

类型 MQCFST

ResolvedQName

描述: 当 ResolvedType 为 MQOT_Q 时， ObjectHandle 引用的队列名称。

PCF 参数: MQCACF_RESOLVED_LOCAL _Q_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH。

ResObjectString

描述: 当 ResolvedType 为 MQOT_TOPIC 时， ObjectHandle 引用的对象名。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型 MQCFST

长度: 长度有所不同。

ResolvedType

描述: ObjectHandle 引用的对象的类型。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型 MQCFIN

PolicyName

描述: 应用于此消息的策略名称。

注: 仅受 AMS 保护的消息

PCF 参数: MQCA_POLICY_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_OBJECT_NAME_LENGTH

XmitqMsgId

描述: 传输队列头中消息的消息标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQBACF_XQH_MSG_ID

跟踪级别: 2

类型 MQCFBS

长度: MQ_MSG_ID_LENGTH

XmitqCorrelId

描述: 传输队列头中消息的相关标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQBACF_XQH_CORREL_ID

跟踪级别: 2

类型 MQCFBS

长度: MQ_CORREL_ID_LENGTH

XmitqPutTime

描述: 消息在传输队列头中的放置时间。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_PUT_TIME

跟踪级别: 2

类型 MQCFST

长度: MQ_PUT_TIME_LENGTH

XmitqPutDate

描述: 消息在传输队列头中的放置日期。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_PUT_DATE

跟踪级别: 2

类型 MQCFST

长度: MQ_PUT_DATE_LENGTH

XmitqRemoteQName

描述: 传输队列头中消息的远程队列目标。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_REMOTE_Q_Name

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH

XmitqRemoteQMgr

描述: 传输队列头中消息的消息标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_REMOTE_Q_MGR

跟踪级别: 2

类型 MQCFST

长度: MQ_MSG_ID_LENGTH

MsgDescStructure

描述: MQMD 结构。 如果使用了 V 4 MQGMO 来请求返回消息句柄而不是 MQMD ， 那么将省略此参数

PCF 参数: MQBACF_MQMD_STRUCT

跟踪级别: 3

类型 MQCFBS

长度: MQMD 结构的长度 (以字节计) (实际大小取决于结构版本)

GetMsgOptsStructure

描述: MQGMO 结构。

PCF 参数: MQBACF_MQGMO_STRUCT

跟踪级别: 3

类型 MQCFBS

长度: MQGMO 结构的长度 (以字节计) (实际大小取决于结构版本)

MQCBContextStructure

描述: MQCBC 结构。

PCF 参数: MQBACF_MQCBC_STRUCT

跟踪级别: 3

类型 MQCFBS

长度: MQCBC 结构的长度 (以字节计) (实际大小取决于结构版本)

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如， 作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQCB
应用程序已启动管理回调 MQI 函数

CallbackOperation

描述: 管理回调函数操作。 设置为其中一个 MQOP_ * 值

PCF 参数: MQIACF_MQCB_OPERATION

跟踪级别: 1

类型 MQCFIN

CallbackType

描述: 回调函数的类型 (MQCBD 结构中的 CallbackType 字段)。 设置为其中一个 MQCBT_ * 值

PCF 参数: MQIACF_MQCB_TYPE

跟踪级别: 1

类型 MQCFIN

CallbackOptions

描述: 回调选项。 设置为其中一个 MQCBDO_ * 值

PCF 参数: MQIACF_MQCB_OPTIONS

跟踪级别: 1

类型 MQCFIN

CallbackFunction

描述: 回调函数的指针 (如果作为函数调用启动)。

PCF 参数: MQBACF_MQCB_FUNCTION

跟踪级别: 1

类型 MQCFBS

长度: MQPTR 的大小

CallbackName

描述: 作为动态链接程序启动的回调函数的名称。

PCF 参数: MQCACF_MQCB_NAME

跟踪级别: 1

类型 MQCFST

长度: MQCHAR128 的大小

ObjectHandle

描述: 对象句柄

PCF 参数: MQIACF_HOBJ

跟踪级别: 1

类型 MQCFIN

MaxMsgLength

描述: 最大消息长度。 设置为整数或特殊值 MQCBD_FULL_MSG_LENGTH

PCF 参数: MQIACH_MAX_MSG_LENGTH

跟踪级别: 2

类型 MQCFIN

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型 MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型 MQCFIN

ResolvedQName

描述: 当 ResolvedType 为 MQOT_Q 时， ObjectHandle 引用的队列名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH。

ResObjectString

描述: 当 ResolvedType 为 MQOT_TOPIC 时， ObjectHandle 引用的对象名。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型 MQCFST

长度: 长度有所不同。

ResolvedType

描述: ObjectHandle 引用的对象的类型。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型 MQCFIN

CallBack DescriptorStructure

描述: MQCBD 结构。 如果将 NULL MQCBC 值传递到 MQCB 调用，那么将省略此参数。

PCF 参数: MQBACF_MQCBD_STRUCT

跟踪级别: 3

类型 MQCFBS

长度: MQCBC 结构的长度 (以字节计)

MsgDescStructure

描述: MQMD 结构。 如果将 NULL MQMD 值传递到 MQCB 调用，那么将省略 MsgDesc 结构参数。

PCF 参数: MQBACF_MQMD_STRUCT

跟踪级别: 3

类型 MQCFBS

长度: MQMD 结构的长度 (以字节计) (实际大小取决于结构版本)

GetMsgOptsStructure

描述: MQGMO 结构。 如果将 NULL MQGMO 值传递到 MQCB 调用，那么将省略此参数。

PCF 参数: MQBACF_MQGMO_STRUCT

跟踪级别: 3

类型 MQCFBS

长度: MQGMO 结构的长度 (以字节计) (实际大小取决于结构版本)

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQCLOSE
应用程序已启动 MQCLOSE MQI 函数

ObjectHandle

描述: 对象句柄

PCF 参数: MQIACF_HOBJ

跟踪级别: 1

类型 MQCFIN

CloseOptions

描述: 关闭选项

PCF 参数: MQIACF_CLOSE_OPTIONS

跟踪级别: 1

类型 MQCFIN

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型 MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型 MQCFIN

ResolvedQName

描述: 当 ResolvedType 为 MQOT_Q 时， ObjectHandle 引用的队列名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH。

ResObjectString

描述: 当 ResolvedType 为 MQOT_TOPIC 时， ObjectHandle 引用的对象名。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型 MQCFST

长度: 长度有所不同。

ResolvedType

描述: ObjectHandle 引用的对象的类型。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型 MQCFIN

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQCMIT
应用程序已启动 MQCMIT MQI 函数

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型 MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型 MQCFIN

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQCONN 和 *MQCONNX*
应用程序已启动 MQCONN 或 MQCONNX MQI 函数

ConnectionId

描述: 连接标识 (如果可用) 或 MQCONNID_NONE (如果未提供)

PCF 参数: MQBACF_CONNECTION_ID

跟踪级别: 1

类型: MQCFBS

最大长度: MQ_CONNECTION_ID_LENGTH

QueueManagerName

描述: MQCONN (X) 调用中使用的队列管理器的 (未解析) 名称

PCF 参数: MQCA_Q_MGR_NAME

跟踪级别: 1

类型: MQCFST

最大长度: MQ_Q_MGR_NAME_LENGTH

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

ConnectOptions

描述: 从 MQCNO_ * 值派生的连接选项

注: 仅 MQCONNX

PCF 参数: MQIACF_CONNECT_OPTIONS

跟踪级别: 2

类型: MQCFIN

ConnectionOptionsStructure

描述: MQCNO 结构。

注: 仅 MQCONNX)

PCF 参数: MQBACF_MQCNO_STRUCT

跟踪级别: 3

类型: MQCFBS

最大长度: MQCNO 结构的长度 (以字节为单位) (实际大小取决于结构版本)

ChannelDefinitionStructure

描述: MQCD 结构。

注: 仅客户机连接

PCF 参数: MQBACF_MQCD_STRUCT

跟踪级别: 3

类型: MQCFBS

最大长度: MQCD 结构的长度 (以字节为单位) (实际大小取决于结构版本)

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQCTL
应用程序已启动 MQCTL MQI 函数

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

CtlOperation

描述: MQOP_ * 值之一

PCF 参数: MQIACF_CTL_OPERATION

跟踪级别: 1

类型: MQCFIN

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQDISC
应用程序已启动 MQDISC MQI 函数

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

MQGET
应用程序已启动 MQGET MQI 函数

ObjectHandle

描述: 对象句柄

PCF 参数: MQIACF_HOBJ

跟踪级别: 1

类型: MQCFIN

GetOptions

描述: 来自 MQGMO.Options

PCF 参数: MQIACF_GET_OPTIONS

跟踪级别: 1

类型: MQCFIN

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

MsgBuffer

描述: 消息数据。 如果 TRACEDATA=NONE , 那么将省略此参数

PCF 参数: MQBACF_MESSAGE_DATA

跟踪级别: 1

类型: MQCFBS

最大长度: 长度由 APPTRACE 配置中设置的 TRACEDATA () 参数控制。 (作为 MQIACF_TRACE_DATA_LENGTH 包含在跟踪消息中)。

MsgLength

描述: 消息的长度。

PCF 参数: MQIACF_MSG_LENGTH

跟踪级别: 1

类型: MQCFIN

HighResTime

描述: 自 1970 年 1 月 1 午夜 (UTC) 以来的操作时间 (以微秒为单位)

注: 此计时器的准确性根据平台对高分辨率计时器的支持而有所不同

PCF 参数: MQIAMO64_HIGHRES_TIME

跟踪级别: 2

类型: MQCFIN64

BufferLength

描述: 应用程序提供的缓冲区的长度

PCF 参数: MQIACF_BUFFER_LENGTH

跟踪级别: 2

类型: MQCFIN

ObjectName

描述: 打开的对象的名称

PCF 参数: MQCACF_OBJECT_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

ResolvedQName

描述: 从中检索消息的队列的本地名称。

PCF 参数: MQCACF_RESOLVED_Q_NAME

跟踪级别: 2

类型: MQCFST

最大长度: MQ_Q_NAME_LENGTH

ReportOptions

描述: 消息报告选项

PCF 参数: MQIACF_REPORT

跟踪级别: 2

类型: MQCFIN

MsgType

描述: 消息类型

PCF 参数: MQIACF_MSG_TYPE

跟踪级别: 2

类型: MQCFIN

Expiry

描述: 消息生命周期

PCF 参数: MQIACF_EXPIRY

跟踪级别: 2

类型: MQCFIN

Format

描述: 消息数据的格式名称

PCF 参数: MQCACH_FORMAT_NAME

跟踪级别: 2

类型: MQCFST

最大长度: MQ_FORMAT_LENGTH

Priority

描述: 消息优先级

PCF 参数: Mqiacf_priority

跟踪级别: 2

类型: MQCFIN

Persistence

描述: 消息持久性

PCF 参数: Mqiacf_persistence

跟踪级别: 2

类型: MQCFIN

MsgId

描述: 消息标识

PCF 参数: MQBACF_MSG_ID

跟踪级别: 2

类型: MQCFBS

最大长度: MQ_MSG_ID_LENGTH

CorrelId

描述: 相关标识

PCF 参数: MQBACF_CORREL_ID

跟踪级别: 2

类型: MQCFBS

最大长度: MQ_CORREL_ID_LENGTH

ReplyToQueue

描述:

PCF 参数: MQCACF_REPLY_TO_Q

跟踪级别: 2

类型: MQCFST

最大长度: MQ_Q_NAME_LENGTH

ReplyToQMgr

描述:

PCF 参数: MQCACF_REPLY_TO_Q_MGR

跟踪级别: 2

类型: MQCFST

最大长度: MQ_Q_MGR_NAME_LENGTH

CodedCharSetId

描述: 消息数据的字符集标识

PCF 参数: MQIA_CODED_CHAR_SET_ID

跟踪级别: 2

类型: MQCFIN

Encoding

描述: 消息数据的数字编码。

PCF 参数: Mqiacf_encoding

跟踪级别: 2

类型: MQCFIN

PutDate

描述:

PCF 参数: MQCACF_PUT_DATE

跟踪级别: 2

类型: MQCFST

最大长度: MQ_PUT_DATE_LENGTH

PutTime

描述:

PCF 参数: MQCACF_PUT_TIME

跟踪级别: 2

类型: MQCFST

最大长度: MQ_PUT_TIME_LENGTH

ResolvedQName

描述: 当 ResolvedType 为 MQOT_Q 时， ObjectHandle 引用的队列名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH。

ResObjectString

描述: 当 ResolvedType 为 MQOT_TOPIC 时， ObjectHandle 引用的对象名。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型 MQCFST

长度: 长度有所不同。

ResolvedType

描述: ObjectHandle 引用的对象的类型。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型 MQCFIN

PolicyName

描述: 应用于此消息的策略名称。

注: 仅受 AMS 保护的消息

PCF 参数: MQCA_POLICY_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_OBJECT_NAME_LENGTH

XmitqMsgId

描述: 传输队列头中消息的消息标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQBACF_XQH_MSG_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_MSG_ID_LENGTH

XmitqCorrelId

描述: 传输队列头中消息的相关标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQBACF_XQH_CORREL_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_CORREL_ID_LENGTH

XmitqPutTime

描述: 消息在传输队列头中的放置时间。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_PUT_TIME

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_TIME_LENGTH

XmitqPutDate

描述: 消息在传输队列头中的放置日期。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_PUT_DATE

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_DATE_LENGTH

XmitqRemoteQName

描述: 传输队列头中消息的远程队列目标。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_REMOTE_Q_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

XmitqRemoteQMgr

描述: 传输队列头中消息的远程队列管理器目标。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_REMOTE_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

MsgDescStructure

描述: MQMD 结构。

PCF 参数: MQBACF_MQMD_STRUCT

跟踪级别: 3

类型: MQCFBS

最大长度: MQMD 结构的长度 (以字节计) (实际大小取决于结构版本)

GetMsgOptsStructure

描述: MQGMO 结构。

PCF 参数: MQBACF_MQGMO_STRUCT

跟踪级别: 3

类型: MQCFBS

最大长度: MQGMO 结构的长度 (以字节计) (实际大小取决于结构版本)

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQINQ
应用程序已启动 MQINQ MQI 函数

ObjectHandle

描述: 对象句柄
PCF 参数: MQIACF_HOBJ
跟踪级别: 1
类型: MQCFIN

CompCode

描述: 指示操作结果的完成代码
PCF 参数: MQIACF_COMP_CODE
跟踪级别: 1
类型: MQCFIN

Reason

描述: 操作的原因码结果
PCF 参数: MQIACF_REASON_CODE
跟踪级别: 1
类型: MQCFIN

SelectorCount

描述: 选择器数组中提供的选择器计数。
PCF 参数: MQIACF_SELECTOR_COUNT
跟踪级别: 2
类型: MQCFIN

Selectors

描述: 其值必须由 MQINQ 返回的属性 (整数或字符) 的列表。
PCF 参数: MQIACF_SELECTORS
跟踪级别: 2
类型: MQCFIL

ResolvedQName

描述: 当 ResolvedType 为 MQOT_Q 时， ObjectHandle 引用的队列名称。
PCF 参数: MQCACF_RESOLVED_Q_NAME
跟踪级别: 2
类型: MQCFST
最大长度: MQ_Q_NAME_LENGTH

ResObjectString

描述: 当 ResolvedType 为 MQOT_TOPIC 时， ObjectHandle 引用的对象名。
PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型: MQCFST

最大长度: 长度变化

ResolvedType

描述: ObjectHandle 引用的对象的类型。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型: MQCFIN

IntAttrCount

描述: 查询操作返回的整数属性数

PCF 参数: MQIACF_INTATTR_COUNT

跟踪级别: 3

类型: MQCFIN

IntAttrs

描述: 查询操作返回的整数属性值。 仅当 MQINQ 返回时 IntAttrCount is> 0 时，才会显示此参数。

PCF 参数: MQIACF_INT_ATTRS

跟踪级别: 3

类型: MQCFIL

CharAttrs

描述: 查询操作返回的字符属性。 这些值并置在一起。 仅当 MQINQ 返回时 CharAttr 长度大于 0 时，才会包含此参数。

PCF 参数: MQCACF_CHAR_ATTRS

跟踪级别: 3

类型: MQCFST

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQOPEN
应用程序已启动 MQOPEN MQI 函数

ObjectType

描述: 在 MQOT.ObjectType

PCF 参数: MQIACF_OBJECT_TYPE

跟踪级别: 1

类型: MQCFIN

ObjectName

描述: 在尝试任何队列名称解析之前传递到 MQI 调用的对象的名称。

PCF 参数: MQCACF_OBJECT_NAME

跟踪级别: 1

类型: MQCFST

最大长度: MQ_Q_NAME_LENGTH

ObjectQMgrName

描述: 在尝试任何队列名称解析之前传递到 MQI 调用的对象队列管理器的名称。

PCF 参数: MQCACF_OBJECT_Q_MGR_NAME

跟踪级别: 1

类型: MQCFST

最大长度: MQ_Q_MGR_NAME_LENGTH

ObjectHandle

描述: 对象句柄

PCF 参数: MQIACF_HOBJ

跟踪级别: 1

类型: MQCFIN

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

OpenOptions

描述: 用于打开对象的选项

PCF 参数: MQIACF_OPEN_OPTIONS

跟踪级别: 1

类型: MQCFIN

AlternateUserId

描述: 仅当指定了 MQOO_ALTERNATE_USER_AUTHORITY 时才包含

PCF 参数: MQCACF_ALTERNATE_USERID

跟踪级别: 2

类型: MQCFST

最大长度: MQ_USER_ID_LENGTH

RecsPresent

描述: 存在的对象名记录数。 仅当 MQOD 版本> = MQOD_VERSION_2 时才包含

PCF 参数: MQIACF_RECS_PRESENT

跟踪级别: 1

类型: MQCFIN

KnownDestCount

描述: 仅当 MQOD 版本> = MQOD_VERSION_2 时，才包括成功打开的本地队列数

PCF 参数: MQIACF_KNOWN_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

UnknownDestCount

描述: 仅当 MQOD 版本> = MQOD_VERSION_2 时成功打开的远程队列数

PCF 参数: MQIACF_UNKNOWN_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

InvalidDestCount

描述: 仅当 MQOD 版本> = MQOD_VERSION_2 时，未能打开的队列数

PCF 参数: MQIACF_INVALID_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

DynamicQName

描述: 作为输入传递到 MQOPEN 调用的动态队列名称。

PCF 参数: MQCACF_DYNAMIC_Q_NAME

跟踪级别: 2

类型: MQCFST

最大长度: MQ_Q_NAME_LENGTH

ResolvedLocalQName [1] [2]

描述: 包含执行名称解析后的本地队列名称。 (例如，对于远程队列，这将是传输队列的名称)

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型: MQCFST

范围: 如果 MQOD.Version 小于 MQOD_VERSION_3 ，此版本包含 MQOD.ObjectName 字段。 如果 MQOD.Version 等于或高于 MQOD_VERSION_3 ，这包含 MQOD 中的值。 ResolvedQName 字段。

最大长度: MQ_Q_NAME_LENGTH

ResolvedLocalQMgrName [1] [2]

描述: 执行名称解析后的本地队列管理器名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_MGR

跟踪级别: 2

类型: MQCFST

范围: 仅当 MQOD.Version > = MQOD_VERSION_3

最大长度: MQ_Q_MGR_NAME_LENGTH

ResolvedQName [1] [2]

描述: 已执行名称解析后的队列名称。

PCF 参数: MQCACF_RESOLVED_Q_NAME

跟踪级别: 2

类型: MQCFST

范围: 如果 MQOD.Version 小于 MQOD_VERSION_3 ，此版本包含 MQOD.ObjectName 字段。 如果 MQOD.Version 等于或高于 MQOD_VERSION_3 ，这包含 MQOD 中的值。 ResolvedQName 字段。

最大长度: MQ_Q_NAME_LENGTH

ResolvedQMgrName [1] [2]

描述: 在执行名称解析后包含队列管理器名称。 如果 MQOD.Version 低于 MQOD_VERSION_3 ，这包含 MQOD 的值。 ObjectQMgrMQOPEN 调用完成后的 "名称" 字段。 如果 MQOD.Version 等于或高于 MQOD_VERSION_3 ，这包含 MQOD 中的值。 ResolvedQMgr 名称字段。

PCF 参数: MQCACF_RESOLVED_Q_MGR

跟踪级别: 2

类型: MQCFST

最大长度: MQ_Q_MGR_NAME_LENGTH

AlternateSecurityId

描述: 备用安全标识。 仅当 MQOD.Version 等于或大于 MQOD_VERSION_3，指定了 MQOO_ALTERNATE_USER_AUTHORITY 和 MQOD.AlternateSecurityId 不等于 MQSID_NONE。

PCF 参数: MQBACF_ALTERNATE_SECURITYID

跟踪级别: 2

类型: MQCFBS

最大长度: MQ_SECURITY_ID_LENGTH

ObjectString

描述: 长对象名。 仅在 MQOD.Version 等于或高于 MQOD_VERSION_4 和 MQOD.ObjectString 为 MQVS_NULL_TERMINATED 或大于零。

PCF 参数: MQCACF_OBJECT_STRING

跟踪级别: 2

类型: MQCFST

最大长度: 长度有所不同。

SelectionString

描述: 选定项字符串。 仅在 MQOD.Version 等于或高于 MQOD_VERSION_4 和 MQOD 的 VSL 思字段。 SelectionString 是 MQVS_NULL_TERMINATED 或大于零。

PCF 参数: MQCACF_SELECTION_STRING

跟踪级别: 2

类型: MQCFST

最大长度: 长度有所不同。

ResObjectString

描述: 队列管理器解析 ObjectName 字段中提供的名称后的长对象名。 仅包含用于引用主题对象 (如果为 MQOD.Version 等于或大于 MQOD_VERSION_4 ， VSL 思为 MQVS_NULL_TERMINATED 或大于零。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型: MQCFST

最大长度: 长度有所不同。

ResolvedType

描述: 要打开的已解析 (基本) 对象的类型。 仅在 MQOD.Version 等于或高于 MQOD_VERSION_4。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型: MQCFIN

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

应用程序活动分发列表 *PCF* 组标题结构
如果 MQOPEN 函数打开分发列表，那么 MQOPEN 参数针对分发列表中的每个队列包含一个 AppActivityDistList PCF 组，直至 RecsPresent 中编号的结构数为止。 Ap-pActivityDistList PCF 组组合 MQOR 和 MQRR 结构中的信息以标识队列名称，并指示队列上的打开操作的结果。 AppActivityDistList 组始终以以下 MQCFGR 结构开头:

表 *29: AppActivityDistList* 组 *MQCFGR* 结构

MQCFGR 字段	**值**	**描述**
类型	MQCFT_GROUP	
StrucLength	MQCFGR 结构的长度 (以字节计)	
参数	MQGACF_APP_DIST_LIST	分发列表组参数
ParameterCount	4	此组中包含的 MQCFGR 结构后面的参数结构数。

ObjectName

描述: 分发列表 MQ_Q_NAME_LENGTH 中队列的名称。 仅在提供 MQOR 结构时包含。

PCF 参数: MQCACF_OBJECT_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH。 仅在提供 MQOR 结构时包含。

ObjectQMgrName

描述: 定义了 ObjectName 中指定的队列的队列管理器的名称。

PCF 参数: MQCACF_OBJECT_Q_MGR_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_MGR_NAME_LENGTH。 仅在提供 MQOR 结构时包含。

CompCode

描述: 指示此对象的打开结果的完成代码。 仅当提供了 MQRR 结构并且 MQOPEN 的原因码为 MQRC_MULTIPLE_REASON 时才包含

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 2

类型: MQCFIN

Reason

描述: 指示此对象的打开结果的原因码。 仅当提供了 MQRR 结构并且 MQOPEN 的原因码为 MQRC_MULTIPLE_REASON 时才包含

[1] 仅当要打开的对象解析为队列，并且针对 MQOO_INPUT_ * ， MQOO_OUTPUT 或 MQOO_BROWSE 打开队列时，才会包含此参数

[2] 仅当 ResolvedLocalQName 参数与 ResolvedQName 参数不同时，才会包含此参数。

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 2

类型: MQCFIN

MQPUT
应用程序已启动 MQPUT MQI 函数。

ObjectHandle

描述: 对象句柄

PCF 参数: MQIACF_HOBJ

跟踪级别: 1

类型: MQCFIN

PutOptions

描述: 来自 MQPMO.Options

PCF 参数: MQIACF_PUT_OPTIONS

跟踪级别: 1

类型: MQCFIN

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

MsgBuffer

描述: 消息数据。

PCF 参数: MQBACF_MESSAGE_DATA

跟踪级别: 1

类型: MQCFBS

长度: 长度由 APPTRACE 配置中设置的 TRACEDATA () 参数控制。 如果 TRACEDATA=NONE ，那么将省略此参数。

MsgLength

描述: 消息的长度。

PCF 参数: MQIACF_MSG_LENGTH

跟踪级别: 1

类型: MQCFIN

RecsPresent

描述: 存在的放入消息记录或响应记录数。 仅当 MQPMO 版本> = MQPMO_VERSION_2 时包含

PCF 参数: MQIACF_RECS_PRESENT

跟踪级别: 1

类型: MQCFIN

KnownDestCount

描述: 成功发送到本地队列的消息数

PCF 参数: MQIACF_KNOWN_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

UnknownDestCount

描述: 成功发送到远程队列的消息数

PCF 参数: MQIACF_UNKNOWN_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

InvalidDestCount

描述: 无法发送的消息数

PCF 参数: MQIACF_INVALID_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

HighResTime

描述: 自 1970 年 1 月 1st 午夜 (UTC) 以来的操作时间 (以微秒为单位)

注: 此计时器的准确性根据平台对高分辨率计时器的支持而有所不同。

PCF 参数: MQIAMO64_HIGHRES_TIME

跟踪级别: 2

类型: MQCFIN64

ObjectName

描述: 打开的对象的名称。

PCF 参数: MQCACF_OBJECT_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

ResolvedQName

描述: 执行队列名称解析后的队列名称。

PCF 参数: MQCACF_RESOLVED_Q_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

ResolvedQMgrName

描述: 执行名称解析后的队列管理器名称。

PCF 参数: MQCACF_RESOLVED_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_MGR_NAME_LENGTH

ResolvedLocalQName [3]

描述: 包含执行名称解析后的本地队列名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型: MQCFST

ResolvedLocalQMgrName [3]

描述: 在执行名称解析后包含本地队列管理器名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_MGR_NAME_LENGTH

ReportOptions

描述: 消息报告选项

PCF 参数: MQIACF_REPORT

跟踪级别: 2

类型: MQCFIN

MsgType

描述: 消息类型

PCF 参数: MQIACF_MSG_TYPE

跟踪级别: 2

类型: MQCFIN

Expiry

描述: 消息生命周期

PCF 参数: MQIACF_EXPIRY

跟踪级别: 2

类型: MQCFIN

Format

描述: 消息数据的格式名称

PCF 参数: MQCACH_FORMAT_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_FORMAT_LENGTH

Priority

描述: 消息优先级

PCF 参数: Mqiacf_priority

跟踪级别: 2

类型: MQCFIN

Persistence

描述: 消息持久性

PCF 参数: Mqiacf_persistence

跟踪级别: 2

类型: MQCFIN

MsgId

描述: 消息标识

PCF 参数: MQBACF_MSG_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_MSG_ID_LENGTH

CorrelId

描述: 相关标识

PCF 参数: MQBACF_CORREL_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_CORREL_ID_LENGTH

ReplyToQueue

描述:

PCF 参数: MQCACF_REPLY_TO_Q

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

ReplyToQMgr

描述:

PCF 参数: MQCACF_REPLY_TO_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_MGR_NAME_LENGTH

CodedCharSetId

描述: 消息数据的字符集标识

PCF 参数: MQIA_CODED_CHAR_SET_ID

跟踪级别: 2

类型: MQCFIN

Encoding

描述: 消息数据的数字编码。

PCF 参数: Mqiacf_encoding

跟踪级别: 2

类型: MQCFIN

PutDate

描述:

PCF 参数: MQCACF_PUT_DATE

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_DATE_LENGTH

PutTime

描述:

PCF 参数: MQCACF_PUT_TIME

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_TIME_LENGTH

ResolvedQName

描述: 当 ResolvedType 为 MQOT_Q 时， ObjectHandle 引用的队列名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH。

ResObjectString

描述: 当 ResolvedType 为 MQOT_TOPIC 时， ObjectHandle 引用的对象名。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型: MQCFST

长度: 长度有所不同。

ResolvedType

描述: ObjectHandle 引用的对象的类型。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型: MQCFIN

PolicyName

描述: 应用于此消息的策略名称。

注: 仅受 AMS 保护的消息

PCF 参数: MQCA_POLICY_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_OBJECT_NAME_LENGTH

XmitqMsgId

描述: 传输队列头中消息的消息标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQBACF_XQH_MSG_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_MSG_ID_LENGTH

XmitqCorrelId

描述: 传输队列头中消息的相关标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQBACF_XQH_CORREL_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_CORREL_ID_LENGTH

XmitqPutTime

描述: 消息在传输队列头中的放置时间。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_PUT_TIME

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_TIME_LENGTH

XmitqPutDate

描述: 消息在传输队列头中的放置日期。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_PUT_DATE

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_DATE_LENGTH

XmitqRemoteQName

描述: 传输队列头中消息的远程队列目标。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_REMOTE_Q_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

XmitqRemoteQMgr

描述: 传输队列头中消息的远程队列管理器目标。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_REMOTE_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

PutMsgOptsStructure

描述: MQPMO 结构。

PCF 参数: MQBACF_MQPMO_STRUCT

跟踪级别: 3

类型: MQCFBS

长度: MQPMO 结构的长度 (字节) (实际大小取决于结构版本)

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQPUT 应用程序活动分发列表 *PCF* 组头结构

如果 MQPUT 函数正在放入分发列表，那么 MQPUT 参数包含一个 AppActivityDistList PCF 组。 对于分发列表中的每个队列，请参阅 第 226 页的『应用程序活动分发列表 PCF 组标题结构』。 AppActivityDistList PCF 组组合 MQPMR 和 MQRR 结构中的信息以标识 PUT 参数，并指示对每个队列执行 PUT 操作的结果。 对于 MQPUT 操作， AppActivityDistList 组包含以下部分或全部参数 (如果原因码为 MQRC_MULTIPLE_REASON 并且其他参数由 MQPMO.PutMsgRecFields 字段确定，那么将显示 CompCode 和 Reason):

CompCode

描述: 指示操作结果的完成代码。 仅在提供了 MQRR 结构并且 MQPUT 的原因码为 MQRC_MULTIPLE_REASON 时包含

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 2

类型: MQCFIN

Reason

描述: 指示此对象的放置结果的原因码。 仅在提供了 MQRR 结构并且 MQPUT 的原因码为 MQRC_MULTIPLE_REASON 时包含

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 2

类型: MQCFIN

MsgId

描述: 消息标识。 仅当 MQPMR 结构为 provided.and PutMsgRecFields 包含 MQPMRF_MSG_ID 时包含

PCF 参数: MQBACF_MSG_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_MSG_ID_LENGTH

CorrelId

描述: 相关标识。 仅当 MQPMR 结构为 provided.and PutMsgRecFields 包含 MQPMRF_CORREL_ID 时才包含

[3] 仅当 ResolvedLocalQName 参数与 ResolvedQName 参数不同时，才会包含此参数。

PCF 参数: MQBACF_CORREL_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_CORREL_ID_LENGTH

GroupId

描述: 组标识。 仅当 MQPMR 结构为 provided.and PutMsgRecFields 包含 MQPMRF_GROUP_ID 时才包含

PCF 参数: MQBACF_GROUP_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_GROUP_ID_LENGTH

Feedback

描述: 反馈。 仅当 MQPMR 结构为 provided.and PutMsgRecFields 包含 MQPMRF_FEEDBACK 时包含

PCF 参数: MQIACF_FEEDBACK

跟踪级别: 2

类型: MQCFIN

AccountingToken

描述: AccountingToken. 仅当 MQPMR 结构为 provided.and PutMsgRecFields 时才包含 MQPMRF_ACCOUNTING_TOKEN

PCF 参数: MQBACF_ACCOUNTING_TOKEN

跟踪级别: 2

类型: MQCFBS

长度: MQ_ACCOUNTING_TOKEN_LENGTH。

MQPUT1
应用程序已启动 MQPUT1 MQI 函数

ObjectType

描述: 在 MQOT.ObjectType

PCF 参数: MQIACF_OBJECT_TYPE

跟踪级别: 1

类型: MQCFIN

ObjectName

描述: 在尝试任何队列名称解析之前传递到 MQI 调用的对象的名称。

PCF 参数: MQCACF_OBJECT_NAME

跟踪级别: 1

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

ObjectQMgrName

描述: 在尝试任何队列名称解析之前传递到 MQI 调用的对象队列管理器的名称。

PCF 参数: MQCACF_OBJECT_Q_MGR_NAME

跟踪级别: 1

类型: MQCFST

长度: MQ_Q_MGR_NAME_LENGTH

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

PutOptions

描述: 来自 MQPMO.Options

PCF 参数: MQIACF_PUT_OPTIONS

跟踪级别: 1

类型: MQCFIN

AlternateUserId

描述: 仅当指定了 MQPMO_ALTERNATE_USER_AUTHORITY 时才包括在内。

PCF 参数: MQCACF_ALTERNATE_USERID

跟踪级别: 2

类型: MQCFST

长度: MQ_USER_ID_LENGTH

RecsPresent

描述: 存在的对象名记录数

PCF 参数: MQIACF_RECS_PRESENT

跟踪级别: 1

类型: MQCFIN

KnownDestCount

描述: 成功打开的本地队列数

PCF 参数: MQIACF_KNOWN_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

UnknownDestCount

描述: 成功打开的远程队列数

PCF 参数: MQIACF_UNKNOWN_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

InvalidDestCount

描述: 未能打开的队列数

PCF 参数: MQIACF_INVALID_DEST_COUNT

跟踪级别: 1

类型: MQCFIN

MsgBuffer

描述: 消息数据。

PCF 参数: MQBACF_MESSAGE_DATA

跟踪级别: 1

类型: MQCFBS

长度: 长度由 APPTRACE 配置中设置的 TRACEDATA () 参数控制。 如果 TRACEDATA=NONE , 那么将省略此参数。

MsgLength

描述: 消息的长度。

PCF 参数: MQIACF_MSG_LENGTH

跟踪级别: 1

类型: MQCFIN

HighResTime

描述: 自 1970 年 1 月 1st 午夜 (UTC) 以来的操作时间 (以微秒为单位)

注: 根据平台对高分辨率计时器的支持，此计时器的准确性将有所不同。

PCF 参数: MQIAMO64_HIGHRES_TIME

跟踪级别: 2

类型: MQCFIN64

ResolvedQName

描述: 执行队列名称解析后的队列名称。

PCF 参数: MQCACF_RESOLVED_Q_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

ResolvedQMgrName

描述: 执行名称解析后的队列管理器名称。

PCF 参数: MQCACF_RESOLVED_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_MGR_NAME_LENGTH

ResolvedLocalQName [4]

描述: 执行名称解析后包含本地队列名称

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型: MQCFST

ResolvedLocalQMgrName [4]

描述: 在执行名称解析后包含本地队列管理器名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_MGR_NAME_LENGTH

AlternateSecurityId

描述: 备用安全标识。 仅当 MQOD.Version 等于或大于 MQOD_VERSION_3 和 MQOD.AlternateSecurityId 不等于 MQSID_NONE。

PCF 参数: MQBACF_ALTERNATE_SECURITYID

跟踪级别: 2

类型: MQCFBS

长度: MQ_SECURITY_ID_LENGTH

ObjectString

描述: 长对象名。 仅在 MQOD.Version 等于或高于 MQOD_VERSION_4 和 MQOD.ObjectString 为 MQVS_NULL_TERMINATED 或大于零。

PCF 参数: MQCACF_OBJECT_STRING

跟踪级别: 2

类型: MQCFST

长度: 长度有所不同。

ResObjectString

描述: 队列管理器解析 ObjectName 字段中提供的名称后的长对象名。 仅包含用于引用主题对象 (如果为 MQOD.Version 等于或大于 MQOD_VERSION_4 , VSL 思为 MQVS_NULL_TERMINATED 或大于零。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型: MQCFST

长度: 长度有所不同。

ResolvedType

描述: 要打开的已解析 (基本) 对象的类型。 仅在 MQOD.Version 等于或高于 MQOD_VERSION_4。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型: MQCFIN

ReportOptions

描述: 消息报告选项

PCF 参数: MQIACF_REPORT

跟踪级别: 2

类型: MQCFIN

MsgType

描述: 消息类型

PCF 参数: MQIACF_MSG_TYPE

跟踪级别: 2

类型: MQCFIN

Expiry

描述: 消息生命周期

PCF 参数: MQIACF_EXPIRY

跟踪级别: 2

类型: MQCFIN

Format

描述: 消息数据的格式名称

PCF 参数: MQCACH_FORMAT_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_FORMAT_LENGTH

Priority

描述: 消息优先级

PCF 参数: Mqiacf_priority

跟踪级别: 2

类型: MQCFIN

Persistence

描述: 消息持久性

PCF 参数: Mqiacf_persistence

跟踪级别: 2

类型: MQCFIN

MsgId

描述: 消息标识

PCF 参数: MQBACF_MSG_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_MSG_ID_LENGTH

CorrelId

PCF 参数: 相关标识

描述: MQBACF_CORREL_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_CORREL_ID_LENGTH

ReplyToQueue

描述:

PCF 参数: MQCACF_REPLY_TO_Q

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

ReplyToQMgr

描述:

PCF 参数: MQCACF_REPLY_TO_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQCFST

CodedCharSetId

描述: 消息数据的字符集标识

PCF 参数: MQIA_CODED_CHAR_SET_ID

跟踪级别: 2

类型: MQCFIN

Encoding

描述: 消息数据的数字编码。

PCF 参数: Mqiacf_encoding

跟踪级别: 2

类型: MQCFIN

PutDate

描述:

PCF 参数: MQCACF_PUT_DATE

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_DATE_LENGTH

PutTime

描述:

PCF 参数: MQCACF_PUT_TIME

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_TIME_LENGTH

PolicyName

描述: 应用于此消息的策略名称。

注: 仅受 AMS 保护的消息

PCF 参数: MQCA_POLICY_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_OBJECT_NAME_LENGTH

XmitqMsgId

描述: 传输队列头中消息的消息标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQBACF_XQH_MSG_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_MSG_ID_LENGTH

XmitqCorrelId

描述: 传输队列头中消息的相关标识。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQBACF_XQH_CORREL_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_CORREL_ID_LENGTH

XmitqPutTime

描述: 消息在传输队列头中的放置时间。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_PUT_TIME

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_TIME_LENGTH

XmitqPutDate

描述: 消息在传输队列头中的放置日期。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_PUT_DATE

跟踪级别: 2

类型: MQCFST

长度: MQ_PUT_DATE_LENGTH

XmitqRemoteQName

描述: 传输队列头中消息的远程队列目标。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_REMOTE_Q_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

XmitqRemoteQMgr

描述: 传输队列头中消息的远程队列管理器目标。

注: 仅当格式为 MQFMT_XMIT_Q_HEADER 时

PCF 参数: MQCACF_XQH_REMOTE_Q_MGR

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

PutMsgOptsStructure

描述: MQPMO 结构。

PCF 参数: MQBACF_MQPMO_STRUCT

跟踪级别: 3

类型: MQCFBS

长度: MQPMO 结构的长度 (字节) (实际大小取决于结构版本)

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQPUT1 AppActivityDistList PCF 组头结构

如果 MQPUT1 函数放入分发列表，那么变量参数包含一个 AppActivityDistList PCF 组。 对于分发列表中的每个队列，请参阅 第 226 页的『应用程序活动分发列表 PCF 组标题结构』。 AppActivityDistList PCF 组组合来自 MQOR， MQPMR 和 MQRR 结构的信息，以标识对象和 PUT 参数，并指示每个队列上 PUT 操作的结果。 对于 MQPUT1 操作， AppActivityDistList 组包含以下部分或全部参数 (如果原因码为 MQRC_MULTIPLE_REASON 并且其他参数由 MQPMO.PutMsgRecFields 字段确定，那么将显示 CompCode， Reason， ObjectName 和 ObjectQMgrName):

CompCode

描述: 指示此对象的放置结果的完成代码。 仅当提供了 MQRR 结构并且 MQPUT1 的原因码为 MQRC_MULTIPLE_REASON 时才包含

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 2

类型: MQCFIN

Reason

描述: 指示此对象的放置结果的原因码。 仅当提供了 MQRR 结构并且 MQPUT1 的原因码为 MQRC_MULTIPLE_REASON 时才包含

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 2

类型: MQCFIN

ObjectName

描述: 分发列表中队列的名称。 仅在提供 MQOR 结构时包含。

PCF 参数: MQCACF_OBJECT_NAME

跟踪级别: 2

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

[4] 仅当 ResolvedLocalQName 参数与 ResolvedQName 参数不同时，才会包含此参数。

MsgId

描述: 消息标识。 仅当 MQPMR 结构为 provided.and PutMsgRecFields 包含 MQPMRF_MSG_ID 时包含

PCF 参数: MQBACF_MSG_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_MSG_ID_LENGTH

CorrelId

描述: 相关标识。 仅当 MQPMR 结构为 provided.and PutMsgRecFields 包含 MQPMRF_CORREL_ID 时才包含

PCF 参数: MQBACF_CORREL_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_CORREL_ID_LENGTH

GroupId

描述: 组标识。 仅当 MQPMR 结构为 provided.and PutMsgRecFields 包含 MQPMRF_GROUP_ID 时才包含

PCF 参数: MQBACF_GROUP_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_GROUP_ID_LENGTH

Feedback

描述: 反馈。 仅当 MQPMR 结构为 provided.and PutMsgRecFields 包含 MQPMRF_FEEDBACK 时包含

PCF 参数: MQIACF_FEEDBACK

跟踪级别: 2

类型: MQCFIN

AccountingToken

描述: AccountingToken. 仅当 MQPMR 结构为 provided.and PutMsgRecFields 时才包含 MQPMRF_ACCOUNTING_TOKEN

PCF 参数: MQBACF_ACCOUNTING_TOKEN

跟踪级别: 2

类型: MQCFBS

长度: MQ_ACCOUNTING_TOKEN_LENGTH。

MQSET
应用程序已启动 MQSET MQI 函数

ObjectHandle

描述: 对象句柄

PCF 参数: MQIACF_HOBJ

跟踪级别: 1

类型: MQCFIN

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

SelectorCount

描述: 选择器数组中提供的选择器计数。

PCF 参数: MQIACF_SELECTOR_COUNT

跟踪级别: 2

类型: MQCFIN

Selectors

描述: 要由 MQSET 更新其值的属性 (整数或字符) 的列表。

PCF 参数: MQIACF_SELECTORS

跟踪级别: 2

类型: MQCFIL

ResolvedQName

描述: 当 ResolvedType 为 MQOT_Q 时， ObjectHandle 引用的队列名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH。

ResObjectString

描述: 当 ResolvedType 为 MQOT_TOPIC 时， ObjectHandle 引用的对象名。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型 MQCFST

长度: 长度有所不同。

ResolvedType

描述: ObjectHandle 引用的对象的类型。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型 MQCFIN

IntAttrCount

描述: 要由 set 操作更新的整数属性数。

PCF 参数: MQIACF_INTATTR_COUNT

跟踪级别: 3

类型: MQCFIN

IntAttrs

描述: 整数属性值

PCF 参数: MQIACF_INT_ATTRS

跟踪级别: 3

类型: MQCFIL

范围: 仅当 IntAttrCount is> 0 时，此参数才存在

CharAttrs

描述: 要由 set 操作更新的字符属性。 这些值并置在一起。

PCF 参数: MQCACF_CHAR_ATTRS

跟踪级别: 3

类型: MQCFST

范围: 仅当 CharAttr 长度大于 0 时，才会包含此参数

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQSUB
应用程序已启动 MQSUB MQI 函数

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

SubHandle

描述: 预订句柄

PCF 参数: MQIACF_HSUB

跟踪级别: 1

类型: MQCFIN

ObjectHandle

描述: 对象句柄

PCF 参数: MQIACF_HOBJ

跟踪级别: 1

类型: MQCFIN

Options

描述: 预订选项

PCF 参数: MQIACF_SUB_OPTIONS

跟踪级别: 1

类型: MQCFIN

ObjectName

描述: 对象的名称。

PCF 参数: MQCACF_OBJECT_NAME

跟踪级别: 1

类型: MQCFST

长度: MQ_Q_NAME_LENGTH

ObjectString

描述: 长对象名。

PCF 参数: MQCACF_OBJECT_STRING

跟踪级别: 1

类型: MQCFST

范围: 仅包含在 MQSD.ObjectString 大于零或 MQVS_NULL_TERMINATED。

长度: 长度有所不同。

AlternateUserId

描述:

PCF 参数: MQCACF_ALTERNATE_USERID

跟踪级别: 2

类型: MQCFST

范围: 仅在指定 MQSO_ALTERNATE_USER_AUTHORITY 时包含。

长度: MQ_USER_ID_LENGTH

AlternateSecurityId

描述: 备用安全标识。

PCF 参数: MQBACF_ALTERNATE_SECURITYID

跟踪级别: 2

类型: MQCFBS

范围: 仅当指定了 MQSO_ALTERNATE_USER_AUTHORITY 和 MQSD.AlternateSecurityId 不等于 MQSID_NONE。

长度: MQ_SECURITY_ID_LENGTH

SubName

描述: 预订名称

PCF 参数: MQCACF_SUB_NAME

跟踪级别: 2

类型: MQCFST

范围: 仅当 MQSD.SubName 的 VSL 思长度字段大于零或 MQVS_NULL_TERMINATED 时才包括在内。

长度: 长度有所不同。

SubUserData

描述: 预订用户数据

PCF 参数: MQCACF_SUB_USER_DATA

跟踪级别: 2

类型: MQCFST

范围: 仅当 MQSD.SubName 的 VSL 思长度字段大于零或 MQVS_NULL_TERMINATED 时才包括在内。

长度: 长度有所不同。

SubCorrelId

描述: 预订相关标识

PCF 参数: MQBACF_SUB_CORREL_ID

跟踪级别: 2

类型: MQCFBS

长度: MQ_CORREL_ID_LENGTH

SelectionString

描述: 选定项字符串。

PCF 参数: MQCACF_SELECTION_STRING

跟踪级别: 2

类型: MQCFST

范围: 仅当 MQSD 的 VSL 思长度字段时才包含。 SelectionString 是 MQVS_NULL_TERMINATED 或大于零。

长度: 长度有所不同。

ResolvedQName

描述: 当 ResolvedType 为 MQOT_Q 时， ObjectHandle 引用的队列名称。

PCF 参数: MQCACF_RESOLVED_LOCAL_Q_NAME

跟踪级别: 2

类型 MQCFST

长度: MQ_Q_NAME_LENGTH。

ResObjectString

描述: 当 ResolvedType 为 MQOT_TOPIC 时， ObjectHandle 引用的对象名。

PCF 参数: MQCACF_RESOLVED_OBJECT_STRING

跟踪级别: 2

类型 MQCFST

长度: 长度有所不同。

ResolvedType

描述: ObjectHandle 引用的对象的类型。 可能的值为 MQOT_Q ， MQOT_TOPIC 或 MQOT_NONE。

PCF 参数: MQIACF_RESOLVED_TYPE

跟踪级别: 2

类型 MQCFIN

SubDescriptorStructure

描述: MQSD 结构。

PCF 参数: MQBACF_MQSD_STRUCT

跟踪级别: 3

类型: MQCFBS

长度: MQSD 结构的长度 (以字节计)。

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQSUBRQ
应用程序已启动 MQSUBRQ MQI 函数

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

SubHandle

描述: 预订句柄

PCF 参数: MQIACF_HSUB

跟踪级别: 1

类型: MQCFIN

SubOptions

描述: 来自 MQSB.Options

PCF 参数: MQIACF_SUBRQ_OPTIONS

跟踪级别: 2

类型: MQCFIN

Action

描述: 预订请求操作 (MQSR_ *)

PCF 参数: MQIACF_SUBRQ_ACTION

跟踪级别: 2

类型: MQCFIN

NumPubs

描述: 由于此调用 (来自 MQSB.NumPubs)

PCF 参数: MQIACF_NUM_PUBS

跟踪级别: 2

类型: MQCFIN

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

MQSTAT
应用程序已启动 MQSTAT MQI 函数

CompCode

描述: 指示操作结果的完成代码

PCF 参数: MQIACF_COMP_CODE

跟踪级别: 1

类型: MQCFIN

Reason

描述: 操作的原因码结果

PCF 参数: MQIACF_REASON_CODE

跟踪级别: 1

类型: MQCFIN

Type

描述: 正在请求的状态信息的类型

PCF 参数: MQIACF_STATUS_TYPE

跟踪级别: 2

类型: MQCFIN

StatusStructure

描述: MQSTS 结构。

PCF 参数: MQBACF_MQSTS_STRUCT

跟踪级别: 3

类型: MQCFBS

长度: MQSTS 结构的长度 (字节) (实际大小取决于结构版本)

QMgrOpDuration

描述: 队列管理器中的近似 API 调用持续时间 (以微秒为单位)。

持续时间不包括在队列管理器外部花费的时间。 例如，作为 IBM MQ 客户机所用的时间。

注: 根据企业使用的平台，此计时器的准确性会有所不同。

PCF 参数: MQIAMO64_QMGR_OP_DURATION

跟踪级别: 2

类型 MQCFIN64

应用程序活动 *XA* 操作的变量参数

XA 操作是应用程序可以进行的 API 调用，以使 MQ 能够参与事务。 以下部分中定义了每个操作的参数。

跟踪级别指示要包含在跟踪中的参数所需的跟踪详细程度级别。 可能的跟踪级别值为:

1. 低

   当为应用程序配置了 “low”， “medium” 或 “high” 活动跟踪时，将包含此参数。 此设置表示参数始终包含在操作的 AppActivityData 组中。 这组参数足以跟踪应用程序进行的 MQI 调用，并查看它们是否成功。

2. 中等

   仅当为应用程序配置了 “medium” 或 “high” 活动跟踪时，该参数才包含在操作的 AppActivityData 组中。 此参数集添加有关资源的信息，例如，应用程序使用的队列和主题名称。

3. 高

   仅当为应用程序配置了 “高” 活动跟踪时，该参数才包含在操作的 AppActivityData 组中。 这组参数包括传递到 MQI 和 XA 函数的结构的内存转储。 因此，它包含有关 MQI 和 XA 调用中使用的参数的更多信息。 结构内存转储是结构的浅副本。 为避免错误尝试取消引用指针，将结构中的指针值设置为 NULL。

   注: 转储的结构版本不一定与应用程序使用的版本相同。 该结构可由 API 交叉出口，活动跟踪代码或队列管理器修改。 队列管理器可以将结构修改为更高版本，但队列管理器从不将其更改为该结构的较低版本。 这样做会有丢失数据的风险。

AXREG
应用程序已启动 AXREG AX 函数

XID

描述: XID 结构

PCF 参数: MQBACF_XA_XID

跟踪级别: 1

类型: MQCFBS

长度: 大小 (XID)

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

AXUNREG
应用程序已启动 AXUNREG AX 函数

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XACLOSE
应用程序已启动 XAC 洛斯 AX 函数

Xa_info

描述: 用于初始化资源管理器的信息。

PCF 参数: MQCACF_XA_INFO

跟踪级别: 1

类型: MQCFST

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XACOMMIT
应用程序已启动 XACOMMIT AX 函数

XID

描述: XID 结构

PCF 参数: MQBACF_XA_XID

跟踪级别: 1

类型: MQCFBS

长度: 大小 (XID)

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XACOMPLETE
应用程序已启动 XACOMPLETE AX 功能

Handle

描述: 异步操作的句柄

PCF 参数: MQIACF_XA_HANDLE

跟踪级别: 1

类型: MQCFIN

Retval

描述: 异步函数的返回值

PCF 参数: MQIACF_XA_RETVAL

跟踪级别: 1

类型: MQCFINMQCFBS

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XAEND
应用程序已启动 XAEND AX 函数

XID

描述: XID 结构

PCF 参数: MQBACF_XA_XID

跟踪级别: 1

类型: MQCFBS

长度: 大小 (XID)

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XAFORGET
应用程序已启动 AXREG AX 函数

XID

描述: XID 结构

PCF 参数: MQBACF_XA_XID

跟踪级别: 1

类型: MQCFBS

长度: 大小 (XID)

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XAOPEN
应用程序已启动 XAOPEN AX 函数

Xa_info

描述: 用于初始化资源管理器的信息。

PCF 参数: MQCACF_XA_INFO

跟踪级别: 1

类型: MQCFST

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XAPREPARE
应用程序已启动 XAP 重新解析 AX 函数

XID

描述: XID 结构

PCF 参数: MQBACF_XA_XID

跟踪级别: 1

类型: MQCFBS

长度: 大小 (XID)

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

Xarecover
应用程序已启动 XARECOVER AX 函数

Count

描述: XID 计数

PCF 参数: MQIACF_XA_COUNT

跟踪级别: 1

类型: MQCFIN

XIDs

描述: XID 结构

注: 此 PCF 参数有多个实例-每个 XID 结构都有一个实例，直至计数 XID

PCF 参数: MQBACF_XA_XID

跟踪级别: 1

类型: MQCFBS

长度: 大小 (XID)

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XAROLLBACK
应用程序已启动 XAROLLBACK AX 函数

XID

描述: XID 结构

PCF 参数: MQBACF_XA_XID

跟踪级别: 1

类型: MQCFBS

长度: 大小 (XID)

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

XASTART
应用程序已启动 XASTART AX 函数

XID

描述: XID 结构

PCF 参数: MQBACF_XA_XID

跟踪级别: 1

类型: MQCFBS

长度: 大小 (XID)

Rmid

描述: 资源管理器标识

PCF 参数: MQIACF_XA_RMID

跟踪级别: 1

类型: MQCFIN

Flags

描述: 标志

PCF 参数: MQIACF_XA_FLAGS

跟踪级别: 1

类型: MQCFIN

XARetCode

描述: 返回码

PCF 参数: MQIACF_XA_RETCODE

跟踪级别: 1

类型: MQCFIN

Multi 用于监视和活动跟踪的系统主题

队列管理器主题树中的系统主题用于资源监视 (其中一些主题类似于统计信息消息的内容)，并用作使用应用程序活动跟踪的方法。

队列管理器主题树的 $SYS/MQ 分支

每个队列管理器的主题树都包含 $SYS/MQ 分支。 队列管理器将发布到此分支中的主题字符串。 授权用户可以预订这些主题字符串，以接收有关队列管理器及其活动的信息。 这些系统主题用于应用程序活动跟踪和监视。 有关更多信息，请参阅 主题树。

$SYS/MQ 分支的根由 SYSTEM.ADMIN.TOPIC 主题对象。 主题树的 $SYS/MQ 分支通过以下方式与主题树的其余部分隔离:

- 在树中高于 $SYS/MQ 的位置使用通配符进行的预订与 $SYS/MQ 分支中的任何主题字符串都不匹配。 SYSTEM.ADMIN.TOPIC 设置为 “块”，无法修改。 当您将通配符与 **runmqsc** 命令 DISPLAY TPSTATUS 配合使用以显示主题树中的节点时，此限制也适用。 要查看 $SYS/MQ 分支中的主题节点，请使用 $SYS/MQ 启动主题字符串。 例如，使用 $SYS/MQ/# 来查看所有节点。
- 您必须在 $SYS/MQ 上获得授权，才能获得使用 $SYS/MQ 主题树的权限。 预订主题字符串的授权基于在主题树中的主题字符串处或高于主题字符串的受管主题对象的授权。 在根 (SYSTEM.BASE.TOPIC) 将授予用户对所有主题字符串的权限。 但是，对于 $SYS/MQ 分支，授予的高于 $SYS/MQ 的访问权不适用于 $SYS/MQ 主题字符串。
- 主题树的 $SYS/MQ 分支与树中较高设置的主题属性隔离。 SYSTEM.ADMIN.TOPIC 不会从主题树中定义的更高级别的主题对象继承任何属性。 例如，更改 SYSTEM.BASE.TOPIC 不会影响 $SYS/MQ 分支的行为。

所有以 $SYS/MQ 开头的主题字符串都保留供 IBM MQ 使用。 这些主题字符串具有以下限制:

- 无法从主题树的 $SYS/MQ 分支启用多点广播。
- $SYS/MQ 分支不支持集群。
- 无法将代理预订机制设置为 “force”。
- 应用程序无法发布到 $SYS/MQ 主题字符串。
- 发布和预订作用域缺省为仅本地队列管理器。
- 不得在以下位置使用通配符:
  - $SYS/MQ/
  - $SYS/MQ/INFO
  - $SYS/MQ/INFO/QMGR
  - $SYS/MQ/INFO/QMGR/*queue_manager_name*

– $SYS/MQ/INFO/QMGR/*queue_manager_name*/ActivityTrace

尝试在这些点使用通配符会导致预订失败，原因为 MQRC_ADMIN_TOPIC_STRING_ERROR。

通过发布/预订提供的性能监视 API

您可以使用简单发布/预订机制以及 amqsrua 样本应用程序或您自己的应用程序来监视队列管理器统计信息。 统计信息发布到 $SYS/MQ/INFO/QMGR 下的系统主题中，以帮助用户监视资源。 可以通过运行 amqsrua 样本应用程序或编写以类似于 **amqsrua** 的方式预订资源监视系统主题的应用程序来查看这些统计信息。

相关概念

第 179 页的『应用程序活动跟踪』
应用程序活动跟踪生成有关连接到队列管理器的应用程序的行为的详细信息。 它跟踪应用程序的行为，并提供应用程序在与 IBM MQ 资源交互时使用的参数的详细视图。 它还显示了应用程序发出的 MQI 调用的序列。

Windows Linux 使用 amqsrua 命令监视系统资源使用情况

您可以使用 **amqsrua** 命令来查询与队列管理器的系统资源使用情况相关的性能数据。

关于此任务

amqsrua 样本应用程序展示了一种使用 IBM MQ 监视发布并显示由队列管理器发布的性能数据的方法。 此数据可以包含有关 CPU，内存和磁盘使用情况的信息。 您还可以查看等同于 STATMQI PCF 统计数据的数据。 数据每 10 秒发布一次，并在命令运行时报告。

您可以仅使用队列管理器名称运行该命令，并以交互方式逐步选择在每个步骤中可供队列管理器使用的 **CLASS**， **TYPE** 和 **object** 参数。 如果您知道要查看其信息的 **CLASS**， **TYPE** 和 **object** 名称，那么可以在运行 **amqsrua** 命令时指定这些名称。

缺省情况下， amqsrua 应用程序查找队列管理器在主题树 $SYS/MQ/INFO/QMGR 下发布的统计信息。 其他组件或应用程序可以使用类似机制在不同的主题起点下发布。 从 IBM MQ 9.1.0 开始，您可以使用 **-p** 参数来指定 amqsrua 在 Linux 和 Windows 上查找这些其他组件的统计信息的位置。

-m
队列管理器名称。 该队列管理器必须正在运行。 如果未指定队列管理器名称，那么将显示缺省队列管理器的统计信息。

-c
类名。 IBM MQ 资源使用情况发布与类相关联。 这些类表示描述可用资源使用情况信息的元数据树中的顶级。

CPU
返回有关 CPU 使用率的信息。

磁盘
返回有关磁盘使用情况的信息。

STATMQI
返回有关 MQI 使用情况的信息。

STATQ
返回有关每个队列的 MQI 使用情况的信息。

STATAPP
返回有关指定应用程序的使用情况统计信息的信息。

请参阅 第 263 页的『开发您自己的资源监视应用程序』，以获取有关如何为元数据指定主题树的信息，以及有关可以在应用程序名称中使用的字符的 使用受支持编程语言中的应用程序名称 的信息。

注: 您可以使用 **-o** 选项来指定正在为 STATAPP 监视的应用程序名称以及 STATQ 的队列名称。

-t
TYPE 名称。 IBM MQ 资源使用情况发布与类中的类型相关联。 每个出版物都包含允许找到 `class/type/element` 定义以及处理生成的出版物的类和类型。 `class/type/element` 描述在队列管理器启动时作为元数据发布。

-o
对象名称。 资源使用情况发布是由一系列 PCF 元素组成的 PCF 消息。 将在元数据中公布针对每个类/类型对发布的 PCF 元素。 叶子存储在描述每个元素的树中，从而允许处理这些元素。

-p
元数据前缀。 指定主题树起点， **amqsrua** 可以在该起点中查找由队列管理器发布的统计信息。 缺省主题树为 $SYS/MQ/INFO/QMGR，但其他组件或应用程序可能会在另一主题树起点下发布统计信息。

-n
发布计数。 您可以指定在命令结束之前返回的报告数。 大约每 10 秒发布一次数据，因此如果输入值 50，那么该命令将返回 500 秒内的 50 个报告。 如果未指定此参数，那么该命令将运行，直到发生错误或队列管理器关闭为止。

-s
模型队列。 (可选) 指定要使用的模型队列。 (缺省情况下， **amqsrua** 使用 SYSTEM.DEFAULT.MODEL.QUEUE 队列。)

-h
用法

过程

1. 从样本目录中，发出以下命令以显示队列管理器的可用数据:
   - Linux 在 Linux 上， *MQ_INSTALLATION_PATH*/samp/bin:
     ```
     ./amqsrua -m QMgrName
     ```
   - Windows 在 Windows 上， *MQ_INSTALLATION_PATH*\tools\c\Samples\Bin64:
     ```
     amqsrua -m QMgrName
     ```

   其中 *QMgrName* 指定要查询的队列管理器的名称。 该队列管理器必须正在运行。 如果未指定队列管理器名称，那么将使用缺省队列管理器。

   提供了以下选项:
   ```
   CPU : Platform central processing units
   DISK : Platform persistent data stores
   STATMQI : API usage statistics
   STATQ : API per-queue usage statistics
   Enter Class selection
   ==>
   ```
2. 从 CLASS 选项列表中，输入 STATMQI。
   ```
   ==> STATMQI
   CONNDISC : MQCONN and MQDISC
   OPENCLOSE : MQOPEN and MQCLOSE
   INQSET : MQINQ and MQSET
   PUT : MQPUT
   GET : MQGET
   SYNCPOINT : Commit and rollback
   SUBSCRIBE : Subscribe
   PUBLISH : Publish
   Enter Type selection
   ==>
   ```
3. 从 TYPE 选项列表中，输入 PUT。
   ```
   ==>PUT
   Publication received PutDate:20170329 PutTime:17045485 Interval:4 minutes,13.978 seconds
   Interval total MQPUT/MQPUT1 count 22
   ```

```
Interval total MQPUT/MQPUT1 byte count 25284 100/sec
Non-persistent message MQPUT count 22
Persistent message MQPUT count 0
Failed MQPUT count 0
Non-persistent message MQPUT1 count 0
Persistent message MQPUT1 count 0
Failed MQPUT1 count 0
Put non-persistent messages - byte count 25284 100/sec
Put persistent messages - byte count 0
MQSTAT count 0

Publication received PutDate:20170329 PutTime:17050485 Interval:10.001 seconds
Interval total MQPUT/MQPUT1 count 1
Interval total MQPUT/MQPUT1 byte count 524 52/sec
Non-persistent message MQPUT count 1
Persistent message MQPUT count 0
Failed MQPUT count 0
Non-persistent message MQPUT1 count 0
Persistent message MQPUT1 count 0
Failed MQPUT1 count 0
Put non-persistent messages - byte count 524 52/sec
Put persistent messages - byte count 0
MQSTAT count 0
```

结果

您已使用 **amqsrua** 样本应用程序以交互方式查看队列管理器在系统主题上的元数据前缀 $SYS/MQ/INFO/QMGR 下发布的统计信息。

注: 可供队列管理器使用的资源发布的类和类型可能有所不同，具体取决于其配置，版本和平台。 以交互方式使用 **amqsrua** 来查找可用于特定队列管理器的类，类型和元素。

下一步做什么

要开发您自己的监视应用程序，请参阅 第 263 页的『开发您自己的资源监视应用程序』。

相关任务

在 Windows 上准备和运行样本程序

在 UNIX 和 Linux 上准备和运行样本程序

ALW 开发您自己的资源监视应用程序

您可以开发自己的应用程序来监视系统资源。

每个队列管理器将资源使用情况数据发布到主题。 这些主题的订户将使用此数据。 当队列管理器启动时，队列管理器将在元主题上发布一组消息。 这些消息描述队列管理器支持的资源使用情况主题以及发布到这些主题的消息的内容。 管理工具可以预订元数据以发现可用的资源使用情况信息以及有关哪些主题的信息，然后预订已发布的主题。

元数据的主题树具有以下结构:

```
$SYS/MQ/INFO/QMGR/QMGR-NAME/Monitor/class[/instance]/type]
```

将以下结构用于主题树的元数据:

```
$SYS/MQ/INFO/QMGR/QMGR-NAME/Monitor/class[/resourceid]/type]
```

以明确您正在指定要监视的资源 (最多具有 28 个字符的应用程序名称)，并避免与内部元数据类型属性混淆。

有关可能的类的列表，请参阅 第 261 页的『使用 amqsrua 命令监视系统资源使用情况』。

amqsrua 应用程序的源代码作为 IBM MQ 样本提供。 您可以使用此样本应用程序作为创建您自己的监视应用程序的指南。 您可以从 IBM MQ 客户机安装中检索样本的源。 源文件名为 amqsruaa.c，位于样本目录中:

- Linux AIX 在 AIX and Linux 平台上， *MQ_INSTALLATION_PATH*/samp/

- Windows 在 Windows 平台上， *MQ_INSTALLATION_PATH*\tools\c\Samples\

amqsrua 应用程序预订 IBM MQ 资源使用情况主题，并对生成的已发布 PCF 数据进行格式化。 应用程序源提供了有关如何预订和使用此类管理数据的基本示例。 amqsrua 应用程序完成以下任务:

- 创建由输入参数标识的主题的非持久预订。
- 重复调用 MQGET 以从主题获取消息，并写入 stdout。
- 针对每个 MQI 原因 (MQRC_NONE 除外) 写入一条消息。
- 如果 MQI 完成代码为 MQCC_FAILED ，或者已使用所请求的资源使用情况发布数，那么停止。

Multi 在系统主题上发布的度量

度量分类为类，子分类为类型。 在每个度量值类和类型下都发布了各种度量值。

索引

Windows Linux 请参阅 第 261 页的『使用 amqsrua 命令监视系统资源使用情况』，以获取有关如何收集所列选项 (NHAREPLICA 除外) 的数据的信息。

您还可以使用 ALTER QMGR 命令 在队列管理器级别监视 STATMQI 和 STATQ ，或者使用本地队列属性 STATQ 来监视各个队列; 请参阅 ALTER QUEUE 以获取此选项。

CPU (平台中央处理器)

介绍

其中统计信息引用 当前时间间隔，这是 MQIAMO64_MONITOR_INTERVAL 参数在发布的消息中定义的时间间隔。

统计信息通常每 10 秒发布一次，即 发布时间间隔，只要至少有一个活动订户，但应该始终从消息中获取精确时间间隔。

要点: 除非另有指定，否则度量值是捕获时时间点的绝对值。

SystemSummary (CPU 性能-整个平台)

用户 CPU 时间百分比 *X*%

CPU 在非特权代码中使用的平均时间百分比 (过去 10 秒时间间隔内耗用的时间)。

系统 CPU 时间百分比 *X*%

CPU 在特权代码中使用的平均时间百分比 (过去 10 秒时间间隔内)。

CPU 负载-平均 1 分钟 *X*

1 分钟的平均负载。 "负载平均值" 是整个行业的术语，但报告的准确值可能因平台而异。

CPU 负载-平均 5 分钟 *X*

5 分钟负载平均值。 "负载平均值" 是整个行业的术语，但报告的准确值可能因平台而异。

CPU 负载-平均 15 分钟 *X*

15 分钟的平均负载。 "负载平均值" 是整个行业的术语，但报告的准确值可能因平台而异。

CPU 系统摘要

RAM 可用百分比 *X*%

RAM 总字节数 *X*MB

RAM 可用百分比 *X*%

RAM 总字节数 *X*MB

QMgrSummary (CPU 性能-正在运行的队列管理器)

用户 CPU 时间-队列管理器 *X*% 的估算百分比

当此队列管理器的进程使用非特权代码时， CPU 使用的平均时间百分比 (过去 10 秒时间间隔)。

队列管理器 *X* 的系统 CPU 时间百分比估算值

当此队列管理器的进程处于特权代码中时， CPU 使用的平均时间百分比 (过去 10 秒时间间隔内)。

RAM 总字节数-队列管理器 *X*MB 的估算值

这是队列管理器使用的内存的近似值。

DISK (平台持久数据存储器)

SystemSummary 和 *QMgrSummary* 是捕获时的绝对值。 请参阅 简介 以获取 发布时间间隔的详细信息。

SystemSummary (磁盘使用率-平台范围)

MQ 错误文件系统-正在使用的字节数 *X*MB

MQ 错误文件系统可用空间 *X*%

MQ FDC 文件计数 *X*

MQ 跟踪文件系统-正在使用的字节数 *X*MB

MQ 跟踪文件系统可用空间 *X*%

QMgrSummary (磁盘使用情况-正在运行的队列管理器)

队列管理器文件系统-正在使用的字节数 *X*MB

队列管理器文件系统可用空间 *X*%

日志 (磁盘使用情况-队列管理器恢复日志)

日志-正在使用的字节数 *X*

最大日志字节数 *X*

当所有主扩展数据块和辅助扩展数据块已满时，可以写入日志的最大字节数。 这小于日志文件系统的大小

日志文件系统-正在使用的字节数 *X*

日志文件系统-最大字节数 *X*

为当前时间间隔 *X* 写入的日志物理字节数。

请参阅 简介 以获取 当前时间间隔的定义。

针对 当前时间间隔 *X* 写入的日志逻辑字节数

日志写入等待时间 *X* uSec

表示单次写入磁盘所用时间的滚动平均值。

其中 **LogWriteIntegrity**=*TripleWrite*，写入磁盘的物理字节数大于写入的逻辑字节数。

日志写入大小 *X*，也是滚动平均值。

日志被等待归档的扩展数据块占用 *X*。

仅当 **logtype**= `linear` 且 **LogManagement** = `archive` 时才发布。 请参阅 qm.ini 文件的日志节 以获取更多信息。

介质恢复所需的日志空间 (MB) *X*。

仅当 **logtype**= linear 时才发布。

可复用扩展数据块占用的日志空间 (MB) *X*

仅当 **logtype**= linear 且 **LogManagement** = automatic 时才发布。 请参阅 qm.ini 文件的日志节 以获取更多信息。

日志-当前正在使用的主空间 *X*%。

正在使用的日志文件空间占主日志的百分比。 此值可以大于 100%。

日志-工作负载主空间利用率 *X*%。

正在使用的日志文件空间占最近历史记录的滚动平均值的百分比。

STATMQI (API 使用情况统计信息)

所有 API 使用情况统计信息都反映了已发布时间间隔的出现次数和/或百分比。 请参阅 简介 以获取 已发布时间间隔的定义。

统计信息输出失败的 MQI 调用数的计数，但并非每个失败的 MQI 调用都出现在这些统计信息中-事实上，并非每个 MQI 调用的失败都会记录其统计信息。 这是因为在 MQI 调用到达记录了统计信息的队列管理器内部之前，会诊断 MQI 调用失败的许多原因。

此示例包括返回到客户机应用程序的 MQRC_HCONN_ERROR。 如果客户机应用程序传递了错误的 **hconn**，那么 MQ 客户机会诊断该错误并返回 MQRC_HCONN_ERROR，而不将 MQI 调用传递到队列管理器。 因此，失败的 MQI 调用从不出现在队列管理器记录的统计信息中。

失败的 MQI 调用的统计信息很有用，因为它们使客户能够对编写不佳的应用程序进行故障诊断，这些应用程序会生成不必要的失败的 MQI 调用，从而影响性能。 统计信息中记录的各种 MQI 调用的失败原因的一些示例:

- MQCONN/MQCONNX/MQOPEN 在由队列管理器而不是客户机诊断时返回 2035 MQRC_NOT_AUTHORIZED。 例如，以无人身份运行 **amqsput** 。
- MQPUT/MQPUT1 返回 2053 MQRC_Q_FULL，因为已超过 MAXDEPTH。
- MQGET 在浏览或破坏性地从空队列获取时返回 2033 MQRC_NO_MSG_AVAILABLE
- MQSUBRQ 返回 2437 MQRC_NO_RETAINED_MSG，因为没有保留消息

CONNDISC (MQCONN 和 MQDISC)

MQCONN/MQCONNX 计数 *X*

失败的 MQCONN/MQCONNX 计数 *X*

并发连接-高水位标记 *X*

MQDISC 计数 *X*

OPENCLOSE (MQOPEN 和 MQCLOSE)

MQOPEN 计数 *X Y*/秒

失败的 MQOPEN 计数 *X*

MQCLOSE 计数 *X Y*/秒

失败的 MQCLOSE 计数 *X*

INQSET (MQINQ 和 MQSET)

MQINQ 计数 *X*

失败的 MQINQ 计数 *X*

MQSET 计数 *X*

失败的 MQSET 计数 *X*

PUT (MQPUT)

时间间隔总计 MQPUT/MQPUT1 计数 *X*

时间间隔总计 MQPUT/MQPUT1 字节计数 *X Y*/秒

非持久消息 MQPUT 计数 *X*

持久消息 MQPUT 计数 *X*

失败的 MQPUT 计数 *X*

非持久消息 MQPUT1 计数 *X*

持久消息 MQPUT1 计数 *X*

失败 MQPUT1 计数 *X*

放入非持久消息-字节计数 *X Y*/秒

放入持久消息-字节计数 *X*

MQSTAT 计数 *X*

GET (MQGET)

时间间隔总破坏性获取计数 *X*

时间间隔总破坏性获取字节计数 *X Y*/秒

非持久消息破坏性获取计数 *X*

持久消息破坏性获取计数 *X*

失败的 MQGET-计数 *X*

已获取非持久消息-字节计数 *X Y*/秒

获取持久消息-字节计数 *X*

非持久消息浏览计数 *X*

持久消息浏览-计数 *X*

浏览失败计数 *X*

非持久消息浏览-字节计数 *X Y*/秒

持久消息浏览-字节计数 *X*

到期消息计数 *X*

清除的队列计数 *X*

MQCB 计数 *X*

失败的 MQCB 计数 *X*

MQCTL 计数 *X*

SYNCPOINT (落实和回滚)

落实计数 *X*

回滚计数 *X*

SUBSCRIBE (预订)

创建持久预订计数 *X*

变更持久预订计数 *X*

恢复持久预订计数 *X*

创建非持久预订计数 *X*

创建/更改/恢复预订计数 *X* 失败

删除持久预订计数 *X*

删除非持久预订计数 *X*

预订删除失败计数 *X*

MQSUBRQ 计数 *X*

失败的 MQSUBRQ 计数 *X*

持久订户-高水位标记 *X*

持久订户-低水位标记 *X*

非持久订户-高水位标记 *X*

非持久订户-低水位标记 *X*

PUBLISH (发布)

主题 MQPUT/MQPUT1 时间间隔总计 *X*

放入 *X Y*/秒的时间间隔总主题字节数

已发布到订户-消息计数 *X*

已发布到订户-字节计数 *X*

非持久主题 MQPUT/MQPUT1 计数 *X*

持久-主题 MQPUT/MQPUT1 计数 *X*

失败主题 MQPUT/MQPUT1 计数 *X*

STATQ (每个队列的 API 使用情况统计信息)

一般 (一般)

消息已到期 *X* (从 GET 移至 IBM MQ 9.3.0 和更高版本 CD)

队列清除计数 *X* (从 IBM MQ 9.3.0 和更高版本 CD 的 GET 移动)

平均排队时间 *X* uSec (从 GET 移动到 IBM MQ 9.3.0 和更高版本 CD)

队列深度 *X* (从 IBM MQ 9.3.0 和更高版本 CD 的 GET 移动)

OPENCLOSE (MQOPEN 和 MQCLOSE)

MQOPEN 计数 *X*

MQCLOSE 计数 *X*

INQSET (MQINQ 和 MQSET)

MQINQ 计数 *X*

MQSET 计数 *X*

PUT (MQPUT 和 MQPUT1)

MQPUT/MQPUT1 计数 *X*

MQPUT 字节计数 *X*

MQPUT 非持久消息计数 *X*

MQPUT 持久消息计数 *X*

回滚 MQPUT 计数 *X*

MQPUT1 非持久消息计数 *X*

MQPUT1 持久消息计数 *X*

非持久字节计数 *X*

持久字节计数 *X*

锁定争用 *X*%

尝试锁定导致等待另一进程先释放锁定的队列的百分比。 减少锁定争用可能会增加系统的最大吞吐量，因为获取当前未锁定的锁定比等待释放锁定更有效。

队列避免放置 *X*%

如果在有正在等待的 getter 时将消息放入队列，那么可能不需要对消息进行排队，因为可以将消息立即传递到 getter。 所以这条消息据说是避开了队列， "避免的队列放入" 就是这类消息的计数。 增加队列避免可能会增加系统的最大吞吐量，因为这可避免将消息放入队列并使其再次离开的成本。

队列避免的字节数 *X*%

如果在有正在等待的 getter 时将消息放入队列，那么可能不需要对消息进行排队，因为可以将消息立即传递到 getter。 所以这个消息说是避开了队列， "队列避免字节" 就是这样的字节的计数。 增加队列避免可能会增加系统的最大吞吐量，因为这可避免将消息放入队列并使其再次离开的成本。

GET (MQGET)

MQGET 计数 *X*

MQGET 字节计数 *X*

破坏性 MQGET 非持久消息计数 *X*
破坏性 MQGET 持久消息计数 *X*
回滚 MQGET 计数 *X*
破坏性 MQGET 非持久字节计数 *X*
破坏性 MQGET 持久字节计数 *X*
MQGET 浏览非持久消息计数 *X*
MQGET 浏览持久消息计数 *X*
MQGET 浏览非持久字节计数 *X*
MQGET 浏览持久字节计数 *X*
消息已到期 *X* (已从 IBM MQ 9.3 移至 GENERAL)
队列清除计数 *X* (已从 IBM MQ 9.3 移至 常规)
平均排队时间 *X* uSec (从 IBM MQ 9.3 移至 GENERAL)
队列深度 *X* (已从 IBM MQ 9.3 移至 GENERAL)
破坏性 MQGET 失败 *X*
破坏性 MQGET 失败，返回 MQRC_NO_MSG_AVAILABLE *X*
破坏性 MQGET 失败，带有 MQRC_TRUNCATED_MSG_FAILED *X*
MQGET 浏览失败 *X*
MQGET 浏览失败，返回 MQRC_NO_MSG_AVAILABLE *X*
MQGET 浏览失败，返回 MQRC_TRUNCATED_MSG_FAILED *X*

STATAPP (每个应用程序的使用情况统计信息)

INSTANCE (实例统计信息)

实例计数 *X* 绝对
可移动实例计数 *X* 绝对
实例短缺计数 *X* 绝对
实例已启动 *X* 时间间隔
已启动出站实例移动 *X* 时间间隔
已完成出站实例移动 *X* 时间间隔
实例在重新连接 *X* 时间间隔 期间结束
实例已结束 *X* 时间间隔

NHAREPLICA (每个实例的本机 HA 统计信息)

REPLICATION (复制统计信息)

发送的同步日志字节数 *X*
已发送的捕获日志字节数 *X*
V 9.4.0 发送的同步压缩日志字节数 *X*
V 9.4.0 已发送的捕获压缩日志字节数 *X*
V 9.4.0 发送的同步未压缩日志字节数 *X*
V 9.4.0 已发送的捕获未压缩日志字节数 *X*
日志写入平均应答等待时间 *X uSec*
日志写入平均应答大小 *X*
积压字节数 *X*
积压平均字节数 *X*

相关信息
使用 IBM MQ Operator 时发布的度量

实时监视

实时监视是一种技术，允许您确定队列管理器中队列和通道的当前状态。 在发出命令时，返回的信息是准确的。

提供了许多命令，当发出这些命令时，这些命令将返回有关队列和通道的实时信息。 可以针对一个或多个队列或通道返回信息，并且数量可能有所不同。 可以在以下任务中使用实时监视:

- 帮助系统管理员了解其 IBM MQ 系统的稳定状态。 这有助于在系统中发生问题时进行问题诊断。
- 随时确定队列管理器的情况，即使未检测到特定事件或问题也是如此。
- 帮助确定系统中问题的原因。

通过实时监视，可以返回队列或通道的信息。 返回的实时信息量由队列管理器，队列和通道属性控制。

- 您可以通过发出命令来监视队列，以确保正确处理队列。 必须先启用某些队列属性以进行实时监视，然后才能使用这些属性。
- 通过发出命令来监视通道，以确保通道正常运行。 必须先启用某些通道属性以进行实时监视，然后才能使用这些属性。

队列和通道的实时监视是对性能和通道事件监视的补充，也是与之分开的。

用于控制实时监视的属性

如果启用了实时监视，那么某些队列和通道状态属性将保存监视信息。 如果未启用实时监视，那么不会在这些监视属性中保存任何监视信息。 示例演示如何使用这些队列和通道状态属性。

您可以对各个队列或通道启用或禁用实时监视，也可以对多个队列或通道启用或禁用实时监视。 要控制个别队列或通道，请设置队列属性 MONQ 或通道属性 MONCHL ，以启用或禁用实时监视。 要同时控制多个队列或通道，请使用队列管理器属性 MONQ 和 MONCHL 在队列管理器级别启用或禁用实时监视。 对于具有使用缺省值 QMGR 指定的监视属性的所有队列和通道对象，将在队列管理器级别控制实时监视。

自动定义的集群发送方通道不是 IBM MQ 对象，因此不具有与通道对象相同的属性。 要控制自动定义的集群发送方通道，请使用队列管理器属性 MONACLS。 此属性确定是启用还是禁用队列管理器中自动定义的集群发送方通道以进行通道监视。

对于通道的实时监视，可以将 MONCHL 属性设置为以下三个监视级别之一: 低，中或高。 您可以在对象级别或队列管理器级别设置监视级别。 级别的选择取决于您的系统。 收集监视数据可能需要一些在计算上相对昂贵的指令，例如获取系统时间。 为了降低实时监控的效果，中，低监控选项会定期测量数据的样本，而不是一直收集数据。 第 270 页的表 30 汇总了可用于对通道进行实时监视的监视级别:

表 *30:* 监视级别

级别	描述	用法
低	定期测量一小部分数据样本。	用于处理大量消息的对象。
中等	定期测量数据样本。	对于大多数对象。
高	定期测量所有数据。	对于每秒仅处理几条消息的对象，其中最新的信息很重要。

对于队列的实时监视，您可以将 MONQ 属性设置为三个监视级别之一 (低，中或高)。 但是，这些值之间没有区别。 这些值全部启用数据收集，但不影响样本的大小。

示例

以下示例演示如何设置必要的队列，通道和队列管理器属性以控制监视级别。 对于所有示例，启用监视时，队列和通道对象具有中等级别的监视。

1. 要对队列管理器级别的所有队列和通道同时启用队列和通道监视，请使用以下命令:

```
ALTER QMGR MONQ(MEDIUM) MONCHL(MEDIUM)
ALTER QL(Q1) MONQ(QMGR)
ALTER CHL(QM1.TO.QM2) CHLTYPE(SDR) MONCHL(QMGR)
```

2. 要对所有队列和通道 (本地队列 Q1 和发送方通道 QM1.TO.QM2 除外) 启用监视，请使用以下命令:

```
ALTER QMGR MONQ(MEDIUM) MONCHL(MEDIUM)
ALTER QL(Q1) MONQ(OFF)
ALTER CHL(QM1.TO.QM2) CHLTYPE(SDR) MONCHL(OFF)
```

3. 要对所有队列和通道 (本地队列 Q1 和发送方通道 QM1.TO.QM2 除外) 同时禁用队列和通道监视，请使用以下命令:

```
ALTER QMGR MONQ(OFF) MONCHL(OFF)
ALTER QL(Q1) MONQ(MEDIUM)
ALTER CHL(QM1.TO.QM2) CHLTYPE(SDR) MONCHL(MEDIUM)
```

4. 要对所有队列和通道禁用队列和通道监视，而不考虑各个对象属性，请使用以下命令:

```
ALTER QMGR MONQ(NONE) MONCHL(NONE)
```

5. 要控制自动定义的集群发送方通道的监视功能，请使用以下命令:

```
ALTER QMGR MONACLS(MEDIUM)
```

6. 要指定自动定义的集群发送方通道将使用队列管理器设置进行通道监视，请使用以下命令:

```
ALTER QMGR MONACLS(QMGR)
```

相关概念

第 270 页的『实时监视』
实时监视是一种技术，允许您确定队列管理器中队列和通道的当前状态。 在发出命令时，返回的信息是准确的。

第 287 页的『Using IBM MQ online monitoring』
You can collect monitoring data for queues and channels (including automatically defined cluster-server channels) by setting the MONQ, MONCHL, and MONACLS attributes.

相关任务

第 271 页的『显示队列和通道监视数据』
要显示队列或通道的实时监视信息，请使用 IBM MQ Explorer 或相应的 MQSC 命令。 某些监视字段显示以逗号分隔的指示符值对，这有助于您监视队列管理器的操作。 示例演示如何显示监视数据。

显示和改变队列管理器属性

监视 (MONCHL)

显示队列和通道监视数据

要显示队列或通道的实时监视信息，请使用 IBM MQ Explorer 或相应的 MQSC 命令。 某些监视字段显示以逗号分隔的指示符值对，这有助于您监视队列管理器的操作。 示例演示如何显示监视数据。

关于此任务

显示以逗号分隔的一对值的监视字段提供自对对象启用监视以来或从启动队列管理器时开始测量的时间的短期和长期指示符:

- 短期指标是该对中的第一个值，并以这样的方式计算，即给予更多最近的测量以更高的权重，并将对该值产生更大的影响。 这表明最近测量的趋势。

- 在对中的第二个值中的长期指标，并以这样的方式计算，使得最近的测量没有得到如此高的权重。 这指示资源性能的长期活动。

这些指示符值对于检测队列管理器操作中的更改最有用。 这需要了解这些指标在正常使用时所显示的时间，以便发现这些时间的增加。 通过定期收集和检查这些值，可以检测队列管理器操作中的波动。 这可能指示性能发生了更改。

获取实时监控信息，如下所示:

过程

1. 要显示队列的实时监视信息，请使用 IBM MQ Explorer 或 MQSC 命令 DISPLAY QSTATUS 并指定可选参数 MONITOR。
2. 要显示通道的实时监视信息，请使用 IBM MQ Explorer 或 MQSC 命令 DISPLAY CHSTATUS 并指定可选参数 MONITOR。

示例

队列 Q1 将属性 MONQ 设置为缺省值 QMGR，而拥有该队列的队列管理器将属性 MONQ 设置为 MEDIUM。 要显示为此队列收集的监视字段，请使用以下命令:

```
DISPLAY QSTATUS(Q1) MONITOR
```

队列 Q1 的监视字段和监视级别如下所示:

```
QSTATUS(Q1)
TYPE(QUEUE)
MONQ(MEDIUM)
QTIME(11892157,24052785)
MSGAGE(37)
LPUTDATE(2005-03-02)
LPUTTIME(09.52.13)
LGETDATE(2005-03-02)
LGETTIME(09.51.02)
```

发送方通道 QM1.TO.QM2 将 MONCHL 属性设置为缺省值 QMGR，而拥有该队列的队列管理器将 MONCHL 属性设置为 MEDIUM。 要显示为此发送方通道收集的监视字段，请使用以下命令:

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

发送方通道 QM1.TO.QM2 的监视字段和监视级别如下所示:

```
CHSTATUS(QM1.TO.QM2)
XMITQ(Q1)
CONNAME(127.0.0.1)
CURRENT
CHLTYPE(SDR)
STATUS(RUNNING)
SUBSTATE(MQGET)
MONCHL(MEDIUM)
XQTIME(755394737,755199260)
NETTIME(13372,13372)
EXITTIME(0,0)
XBATCHSZ(50,50)
COMPTIME(0,0)
STOPREQ(NO)
RQMNAME(QM2)
```

相关概念

第 270 页的『实时监视』
实时监视是一种技术，允许您确定队列管理器中队列和通道的当前状态。 在发出命令时，返回的信息是准确的。

相关参考
显示队列状态

监视队列

使用此页面来查看可帮助您解决队列问题的任务以及为该队列提供服务的应用程序。 提供了各种监视选项来确定问题

通常，正在处理的队列问题的第一个标志是队列 (CURDEPTH) 上的消息数增加。 如果您期望在一天中的特定时间或在特定工作负载下增加，那么越来越多的消息可能不会指示问题。 但是，如果您对不断增加的消息数没有解释，那么可能要调查原因。

您可能具有应用程序队列 (其中应用程序存在问题) 或传输队列 (其中通道存在问题)。 当为队列提供服务的应用程序是通道时，提供了其他监视选项。

以下示例调查名为 Q1 的特定队列的问题，并描述在各种命令的输出中查看的字段:

确定应用程序是否已打开队列

如果队列有问题，请检查应用程序是否已打开该队列

关于此任务

执行以下步骤以确定应用程序是否已打开队列:

过程

1. 确保针对队列运行的应用程序是您期望的应用程序。 对有问题的队列发出以下命令:

   ```
   DISPLAY QSTATUS(Q1) TYPE(HANDLE) ALL
   ```

   在输出中，查看 APPLTAG 字段，并检查是否显示了应用程序的名称。 如果未显示应用程序的名称，或者如果根本没有输出，请启动应用程序。
2. 如果队列是传输队列，请在 CHANNEL 字段中查看输出。

   如果通道名称未显示在 CHANNEL 字段中，请确定通道是否正在运行。
3. 确保针对队列运行的应用程序打开了队列以进行输入。 发出以下命令:

   ```
   DISPLAY QSTATUS(Q1) TYPE(QUEUE) ALL
   ```

   在输出中，查看 IPPROCS 字段以查看是否有任何应用程序打开了队列以进行输入。 如果值为 0 ，并且这是用户应用程序队列，请确保应用程序打开队列以进行输入，从而将消息从队列中取出。

检查队列上的消息是否可用

如果队列中有大量消息，并且应用程序未处理其中任何消息，请检查队列中的消息是否可供应用程序使用

关于此任务

执行以下步骤以调查应用程序未处理来自队列的消息的原因:

过程

1. 确保应用程序在处理队列中的所有消息时，不会要求提供特定消息标识或相关标识。
2. 虽然队列的当前深度可能显示队列中的消息数越来越多，但队列中的某些消息可能无法由应用程序获取，因为它们未落实; 当前深度包括队列中未落实的 MQPUT 消息数。 发出以下命令:

   ```
   DISPLAY QSTATUS(Q1) TYPE(QUEUE) ALL
   ```

   在输出中，查看 UNCOM 字段以查看队列上是否有任何未落实的消息。

3. 如果应用程序尝试从队列中获取任何消息，请检查放入应用程序是否正确落实了这些消息。 发出以下命令以查找将消息放入此队列的应用程序的名称:

```
DISPLAY QSTATUS(Q1) TYPE(HANDLE) OPENTYPE(OUTPUT)
```

4. 然后发出以下命令，在 *appltag* 中插入来自先前命令的输出的 APPLTAG 值:

```
DISPLAY CONN(*) WHERE(APPLTAG EQ appltag) UOWSTDA UOWSTTI
```

这将显示启动工作单元的时间，并将帮助您发现应用程序是否正在创建长时间运行的工作单元。 如果放置应用程序是通道，那么您可能要调查批处理需要很长时间才能完成的原因。

检查应用程序是否正在从队列中获取消息

如果队列和服务该队列的应用程序存在问题，请检查应用程序是否正在从队列中获取消息

关于此任务

要检查应用程序是否正在从队列中获取消息，请执行以下检查:

过程

1. 确保对该队列运行的应用程序实际上正在处理来自该队列的消息。 发出以下命令:

```
DISPLAY QSTATUS(Q1) TYPE(QUEUE) ALL
```

在输出中，查看 LGETDATE 和 LGETTIME 字段，这些字段显示从队列完成最后一次获取的时间。

2. 如果上次从该队列获取的时间比预期的长，请确保应用程序正确处理消息。

如果应用程序是通道，请检查消息是否正在通过该通道移动

确定应用程序是否可以足够快地处理消息

如果消息正在队列上构建，但您的其他检查未发现任何处理问题，请检查应用程序是否可以足够快地处理消息。 如果应用程序是通道，请检查通道是否可以足够快地处理消息。

关于此任务

要确定应用程序是否足够快地处理消息，请执行以下测试:

过程

1. 定期发出以下命令以收集有关队列的性能数据:

```
DISPLAY QSTATUS(Q1) TYPE(QUEUE) ALL
```

如果 QTIME 指示符中的值很高，或者在一段时间内正在增加，并且您已通过检查队列上的消息是否可用来排除长时间运行工作单元的可能性，那么获取应用程序可能跟不上放置应用程序。

2. 如果获取应用程序无法跟上放置应用程序，请考虑添加另一个获取应用程序以处理队列。

是否可以添加另一个获取应用程序取决于应用程序的设计以及队列是否可以由多个应用程序共享。 诸如消息分组或按相关标识获取之类的功能可能有助于确保两个应用程序可以同时处理一个队列。

在当前深度未增加时检查队列

即使队列的当前深度未增加，监视队列以检查应用程序是否正确处理消息仍可能有用。

关于此任务

要收集有关队列的性能数据，请定期发出以下命令:

过程

定期发出以下命令:

```
DISPLAY QSTATUS(Q1) TYPE(QUEUE) MSGAGE QTIME
```

在输出中，如果 MSGAGE 中的值在一段时间内增加，并且您的应用程序旨在处理所有消息，那么这可能指示根本未处理某些消息。

监视通道

使用此页面来查看可帮助您解决传输队列问题以及服务该队列的通道问题的任务。 提供了各种通道监视选项来确定问题。

通常，正在处理的队列问题的第一个标志是队列 (CURDEPTH) 上的消息数增加。 如果您期望在一天中的特定时间或在特定工作负载下增加，那么越来越多的消息可能不会指示问题。 但是，如果您对不断增加的消息数没有解释，那么可能要调查原因。

为传输队列提供服务的通道可能存在问题。 提供了各种通道监视选项来帮助您确定问题。

以下示例调查名为 QM2 的传输队列和名为 QM1.TO.QM2。 此通道用于将消息从队列管理器 QM1 发送到队列管理器 QM2。 队列管理器 QM1 上的通道定义是发送方或服务器通道，而队列管理器 QM2 上的通道定义是接收方或请求者通道。

确定通道是否正在运行

如果传输队列存在问题，请检查通道是否正在运行。

关于此任务

执行以下步骤以检查为传输队列提供服务的通道的状态:

过程

1. 发出以下命令以了解您期望处理传输队列 QM2:

   ```
   DIS CHANNEL(*) WHERE(XMITQ EQ QM2)
   ```

   在此示例中，此命令的输出显示为传输队列提供服务的通道是 QM1.TO.QM2
2. 发出以下命令以确定通道 QM1.TO.QM2:

   ```
   DIS CHSTATUS(QM1.TO.QM2) ALL
   ```
3. 检查 **CHSTATUS** 命令输出的 STATUS 字段:
   - 如果 STATUS 字段的值为 RUNNING，请检查通道是否正在移动消息
   - 如果该命令的输出未显示任何状态，或者 STATUS 字段的值为 STOPPED， RETRY， BINDING 或 REQUESTING，请执行相应的步骤，如下所示:
4. 可选： 如果 STATUS 字段的值未显示任何状态，那么通道处于不活动状态，因此请执行以下步骤:
   a) 如果通道应该已由触发器自动启动，请检查传输队列上的消息是否可用。
   如果传输队列上有可用消息，请检查传输队列上的触发器设置是否正确。
   b) 发出以下命令以手动再次启动通道:

   ```
   START CHANNEL(QM1.TO.QM2)
   ```
5. 可选： 如果 STATUS 字段的值为 STOPPED，请执行以下步骤:
   a) 请检查错误日志以确定通道停止的原因。 如果通道由于错误而停止，请更正问题。
   另请确保通道具有为重试属性指定的值: *SHORTRTY* 和 *LONGRTY*。 如果发生瞬态故障 (例如网络错误) ，那么通道将尝试自动重新启动。

b) 发出以下命令以手动再次启动通道:

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

z/OS 在 IBM MQ for z/OS 上，您可以使用命令事件消息来检测用户何时停止通道。

6. 可选： 如果 STATUS 字段的值为 RETRY，请执行以下步骤:
   a) 请检查错误日志以识别错误，然后更正问题。
   b) 发出以下命令以手动再次启动通道:

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

   或 等待通道在下一次重试时成功连接。
7. 可选： 如果 STATUS 字段的值为 BINDING 或 REQUESTING，那么通道尚未成功连接到合作伙伴。 执行以下步骤:
   a) 在通道两端发出以下命令以确定通道的子状态:

```
DIS CHSTATUS(QM1.TO.QM2) ALL
```

   注:

   i) 在某些情况下，可能仅在通道的一端存在子状态。

   ii) 许多子状态是暂时性的，因此发出命令几次以检测通道是否卡在特定子状态。

   b) 检查 第 276 页的表 31 以确定要执行的操作:

表 *31:* 通过状态绑定或请求看到的子状态

启动 MCA 子状态 [1]	响应 MCA 子状态 [2]	注意
名称服务器		正在启动的 MCA 正在等待名称服务器请求完成。 确保在通道属性 CONNAME 中指定了正确的主机名，并且正确设置了名称服务器。
SCYEXIT	SCYEXIT	MCA 当前通过安全出口 在对话 中。 有关更多信息，请参阅 第 278 页的『确定通道是否可以足够快地处理消息』。
	CHADEXIT	通道自动定义出口当前正在执行。 有关更多信息，请参阅 第 278 页的『确定通道是否可以足够快地处理消息』。
RCVEXIT SENDEXIT MSGEXIT MREXIT	RCVEXIT SENDEXIT MSGEXIT MREXIT	在 MQXR_INIT 的通道启动时调用出口。 如果这需要很长时间，请查看出口的此部分中的处理。 有关更多信息，请参阅 第 278 页的『确定通道是否可以足够快地处理消息』。
序列化	序列化	此子状态仅适用于处置为 SHARED 的通道。
网络连接		如果由于网络配置不正确而导致连接延迟，那么将显示此子状态。
SSL 握手	SSL 握手	TLS 握手由多个发送和接收组成。 如果网络时间很慢，或者与查找 CRL 的连接很慢，那么这会影响执行握手所花费的时间。 z/OS 在 IBM MQ for z/OS 上，此子状态也可能表示没有足够的 SSLTASKS。

注意:

i) 启动 MCA 是启动对话的通道的结束。 这可以是发件人，集群发件人，标准服务器和请求者。 在服务器/请求者对中，它是从中启动通道的结束。

ii) 响应的 MCA 是响应启动对话的请求的通道的结束。 这可以是接收方，集群接收方，请求者 (当服务器或发送方启动时)，服务器 (当请求者启动时) 和发送方 (在请求者-发送方回调对通道中)。

正在检查通道是否正在移动消息

如果传输队列有问题，请检查通道是否正在移动消息

开始之前

发出命令 DIS CHSTATUS(QM1.TO.QM2) ALL。 如果 STATUS 字段的值为 RUNNING，那么通道已成功连接到伙伴系统。

检查传输队列上是否没有未落实的消息，如 第 273 页的『检查队列上的消息是否可用』中所述。

关于此任务

如果有消息可供通道获取和发送，请执行以下检查:

过程

1. 在显示通道状态命令 DIS CHSTATUS(QM1.TO.QM2) ALL 的输出中，查看以下字段:

   MSGS
   该会话期间（自启动通道以来）发送或接收的消息数（或者，对于服务器连接通道，处理 MQI 调用的数量）。

   BUFSSENT
   发送的传输缓冲区的数量。 这包括仅发送控制信息的传输。

   BYTSSENT
   该会话期间（自启动通道以来）发送的字节数。 这包括由消息通道代理程序发送的控制信息。

   LSTMSGDA
   发送最后一条消息或处理 MQI 调用的日期，请参阅 LSTMSGTI。

   LSTMSGTI
   发送最后一条消息或处理 MQI 调用的时间。 对于发送方或服务器，它是发送上一个消息（如果将其分割，那么是它的最后一部分）的时间。 对于请求方或接收方，它是将上一个消息放到其目标队列的时间。 对于服务器连接通道，它是完成上一个 MQI 调用的时间。

   CURMSGS
   对于发送通道，它是当前批次中已发送的消息数。 对于接收通道，它是当前批次中已接收的消息数。 在落实此批次时，发送通道和接收通道的这个值都复位为零。

   状态
   这是通道的状态，可以是 Starting、Binding、Initializing、Running、Stopping、Retrying、Paused、Stopped 或 Requesting。

   SUBSTATE
   通道当前正在执行的操作。

   INDOUBT
   当前通道是否处于不确定状态。 这仅是 YES，发送消息通道代理程序正在等待已成功收到其已发送的一批消息的确认。 在其他情况下，它都是“否”，包括在消息发送期间，但必须在请求应答前。 对于接收通道，值始终为 NO。

2. 确定通道自启动以来是否已发送任何消息。 如果已发送任何消息，请确定发送最后一条消息的时间。
3. 通道可能已启动尚未完成的批处理，如 CURMSGS 中的非零值所指示。 如果 INDOUBT 为 YES，那么通道正在等待接收通道另一端接收到批处理的应答。 查看输出中的 SUBSTATE 字段，并参阅 第 277 页的表 32:

表 *32:* 发送方和接收方 *MCA* 子状态

发送方子状态	接收方 SUBSTATE	注意
MQGET	接收	静态通道的正常状态。

表 32: 发送方和接收方 *MCA* 子状态 (继续)		
发送方子状态	**接收方 SUBSTATE**	**注意**
发送	接收	SEND 通常是一种暂时性状态。 如果看到 SEND，那么表示通信协议缓冲区已填充。 这可能指示网络问题。
接收		如果发送方在任何时间长度内处于 RECEIVE 子状态，那么它正在等待对批处理完成或脉动信号的响应。 您可能想要检查完成批处理需要很长时间的原因。

注: 您可能还希望确定通道是否可以足够快地处理消息，尤其是当通道具有与出口处理相关联的子状态时。

检查批处理需要很长时间才能完成的原因

批处理可能需要很长时间才能完成包含慢速网络或通道正在使用消息重试处理的原因。

关于此任务

当发送方通道发送了一批消息时，它将等待来自接收方的该批消息的确认，除非该通道已由管道传送。 此任务中描述的因素可能会影响发送方通道等待的时间长度。

过程

- 检查网络是否缓慢。

  NETTIME 值是将批处理结束请求发送到通道的远程端并接收响应所花费的时间量 (以微秒为单位)，减去处理批处理结束请求所花费的时间。 由于以下任一原因，此值可能较大:

  - 网络很慢。 网络速度慢会影响完成批处理所需的时间。 生成 NETTIME 字段的指示符的测量在批处理结束时进行测量。 但是，受网络减速影响的第一批没有用 NETTIME 值的变化来表示，因为它是在批处理结束时测量的。
  - 请求在远程端排队，例如通道可以重试放置，或者由于页集 I/O，放置请求可能很慢。 完成任何排队的请求后，将测量批处理请求结束的持续时间。 因此，如果您获得较大的 NETTIME 值，请在远程端检查异常处理。
- 检查通道是否正在使用消息重试。

  如果接收方通道未能将消息放入目标队列，那么它可能会使用消息重试处理，而不是立即将消息放入死信队列。 重试处理可能会导致批处理变慢。 在两次 MQPUT 尝试之间，通道将具有 STATUS (PAUSED)，指示它正在等待消息重试时间间隔过去。

确定通道是否可以足够快地处理消息

如果在传输队列上正在构建消息，但您未发现任何处理问题，请确定通道是否可以足够快地处理消息。

开始之前

在一段时间内重复发出以下命令以收集有关通道的性能数据:

```
DIS CHSTATUS(QM1.TO.QM2) ALL
```

关于此任务

确认传输队列上没有未落实的消息，如 第 273 页的『检查队列上的消息是否可用』中所述，然后检查显示通道状态命令的输出中的 XQTIME 字段。 当 XQTIME 指标的值持续较高或在测量周期内增加时，指示通道未与放置应用程序保持同步。

执行以下测试:

过程

1. 检查出口是否正在处理。

如果在传递这些消息的通道上使用出口，那么这些出口可能会增加处理消息所耗用的时间。 要确定是否存在这种情况，请执行以下检查:

a) 在命令 DIS CHSTATUS(QM1.TO.QM2) ALL 的输出中，检查 EXITTIME 字段。

如果在出口中花费的时间高于预期，请复查出口中的处理是否存在任何不必要的循环或额外处理，尤其是在消息，发送和接收出口中。 此类处理会影响在通道中移动的所有消息。

b) 在命令 DIS CHSTATUS(QM1.TO.QM2) ALL 的输出中，检查 SUBSTATE 字段。

如果通道在相当长的时间内具有下列其中一个子状态，请查看出口中的处理:

- SCYEXIT
- RCVEXIT
- SENDEXIT
- MSGEXIT
- MREXIT

有关通道子状态的更多信息，请参阅表 第 277 页的表 32。

2. 检查网络是否缓慢。

如果消息在通道中移动速度不够快，可能是因为网络速度很慢。 要确定是否存在这种情况，请执行以下检查:

a) 在命令 DIS CHSTATUS(QM1.TO.QM2) ALL 的输出中，检查 NETTIME 字段。

这些指示符是在发送通道向其合作伙伴请求响应时测量的。 这会在每个批处理结束时发生，并且在脉动信号期间通道处于空闲状态时发生。

b) 如果此指标显示往返所需时间超过预期，请使用其他网络监视工具来调查网络的性能。

3. 检查通道是否正在使用压缩。

如果通道正在使用压缩，那么这将增加处理消息所耗用的时间。 如果通道仅使用一个压缩算法，请执行以下检查:

a) 在命令 DIS CHSTATUS(QM1.TO.QM2) ALL 的输出中，检查 COMPTIME 字段。

这些指示符显示压缩或解压期间所花费的时间。

b) 如果所选压缩未按预期数量减少要发送的数据量，请更改压缩算法。

4. 如果通道正在使用多个压缩算法，请执行以下检查:

a) 在命令 DIS CHSTATUS(QM1.TO.QM2) ALL 的输出中，检查 COMPTIME， COMPHDR 和 COMPMSG 字段。

b) 更改在通道定义上指定的压缩算法，或考虑编写消息出口以覆盖通道对特定消息的压缩算法选择 (如果压缩速率或算法选择未提供所需的压缩或性能)。

解决集群通道的问题

如果在 SYSTEM.CLUSTER.TRANSMIT.QUEUE 队列，诊断问题的第一步是发现哪些通道或通道在传递消息时迂到问题。

关于此任务

要发现使用 SYSTEM.CLUSTER.TRANSMIT.QUEUE 在传递消息时迂到问题。 执行下列检查:

过程

1. 发出以下命令:

```
DIS CHSTATUS(*) WHERE(XQMSGSA GT 1)
```

注: 如果您有一个忙碌的集群，其中有许多消息在移动，请考虑发出此命令并指定更高的数字，以消除只有几条消息可用于传递的通道。

2. 查看在 XQMSGSA 字段中具有较大值的一个或多个通道的输出。 确定通道未移动消息或移动消息速度不够快的原因。 使用 第 275 页的『监视通道』 中概述的任务来诊断发现导致构建的通道问题。

监视集群

在集群中，您可以监视应用程序消息，控制消息和日志。 在队列的两个或多个实例之间进行集群负载均衡时，存在特殊监视注意事项。

监视集群中的应用程序消息

通常，离开队列管理器的所有集群消息都通过 SYSTEM.CLUSTER.TRANSMIT.QUEUE 传递，而不考虑使用哪个集群发送方通道来传输消息。 每个通道都在与所有其他集群发送方通道并行地排出针对该通道的消息。 此队列上不断增加的消息可能指示一个或多个通道存在问题，必须对其进行调查:

- 必须针对集群设计相应地监视队列的深度。
- 以下命令将返回具有多个正在传输队列上等待的消息的所有通道:

```
DIS CHSTATUS(*) WHERE(XQMSGSA GT 1)
```

  对于单个队列上的所有集群消息，在开始填充时并不总是容易看到哪个通道存在问题。 使用此命令是查看哪个通道负责的简单方法。

您可以将集群队列管理器配置为具有多个传输队列。 如果将队列管理器属性 DEFCLXQ 更改为 CHANNEL，那么每个集群发送方通道都与不同的集群传输队列相关联。 或者，您可以手动配置单独的传输队列。 要显示与集群发送方通道关联的所有集群传输队列，请运行以下命令:

```
DISPLAY CLUSQMGR (qmgrName) XMITQ
```

定义集群传输队列，以便它们遵循左侧具有队列名称的固定主干的模式。 然后，可以使用通用队列名称来查询 **DISPLAY CLUSMGR** 命令返回的所有集群传输队列的深度:

```
DISPLAY QUEUE (qname *) CURDEPTH
```

监视集群中的控制消息

SYSTEM.CLUSTER.COMMAND.QUEUE 队列用于处理队列管理器的所有集群控制消息，这些消息由本地队列管理器生成或从集群中的其他队列管理器发送到此队列管理器。 当队列管理器正确维护其集群状态时，此队列趋向于零。 但是，在某些情况下，此队列上的消息深度可能会临时增大:

- 队列中有大量消息表示流失率处于集群状态。
- 进行重大更改时，允许队列在这些更改之间进行结算。 例如，移动存储库时，允许队列在移动第二个存储库之前达到零。

当此队列上存在积压的消息时，不会处理对集群状态或与集群相关的命令的更新。 如果长时间未从此队列中除去消息，那么需要进一步调查，最初是通过检查队列管理器错误日志 (或 z/OS 上的 CHINIT 日志) ，这可能说明导致此情况的进程。

SYSTEM.CLUSTER.REPOSITORY.QUEUE 将集群存储库高速缓存信息作为大量消息保存。 通常，消息会始终存在于此队列中，而对于更大的集群则会存在更多消息。 因此，此队列上的消息深度不是值得关注的问题。

监视日志

由于信息高速缓存和集群的分布式性质，在最初发生问题后的许多天 (甚至几个月) 内，集群中发生的问题可能不会向应用程序显示外部症状。 但是，原始问题通常在 IBM MQ 错误日志 (以及 z/OS 上的 CHINIT 日志) 中报告。 因此，主动监视这些日志以获取与集群相关的任何消息至关重要。 必须阅读和理解这些消息，并在必要时采取任何行动。

例如: 中断与集群中的队列管理器的通信可能会导致了解由于集群通过重新发布信息定期重新验证集群资源的方式而正在删除的某些集群资源。 消息 AMQ9465 或 z/OS 系统上的 CSQX465I 报告了可能发生的此类事件的警告。 此消息指示需要调查问题。

负载均衡的特殊注意事项

当集群在队列的两个或多个实例之间进行负载均衡时，使用应用程序必须处理每个实例上的消息。 如果一个或多个使用应用程序的应用程序终止或停止处理消息，那么集群可能会继续向这些队列实例发送消息。 在这种情况下，直到应用程序再次正常工作时，才会处理这些消息。 因此，对应用程序的监视是解决方案的重要组成部分，在这种情况下，必须采取行动来重新路由消息。 可在以下样本中找到用于自动执行此类监视的机制的示例: 集群队列监视样本程序 (AMQSCLM)。

相关概念

第 340 页的『调整分布式发布/预订网络』
使用此部分中的调整提示可帮助提高 IBM MQ 分布式发布/预订集群和层次结构的性能。

第 344 页的『在发布/预订网络中平衡生产者和消费者』
异步消息传递性能中的一个重要概念是 *balance*。 除非消息使用者与消息生产者平衡，否则积压的未使用消息可能会累积并严重影响多个应用程序的性能。

监视传输队列切换

监视集群发送方通道切换传输队列的过程非常重要，以便最大程度地降低对企业的影响。 例如，当工作负载较高或同时切换多个通道时，不应尝试此过程。

切换通道的过程

用于切换通道的过程为:

1. 通道打开新的传输队列以进行输入，并开始从该队列中获取消息 (使用按相关标识获取)
2. 队列管理器启动后台进程以将排队等待通道的任何消息从其旧传输队列移至其新传输队列。 在移动消息时，通道的任何新消息都将排队到旧传输队列以保留排序。 如果通道的旧传输队列上有大量消息，或者新消息正在快速到达，那么此过程可能需要一段时间才能完成。
3. 如果没有已落实或未落实的消息仍在其旧传输队列上排队等待通道，那么交换机已完成。 现在，新消息将直接放入新的传输队列中。

为了避免大量通道同时切换的事件， IBM MQ 提供了切换一个或多个未在运行的通道的传输队列的能力。 日期:

- IBM MQ for Multiplatforms 该命令称为 **runswchl**
- IBM MQ for z/OS CSQUTIL 实用程序可用于处理 SWITCH CHANNEL 命令

监视交换机操作的状态

要了解交换机操作的状态，管理员可以执行以下操作:

- 监视队列管理器错误日志 (AMQERR01.LOG) ，其中输出消息以指示操作期间的以下阶段:
  - 交换机操作已启动
  - 消息移动已启动
  - 定期更新要移动的消息数 (如果交换机操作未快速完成)
  - 消息移动已完成
  - 交换机操作已完成

  在 z/OS 上，这些消息输出到队列管理器作业日志，而不是通道启动程序作业日志，但如果在启动时启动交换机，那么单个消息由通道输出到通道启动程序作业日志。
- 使用 DISPLAY CLUSQMGR 命令来查询每个集群发送方通道当前正在使用的传输队列。
- 以查询方式运行 **runswchl** 命令 (或在 z/OS 上运行 CSQUTIL) 以确定一个或多个通道的切换状态。 此命令的输出标识每个通道的以下内容:
  - 通道是否具有暂挂的交换机操作
  - 通道从哪个传输队列切换到哪个传输队列
  - 旧传输队列上保留的消息数

每个命令都非常有用，因为在一次调用中，您可以确定每个通道的状态，配置更改所产生的影响以及所有交换机操作是否已完成。

可能发生的潜在问题

请参阅 切换传输队列时的潜在问题 ，以获取在切换传输队列时可能迁到的一些问题及其原因和最可能的解决方案的列表。

相关概念

第 340 页的『调整分布式发布/预订网络』
使用此部分中的调整提示可帮助提高 IBM MQ 分布式发布/预订集群和层次结构的性能。

第 344 页的『在发布/预订网络中平衡生产者和消费者』
异步消息传递性能中的一个重要概念是 *balance*。 除非消息使用者与消息生产者平衡，否则积压的未使用消息可能会累积并严重影响多个应用程序的性能。

Multi 监视应用程序均衡

您可以使用 **DISPLAY APSTATUS** 命令来监视跨统一集群的应用程序均衡状态，并调查应用程序不均衡的原因 (如果这是意外情况)。

在集群中的队列管理器之间监视应用程序的当前状态

从统一集群中的任何队列管理器，您可以通过运行 DIS APSTATUS 命令来获取集群的所有队列管理器中应用程序的当前状态的概述。

从 IBM MQ 9.2.0 开始， **TYPE** 字段也会显示在输出中。

例如，在队列管理器刚启动后，您可能会看到类似如下的输出:

```
    1 : DIS APSTATUS(*) type(APPL)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)                         CLUSTER(UNIDEMO)
   COUNT(8)                                MOVCOUNT(8)
   BALANCED(UNKNOWN)
   TYPE (APPL)
```

这说明在统一集群中有一个名为 MYAPP 的应用程序，目前有 8 个实例，全部 8 个实例都被认为是在统一集群周围可移动的。 UNKNOWN 的均衡值是临时值，指示队列管理器尚未尝试在必要时重新均衡应用程序。

短时间后，您更有可能看到以下输出:

```
    1 : DIS APSTATUS(*) type(APPL)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)                         CLUSTER(UNIDEMO)
   COUNT(8)                                MOVCOUNT(8)
   BALANCED(NO)
   TYPE (APPL)
```

此输出显示应用程序有 8 个实例，但它们在统一集群中不平衡。 此时，值得关注的是集群中应用程序的分布情况。

要执行此操作，请再次运行 **DIS APSTATUS** 命令。 请注意，您可以对统一集群中的任何队列管理器运行此命令:

```
1 : DIS APSTATUS(*) type(QMGR)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)                         ACTIVE(YES)
   COUNT(6)                                MOVCOUNT(6)
   BALSTATE(HIGH)                          LMSGDATE(2019-05-24)
   LMSGTIME(13:11:10)                      QMNAME(UNID001)
   QMID(UNID001_2019-05-24_13.09.35)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)                         ACTIVE(YES)
   COUNT(1)                                MOVCOUNT(1)
```

```
   BALSTATE(LOW)                           LMSGDATE(2019-05-24)
   LMSGTIME(13:11:03)                      QMNAME(UNID002)
   QMID(UNID002_2019-05-24_13.09.39)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)                         ACTIVE(YES)
   COUNT(1)                                MOVCOUNT(1)
   BALSTATE(LOW)                           LMSGDATE(2019-05-24)
   LMSGTIME(13:11:07)                      QMNAME(UNID003)
   QMID(UNID003_2019-05-24_13.09.43)
   TYPE (QMGR)
```

由此，您可以在此时看到队列管理器 UNID001 有六个实例，但队列管理器 UNID0002 和 UNID0003 各只有一个实例。 BALSTATE 输出指示队列管理器上次报告平衡状态的时间。 但是，请注意，实例计数可能比 BALSTATE 字段更新。

此输出还很好地指示统一集群正在分发有关此应用程序的信息。 应用程序对于统一集群中的所有队列管理器实例都是已知的，最后一条消息的日期和时间是最新的。

此外， ACTIVE 字段指示集群中的所有队列管理器都被视为彼此通信; 如果 ACTIVE 在任何队列管理器上设置为 NO ，那么表明与它的常规通信已中断。

如果留给自己重新平衡，输出最终会产生如下的结果:

```
    1 : DIS APSTATUS(*) type(QMGR)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)                         ACTIVE(YES)
   COUNT(3)                                MOVCOUNT(3)
   BALSTATE(OK)                            LMSGDATE(2019-05-24)
   LMSGTIME(13:14:22)                      QMNAME(UNID001)
   QMID(UNID001_2019-05-24_13.09.35)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)                         ACTIVE(YES)
   COUNT(3)                                MOVCOUNT(3)
   BALSTATE(OK)                            LMSGDATE(2019-05-24)
   LMSGTIME(13:13:53)                      QMNAME(UNID002)
   QMID(UNID002_2019-05-24_13.09.39)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)                         ACTIVE(YES)
   COUNT(2)                                MOVCOUNT(2)
   BALSTATE(OK)                            LMSGDATE(2019-05-24)
   LMSGTIME(13:13:47)                      QMNAME(UNID003)
   QMID(UNID003_2019-05-24_13.09.43)
   TYPE (QMGR)
```

队列管理器的 BALSTATE 现在正常，指示已实现稳定状态。

监视个别应用程序实例

您可以查看各个应用程序实例，但可以对每个队列管理器执行此操作。 查看 UNID001: 上的输出

```
 1 : DIS APSTATUS(*) type(LOCAL)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)
   CONNTAG(MQCT02DFE75C02EA0A20UNID001_2019-05-24_13.09.35MYAPP)
   CONNS(1)                                IMMREASN(NONE)
   IMMCOUNT(0)                             IMMDATE( )
   IMMTIME( )                              MOVABLE(YES)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)
   CONNTAG(MQCT02DFE75C02E50A20UNID001_2019-05-24_13.09.35MYAPP)
   CONNS(1)                                IMMREASN(NONE)
   IMMCOUNT(0)                             IMMDATE( )
   IMMTIME( )                              MOVABLE(YES)
AMQ8932I: Display application status details.
   APPLNAME(MYAPP)
   CONNTAG(MQCT02DFE75C02E60A20UNID001_2019-05-24_13.09.35MYAPP)
   CONNS(1)                                IMMREASN(NONE)
   IMMCOUNT(0)                             IMMDATE( )
   IMMTIME( )                              MOVABLE(YES)
   TYPE (LOCAL)
One MQSC command read.
```

可动 (YES) 的存在指示如果需要，可以将此实例移动到集群中的另一个队列管理器。 在以下示例中，应用程序实例不可移动，因为它未作为客户机进行连接:

```
    3 : DISPLAY APSTATUS('ServerApp') TYPE(LOCAL)
AMQ8932I: Display application status details.
   APPLNAME(ServerApp)
   CONNTAG(MQCT02DFE75C01800B20UNID001_2019-05-24_13.09.35ServerApp)
   CONNS(1)                                IMMREASN(NOTCLIENT)
   IMMCOUNT(0)                             IMMDATE( )
   IMMTIME( )                              MOVABLE(NO)
   TYPE (LOCAL)
```

您可以使用 **CONNTAG** 字段来查看来自该实例的各个队列管理器连接，如果应用程序实例为 IMM 腻 (NOTRECONN) ，那么这些连接很有用。 在以下示例中，客户机应用程序不可移动，并且调查连接选项会显示它在 **CONNOPTS** 字段中没有 MQCNO_RECONNECT。

```
     1 : DISPLAY APSTATUS('ClientApp') TYPE(LOCAL)
AMQ8932I: Display application status details.
   APPLNAME(ClientApp)
   CONNTAG(MQCT02DFE75C01CB0B20UNID001_2019-05-24_13.09.35ClientApp)
   CONNS(1)                                IMMREASN(NOTRECONN)
   IMMCOUNT(0)                             IMMDATE( )
   IMMTIME( )                              MOVABLE(NO)
   TYPE (LOCAL)

     2 : DISPLAY CONN(*) TYPE(CONN) WHERE(CONNTAG eq
'MQCT02DFE75C01CB0B20UNID001_2019-05-24_13.09.35ClientApp') ALL
AMQ8276I: Display Connection details.
   CONN(02DFE75C01CB0B20)
   EXTCONN(414D5143554E49443030312020202020)
   TYPE(CONN)
   PID(14656)                              TID(20)
   APPLDESC(IBM MQ Channel)                APPLTAG(ClientApp)
   APPLTYPE(USER)                          ASTATE(NONE)
   CHANNEL(SYSTEM.DEF.SVRCONN)             CLIENTID( )
   CONNAME(127.0.0.1)
   CONNOPTS(MQCNO_HANDLE_SHARE_BLOCK,MQCNO_SHARED_BINDING)
   USERID(MyUserid)                        UOWLOG( )
   UOWSTDA( )                              UOWSTTI( )
   UOWLOGDA( )                             UOWLOGTI( )
   URTYPE(QMGR)
   EXTURID(XA_FORMATID[] XA_GTRID[] XA_BQUAL[])
   QMURID(0.0)                             UOWSTATE(NONE)
   CONNTAG(MQCT02DFE75C01CB0B20UNID001_2019-05-24_13.09.35ClientApp)
   TYPE (CONN)
```

相关概念

自动应用程序均衡

z/OS Monitoring performance and resource usage on z/OS

Use this topic to understand the facilities available to monitor the performance, and resource usage of your IBM MQ for z/OS subsystems.

Related tasks

Configuring queue managers on z/OS

Administering IBM MQ for z/OS

z/OS Introduction to monitoring IBM MQ for z/OS

Use this topic as an overview of the monitoring facilities available for IBM MQ for z/OS. For example, obtaining snapshots, using IBM MQ trace, online monitoring, and events.

This topic describes how to monitor the performance and resource usage of IBM MQ.

- It outlines some of the information that you can retrieve and briefly describes a general approach to investigating performance problems. See “Investigating performance problems” on page 291 for more information.

- It describes how you can collect statistics about the performance of IBM MQ by using SMF records.
- It describes how to gather accounting data to enable you to charge your customers for their use of your IBM MQ systems.
- It describes how to use IBM MQ events (alerts) to monitor your systems.

Here are some of the tools you might use to monitor IBM MQ; they are described in the sections that follow:

- Tools provided by IBM MQ:
  - Using DISPLAY commands
  - “Using CICS adapter statistics” on page 286
  - “Using IBM MQ events” on page 288
- z/OS service aids:
  - “Using System Management Facility” on page 288
- Other IBM licensed programs:
  - Using the Resource Measurement Facility
  - Using Tivoli Decision Support for z/OS
  - Using the CICS monitoring facility

Information about interpreting the data gathered by the performance statistics trace is given in “Interpreting IBM MQ for z/OS performance statistics” on page 292.

Information about interpreting the data gathered by the accounting trace is given in“Interpreting IBM MQ for z/OS accounting data” on page 327.

z/OS *Getting snapshots of IBM MQ using the DISPLAY commands*

IBM MQ provides the MQSC facility which can give a snapshot of the performance, and resource usage using the DISPLAY commands.

You can get an idea of the current state of IBM MQ by using the DISPLAY commands and, for the CICS adapter, the CICS adapter panels.

Using DISPLAY commands

You can use the IBM MQ MQSC DISPLAY or PCF Inquire commands to obtain information about the current state of IBM MQ. They provide information about the status of the command server, process definitions, queues, the queue manager, and its associated components. These commands are:

MQSC command	PCF command
DISPLAY ARCHIVE	Inquire Archive
DISPLAY AUTHINFO	Inquire Authentication Information Object
DISPLAY CFSTATUS	Inquire CF Structure Status
DISPLAY CFSTRUCT	Inquire CF Structure
DISPLAY CHANNEL	Inquire Channel
DISPLAY CHINIT	Inquire Channel Initiator
DISPLAY CHSTATUS	Inquire Channel Status
DISPLAY CMDSERV	
DISPLAY CLUSQMGR	Inquire Cluster Queue Manager
DISPLAY CONN	Inquire Connection
DISPLAY GROUP	Inquire Group

MQSC command	PCF command
DISPLAY LOG	Inquire Log
DISPLAY PROCESS	Inquire Process
DISPLAY QMGR	Inquire Queue Manager
DISPLAY QSTATUS	Inquire Queue Status
DISPLAY QUEUE	Inquire Queue
DISPLAY SECURITY	Inquire Security
DISPLAY STGCLASS	Inquire Storage Class
DISPLAY SYSTEM	Inquire System
DISPLAY TRACE	
DISPLAY USAGE	Inquire Usage

For the detailed syntax of each command, see MQSC commands or PCF commands. All of the functions of these commands (except DISPLAY CMDSERV and DISPLAY TRACE) are also available through the operations and control panels.

These commands provide a snapshot of the system only at the moment the command was processed. If you want to examine trends in the system, you must start an IBM MQ trace and analyze the results over a period of time.

z/OS *Using CICS adapter statistics*

If you are an authorized CICS user, you can use the CICS adapter control panels to display CICS adapter statistics dynamically.

These statistics provide a snapshot of information related to CICS thread usage and situations when all threads are busy. The display connection panel can be refreshed by pressing the Enter key.

For more information about configuring the IBM MQ CICS adapter, see the Configuring connections to MQ section of the CICS documentation.

z/OS *Using IBM MQ trace*

You can record performance statistics and accounting data for IBM MQ by using the IBM MQ trace facility. Use this topic to understand how to control IBM MQ trace.

The data generated by IBM MQ is sent to:

- The System Management Facility (SMF), specifically as SMF record type 115, subtypes 1 and 2 for the performance statistics trace
- The SMF, specifically as SMF record type 116, subtypes zero, 1, and 2 for the accounting trace.

If you prefer, the data generated by the IBM MQ accounting trace can also be sent to the generalized trace facility (GTF).

Starting IBM MQ trace

You can start the IBM MQ trace facility at any time by issuing the IBM MQ START TRACE command.

Accounting data can be lost if the accounting trace is started or stopped while applications are running. To collect accounting data successfully, the following conditions must apply:

- The accounting trace must be active when an application starts, and it must still be active when the application finishes.
- If the accounting trace is stopped, any accounting data collection that was active stops.

You can also start collecting some trace information automatically if you specify YES on the SMFSTAT (SMF STATISTICS) and SMFACCT (SMF ACCOUNTING) parameters of the CSQ6SYSP macro. z/OS These parameters are described in Using CSQ6SYSP.

Before starting an IBM MQ trace, read “Using System Management Facility” on page 288.

Controlling IBM MQ trace

To control the IBM MQ trace data collection at start-up, specify values for the parameters in the CSQ6SYSP macro when you customize IBM MQ. z/OS See Using CSQ6SYSP for details.

You can control IBM MQ tracing when the queue manager is running with these commands:

- START TRACE
- ALTER TRACE
- STOP TRACE

You can choose the destination to which trace data is sent. Possible destinations are:

SMF
System Management Facility

GTF
Generalized Trace Facility (accounting trace only)

SRV
Serviceability routine for diagnostic use by IBM service personnel

For daily monitoring, information is sent to SMF (the default destination). SMF data sets typically contain information from other systems; this information is not available for reporting until the SMF data set is dumped.

You can also send accounting trace information to the GTF. This information has an event identifier of 5EE. z/OS The The MQI call and user parameter, and z/OS generalized trace facility (GTF) describes how to deal with IBM MQ trace information sent to the GTF.

For information about IBM MQ commands, see MQSC commands.

Effect of trace on IBM MQ performance

Using the IBM MQ trace facility can have a significant effect on IBM MQ and transaction performance. For example, if you start a global trace for class 1 or for all classes, it is likely to increase processor usage and transaction response times by approximately 50%. However, if you start a global trace for classes 2 - 4 alone, the increase in processor usage and transaction response times is likely to be less than 1% additional processor cost to the cost of IBM MQ calls. The same applies for a statistics or accounting trace.

z/OS *Using IBM MQ online monitoring*

You can collect monitoring data for queues and channels (including automatically defined cluster-server channels) by setting the MONQ, MONCHL, and MONACLS attributes.

Table 33 on page 288 summarizes the commands to set these attributes at different levels and to display the monitoring information.

Table 33. Setting and displaying attributes to control online monitoring

Attribute	Applicable at this level	Set using command	Display monitoring information using command
MONQ	Queue	DEFINE QLOCAL DEFINE QMODEL ALTER QLOCAL ALTER QMODEL	DISPLAY QSTATUS
	Queue manager	ALTER QMGR	
MONCHL	Channel	DEFINE CHANNEL ALTER CHANNEL	DISPLAY CHSTATUS
	Queue manager	ALTER QMGR	
MONACLS	Queue manager	ALTER QMGR	

For full details of these commands, see MQSC commands. For more information about online monitoring, see “监视 IBM MQ 网络” on page 5.

z/OS *Using IBM MQ events*

IBM MQ instrumentation events provide information about errors, warnings, and other significant occurrences in a queue manager. You can monitor the operation of all your queue managers by incorporating these events into your own system management application.

IBM MQ instrumentation events fall into the following categories:

Queue manager events
These events are related to the definitions of resources within queue managers. For example, an application attempts to put a message to a queue that does not exist.

Performance events
These events are notifications that a threshold condition has been reached by a resource. For example, a queue depth limit has been reached, or the queue was not serviced within a predefined time limit.

Channel events
These events are reported by channels as a result of conditions detected during their operation. For example, a channel instance is stopped.

Configuration events
These events are notifications that an object has been created, changed, or deleted.

When an event occurs, the queue manager puts an *event message* on the appropriate *event queue*, if defined. The event message contains information about the event that can be retrieved by a suitable IBM MQ application.

IBM MQ events can be enabled using the IBM MQ commands or the operations and control panels.

See “事件类型” on page 11 for information about the IBM MQ events that generate messages, and for information about the format of these messages. See Event message reference for information about enabling the events.

z/OS *Using System Management Facility*

You can use SMF to collect statistics and accounting information. To use SMF, certain parameters must be set in z/OS and in IBM MQ.

System management facility (SMF) is a z/OS service aid used to collect information from various z/OS subsystems. This information is dumped and reported periodically, for example, hourly. You can use SMF with the IBM MQ trace facility to collect data from IBM MQ. In this way you can monitor *trends*, for example, in system utilization and performance, and collect accounting information about each user ID using IBM MQ.

To record performance statistics (record type 115) to SMF specify the following in the SMFPRMxx member of SYS1.PARMLIB or with the SETSMF z/OS operator command.

```
SYS(TYPE(115))
```

To record accounting information (record type 116) to SMF specify the following in the SMFPRMxx member of SYS1.PARMLIB or with the SETSMF z/OS operator command.

```
SYS(TYPE(116))
```

To use the z/OS command SETSMF, either PROMPT(ALL) or PROMPT(LIST) must be specified in the SMFPRM *xx* member. See SMFPRMxx (system management facilities (SMF) parameters) for more information.

You can start collecting some trace information automatically if you specify YES on the SMFSTAT (SMF STATISTICS) and SMFACCT (SMF ACCOUNTING) parameters of the CSQ6SYSP macro; this is described in Using CSQ6SYSP.

Specifying YES on the SMFSTAT and SMFACCT parameters enables you to collect trace information as a queue manager starts.

You can also start collection of the data when the queue manager is running with the **START TRACE** command, specifying `START TRACE(A)` or `START TRACE(S)`.

You can turn on or off the recording of accounting information at the queue or queue manager level using the ACCTQ parameter of the **DEFINE QLOCAL**, **DEFINE QMODEL**, **ALTER QLOCAL**, **ALTER QMODEL**, or **ALTER QMGR** commands. See MQSC commands for details of these commands.

You can control the collection of channel accounting data at the channel or queue manager level using the **STATCHL** parameter of the **DEFINE CHANNEL**, **ALTER CHANNEL** or **ALTER QMGR** commands.

You can specify the interval at which IBM MQ collects statistics and accounting data in one of these ways:

- You can collect statistics data and accounting data at different intervals, using STATIME (statistics data) and ACCTIME (accounting data) in your system parameters (described in Using CSQ6SYSP).
- You can collect statistics data and accounting data at the same interval by specifying a value for STATIME in your system parameters (described in Using CSQ6SYSP).
- You can collect statistics data and accounting data by specifying zero for STATIME.

SMF must be running before you can send data to it. For more information about SMF, see the z/OS MVS System Management Facilities (SMF) manual.

For the statistics and accounting data to be reset, at least one MQI call must be issued during the accounting interval.

Allocating additional SMF buffers

When you start a trace, you must ensure that you allocate adequate SMF buffers. Specify SMF buffering on the VSAM BUFSP parameter of the access method services DEFINE CLUSTER statement. Specify CISZ(4096) and BUFSP(81920) on the **DEFINE CLUSTER** statement for each SMF VSAM data set.

If an SMF buffer shortage occurs, SMF rejects any trace records sent to it. IBM MQ sends a CSQW133I message to the z/OS console when this occurs. IBM MQ treats the error as temporary and remains active

even though SMF data can be lost. When the shortage has been alleviated and trace recording has resumed, IBM MQ sends a CSQW123I message to the z/OS console.

Reporting data in SMF

You can use the SMF program IFASMFDP (or IFASMFDL if logstreams are being used) to dump SMF records to a sequential data set so that they can be processed.

There are several ways to report on this data, for example:

- Write an application program to read and report information from the SMF data set. You can then tailor the report to fit your exact needs.
- Use Performance Reporter to process the records. For more information, see "Using other products with IBM MQ" on page 290.

No interval CLASS(03) SMF accounting records produced during long running processes

You are collecting CLASS(3) SMF116 accounting records for IBM MQ, but are getting no records produced while a long running process runs.

The CLASS(3) SMF116 accounting records normally are produced only when a process ends. For long running processes, for example CICS, this might not produce a sufficient number of records, as the process can run for a month or longer. However, you might want to gather SMF116 records at set time intervals while a process is running.

To gather CLASS(3) SMF116 accounting records you must set the following:

SMFACCT
=YES

SMFSTAT
=YES or NO, where

YES
Causes records to be produced if a collection broadcast is received.

No
Causes you to get a CLASS(3) SMF116 record produced only when a process ends

and issue the following command:

```
START TRACE(ACCTG) DEST(SMF) CLASS(03)
```

If you have set SMFSTAT=YES and a collection broadcast occurs, an interval CLASS(3) SMF116 accounting record is produced for any process currently running that was also running at the time of the previous collection broadcast.

You can set the collection broadcast to occur on a regular time interval by setting STATIME in CSQ6SYSP as follows:

- If your STATIME has been set to a value greater than 0, that is your broadcast interval in minutes.
- If your STATIME = 0 the SMF broadcast of your system is used (SMF INTVAL)
- If your STATIME = 0 and your SMF INTVAL is not set, no broadcast occurs and no interval records are produced

z/OS *Using other products with IBM MQ*

You can use other products to help you to improve the presentation of, or to augment statistics related to, performance and accounting. For example, Resource Measurement Facility, Tivoli Decision Support, and CICS monitoring.

Using the Resource Measurement Facility

Resource Measurement Facility (RMF) is an IBM licensed program (program number 5685-029) that provides system-wide information about processor utilization, I/O activity, storage, and paging. You can use RMF to monitor the utilization of physical resources across the whole system dynamically. For more information, see the *z/OS Resource Measurement Facility User's Guide*.

Using Tivoli Decision Support for z/OS

You can use Tivoli Decision Support for z/OS to interpret RMF and SMF records.

Tivoli Decision Support for z/OS is an IBM licensed program (program number 5698-B06) that enables you to manage the performance of your system by collecting performance data in a Db2® database and presenting the data in various formats for use in systems management. Tivoli Decision Support for can generate graphic and tabular reports using systems management data it stores in its Db2 database. It includes an administration dialog, a reporting dialog, and a log collector, all of which interact with a standard Db2 database.

This is described in the *IBM Tivoli Decision Support for z/OS: Administration Guide and Reference*.

Using the CICS monitoring facility

The CICS monitoring facility provides performance information about each CICS transaction running. It can be used to investigate the resources used and the time spent processing transactions. For background information, see the *CICS Performance Guide* and Developing CICS System Programs, together with the two companion reference manuals, formerly called the *CICS Customization Guide*.

z/OS *Investigating performance problems*

Performance problems can arise from various factors. For example, incorrect resource allocation, poor application design, and I/O restraints. Use this topic to investigate some of the possible causes of performance problems.

Performance can be adversely affected by:

- Buffer pools that are an incorrect size
- Lack of real storage
- I/O contention for page sets or logs
- Log buffer thresholds that are set incorrectly
- Incorrect setting of the number of log buffers
- Large messages
- Units of recovery that last a long time, incorporating many messages for each sync point
- Messages that remain on a queue for a long time
- RACF® auditing
- Unnecessary security checks
- Inefficient program design

When you analyze performance data, always start by looking at the overall system before you decide that you have a specific IBM MQ problem. Remember that almost all symptoms of reduced performance are magnified when there is contention. For example, if there is contention for DASD, transaction response times can increase. Also, the more transactions there are in the system, the greater the processor usage and greater the demand for both virtual and real storage.

In such situations, the system shows heavy use of *all* its resources. However, the system is actually experiencing normal system stress, and this stress might be hiding the cause of a performance reduction. To find the cause of such a loss of performance, you must consider all items that might be affecting your active tasks.

Investigating the overall system

Within IBM MQ, the performance problem is either increased response time or an unexpected and unexplained heavy use of resources. First check factors such as total processor usage, DASD activity, and paging. An IBM tool for checking total processor usage is resource management facility (RMF). In general, you must look at the system in some detail to see why tasks are progressing slowly, or why a specific resource is being heavily used.

Start by looking at general task activity, then focus on particular activities, such as specific tasks or a specific time interval.

Another possibility is that the system has limited real storage; therefore, because of paging interrupts, the tasks progress more slowly than expected.

Investigating individual tasks

You can use the accounting trace to gather information about IBM MQ tasks. These trace records tell you a great deal about the activity that the task has performed, and about how much time the task spent suspended, waiting for latches. The trace record also includes information about how much Db2 and coupling facility activity were performed by the task.

Interpreting IBM MQ accounting data is described in “Interpreting IBM MQ for z/OS accounting data” on page 327.

Long running units of work can be identified by the presence of message CSQR026I in the job log. This message indicates that a task has existed for more than three queue manager checkpoints and its log records have been shunted. z/OS For a description of log record shunting, see The log files.

z/OS Interpreting IBM MQ for z/OS performance statistics

Use this topic as an index to the different SMF records created by IBM MQ for z/OS.

IBM MQ for z/OS performance statistics are written as SMF type 115 records. Statistics records are produced periodically at a time interval specified by the **STATIME** parameter of the CSQ6SYSP system parameter module, or at the SMF global recording interval if you specify zero for **STATIME**. The information provided in the SMF records comes from the following components of IBM MQ:

Buffer manager	Manages the buffer pools in virtual storage and the writing of pages to page sets as the buffer pools become full. Also manages the reading of pages from page sets.
Coupling facility manager	Manages the interface with the coupling facility.
Data manager	Manages the links between messages and queues. It calls the buffer manager to process the pages with messages on them.
Db2 manager	Manages the interface with the Db2 database that is used as the shared repository.
Lock manager	Manages locks
Log manager	Manages the writing of log records, which are essential for maintaining the integrity of the system if there is a back out request, or for recovery, if there is a system or media failure.
Message manager	Processes all IBM MQ API requests.
Storage manager	Manages storage, for example, storage pool allocation, expansion, and deallocation.
Topic manager	Manages the topic and subscription information

Coupling facility SMDS manager	Manages the shared message data sets (SMDS) for large messages stored in the coupling facility.

IBM MQ statistics are written to SMF as SMF type 115 records. The following subtypes can be present:

1
System information, for example, related to the logs and storage.

2
Information about number of messages and paging information. Queue sharing group information related to the coupling facility and Db2.

5 and 6
Detailed information about internal storage usage in the queue manager address space. While you can view this information, some of it is intended only for IBM use.

7
Storage manager summary information. While you can view this information, some of it is intended only for IBM use.

201
Page set input/output information

215
Buffer pool information

216
Queue information

231
System information for the channel initiator address space.

Note that:

- Subtype 1, 2, 201, and 215 records are created with statistics trace class 1.
- Subtype 5, 6, and 7 records are created with statistics trace class 3.
- Subtype 231 records are created with statistics trace class 4.
- Subtype 216 records are created with statistics trace class 5.

The subtype is specified in the SM115STF field (shown in Table 34 on page 296).

z/OS 收集 *SMF* 类型 *115* 和类型 *116* 记录所需的命令

使用本主题作为收集类型 115 和类型 116 SMF 记录所需命令的参考。

使用 START TRACE 命令

使用动态版本的开始追踪使用以下选项的命令来收集记录：

- `START TRACE(STAT) DEST SMF CLASS(*)`和 `START TRACE(ACCTG) DEST SMF CLASS(*)`启动 1 至 3 类的跟踪
- `START TRACE(STAT) DEST SMF CLASS(4)`和 `START TRACE(ACCTG) DEST SMF CLASS(4)`分别启动通道发起者统计和通道会计数据。

  看规划通道发起方 SMF 数据有关 4 类通道启动器信息的更多详细信息。
- `START TRACE(STAT) DEST SMF CLASS(5)`开始队列统计。

注意: 如果你使用控制台版本的命令添加 `cpf` 到您发出的命令的开始。

对于 CLASS(4) 通道统计，需要在 CHANNEL 定义上设置 STATCHL 属性。 有关更多信息，请参阅第 125 页的『控制通道统计信息收集』。

对于 CLASS(5) 队列统计信息，您需要在 QUEUE 和/或 QMGR 定义上设置 STATQ 属性。 有关更多信息，请参阅第 124 页的『控制队列统计信息收集』。

SMF 类型 115 记录 -IBM MQ 统计数据

要验证当前的统计数据收集情况，请发出显示轨迹命令 DISPLAY TRACE(STAT)。 您应该看到以下内容：

```
RESPONSE=MPX1
 CSQW127I QML1 CURRENT TRACE ACTIVITY IS -
 TNO TYPE  CLASS        DEST        USERID  RMID
 02  STAT  01,02,03,04  SMF          *      *
 END OF TRACE REPORT
```

注: 将 ZPARM SMFSTAT 或 ZPARM SMFACCT 属性设置为星号现在仅控制类 1、2 和 3。 必须使用 START TRACE 命令打开类 4 和 5。

SMF 类型 116 3 类和 4 类数据 –IBM MQ 任务和渠道会计记录

要验证当前的统计数据收集情况，请发出显示轨迹命令 DISPLAY TRACE(ACCT)。 您应该看到以下内容：

```
RESPONSE=MPX1
 CSQW127I QML1 CURRENT TRACE ACTIVITY IS -
 TNO TYPE  CLASS        DEST        USERID  RMID
 03  ACCTG  *           SMF          *      *
 END OF TRACE REPORT
 CSQ9022I QML1 CSQWVCM1  'DISPLAY TRACE'  NORMAL COMPLETION
```

你可以转 SMF116 由于不需要队列管理器循环，因此可以使用以下命令动态地打开和关闭数据收集：

```
START TRACE(ACCTG) DEST(SMF) CLASS(3)
START TRACE(ACCTG) DEST(SMF) CLASS(4)

STOP TRACE(ACCTG) DEST(SMF) CLASS(3)
STOP TRACE(ACCTG) DEST(SMF) CLASS(4)
```

信道统计数据需要额外设置才能生成 SMF。 步骤如下：

- 发出命令来开启收集所有发送方、接收方和客户端连接的信息：

```
ALTER QMGR STATCHL(HIGH)
```

- 如果正在使用集群，则发出命令来打开自动定义集群通道的信息收集：

```
ALTER QMGR STATCLS(HIGH)
```

注: 在上面的语句中，HIGH、MED 或 LOW 的值具有相同的效果。

- 验证所有通道是否已将 STATCHL 设置为 QMGR：

```
DISPLAY CHANNEL(*) STATCHL
```

验证 STATIME

注意: 您应该将间隔设置为不超过五分钟，以符合最新工具的要求，使用设定系统命令，尽管默认值是 30 分钟。

如果您正在评估队列共享组 (QSG)，请确保 STATIME 在整个 QSG 中一致。

- 使用以下命令显示系统设置显示系统命令：

```
DISPLAY SYSTEM
```

- 如果 STATIME 值为零，意味着统计间隔设置为默认 LPAR 值，那么这通常是可以的。

  曾经有过 LPAR 容量非常大导致 SMF 数据生成持续时间不一的情况。 如果在评估数据时发现持续时间差异很大，请将所有队列管理器的 STATIME 设置为非零值。

- 以下示例将 STATIME 间隔设置为五分钟：

```
SET SYSTEM STATIME(05)
```

或者如果 IBM MQ 9.2.4 或以上：

```
SET SYSTEM STATIME(05.00)
```

注： 新的间隔直到当前间隔过去后才会生效，因此您需要在数据收集开始之前进行更改。

验证处于发行版级别的队列管理器的 ACCTIME 值 IBM MQ 9.2.4 或更高：

- 如果 ACCTIME 设置为 -1，则它与 STATIME 属性相同。
- 如果此评估针对 QSG，请确保 ACCTIME 在整个 QSG 中一致。 如果不是，请在开始收集数据之前将值设置为相同的间隔。
- 如果 ACCTTIME 设置为 30 或更高，请使用设定系统命令。 以下示例将 ACCTIME 间隔设置为五分钟：

```
SET SYSTEM ACCTIME(05.00)
```

注： 新的间隔直到当前间隔过去后才会生效，因此您需要在数据收集开始之前进行更改。

验证队列管理器上的 ACCTQ 设置

- 使用以下命令显示您正在调查的队列管理器的 ACCTQ 设置：

```
DISPLAY QMGR ACCTQ
```

- 如果值为 ACCTQ(ON)，则无需采取进一步措施。 否则，发出以下命令：

```
ALTER QMGR ACCTQ(ON)
```

- 如果正在使用集群，请确保已为 SYSTEM.CLUSTER.TRANSMIT.QUEUE 以及队列管理器托管的任何其他命名集群传输队列。 要确定是否为集群传输队列启用了记帐：
  - 使用以下命令显示队列的 ACCTQ 设置：

```
DISPLAY QL(SYSTEM.CLUSTER.TRANSMIT.QUEUE) ACCTQ
```

  - 如果值为 ACCTQ(ON)，则无需采取进一步措施。 否则，发出以下命令：

```
ALTER QL(SYSTEM.CLUSTER.TRANSMIT.QUEUE) ACCTQ(ON)
```

- 对于所有其他大容量队列，请验证 ACCTQ 的值是否为 ON，或将 ACCTG 设置为 QMGR。

z/OS *Layout of an SMF type 115 record*

You can use this section as a reference for the format of an SMF type 115 record.

The standard layout for SMF records involves three parts:

SMF header
Provides format, identification, and time and date information about the record itself.

Self-defining section
Defines the location and size of the individual data records within the SMF record.

Data records
The actual data from IBM MQ that you want to analyze.

For more information about SMF record formats, see *z/OS MVS System Management Facilities (SMF)*.

Related reference

“The SMF header” on page 296
Use this topic as a reference for the format of the SMF header.

“Self-defining sections” on page 296

Use this topic as a reference for format of the self-defining sections of the SMF record.

"Examples of SMF statistics records" on page 297
Use this topic to understand some example SMF records.

z/OS The SMF header

Use this topic as a reference for the format of the SMF header.

Table 34 on page 296 shows the format of SMF record header (SM115).

Table 34. SMF record 115 header description

Offset: Dec	Offset: Hex	Type	Len	Name	Description	Example
0	0	Structure	28	SM115	SMF record header.	
0	0	Integer	2	SM115LEN	SMF record length.	14A0
2	2		2		Reserved.	
4	4	Integer	1	SM115FLG	System indicator.	5E
5	5	Integer	1	SM115RTY	Record type. The SMF record type, for IBM MQ statistics records this is always 115 (X'73').	73
6	6	Integer	4	SM115TME	Time when SMF moved record.	00355575
10	A	Integer	4	SM115DTE	Date when SMF moved record.	0100223F
14	E	Character	4	SM115SID	z/OS subsystem ID. Defines the z/OS subsystem on which the records were collected.	D4E5F4F1 (MV41)
18	12	Character	4	SM115SSI	IBM MQ subsystem ID.	D4D8F0F7 (MQ07)
22	16	Integer	2	SM115STF	Record subtype.	0002
24	18	Character	3	SM115REL	IBM MQ version.	F6F0F0 (600)
27	1B		1		Reserved	
28	1C	Character	0	SM115END	End of SMF header and start of self-defining section.	

z/OS Self-defining sections

Use this topic as a reference for format of the self-defining sections of the SMF record.

A self-defining section of a type 115 SMF record tells you where to find a statistics record, how long it is, and how many times that type of record is repeated (with different values). The self-defining sections follow the header, at fixed offsets from the start of the SMF record. Each statistics record can be identified by an eye-catcher string.

The following types of self-defining section are available to users for type 115 records. Each self-defining section points to statistics data related to one of the IBM MQ components. Table 35 on page 297 summarizes the sources of the statistics, the eye-catcher strings, and the offsets of the self-defining sections from the start of the SMF record header.

Table 35. Offsets to self-defining sections				
Source of statistics	**Record subtype (SM115STF)**	**Offset of self-defining section**		**Eye-catcher of data**
		Dec	**Hex**	
Storage manager	1	100	X'64'	QSST
Log manager	1	116	X'74'	QJST
Message manager	2	36	X'24'	QMST
Data manager	2	44	X'2C'	QIST
No longer used. The self-defining section will be binary zeros.	2	52	X'34'	
Lock manager	2	60	X'3C'	QLST
Db2 manager	2	68	X'44'	Q5ST
Coupling Facility manager	2	76	X'4C'	QEST
Topic manager	2	84	X'54'	QTST
SMDS usage	2	92	X'5C'	QESD
Buffer manager - one for each buffer pool	215	36	X'24'	QPST
Channel initiator	231			QWSX
Data manager page set - one for each page set	201	36	X'24'	QIS1
Storage manager	5	36	X'24'	QSPH
Storage manager	6	36	X'24'	QSGM
Storage manager	7	36	X'24'	QSRS
Queues	216	36	X'24'	QQST

Note: Some of the storage manager information in subtype 5, 6 and 7 records is intended only for IBM use. Other self-defining sections that are not listed contain data for IBM use only.

Each self-defining section is two fullwords long and has this format:

```
ssssssssllllnnnn
```

where:

- *ssssssss* is a fullword containing the offset from the start of the SMF record.
- *llll* is a halfword giving the length of this data record.
- *nnnn* is a halfword giving the number of data records in this SMF record.

For more information see, “Examples of SMF statistics records” on page 297.

Note: Always use offsets in the self-defining sections to locate the statistics records.

z/OS *Examples of SMF statistics records*

Use this topic to understand some example SMF records.

Figure 20 on page 298 shows an example of part of the SMF record for subtype 1. Subtype 1 includes the storage manager and log manager statistics records. The SMF record header is shown underlined.

The self-defining section at offset X'64' refers to storage manager statistics and the self-defining section at offset X'74' refers to log manager statistics, both shown in **bold**.

The storage manager statistics record is located at offset X'0000011C' from the start of the header and is X'48' bytes long. There is one set of storage manager statistics, identified by the eye-catcher string QSST. The start of this statistics record is also shown in the example.

The log manager statistics record is located at offset X'00000164' from the start of the header and is X'78' bytes long. There is one set of log manager statistics, identified by the eye-catcher string QJST.

```
000000  02000000 5E730035 55750100 223FD4E5  *....;.........MV*
000010  F4F1D4D8 F0F70001 F6F0F000 000001DC  *41MQ07..600.....*
000020  00240001 00000000 00000000 00000000  *.$..............*
000030  00000000 00000000 00000000 0000007C  *...............@*
000040  00400001 000000BC 00600001 00000000  *. .......-......*
000050  00000000 00000000 00000000 00000000  *................*
000060  00000000 0000011C 00480001 00000000  *.............H..*
000070  00000000 00000164 00780001 00000000  *.............x..*
000080  00000000 00000000 00000000 00000000  *................*
.
.
000110  00000000 00000000 00000000 003C0048  *.............<.H*
000120  D8E2E2E3 0000004F 00000003 00000002  *QSST...|........*
```

Figure 20. SMF record 115, subtype 1

Figure 21 on page 299 shows an example of part of the SMF record for subtype 2. Subtype 2 includes the statistics records for the message, data, lock, coupling facility, topic, and Db2 managers. The SMF record header is shown underlined; the self-defining sections are shown alternately **bold** and *italic*.

- The self-defining section at offset X'24' refers to message manager statistics. The message manager statistics record is located at offset X'00000064' from the start of the header and is X'48' bytes long. There is one set of these statistics, identified by the eye-catcher string QMST.
- The self-defining section at offset X'2C' refers to data manager statistics. The data manager statistics record is located at offset X'000000AC' from the start of the header and is X'50' bytes long. There is one set of these statistics, identified by the eye-catcher string QIST.
- In releases prior to IBM MQ 9.1.0, the self-defining section at offset X'34' referred to buffer manager statistics. As this SMF record was taken from an IBM MQ 9.1.0 queue manager, the buffer manager self-defining section is set to zeros to indicate that there are no buffer manager statistics. Instead, these statistics are in SMF 115 subtype 215 records.
- The self-defining section at offset X'3C' refers to lock manager statistics. The lock manager statistics record is located at offset X'000000FC' from the start of the header and is X'20' bytes long. There is one set of these statistics, identified by the eye-catcher string QLST.
- The self-defining section at offset X'44' refers to Db2 manager statistics. The Db2 manager statistics record is located at offset X'0000011C' from the start of the header and is X'2A0' bytes long. There is one set of these statistics, identified by the eye-catcher string Q5ST.
- The self-defining section at offset X'4C' refers to coupling facility manager statistics. The coupling facility manager statistics record is located at offset X'000003BC' from the start of the header and is X'1008' bytes long. There is one set of these statistics, identified by the eye-catcher string QEST.
- The self-defining section at offset X'54' refers to topic manager statistics. The topic manager statistics record is located at offset X'000013C4' from the start of the header and is X'64' bytes long. There is one set of these statistics, identified by the eye-catcher string QTST.
- The self-defining section at offset X'5C' is for SMDS statistics. This self defining section is set to zeros indicating that SMDS is not being used.

```
000000 09F40000 5E730033 4DBE0113 142FD4E5 *.4..;...(.....MV*
000010 F4F1D4D8 F2F10002 F9F1F000 00001428 *41MQ21..910.....*
000020 00240001 00000064 00480001 000000AC *.................*
000030 00500001 00000000 00000000 000000FC *.................*
000040 00200001 0000011C 02A00001 000003BC *.................*
000050 10080001 000013C4 00640001 00000000 *.......D........*
000060 00000000 D40F0048 D8D4E2E3 00000000 *....M...QMST....*
000080 00000000 00000000 00000000 00000000 *.................*
000090 00000000 00000000 00000000 00000000 *.................*
0000A0 00000000 00000000 00000000 C90F0050 *............I..&*
0000B0 D8C9E2E3 00000000 00000000 00000000 *QIST............*
0000C0 00000000 00000000 00000000 00000000 *.................*
0000D0 00000000 00000000 00000000 00000000 *.................*
0000E0 00000000 00000000 00000000 00000000 *.................*
0000F0 00000000 00000000 00000000 D30F0020 *............L...*
000100 D8D3E2E3 00000000 00000000 00000000 *QLST............*
000110 00000000 00000000 00000000 F50F02A0 *............5...*
000120 D8F5E2E3 00000008 00000000 00000000 *Q5ST............*
.
.
```

Figure 21. SMF record 115, subtype 2

z/OS *Processing type 115 SMF records*

Use this topic as a reference for processing type 115 SMF records.

You must process any data you collect from SMF to extract useful information. When you process the data, verify that the records are from IBM MQ and that they are the records you are expecting.

Validate the values of the following fields:

- SM115RTY, the SMF record number, must be X'73' (115)
- SM115STF, the record subtype, must be 0001, 0002, 0005, 0006, 0007, 0201, 0215, or 0231

Reading from the active SMF data sets (or SMF logstreams) is not supported. You must use the SMF program IFASMFDP (or IFASMFDL if logstreams are being used) to dump SMF records to a sequential data set so that they can be processed. For more information see "Using System Management Facility" on page 288.

Details of the structures and fields can be found in IBM MQ SupportPac MP1B.

There is a C sample program called CSQ4SMFD which prints the contents of SMF type 115 and 116 records from the sequential data set. The program is provided as source in thlqual.SCSQC37S and in executable format in thlqual.SCSQLOAD. Sample JCL is provided in thlqual.SCSQPROC(CSQ4SMFJ).

z/OS *Storage manager data records*

Use this topic as a reference for storage manager data records.

The format of the storage manager statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQSST).

The data contains information about the number of fixed and variable storage pools that the queue manager has allocated, expanded, contracted, and deleted during the statistics interval, plus the number of GETMAIN, FREEMAIN, and STORAGE requests to z/OS, including a count of those requests that were unsuccessful. Additional information includes a count of the number of times the short-on-storage condition was detected and a count of the number of abends that occurred as a result of that condition.

Additional data about storage usage in the queue manager is produced by class 2 and class 3 statistics trace. While you can view this information, some of it is intended only for IBM use.

- The format of the storage manager pool header statistics record, which is present in subtype 5 records, is described in assembler macro thlqual.SCSQMACS(CSQDQSPH).
- The format of the storage manager getmain statistics record, which is present in subtype 6 records, is described in assembler macro thlqual.SCSQMACS(CSQDQSGM).

- The format of the storage manager region summary record, which is present in subtype 7 records, is described in assembler macro thlqual.SCSQMACS(CSQDQSRS).

z/OS 日志管理器数据记录

使用本主题作为日志管理器数据记录格式的参考。

日志管理器统计记录的格式在汇编宏中描述 `thlqual.SCSQMACS(CSQDQJST)`。

在统计信息中，这些计数很重要:

1. 日志写入请求总数:

$$N_{logwrite} = QJSTWRNW + QJSTWRF$$

2. 日志读取请求总数:

$$N_{logread} = QJSTRBUF + QJSTRACT + QJSTRARH$$

下表描述了可使用日志管理器统计信息检查的问题症状。

症状 1 QJSTWTB 非零。 **原因** 正在将存储中缓冲区写入活动日志时暂挂任务。 写入活动日志时可能发生问题。 CSQ6LOGP 中的 OUTBUFF 参数太小。 **操作** 调查写入活动日志的问题。 增大 CSQ6LOGP 中 OUTBUFF 参数的值。
症状 2 比率: $QJSTWTL/N_{logread}$ 大于 1%。 **原因** 已启动必须从归档日志中读取的日志读取，但 IBM MQ 无法分配数据集，因为已分配 MAXRTU 数据集。 **操作** 增加 MAXRTU。
症状 3 比率: $QJSTRARH/N_{logread}$ 大于正常值。 **原因** 大多数日志读取请求应该来自输出缓冲区或活动日志。 为了满足回退请求，将从存储中缓冲区，活动日志和归档日志中读取恢复单元记录。 长时间运行的恢复单元 (在许多分钟的时间段内) 可能将日志记录分布在许多不同的日志中。 这会降低性能，因为必须执行额外的工作才能恢复日志记录。 **操作** 更改应用程序以缩短恢复单元的长度。 此外，请考虑增大活动日志的大小，以减少单个恢复单元分布在多个日志中的可能性。 **其他指针** 比率 $N_{logread}/N_{logwrite}$ 指示必须回退的工作量。

症状 4
QJSTLLCP 超过 10 个小时。

原因
在繁忙的系统上，您预计通常每小时会看到 10 个检查点。 如果 QJSTLLCP 值大于此值，那么表明队列管理器的设置存在问题。

最可能的原因是 CSQ6SYSP 中的 LOGLOAD 参数太小。 导致检查点的另一个事件是当活动日志填满并切换到下一个活动日志数据集时。 如果您的日志太小，那么这可能会导致频繁的检查点。

QJSTLLCP 是检查点总数的计数。

操作
增大 LOGLOAD 参数，或者根据需要增大日志数据集的大小。

症状 5
QJSTCmpFail > 0 或 QJSTCmpComp 不小于 QJSTCmpUncmp

原因

队列管理器无法显着压缩日志记录。

QJSTCmpFail 是队列管理器无法实现任何缩短记录长度的次数。 您应该将该数字与 QJSTCmpReq (压缩请求数) 进行比较，以了解失败次数是否显着。

QJSTCmpComp 是写入日志的压缩字节总数， QJSTCmpUncmp 是压缩前的总字节数。 这两个总数都不包含针对不适合压缩的日志记录写入的字节数。 如果数字相似，那么压缩几乎没有好处。

操作
关闭日志压缩。 发出 SET LOG COMPLOG (NONE) 命令。 请参阅 SET LOG 命令以获取详细信息。

注: 在系统启动后生成的第一组统计信息中，由于正在进行的恢复单元的解析，可能存在重要的日志活动。

对 zHyper 写的更改

V 9.4.0

从 IBM MQ 9.3.5 开始， zHyper 写处理将改变 QJSTHWC 和 QJSTHWE 统计信息的行为。

在 IBM MQ 9.3.5 之前:

- QJSTHWC 是 SMF 时间间隔内在 zHyper 支持写的卷上使用的日志数据集数。 卷的 zHyper 写功能是在队列管理器启动时获得的，可随时间推移而更改，因此此信息可能是旧信息。
- QJSTHWE 是在启用了 zHyper 写的情况下写入 SMF 时间间隔内使用的日志数据集数。 如果日志数据集位于支持写的 zHyper 卷上，并且已通过设置 ZHYWRITE (YES) 为 zHyper 写启用了队列管理器，那么会发生此情况。

从 IBM MQ 9.3.5 中:

- QJSTHWC 是 SMF 时间间隔内在 zHyper 支持写的卷上使用的日志数据集数。 卷的 zHyper 写功能是在队列管理器启动时获得的，可随时间推移而更改，因此此信息可能是旧信息。
- QJSTHWE 是在启用了 zHyper 写的情况下写入 SMF 时间间隔内使用的日志数据集数。 如果已通过设置 ZHYWRITE (YES) 为 zHyper 写启用队列管理器，那么会发生此情况。

注意: 卷的 zHyper 写功能可能会随时间变化。 从 IBM MQ 9.3.5 开始，这可能会导致 QJSTHWE 大于 QJSTHWC 的场景。

zHyper 链接统计信息

V 9.4.0

从 IBM MQ 9.4.0 开始，已添加日志管理器统计信息以跟踪 zHyper 链接的性能。

以下统计信息已添加到 QJST:

名称	类型	长度字节	描述
QJSTHLSCIW	无符号整数	4	请求的单页写入次数 zHyper 关联。
QJSTHLMCIW	无符号整数	4	请求的多页写入次数 zHyper 关联。
QJSTHLCICNTW	无符号整数	4	请求写入的位置已写入的页数 zHyper 关联。
QJSTHLSCIS	无符号整数	4	成功使用 zHyper 链接的页面写入数。 即，写是同步进行的。
QJSTHLMCIS	无符号整数	4	成功使用的多页写入次数 zHyper 关联。 即，写是同步进行的。
QJSTHLCICNTS	无符号整数	4	使用以下方式成功写入的页数 zHyper 关联。 即，写是同步进行的。
QJSTHLSCIF	无符号整数	4	尝试使用 zHyper 链接的单页写操作数，但无法同步进行写操作。 而是以异步方式成功进行了写入。
QJSTHLMCIF	无符号整数	4	尝试使用 zHyper 链接的多页写入数，但无法同步进行写入。 而是以异步方式成功进行了写入。
QJSTHLCICNTF	无符号整数	4	写入尝试使用的页面数 zHyper 链接，但无法同步写入。 而是以异步方式成功进行了写入。
QJSTHLSCICONCNT	无符号整数	4	之前单页写入的次数 zHyper 链接写入会话已建立。 请参阅“注”(第 302 页的『1』)。
QJSTHLMCICONCNT	无符号整数	4	之前多页写入的次数 zHyper 链接写入会话已建立。 请参阅“注”(第 302 页的『1』)。
QJSTHLCONCICNT	无符号整数	4	之前写入的页数 zHyper 鏈接已建立。 请参阅“注”(第 302 页的『1』)。
QJSTHLSCIWTMAX	无符号整数	8	最长 zHyper 链接写入时间间隔，单页。
QJSTHLSCIWTMIN	无符号整数	8	最短 zHyper 链接写入时间间隔，单页。
QJSTHLSCIWTTOT	无符号整数	8	全部的 zHyper 链接写入时间间隔，单页。
QJSTHLMCIWTMAX	无符号整数	8	最长 zHyper 链接间隔写入时间，多页。
QJSTHLMCIWTMIN	无符号整数	8	最短 zHyper 链接间隔写入时间，多页。
QJSTHLMCIWTTOT	无符号整数	8	全部的 zHyper 链接间隔写入时间，多页。
QJSTHLIOSQU	无符号整数	16	平方和 zHyper 链接写入时间，用于单页写入。
QJSTHLC	无符号整数	4	此 SMF 间隔内使用的新日志数，包括 zHyper 可链接。
QJSTHLE	无符号整数	4	此 SMF 间隔内使用的新日志数，包括 zHyper 鏈接已啟用。

注:

1. 切换到新的活动日志副本时，可能需要一段时间才能 zHyper 与 DASD 建立链接写入会话。

z/OS *Message manager data records*

Use this topic as a reference for message manager data records.

The format of the message manager statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQMST).

The data gives you counts of different IBM MQ API requests.

z/OS *Data manager data records*

Use this topic as a reference for the format of the Data Manager data records.

The format of the data manager statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQIST).

The data gives you counts of different object requests.

z/OS *Data manager page set data records*

Use this section as a reference for the format of the data manager page set data records

The format of the data manager page set statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQIS1).

The page set usage information helps to facilitate better management of local queues within a queue manager by recording information such as page set input-output rates and highest usage.

The data provides the same basic page set information as output by the MQSC **DISPLAY USAGE TYPE(PAGESET)** command, or the PCF Inquire Usage (**MQCMD_INQUIRE_USAGE**) command.

For example:

- The total pages
- The current used pages
- Unused persistent and nonpersistent pages
- Expansion method
- Number of extends
- Number of stripes

The data also provides some performance indicators, together with other performance information. For example:

- How many times deferred write, immediate write, and read page I/O requests happened during the SMF interval,
- The number of pages moved, elapsed time and number of read and write operations.
- How many pages written in checkpoints.
- Has expansion occurred during the SMF interval?
- How many times the page set became full.
- An indication of where new space is being allocated within the page set.

From the information displayed, you should be able to understand the general status of each page set, and consider whether you need to retune the system.

Related reference

"The SMF header" on page 296
Use this topic as a reference for the format of the SMF header.

"Self-defining sections" on page 296
Use this topic as a reference for format of the self-defining sections of the SMF record.

"Examples of SMF statistics records" on page 297
Use this topic to understand some example SMF records.

z/OS *Buffer manager data records*

Use this topic as a reference for the format of buffer manager data records.

The format of the buffer manager statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQPST).

Note: Buffer manager statistics records will only be created for buffer pools that are defined. If a buffer pool is defined but not used then no values will be set and its buffer manager statistics record will not contain any data.

For information about efficiently managing your buffer pools, see "Managing your buffer pools" on page 305.

When interpreting the statistics, you are recommended to consider the following factors because the values of these fields can be used to improve the performance of your system:

1. If QPSTSOS, QPSTDMC, or QPSTIMW is greater than zero, you should either increase the size of the buffer pool or reallocate the page sets to different buffer pools.
   - QPSTSOS is the number of times that there were no buffers available for page get requests. If QPSTSOS ever becomes nonzero, it shows that IBM MQ is under severe stress. The buffer pool size should be increased. If increasing the buffer pool size does not make the value of QPSTSOS zero, there might be I/O contention on the DASD page sets.
   - QPSTDMC is the number of updates that were performed synchronously because there was either more than 95% of the pages in the buffer pool waiting for write I/O, or there was less than 5% of the buffer pool available for read requests. If this number is not zero, the buffer pool might be too small and should be enlarged. If increasing the buffer pool size does not reduce QPSTDMC to zero, there might be I/O contention on the DASD page sets.
   - QPSTIMW is a count of the number of times pages were written out synchronously. If QPSTDMC is zero, QPSTIMW is the number of times pages were found on the queue waiting for write I/O that had been there for at least two checkpoints.
2. For buffer pool zero and buffer pools that contain short-lived messages:
   - QPSTDWT should be zero, and the percentage QPSTCBSL/QPSTNBUF should be greater than 15%.

     QPSTDWT is the number of times the asynchronous write processor was started because there was either more than 85% of the pages in the buffer pool waiting for write I/O, or there was less than 15% of the buffer pool available for read requests. Increasing the buffer pool size should reduce this value. If it does not, the pattern of access is one of long delays between puts and gets.
   - QPSTTPW might be greater than zero due to checkpointing activity.
   - QPSTRIO should be zero unless messages are being read from a page set after the queue manager is restarted.

     The ratio of QPSTRIO to QPSTGETP shows the efficiency of page retrieval within the buffer pool. Increasing the buffer pool size should decrease this ratio and, therefore, increase the page retrieval efficiency. If this does not happen, it indicates that pages are not being frequently reaccessed. This implies a transaction pattern where there is a long delay between messages being put and then later retrieved.

     The ratio of QPSTGETN to QPSTGETP indicates the number of times an empty page, as opposed to a non-empty page, has been requested. This ratio is more an indication of transaction pattern, than a value that can be used to tune the system.
   - If QPSTSTL has a value greater than zero, this indicates that pages that have not been used before are now being used. This might be caused by an increased message rate, messages not being processed as fast as they were previously (leading to a buildup of messages), or larger messages being used.

     QPSTSTL is a count of the number of times a page access request did not find the page already in the buffer pool. Again, the lower the ratio of QPSTSTL to (QPSTGETP + QPSTGETN) is, the higher the

page retrieval efficiency. Increasing the buffer pool size should decrease this ratio but, if it does not, it is an indication that there are long delays between puts and gets.

- You are recommended to have sufficient buffers to handle your peak message rate.

3. For buffer pools with long-lived messages, where there are more messages than can fit into the buffer pool:
   - (QPSTRIO+QPSTWIO)/Statistics interval is the I/O rate to page sets. If this value is high, you should consider using multiple page sets on different volumes to allow I/O to be carried out in parallel.
   - Over the period of time that the messages are processed (for example, if messages are written to a queue during the day and processed overnight) the number of read I/Os (QPSTRIO) should be approximately the total number of pages written (QPSTTPW). This shows that one page is read for every page written.

     If QPSTRIO is much larger than QPSTTPW, this shows that pages are being read in multiple times. This might be a result of the application using MQGET by *MsgId* or *CorrelId* when the queue is not indexed, or browsing messages on the queue using get next.

     The following actions might relieve this problem:

     a. Increase the size of the buffer pool so that there are enough pages to hold the queue, in addition to any changed pages.

     b. Use the INDXTYPE queue attribute, which allows a queue to be indexed by *MsgId* or *CorrelId* and eliminates the need for a sequential scan of the queue.

     c. Change the design of the application to eliminate the use of MQGET with *MsgId* or *CorrelId*, or the get next with browse option.

        Note: Applications using long-lived messages typically process the first available message and do not use MQGET with *MsgId* or *CorrelId*, and they might browse only the first available message.

     d. Move page sets to a different buffer pool to reduce contention between messages from different applications.

z/OS *Managing your buffer pools*

To manage your buffer pools efficiently, you must consider the factors that affect the buffer pool I/O operations and also the statistics associated with the buffer pools.

The following factors affect buffer pool I/O operations.

- If a page containing the required data is not found in the buffer pool, it is read in synchronously to an available buffer from its DASD page set.
- Whenever a page is updated, it is put on an internal queue of pages to be (potentially) written out to DASD. This means that the buffer used by that page is unavailable for use by any other page until the buffer has been written to DASD.
- If the number of pages queued to be written to DASD exceeds 85% of the total number of buffers in the pool, an asynchronous write processor is started to put the buffers to DASD.

  Similarly, if the number of buffers available for page get requests become less than 15% of the total number of buffers in the pool, the asynchronous write processor is started to perform the write I/O operations.

  The write processor stops when the number of pages queued to be written to DASD has fallen to 75% of the total number of buffers in the pool.
- If the number of pages queued for writing to DASD exceeds 95% of the total number of buffers in the pool, all updates result in a synchronous write of the page to DASD.

  Similarly, if the number of buffers available for page get requests becomes less than 5% of the total number of buffers in the pool, all updates result in a synchronous write of the page to DASD.
- If the number of buffers available for page get requests ever reaches zero, a transaction that encounters this condition is suspended until the asynchronous write processor has finished.

- If a page is frequently updated, the page spends most of its time on the queue of pages waiting to be written to DASD. Because this queue is in least recently used order, it is possible that a frequently updated page placed on this least recently used queue is never written out to DASD. For this reason, at the time of update, if the page is found to have been waiting on the write operation to DASD queue for at least two checkpoints, it is synchronously written to DASD. Updating occurs at checkpoint time and is suspended until the asynchronous write processor has finished.

  The aim of this algorithm is to maximize the time pages spend in buffer pool memory while allowing the system to function if the system load puts the buffer pool usage under stress.

z/OS *Lock manager data records*

Use this topic as a reference to the format of the lock manager data records.

The format of the lock manager statistics record is described in assembler macro thlqual.SCSQMACS(CSQDQLST).

The records contain data about the following information:

- The number of lock get requests and lock release requests.
- The number of times a lock get request determined that the requested lock was already held.

z/OS *Db2 manager data records*

Use this topic as a reference to the format of the Db2 manager data records.

The format of the Db2 manager statistics record is described in the following table and in assembler macro thlqual.SCSQMACS(CSQDQ5ST) and C header file thlqual.SCSQC370(CSQDSMFC). The field names in C are all in lowercase, for example q5st, q5stid.

If the queue manager was not started as a member of a queue sharing group, no data is recorded in this record.

Table 36. Db2 statistics record (Q5ST)

Offset: Dec	Offset: Hex	Type	Len	Name	Description
0	0	Structure	668	Q5ST	Db2 manager statistics
0	0	Bitstring	2	Q5STID	Control block identifier
2	2	Integer	2	Q5STLL	Control block length
4	4	Character	4	Q5STEYEC	Control block eye catcher
8	8	Character	660	Q5STZERO	QMST part cleared on occasion
8	8	Integer	4	NUMTASK	Number of server tasks
12	C	Integer	4	ACTTASK	Number of active server tasks
16	10	Integer	4	CONNCNT	Number of connect requests
20	14	Integer	4	DISCCNT	Number of disconnect requests
24	18	Integer	4	DHIGMAX	Max. request queue depth
28	1C	Integer	4	ABNDCNT	Number of Db2SRV task abends
32	20	Integer	4	REQUCNT	Number of requests requeued
36	24	Integer	4	DEADCNT	Number of deadlock timeouts
40	28	Integer	4	DELECNT	Number of delete requests
44	2C	Integer	4	LISTCNT	Number of list requests

Table 36. Db2 statistics record (Q5ST) (continued)

Offset: Dec	Offset: Hex	Type	Len	Name	Description
48	30	Integer	4	READCNT	Number of read requests
52	34	Integer	4	UPDTCNT	Number of update requests
56	38	Integer	4	WRITCNT	Number of write requests
60	3C	Integer	4	SCSSEL	SCST (shared-channel-status) selects
64	40	Integer	4	SCSINS	SCST inserts
68	44	Integer	4	SCSUPD	SCST updates
72	48	Integer	4	SCSDEL	SCST deletes
76	4C	Integer	4	SSKSEL	SSKT (shared-sync-key) selects
80	50	Integer	4	SSKINS	SSKT inserts
84	54	Integer	4	SSKDEL	SSKT deletes
88	58	Integer	4	SCSBFTS	SCST number of times buffer too small
92	5C	Integer	4	SCSMAXR	SCST maximum rows on query
96	60	Integer	4	* (2)	Reserved
104	68	Character	8	DELETCUW	Cumulative STCK difference - Thread delete
112	70	Character	8	DELETMXW	Maximum STCK difference - Thread delete
120	78	Character	8	DELESCUW	Cumulative STCK difference - SQL delete
128	80	Character	8	DELESMXW	Maximum STCK difference - SQL delete
136	88	Character	8	LISTTCUW	Cumulative STCK difference - Thread list
144	90	Character	8	LISTTMXW	Maximum STCK difference - Thread list
152	98	Character	8	LISTSCUW	Cumulative STCK difference - SQL list
160	A0	Character	8	LISTSMXW	Maximum STCK difference - SQL list
168	A8	Character	8	READTCUW	Cumulative STCK difference - Thread read
17 6	B0	Character	8	READTMXW	Maximum STCK difference - Thread read
184	B8	Character	8	READSCUW	Cumulative STCK difference - SQL read
192	C0	Character	8	READSMXW	Maximum STCK difference - SQL read
200	C8	Character	8	UPDTTCUW	Cumulative STCK difference - Thread update
208	D0	Character	8	UPDTTMXW	Maximum STCK difference - Thread update
216	D8	Character	8	UPDTSCUW	Cumulative STCK difference - SQL update
224	E0	Character	8	UPDTSMXW	Maximum STCK difference - SQL update
232	E8	Character	8	WRITTCUW	Cumulative STCK difference - Thread write
240	F0	Character	8	WRITTMXW	Maximum STCK difference - Thread write
248	F8	Character	8	WRITSCUW	Cumulative STCK difference - SQL write
256	100	Character	8	WRITSMXW	Maximum STCK difference - SQL write
264	108	Character	8	SCSSTCUW	Cumulative STCK difference - Thread select

Table 36. Db2 statistics record (Q5ST) (continued)

Offset: Dec	Offset: Hex	Type	Len	Name	Description
272	110	Character	8	SCSSTMXW	Maximum STCK difference - Thread select
280	118	Character	8	SCSSSCUW	Cumulative STCK difference - SQL select
288	120	Character	8	SCSSSMXW	Maximum STCK difference - SQL select
296	128	Character	8	SCSITCUW	Cumulative STCK difference - Thread insert
304	130	Character	8	SCSITMXW	Maximum STCK difference - Thread insert
312	138	Character	8	SCSISCUW	Cumulative STCK difference - SQL insert
320	140	Character	8	SCSISMXW	Maximum STCK difference - SQL insert
328	148	Character	8	SCSUTCUW	Cumulative STCK difference - Thread update
336	150	Character	8	SCSUTMXW	Maximum STCK difference - Thread update
344	158	Character	8	SCSUSCUW	Cumulative STCK difference - SQL update
352	160	Character	8	SCSUSMXW	Maximum STCK difference - SQL update
360	168	Character	8	SCSDTCUW	Cumulative STCK difference - Thread delete
368	170	Character	8	SCSDTMXW	Maximum STCK difference - Thread delete
376	178	Character	8	SCSDSCUW	Cumulative STCK difference - SQL delete
384	180	Character	8	SCSDSMXW	Maximum STCK difference - SQL delete
392	188	Character	8	SSKSTCUW	Cumulative STCK difference - Thread select
400	190	Character	8	SSKSTMXW	Maximum STCK difference - Thread select
408	198	Character	8	SSKSSCUW	Cumulative STCK difference - SQL select
416	1A0	Character	8	SSKSSMXW	Maximum STCK difference - SQL select
424	1A8	Character	8	SSKITCUW	Cumulative STCK difference - Thread insert
432	1B0	Character	8	SSKITMXW	Maximum STCK difference - Thread insert
440	1B8	Character	8	SSKISCUW	Cumulative STCK difference - SQL insert
448	1C0	Character	8	SSKISMXW	Maximum STCK difference - SQL insert
456	1C8	Character	8	SSKDTCUW	Cumulative STCK difference - Thread delete
464	1D0	Character	8	SSKDTMXW	Maximum STCK difference - Thread delete
472	1D8	Character	8	SSKDSCUW	Cumulative STCK difference - SQL delete
480	1E0	Character	8	SSKDSMXW	Maximum STCK difference - SQL delete
488	1E8	Integer	4	LMSSEL	Number of Db2 BLOB read requests
492	1EC	Integer	4	LMSINS	Number of Db2 BLOB insert requests
496	1F0	Integer	4	LMSUPD	Number of Db2 BLOB update requests
500	1F4	Integer	4	LMSDEL	Number of Db2 BLOB delete requests
504	1F8	Integer	4	LMSLIS	Number of Db2 BLOB list requests
508	IFC	64 bit integer	8	LMSSTCUW	Total elapsed time for all thread read BLOB requests

Table 36. Db2 statistics record (Q5ST) (continued)

Offset: Dec	Offset: Hex	Type	Len	Name	Description
516	204	64 bit integer	8	LMSSTMXW	Maximum elapsed time for a thread read BLOB request
524	20C	64 bit integer	8	LMSSSCUW	Total elapsed time for all SQL read BLOB requests
532	214	64 bit integer	8	LMSSSMXW	Maximum elapsed time for an SQL read BLOB request
540	21C	64 bit integer	8	LMSITCUW	Total elapsed time for all thread insert BLOB requests
548	224	64 bit integer	8	LMSITMXW	Maximum elapsed time for a thread insert BLOB request
556	22C	64 bit integer	8	LMSISCUW	Total elapsed time for all SQL insert BLOB requests
564	234	64 bit integer	8	LMSISMXW	Maximum elapsed time for an SQL insert BLOB request
572	23C	64 bit integer	8	LMSUTCUW	Total elapsed time for all thread update BLOB requests
580	244	64 bit integer	8	LMSUTMXW	Maximum elapsed time for a thread update BLOB request
588	24C	64 bit integer	8	LMSUSCUW	Total elapsed time for all SQL update BLOB requests
596	254	64 bit integer	8	LMSUSMXW	Maximum elapsed time for an SQL update BLOB request
604	25C	64 bit integer	8	LMSDTCUW	Total elapsed time for all thread delete BLOB requests
612	264	64 bit integer	8	LMSDTMXW	Maximum elapsed time for a thread delete BLOB request
620	26C	64 bit integer	8	LMSDSCUW	Total elapsed time for all SQL delete BLOB requests
628	274	64 bit integer	8	LMSDSMXW	Maximum elapsed time for an SQL delete BLOB request
636	27C	64 bit integer	8	LMSLTCUW	Total elapsed time for all thread list BLOB requests
644	284	64 bit integer	8	LMSLTMXW	Maximum elapsed time for a thread list BLOB request
652	28C	64 bit integer	8	LMSLSCUW	Total elapsed time for all SQL list BLOB requests
660	294	64 bit integer	8	LMSLSMXW	Maximum elapsed time for an SQL list BLOB request

The data contains counts for each request type that the Db2 resource manager supports. For these request types, maximum and cumulative elapse times are kept for the following:

- The time spent in the Db2 resource manager as a whole (called the thread time).

- The time that was spent performing the RRSAF and SQL parts of the request (a subset of the thread time called the SQL time).

Information is also provided for:

- The number of server tasks attached.
- The maximum overall request depth against any of the server tasks.
- The number of times any of the server task requests terminated abnormally.

If the abnormal termination count is not zero, a requeue count is provided indicating the number of queued requests that were requeued to other server tasks as a result of the abnormal termination.

If the average thread time is significantly greater that the average SQL time, this might indicate that thread requests are spending an excessive amount of time waiting for a server task to process the SQL part of the request. If this is the case, examine the DHIGMAX field and, if the value is greater than one, consider increasing the number of Db2 server tasks specified in the QSGDATA parameter of the CSQ6SYSP system parameter macro.

z/OS *Coupling facility manager data records*

Use this topic as a reference to the format of the coupling facility manager data records.

The format of the coupling facility manager statistics record is described in the following table and in assembler macro thlqual. SCSQMACS(CSQDQEST) and C header file thlqual.SCSQC370(CSQDSMFC). The field names in C are all in lowercase, for example qest, qestid.

If the queue manager was not started as a member of a queue sharing group, no data is recorded in this record.

Table 37. Coupling facility statistics record (QEST)

Offset: Dec	Offset: Hex	Type	Len	Name	Description
0	0	Structure	4104	QEST	CF manager statistics
0	0	Bitstring	2	QESTID	Control block identifier
2	2	Integer	2	QESTLL	Control block length
4	4	Character	4	QESTEYEC	Control block eye catcher
8	8	Character	4096	QESTZERO	QEST part cleared on occasion
8	8	Character	64	QESTSTUC (0:63)	Array (one entry per structure)
8	8	Character	12	QESTSTR	Structure name
20	14	Integer	4	QESTSTRN	Structure number
24	18	Integer	4	QESTCSEC	Number of IXLLSTE calls
28	1C	Integer	4	QESTCMEC	Number of IXLLSTM calls
32	20	Character	8	QESTSSTC	Time spent doing IXLLSTE calls
40	28	Character	8	QESTMSTC	Time spent doing IXLLSTM calls
48	30	Integer	4	QESTRSEC	Number of IXLLSTE redrives
52	34	Integer	4	QESTRMEC	Number of IXLLSTM redrives
56	38	Integer	4	QESTSFUL	Number of structure fulls

Table 37. Coupling facility statistics record (QEST) (continued)					
Offset: Dec	**Offset: Hex**	**Type**	**Len**	**Name**	**Description**
60	3C	Integer	4	QESTMNUS	Maximum number of entries in use
64	40	Integer	4	QESTMLUS	Maximum number of elements in use
68	44	Character	4	*	Reserved
4104	1008	Character	0	*	End of control block

The data contains information for each coupling facility list structure, including the CSQ_ADMIN structure, that the queue manager could connect to during the statistics interval. The information for each structure includes the following:

- The number of and cumulative elapsed times for IXLLSTE and IXLLSTM requests.
- The number of times a request had to be retried because of a timeout.
- The number of times a 'structure full' condition occurred.

z/OS *Topic manager data records*

Use this topic as a reference to the format of the topic manager data records.

The format of the topic manager statistics record is described in the following table and in assembler macro thlqual.SCSQMACS(CSQDQTST) and C header file thlqual.SCSQC370(CSQDSMFC). The field names in C are all in lowercase, for example qtst, qtstid.

Table 38. Topic manager statistics record (QTST)					
Offset: Dec	**Offset: Hex**	**Type**	**Len**	**Name**	**Description**
0	0	Structure	96	QTST	Topic manager statistics
0	0	Bitstring	2	QTSTID	Control block identifier
2	2	Integer	2	QTSTLL	Control block length
4	4	Character	4	TESTEYEC	Control block eye catcher
8	8	Character	88	QTSTZERO	QTST part cleared on occasion
8	8	Integer	4	QTSTSTOT	Total subscription requests
12	0C	Integer	4	QTSTSDUR	Durable subscription requests
16	10	Integer	4	QTSTSHIG (1:3)	Subscription high water mark array (API, ADMIN, PROXY)
28	1C	Integer	4	QTSTSLOW (1:3)	Subscription low water mark array (API, ADMIN, PROXY)
40	28	Integer	4	QTSTSEXP	Subscriptions expired
44	2C	Integer	4	QTSTTMSG	Total messages put to Sub queue
48	30	Integer	4	QTSTSPHW	Single publish subscriber high water mark
52	34	Integer	4	QTSTPTOT (1:3)	Total Publication requests (API, ADMIN, PROXY)
64	40	Integer	4	QTSTPTHI	Total publish high water mark

Table 38. Topic manager statistics record (QTST) (continued)					
Offset: Dec	**Offset: Hex**	**Type**	**Len**	**Name**	**Description**
68	44	Integer	4	QTSTPTLO	Total publish low water mark
72	48	Integer	4	QTSTPNOS	Count of publishes to no subscriber
76	4C	Integer	4	*	Reserved
80	50	Bitstring	8	QTSTETHW	Elapse time HW on publish
88	58	Bitstring	8	QTSTETTO	Elapse time total on publish

z/OS *Coupling facility manager SMDS data records*

Use this topic as a reference to the format of the coupling facility manager shared message data set (SMDS) data records.

The format of the coupling facility manager shared message data set (SMDS) statistics record is described in assembler macro `thlqual.SCSQMACS(CSQDQESD)` and in C header file `thlqual.SCSQC370(CSQDSMFC)`.

The statistics provide information about the utilization of the owned shared message data set, I/O activity for the group of shared message data sets, and SMDS buffer utilization.

If the queue manager was not started as a member of a queue sharing group, no data is recorded in this record.

z/OS *Layout of channel initiator SMF type 115 records*

The layout of channel initiator statistics data (SMF type 115, subtype 231) records is described in this topic.

Self-defining section

The self-defining section for the channel initiator statistics data follows the standard SMF header. It is structured in the standard triplet format. The format of the triplets is described in structure `qwsx` in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQWSX)`.

Table 39 on page 312 shows the format of the self-defining section.

Table 39. Structure of the channel initiator statistics self-defining section					
Offset: Dec	**Offset : Hex**	**Type**	**Length**	**Name**	**Description**
0	0	Integer	4	QWSX0PSO	Offset from the start of the SMF record to the first instrumentation standard header (QWHS)
4	4	Integer	2	QWSX0PSL	Length of the QWHS
6	6	Integer	2	QWSX0PSN	Number of instances of QWHS
8	8	Integer	4	QWSX0R1O	Offset from the start of the SMF record to the first channel initiator control information block (QCCT)
12	C	Integer	2	QWSX0R1L	Length of the QCCT
14	E	Integer	2	QWSX0R1N	Number of instances of QCCT

Table 39. Structure of the channel initiator statistics self-defining section (continued)					
Offset: Dec	**Offset : Hex**	**Type**	**Length**	**Name**	**Description**
16	10	Integer	4	QWSX0R2O	Offset from the start of the SMF record to the first dispatcher task block (QCT_DSP)
20	14	Integer	2	QWSX0R2L	Length of the QCT_DSP
22	16	Integer	2	QWSX0R2N	Number of instances of QCT_DSP
24	18	Integer	4	QWSX0R3O	Offset from the start of the SMF record to the first adapter task block (QCT_ADP)
28	1C	Integer	2	QWSX0R3L	Length of the QCT_ADP
30	1E	Integer	2	QWSX0R3N	Number of instances of QCT_ADP
32	20	Integer	4	QWSX0R4O	Offset from the start of the SMF record to the first SSL task block (QCT_SSL)
36	24	Integer	2	QWSX0R4L	Length of the QCT_SSL
38	26	Integer	2	QWSX0R4N	Number of instances of QCT_SSL
40	28	Integer	4	QWSX0R5O	Offset from the start of the SMF record to the first DNS task block (QCT_DNS)
44	2C	Integer	2	QWSX0R5L	Length of the QCT_DNS
46	2E	Integer	2	QWSX0R5N	Number of instances of QCT_DNS

Typically one record contains all the data. If there are a large number of dispatchers, adapters, or SSL tasks, the data is split over more than one record.

If this happens, the count of instances of some type of tasks can be zero, and information about a group of tasks can be spread across multiple records. The channel initiator control information block (QCCT) is only present in the first record. For example the data could be split between two SMF records like this:

Table 40. Example data		
Count	**First record**	**Last record**
QWHS	1	1
QCCT	1	0
QCT_DSP	50	5
QCT_ADP	0	10
QCT_SSL	0	3
QCT_DNS	0	1

This example shows that there were 55 dispatcher TCBs running during the SMF interval.

Instrumentation standard header (QWHS)

The format of the QWHS is described in structure `qwhs` in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQWHS)`. It contains the following key fields that are relevant to channel initiator SMF 115 records:

Table 41. Key fields in the QWHS		
Name	**Length**	**Description**
QWHSNDA	1 byte	Number of self-defining sections
QWHSSSID	4 bytes	Subsystem name
QWHSSMFC	1 bit	Indicates whether there are multiple SMF records containing information for this interval. If this bit is on, information for this interval is continued in further SMF records. If this bit is off, this is the last or only record. The subsystem ID in QWHSSSID, and the SMF interval start time in QWHSTIME, can be used to group multiple records for the same interval.
QWHSTIME	8 bytes	Local time of the start of the interval in STCK format
QWHSDURN	8 bytes	Duration from the start of the interval to the end of the interval in STCK format
QWHSSTCK	8 bytes	End of the interval in UTC in STCK format

z/OS Channel initiator statistics data records

Use this topic as a reference for channel initiator statistics data records.

The format of the channel initiator statistics data record contains two parts:

- The first part is the channel initiator control information block, described in assembler macro `thlqual.SCSQMACS(CSQDQCCT)`. For further information, see "Channel initiator control information block" on page 315.
- The second part is the channel initiator task block, described in assembler macro `thlqual.SCSQMACS(CSQDQCTA)`.

  The channel initiator task block contains information about the four types of task within the CHINIT. For further information, see:

  - "Dispatcher tasks" on page 316
  - "Adapter tasks" on page 317
  - "Domain Name Server (DNS) task" on page 318
  - "SSL tasks" on page 319

  Each task includes:

  - The elapsed time that the task spent processing requests in the interval (*qcteltm*)
  - The CPU time used by the task in the interval, which is made up of CPU used while processing requests and CPU used between requests (*qctcptm*)
  - The total wait time of this task in the interval (*qctwttm*)
  - The number of requests in the interval (*qctreqn*)

  You can use this information to see how busy the task was, and determine whether you need to add more tasks based on the analysis.

  For TLS and DNS tasks, the duration of the longest request (*qctlgdu*, *qctlsdu*) and the time of day when this occurred (*qctlgdm*, *qctlsdm*) are also included.

  These can be useful to identify when channel requests took a long time. For example, a DNS lookup request going to a server outside of your enterprise taking seconds rather than milliseconds.

The CPU time (qctcptm) value includes all CPU consumed by the task, both processing requests and between processing requests. The elapsed time (*qcteltm*) value only includes time while processing requests. This means that the CPU time may be greater than the elapsed time.

The example accounting data in the following tasks has been formatted using IBM MQ SupportPac MP1B.

Both of the parts are also described in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`. Note that the field names in C are all in lowercase, for example, *qcct*, *qct_adp*.

z/OS *Channel initiator control information block*

Use this topic as a reference for the channel initiator control information block.

The channel initiator control information block contains basic information for this CHINIT, including:

- CHINIT job name (*qcctjobn*)
- QSG name if it is in a queue sharing group (*qcctqsgn*)
- Peak number used of current channels (*qcctnocc*)
- Peak number used of active channels (*qcctnoac*)
- MAXCHL - maximum permitted current channels (*qcctmxcc*)
- ACTCHL - maximum permitted active channels (*qcctmxac*)
- TCPCHL - maximum permitted TCP/IP channels (*qcctmxtp*)
- LU62CHL - maximum permitted LU62 channels (*qcctmxlu*)
- V 9.4.0 31-bit storage used by CHINIT in the extended private region (*qcctstus*). This information is also provided by the CSQX004I message in the CHINIT job log.
- V 9.4.0 64-bit storage limit available to the CHINIT (*qcctslim*)
- V 9.4.0 64-bit storage used by CHINIT (*qcctstab*). This information is also provided by the CSQX004I message in the CHINIT job log.

The format of the channel initiator control information block is described in structure `qcct` in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQCCS)`.

You can use this information to see if the number of active channels is approaching the configured maximum value. Note that the number of current and active channels are the values when the record was created. So, between the two intervals there might have been more than this number of channels active.

Channel information from SMF data

Here is an example of channel information from SMF data:

V 9.4.0

```
MV4A,MQ27,2023/10/02,11:53:02,VRM:934,
From 2023/10/02,11:52:52 to 2023/10/02,11:53:02, duration 10 seconds.
Peak number used of current channels..... 1
Peak number used of active channels ..... 1
MAXCHL. Max allowed current channels..... 9999
ACTCHL. Max allowed active channels..... 9999
TCPCHL. Max allowed TCP/IP channels..... 9999
LU62CHL. Max allowed LU62 channels..... 200
31-bit storage used..... 436 MB
64-bit storage limit.....16384 PB
64-bit storage used..... 187 MB
64-bit storage free.....16384 PB
```

You can monitor the storage usage and see whether the value is trending upwards. If the total used is approaching the total storage available, you might be running out of storage, and so might not be able to support many more channels.

If the numbers of active current channels are tending towards the maximum number of channels, you might need to increase the maximum number of channels.

z/OS *Dispatcher tasks*

This topic contains example data for the dispatcher tasks statistics, and information about how to interpret the data.

The format of the dispatcher task block is described in structure `qct_dsp` in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQCTA)`.

Example data

```
Task, Type, Requests, Busy %,  CPU used, CPU %, "avg CPU", "avg ET"
   ,     ,        ,      ,   Seconds,      ,  uSeconds, uSeconds
   0, DISP,    26587,    0.4,  0.592463,   0.1,         22,      127
   1, DISP,    26963,    0.3,  0.588092,   0.1,         22,      112
   2, DISP,   864329,    2.7,  2.545668,   0.3,          3,       28
   3, DISP,    26875,    0.4,  0.590825,   0.1,         22,      120
   4, DISP,    26874,    0.4,  0.603285,   0.1,         22,      123
Summ, DISP,   971628,    0.8,  4.920332,   0.1,          5,       38
```

The example data shows that there were five dispatchers. A channel is associated with a dispatcher when it starts. The channel initiator tries to distribute work across all the dispatchers when allocating a channel to a dispatcher. This example shows that one dispatcher is processing more requests than other dispatchers. This is normal, as some channels might stop, so the dispatcher is processing fewer channels, and some channels can be busier than others.

- 4.9 seconds of CPU were used by the dispatchers.
- The average request used 5 microseconds of CPU and took 38 microseconds elapsed time.
- A dispatcher is used to send and receive data over a communications network, and this is not usually dependent on external events. The average elapsed time should, therefore, be close to the average CPU time used. The CPU time (*qctcptm*) value includes all CPU consumed by the task, both processing requests and between processing requests.

  The elapsed time (*qcteltm*) value only includes time while processing requests. This means that the CPU time may be greater than the elapsed time. If the CHINIT is delayed due to lack of CPU, then the ratio of average elapsed time to average CPU time is much larger, compared to when the CHINIT is not delayed for CPU.
- The average CPU used per request depends on the message traffic. For example, bigger messages use more CPU than smaller messages.

The fields are calculated from:

- Duration: qwhs.qwhsdurn
- Requests : qctreqn
- Busy %: qcteltm and duration
- CPU used: qctcptm
- CPU %: qctcptm and duration
- Average CPU: qctcptm and qctreqn
- Average ET: qcteltm and qctreqn

Usually, the number of dispatchers should be less than, or equal to, the number of processors in the LPAR. If you have more dispatchers than processors in the LPAR they might compete for CPU resources. For more information about tuning your system, see SupportPac MP16.

Channels have an affinity to a dispatcher, so you might find that some dispatchers process many more requests than another dispatcher.

You can use the ALTER QMGR CHIDISPS() command to change the number of dispatchers used. Any change comes into effect the next time the channel initiator is started.

z/OS *Adapter tasks*

This topic contains example data for the adapter tasks statistics, and information about how to interpret the data.

The format of the adapter task block is described in structure `qct_adp` in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQCTA)`.

Example data

```
Task, Type, Requests, Busy %,  CPU used, CPU %, "avg CPU", "avg ET"
   ,    ,        ,      ,    Seconds,      ,  uSeconds, uSeconds
   0, ADAP,   470297,   10.2, 41.290670,   4.6,        88,      194
   1, ADAP,    13907,    0.6,  1.589428,   0.2,       114,      365
   2, ADAP,     2517,    0.2,  0.185325,   0.0,        74,      746
   3, ADAP,     1095,    0.1,  0.085774,   0.0,        78,      907
   4, ADAP,      535,    0.1,  0.040743,   0.0,        76,      947
   5, ADAP,      220,    0.0,  0.016228,   0.0,        74,     1175
   6, ADAP,       82,    0.0,  0.005521,   0.0,        67,     1786
   7, ADAP,       80,    0.0,  0.004248,   0.0,        53,     1160
Summ, ADAP,   488733,    1.4, 43.217938,   0.6,        88,      205
```

The fields are calculated from:

- Duration: *qwhs.qwhsdurn*
- Requests: *qctreqn*
- Busy %: *qcteltm* and duration
- CPU used: *qctcptm*
- CPU %: *qctcptm* and duration
- Average CPU: *qctcptm* and *qctreqn* average
- ET: *qcteltm* and *qctreqn*

This example shows that there were eight adapter tasks.

Adapter number 0

- Processed the majority of the requests (470297 out of 488733)
- Was busy 10.2% of the interval
- Used 41.3 seconds of CPU

Overall

The average CPU per request was 88 microseconds of CPU and took 205 microseconds

The adapters process IBM MQ requests. Some of these requests might wait, for example, for log I/O during a commit, so the average Elapsed Time per request has little meaning.

The CPU time (*qctcptm*) value includes all CPU consumed by the task, both processing requests and between processing requests. The elapsed time (*qcteltm*) value only includes time while processing requests. This means that the CPU time may be greater than the elapsed time.

When an IBM MQ request is made the first free adapter task is used.

- If there is at least one adapter that has been little used (less than 1%) busy, you have enough adapters.
- If at least one adapter was not used, you have enough adapters defined.
- If all the adapters were used, you might need to allocate more adapters.
- If all of the adapters were used, and they were all busy for most of the interval, you need to allocate more adapters.

You can use the ALTER QMGR CHIADAPS() command to change the number of adapters used. Any changes come into effect the next time the channel initiator is started.

Attention: If there are too many adapters acting on a small set of queues, you might get contention within the queue manager.

Related reference

ALTER QMGR

z/OS *Domain Name Server (DNS) task*

This topic contains example data for the DNS tasks statistics, and information about how to interpret the data.

The format of the DNS task block is described in structure `qct_dns` in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQCTA)`.

Example data

```
Task, Type, Requests, Busy %,  CPU used, CPU %, "avg CPU", "avg ET",  longest,
date,            time
    ,     ,         ,       ,   Seconds,      ,  uSeconds, uSeconds, uSeconds,           ,
   0,  DNS,    14002,    0.0,  0.122578,   0.0,          9,       11,       463, 2014/03/18,
12:56:33.987671
Summ,  DNS,    14002,    0.0,  0.122578,   0.0,          9,       11,       463, 2014/03/18,
12:56:33.987671
```

The channel initiator uses a single DNS task. The example shows that the task processed 14002 requests and on average the request used 9 microseconds of CPU and took 11 microseconds of elapsed time.

The longest DNS request took 463 microseconds elapsed time, and this occurred at 12:56:33 local time.

The fields are calculated from:

- Duration: *qwhs.qwhsdurn*
- Requests : *qctreqn*
- Busy %: *qcteltm* and duration
- CPU used: *qctcptm*
- CPU %: *qctcptm* and duration
- Average CPU: *qctcptm* and *qctreqn*
- Average ET: *qcteltm* and *qctreqn*
- Longest: *qctlgdu*
- Longest at: *qctlgtm*

The DNS task can go out of your enterprise to look up the IP address associated with a name. If the average Elapsed time is significantly more than the average CPU time used, you might have some long requests.

If the value of the longest request time is unacceptable you should work with your network team to investigate why you are having long requests. It might be that you have an invalid name in your connections.

If the DNS task is busy for 25% of the duration, consider investigating the cause further.

The CPU time (*qctcptm*) value includes all CPU consumed by the task, both processing requests and between processing requests. The elapsed time (*qcteltm*) value only includes time while processing requests. This means that the CPU time might be greater than the elapsed time.

Note: There are requests to the DNS task that are not DNS lookups, so you might have the number of requests being greater than zero - but no longest request information.

z/OS *SSL tasks*

This topic contains example data for the SSL tasks statistics, and information about how to interpret the data.

The format of the SSL task block is described in structure `qct_ssl` in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQCTA)`.

Example data

```
Task, Type, Requests, Busy %,  CPU used, CPU %, "avg CPU", "avg ET",  longest,
date,            time
   ,     ,         ,       ,   Seconds,      ,  uSeconds, uSeconds, uSeconds,           ,
   0,  SSL,     3112,    1.2,  0.248538,   0.3,         80,      362,     8864, 2014/03/18,
12:46:40.237697
   1,  SSL,     3070,    1.2,  0.245433,   0.3,         80,      359,     4714, 2014/03/18,
12:46:18.938022
   2,  SSL,     3170,    1.2,  0.255557,   0.3,         81,      362,     7273, 2014/03/18,
12:46:35.358145
   3,  SSL,     3060,    1.2,  0.246542,   0.3,         81,      365,    13164, 2014/03/18,
12:46:44.514045
   4,  SSL,     3120,    1.3,  0.251927,   0.3,         81,      373,    22438, 2014/03/18,
12:46:22.134123
Summ,  SSL,     15532,   1.2,  1.247998,   0.3,         80,      364,    22438, 2014/03/18,
12:46:22.134123
```

This example data shows that the average request took 364 microseconds. The longest request was for SSL task 4, took 22,438 microseconds, and occurred at 12:46:22.134123 local time.

The fields are calculated from:

- Duration: *qwhs.qwhsdurn*
- Requests : *qctreqn*
- Busy %: *qcteltm* and duration
- CPU used: *qctcptm*
- CPU %: *qctcptm* and duration
- Average CPU: *qctcptm* and *qctreqn*
- Average ET: *qcteltm* and *qctreqn*
- Longest: *qctlsdu* longest at: *qctlstm*

A running channel is associated with an SSL task, in a similar way that a channel is associated with a dispatcher. The SSL tasks can use the cryptographic coprocessors available to the LPAR. So, the elapsed time can include time spent on a coprocessor. You should monitor the average elapsed time throughout the day. If this time increases significantly during peak periods you should work with your z/OS systems programmers, as your coprocessors might be over-used.

If the SSL tasks are busy for a significant proportion of the interval, increasing the number of SSL tasks might help. If the SSL tasks are waiting for external resources such as a coprocessor, increasing the number of SSL tasks has little effect.

You can use the ALTER QMGR SSLTASKS() command to change the number of SSL tasks used. Any changes come into effect the next time the channel initiator is started.

The CPU time (*qctcptm*) value includes all CPU consumed by the task, both processing requests and between processing requests. The elapsed time (*qcteltm*) value only includes time while processing requests. This means that the CPU time might be greater than the elapsed time.

Related reference

ALTER QMGR

V 9.4.0 z/OS *Queue data records*

Use this topic as a reference for queue (SMF type 115, subtype 216) data records. The statistics are designed to make it easier for you to monitor usage and performance of your queue over time, and give an insight into what happened with your queue during the last SMF interval. This includes all the DISPLAY QSTATUS information and information on message flow, expiry, high and low watermarks and more.

The format of the queue statistics data record is described in assembler macro `thlqual.SCSQMACS(CSQDQQST)`.

The queue statistics record contains information on the performance of selected queues and includes the following fields:

QQSTID – Control block identifier
The identifier for the queue statistics control block; is always `x'D80F'`.

QQSTLL – Length of control block
The length of a queue statistics record.

QQSTEYEC – Control block eyecatcher
The eyecatcher used to make identification of the control block easier; is always `'QQST'`

QQSTQNAM – Queue name
The name of the queue.

QQSTFLAG
An array of bits containing the following information about the queue:

QQSTDISP – Queue disposition
This bit identifies whether the queue is of private or shared disposition. If the bit is on, then it is a shared queue.

QQSTPART – Partial record identifier
This bit identifies whether the record is a full or partial record. If the bit is on, then it is a partial record. When this flag is set there was an issue accessing the information on the queue, for example if there is a CF structure failure.

In a shared queue partial record, the accuracy of **qqstdpth**, **qqstmage**, and **qqstuncm** cannot be guaranteed. Therefore, the fields **qqstdpth** and **qqstmage** are populated with `x'00'` and the **qqstuncm** flag is not set.

In a private queue partial record, the accuracy of **qqstmage** cannot be guaranteed, therefore the field is populated with `x'00'`.

QQSTUNCM – Uncommitted changes pending
This bit indicates whether there are any uncommitted changes (puts and gets) pending for the queue. If the bit is on, there are uncommitted changes.

This is checked and set at the time of the SMF data collection and provides the same result as a DISPLAY QSTATUS command would if run at the time the SMF record was generated.

If the queue is a shared queue and QQSTPART is set, this bit is always off as there might have been an issue obtaining the correct value.

QQSTPSID – Page set ID
The page set ID where the queue is located, if allocated and a private queue. If the queue is a shared queue, or a private queue that does not have a page set assigned to it, this field is set to -1 (`x'FFFF'`).

This value is correct at the time the SMF record was generated. It is possible that the page set changed during the SMF interval, in which case, the value reflected in the next SMF record will be the new page set.

QQSTBPID – Buffer pool ID
The buffer pool ID used by the queue, if allocated and a private queue. If the queue is a shared queue, or a private queue that does not have a buffer pool assigned to it, this field is set to -1 (`x'FFFF'`).

This value is correct at the time the SMF record was generated. It is possible that the buffer pool changed during the SMF interval. In this case, the value reflected in the SMF record is the new buffer pool.

QQSTQSGN – QSG name
The Queue Sharing Group name that the queue manager is a member of, if it is a shared queue. If the queue is a private queue this field is blank.

QQSTCFST – CF Structure name
The coupling facility (CF) structure name the queue uses if it is a shared queue. If the queue is a private queue this field is blank.

QQSTDPTH – Current depth
The depth of the queue at the time the SMF data was captured.

If the queue is a shared queue and QQSTPART is set, this value is always zero as there might have been an issue obtaining the correct value.

QQSTOPCT – Current open for output count
The number of handles that are currently open for output for the queue at the time when the SMF data was captured. For shared queues, the number returned applies only to the queue manager generating the record. The number is not the total for all the queue managers in the queue sharing group.

This is the same as OPPROCS from a DISPLAY QSTATUS command.

This is checked and set at the time of the SMF data collection and provides the same result as a DISPLAY QSTATUS command would, if run at the time the SMF record was generated.

QQSTIPCT – Current open for input count
The number of handles that are currently open for input for the queue at the time when the SMF data was captured. For shared queues, the number returned applies only to the queue manager generating the record. The number is not the total for all the queue managers in the queue sharing group.

This is the same as IPPROCS from a DISPLAY QSTATUS command.

This is checked and set at the time of the SMF data collection and provides the same result as a DISPLAY QSTATUS command would, if run at the time the SMF record was generated.

QQSTMAGE – Oldest message age
The age, in seconds, of the oldest message on the queue.

This is checked and set at the time of the SMF data collection and provides the same result as a DISPLAY QSTATUS command would, if run at the time the SMF record was generated.

If QQSTPART is set, this value is always zero as there might have been an issue obtaining the correct value.

QQSTQTST – Short term QTIME
The interval, in microseconds, between messages being put on the queue and then being destructively read. Value based on the last few messages processed. For shared queues, the values shown are for measurements collected on this queue manager only.

This is the same as the first value in QTIME from a DISPLAY QSTATUS command.

This is checked and set at the time of the SMF data collection and provides the same result as a DISPLAY QSTATUS command would, if run at the time the SMF record was generated.

QQSTQTLT – Long term QTIME
The interval, in microseconds, between messages being put on the queue and then being destructively read. The value is based on a larger sample of the recently processed messages. For shared queues, the values shown are for measurements collected on this queue manager only.

This is the same as the second value in QTIME from a DISPLAY QSTATUS command.

This is checked and set at the time of the SMF data collection and provides the same result as a DISPLAY QSTATUS command would, if run at the time the SMF record was generated.

QQSTLPUT – Last put date/time
The time, in store clock format, at which the last message was put to the queue since the queue manager started. For shared queues, the value shown is for messages put by this queue manager only.

This is the same as LPUTDATE and LPUTTIME from a DISPLAY QSTATUS command.

This is checked and set at the time of the SMF data collection and provides the same result as a DISPLAY QSTATUS command would, if run at the time the SMF record was generated.

QQSTLGET – Last get date/time
The time, in store clock format, at which the last message was retrieved from the queue since the queue manager started. For shared queues, the value shown is for messages put by this queue manager only.

A message being browsed does not count as a message being retrieved.

This is the same as LGETDATE and LGETTIME from a DISPLAY QSTATUS command.

This is checked and set at the time of the SMF data collection and provides the same result as a DISPLAY QSTATUS command would, if run at the time the SMF record was generated.

QQSTDPHI – Highest depth
The highest depth reached by the queue during the SMF interval.

For shared queues, queue managers only have partial information about the change in depth of the queue over time. The QQSTDPHI value is based off this partial information as follows:

- At the start of the interval the value of QQSTDPHI is set to zero.
- When an application puts a message to the queue in the interval the queue manager checks the depth of the queue, including the message just being put. If this value is higher than the current value of QQSTDPHI, then it is used as the new value of QQSTDPHI.
- When SMF data for the queue is collected, the queue manager will check if the current queue depth is higher than QQSTDPHI, if so the current queue depth is used as the new value of QQSTDPHI.

This approach means that the value of QQSTDPHI does not take into account messages put by other queue managers in the queue sharing group, unless those messages contributed to the queue depth at the point where SMF data is collected.

QQSTDPLO – Lowest depth
The lowest depth reached by the queue during the SMF interval.

For shared queues, queue managers only have partial information about the change in depth of the queue over time. The QQSTDPLO value is based off this partial information as follows:

- At the start of the interval the value of QQSTDPLO is set to a special value.
- The first time during the interval the queue manager obtains the depth of the queue, QQSTDPLO to that value.
- When an application puts a message to the queue in the interval the queue manager checks the depth of the queue, including the message just being put. If this value is lower than the current value of QQSTDPLO, then it is used as the new value of QQSTDPLO.
- When SMF data for the queue is collected, the queue manager will check if the current queue depth is lower than QQSTDPLO, if so the current queue depth is used as the new value of QQSTDPLO.

This approach means that the value of QQSTDPLO does not take into account messages got by other queue managers in the queue sharing group, unless those messages contributed to the queue depth at the point where SMF data is collected.

QQSTPUTS – MQPUT count
The number of messages put to the queue using MQPUT during the SMF interval.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTPUT1 – MQPUT1 count
The number of messages put to the queue using MQPUT1 during the SMF interval.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTNPPT – Non-persistent MQPUT count
The number of non-persistent messages put to the queue using MQPUT during the SMF interval.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTPPT – Persistent MQPUT count
The number of persistent messages put to the queue using MQPUT during the SMF interval.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTNPP1 – Non-persistent MQPUT1 count
The number of non-persistent messages put to the queue using MQPUT1 during the SMF interval.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTPP1 – Persistent MQPUT1 count
The number of persistent messages put to the queue using MQPUT1 during the SMF interval.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTPUTB – MQPUT bytes
The number of bytes of message data, including any message properties, put to the queue using MQPUT during the SMF interval. This does not include message headers in the calculation of the size.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTPT1B – MQPUT1 bytes
The number of bytes of message data, including any message properties, put to the queue using MQPUT1 during the SMF interval. This does not include message headers in the calculation of the size.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTNPPB – Non-persistent MQPUT bytes
The number of bytes of non-persistent message data, including any message properties, put to the queue using MQPUT during the SMF interval. This does not include message headers in the calculation of the size.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTPPB – Persistent MQPUT bytes
The number of bytes of persistent message data, including any message properties, put to the queue using MQPUT during the SMF interval. This does not include message headers in the calculation of the size.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTNP1B – Non-persistent MQPUT1 bytes
The number of bytes of non-persistent message data, including any message properties, put to the queue using MQPUT1 during the SMF interval. This does not include message headers in the calculation of the size.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTP1B – Persistent MQPUT1 bytes
The number of bytes of persistent message data, including any message properties, put to the queue using MQPUT1 during the SMF interval. This does not include message headers in the calculation of the size.

For shared queues, the count only includes messages put through the queue manager that generated the SMF record.

QQSTFLPT – Failed MQPUT count
The number of MQPUT calls targeting the queue, which failed with a completion code of MQCC_FAILED, during the SMF interval.

For shared queues, the count only includes failed puts attempted through the queue manager that generated the SMF record.

QQSTFLP1 – Failed MQPUT1 count
The number of MQPUT1 calls targeting the queue, which failed with a completion code of MQCC_FAILED, during the SMF interval.

For shared queues, the count only includes failed puts attempted through the queue manager that generated the SMF record.

QQSTFPTC – Fast puts to a waiting getter count
The number of MQPUT and MQPUT1 calls targeting the queue, that were fast put to a waiting getter during the SMF interval.

Note: *Put to a waiting getter* is a technique whereby a message might not actually be put onto a queue if there is an application already waiting to get the message. Certain conditions must be satisfied for this to occur, in particular the message must be non-persistent and the putting and getting application must be processing the message outside syncpoint control.

If these conditions are met, then the message is transferred from the putting application's buffer into the getting application's buffer without actually touching the IBM MQ queue. This removes a lot of processing involved in putting the message on the queue and therefore leads to increased throughput and lower CPU costs.

QQSTFPTB – Fast puts to a waiting getter bytes
The number of message and properties bytes from MQPUT and MQPUT1 calls targeting the queue, that were fast put to a waiting getter during the SMF interval. This does not include message headers in the calculation of the size.

QQSTSTRM – Streamed message count
The number of messages that were successfully streamed from the queue during the interval. This is always zero if STREAMQ is not set for the queue.

QQSTMSMI – Minimum message size put
The minimum message size, in bytes, put to the queue during the SMF interval.

This includes message and properties bytes, and does not include message headers such as the MQMD.

QQSTMSMA – Maximum message size put
The maximum message size, in bytes, put to the queue during the SMF interval.

This includes message and properties bytes, and does not include message headers such as the MQMD.

QQSTMSAV – Average message size put
The average message size, in bytes, put to the queue during the SMF interval.

This includes message and properties bytes, and does not include message headers such as the MQMD.

QQSTGETS – Destructive MQGET count
The number of messages got from the queue using destructive MQGET during the SMF interval.

For shared queues, the count only includes messages got through the queue manager that generated the SMF record.

QQSTNPDG – Non-persistent destructive MQGET count
The number of non-persistent messages got from the queue using destructive MQGET during the SMF interval.

For shared queues, the count only includes messages got through the queue manager that generated the SMF record.

QQSTPDG – Persistent destructive MQGET count
The number of persistent messages got from the queue using destructive MQGET during the SMF interval.

For shared queues, the count only includes messages got through the queue manager that generated the SMF record.

QQSTGETB – Destructive MQGET byte count
The number of message and properties bytes got from the queue using destructive MQGET during the SMF interval.

For shared queues, the count only includes messages got through the queue manager that generated the SMF record.

QQSTNPDB – Non-persistent destructive MQGET bytes
The number of non-persistent message and properties bytes got from the queue using destructive MQGET during the SMF interval.

For shared queues, the count only includes messages got through the queue manager that generated the SMF record.

QQSTPDB – Persistent destructive MQGET bytes
The number of persistent message and properties bytes got from the queue using destructive MQGET during the SMF interval.

For shared queues, the count only includes messages got through the queue manager that generated the SMF record.

QQSTBRWS – Non-destructive MQGET count
The number of messages browsed from the queue during the SMF interval.

For shared queues, the count only includes messages browsed through the queue manager that generated the SMF record.

QQSTNPBR – Non-persistent non-destructive MQGET count
The number of non-persistent messages browsed from the queue during the SMF interval.

For shared queues, the count only includes messages browsed through the queue manager that generated the SMF record.

QQSTPBR – Persistent non-destructive MQGET count
The number of persistent messages browsed from the queue during the SMF interval.

For shared queues, the count only includes messages browsed through the queue manager that generated the SMF record.

QQSTBRWB – Non-destructive MQGET bytes
The number of message and properties bytes browsed from the queue during the SMF interval.

For shared queues, the count only includes messages browsed through the queue manager that generated the SMF record.

QQSTNPBB – Non-persistent non-destructive MQGET bytes
The number of non-persistent message and properties bytes browsed from the queue during the SMF interval.

For shared queues, the count only includes messages browsed through the queue manager that generated the SMF record.

QQSTPBB – Persistent non-destructive MQGET bytes
The number of persistent message and properties bytes browsed from the queue during the SMF interval.

For shared queues, the count only includes messages browsed through the queue manager that generated the SMF record.

QQSTFLGT – Failed destructive MQGET count
The number of destructive MQGET calls targeting the queue, that failed with a completion code of MQCC_FAILED, during the SMF interval.

Also included in the count is MQCC_WARNING when accompanied by a return code of MQRC_TRUNCATED_MSG_FAILED. However, not included in this count is any MQGET with a wait that receives MQRC_NO_MSG_AVAILABLE.

For shared queues, the count only includes failed MQGET attempts through the queue manager that generated the SMF record.

QQSTNMAG – Failed destructive MQGET with MQRC_NO_MSG_AVAILABLE count
The number of destructive MQGET calls, without wait, targeting the queue, that failed with both a completion code of MQCC_FAILED and a return code of MQRC_NO_MSG_AVAILABLE, during the SMF interval.

This value is a subset of QQSTFLGT.

For shared queues, the count only includes failed MQGET, without wait, attempts through the queue manager that generated the SMF record.

QQSTTMFB – Failed destructive MQGET with MQRC_TRUNCATED_MSG_FAILED count
The number of destructive MQGET calls targeting the queue, that failed with both a completion code of MQCC_WARNING and a return code of MQRC_TRUNCATED_MSG_FAILED, during the SMF interval.

This value is a subset of QQSTFLGT.

For shared queues, the count only includes failed MQGET attempts through the queue manager that generated the SMF record.

QQSTFLGW – No message available for destructive MQGET with a wait count
The number of times that there is no message available for destructive MQGET calls with a wait, targeting the queue, causing the MQGET to continue waiting, during the SMF interval.

QQSTRDGW – Re-driven destructive MQGET with a wait count
The number of times that destructive MQGET calls with a wait are re-driven to check if there is a message on the queue that matches their criteria, during the SMF interval.

When a new message arrives on the queue, all eligible waiting MQGET calls are woken up to attempt to get the message. Each MQGET with a wait that is woken up to check for a valid message increments this count by one. If any of these MQGET with wait fails to get the message, QQSTFLGW is incremented by one and the MQGET goes back into a waiting state.

QQSTFLBW – No message available for non-destructive MQGET with a wait count
The number of times that there is no message available for non-destructive MQGET calls with a wait, targeting the queue, causing the MQGET to continue waiting, during the SMF interval.

QQSTRDBW – Re-driven non-destructive MQGET with a wait count
The number of times that non-destructive MQGET calls with a wait are re-driven to check if there is a message on the queue that matches their criteria, during the SMF interval.

When a new message arrives on the queue, all eligible waiting MQGET calls are woken up to attempt to browse the message. Each MQGET with a wait that is woken up to check for a valid message increments this count by one. If any of these MQGET with wait fails to browse the message, QQSTFLBW is incremented by one and the MQGET goes back into a waiting state.

QQSTSAGT – Destructive MQGET with MQRC_SIGNAL_REQUEST_ACCEPTED count
The number of destructive MQGET calls targeting the queue, that complete with both a completion code of MQCC_WARNING and a return code of MQRC_SIGNAL_REQUEST_ACCEPTED, during the SMF interval.

QQSTSABR – Non-destructive MQGET with MQRC_SIGNAL_REQUEST_ACCEPTED count
The number of non-destructive MQGET calls targeting the queue, that complete with both a completion code of MQCC_WARNING and a return code of MQRC_SIGNAL_REQUEST_ACCEPTED, during the SMF interval.

QQSTIPHI – High watermark for IPPROC
The highest number of concurrent input handles open on the queue during the SMF interval.

For shared queues, the watermark only includes handles owned through the queue manager that generated the SMF record.

QQSTIPLO – Low watermark for IPPROC
The lowest number of concurrent input handles open on the queue during the SMF interval.

For shared queues, the watermark only includes handles owned through the queue manager that generated the SMF record.

QQSTOPHI – High watermark for OPPROC
The highest number of concurrent output handles open on the queue during the SMF interval.

For shared queues, the watermark only includes handles owned through the queue manager that generated the SMF record.

QQSTOPLO – Low watermark for OPPROC
The lowest number of concurrent output handles open on the queue during the SMF interval.

For shared queues, the watermark only includes handles owned through the queue manager that generated the SMF record.

QQSTOPEN – Successful MQOPEN count
The number of times the queue was successfully opened during the SMF interval. This does not include opens performed as part of an MQPUT1 call.

For shared queues, the count only includes when the queue is opened through the queue manager that generated the SMF record.

QQSTCLOS – MQCLOSE count
The number of times the queue was successfully closed using MQCLOSE, during the SMF interval.

For shared queues, the count only includes when the queue is closed through the queue manager that generated the SMF record.

QQSTINQR – MQINQ count
The number of MQINQ calls that completed with a completion code of MQCC_OK or MQCC_WARNING, during the SMF interval.

QQSTSET – MQSET count
The number of MQSET calls that completed with a completion code of MQCC_OK during the SMF interval.

QQSTEXPR – Expired messages count
The number of expired messages cleared from the queue during the SMF interval.

This includes messages expired by an application issuing an MQGET, by the expired message scanning task or by a REFRESH QMGR TYPE(EXPIRY) command.

QQSTRBPT – Rolled back MQPUT counts
The number of messages that were put to the queue, that have been rolled back off the queue, during the SMF interval.

QQSTRBGT – Rolled back MQGET counts
The number of messages destructively read from the queue, that have been rolled back onto the queue, during the SMF interval.

z/OS Interpreting IBM MQ for z/OS accounting data

IBM MQ for z/OS accounting data is written as SMF type 116 records. Use this topic as a reference to the different types of accounting data records.

IBM MQ accounting information can be collected for the following subtypes:

0
Message manager accounting records (how much processor time was spent processing IBM MQ API calls and the number of MQPUT and MQGET calls). This information is produced when a named task disconnects from IBM MQ, and so the information contained within the record might cover many hours.

1
Accounting data for each task, at thread and queue level.

2
Additional queue-level accounting data (if the task used more queues than could fit in the subtype 1 record).

10
Accounting data for channels.

Note: Accounting information for specific channels can be enabled or suppressed by the **STATCHL** channel attribute, and the **STATACLS** queue manager attribute.

Note that:

- Subtype 0 records are produced with accounting trace class 1.
- Subtype 1 and 2 records are produced with accounting trace class 3.
- Subtype 10 records are produced with accounting trace class 4.

z/OS *Layout of an SMF type 116 record*

Use this topic as a reference to the format of an SMF type record.

The standard layout for SMF records involves three parts:

SMF header
Provides format, identification, and time and date information about the record itself.

Self-defining section
Defines the location and size of the individual data records within the SMF record.

Data records
The actual data from IBM MQ that you want to analyze.

For more information about SMF record formats, see *z/OS MVS System Management Facilities (SMF)*.

The SMF header

Table 42 on page 328 shows the format of SMF record header (SM116).

Table 42. SMF record header description

Offset : Dec	Offset: Hex	Type	Len	Name	Description	Example
0	0	Structure	28	SM116	SMF record header.	
0	0	Integer	2	SM116LEN	SMF record length.	01A4
2	2		2		Reserved.	
4	4	Integer	1	SM116FLG	System indicator.	5E
5	5	Integer	1	SM116RTY	Record type. The SMF record type, for IBM MQ accounting records this is always 116 (X'74').	74
6	6	Integer	4	SM116TME	Time when SMF moved record.	00356124
10	A	Integer	4	SM116DTE	Date when SMF moved record.	0100223F
14	E	Character	4	SM116SID	z/OS subsystem ID. Defines the z/OS subsystem on which the records were collected.	D4E5F4F1 (MV41)
18	12	Character	4	SM116SSI	IBM MQ subsystem ID.	D4D8F0F7 (MQ07)
22	16	Integer	2	SM116STF	Record subtype.	0000
24	18	Character	3	SM116REL	IBM MQ version.	F9F3F0 (930)
27	1B		1		Reserved.	
28	1C	Character	0	SM116END	End of SMF header and start of self-defining section.	

Note: The (hexadecimal) values in the right-hand column relate to Figure 22 on page 330.

Self-defining sections

A self-defining section of an SMF record tells you where to find an accounting record, how long it is, and how many times that type of record is repeated (with different values). The self-defining sections follow the header, at a fixed offset from the start of the SMF record.

Each self-defining section points to accounting related data. Table 43 on page 329 summarizes the offsets from the start of the SMF record header.

Table 43. Offsets to self-defining sections

Record subtype (SMF116STF)	Source of accounting data	Offset of self-defining section		See...
		Dec	Hex	
All	Common header	28	X'1C'	“Common IBM MQ SMF header” on page 331
0	Message manager	44	X'2C'	“Message manager data records” on page 333
1	Thread identification record	36	X'24'	“Thread-level and queue-level data records” on page 334
1	Thread-level accounting	44	X'2C'	“Thread-level and queue-level data records” on page 334
1	Queue-level accounting	52	X'34'	“Thread-level and queue-level data records” on page 334. This section is present only if the WTASWQCT field in the task-related information (WTAS) structure is non-zero.
2	Thread identification record	36	X'24'	“Thread-level and queue-level data records” on page 334
2	Queue-level accounting	44	X'2C'	“Thread-level and queue-level data records” on page 334
10	Channel accounting			“Channel accounting data records” on page 337

Note: Other self-defining sections refer to data for IBM use only.

Each self-defining section is two fullwords long and has this format:

```
ssssssssllllnnnn
```

where:

ssssssss
Fullword containing the offset from start of the SMF record.

llll
Halfword giving the length of this data record.

nnnn
Halfword giving the number of data records in this SMF record.

Figure 22 on page 330 shows an example of part of an SMF type 116 record. The numbers in the left-hand column represent the offset, in hexadecimal, from the start of the record. Each line corresponds to sixteen bytes of data, where each byte is two hexadecimal characters, for example 0C. The characters in the right-hand column represent the printable characters for each byte. Non-printable characters are shown by a period (.) character.

In this example, alternate fields in the SMF header are underlined to help you to see them; refer to Table 42 on page 328 to identify them. The self defining section for one of the message manager accounting data records (at the offset given in Table 43 on page 329) is shown in **bold**.

```
000000 01A40000 5E740035 61240100 223FD4E5  *....;.../.....MV*
000000 F4F1D4D8 F0F70000 F6F0F000 00000134  *41MQ07..600.....*
000000 00700001 00000054 00B00001 00000104  *.................*
000000 00300001 00000000 00000000 00000000  *.................*
000000 00000000 00000000 00000000 00000000  *.................*
```

Figure 22. Part of an SMF record 116 showing the header and self-defining sections

The self-defining section for the type of message manager accounting data is located at offset X'2C' from the start of the SMF record and contains this information:

- The offset of the message manager accounting data is located X'00000104' bytes from the start of the SMF record.
- This message manager record is X'0030' bytes long.
- There is one record (X'0001').

Note: Always use offsets in the self-defining sections to locate the accounting records.

z/OS *Processing type 116 SMF records*

Use this topic as a reference to the format of the processing type accounting record.

Any accounting data you collect from SMF must be processed to extract useful information. When you process the data, verify that the records are from IBM MQ and that they are the records you are expecting.

Validate the value of the following fields:

- SM116RTY, the SMF record number = X'74' (116)
- SM116STF, the record subtype, must be 0000, 0001, 0002, or 0010

Reading from the active SMF data sets (or SMF logstreams) is not supported. You must use the SMF program IFASMFDP (or IFASMFDL if logstreams are being used) to dump SMF records to a sequential data set so that they can be processed. For more information see “Using System Management Facility” on page 288.

Details of the structures and fields can be found in IBM MQ SupportPac MP1B.

There is a C sample program called CSQ4SMFD which prints the contents of SMF type 115 and 116 records from the sequential data set. The program is provided as source in thlqual.SCSQC37S and in executable format in thlqual.SCSQLOAD. Sample JCL is provided in thlqual.SCSQPROC(CSQ4SMFJ).

You need to update the SMFIN DD card with the name of the SMF data set. Use the z/OS command '/D SMF' to show the name of the data set, and you need to update the DUMPOUT DD card with the name for the output data set.

You also need to specify the START and END times that you require.

The following sample JCL extracts SMF records from SMF data sets and dumps them to the SMFOUT data set:

```
//SMFDUMP EXEC PGM=IFASMFDP,REGION=0M
//SYSPRINT DD SYSOUT=
//SMFIN  DD DSN=xxxxxx.MANA,DISP=SHR
//SMFOUT  DD DSN=xxxxxx.SMFOUT,SPACE=(CYL,(1,1)),DISP=(NEW,CATLG)
//SYSIN DD *
INDD(SMFIN,OPTIONS(DUMP))
OUTDD(SMFOUT,TYPE(116))
OUTDD(SMFOUT,TYPE(115))
START(1159) END(1210)
/*
```

The following sample JCL extracts SMF records from the SMF log stream named in LSNAME and dumps them to the SMFOUT data set:

```
//SMFDUMP EXEC PGM=IFASMFDL,REGION=0M
//SYSPRINT DD SYSOUT=*
//SMFOUT  DD DSN=xxxxxx.SMFOUT,SPACE=(CYL,(1,1)),DISP=(NEW,CATLG)
//SYSIN DD *
LSNAME(IFASMF.MQ,OPTIONS(DUMP))
OUTDD(SMFOUT,TYPE(116))
OUTDD(SMFOUT,TYPE(115))
START(1159) END(1210)
/*
```

z/OS *Common IBM MQ SMF header*

Use this topic as a reference to the common IBM MQ SMF header type accounting record.

The format of this record is described in Table 44 on page 331 and in assembler macros thlqual.SCSQMACS(CSQDQWHS) and thlqual.SCSQMACS(CSQDQWHC), and C header file thlqual.SCSQC370(CSQDSMFC). The field names in C are all in lowercase, for example qwhs, qwhsnsda.

The QWHS data includes the subsystem name. For subtype 1 records, it also shows whether there are queue-level accounting records present. If the QWHSNSDA field is 3 or less, there are not, and the corresponding self-defining section (at offset X'34') is not set.

The QWHC data gives you information about the user (for example, the user ID (QWHCAID) and the type of application (QWHCATYP)). The QWHC section is completed only for subtype 0 records. The equivalent information is present in the thread identification record for subtype 1 and 2 records.

Table 44. Structure of the common IBM MQ SMF header record QWHS

Offset: Dec	Offset : Hex	Type	Length	Name	Description
0	0	Structure	128	QWHS	
0	0		6		Reserved
6	6	Character	1	QWHSNSDA	Number of self defining sections in the SMF records
7	7		5		Reserved
12	C	Character	4	QWHSSSID	Subsystem name
16	10		24		Reserved
40	28	Character	8	QWHCAID	User ID associated with the z/OS job
48	30	Character	12	QWHCCV	Thread cross-reference
60	3C	Character	8	QWHCCN	Connection name
68	44		8		Reserved

Table 44. Structure of the common IBM MQ SMF header record QWHS (continued)					
Offset: Dec	**Offset : Hex**	**Type**	**Length**	**Name**	**Description**
76	4C	Character	8	QWHCOPID	User ID associated with the transaction
84	54	Integer	4	QWHCATYP	Type of connecting system (1=CICS, 2=Batch or TSO, 3=IMS control region, 4=IMS MPP or BMP, 5=Command server, 6=Channel initiator, 7=RRS Batch)
88	58	Character	22	QWHCTOKN	Accounting token set to the z/OS accounting information for the user
110	6E	Character	16	QWHCNID	Network identifier
126	7E		2		Reserved

z/OS *Combining CICS and IBM MQ performance data*

Use this topic as a reference to the combination of IBM MQ and CICS performance data.

The common IBM MQ SMF header type accounting record section, QWHCTOKN, is used to correlate CICS type 110 SMF records with IBM MQ type 116 SMF records.

CICS generates an LU6.2 unit-of-work token, for each CICS task. The token is used to generate an accounting token that is written to QWHCTOKN in the correlation header of subtype zero records.

Details are also written to the WTIDACCT section in subtype 1 and 2 records. The accounting token enables correlation between CICS and IBM MQ performance data for a transaction.

z/OS *Thread cross-reference data*

Use this topic as a reference to the format of the thread cross-reference type accounting record.

The interpretation of the data in the thread cross-reference (QWHCCV) field varies. This depends on what the data relates to:

- CICS connections (QWHCATYP=1) - see Table 45 on page 332
- IMS connections (QWHCATYP=3 or 4) - see Table 46 on page 332
- Batch connections (QWHCATYP=2 or 7) - this field consists of binary zeros
- Others - no meaningful data

Table 45. Structure of the thread cross-reference for a CICS system				
Offset: Dec	**Offset: Hex**	**Type**	**Length**	**Description**
48	30	Character	4	CICS thread number.
52	34	Character	4	CICS transaction name.
56	38	Integer	4	CICS task number.

Some entries contain blank characters. These apply to the task, rather than to a specific transaction.

Table 46. Structure of the thread cross-reference for an IMS system				
Offset: Dec	**Offset: Hex**	**Type**	**Length**	**Description**
48	30	Character	4	IMS partition specification table (PST) region identifier.

Table 46. Structure of the thread cross-reference for an IMS system (continued)				
Offset: Dec	**Offset: Hex**	**Type**	**Length**	**Description**
52	34	Character	8	IMS program specification block (PSB) name.

z/OS *Message manager data records*

Use this topic as a reference to the format of the message manager accounting records.

The message manager is the component of IBM MQ that processes all API requests. The format of the message manager accounting records is described in assembler macro thlqual.SCSQMACS(CSQDQMAC).

The QMAC data gives you information about the processor time spent processing IBM MQ calls, and counts of the number of MQPUT and MQGET requests for messages of different sizes.

Note: A single IMS application might write two SMF records. In this case, add the figures from both records to provide the correct totals for the IMS application.

Records containing zero processor time

Records are sometimes produced that contain zero processor time in the QMACCPUT field. These records occur when long running tasks identified to IBM MQ either terminate or are prompted to output accounting records by accounting trace being stopped. Such tasks exist in the CICS adapter and in the channel initiator (for distributed queuing). The number of these tasks with zero processor time depends upon how much activity there has been in the system:

- For the CICS adapter, this can result in up to nine records with zero processor time.
- For the channel initiator, the number of records with zero processor time can be up to the sum of `Adapters + Dispatchers + 6`, as defined in the queue manager attributes.

These records reflect the amount of work done under the task, and can be ignored.

z/OS *Sample subtype zero accounting record*

Use this topic as a reference to the format of the subtype zero accounting records.

Figure 23 on page 333 shows a type 116, subtype zero SMF record. In this figure, the SMF record header and the QMAC accounting data record are underlined. The self-defining sections are in bold.

```
000000  01A40000 5E740035 61240100 223FD4E5   *....;.../.....MV*
000010  F4F1D4D8 F0F70000 F6F0F000 00000134    *41MQ07..600.....*
000020  00700001 00000054 00B00001 00000104   *.............*
000030  00300001 00000000 00000000 00000000   *.............*
000040  00000000 00000000 00000000 00000000   *.............*
000050  00000000 B478AB43 9C6C2280 B478AB47   *.........%.....*
000060  9DB47E02 00000000 04C0F631 00000001   *..=.....{6.....*
000070  9880E72D 00000000 014D9540 00000000   *..X.....(. ....*
000080  08480C80 00000010 40404040 40404040   *.......        *
000090  00000000 00000000 00000051 00000000   *.............*
0000A0  00000000 00000000 00000000 00000000   *.............*
0000B0  00000000 00000000 00000000 00000000   *.............*
0000C0  00000000 00000000 00000000 00000000   *.............*
0000D0  00000000 00000000 00000000 00000000   *.............*
0000E0  00000000 00000000 00000000 00000000   *.............*
0000F0  00000000 00000000 00000000 00000000   *.............*
000100  00000000 D4140030 D8D4C1C3 00000000   *....M...QMAC....*
000110  689C738D 00000050 00000000 00000050   *.......&.......&*
000120  0000000A 00000000 00000000 00000000   *.............*
000130  00000000 0024011A 00030710 02DAACF0   *.............0*
```

Figure 23. Example SMF type 116, subtype zero record

z/OS *Thread-level and queue-level data records*

Use this topic as a reference to the format of the thread-level and queue-level accounting records.

Thread level accounting records are collected for each task using IBM MQ. In addition, queue-level accounting records are gathered about each queue that the task opens. A queue-level accounting record is written for each queue that the task has used since the thread-level accounting record was last written.

If the task uses a queue that is configured with a streaming queue, there is no queue-level accounting record for the streaming queue. Instead, the accounting record for the original queue accumulates data for the data points that would have been associated with the streaming queue.

The only exception to this is that the PUTN/PUT1N value shows the number of MQPUT/MQPUT1 requests made by the application, and excludes the extra MQPUT requests made to the streaming queue.

So, for example, if an application issues a single MQPUT request, the:

- PUTN value is 1
- Elapsed time (PUTET) and CPU time (PUTCT) for the MQPUT include the time taken to put to both the primary and streaming queue
- Number of page set requests (PUTPSN) includes those for both the primary and secondary queue, and so on

For each task, data is written to SMF when the task finishes.

From IBM MQ 9.3.0 onwards, for long running tasks, data is also written at the interval specified by either the ACCTIME, or STATIME, parameter of the CSQ6SYSP system parameter macro, or by the system SMF statistics broadcast, provided that the task was running the previous time data was gathered.

Thread-level and queue-level accounting records are produced if you specify class 3 when you start the accounting trace. For example, use the following command:

```
START TRACE(ACCTG) DEST(SMF) CLASS(3)
```

The thread level accounting information is written to an SMF type 116, subtype 1 record, and is followed by queue-level records. If the task opened many queues, further queue information is written to one or more SMF type 116 subtype 2 records. A thread identification control block is included in each subtype 1 and 2 record to enable you to relate each record to the correct task. Typically, the maximum number of queue-level records in each SMF record is about 45.

The format of the thread-level accounting record is described in assembler macro `thlqual.SCSQMACS(CSQDWTAS)`. The format of the queue-level accounting record is described in assembler macro `thlqual.SCSQMACS(CSQDWQ)`. The format of the thread identification record is described in assembler macro `thlqual.SCSQMACS(CSQDWTID)`. All these records are also described in C header file `thlqual.SCSQC370(CSQDSMFC)`. The field names in C are all in lowercase, for example `wtas`, `wtasshex`.

z/OS *Meaning of the channel names*

Use this topic as a reference to the meaning of channel names.

The channel name in the WTID is constructed as shown in the following example. In this example a sender channel exists from queue manager QM1 to queue manager QM2.

The meaning of channel names are described in the following table.

Table 47. Meaning of channel names

Field name	Meaning	Example
For queue manager QM1 the sender channel has the following fields set:		
WTIDCCN	The job name	QM1CHIN
WTIDCHL	The channel name	QM1.QM2

Table 47. Meaning of channel names (continued)		
Field name	**Meaning**	**Example**
WTIDCHLC	This is defined in the CONNAME of the channel	WINMVS2B(2162)
For queue manager QM2 the receiver channel has the following fields set:		
WTIDCCN	The job name	QM2CHIN
WTIDCHL	The channel name	QM1.QM2
WTIDCHLC	Where the channel came from	9.20.101.14

z/OS *Sample subtype 1 and subtype 2 records*

Use this topic as a reference to the format of the subtype 1 and subtype 2 accounting records.

Figure 24 on page 335 and Figure 25 on page 335 show examples of SMF type 116, subtype 1 and subtype 2 records. These two accounting records were created for a batch job that opened 80 queues. Because many queues were opened, a subtype 2 record was required to contain all the information produced.

```
000000  703C0000 5E74002D 983B0100 229FD4E5  *....;.........MV*
000010  F4F1D4D8 F0F70001 F6F0F000 00006FCC  *41MQ07..600...?.*
000020  00700001 0000003C 00D00001 0000010C  *.........}......*
000030  02C00001 000003CC 02400030 F70000D0  *.{....... ..7..}*
000040  E6E3C9C4 00000000 00000000 00000040  *WTID........... *
.
.
.
000100  00000000 00000000 7F4A4BB8 F70102C0  *........"...7..{*
000110  E6E3C1E2 B4802373 0BF07885 7F4AE718  *WTAS.....0..".X.*
```

Figure 24. Example SMF type 116, subtype 1 record

The first self-defining section starts at X'24' and is **bold** in the example; X'0000003C' is the offset to the WTID data record, X'00D0' is the length of the WTID record, and X'0001' is the number of WTID records.

The second self-defining section starts at X'2C' and is in *italic* ; X'0000010C' is the offset to the WTAS data record, X'02C0' is the length of the WTAS record, and X'0001' is the number of WTAS records.

The third self-defining section starts at X'34' and is **bold** in the example; X'000003CC' is the offset to the first WQST data record, X'0240' is the length of the WQST record, and X'0030' is the number of WQST records.

Figure 25 on page 335 shows an example of an SMF type 116, subtype 2 record.

```
000000  49740000 5E74002D 983B0100 229FD4E5  *....;.........MV*
000010  F4F1D4D8 F0F70002 F6F0F000 00004904  *41MQ07..600.....*
000020  00700001 00000034 00D00001 00000104  *.........}......*
000030  02400020 F70000D0 E6E3C9C4 00000002  *. ..7..}WTID....*
.
.
.
000100  7F4A4BB8 F7020240 E6D8E2E3 00000001  *"...7.. WQST....*
```

Figure 25. Example SMF type 116, subtype 2 record

The first self-defining section starts at X'24' and is **bold** in the example; X'00000034' is the offset to the WTID data record, X'00D0' is the length of the WTID record, and X'0001' is the number of WTID records.

The second self-defining section starts at X'2C' and is in *italic* ; X'00000104' is the offset to the first WQST data record, X'0240' is the length of the WQST record, and X'0020' is the number of WQST records.

Figure 26 on page 336 shows an example of an SMF type 116, subtype 1 record where no queues have been opened and there are consequently no self-defining sections for WQST records..

```
000000           5E740039 4E9B0104 344FD4E5  *    .........|MV*
000010  F4F1D4D8 F0F70001 F6F0F000 000003DC  *41MQ07..600.....*
000020  00800001 00000034 00D00001 00000104  *................*
000030  02D80001 F70000D0 E6E3C9C4 00000002  *.Q..7...WTID....*
000040  C1F8C5C1 C4C5D740 C1F8C5C1 C4C54040  *A8EADEP A8EADE  *
000050  40404040 40404040 00000000 00000000  *        ........*
000060  40404040 40404040 4040               *          *
```

Figure 26. Example SMF type 116, subtype 1 record with no WQST data records

The first self-defining section starts at X'24' and is **bold** in the example; X'00000034' is the offset to the WTID data record, X'00D0' is the length of the WTID record, and X'0001' is the number of WTID records.

The second self-defining section starts at X'2C' and is in *italic* ; X'0000010C' is the offset to the WTAS data record, X'02D8' is the length of the WTAS record, and X'0001' is the number of WTAS records.

There is no self-defining section describing a WQST data record, equivalent to the third self-defining section in Figure 24 on page 335.

z/OS *Layout of channel initiator SMF type 116 records*

The layout of channel accounting data (SMF type 116, subtype 10) records is described in this topic.

Self-defining section

The self-defining section for the channel accounting data follows the standard SMF header. It is structured in the standard triplet format. The format of the triplets is described in structure qws5 in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQWS5)`.

Table 48 on page 336 shows the format of the self-defining section.

Table 48. Structure of the channel accounting self-defining section

Offset: Dec	Offset : Hex	Type	Length	Name	Description
0	0	Integer	4	QWS50PSO	Offset from the start of the SMF record to the first instrumentation standard header (QWHS)
4	4	Integer	2	QWS50PSL	Length of the QWHS
6	6	Integer	2	QWS50PSN	Number of instances of QWHS
8	8	Integer	4	QWS50R1O	Offset from the start of the SMF record to the first channel accounting data record (QCST)
12	C	Integer	2	QWS50R1L	Length of the QCST
14	E	Integer	2	QWS50R1N	Number of instances of QCST

Instrumentation standard header (QWHS)

The format of the QWHS is described in structure qwhs in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`, and in assembler macro `thlqual.SCSQMACS(CSQDQWHS)`. It contains the following key fields that are relevant to channel initiator SMF 116 records:

Table 49. Key fields in the QWHS

Name	Length	Description
QWHSNDA	1 byte	Number of self-defining sections
QWHSSSID	4 bytes	Subsystem name
QWHSSMFC	1 bit	Indicates whether there are multiple SMF records containing information for this interval. If this bit is on, information for this interval is continued in further SMF records. If this bit is off, this is the last or only record.
QWHSTIME	8 bytes	Local time of the start of the interval in STCK format
QWHSDURN	8 bytes	Duration from the start of the interval to the end of the interval in STCK format
QWHSSTCK	8 bytes	End of the interval in UTC in STCK format

z/OS *Channel accounting data records*

Use this topic as a reference for channel accounting data records.

The format of the channel accounting data record is described in assembler macro `thlqual.SCSQMACS(CSQDQCST)`. The format is also described in the C programming language header file `thlqual.SCSQC370(CSQDSMFC)`. Note that the field names in C are all in lowercase, for example, *qcst*.

The channel accounting data gives you information about the status and statistics of each channel instance, including:

- Average network time (*qcstntav*)
- Average time on exit (*qcstetav*)
- Channel batch data limit (*qcstcbdl*)
- Channel batch interval (*qcstcbit*)
- Channel batch size (*qcstcbsz*)
- Channel dispatcher number (*qcstdspn*)
- Channel disposition (*qcstchdp*)
- Channel name (*qcstchnm*)
- Channel state (*qcstchst*)
- Channel started time (*qcststrt*)
- Channel status collected time (*qcstcltm*)
- Channel stopped time (*qcstludt*)
- Channel type (*qcstchty*)
- Common name (CN) from SSLCERTI (*qcstslcn*)
- Compression rate (*qcstcpra*)
- Connection name (*qcstcnnm*)
- Current shared conversations (*qcstcscv*)

- DNS resolution time (*qcstdnrt*)
- Effective value of STATCHL parameter (*qcststcl*)
- Last message time (*qcstlmst*)
- Maximum network time (qcstntmx)
- Maximum time on exit (*qcstetmx*)
- Minimum network time (*qcstntmn*)
- Minimum time on exit (*qcstetmn*)
- Name of the remote queue manager or application (*qcstrqmn*)
- Number of batches (*qcstbatc*)
- Number of bytes for message data (*qcstnbyt*)
- Number of bytes for persistent message data (*qcstnpby*)
- Number of bytes received for both message data and control information (*qcstbyrc*)
- Number of bytes sent for both message data and control information (*qcstbyst*)
- Number of full batches (*qcstfuba*)
- Number of messages, or number of MQI calls (*qcstnmsg*)
- Number of persistent messages (*qcstnpmg*)
- Number of put retries (*qcstptrc*)
- Number of transmission queue becoming empty (*qcstqetc*)
- Number of transmission buffers received (**qcstbfrc**)
- Number of transmission buffers sent (*qcstbfst*)
- Serial number from SSLPEER (*qcstslsn*)
- SSL CipherSpec (zero means TLS not used) (*qcstslcs*)
- The date and time of maximum network time (*qcstntdt*)
- The date and time of maximum time on exit (*qcstetdt*)

Note, that for the channel accounting field *qcstetmn* (Minimum time on exit) and *qcstntmn* (Minimum network time) these two fields will be initialized to the hexadecimal value of 8FFFFFFF when unused.

You can use this information to see the throughput of a channel, if the actual batches are approaching the limit, the latency of the network, information about the remote end, performance of user exit, and so on.

Here is an example of the channel accounting data which has been formatted with IBM MQ SupportPac MP1B.

The fields available are based on the display channel status command (DIS CHS) and channel statistics by IBM MQ on platforms except z/OS, with some additional fields.

```
The data and time of the start and end of the record in local time, and the duration
SMF interval start        2014/03/26,02:30:00
SMF interval end          2014/03/26,02:45:00
SMF interval duration     899.997759 seconds

Information about the channel

Connection name        9.20.4.159
Channel disp           PRIVATE
Channel type           RECEIVER
Channel status         CLOSING
Channel STATCHL        HIGH


Start date & time            2014/03/26,02:44:58
Channel status collect time  2014/03/26,02:45:00
Last status changed          1900/01/01,00:00:00
Last msg time                2014/03/26,02:44:59
```

```
Batch size                      50
Messages/batch                  3.3
Number of messages              1,102
Number of persistent messages   1,102
Number of batches               335
Number of full batches          0
Number of partial batches       335
Buffers sent                    337
Buffers received                1,272
Message data                    5,038,344   4 MB
Persistent message data         5,038,344   4 MB
Non persistent message data     0   0 B
Total bytes sent                9,852   9 KB
Total bytes received            5,043,520   4 MB
Bytes received/Batch            15,055  14 KB
Bytes sent/Batch                29  29 B
Batches/Second                  1
Bytes received/message          4,576   4 KB
Bytes sent/message              8   8 B
Bytes received/second           28,019  27 KB/sec
Bytes sent/second               54  54 B/sec
Compression rate                0

The name of the queue manager at the remote end of the connection
Remote qmgr/app            MQPH
Put retry count            0
```

调整 IBM MQ 网络

使用此部分中的调整提示可帮助提高队列管理器网络的性能。

调整客户机和服务器连接通道

SHARECNV 的缺省设置为 10，这允许每个通道实例最多 10 个客户机对话。 但是，使用不同数目的共享对话可以提高性能。 如果您不需要共享对话，或者正在使用分布式服务器，请将 **SHARECNV** 设置为 1。 如果现有客户机应用程序在将 **SHARECNV** 设置为 1 或更高版本时未正确运行，请将 **SHARECNV** 设置为 0。

关于此任务

对于某些配置，使用共享对话会带来显着好处。 但是，对于分布式服务器，在使用缺省配置 10 个共享对话的通道上处理消息的速度平均比不使用共享对话的通道慢 15%。 在共享对话的 MQI 通道实例上，套接字上的所有对话都由同一线程接收。 如果共享套接字的对话都很忙，那么对话线程会相互争用以使用接收线程。 争用会导致延迟，在这种情况下，使用较少的共享对话会更好。

使用 **SHARECNV** 参数可指定要在特定 TCP/IP 客户机通道实例上共享的最大对话数。 有关所有可能值的详细信息，请参阅 支持的 IBM MQ 客户机: 客户机连接和服务器连接通道的缺省行为。

如果将 **SHARECNV** 设置为 1 或更高版本，那么将启用以下性能增强功能:

- 双向脉动信号
- 管理员停止-停顿
- 预读
- 异步-由客户机应用程序使用

如果您不需要共享对话，那么这两个设置将提供最佳性能:

- SHARECNV(1).
- SHARECNV(0).

注意:

- 如果客户机连接 **SHARECNV** 值与服务器连接 **SHARECNV** 值不匹配，那么将使用最低值。
- 当针对非重入库链接或编译应用程序时，即使在 CLNTCONN 和 SVRCONN 中设置了较大的值，也会协商 CURSHCNV (0) 值。

要优化给定通道实例的性能，请完成以下任何步骤。

过程

- 监视使用缺省 SHARECNV 值 10 的通道。

  SHARECNV(10) 的缺省设置在许多场景中运行良好，但可能不是给定通道实例的最佳设置。 例如，对于分布式服务器，在使用此设置的通道上处理消息的速度平均比不使用共享对话的通道慢 15%。

  要确保缺省设置适用于给定的通道实例，请使用此设置监视通道的执行情况。

- 将 SHARECNV 值设置为 2 或更多。

  可以将 SHARECNV(2) 设置为 SHARECNV(999999999)。 要确保您选择的设置适合于给定的通道实例，请使用新设置来监视通道的执行情况。

- 设置 SHARECNV 值 1。

  如果不需要共享对话，请尽可能使用此设置。 它消除了使用接收线程的争用，并且您的客户机应用程序可以利用 "关于此任务" 部分中描述的性能增强功能。

  通过此设置，分布式服务器性能显着提高。 性能改进适用于发出非预读同步获取等待调用的客户机应用程序; 例如 C 客户机 MQGET 等待调用。 连接这些客户机应用程序时，分布式服务器使用的线程更少，内存更少，吞吐量也会增加。

  如果服务器连接了通过套接字共享对话的客户机，并且您将共享对话设置从 SHARECNV(10) 减少到 SHARECNV(1)，那么这将产生以下影响:

  – 增加了服务器上的套接字使用率。

  – 增加了服务器上的通道实例。

  在这种情况下，您还可以选择增加 **MaxChannels** 和 **MaxActiveChannels** 的设置。

  注: 您还可以设置 MQCONNX 选项 MQCNO_NO_CONV_SHARING，并将应用程序连接到 **SHARECNV** 设置为大于 1 的值的通道。 结果与将应用程序连接到 **SHARECNV** 设置为 1 的通道相同。

- 设置 SHARECNV 值 0。

  通道实例的行为与 IBM WebSphere MQ 6.0 服务器或客户机连接通道的行为完全相同。 您不会获得共享对话，也不会获得将 **SHARECNV** 设置为 1 或更高版本时可用的性能增强功能。 仅当现有客户机应用程序在将 **SHARECNV** 设置为 1 或更高版本时未正确运行时，才使用值 0 。

相关概念

受支持的 IBM MQ 客户机: 客户机连接和服务器连接通道的缺省行为

调整分布式发布/预订网络

使用此部分中的调整提示可帮助提高 IBM MQ 分布式发布/预订集群和层次结构的性能。

相关概念

第 280 页的『监视集群』

在集群中，您可以监视应用程序消息，控制消息和日志。 在队列的两个或多个实例之间进行集群负载均衡时，存在特殊监视注意事项。

直接路由的发布/预订集群性能

在直接路由的发布/预订集群中，会将诸如集群主题和代理预订之类的信息推送到集群的所有成员，而不考虑是否所有集群队列管理器都在主动参与发布/预订消息传递。 此过程可能会在系统上产生显着的额外负载。 要降低集群管理对性能的影响，您可以在非高峰时间执行更新，定义参与发布/预订的队列管理器的小得多的子集，并使其成为 "重叠" 集群，或者切换到使用主题主机路由。

发布/预订集群中的队列管理器上有两个工作负载源:

- 直接处理应用程序的消息。
- 处理管理集群所需的消息和通道。

在典型的点到点集群中，集群系统工作负载在很大程度上仅限于集群成员根据需要明确请求的信息。 因此，在非常大的点到点集群 (例如包含数千个队列管理器的集群) 之外的任何其他集群中，您可以在很大程度上降低管理集群的性能效果。 但是，在直接路由的发布/预订集群中，会将诸如集群主题，队列管理器成员资格

和代理预订之类的信息推送到集群的所有成员，而不考虑是否所有集群队列管理器都在主动参与发布/预订消息传递。 这可能会在系统上产生显着的额外负载。 因此，您需要考虑集群管理在其计时和大小方面对队列管理器性能的影响。

直接路由集群的性能特征

在核心管理任务方面，将点到点集群与直接路由的发布/预订集群进行比较。

首先，点对点集群:

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

总之，点到点集群中队列管理器上的负载与它为应用程序处理的消息流量相关，与集群大小没有直接关系。

第二，直接路由的发布/预订集群:

1. 定义新集群主题时，会将信息推送到完整存储库队列管理器，并从该存储库直接推送到集群的所有成员，从而导致通道从完整存储库启动到集群的每个成员 (如果尚未启动)。 如果这是第一个直接集群主题，那么将向每个队列管理器成员发送有关集群中所有其他队列管理器成员的信息。
2. 在新主题字符串上创建对集群主题的预订时，会立即将该信息从该队列管理器直接推送到集群的所有其他成员，从而导致通道从该队列管理器启动到该集群的每个成员 (如果尚未启动)。
3. 当新队列管理器加入现有集群时，有关所有集群主题 (以及所有队列管理器成员 (如果定义了直接集群主题)) 的信息将从完整存储库队列管理器推送到新队列管理器。 然后，新队列管理器将集群中集群主题的所有预订的知识与集群的所有成员同步。

总之，直接路由的发布/预订集群中的任何队列管理器上的集群管理负载都会随着集群中不同主题字符串上的队列管理器，集群主题和预订更改的数量而增加，而不考虑这些集群主题在每个队列管理器上的本地使用情况。

在大型集群中，或者在预订更改率较高的集群中，此级别的集群管理可能是所有队列管理器中的显着开销。

降低直接路由发布/预订对性能的影响

要降低集群管理对直接路由的发布/预订集群性能的影响，请考虑以下选项:

- 在一天中的非高峰时间执行集群，主题和预订更新。
- 定义发布/预订中涉及的队列管理器的小得多的子集，并使其成为 "重叠" 集群。 然后，此集群是定义了集群主题的集群。 虽然某些队列管理器现在位于两个集群中，但发布/预订的总体效果会降低:
  - 发布/预订集群的大小较小。
  - 不在发布/预订集群中的队列管理器受集群管理流量的影响要小得多。

如果先前选项无法充分解决性能问题，请考虑改为使用 主题主机路由 发布/预订集群。 有关发布/预订集群中的直接路由和主题主机路由的详细比较，请参阅 设计发布/预订集群。

相关概念

主题主机路由的发布/预订集群性能
主题主机路由发布/预订集群使您能够精确控制哪些队列管理器托管每个主题。 这些主题主机将成为主题树的该分支的 路由 队列管理器。 此外，没有预订或发布程序的队列管理器无需与主题主机连接。 此配置可以显着减少集群中队列管理器之间的连接数以及队列管理器之间传递的信息量。

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

发布/预订网络中的预订性能

IBM MQ 中的分布式发布/预订通过传播有关在队列管理器网络中创建了不同主题字符串的预订的位置的知识来工作。 这使发布消息的队列管理器能够识别哪些其他队列管理器需要已发布消息的副本以匹配其预订。

主题主机路由的发布/预订集群性能

主题主机路由发布/预订集群使您能够精确控制哪些队列管理器托管每个主题。 这些主题主机将成为主题树的该分支的 路由 队列管理器。 此外，没有预订或发布程序的队列管理器无需与主题主机连接。 此配置可以显着减少集群中队列管理器之间的连接数以及队列管理器之间传递的信息量。

主题主机路由的发布/预订集群的行为如下所示:

- 在集群中的各个 主题主机 队列管理器上手动定义主题。
- 在集群队列管理器上进行预订时，仅在主题主机上创建代理预订。
- 当应用程序将信息发布到主题时，接收队列管理器会将发布转发到托管该主题的队列管理器。 然后，主题主机将该发布发送到集群中对该主题具有有效预订的所有队列管理器。

有关主题主机路由的更详细介绍，请参阅 集群中的主题主机路由。

对于许多配置，主题主机路由是比 直接路由 更合适的拓扑，因为它具有以下优点:

- 提升了大型集群的可伸缩性。 只有主题主机队列管理器需要能够连接到集群中的所有其他队列管理器。 因此，队列管理器之间存在较少的通道，队列管理器间的发布/预订管理流量比用于直接路由的少。 在队列管理器上进行预订更改时，只需要通知主题主机队列管理器。
- 对物理配置进行更多控制。 对于直接路由，所有队列管理器采用所有角色，因此全都需要同等能力。 对于主题主机路由，您可显式选择主题主机队列管理器。 因此，您可以确保这些队列管理器在合适的设备上运行，并且可以将功能差一些的系统用于其他队列管理器。

但是，主题主机路由还将对您的系统施加一些约束:

- 系统配置和维护需要比直接路由更多的规划。 您需要决定在主题树中要对哪些点建立集群，以及集群中主题定义的位置。
- 与直接路由主题一样，在定义新的主题主机路由主题时，该信息会推送到完整存储库队列管理器，并从中定向到集群的所有成员。 此事件将导致通道从完整存储库启动到集群的每个成员（如果尚未启动）。
- 发布始终会从非主机队列管理器发送至主机队列管理器，即使集群中没有预订也是如此。 因此，通常期望存在预订时，或全局连接和知晓的开销大于额外发布流量的风险时，应使用路由的主题。
- 在非主机队列管理器上发布的消息不会直接进入托管预订的队列管理器，而是会始终通过主题主机队列管理器进行路由。 此方法可能会增加集群的总开销，并且可能会增加消息等待时间而降低性能。

  注: 对于某些配置，您可以有效除去此约束，如 使用集中发布程序或订户的主题主机路由中所述。
- 使用单一主题主机队列管理器将为发布到某个主题的所有消息引入单一故障点。 您可以通过定义多个主题主机来移除此单一故障点。 然而，拥有多个主机将影响预订收到已发布消息的顺序。
- 主题主机队列管理器将产生额外消息负载，因为它们需要处理来自多个队列管理器的发布流量。 可以减轻此负载：将多个主题主机用于单个主题（在此情况下，不会维护消息顺序），或使用不同的队列管理器来托管主题树的不同分支的路由主题。

具有集中发布程序或订户的主题主机路由

要除去始终通过主题主机队列管理器将发布路由到预订时产生的额外 "中继段"，请在托管该主题的同一队列管理器上配置发布程序或预订。 在以下两种情况下，此方法会带来最大的性能优势:

- 具有许多发布者和少量预订的主题。 在这种情况下，请在主题主机队列管理器上托管预订。
- 具有少量发布者和许多预订的主题。 在这种情况下，请在主题主机队列管理器上托管发布程序。

下图显示了同时托管预订的主题主机队列管理器。 此方法将除去发布程序与订户之间的额外 "中继段"，并减少在集群的所有成员之间不必要的预订知识共享:

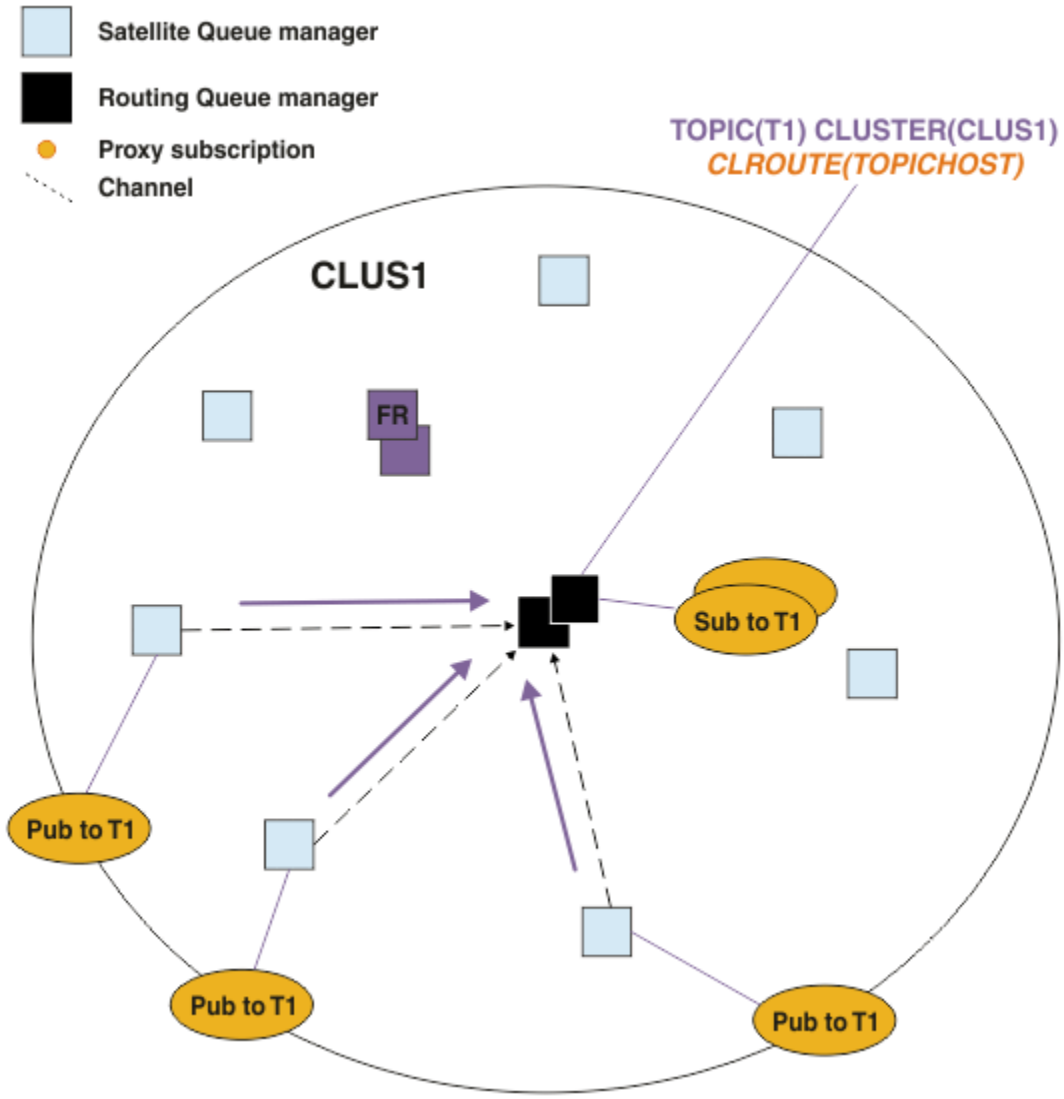


图 *27:* 在主题主机队列管理器上托管预订

下图显示了同时托管发布程序的主题主机队列管理器。 此方法将除去发布程序与订户之间的额外 "中继段"，并减少在集群的所有成员之间不必要的预订知识共享:

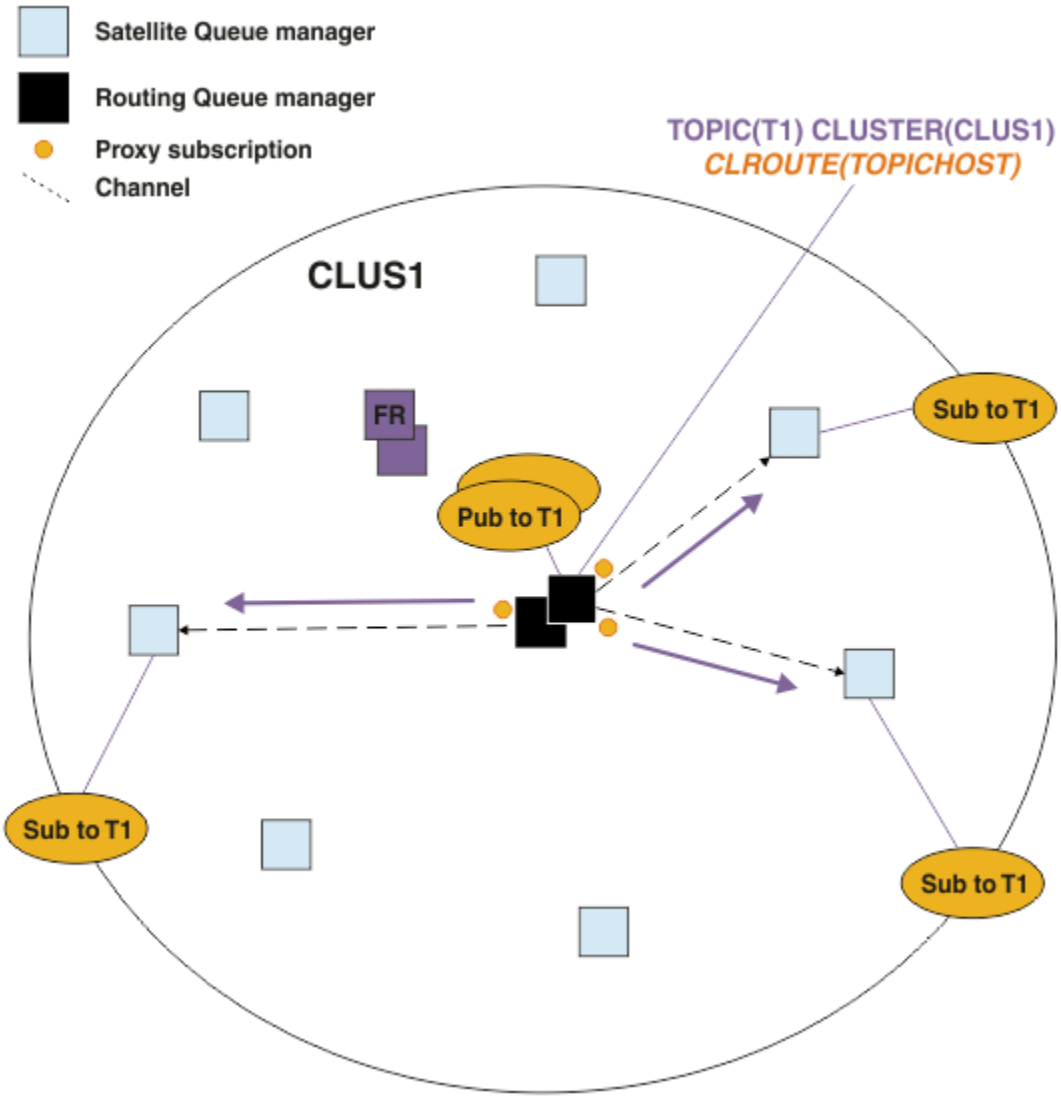


图 *28:* 在主题主机队列管理器上托管发布

相关概念

直接路由的发布/预订集群性能

在直接路由的发布/预订集群中，会将诸如集群主题和代理预订之类的信息推送到集群的所有成员，而不考虑是否所有集群队列管理器都在主动参与发布/预订消息传递。 此过程可能会在系统上产生显着的额外负载。 要降低集群管理对性能的影响，您可以在非高峰时间执行更新，定义参与发布/预订的队列管理器的小得多的子集，并使其成为 "重叠" 集群，或者切换到使用主题主机路由。

在发布/预订网络中平衡生产者和消费者

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

发布/预订网络中的预订性能

IBM MQ 中的分布式发布/预订通过传播有关在队列管理器网络中创建了不同主题字符串的预订的位置的知识来工作。 这使发布消息的队列管理器能够识别哪些其他队列管理器需要已发布消息的副本以匹配其预订。

在发布/预订网络中平衡生产者和消费者

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

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

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

将与远程队列管理器上的预订匹配的每个发布 (基于代理预订) 放入传输队列。 如果多个远程队列管理器具有该发布的代理预订，那么会将消息的多个副本放入传输队列，每个副本都以不同的发送方通道为目标。

在发布/预订集群中，这些发布以托管预订的远程队列管理器上的 SYSTEM.INTER.QMGR.PUBS 队列为目标。 在层次结构中，每个发布都以 SYSTEM.BROKER.DEFAULT.STREAM 队列或远程队列管理器上的 SYSTEM.QPUBSUB.QUEUE.NAMELIST 中列出的任何其他流队列为目标。 每个队列管理器都会处理到达该队列的消息，并将这些消息传递到该队列管理器上的正确预订。

因此，请在以下可能出现瓶颈的位置监视负载:

- 监视各个预订队列上的负载。
  - 此瓶颈意味着预订应用程序不会像发布发布一样快速使用发布内容。
- 监视 SYSTEM.INTER.QMGR.PUBS 队列或流队列上的负载。
  - 此瓶颈意味着队列管理器接收来自一个或多个远程队列管理器的发布的速度比它可以将这些发布分发到本地预订的速度更快。
  - 在集群中使用主题主机路由时，在主题主机队列管理器上看到时，请考虑创建其他队列管理器主题主机，以允许在它们之间均衡发布工作负载。 但是，这将影响跨发布的消息排序。 请参阅 使用单个主题的多个主题主机进行主题主机路由。
- 监视发布队列管理器与预订队列管理器之间的通道处的负载，这些通道由发布队列管理器上的传输队列提供。
  - 此瓶颈意味着一个或多个通道未在运行，或者消息发布到本地队列管理器的速度超过了这些通道可以将它们传递到远程队列管理器的速度。
  - 使用发布/预订集群时，请考虑在目标队列管理器上定义其他集群接收方通道。 这允许在发布工作负载之间进行平衡。 但是，这会影响跨发布的消息排序。 还要考虑移至多集群传输队列配置，因为这可以在某些情况下提高性能。
- 如果发布应用程序正在使用排队的发布/预订接口，请监视 (a) SYSTEM.BROKER.DEFAULT.STREAM 队列以及 SYSTEM.QPUBSUB.QUEUE.NAMELIST 中列出的任何其他流队列处的负载; 和 (b) SYSTEM.BROKER.DEFAULT.SUBPOINT 队列以及 SYSTEM.QPUBSUB.SUBPOINT.NAMELIST 中列出的任何其他子点队列。
  - 此瓶颈意味着本地发布应用程序放置消息的速度比本地队列管理器处理消息的速度要快。

相关概念

直接路由的发布/预订集群性能
在直接路由的发布/预订集群中，会将诸如集群主题和代理预订之类的信息推送到集群的所有成员，而不考虑是否所有集群队列管理器都在主动参与发布/预订消息传递。 此过程可能会在系统上产生显着的额外负载。 要降低集群管理对性能的影响，您可以在非高峰时间执行更新，定义参与发布/预订的队列管理器的小得多的子集，并使其成为 "重叠" 集群，或者切换到使用主题主机路由。

主题主机路由的发布/预订集群性能
主题主机路由发布/预订集群使您能够精确控制哪些队列管理器托管每个主题。 这些主题主机将成为主题树的该分支的 路由 队列管理器。 此外，没有预订或发布程序的队列管理器无需与主题主机连接。 此配置可以显着减少集群中队列管理器之间的连接数以及队列管理器之间传递的信息量。

发布/预订网络中的预订性能
IBM MQ 中的分布式发布/预订通过传播有关在队列管理器网络中创建了不同主题字符串的预订的位置的知识来工作。 这使发布消息的队列管理器能够识别哪些其他队列管理器需要已发布消息的副本以匹配其预订。

第 280 页的『监视集群』
在集群中，您可以监视应用程序消息，控制消息和日志。 在队列的两个或多个实例之间进行集群负载均衡时，存在特殊监视注意事项。

发布/预订网络中的预订性能

IBM MQ 中的分布式发布/预订通过传播有关在队列管理器网络中创建了不同主题字符串的预订的位置的知识来工作。 这使发布消息的队列管理器能够识别哪些其他队列管理器需要已发布消息的副本以匹配其预订。

此方法可最大程度地将已发布的消息发送到不存在匹配预订的队列管理器。 但是，通过频繁的订阅创建和删除，当被订阅的主题字符串数量高且不断变化时，订阅知识的传播会成为一个显着的开销。

您可以通过调整发布和预订在发布/预订网络中的流动方式来影响性能。 如果网络流量很少发布，并且快速创建，删除或更改预订，那么可以停止将预订信息流至所有队列管理器，而改为将所有发布转发至网络中的所有队列管理器。 您还可以在已连接的队列管理器之间限制给定主题的代理预订和发布的流，限制包含通配符的代理预订的流，并减少主题字符串的数量和瞬态性质。

个别预订传播和 到处发布

随时随地发布 是单个预订传播的替代方法。 通过个别传播，只有在队列管理器上具有匹配预订的发布才会转发到该队列管理器。 通过 随时随地发布，所有发布都将转发到网络中的所有队列管理器。 然后，接收队列管理器会交付与本地预订匹配的发布。

个别预订传播

此机制会产生最少的队列管理器间发布流量，因为只会发送与队列管理器上的预订匹配的那些发布。

但是:

- 对于预订的每个单独主题字符串，将向发布/预订拓扑中的其他队列管理器发送代理预订。 队列管理器集合取决于正在使用的路由模型，如 规划分布式发布/预订网络中所述。
  - 如果要创建或删除数千个预订 (例如，在重新启动队列管理器之后重新创建所有非持久预订) ，或者如果预订集正在快速更改，并且每个预订都具有不同的主题字符串，那么此消息传递开销非常大。
  - 代理预订所传播到的队列管理器数也会影响开销的规模。
- 使用异步消息传递将代理预订流至其他队列管理器。 这具有以下效果:
  - 创建预订与其他队列管理器创建，交付和处理代理预订之间存在延迟。
  - 在该时间间隔内在这些队列管理器上发布的消息不会传递到远程预订。

到处发布

通过此机制，系统上没有每个主题字符串代理预订开销。 这意味着快速预订创建，删除或更改不会导致网络负载和处理增加。

由于所有发布都流向所有队列管理器，因此在创建预订和将发布流向队列管理器之间也没有延迟。 因此，没有将发布传递到新创建的远程预订的窗口。

但是:

- 将所有发布发送到发布/预订拓扑中的所有队列管理器可能会导致过量的网络流量，其中发布在每个队列管理器上都没有匹配的预订。
  - 拓扑中的队列管理器数越大，开销越大。

当您期望从很大一部分队列管理器预订发布时，或者由于预订更改频率而导致代理预订开销过大时，应考虑使用 所有位置都发布 机制。 在将发布发送到所有队列管理器 (而不是具有匹配预订的队列管理器) 时迁到消息传递流量增加的情况下，应该使用单个代理预订转发。

您可以在主题树中的任何级别设置 处处发布 行为。 要启用 所有位置发布，请将高级主题对象的 **PROXYSUB** 参数设置为 FORCE 。 这将导致单个通配符代理预订与主题树中此主题对象下的所有主题匹配。 在集群主题对象上设置此属性时， **PROXYSUB(FORCE)** 属性将传播到网络中的每个队列管理器，而不仅仅是定义主题的队列管理器。

注: 在层次结构中使用时，将在每个队列管理器上单独设置 **PROXYSUB(FORCE)** ，因此拓扑机制自然会限制通道数。 但是，在集群中使用时，可能会启动许多其他通道:

- 在主题主机路由集群中，将启动从每个队列管理器到每个主题主机队列管理器的通道。
- 在直接路由的集群中，将启动从每个队列管理器到每个其他队列管理器的通道。

在直接路由集群中，启动许多通道的开销最为明显，并且可能导致性能问题。 请参阅第 340 页的『直接路由的发布/预订集群性能』。

限制已连接队列管理器之间代理预订和发布的流的其他方法

合并主题字符串
使用许多不同的瞬态主题字符串会在连接发布程序或预订的系统中的每个队列管理器上引入某种级别的管理开销。 您应该定期评估主题字符串的使用情况，以查看它们是否可以合并。 减少主题字符串的数量和瞬态性质，从而减少对它们的发布者和预订，从而减少对系统的影响。

限制发布和预订范围
对于给定主题，可以使用 发布作用域 和 预订作用域 设置将发布和预订保留在定义了这些发布和预订的队列管理器的本地位置。

对通配符主题进行的块预订
您可以通过将 **Topic** 属性 **通配符** 设置为 BLOCK 来限制包含通配符的代理预订流。 请参阅 代理预订中的通配符。

另请参阅第 344 页的『在发布/预订网络中平衡生产者和消费者』

监视集群中的代理预订流量

在考虑来自代理预订流量的系统负载时，除了监视第 344 页的『在发布/预订网络中平衡生产者和消费者』中列出的队列外，还要监视以下集群队列:

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

这些队列上的任何重要消息积压意味着预订更改速率对于系统而言太大，或者队列管理器在集群中未正常运行。 如果您怀疑特定队列管理器存在问题，请检查是否未对该队列管理器禁用发布/预订支持。 请参阅 ALTER QMGR 中的 **PSMODE** 。

相关概念
直接路由的发布/预订集群性能
在直接路由的发布/预订集群中，会将诸如集群主题和代理预订之类的信息推送到集群的所有成员，而不考虑是否所有集群队列管理器都在主动参与发布/预订消息传递。 此过程可能会在系统上产生显着的额外负载。 要降低集群管理对性能的影响，您可以在非高峰时间执行更新，定义参与发布/预订的队列管理器的小得多的子集，并使其成为 "重叠" 集群，或者切换到使用主题主机路由。

主题主机路由的发布/预订集群性能
主题主机路由发布/预订集群使您能够精确控制哪些队列管理器托管每个主题。 这些主题主机将成为主题树的该分支的 路由 队列管理器。 此外，没有预订或发布程序的队列管理器无需与主题主机连接。 此配置可以显着减少集群中队列管理器之间的连接数以及队列管理器之间传递的信息量。

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

发布/预订网络中的代理预订

减少主题树中不需要的主题数

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

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

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

队列管理器可识别不需要的主题，因为它已未使用 30 分钟。 队列管理器将从主题树中为您除去未使用的主题。 可以通过改变队列管理器属性 **TREELIFE** 来更改 30 分钟的持续时间。 您可以帮助队列管理器除去不需要的主题，方法是确保该主题对队列管理器显示为未使用。 第 348 页的『什么是未使用的主题?』 部分说明了什么是未使用的主题。

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

什么是未使用的主题?

在定义未使用的主题之前，什么是主题?

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

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

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

主题树中未使用的主题具有以下属性。

它未与主题对象关联

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

它没有保留发布

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

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

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

它没有子主题

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

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

节点没有活动发布程序

节点的活动发布程序是打开主题以进行输出的应用程序。

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

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

没有节点的活动订户

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

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

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

管理主题树中的主题数

总之，可以通过多种方法来管理主题树中的主题数。

显示 TPCOUNT

定期使用 **runmqsc** 命令 DISPLAY PUBSUB ALL 来显示 **TPCOUNT** 属性。 这是主题树中的主题节点数。 如果数字正在增长，那么可能指示需要较短的 TREELIFE，或者需要重新设计主题本身。

修改 TREELIFE

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

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

在异常情况下，重新启动队列管理器

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

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

相关概念

主题树

Windows Linux MQ Adv. MQ Adv. VUE Aspera gateway 可提高高延迟网络的性能

IBM Aspera faspio Gateway 提供了快速 TCP/IP 隧道，可以显着提高 IBM MQ 的网络吞吐量。

Aspera gateway 可用于提高队列管理器通道的性能。 如果网络具有高延迟或倾向于丢失包，那么它尤其有效，并且通常用于加快不同数据中心的队列管理器之间的连接。

但是，对于不丢失包的快速网络，在使用 Aspera gateway 时性能会下降，因此在定义 Aspera gateway 连接之前和之后检查网络性能很重要。

在任何授权平台上运行的队列管理器都可以通过 Aspera gateway 进行连接。 网关本身部署在 Red Hat®，Ubuntu Linux 或 Windows 上。

有关更多信息，请参阅 在 Linux 或 Windows 上定义 Aspera gateway 连接。

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 IBM 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: