7.5

IBM WebSphere 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 257 页的『声明』中的信息。

此版本适用于 IBM® WebSphere MQ V 7 发行版 5 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

监控和性能

IBM WebSphere MQ 中提供了许多监视技术，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。

根据队列管理器网络的大小和复杂性，您可以从监视队列管理器网络获取一系列信息。 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。

相关任务

配置

管理 WebSphere MQ

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 WebSphere 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

事件消息引用

第 7 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

事件消息格式

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 WebSphere 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 6 页的图 1 说明了检测事件的概念。

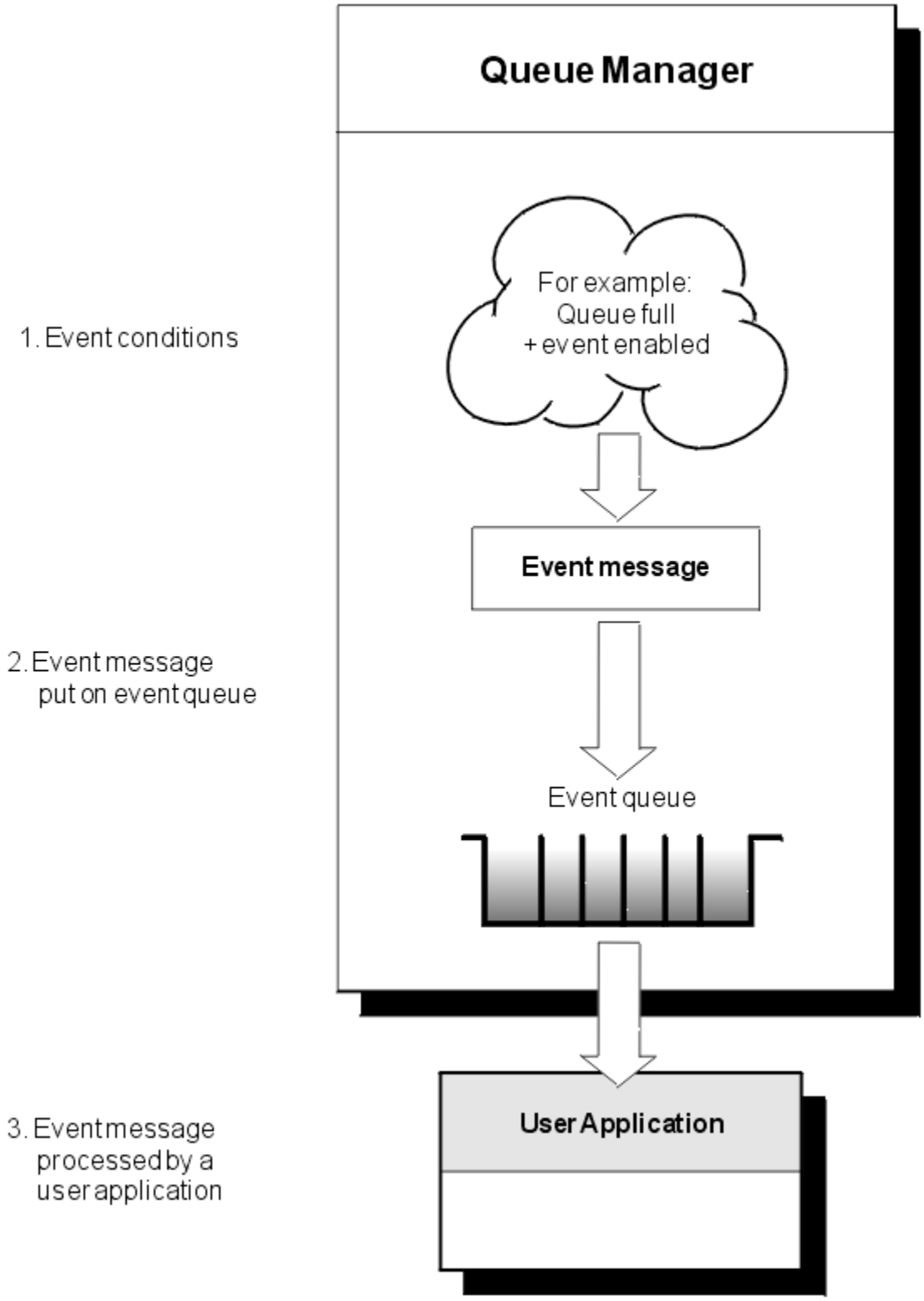


图 *1:* 了解检测事件

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。
- 对于 IBM i， Windows 和 UNIX and Linux® 系统，应用程序尝试打开一个指定未在 IBM WebSphere MQ 上授权的用户标识的队列。
- 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 17 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 42 页的『用于监视检测事件的样本程序』
使用此页面来查看用于监视检测事件的样本 C 程序

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 WebSphere 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 WebSphere MQ 应用程序监视活动。 特别是，您可以从单个节点 (针对支持 IBM WebSphere MQ 事件的那些节点) 监视系统中的所有节点，如 第 8 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。

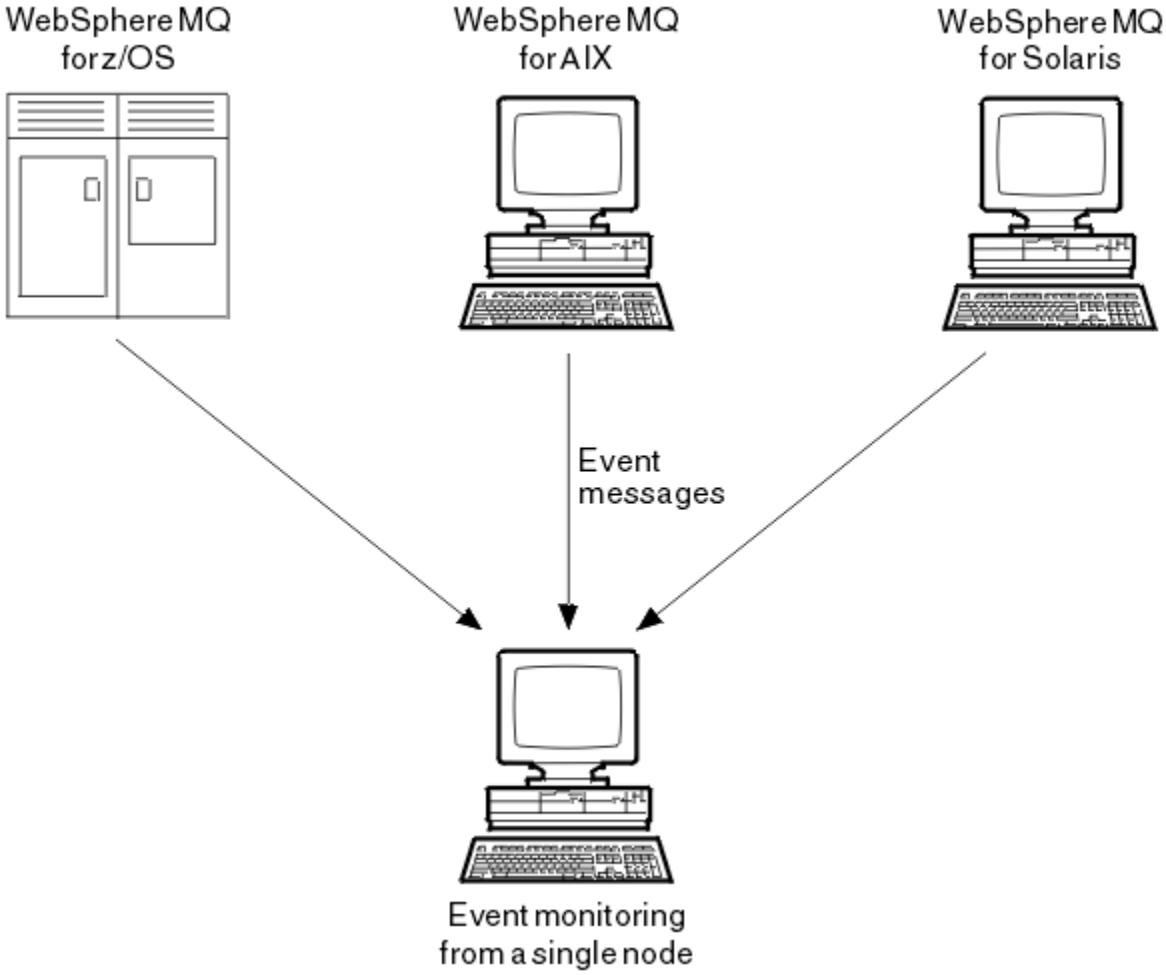


图 *2:* 在单个节点上跨不同平台监视队列管理器

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。

以下示例是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。

WebSphere 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：

权限 (仅在 Windows 和 UNIX 系统上)
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 Windows 和 UNIX 系统上有效。

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
本地事件指示应用程序 (或队列管理器) 无法访问本地队列或其他本地对象。 例如，应用程序可能会尝试访问尚未定义的对象。 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
远程事件指示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，可能未正确定义要使用的传输队列。 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 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

SSL 事件

唯一的安全套接字层 (SSL 或 TLS) 事件是 "通道 SSL 错误" 事件。 当使用 SSL 或 TLS 的通道无法建立 SSL 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。

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。

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 。

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时，将报告记录器事件。

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
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

事件类型	请参阅以下主题
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 WebSphere MQ 脚本命令 (MQSC)。
- 相应的 IBM WebSphere MQ PCF 命令。
- IBM WebSphere 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 5 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。

相关任务

自动执行管理任务

使用可编程命令格式

相关参考

第 7 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 14 页的表 1 总结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 ，并指定相应的队列管理器属性。 第 14 页的表 2 概述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。

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，别名队列或远程队列的本地定义。 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。

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 13 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 16 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 WebSphere 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。

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 WebSphere 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 84 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 88 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 89 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 WebSphere MQ 命令 (MQSC) 来设置队列属性。 在 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 7 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 17 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"。

相关概念

第 17 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 19 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 20 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 20 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

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。

队列服务时间间隔上限
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 19 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。

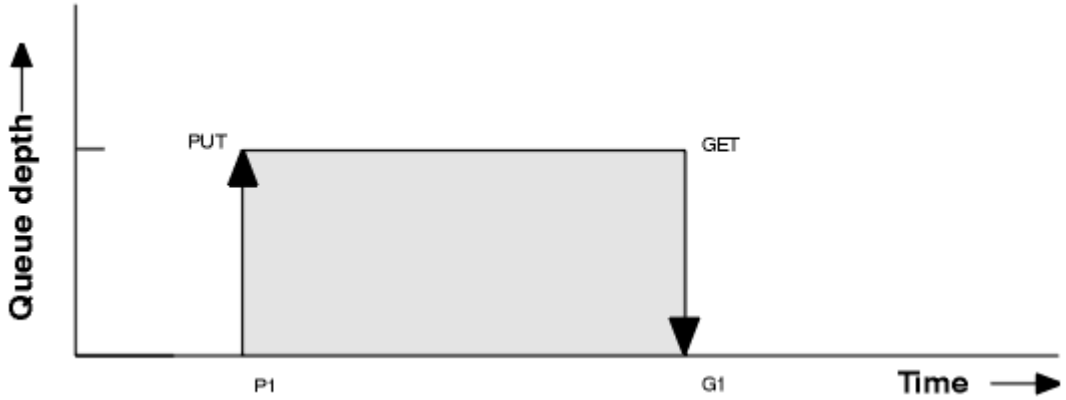


图 *3:* 了解队列服务时间间隔事件

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 20 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
队列服务时间间隔上限
QServiceInterval 事件 (MQLONG)
ServiceInterval 事件属性

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 20 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，为队列上的 "队列服务时间间隔高" 或 "正常" 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

使用这些示例来了解可以从队列服务时间间隔事件获取的信息

这三个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。

这些示例随附的数字具有相同的结构:

- 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 18 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 19 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。

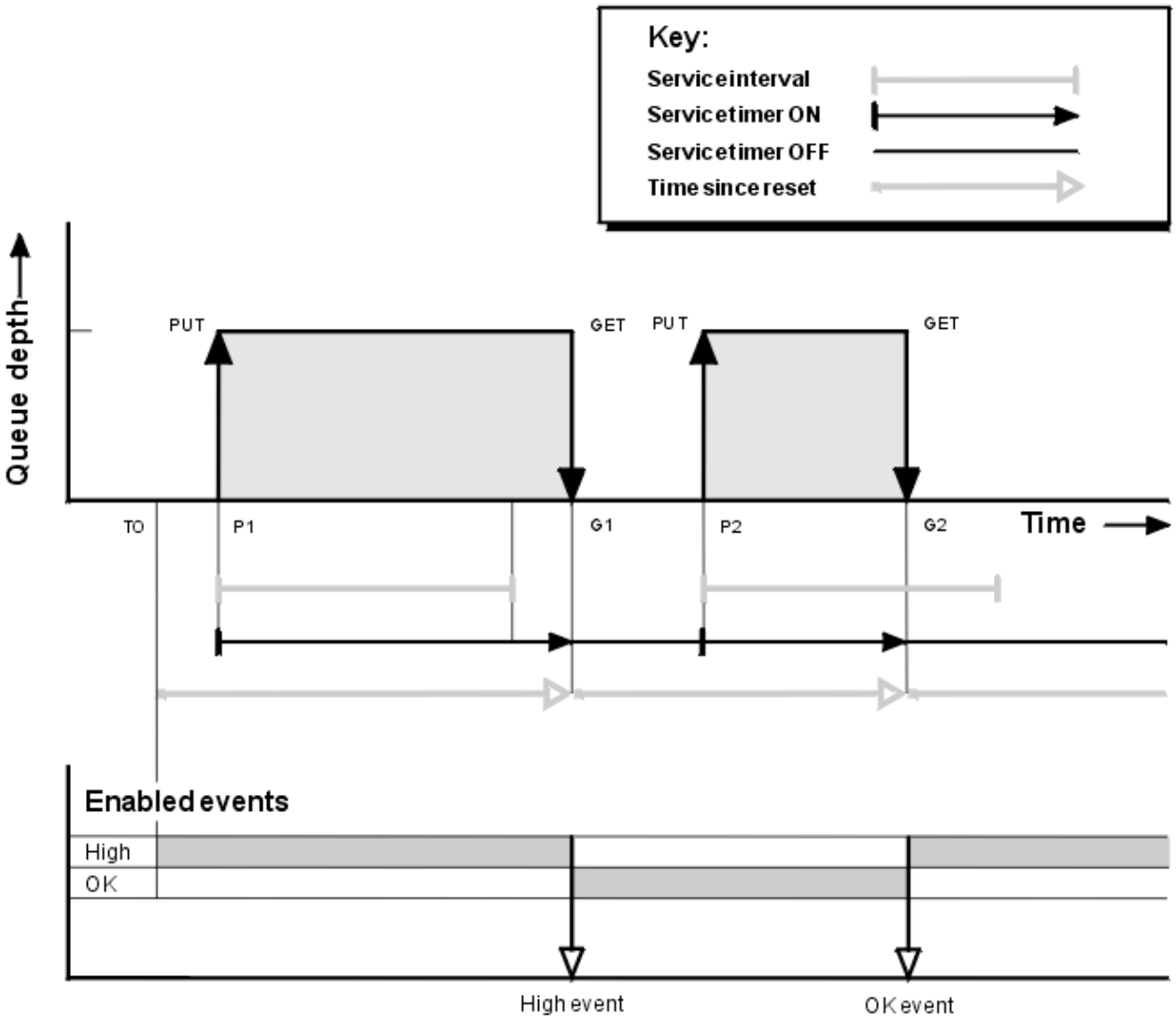


图 *4:* 队列服务时间间隔事件-示例 *1*

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 23 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)*

	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 22 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。

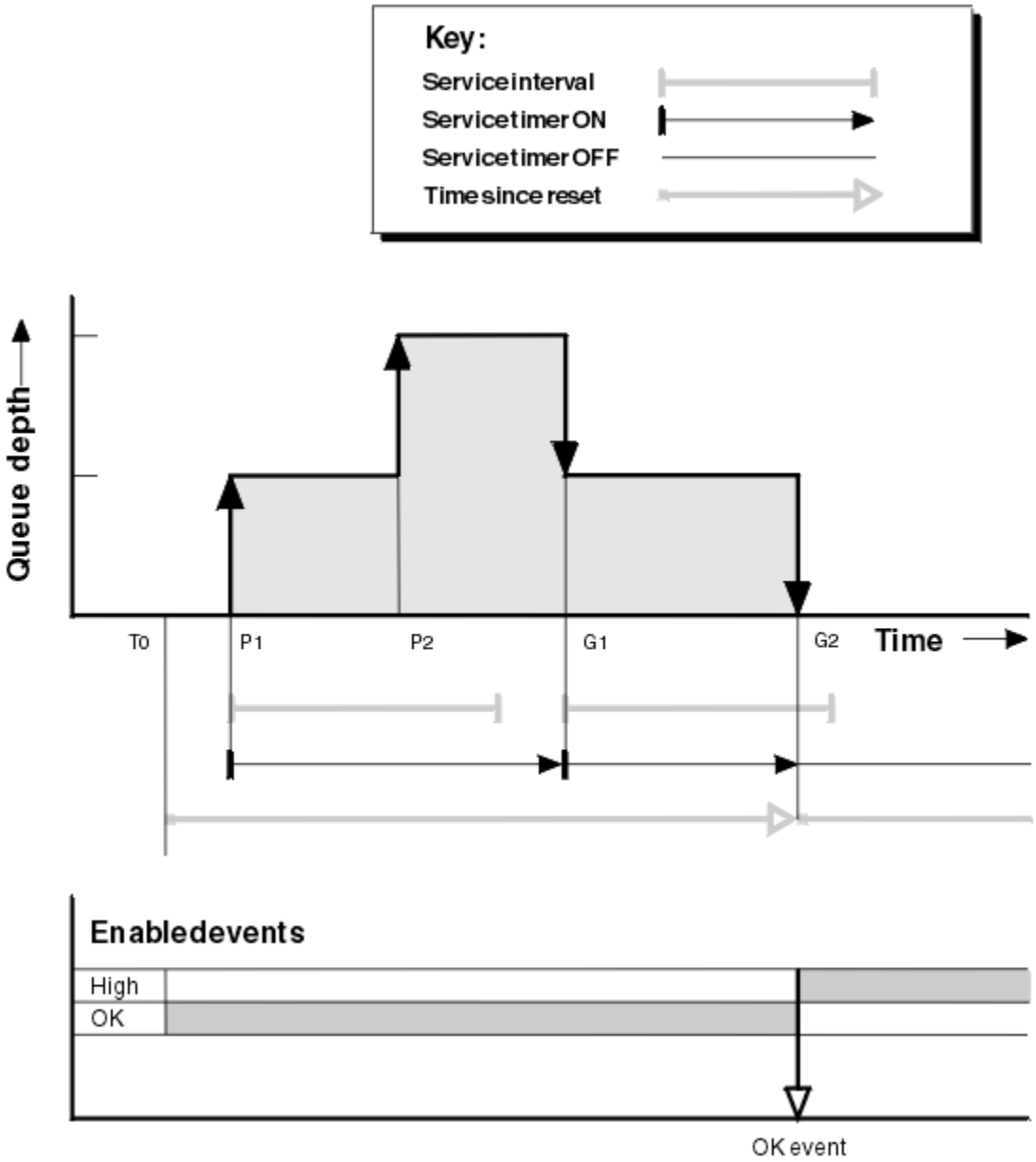


图 *5:* 队列服务时间间隔事件-示例 *2*

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 25 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)*	
	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。

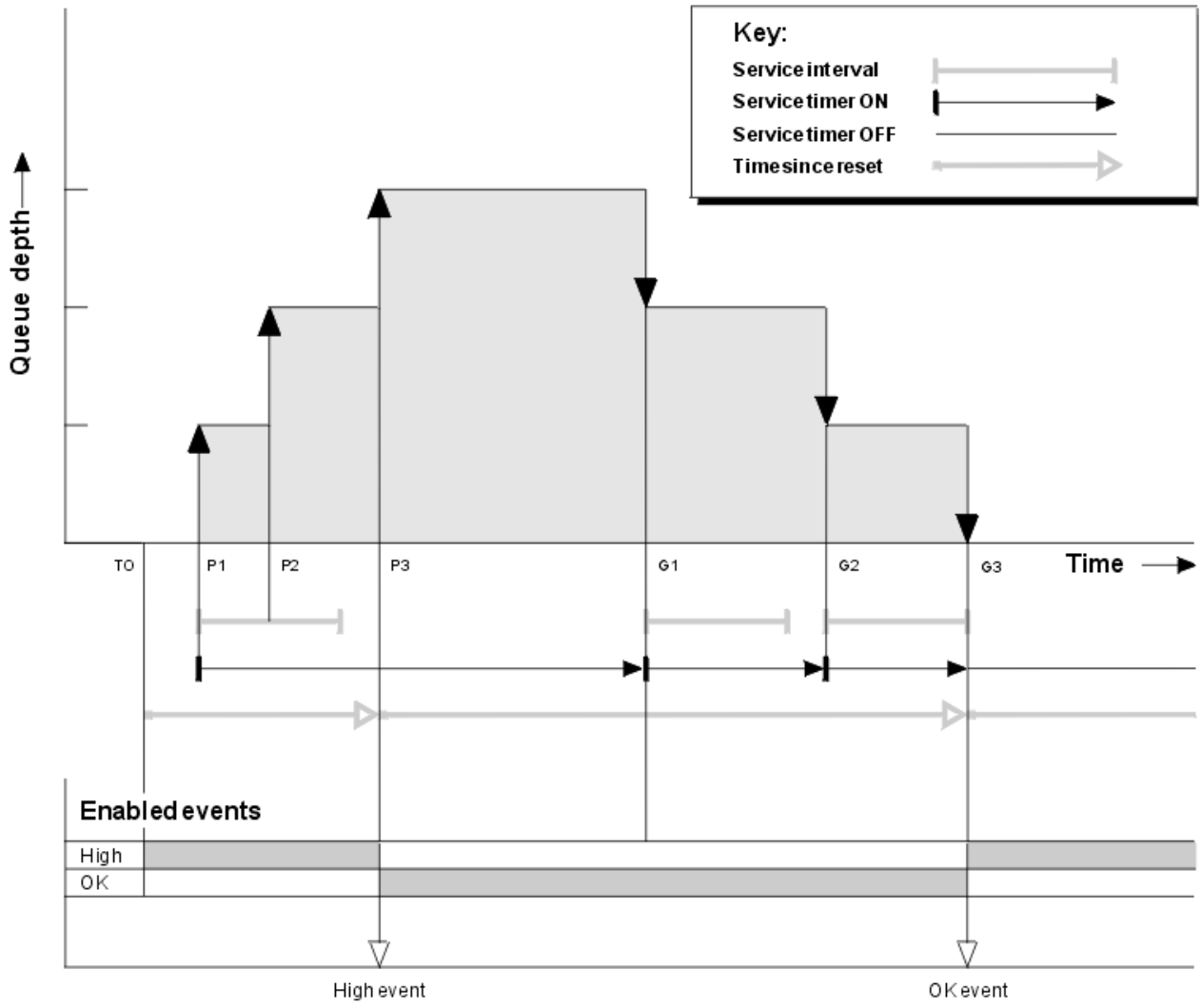


图 *6:* 队列服务时间间隔事件-示例 *3*

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，第三个获取会清空队列，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 26 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)*

	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 WebSphere 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， WebSphere 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 28 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 "队列深度高" 事件，这将导致队列深度大于或等于 "队列深度高" 限制所确定的值。
  - 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。
- 当获取操作将消息从队列中除去时，将生成 "队列深度下限" 事件，此事件导致队列深度小于或等于 "队列深度下限" 限制所确定的值。
  - "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。
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)
   - *QDepthLowEvent* (MQSC 中的 QDPLOEV)
   - *QDepthMaxEvent* (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)
   - *QDepthLowLimit* (MQSC 中的 QDEPTHLO)

   限制: QDEPTHHI 不得小于 QDEPTHLO。

   如果 QDEPTHHI 等于 QDEPTHLO ，那么每当队列深度通过任一方向的值时都会生成事件消息，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:

当通过获取操作从队列中除去到期消息时，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 "队列深度下限" 限制所确定的值。

IBM WebSphere MQ 仅在成功执行 get 操作期间生成低事件消息。 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，不会重置队列深度高事件和队列深度低事件。

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)
```

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 WebSphere 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 26 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 27 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 29 页的图 7 显示了队列深度随时间变化的情况。

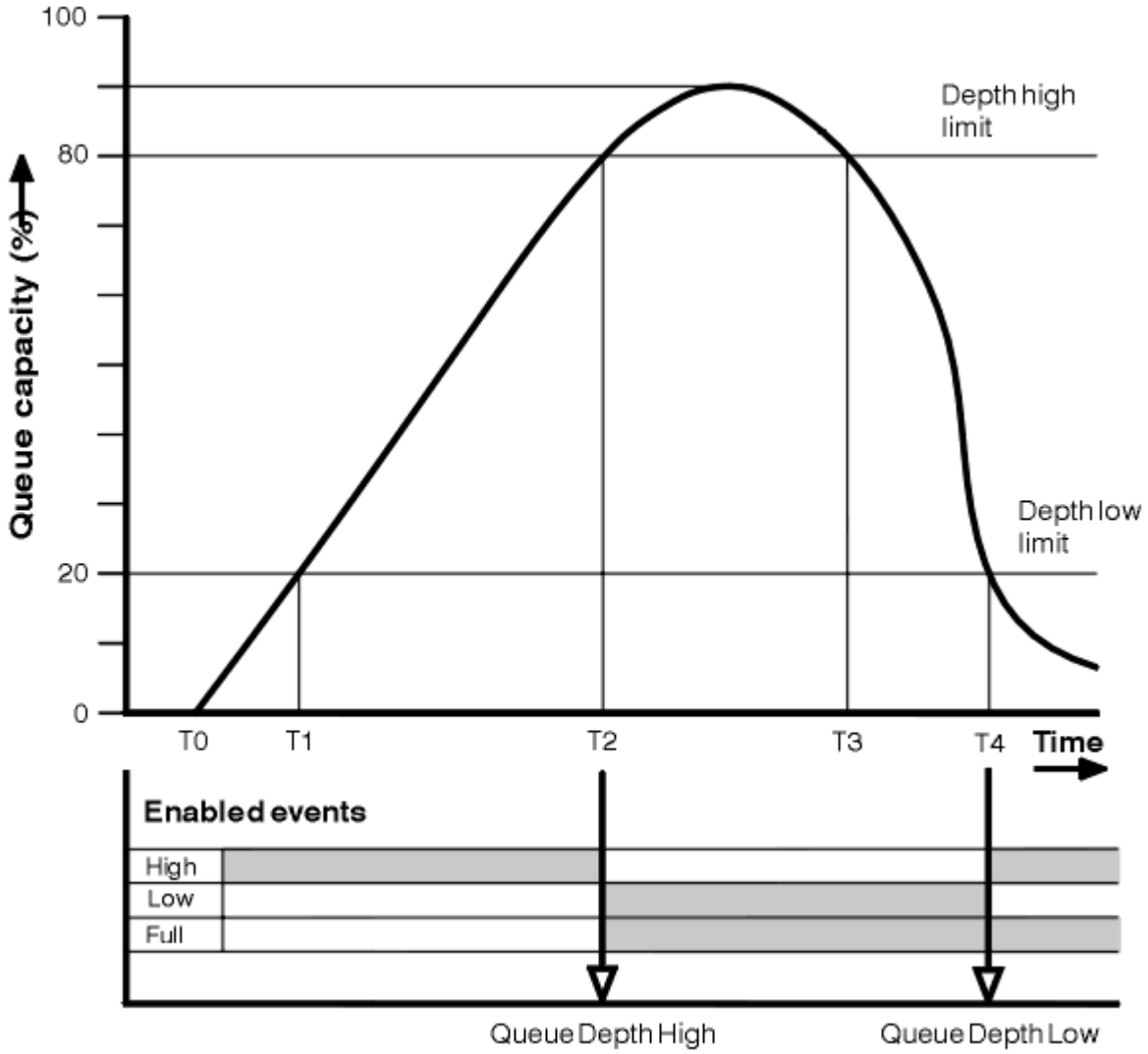


图 *7:* 队列深度事件 *(1)*

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 30 页的表 8 汇总了队列事件统计信息， 第 30 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 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 31 页的图 8 显示了队列深度随时间变化的情况。

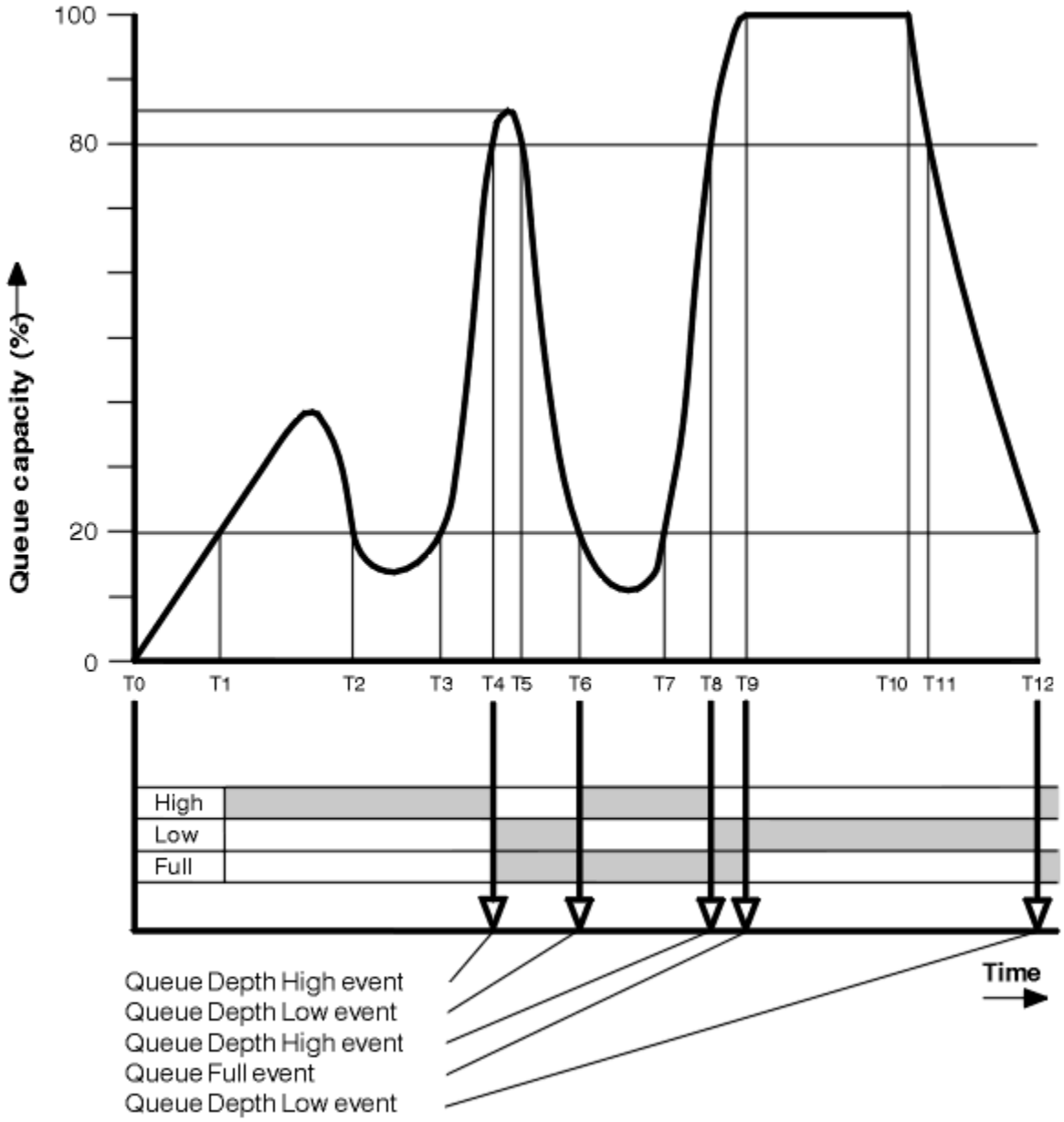


图 *8:* 队列深度事件 *(2)*

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 32 页的表 10 汇总了队列事件统计信息， 第 32 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 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。

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。

对于更改对象事件，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。

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

相关概念

第 11 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 7 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 32 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

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 32 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

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 32 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，在分布式平台上，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 7 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

未生成命令事件时

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 35 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

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 35 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 36 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

记录器事件

记录器事件是队列管理器已开始写入新的日志扩展数据块的通知。

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。

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 7 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

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。

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迂到错误，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 37 页的『记录器事件』
记录器事件是队列管理器已开始写入新的日志扩展数据块的通知。

相关参考

LoggerEvent (MQLONG)
重置队列管理器

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 37 页的『记录器事件』
记录器事件是队列管理器已开始写入新的日志扩展数据块的通知。

相关参考

LoggerEvent (MQLONG)
第 37 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

用于监视记录器事件队列的样本程序

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 output*/
/*               a message to stdout when a logger event occurs               */
/* <N_OCO_COPYRIGHT>                                                          */
/* Licensed Materials - Property of IBM                                       */
/*                                                                            */
/* 63H9336                                                                    */
/* (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.                                                                  */
/* <NOC_COPYRIGHT>                                                            */
/******************************************************************************/
/*                                                                            */
/* Function: AMQSLOG is a sample program which monitors the logger event      */
/* queue for new event messages, reads those messages, and puts the 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_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"};

/****************************************************************************/
/* 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;
  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 = (char *)malloc(MAX_MESSAGE_LENGTH);
  if (!pBuffer)
  {
    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 != MQCC_OK)
  {
    printf("Error in call to MQCONN, Reason %d, CompCode %d\n", Reason,
    CompCode);
    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)
  {
    printf("MQOPEN failed for queue manager %.48s Queue %.48s Reason: %d\n",
                                          ObjDesc.ObjectQMgrName,
                                    ObjDesc.ObjectName,
                                    Reason);
    goto MOD_EXIT;
  }
  else
  {
    ProcessPCF(hConn, hEventQueue, pBuffer);
  }

  MOD_EXIT:

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

  /*******************************************************************/
  /* 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 msg*/
/*                                                                           */
/* Output Parameters: None                                                   */
/*                                                                           */
/* Logic: Wait for messages to appear on the logger event queue and display  */
/* their 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    Reason  = 0;
  MQLONG    CompCode;
  MQLONG    MsgLen;
  PMQCHAR   Parm = NULL;
                                     /* Set timeout value           */
  Gmo.Options     |= MQGMO_WAIT;
  Gmo.Options |= MQGMO_CONVERT;
  Gmo.WaitInterval = MQWI_UNLIMITED;
  /*******************************************************************/
  /* Process response Queue                                          */
  /*******************************************************************/
  while (Reason == MQCC_OK)
  {
    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 != MQCC_OK)
{
  switch(Reason)
  {
    case MQRC_NO_MSG_AVAILABLE:
         printf("Timed out");
         break;

    default:
         printf("MQGET failed RC(%d)\n", Reason);
         break;
  }
  goto MOD_EXIT;
}

/*****************************************************************/
/* Only expect PCF event messages on this queue                  */
/*****************************************************************/
if (memcmp(Mqmd.Format, MQFMT_EVENT, sizeof(Mqmd.Format)))
{
 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)
{
 printf("------------------------------------------------------------------\n");
 printf("Event Message Received\n");

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

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

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

pPCFCmd  = (char *)  (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("%-32d",pCfst -> Parameter);
         }

         fwrite( pCfst -> String, pCfst -> StringLength, 1, stdout);
         pPCFCmd += pCfst -> StrucLength;
         break;
    default:
         printf("Unrecoginised datatype %d returned\n",pCfst->Type);
         goto MOD_EXIT;
  }
```

```
      putchar('\n');
    }
    printf("------------------------------------------------------------\n");
  }
MOD_EXIT:

 return;
}

/**************************************************************************/
/* Function: ParmToString                                                 */
/**************************************************************************/
/*                                                                        */
/* Input Parameters:  Parameter for which to get string description       */
/*                                                                        */
/* Output Parameters: None                                                */
/*                                                                        */
/* Logic: Takes a parameter as input and returns a pointer to a string    */
/* description for that parameter, or NULL if the parameter does not  */
/* have an associated string description                                  */
/**************************************************************************/

static PMQCHAR ParmToString(MQLONG Parameter){
  long i;
  for (i=0 ; i< sizeof(ParmTable)/sizeof(ParmTableEntry); i++)
  {
    if (ParmTable[i].ConstVal == Parameter ParmTable[i].Desc)
      return ParmTable[i].Desc;
  }
  return NULL;
}
```

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 38 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 36 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 37 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

用于监视检测事件的样本程序

使用此页面来查看用于监视检测事件的样本 C 程序

此样本程序不是任何 IBM WebSphere MQ 产品的一部分，因此未作为实际物理项提供。 此示例不完整，因为它未枚举指定操作的所有可能结果。 但是，您可以使用此样本作为您自己的程序的基础，这些程序使用事

件，尤其是事件消息中使用的 PCF 格式。 但是，您需要先修改此程序，然后才能在自己的系统上运行此程序。

```
/*********************************************************************/
/*                                                                   */
/* Program name: EVMON                                               */
/*                                                                   */
/* Description: C program that acts as an event monitor              */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/* Function:                                                         */
/*                                                                   */
/*                                                                   */
/*   EVMON is a C program that acts as an event monitor - reads an   */
/*   event queue and tells you if anything appears on it             */
/*                                                                   */
/*   Its first parameter is the queue manager name, the second is    */
/*   the event queue name. If these are not supplied it uses the     */
/*   defaults.                                                       */
/*                                                                   */
/*********************************************************************/
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#ifndef min
  #define min(a,b)        (((a) < (b)) ? (a) : (b))
#endif

/*********************************************************************/
/* includes for MQI                                                  */
/*********************************************************************/
#include <cmqc.h>
#include <cmqcfc.h>
void printfmqcfst(MQCFST* pmqcfst);
void printfmqcfin(MQCFIN* pmqcfst);
void printreas(MQLONG reason);

 #define PRINTREAS(param)                                           \
    case param:                                                     \
      printf("Reason = %s\n",#param);                               \
      break;



/*********************************************************************/
/* global variable                                                   */
/*********************************************************************/
MQCFH    *evtmsg;                   /* evtmsg message buffer       */

int main(int argc, char **argv)
{
  /*******************************************************************/
  /* declare variables                                               */
  /*******************************************************************/
  int  i;                            /* auxiliary counter          */
  /*******************************************************************/
  /* Declare MQI structures needed                                   */
  /*******************************************************************/
  MQOD     od = {MQOD_DEFAULT};      /* Object Descriptor           */
  MQMD     md = {MQMD_DEFAULT};      /* Message Descriptor          */
  MQGMO   gmo = {MQGMO_DEFAULT};     /* get message options         */
  /*******************************************************************/
  /* note, uses defaults where it can                                */
  /*******************************************************************/

  MQHCONN  Hcon;                     /* connection handle           */
  MQHOBJ   Hobj;                     /* object handle               */
  MQLONG   O_options;                /* MQOPEN options              */
  MQLONG   C_options;                /* MQCLOSE options             */
  MQLONG   CompCode;                 /* completion code             */
  MQLONG   OpenCode;                 /* MQOPEN completion code      */
  MQLONG   Reason;                   /* reason code                 */
  MQLONG   CReason;                  /* reason code for MQCONN      */
  MQLONG   buflen;                   /* buffer length               */
  MQLONG   evtmsglen;                /* message length received     */
  MQCHAR   command[1100];            /* call command string ...     */
```

```
MQCHAR   p1[600];                  /* ApplId insert               */
MQCHAR   p2[900];                  /* evtmsg insert               */
MQCHAR   p3[600];                  /* Environment insert          */
MQLONG   mytype;                   /* saved application type      */
char     QMName[50];               /* queue manager name          */
MQCFST  *paras;                    /* the parameters              */
int      counter;                  /* loop counter                */
time_t   ltime;

/*****************************************************************/
/* Connect to queue manager                                      */
/*****************************************************************/
QMName[0] = 0;                      /* default queue manager      */
if (argc > 1)
  strcpy(QMName, argv[1]);
MQCONN(QMName,                      /* queue manager              */
       &Hcon,                   /* connection handle          */
       &CompCode,               /* completion code            */
       &CReason);               /* reason code                */



/*****************************************************************/
/* Initialize object descriptor for subject queue                */
/*****************************************************************/
strcpy(od.ObjectName, "SYSTEM.ADMIN.QMGR.EVENT");
if (argc > 2)
  strcpy(od.ObjectName, argv[2]);

/*****************************************************************/
/* Open the event queue for input; exclusive or shared.  Use of  */
/* the queue is controlled by the queue definition here          */
/*****************************************************************/

O_options = MQOO_INPUT_AS_Q_DEF     /* open queue for input       */
      + MQOO_FAIL_IF_QUIESCING      /* but not if qmgr stopping   */
      + MQOO_BROWSE;
MQOPEN(Hcon,                        /* connection handle          */
       &od,                     /* object descriptor for queue*/
       O_options,                   /* open options               */
       &Hobj,                   /* object handle              */
       &CompCode,               /* completion code            */
       &Reason);                /* reason code                */

/*****************************************************************/
/*   Get messages from the message queue                         */
/*****************************************************************/
while (CompCode != MQCC_FAILED)
{
  /***************************************************************/
  /* I don't know how big this message is so just get the        */
  /* descriptor first                                            */
  /***************************************************************/
  gmo.Options = MQGMO_WAIT + MQGMO_LOCK
     + MQGMO_BROWSE_FIRST + MQGMO_ACCEPT_TRUNCATED_MSG;
                                    /* wait for new messages      */
  gmo.WaitInterval = MQWI_UNLIMITED;/* no time limit              */
  buflen = 0;                       /* amount of message to get   */

  /***************************************************************/
  /* clear selectors to get messages in sequence                 */
  /***************************************************************/
  memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
  memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));

  /***************************************************************/
  /* wait for event message                                      */
  /***************************************************************/
  printf("...>\n");
  MQGET(Hcon,                       /* connection handle          */
        Hobj,                       /* object handle              */
        &md,                    /* message descriptor         */
        &gmo,                   /* get message options        */
        buflen,                     /* buffer length              */
        evtmsg,                     /* evtmsg message buffer      */
        &evtmsglen,             /* message length             */
        &CompCode,              /* completion code            */
        &Reason);               /* reason code                */



  /***************************************************************/
```

```
/* report reason, if any                                             */
/*******************************************************************/
 if (Reason != MQRC_NONE && Reason != MQRC_TRUNCATED_MSG_ACCEPTED)
{
  printf("MQGET ==> %ld\n", Reason);
}
else
{
  gmo.Options = MQGMO_NO_WAIT + MQGMO_MSG_UNDER_CURSOR;
  buflen = evtmsglen;             /* amount of message to get   */
  evtmsg = malloc(buflen);
  if (evtmsg != NULL)
  {
    /*****************************************************************/
    /* clear selectors to get messages in sequence                   */
    /*****************************************************************/
    memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
    memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));

    /*****************************************************************/
    /* get the event message                                         */
    /*****************************************************************/
    printf("...>\n");
    MQGET(Hcon,                      /* connection handle        */
          Hobj,                      /* object handle            */
          &md,                  /* message descriptor       */
          &gmo,                 /* get message options      */
          buflen,                    /* buffer length            */
          evtmsg,                    /* evtmsg message buffer    */
          &evtmsglen,           /* message length           */
          &CompCode,            /* completion code          */
          &Reason);             /* reason code              */

    /*****************************************************************/
    /* report reason, if any                                         */
    /*****************************************************************/
    if (Reason != MQRC_NONE)
    {
      printf("MQGET ==> %ld\n", Reason);
    }
  }
  else
  {
    CompCode = MQCC_FAILED;
  }
}


/*******************************************************************/
/* . . . process each message received                             */
/*******************************************************************/

if (CompCode != MQCC_FAILED)
{
  /*****************************************************************/
  /* announce a message                                            */
  /*****************************************************************/
  printf("\a\a\a\a\a\a\a");
  time(&ltime);
  printf(ctime(&ltime));

  if (evtmsglen != buflen)
    printf("DataLength = %ld?\n", evtmsglen);
  else
  {
    /*****************************************************************/
    /* right let's look at the data                                  */
    /*****************************************************************/
    if (evtmsg->Type != MQCFT_EVENT)
    {
      printf("Something's wrong this isn't an event message,"
             " its type is %ld\n",evtmsg->Type);
    }
    else
    {
      if (evtmsg->Command == MQCMD_Q_MGR_EVENT)
      {
        printf("Queue Manager event: ");
      }
      else
        if (evtmsg->Command == MQCMD_CHANNEL_EVENT)
        {
```

```
              printf("Channel event: ");
            }
            else
       ⋮


            {
              printf("Unknown Event message, %ld.",
                      evtmsg->Command);
            }

          if      (evtmsg->CompCode == MQCC_OK)
            printf("CompCode(OK)\n");
          else if (evtmsg->CompCode == MQCC_WARNING)
            printf("CompCode(WARNING)\n");
          else if (evtmsg->CompCode == MQCC_FAILED)
            printf("CompCode(FAILED)\n");
          else
            printf("* CompCode wrong * (%ld)\n",
                      evtmsg->CompCode);

          if (evtmsg->StrucLength != MQCFH_STRUC_LENGTH)
          {
            printf("it's the wrong length, %ld\n",evtmsg->StrucLength);
          }

          if (evtmsg->Version != MQCFH_VERSION_1)
          {
            printf("it's the wrong version, %ld\n",evtmsg->Version);
          }

          if (evtmsg->MsgSeqNumber != 1)
          {
            printf("it's the wrong sequence number, %ld\n",
                    evtmsg->MsgSeqNumber);
          }

          if (evtmsg->Control != MQCFC_LAST)
          {
            printf("it's the wrong control option, %ld\n",
                    evtmsg->Control);
          }

          printreas(evtmsg->Reason);
          printf("parameter count is %ld\n", evtmsg->ParameterCount);
          /*******************************************************/
          /* get a pointer to the start of the parameters        */
          /*******************************************************/


          paras = (MQCFST *)(evtmsg + 1);
          counter = 1;
          while (counter <= evtmsg->ParameterCount)
          {
            switch (paras->Type)
            {
              case MQCFT_STRING:
                printfmqcfst(paras);
                paras = (MQCFST *)((char *)paras
                                   + paras->StrucLength);
                break;
              case MQCFT_INTEGER:
                printfmqcfin((MQCFIN*)paras);
                paras = (MQCFST *)((char *)paras
                                   + paras->StrucLength);
                break;
              default:
                printf("unknown parameter type, %ld\n",
                       paras->Type);
                counter = evtmsg->ParameterCount;
                break;
            }
            counter++;
          }
        }
      }   /* end evtmsg action             */
      free(evtmsg);
      evtmsg = NULL;
    }     /* end process for successful GET */
  }       /* end message processing loop    */

  /*****************************************************************/
```

```
   /* close the event queue - if it was opened                        */
   /*******************************************************************/
   if (OpenCode != MQCC_FAILED)
   {
     C_options = 0;                  /* no close options             */
     MQCLOSE(Hcon,                   /* connection handle            */
            &Hobj,               /* object handle               */
            C_options,
            &CompCode,           /* completion code             */
            &Reason);            /* reason code                 */
   /*******************************************************************/
   /* Disconnect from queue manager (unless previously connected)    */
   /*******************************************************************/
   if (CReason != MQRC_ALREADY_CONNECTED)
   {
     MQDISC(&Hcon,               /* connection handle           */
            &CompCode,           /* completion code             */
            &Reason);            /* reason code                 */


 /*******************************************************************/
 /*                                                                 */
 /* END OF EVMON                                                    */
 /*                                                                 */
 /*******************************************************************/
 }

#define PRINTPARAM(param)                                           \
   case param:                                                      \
     {                                                              \
       char *p = #param;                                            \
     strncpy(thestring,pmqcfst->String,min(sizeof(thestring),       \
             pmqcfst->StringLength));                               \
     printf("%s %s\n",p,thestring);                                 \
     }                                                              \
     break;

#define PRINTAT(param)                                              \
   case param:                                                      \
     printf("MQIA_APPL_TYPE = %s\n",#param);                        \
     break;


void printfmqcfst(MQCFST* pmqcfst)
{
  char thestring[100];

  switch (pmqcfst->Parameter)
  {
    PRINTPARAM(MQCA_BASE_Q_NAME)
    PRINTPARAM(MQCA_PROCESS_NAME)
    PRINTPARAM(MQCA_Q_MGR_NAME)
    PRINTPARAM(MQCA_Q_NAME)
    PRINTPARAM(MQCA_XMIT_Q_NAME)
    PRINTPARAM(MQCACF_APPL_NAME)
    ⋮
     default:
      printf("Invalid parameter, %ld\n",pmqcfst->Parameter);
      break;
  }
}




void printfmqcfin(MQCFIN* pmqcfst)
{
  switch (pmqcfst->Parameter)
  {
    case MQIA_APPL_TYPE:
      switch (pmqcfst->Value)
      {
        PRINTAT(MQAT_UNKNOWN)
        PRINTAT(MQAT_OS2)
        PRINTAT(MQAT_DOS)
        PRINTAT(MQAT_UNIX)
        PRINTAT(MQAT_QMGR)
        PRINTAT(MQAT_OS400)
        PRINTAT(MQAT_WINDOWS)
        PRINTAT(MQAT_CICS_VSE)
        PRINTAT(MQAT_VMS)
        PRINTAT(MQAT_GUARDIAN)
```

```
        PRINTAT(MQAT_VOS)
      }
      break;
    case MQIA_Q_TYPE:
      if (pmqcfst->Value == MQQT_ALIAS)
      {
        printf("MQIA_Q_TYPE is MQQT_ALIAS\n");
      }
      else
      ⋮
{
        if (pmqcfst->Value == MQQT_REMOTE)
        {
          printf("MQIA_Q_TYPE is MQQT_REMOTE\n");
          if (evtmsg->Reason == MQRC_ALIAS_BASE_Q_TYPE_ERROR)
          {
            printf("but remote is not valid here\n");
          }
        }
        else
        {
          printf("MQIA_Q_TYPE is wrong, %ld\n",pmqcfst->Value);
        }
      }
      break;


          case MQIACF_REASON_QUALIFIER:
      printf("MQIACF_REASON_QUALIFIER %ld\n",pmqcfst->Value);
      break;

    case MQIACF_ERROR_IDENTIFIER:
      printf("MQIACF_ERROR_INDENTIFIER %ld (X'%lX')\n",
              pmqcfst->Value,pmqcfst->Value);
      break;

    case MQIACF_AUX_ERROR_DATA_INT_1:
      printf("MQIACF_AUX_ERROR_DATA_INT_1 %ld (X'%lX')\n",
              pmqcfst->Value,pmqcfst->Value);
      break;

    case MQIACF_AUX_ERROR_DATA_INT_2:
      printf("MQIACF_AUX_ERROR_DATA_INT_2 %ld (X'%lX')\n",
              pmqcfst->Value,pmqcfst->Value);
      break;
⋮
default :
      printf("Invalid parameter, %ld\n",pmqcfst->Parameter);
      break;
  }
}

   void printreas(MQLONG reason)
{
  switch (reason)
  {
    PRINTREAS(MQRCCF_CFH_TYPE_ERROR)
    PRINTREAS(MQRCCF_CFH_LENGTH_ERROR)
    PRINTREAS(MQRCCF_CFH_VERSION_ERROR)
    PRINTREAS(MQRCCF_CFH_MSG_SEQ_NUMBER_ERR)
    ⋮
    PRINTREAS(MQRC_NO_MSG_LOCKED)
    PRINTREAS(MQRC_CONNECTION_NOT_AUTHORIZED)
    PRINTREAS(MQRC_MSG_TOO_BIG_FOR_CHANNEL)
    PRINTREAS(MQRC_CALL_IN_PROGRESS)
    default:
      printf("It's an unknown reason, %ld\n",
              reason);
      break;
  }
}
```

相关概念

第 5 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。

第 5 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。

相关参考

C 编程

第 38 页的『用于监视记录器事件队列的样本程序』

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 WebSphere 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。
- WebSphere 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 WebSphere MQ 发布/预订引擎的活动可以由以下多个操作组成:

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。

WebSphere 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 49 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 50 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 56 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。

优点	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
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。 消息路由的完整性取决于队列管理器网络中队列管理器的级别，如下所示:

WebSphere MQ V 6.0 及后续发行版中的队列管理器

在 WebSphere MQ V 6.0 或后续发行版中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。 如果队列管理器网络由 WebSphere MQ V 6.0 或后续发行版中的队列管理器组成，那么可以确定完整的消息路由。

WebSphere MQ 版本之前的队列管理器 6.0

在 V 6.0 之前连接到 WebSphere MQ 队列管理器的应用程序 **不会** 记录它们代表消息执行的活动。 如果队列管理器网络包含任何低于 V 6.0 的 WebSphere MQ 队列管理器，那么只能确定部分消息路由。

活动信息的存储方式

WebSphere 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 52 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 56 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 60 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 90 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 活动 PCF 组包含的参数。 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 WebSphere 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 WebSphere 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 活动 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 不 希望在通过队列管理器进行路由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。 除非针对跟踪路由消息生成了活动报告，否则必须手动对这些报告进行排序，在这种情况下，您可以使用 WebSphere MQ 显示路由应用程序来对活动报告进行排序。

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发生在队列管理器 QM1 上，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. 对于将活动报告交付到公共队列的队列管理器网络中的任何队列管理器，请从具有与原始消息的 *MsgId* 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:

- 该消息由低于 V 6.0 的 WebSphere MQ 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 WebSphere 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 106 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 活动 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名为 *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。 此技术由 *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。 *TraceRoute* PCF 组中返回了其他信息。 *TraceRoute* PCF 组中包含的信息仅在生成活动报告时才准确。 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 56 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 WebSphere 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 WebSphere 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 53 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 59 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 59 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 WebSphere MQ 显示路由应用程序。

相关任务

第 53 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：

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，并且要对跟踪路由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，并且要对跟踪路由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 106 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，请增大 *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 累计 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_累积性 _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 WebSphere 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 格式 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 活动 PCF 组。 活动 当 MCA 或用户编写的应用程序代表其执行活动时，PCF 组将写入跟踪路由消息的消息数据。

WebSphere MQ 显示路由应用程序

使用 WebSphere 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 WebSphere MQ 显示路由应用程序生成的跟踪路由消息。

限制: 无法在 WebSphere MQ Version 6.0 之前的队列管理器上或在 WebSphere MQ for z/OS 队列管理器上发出 `dspmqrte` 。 如果您希望通过第一个队列管理器传递跟踪路由消息成为此类型的队列管理器，请使用可选参数 `-c` 作为 WebSphere MQ V 6.0 或更高版本的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:

优先级
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: WebSphere 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: WebSphere 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 64 页的『其他活动信息』中所述。

表 *12: TraceRoute PCF* 组

参数	类型
TraceRoute	MQCFGR
详细信息	MQCFIN
RecordedActivities	MQCFIN
UnrecordedActivities	MQCFIN
DiscontinuityCount	MQCFIN
MaxActivities	MQCFIN
累计	MQCFIN
向前	MQCFIN
传递	MQCFIN

TraceRoute PCF 组中每个参数的描述如下:

详细信息
指定要记录的活动信息的详细级别。 值可以是:

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* 将递增 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。

累积
指定用于累积活动信息的方法。 值可以是:

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:

- WebSphere MQ V 6 (或更高版本) 队列管理器将废弃跟踪路由消息。
- WebSphere MQ V 6 (或更高版本) 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)。
- 在跟踪路由消息上执行的活动数超过 *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。

前进
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
无论是否将采用 *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 62 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 62 页的『2』
2. 确定 *TraceRoute* 组中的 *Deliver* 参数是否包含 MQROUTE_DELI_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 62 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 62 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 62 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:

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 WebSphere MQ 显示路由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 WebSphere MQ 显示路由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 WebSphere MQ 显示路由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 活动 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 WebSphere 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 WebSphere 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
组中的参数	*ParameterName* *ParameterValue*

ParameterName

表 *14:* 参数名	
描述	**包含要由 IBM WebSphere 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*.

表 *14:* 参数名 *(*继续*)*	
描述	**包含要由 IBM WebSphere 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。**
值:	要显示的名称。

ParameterValue

表 *15:* 参数值	
描述	**包含要由 IBM WebSphere 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 WebSphere 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。

示例 *1*

其他活动信息由用户应用程序以 WebSphere MQ 显示路由应用程序无法识别参数标识 的格式记录。

1. WebSphere 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， WebSphere 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更多信息，但是它是以 WebSphere MQ 显示路由应用程序 无法识别参数标识的格式编写的。
3. 跟踪路由消息到达目标队列，并将跟踪路由应答消息返回到 WebSphere MQ 显示路由应用程序。 其他活动信息如下所示:

   ```
   65536: 'Red'
   ```

   WebSphere 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 WebSphere 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 66 页的『示例 2』。

示例 2

其他活动信息由用户应用程序以 IBM WebSphere MQ 显示路由应用程序识别的参数标识 是 的格式记录。

1. IBM WebSphere MQ 显示路由应用程序用于以与 第 65 页的『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 WebSphere MQ 显示路径应用程序显示的 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 WebSphere 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*.
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，其中参数标识是 由 IBM WebSphere 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 WebSphere 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 WebSphere 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。

WebSphere MQ 显示路由应用程序

使用 WebSphere MQ 显示路由应用程序 (**dspmqrte**) 通过命令行界面来处理与跟踪路由消息相关的跟踪路由消息和活动信息。

注: 要对队列管理器运行客户机应用程序，必须安装 "客户机连接" 功能部件。

您可以将 WebSphere MQ 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 WebSphere 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之后，可以立即订购并显示该信息。 或者，可以使用 WebSphere 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 WebSphere 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路由。

相关参考

长石

队列管理器连接

使用此页面来指定 WebSphere MQ 显示路由应用程序连接到的队列管理器

-c

指定 WebSphere 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 WebSphere 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*

WebSphere MQ 显示路由应用程序所连接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 WebSphere 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 *TargetTopicString*

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，那么将使用 WebSphere MQ 显示路由应用程序所连接的队列管理器作为目标队列管理器。

-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 WebSphere 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 WebSphere 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 WebSphere 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 WebSphere 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 *Persistence*
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)。

否
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 *Persistence* 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 *Priority*
指定跟踪路由消息的优先级。 *Priority* 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 *Expiry*
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 none | *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。

到期
设置了报告选项 MQRO_EXPIRATION_WITH_FULL_DATA。

废弃
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 WebSphere 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:

activity 报告选项，使用 -ro 指定
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，那么活动信息 **不会** 累积在跟踪路由消息中。

-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 WebSphere MQ 队列管理器将废弃跟踪路由消息。
- IBM WebSphere 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，那么活动信息 **不会** 累积在跟踪路由消息中。

-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，那么跟踪路由消息 **不会** 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 *Detail*
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。

中

记录在 **low** 中指定的活动。 此外，还会记录由 MCA 执行的发布活动和活动。

high

记录在 **low** 和 **medium** 中指定的活动。 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 WebSphere 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 WebSphere 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 WebSphere 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，那么必须将应答队列指定为 **非** 临时动态队列。

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 *ReplyToQMgr*
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 WebSphere 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 *Deliver*
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:

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。
NO	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，那么跟踪路由消息 **不会** 放入目标队列。

-f *Forward*
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 60 页的『TraceRoute PCF 组』。 正向 的可能值为:

all
跟踪路由消息将转发到任何队列管理器。

警告: 如果转发到版本低于 6.0 的 IBM WebSphere MQ 队列管理器，那么将无法识别跟踪路由消息，并且无论 *-d Deliver* 参数的值如何，都可以将该消息传递到本地队列。

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采用 *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 *Activities*
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 *Expiry*
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。

否
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。

discard 报告选项，使用 -ro 指定
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 WebSphere 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 WebSphere 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 WebSphere 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 WebSphere 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 WebSphere 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 的格式为 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 WebSphere 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 WebSphere 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 summary | all | none | outline *DisplayOption*

摘要

将显示跟踪路由消息经过的队列。

all

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*。

message

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 WebSphere 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 64 页的『其他活动信息』中所述。

WebSphere MQ 显示路由应用程序示例

以下示例显示如何使用 WebSphere 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， WebSphere 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 WebSphere MQ 显示路由应用程序的活动信息将按顺序放置并显示。

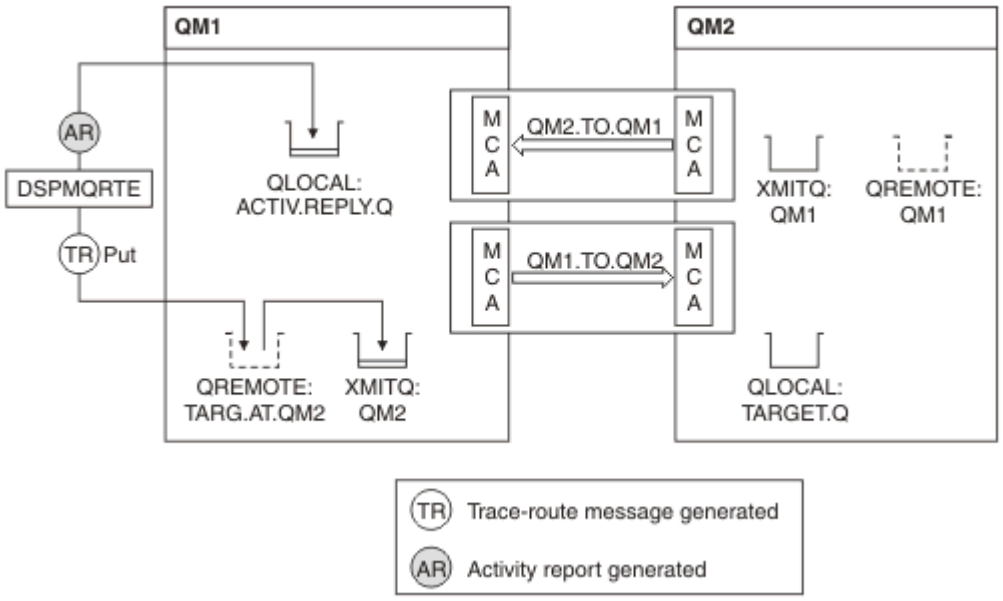


图 *9:* 请求活动报告，图 *1*

- 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 WebSphere 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。

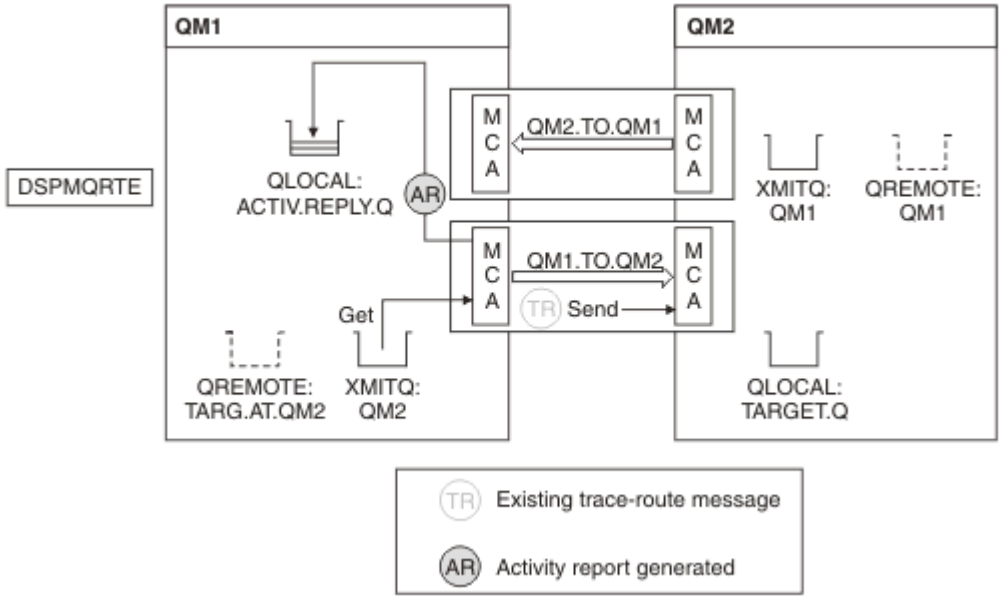


图 *10:* 请求活动报告，图 *2*

- 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 (步骤 第 62 页的『1』， 第 62 页的『4』和 第 62 页的『5』) 中描述的算法，并且 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)。

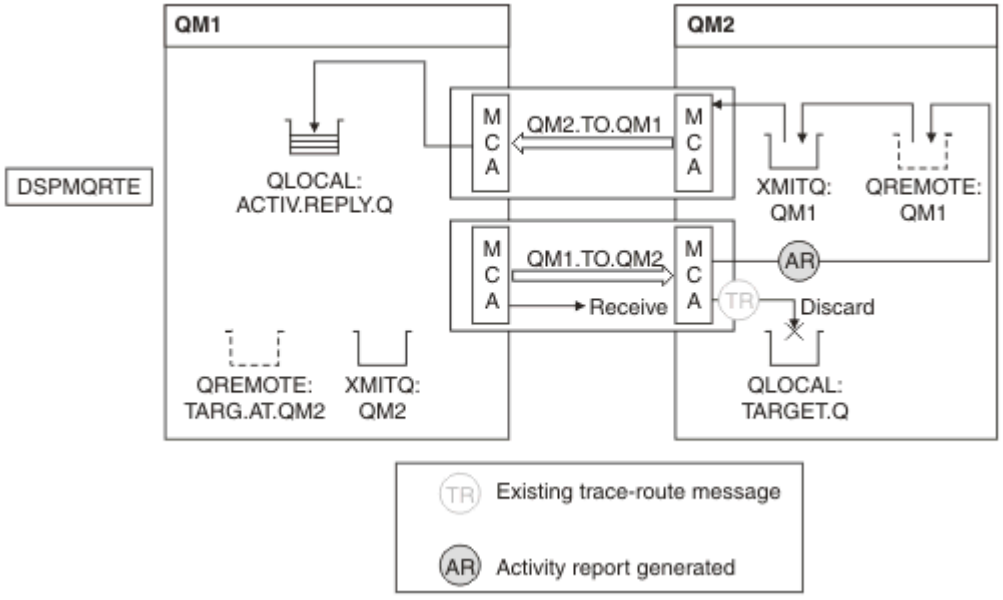


图 *11:* 请求活动报告，图 *3*

- 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 V 5.3.1 或更低版本，那么 MCA 将按 1 递增 TraceRoute PCF 的 DiscontinuityCount 参数。 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)。

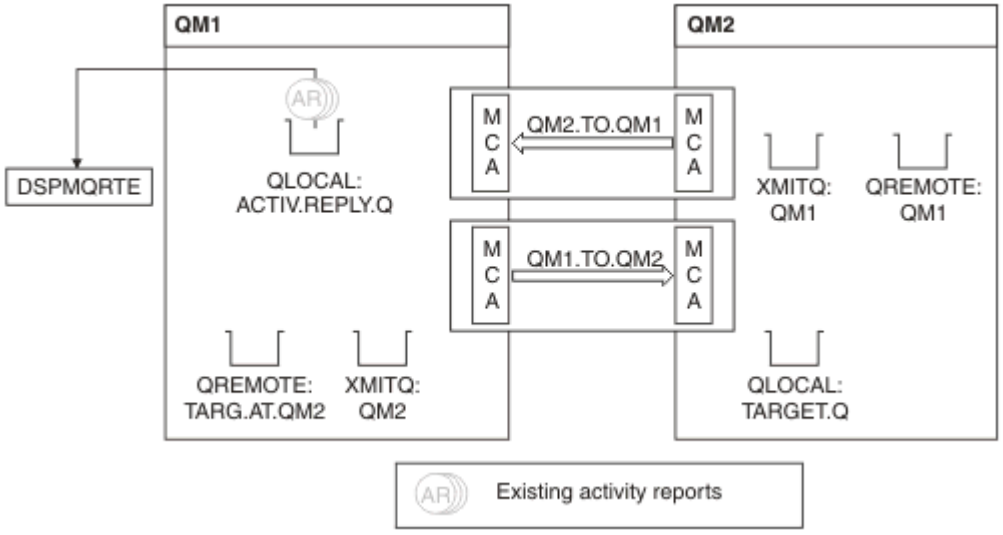


图 *12:* 请求活动报告，图 *4*

- 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， WebSphere 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。

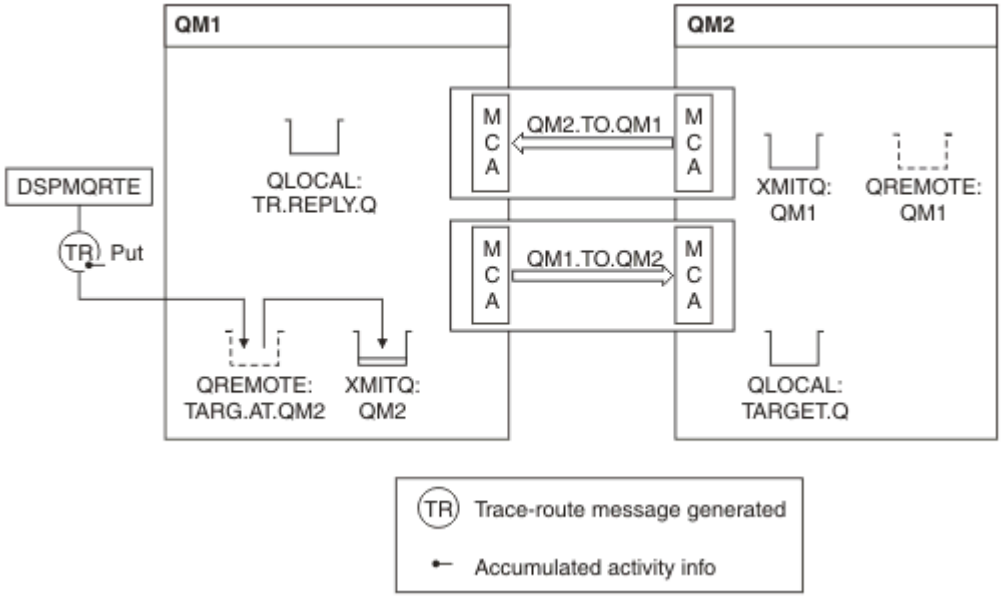


图 *13:* 请求跟踪路由应答消息，图 *1*

- 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 WebSphere 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。

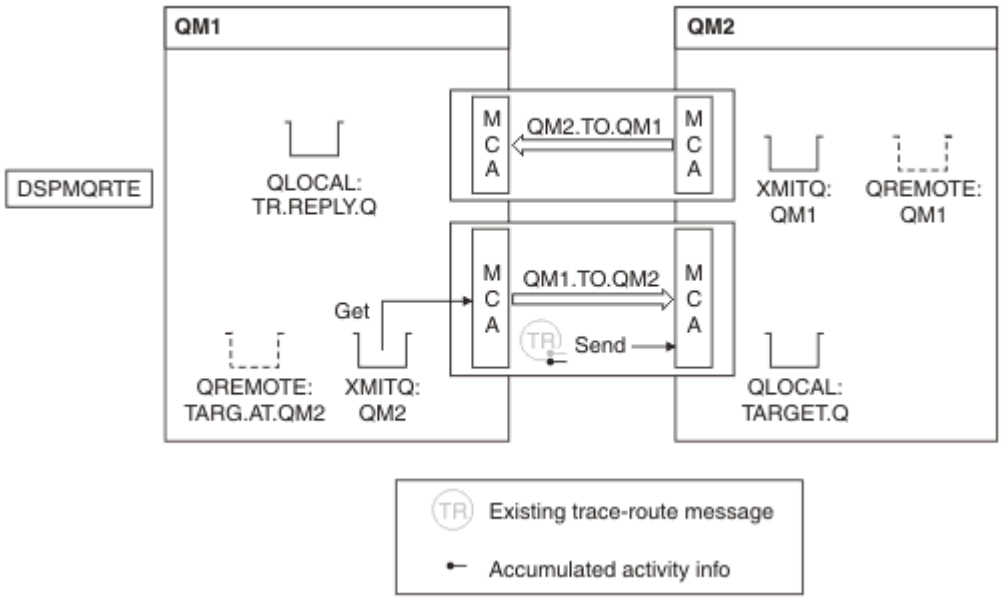


图 *14:* 请求跟踪路由应答消息，图 *2*

- 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。

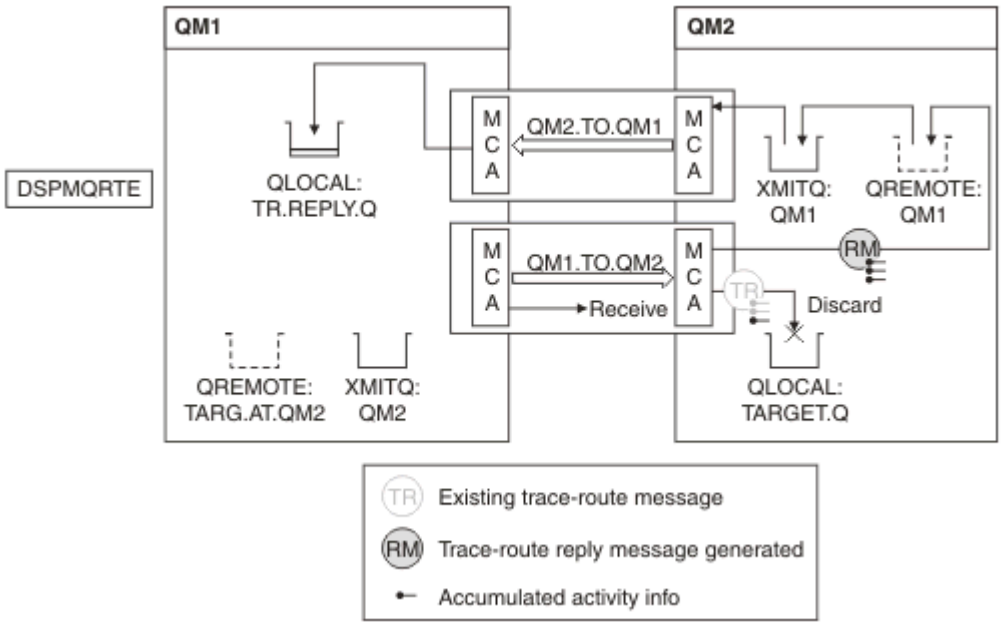


图 *15:* 请求跟踪路由应答消息，图 *3*

- 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。

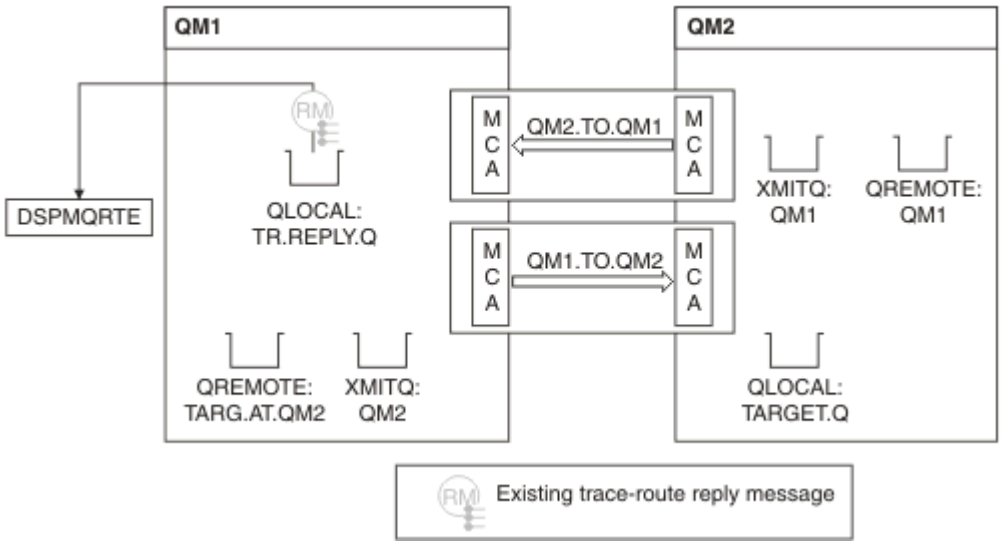


图 *16:* 请求跟踪路由应答消息，图 *4*

- 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 WebSphere 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 72 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， WebSphere MQ 显示路由应用程序将用于从另一个队列读取活动报告。

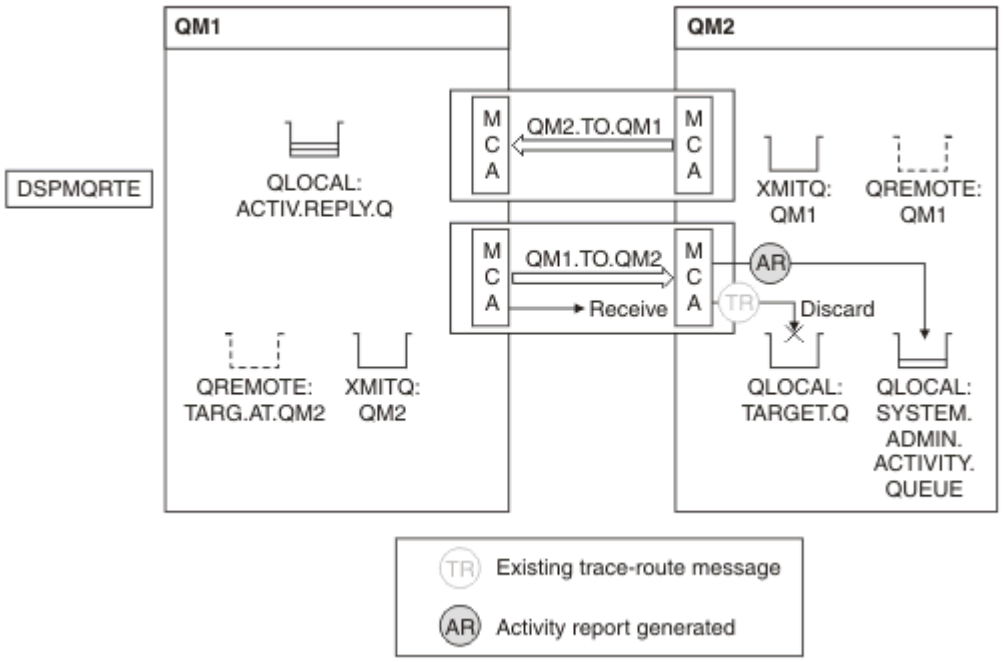


图 *17:* 正在将活动报告交付到系统队列，图 *1*

- 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)。

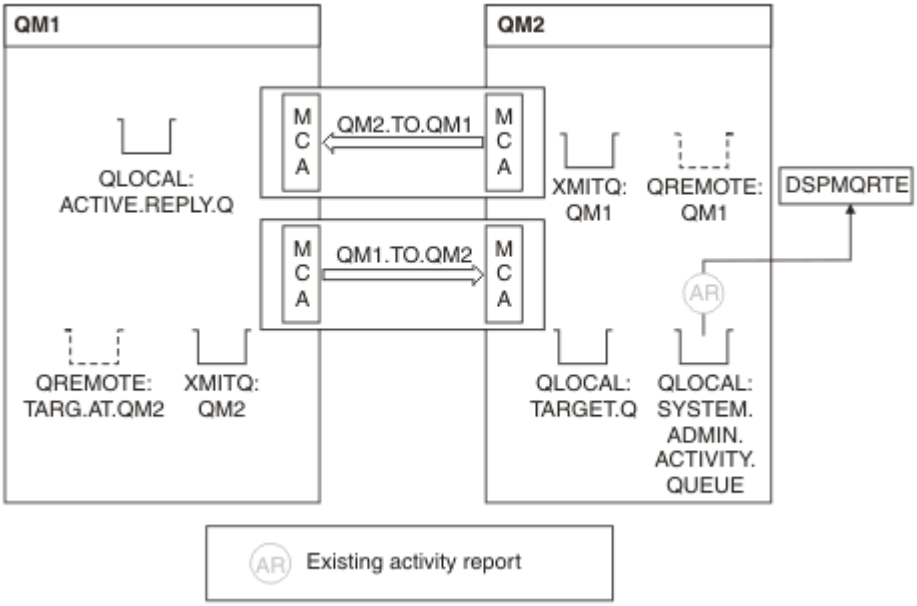


图 *18:* 正在将活动报告交付到系统队列，图 *2*

- 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， WebSphere 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。

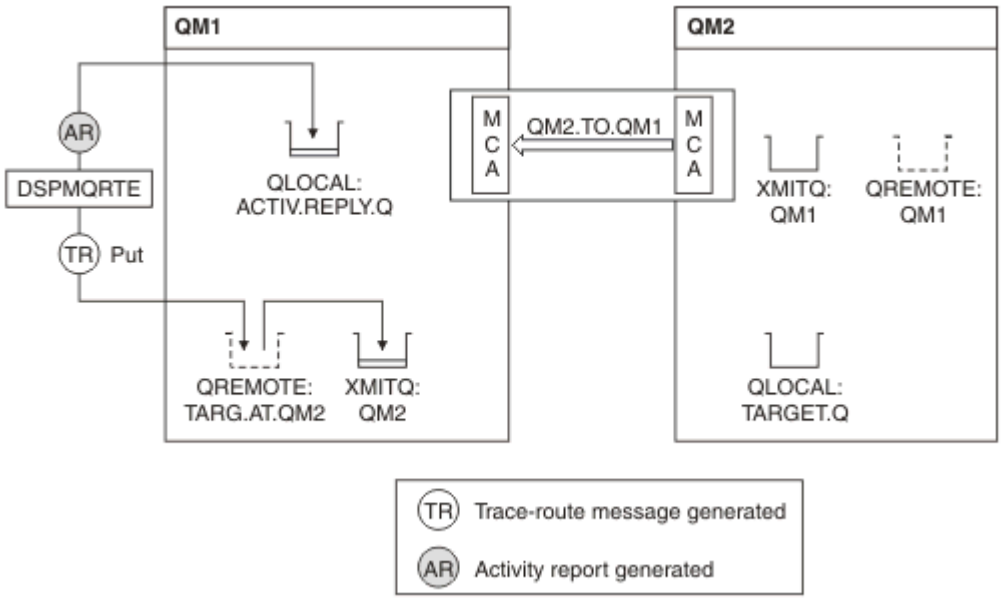


图 *19:* 诊断通道问题

- 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 WebSphere 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 活动 PCF 组以及 *TraceRoute* PCF 组 (如果为跟踪路由消息生成) 组成。

第 83 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 Feedback 编码 编码字符集标识 消息格式 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 控件选项 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] 详细信息 记录的活动 未记录的活动 不连续计数 最大活动数 累计 传递

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 AIX， HP-UX， z/OS， IBM i， Solaris， Linux， Windows 以及所有连接到这些系统的 WebSphere MQ MQI 客户机上受支持。

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 *V* 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
指定在每个结构的 *CodedCharSetId* 字段中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 89 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 活动 PCF 组包含的参数。 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 活动 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: 始终。

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: 始终。

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: 始终。

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: 始终。

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: 始终。

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: 始终。

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: 始终。

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: 始终。

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_QUEUE。

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: 如果将 *V* 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: 如果将 *V* 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: 如果将 *V* 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: 如果将 *V* 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: 如果将 *V* 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: 始终。

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 WebSphere 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 (已获取或已浏览队列上的消息)。

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: 始终。

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: 始终。

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: 始终。

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: 始终。

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: 始终。

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)*

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

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)*

在 "接收" (MQOPER_RECEIVE) 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 (在通道上接收到消息)。

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: 始终。

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: 始终。

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: 始终。

发送 (MQOPER_SEND)

针对 "发送" (MQOPER_SEND) 操作类型 (在通道上发送消息) , 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: 始终。

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: 始终。

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: 始终。

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: 始终。

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 WebSphere 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 WebSphere 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_蓄电池 _none ，那么跟踪路由消息数据将包含 *TraceRoute* PCF 组。
- 如果将 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将包含 *TraceRoute* PCF 组和零个或多个 活动 PCF 组。

第 107 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 Feedback 编码 编码字符集标识 消息格式 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 控件选项 完成代码 原因码 参数计数	TraceRoute 详细信息 记录的活动 未记录的活动 不连续计数 最大活动数 累计 传递

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_累计 _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
指定在每个结构的 *CodedCharSetId* 字段中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 112 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 组以及零个或多个 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
无论是否将采用 *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 WebSphere 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 116 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 Feedback 编码 编码字符集标识 消息格式 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	PCF 头 (MQCFH) 结构类型 结构长度 结构版本 命令标识符 消息序号 控件选项 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 详细信息 记录的活动 未记录的活动 不连续计数 最大活动数 累计 传递

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 90 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 WebSphere MQ 应用程序执行的 MQI 操作的信息，或者记录有关 IBM WebSphere 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 WebSphere MQ 应用程序执行的 MQI 操作的信息，请参阅 第 118 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 WebSphere MQ 系统中发生的活动的信息，请参阅 第 120 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。

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。

记帐消息

记帐消息记录有关 WebSphere 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 WebSphere 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。

相关参考
第 133 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 143 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* ，序列中的最后一条消息具有指定为 MQCFC_LAST 的 MQCFH 中的 *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。

MQI 记帐信息

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:

启用
针对到队列管理器的每个连接收集 MQI 记帐信息。

关闭
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。 队列属性 ACCTQ 可以具有以下值:

启用
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。

关闭
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:

启用
将针对队列属性 ACCTQ 设置为 QMGR 的队列收集队列记帐信息。

关闭
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)
```

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)
```

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。 对于长时间运行的 WebSphere MQ 应用程序，还会写入中间记帐消息。

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时，将为长时间运行的 WebSphere 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 WebSphere 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。

相关参考
第 122 页的『MQI 统计信息』
使用队列管理器属性 STATMQI 来控制 MQI 统计信息的收集

第 122 页的『队列统计信息』
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 123 页的『通道统计信息』
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。 这些属性在分布式平台和 IBM i 上可用。

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。

在时间间隔内，每个队列都有一个 *QStatisticsData* PCF 组处于活动状态。 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*，序列中的最后一条消息具有指定为 MQCFC_LAST 的 MQCFH 中的 *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 *ChlStatisticsData* PCF 组。

对于在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* ，序列中的最后一条消息具有指定为 MQCFC_LAST 的 MQCFH 中的 *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

MQI 统计信息

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:

启用
针对与队列管理器的每个连接收集 MQI 统计信息。

关闭
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)
```

队列统计信息

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 WebSphere 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:

启用
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。

关闭
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:

启用
针对将队列属性 STATQ 设置为 QMGR 的队列收集队列统计信息

关闭
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)
```

distributed 通道统计信息

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。 这些属性在分布式平台和 IBM i 上可用。

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 WebSphere 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 123 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	说明	用途
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。

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
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。

关闭
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。

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
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。

关闭
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。

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
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。

关闭
统计信息不适用于自动定义的集群发送方通道。

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。

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

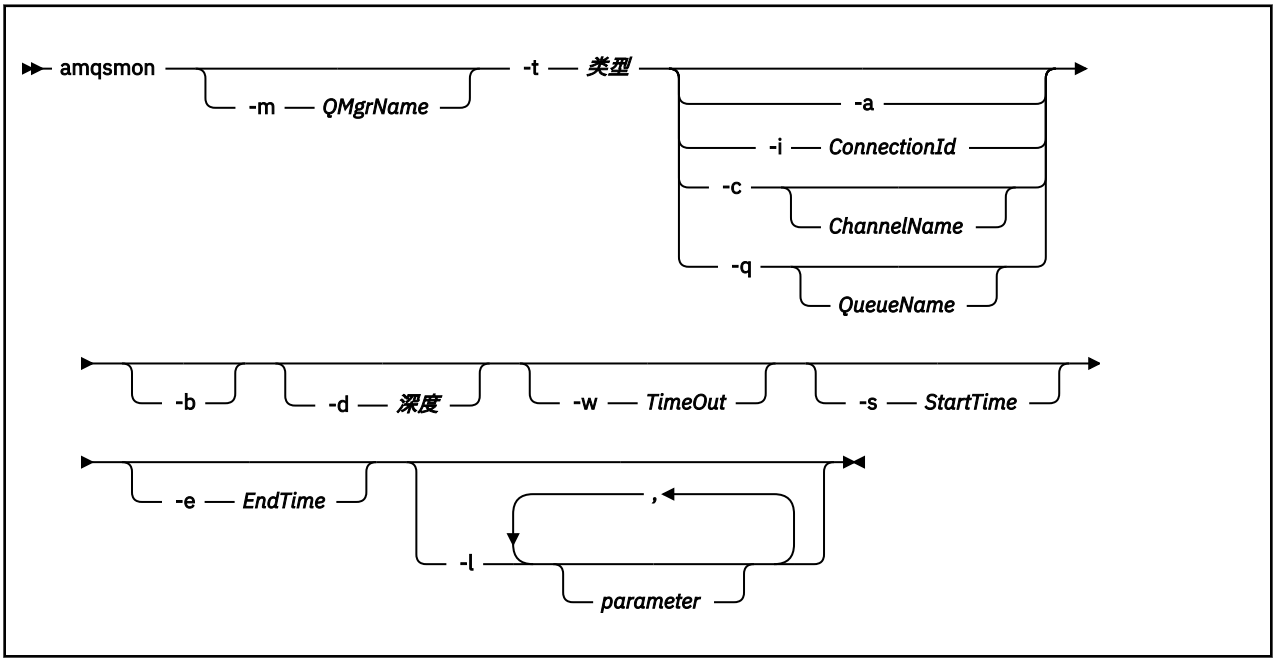
记帐和统计信息消息将写入系统记帐和统计信息队列。 **amqsmon** 是随 WebSphere MQ 提供的样本程序，用于处理来自记帐和统计信息队列的消息，并以可读格式将信息显示到屏幕上。

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:

记帐
将处理记帐记录。 从系统队列 SYSTEM.ADMIN.ACCOUNTING.QUEUE。

statistics
处理统计信息记录。 从系统队列 SYSTEM.ADMIN.STATISTICS.QUEUE。

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时，此参数可用 (*-t statistics*)。

-i *ConnectionId*
仅显示与 *ConnectionId* 指定的连接标识相关的记录。

仅当显示记帐消息时，此参数才可用 (*-t* 记帐)。

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。

-b
浏览消息。

以非破坏性方式检索消息。

-d *Depth*
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*
仅处理放入指定 *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*
仅处理放在指定 *EndTime* 之前的消息。

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 *Parameter*
仅显示所处理记录中的所选字段。 参数 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 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. 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. 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 -a -l 7006,3024
```

此命令的输出如下所示:

```
    ConnectionId: x'414d514354524556312020202020208d0b374203090020'
    ApplicationName: 'runmqsc'

    ConnectionId: x'414d514354524556312020202020208d0b3742010a0020'
    ApplicationName: 'amqsput'

    ConnectionId: x'414d514354524556312020202020208d0b3742010c0020'
    ApplicationName: 'runmqsc'

    ConnectionId: x'414d514354524556312020202020208d0b3742010d0020'
    ApplicationName: 'amqsput'

    ConnectionId: x'414d514354524556312020202020208d0b3742150d0020'
    ApplicationName: 'amqsget'

    5 Records Processed.
```

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 WebSphere MQ 消息。 消息数据包含有关 WebSphere MQ 应用程序执行的 MQI 操作的信息，或者有关 WebSphere 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 控件选项 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 *ChlStatisticsData* 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:	全部 (WebSphere 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: WebSphere 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 WebSphere 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

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: 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:	全部 (WebSphere 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: WebSphere 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 WebSphere 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

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

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。

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

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:	全部 (WebSphere 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: 始终。

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: 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: 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。

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: 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: 成功关闭对象尝试的次数。 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:	全部 (WebSphere 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 WebSphere MQ 能够绕过队列的次数，而不是应用程序正在等待的次数。

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

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。

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:	全部 (WebSphere 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:	始终。

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:	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: 始终。

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: 始终。

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: 始终。

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: 始终。

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 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 133 页的『MQI 记帐消息数据』
- 第 143 页的『队列记帐消息数据』
- 第 153 页的『MQI 统计信息消息数据』
- 第 163 页的『队列统计信息消息数据』
- 第 169 页的『通道统计信息消息数据』

1. 此参数与 WebSphere 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 WebSphere 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 WebSphere 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。

收集应用程序活动跟踪信息

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。 要收集应用程序活动跟踪信息，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 181 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 182 页的『amqsact 样本程序』。

过程

1. 第 176 页的『设置 ACTVTRC 以控制活动跟踪信息的收集』.
2. 第 176 页的『设置 MQCONNX 选项以控制活动跟踪信息的收集』.
3. 第 177 页的『使用 mqat.ini 配置活动跟踪行为』.

4. 第 181 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:

启用
已开启 API 活动跟踪收集

关闭
已关闭 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 176 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 182 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 181 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
   已关闭连接的活动跟踪。

   MQCNO_ ACTIVITY_ TRACE_ENABLED
   为连接打开了活动跟踪。

   注: 如果应用程序针对 MQCONNX 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 177 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 182 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 181 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。 此文件遵循与 mqs.ini 和 qm.ini 文件相同的节键/参数/值对格式。

关于此任务

Linux UNIX 在 UNIX 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 Version 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。

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。

节键

配置文件中允许两种节键类型: AllActivityTrace 节和 ApplicationTrace 节

AllActivity 跟踪节

AllActivityTrace 节定义应用于所有 IBM WebSphere 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 节中的更具体的信息覆盖。

如果指定了多个 AllActivity 跟踪节，那么将使用最后一个节中的值。 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

ApplicationTrace 节

ApplicationTrace 节定义可应用于 IBM WebSphere MQ 连接的特定名称和/或类型的设置。

此节包含 ApplName 和 ApplClass 值，这些值根据 "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。

参数/值对

下表列出了可在活动跟踪配置文件中使用的参数/值对。

表 *26:* 可在活动跟踪配置文件中使用的参数/值对

名称	**节类型**	**值 (缺省为粗体类型)**	**描述**
跟踪	ApplicationTrace	开启/ **关闭**	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。
ActivityInterval	AllActivity 跟踪 ApplicationTrace	**0**-99999999 (**0=off**)	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。
ActivityCount	AllActivity 跟踪 ApplicationTrace	**0**-99999999 (**0=off**)	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时) 写入跟踪消息。
TraceLevel	AllActivity 跟踪 ApplicationTrace	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	AllActivity 跟踪 ApplicationTrace	**0**-104 857 600 (100Mb)	针对 MQGET，MQPUT，MQPUT1 和回调操作跟踪的消息数据量 (以字节计)

表 *26:* 可在活动跟踪配置文件中使用的参数/值对 *(*继续*)*			
名称	**节类型**	**值 (缺省为粗体类型)**	**描述**
ApplName	ApplicationTrace	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。 对于分布式平台，仅 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。 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。
ApplClass	ApplicationTrace	**用户** /MCA/内部/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 WebSphere MQ 连接的说明

下表显示了 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 *27: 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**
USER	MQXACT_EXTERNAL	MQXE_OTHER	仅跟踪用户应用程序
MCA	(任何值)	MQXE_MCA MQXE_MCA_CLNTCONN MQXE_MCA_SVRCONN	客户机和通道 (amqrmppa)
内部	MQXACT_EXTERNAL	MQXE_COMMAND_SERVER MQXE_MQSC	"runmqsc" 和命令服务器
内部	MQXACT_INTERNAL	(任何值)	"可信" 和内部应用程序和流程; 例如， amqzdmaa
ALL	(任何值)	(任何值)	跟踪所有用户和内部连接

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
```

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 WebSphere 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 ApplName 节包含通配符 (*)，那么仅将通配符左侧的字符与连接 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 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 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 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，有多个节与连接 ApplName 和 ApplClass 匹配，那么将使用最后匹配的值，并且将忽略所有其他节。

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。 此示例在 C 样本目录 (与 `amqsact.c file` 相同的目录) 中作为名为 `mqat.ini` 的样本提供

```
AllActivityTrace:
  ActivityInterval=0                    # Time interval between trace messages
                                        #   Values: 0-99999999 (0=off)
                                        #   Default: 0
  ActivityCount=0                       # 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=0                    # Amount of message data traced
                                        #   Values: 0-100000000
                                        #   Default: 0

ApplicationTrace:
  ApplClass=USER                         # Application type
                                        #   Values:  (USER | MCA | INTERNAL | ALL)
                                        #   Default: USER
  ApplName=AppName*                     # Application name (may be wildcarded)
                                        #   (matched to app name without path)
                                        #   Default: *
  Trace=OFF                             # Activity trace switch for application
                                        #   Values:  ( ON | OFF )
                                        #   Default: OFF
ActivityInterval=0                      # Time interval between trace messages
                                        #   Values: 0-99999999 (0=off)
                                        #   Default: 0
  ActivityCount=0                       # 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=0                    # Amount of message data traced
                                        #   Values: 0-100000000
                                        #   Default: 0
```

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 181 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 177 页的『使用 mqat.ini 配置活动跟踪行为』 和第 176 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 177 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

amqsact 样本程序

amqsact 为您格式化应用程序活动跟踪消息，并随 WebSphere MQ 一起提供。

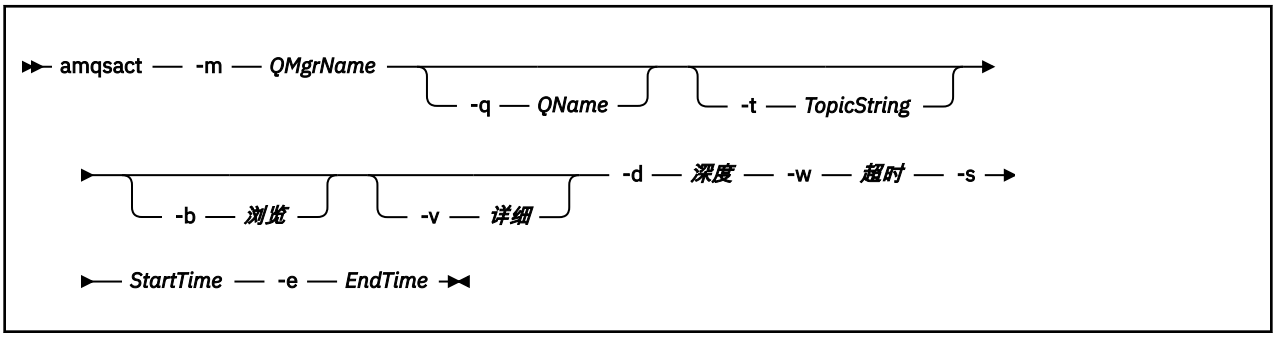
编译的程序位于样本目录中:

- 在 UNIX and Linux MQ_INSTALLATION_PATH/samp/bin 上
- 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 *Depth*
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: '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
```

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 WebSphere MQ 消息。 消息数据包含有关 IBM WebSphere MQ 应用程序执行的 MQI 操作的信息，或有关 IBM WebSphere 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

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

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

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

CommandLevel

描述: IBM WebSphere 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_OTHER MQXE_MCA MQXE_MCA_SVRCONN MQXE_COMMAND_SERVER MQXE_MQSC

标识: MQIACF_API_environment

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

Pointer Size

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。 值是其中一个 MQPL_ * 值。

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

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)。

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)

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)

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

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

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)

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

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)

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

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*[12]**

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*[12]**

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*[12]**

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*[12]**

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

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

[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。

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 *28: 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

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)

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 212 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):

[3] 仅当 ResolvedLocalQName 参数与 ResolvedQName 参数不同时，才会包含此参数。

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

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)

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 212 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

[4] 仅当 ResolvedLocalQName 参数与 ResolvedQName 参数不同时，才会包含此参数。

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

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

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)。

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

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)

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

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 WebSphere 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 WebSphere 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 246 页的表 29 汇总了可用于对通道进行实时监视的监视级别:

表 *29:* 监视级别

level	**描述**	**用途**
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 245 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。

使用队列管理器

相关任务

第 247 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 WebSphere 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。

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 WebSphere 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 WebSphere 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 WebSphere 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 245 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 QSTATUS

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. Then issue the following command, inserting in <appltag> the APPLTAG value from the output of the previous command:

```
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)
   ```

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 252 页的表 30 以确定要执行的操作:

表 *30:* 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 254 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 254 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 254 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	SSL 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。

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 249 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。
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. 如果通道已启动尚未完成的批处理 (由 CURMSGS 中的非零值指示)，那么通道可能正在等待通道的另一端确认该批处理。 查看输出中的 SUBSTATE 字段并参阅 第 253 页的表 31:

表 *31:* 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。
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

使用此页面来查看批处理可能需要很长时间才能完成的一些原因。

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。 以下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。

  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。
- 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 249 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
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 251 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。

为了避免同时切换 IBM WebSphere MQ 的大量通道的事件性，可以使用 **runswchl** 命令来切换未运行的一个或多个通道的传输队列。

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

  .
- 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。

Windows 性能监视器

在 WebSphere MQ V 7.0 和更低版本中，可以使用 Windows 性能监视器来监视 窗口 系统上本地队列的性能。 从 WebSphere MQ V 7.1 开始，此性能监视方法不再可用。

您可以使用 第 245 页的『实时监视』中描述的方法来监视所有受支持平台上的队列。

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能。 有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。 任何对 IBM 产品、程序或服务的引用并非意在明示或默示只能使用 IBM 的产品、程序或服务。 只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务都可以代替 IBM 产品、程序或服务。 但是， 评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。 提供本文档并未授予用户使用这些专利的任何许可。 您可以以书面形式将许可查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

知识产权许可
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 063-8506 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状” 提供本出版物，不附有任何种类的 (无论是明示的还是暗含的) 保证，包括但不限于暗含的有关非侵权，适销和适用于某种特定用途的保证。 某些国家或地区在某些交易中不允许免除明示或暗含的保证。 因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。 此处的信息将定期更改；这些更改将编入本资料的新版本中。 IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式 充当对那些 Web 站点的保证。 那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Corporation
软件互操作性协调员，部门 49XA
北纬 3605 号公路
罗切斯特，明尼苏达州 55901
U.S.A.

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。 因此，在其他操作环境中获得的数据可能会有明显的不同。 有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。 此外，有些测量是通过推算而估计的， 实际结果可能会有差异。 本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中 获取。 IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。 有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息包含日常商业运作所使用的数据和报表的示例。 为了尽可能全面地说明这些数据和报表，这些示例包括个人、公司、品牌和产品的名称。 所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有任何雷同，纯属巧合。

版权许可：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。 如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。 这些示例并未在所有条件下作全面测试。 因此，IBM 不能担保或默示这些程序的可靠性、可维护性或功能。

如果您正在查看本信息的软拷贝，图片和彩色图例可能无法显示。

编程接口信息

编程接口信息 (如果提供) 旨在帮助您创建用于此程序的应用软件。

本书包含有关允许客户编写程序以获取 IBM WebSphere MQ 服务的预期编程接口的信息。

但是，该信息还可能包含诊断、修改和调优信息。 提供诊断、修改和调优信息是为了帮助您调试您的应用程序软件。

要点: 请勿将此诊断，修改和调整信息用作编程接口，因为它可能会发生更改。

商标

IBM IBM 徽标 ibm.com 是 IBM Corporation 在全球许多管辖区域的商标。 当前的 IBM 商标列表可从 Web 上的 “Copyright and trademark information”www.ibm.com/legal/copytrade.shtml 获取。 其他产品和服务名称可能是 IBM 或其他公司的商标。

Microsoft 和 Windows 是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

UNIX 是 Open Group 在美国和其他国家或地区的注册商标。

Linux 是 Linus Torvalds 在美国和/或其他国家或地区的商标。

此产品包含由 Eclipse 项目 (http://www.eclipse.org/) 开发的软件。

Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其附属公司的商标或注册商标。

部件号:

(1P) P/N: