

9.0

管理 *IBM MQ*

IBM

注

在使用本资料及其支持的产品之前，请阅读第 409 页的『声明』中的信息。

本版本适用于 IBM® MQ V 9 发行版 0 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。

当您向 IBM 发送信息时，授予 IBM 以它认为适当的任何方式使用或分发信息的非独占权利，而无需对您承担任何责任。

© Copyright International Business Machines Corporation 2015 年, 2023.

内容

| | |
|-------------------------------------|----------|
| 管理 | 5 |
| 本地和远程管理..... | 8 |
| 使用控制命令管理 IBM MQ..... | 8 |
| 使用 MQSC 命令管理 MQ..... | 9 |
| 脚本 (MQSC) 命令..... | 10 |
| MQSC 命令中的 IBM MQ 对象名..... | 11 |
| 标准输入和输出..... | 11 |
| 以交互方式使用 MQSC 命令..... | 12 |
| 从文本文件运行 MQSC 命令..... | 13 |
| 从批处理文件运行 MQSC 命令..... | 16 |
| 解决 MQSC 命令的问题..... | 17 |
| 使用 PCF 命令自动执行 IBM MQ 管理..... | 18 |
| IBM MQ 可编程命令格式简介..... | 19 |
| 使用 MQAI 来简化 PCF 的使用..... | 29 |
| 使用 REST API 进行管理..... | 63 |
| 开始使用 administrative REST API..... | 63 |
| 使用 administrative REST API..... | 69 |
| 使用 REST API 进行远程管理..... | 74 |
| REST API 时间戳记..... | 77 |
| REST API 错误处理..... | 77 |
| REST API 发现..... | 80 |
| REST API 本地语言支持..... | 81 |
| 使用 IBM MQ Console 进行管理..... | 82 |
| 开始使用 IBM MQ Console..... | 83 |
| 使用本地队列管理器..... | 86 |
| 使用 IBM MQ 对象..... | 87 |
| 使用权限记录..... | 100 |
| 监视系统资源使用情况..... | 103 |
| 配置仪表板布局..... | 111 |
| 仪表板控件..... | 113 |
| 键盘快捷键..... | 114 |
| 使用 IBM MQ Explorer 进行管理..... | 114 |
| 您可以使用 IBM MQ Explorer 执行的操作..... | 114 |
| 设置 IBM MQ Explorer..... | 116 |
| 使用 IBM MQ 任务栏应用程序 (仅限 Windows)..... | 120 |
| IBM MQ 警报监视器应用程序 (仅限 Windows)..... | 121 |
| 管理本地 IBM MQ 对象..... | 121 |
| 启动和停止队列管理器..... | 121 |
| 停止 MQI 通道..... | 124 |
| 使用队列管理器..... | 125 |
| 使用本地队列..... | 127 |
| 使用别名队列..... | 132 |
| 使用死信队列..... | 133 |
| 使用模型队列..... | 149 |
| 处理管理主题..... | 150 |
| 使用预订..... | 153 |
| 使用服务..... | 156 |
| 管理用于触发的对象..... | 162 |
| 在两个系统之间使用 dmpmqmsg 实用程序..... | 164 |
| 管理远程 IBM MQ 对象..... | 167 |
| 通道和远程排队..... | 168 |
| 从本地队列管理器进行远程管理..... | 169 |

| | |
|-------------------------------------------|------------|
| 创建远程队列的本地定义..... | 174 |
| 检查是否已完成用于分布式网络的异步命令..... | 176 |
| 使用远程队列定义作为别名..... | 178 |
| 数据转换..... | 179 |
| 管理 MQ Telemetry..... | 183 |
| 在 Linux 和 AIX 上配置队列管理器以进行遥测..... | 183 |
| 在 Windows 上配置队列管理器以进行遥测..... | 185 |
| 配置分布式排队以将消息发送到 MQTT 客户机..... | 186 |
| MQTT 客户机标识、授权和认证..... | 188 |
| 使用 TLS 的遥测通道认证..... | 193 |
| 发布在遥测通道上的隐私..... | 194 |
| MQTT Java 客户机和遥测通道的 TLS 配置..... | 195 |
| 遥测通道 JAAS 配置..... | 199 |
| 管理 IBM MQ Light..... | 201 |
| 查看 MQ Light 客户机正在使用的 IBM MQ 对象..... | 201 |
| MQ Light 客户机标识、授权和认证..... | 203 |
| 在通道上发布隐私..... | 205 |
| 使用 TLS 配置 MQ Light 客户机..... | 205 |
| 将 MQ Light 客户机与队列管理器断开连接..... | 206 |
| 管理多点广播..... | 206 |
| 多点广播入门..... | 206 |
| IBM MQ 多点广播主题拓扑..... | 207 |
| 控制多点广播消息的大小..... | 208 |
| 为多点广播消息传递启用数据转换..... | 210 |
| 多点广播应用程序监视..... | 210 |
| 多点广播消息可靠性..... | 211 |
| 高级多点广播任务..... | 211 |
| 管理 IBM MQ for IBM i..... | 214 |
| 使用 CL 命令管理 IBM MQ for IBM i..... | 214 |
| 管理 IBM MQ for IBM i 的替代方法..... | 227 |
| IBM i 的工作管理..... | 231 |
| IBM i 上的可用性，备份，恢复和重新启动..... | 237 |
| 停顿 IBM MQ for IBM i..... | 273 |
| 管理 IBM MQ for z/OS..... | 277 |
| 向 IBM MQ for z/OS 发出命令..... | 277 |
| IBM MQ for z/OS 实用程序..... | 284 |
| 操作 IBM MQ for z/OS..... | 286 |
| 编写程序以管理 IBM MQ for z/OS..... | 304 |
| 管理 z/OS 上的 IBM MQ 资源..... | 315 |
| 在 z/OS 上恢复并重新启动..... | 347 |
| IBM MQ 和 IMS..... | 364 |
| 在 z/OS 上操作 Advanced Message Security..... | 376 |
| z/OS Connect 的 IBM MQ for z/OS 服务提供者..... | 377 |
| 声明..... | 409 |
| 编程接口信息..... | 410 |
| 商标..... | 410 |

管理 IBM MQ

要管理 IBM MQ 队列管理器和关联资源，请从可用于激活和管理这些资源的一组任务中选择首选方法。

您可以在本地或远程管理 IBM MQ 对象，请参阅第 8 页的『本地和远程管理』。

可使用多种不同的方法在 IBM MQ 中创建和管理队列管理器及其相关资源。这些方法包括命令行界面、图形用户界面和管理 API。

根据您使用的平台，可使用不同组的命令来管理 IBM MQ：

- 第 5 页的『IBM MQ 控制命令』
- 第 5 页的『IBM MQ 脚本 (MQSC) 命令』
- 第 6 页的『可编程命令格式 (PCF)』
-  administrative REST API
-  第 6 页的『IBM i 控制语言 (CL)』

还有以下其他选项，用于创建和管理 IBM MQ 对象：

-   第 7 页的『IBM MQ Explorer』
-  第 7 页的『IBM MQ Console』
-  第 7 页的『Windows 缺省配置应用程序』
-  第 7 页的『Microsoft 集群服务 (MSCS)』
-  有关 IBM MQ for z/OS 上的管理界面和选项的信息，请参阅第 277 页的『管理 IBM MQ for z/OS』。

您可以使用 PCF 命令，针对本地和远程队列管理器自动完成某些管理和监视任务。还可以在某些平台上使用 IBM MQ 管理界面 (MQAI) 来简化这些命令。有关自动完成管理任务的更多信息，请参阅第 18 页的『使用 PCF 命令自动执行 IBM MQ 管理』。

IBM MQ 控制命令



您可以使用控制命令在队列管理器上亲自执行管理任务。

IBM MQ for Windows, UNIX and Linux® 系统提供您在系统命令行上发出的控制命令。

在 Multiplatforms 版上创建和管理队列管理器中描述了控制命令。有关控制命令的命令参考，请参阅 [IBM MQ 控制命令](#)。

IBM MQ 脚本 (MQSC) 命令

使用 MQSC 命令来管理队列管理器对象，包括队列管理器本身、队列、进程定义、名称列表、通道、客户机连接通道、侦听器、服务和认证信息对象。

可以使用 runmqsc 命令向队列管理器发出 MQSC 命令。可通过从键盘发出命令以交互方式执行此操作，也可以重定向标准输入设备 (stdin)，从 ASCII 文本文件运行一系列命令。在这两种情况下，命令的格式都是相同的。

您可以通过三种方式运行 runmqsc 命令，具体取决于该命令上设置的标志：

- 验证方式，即在本地队列管理器上验证 MQSC 命令，但不运行这些命令
- 直接方式，即在本地队列管理器上运行 MQSC 命令
- 间接方式，即在远程队列管理器上运行 MQSC 命令

MQSC 命令在所有平台（包括 IBM i 和 z/OS）上均可用。 [比较命令集中概述了 MQSC 命令。](#)

ULW 在 UNIX, Linux, and Windows 上，可以将 MQSC 用作在系统命令行上发出的单个命令。要发出更复杂的命令或多个命令，可以将 MQSC 构建到从命令行运行的文件中。MQSC 命令可以发送到远程队列管理器。有关完整的详细信息，请参见[构建命令脚本](#)。

IBM i 要在 IBM i 服务器上发出命令，请在脚本文件中创建命令列表，然后使用 STRMQMMQSC 命令运行该文件。

注意: **IBM i**

1. 请勿将 QTEMP 库用作 STRMQMMQSC 的输入库，因为 QTEMP 库的使用受到限制。必须使用其他库作为命令的输入文件。
2. 在 IBM i 上，假脱机文件中会返回对从脚本文件发出的命令的 MQSC 响应。

第 10 页的『[脚本 \(MQSC\) 命令](#)』包含每条 MQSC 命令的描述及其语法。

有关使用 MQSC 命令的更多信息，请参见第 9 页的『[使用 MQSC 命令管理 MQ](#)』。

可编程命令格式 (PCF)

可编程命令格式 (PCF) 定义可在网络中的程序与任何队列管理器（支持 PCF）之间交换的命令和回复消息。您可以在系统管理应用程序中使用 PCF 命令来管理 IBM MQ 对象：认证信息对象、通道、通道侦听器、名称列表、进程定义、队列管理器、队列、服务以及存储类。可从网络中的单一点运行应用程序，以使用本地队列管理器与任何队列管理器（本地或远程）互通命令和回复信息。

有关 PCF 的更多信息，请参见第 19 页的『[IBM MQ 可编程命令格式简介](#)』。

有关命令和响应的 PCF 和结构的定义，请参见[可编程命令格式参考](#)。

administrative REST API

V 9.0.1

administrative REST API 提供了可用于管理 IBM MQ 的 RESTful 接口。使用 administrative REST API 时，将对表示 IBM MQ 对象的 URL 调用 HTTP 方法。例如，可在以下 URL 上使用 HTTP 方法 GET 来请求有关 IBM MQ 安装的信息：

V 9.0.4

IBM MQ 9.0.4 和更高版本：

```
https://localhost:9443/ibmmq/rest/v1/admin/installation
```

IBM MQ 9.0.3 和更低版本：

```
https://localhost:9443/ibmmq/rest/v1/installation
```

您可以将 administrative REST API 与编程语言的 HTTP/REST 实现配合使用，也可以使用诸如 cURL 之类的工具或 REST 客户机浏览器附加组件。

有关更多信息，请参见 [administrative REST API](#)

IBM i 控制语言 (CL)

IBM i

该语言可用于向 IBM MQ for IBM i 发出管理命令。可以在命令行上或通过编写 CL 程序来发出命令。这些命令会执行与 PCF 命令类似的功能，但格式不同。CL 命令专为服务器而设计，并且 CL 响应是人类易于辨识的，而 PCF 命令则与平台无关，并且命令和响应格式均适用于程序。

有关 IBM i 控制语言 (CL) 的完整详细信息，请参见 [IBM MQ for IBM i CL 命令](#)。

IBM MQ Explorer

Windows Linux

使用 IBM MQ Explorer，您可以执行以下操作：

- 定义和控制各种资源，如队列管理器、队列、进程定义、名称列表、通道、客户机连接通道、侦听器、服务和集群。
- 启动或停止本地队列管理器及其关联的进程。
- 在您的工作站上或从其他工作站查看队列管理器及其关联的对象。
- 检查队列管理器、集群和通道的状态。
- 根据队列状态，检查以确定哪些应用程序、用户或通道打开了特定队列。

在 Windows 和 Linux 系统上，可以使用系统菜单，MQExplorer 可执行文件或 **strmqcfig** 命令来启动 IBM MQ Explorer。

Linux 在 Linux 上，要成功启动 IBM MQ Explorer，您必须能够将文件写入主目录，并且主目录必须存在。

有关更多信息，请参阅第 114 页的『使用 IBM MQ Explorer 进行管理』。

您可以使用 IBM MQ Explorer 在其他平台（包括 z/OS）上管理远程队列管理器，要获取详细信息以及下载 SupportPac MSOT，请参阅 <https://www.ibm.com/support/docview.wss?uid=swg24021041>。

IBM MQ Console

V 9.0.1

您可以使用 IBM MQ Console，从 Web 浏览器管理 IBM MQ。

有关更多信息，请参阅第 82 页的『使用 IBM MQ Console 进行管理』。

Windows 缺省配置应用程序

Windows

您可以使用 Windows 缺省配置程序来创建 IBM MQ 对象的入门模板 (或缺省) 集。表 1: 由 Windows 缺省配置应用程序创建的对象中列出了所创建的缺省对象的摘要。

Microsoft 集群服务 (MSCS)

Windows

Microsoft 集群服务 (MSCS) 支持您将服务器连接到集群，从而提供更高的数据和应用程序可用性，并使您能够更轻松地管理系统。MSCS 可自动检测服务器或应用程序故障并从中恢复。

切勿将 MSCS 意义上的集群与 IBM MQ 集群相混淆。区别如下：

IBM MQ 集群

这些是一台或多台计算机上成组的队列管理器（每组有两个或更多队列管理器），提供自动互连并支持在其间共享队列以实现负载均衡和冗余。

MSCS 集群

这些是成组的计算机，它们以如下方式互连和配置：如果一台计算机发生故障，那么 MSCS 会执行故障转移，将应用程序的状态数据从发生故障的计算机转移到集群中的另一台计算机，并在新的计算机上重新启动其操作。

支持 [Microsoft 集群服务 \(MSCS\)](#) 提供有关如何配置 IBM MQ for Windows 系统以使用 MSCS 的详细信息。

相关信息

[命令集比较](#)

[IBM MQ 技术概述](#)

[规划](#)

[配置](#)

本地和远程管理

您可以在本地或远程管理 IBM MQ 对象。

本地管理

本地管理是指在本地系统上定义的任何队列管理器上执行管理任务。您可以访问其他系统，例如通过 TCP/IP 终端仿真程序 **telnet**，并在那里执行管理。在 IBM MQ 中，您可以将此视为本地管理，因为不涉及任何通道，即，通信由操作系统管理。

远程管理

IBM MQ 支持通过远程管理从单个联系点进行管理。远程管理允许您从本地系统发出命令，这些命令在另一个系统上进行处理，并且也适用于 IBM MQ Explorer。例如，可以发出远程命令来更改远程队列管理器上的队列定义。您不必登录到该系统，尽管您确实需要定义相应的通道。目标系统上的队列管理器和命令服务器必须正在运行。

某些命令不能以此方式发出，尤其是创建或启动队列管理器以及启动命令服务器。要执行此类型的任务，必须登录到远程系统并从该系统发出命令，或者创建一个可以为您发出命令的进程。此限制也适用于 IBM MQ Explorer。

第 167 页的『管理远程 IBM MQ 对象』更详细地描述了远程管理主题。

ULW 使用控制命令管理 IBM MQ

控制命令提供了在 UNIX, Linux, and Windows 上执行许多 IBM MQ 管理任务的方法。

如果要发出控制命令，那么您的用户标识必须是大多数控制命令的 `mqm` 组的成员。有关此操作的更多信息，请参阅在 [UNIX, Linux, and Windows 上管理 IBM MQ 的权限](#)。此外，请注意特定于环境的信息。您的企业所使用的一个或多个平台。

使用在队列管理器上运行的控制命令时，必须使用与您正在使用的队列管理器相关联的安装中的命令。

使用在配置为使用 `CHKLOCL` (必需) 的连接认证的队列管理器上运行的控制命令时，发现连接失败，

- 如果控制命令允许，请提供用户标识和密码。
- 使用 `MQSC` 等效的控制命令 (如果存在这些命令)。
- 使用 `-ns` 选项启动队列管理器，而需要运行无法连接的控制命令。

有关控制命令的完整列表，请参阅 [IBM MQ 控制命令](#)。

在 Windows 系统上使用控制命令

Windows

在 IBM MQ for Windows 中，在命令提示符处输入控制命令。

控制命令及其标志不区分大小写，但这些命令的自变量 (例如队列名称和队列管理器名称) 区分大小写。

例如，在命令中：

```
crtmqm /u SYSTEM.DEAD.LETTER.QUEUE jupiter.queue.manager
```

- 可以输入大写或小写的命令名，也可以两者混合使用。这些都是有效的: `crtmqm`，`CRTMQM` 和 `CRTmqm`。
- 该标志可以输入为 `-u`，`-U`，`/u` 或 `/U`。
- 必须完全按所示输入 `SYSTEM.DEAD.LETTER.QUEUE` 和 `jupiter.queue.manager`。

在 UNIX 和 Linux 系统上使用控制命令

Linux

UNIX

在 IBM MQ for UNIX 和 Linux 系统中，在 shell 窗口中输入控制命令。

在 UNIX 环境中，控制命令 (包括命令名本身，标志和任何自变量) 区分大小写。例如，在命令中：

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE jupiter.queue.manager
```

- 命令名必须是 `crtmqm`，而不是 `CRTMQM`。
- 该标志必须是 `-u`，而不是 `-U`。
- 死信队列称为 `SYSTEM.DEAD.LETTER.QUEUE`。
- 参数指定为 `jupiter.queue.manager`，这与 `JUPITER.queue.manager` 不同。

请小心输入命令，如您在示例中看到的那样。

相关信息

[IBM MQ 控制命令参考](#)

使用 MQSC 命令管理 MQ

如何使用 MQSC 命令来完成常见任务。

MQSC 命令在所有平台 (包括 IBM i 和 z/OS) 上均可用。

您可以使用 MQSC 命令来管理队列管理器对象，包括队列管理器本身，队列，进程定义，通道，客户机连接通道，侦听器，服务，名称列表，集群和认证信息对象。本节处理队列管理器，队列和进程定义；有关通道，客户机连接通道和侦听器对象的概述，请参阅 [对象](#)。有关用于管理队列管理器对象的所有 MQSC 命令的信息，请参阅第 10 页的『脚本 (MQSC) 命令』。

使用 `runmqsc` 命令向队列管理器发出 MQSC 命令。(有关此命令的详细信息，请参阅 `runmqsc`。)您可以通过交互方式执行此操作，从键盘发出命令，也可以重定向标准输入设备 (`stdin`) 以从 ASCII 文本文件运行一系列命令。在这两种情况下，命令的格式都是相同的。(有关从文本文件运行命令的信息，请参阅第 13 页的『从文本文件运行 MQSC 命令』。)

您可以通过三种方式运行 `runmqsc` 命令，具体取决于该命令上设置的标志：

- 验证命令而不运行该命令，其中 MQSC 命令在本地队列管理器上已验证，但未运行。
- 在本地队列管理器上运行命令，其中 MQSC 命令在本地队列管理器上运行。
- 在远程队列管理器上运行命令，其中 MQSC 命令在远程队列管理器上运行。

您还可以运行后跟问号的命令以显示语法。

MQSC 命令中指定的对象属性在本节中以大写形式显示 (例如，`RQMNAME`)，尽管它们不区分大小写。MQSC 命令属性名称限制为 8 个字符。

V 9.0.1 对于 Continuous Delivery，从 IBM MQ 9.0.1 开始，可以使用 `MQPROMPT` 环境变量来设置您选择的提示。除纯文本外，`MQPROMPT` 变量还允许使用 `+VARIABLE+` notation 以与 IBM MQ 服务对象定义相同的方式插入环境变量 (请参阅第 157 页的『定义服务对象』)。例如：

```
sh> export MQPROMPT="+USER+ @ +QMNAME+ @ +MQ_HOST_NAME+> "  
sh> runmqsc MY.QMGR  
5724-H72 (C) Copyright IBM Corp. 1994, 2023.  
Starting MQSC for queue manager MY.QMGR.  
username @ MY.QMGR @ aix1> DISPLAY QMSTATUS
```

V 9.0.0.1 从 IBM MQ 9.0.0 Fix Pack 1 开始，`MQPROMPT` 环境变量在 Long Term Support 发行版中也可用。

MQSC 命令在 [MQSC 命令](#) 部分中进行了详细说明。

Windows **Linux** 在 Windows 或 Linux (x86 和 x86-64 平台) 上，您还可以使用 IBM MQ Explorer 执行本节中描述的操作。有关更多信息，请参阅第 114 页的『使用 IBM MQ Explorer 进行管理』。

相关信息

[runmqsc \(运行 MQSC 命令\)](#)

[MQSC 引用](#)

脚本 (MQSC) 命令

MQSC 命令提供了在 IBM MQ 平台上发出人类可读命令的统一方法。

[MQSC 命令](#)中描述了命令的一般格式。

使用 MQSC 命令时，应遵守以下规则：

- 每个命令以主参数 (动词) 开头，后跟辅助参数 (名词)。然后，如果存在对象的名称或通用名称 (在括号中)，那么在大多数命令中都有该名称或通用名称。在此之后，参数通常可以按任何顺序出现；如果参数具有相应的值，那么该值必须直接出现在与其相关的参数之后。

注：  在 z/OS 上，辅助参数不必是秒。

- 关键字，括号和值可以用任意数目的空格和逗号分隔。语法图中显示的逗号可始终替换为一个或多个空格。每个参数前面必须至少有一个空格 (在主参数之后) z/OS 上的除外。
- 可以在命令的开头或结尾以及参数，标点和值之间出现任意数目的空白。例如，以下命令有效：


```
ALTER QLOCAL ('Account' )      TRIGDPTH ( 1)
```

一对引号内的空格很重要。


- 其他逗号可以出现在允许空格的任何位置，并被视为空格 (当然，除非它们位于用引号括起的字符串中)。
- 不允许重复参数。也不允许重复带有其 "NO" 版本的参数，如 REPLACE NOREPLACE 中的参数。
- 包含空格，小写字符或除以下字符以外的特殊字符的字符串：

- 句点 (.)
- 正斜杠 (/)
- 下划线 (_)
- 百分号 (%)

必须用单引号括起，除非它们是：


-  从 IBM MQ for z/OS 操作和控制面板发出
- 以星号结尾的通用值 (在 IBM i 上，这些值必须用单引号括起)
- 单个星号，例如 TRACE (*) (在 IBM i 上，必须将这些星号括在单引号中)
- 包含冒号的范围规范，例如 CLASS (01:03)

如果字符串本身包含单引号，那么单引号由两个单引号表示。未包含在引号内的小写字符将转换为大写。

-  在 [多平台](#)上，不包含任何字符 (即，两个单引号之间没有空格) 的字符串将被解释为用单引号括起的空格，即，以与 (") 相同的方式进行解释。此情况的例外情况是所使用的属性是下列其中一项：

- TOPICSTR
- SUB
- USERDATA
- SELECTOR

那么两个无空格的单引号将被解释为零长度字符串。

 在 z/OS 上，如果要使用单引号括起空格，那么必须将其作为 (") 输入。不包含字符 (") 的字符串与输入 () 相同。

- 在 IBM WebSphere MQ 7.0 中，基于 MQCHARV 类型 (例如，SELECTOR 和子用户数据) 的那些字符串属性中的任何尾部空格将被视为 "重要"，这意味着 "abc" 不等于 "abc"。

- 左括号后跟右括号，中间没有重要信息，例如

```
NAME ( )
```

除非特别注明，否则无效。

- 关键字不区分大小写: AltER, alter 和 ALTER 都是可接受的。未包含在引号内的任何内容都将转换为大写。
- 为某些参数定义同义词。例如，DEF 始终是 DEFINE 的同义词，因此 DEF QLOCAL 有效。但是，同义词不只是最小字符串;DEFI 不是 DEFINE 的有效同义词。

注: DELETE 参数没有同义词。这是为了避免在使用 DEF (DEFINE 的同义词) 时意外删除对象。

有关使用 MQSC 命令来管理 IBM MQ 的概述，请参阅 [第 9 页的『使用 MQSC 命令管理 MQ』](#)。

MQSC 命令使用某些特殊字符来具有特定含义。有关这些特殊字符以及如何使用这些特殊字符的更多信息，请参阅 [具有特殊含义的字符](#)。

要了解如何使用 MQSC 命令构建脚本，请参阅 [构建命令脚本](#)。

有关 MQSC 命令的完整列表，请参阅 [MQSC 命令](#)。

您可以从源 2CR 发出此命令。要获取源符号的解释，请参阅 [在 z/OS 上使用命令](#)。

相关概念

[第 19 页的『IBM MQ 可编程命令格式简介』](#)

可编程命令格式 (PCF) 定义可在网络中的程序与任何队列管理器 (支持 PCF) 之间交换的命令和回复消息。PCF 简化了队列管理器管理和其他网络管理。它们可用于解决分布式网络的复杂管理问题，尤其是随着网络规模和复杂性的增长。

相关信息

[构建命令脚本](#)

MQSC 命令中的 IBM MQ 对象名

如何在 MQSC 命令中使用对象名。

在示例中，我们对对象使用一些长名称。这是为了帮助您确定要处理的对象的类型。

发出 MQSC 命令时，只需要指定队列的局部名。在我们的示例中，我们使用队列名称，例如：

```
ORANGE.LOCAL.QUEUE
```

名称的 LOCAL.QUEUE 部分用于说明此队列是本地队列。通常，本地队列的名称不需要此参数。

我们还使用名称 saturn.queue.manager 作为队列管理器名称。名称的 queue.manager 部分用于说明此对象是队列管理器。通常，队列管理器的名称不需要此参数。

MQSC 命令中的区分大小写

MQSC 命令 (包括其属性) 可以大写或小写形式编写。MQSC 命令中的对象名将转换为大写 (即，未区分 QUEUE 和队列)，除非将这些名称括在单引号内。如果未使用引号，那么将使用大写名称来处理对象。有关更多信息，请参阅 [具有特殊含义的字符](#)。

runmqsc 命令调用与所有 IBM MQ 控制命令一样，在某些 IBM MQ 环境中区分大小写。请参阅 [第 8 页的『使用控制命令管理 IBM MQ』](#) 以获取更多信息。

标准输入和输出

标准输入设备 (也称为 stdin) 是从中获取系统输入的设备。通常，这是键盘，但您可以指定输入将来自串口或磁盘文件，例如。标准输出设备 (也称为 stdout) 是将系统输出发送到的设备。通常这是一个显示器，但是您可以将输出重定向到串口或文件。

On operating-system commands and IBM MQ control commands, the < operator redirects input. 如果此运算符后跟文件名，那么将从文件中获取输入。类似地，> 运算符重定向输出；如果此运算符后跟文件名，那么输出将定向到该文件。

以交互方式使用 MQSC 命令

您可以使用命令窗口或 shell 以交互方式使用 MQSC 命令。

要以交互方式使用 MQSC 命令，请打开命令窗口或 shell 并输入：

```
runmqsc
```

在此命令中，未指定队列管理器名称，因此 MQSC 命令由缺省队列管理器处理。如果要使用其他队列管理器，请在 `runmqsc` 命令上指定队列管理器名称。例如，要在队列管理器 `jupiter.queue.manager` 上运行 MQSC 命令，请使用以下命令：

```
runmqsc jupiter.queue.manager
```

在此之后，您输入的所有 MQSC 命令都将由此队列管理器处理，假定它位于同一节点上并且已在运行。

现在，您可以根据需要输入任何 MQSC 命令。例如，尝试此操作：

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
```

对于具有过多参数以适合一行的命令，请使用连续字符来指示命令在以下行上继续：

- 减号 (-) 指示命令将从下一行开始继续。
- 加号 (+) 指示命令将从下一行上的第一个非空白字符继续。

命令输入以非连续字符的非空白行的最终字符终止。您还可以通过输入分号 (;) 来显式终止命令输入。(如果在命令输入的最后一行末尾意外地输入了连续字符，那么这特别有用。)

来自 MQSC 命令的反馈

当您发出 MQSC 命令时，队列管理器会返回操作员消息，以确认您的操作或告知您已发生的错误。例如：

```
AMQ8006: IBM MQ queue created.
```

此消息确认已创建队列。

```
AMQ8405: Syntax error detected at or near end of command segment below:-
```

```
AMQ8426: Valid MQSC commands are:
```

```
ALTER
CLEAR
DEFINE
DELETE
DISPLAY
END
PING
REFRESH
RESET
RESOLVE
RESUME
START
STOP
SUSPEND
4 : end
```

此消息指示您发生了语法错误。

这些消息将发送到标准输出设备。如果未正确输入命令，请参阅 [MQSC 命令](#) 以获取正确的语法。

结束 MQSC 命令的交互式输入

要停止使用 MQSC 命令，请输入 END 命令。

或者，可以将 EOF 字符用于您的操作系统。

相关概念

第 13 页的『从文本文件运行 MQSC 命令』

以交互方式运行 MQSC 命令适用于快速测试，但如果您有很长的命令，或者重复使用特定命令序列，请考虑从文本文件重定向 stdin。

相关信息

[runmqsc](#)

UNIX 和 Linux 上的 runmqsc 命令

UNIX 和 Linux 上的 **runmqsc** 命令行支持命令重新调用，命令完成和 Emacs 命令键。

以下命令行编辑器功能可用：

- 使用向上箭头键和向下箭头键重新调用先前输入的命令
- 使用制表键和空格键自动完成命令的下一个关键字
- Emacs 命令键或类似的命令键功能

要使用这些功能，必须安装 **curses** 库。如果 **curses** 库未安装在您的系统上，那么 **runmqsc** 不具有命令行编辑器功能，并且在启动 **runmqsc** 命令行时将显示一条消息。要安装的 **curses** 库的名称取决于 UNIX 平台：

- **AIX** 在 AIX 上，安装 **curses**
- **HP-UX** 在 HP-UX 上，安装 **Xcurses**
- 在所有其他 UNIX 平台和 Linux 上，安装 **ncurses**

定制 Emacs 键绑定

您可以定制绑定到命令的密钥。例如，可以将键绑定到 vi 绑定，而不是缺省 Emacs 键绑定。

通过编辑存储在主目录中的 **.editrc** 文件来定制密钥。有关更多信息，请参阅 FreeBSD 联机帮助页中的 [editrc](#)。

禁用命令重新调用，命令完成和 Emacs 命令键

您可以通过设置环境变量来禁用命令重新调用，命令完成和 Emacs 命令键。将环境变量 **MQ_OVERRIDE_LIBEDIT_LOAD** 设置为 **TRUE**。

当 **runmqsc** 显示以下参考消息时，此环境变量可用作变通方法：

```
AMQ8521I: Command completion and history unavailable
```

从文本文件运行 MQSC 命令

以交互方式运行 MQSC 命令适用于快速测试，但如果您有很长的命令，或者重复使用特定命令序列，请考虑从文本文件重定向 stdin。

要从文本文件重定向 stdin，请首先使用常规文本编辑器创建包含 MQSC 命令的文本文件，然后运行 **runmqsc** 命令。

注：如果通过从文本文件重定向 stdin 以客户机方式运行 **runmqsc** 命令，那么 IBM MQ 期望输入文件的第一行是密码。

使用 **runmqsc** 命令时，请使用重定向运算符。例如，以下命令运行文本文件 `myprog.in` 中包含的一系列命令：

```
runmqsc < myprog.in
```

同样，您也可以将输出重定向到文件。包含用于输入的 MQSC 命令的文件称为 MQSC 命令文件。包含来自队列管理器的应答的输出文件称为输出文件。

要在 **runmqsc** 命令上重定向 `stdin` 和 `stdout`，请使用以下格式的命令：

```
runmqsc < myprog.in > myprog.out
```

此命令调用 MQSC 命令文件中包含的 MQSC 命令 `myprog.in`。由于未指定队列管理器名称，因此 MQSC 命令将针对缺省队列管理器运行。输出将发送到文本文件 `myprog.out`。第 14 页的图 1 显示了从 MQSC 命令文件 `myprog.in` 中抽取的内容，而第 15 页的图 2 显示了 `myprog.out` 中输出的相应抽取。

要在 **runmqsc** 命令上重定向 `stdin` 和 `stdout`，对于非缺省队列管理器 (`saturn.queue.manager`)，请使用以下格式的命令：

```
runmqsc saturn.queue.manager < myprog.in > myprog.out
```

MQSC 命令文件

MQSC 命令以人类可读格式 (即 ASCII 文本) 编写。第 14 页的图 1 是从 MQSC 命令文件中抽取的，其中显示了 MQSC 命令 (**DEFINE QLOCAL**) 及其属性。MQSC 命令包含每个 MQSC 命令及其语法的描述。

```
.  
. .  
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE +  
DESCR(' ') +  
PUT(ENABLED) +  
DEFPRTY(0) +  
DEFPSIST(NO) +  
GET(ENABLED) +  
MAXDEPTH(5000) +  
MAXMSGL(1024) +  
DEFSOPT(SHARED) +  
NOHARDENBO +  
USAGE(NORMAL) +  
NOTRIGGER;  
. .  
.
```

图 1: 从 MQSC 命令文件中抽取

为了在 IBM MQ 环境中实现可移植性，请将 MQSC 命令文件中的行长度限制为 72 个字符。加号指示命令在下一行继续执行。

MQSC 命令报告

runmqsc 命令返回一个报告，该报告将发送到 `stdout`。报告包含：

- 用于将 MQSC 命令标识为报告源的头：

```
Starting MQSC for queue manager jupiter.queue.manager.
```

其中，`jupiter.queue.manager` 是队列管理器的名称。

- 发出的 MQSC 命令的可选编号列表。缺省情况下，输入的文本将回传到输出。在此输出中，每个命令都以序号作为前缀，如第 15 页的图 2 中所示。但是，可以在 `runmqsc` 命令上使用 `-e` 标志来禁止输出。
- 任何发现出错的命令的语法错误消息。
- 指示运行每个命令的结果的操作员消息。例如，成功完成 `DEFINE QLOCAL` 命令的操作程序消息为：

```
AMQ8006: IBM MQ queue created.
```

- 运行脚本文件时由于一般错误而产生的其他消息。
- 报告的简要统计摘要，指示读取的命令数，存在语法错误的命令数以及无法处理的命令数。

注：队列管理器尝试仅处理那些没有语法错误的命令。

```
Starting MQSC for queue manager jupiter.queue.manager.
.
.
12:  DEFINE QLOCAL('ORANGE.LOCAL.QUEUE') REPLACE +
:     DESCR(' ') +
:     PUT(ENABLED) +
:     DEFPRTY(0) +
:     DEFPSIST(NO) +
:     GET(ENABLED) +
:     MAXDEPTH(5000) +
:     MAXMSGL(1024) +
:     DEFSOPT(SHARED) +
:     NOHARDENBO +
:     USAGE(NORMAL) +
:     NOTRIGGER;
AMQ8006: IBM MQ queue created.
.
.
.
```

图 2: 从 MQSC 命令报告文件中抽取

运行提供的 MQSC 命令文件

IBM MQ 随附了以下 MQSC 命令文件：

amqscos0.tst

样本程序使用的对象的定义。

amqscic0.tst

CICS 事务的队列定义。

Windows 在 IBM MQ for Windows 中，这些文件位于目录 `MQ_INSTALLATION_PATH\tools\mqsc\samples` 中。`MQ_INSTALLATION_PATH` 表示 IBM MQ 安装所在的高级目录。

Linux **UNIX** 在 UNIX and Linux 系统上，这些文件位于目录 `MQ_INSTALLATION_PATH/samp` 中。`MQ_INSTALLATION_PATH` 表示 IBM MQ 安装所在的高级目录。

运行这些命令的命令为：

```
runmqsc < amqscos0.tst >test.out
```

使用 runmqsc 验证命令

您可以使用 **runmqsc** 命令来验证本地队列管理器上的 MQSC 命令，而不必实际运行这些命令。为此，请在 **runmqsc** 命令中设置 **-v** 标志，例如：

```
runmqsc -v < myprog.in > myprog.out
```

针对 MQSC 命令文件调用 **runmqsc** 时，队列管理器会验证每个命令并返回报告，而不会实际运行 MQSC 命令。这允许您检查命令文件中命令的语法。如果您执行以下操作，那么这一点尤为重要：

- 从命令文件运行大量命令。
- 多次使用 MQSC 命令文件。

返回的报告与 [第 15 页的图 2](#) 中显示的报告类似。

不能使用此方法远程验证 MQSC 命令。例如，如果尝试此命令：

```
runmqsc -w 30 -v jupiter.queue.manager < myprog.in > myprog.out
```

将忽略用于指示队列管理器是远程的 **-w** 标志，并以验证方式在本地运行该命令。30 是 IBM MQ 等待来自远程队列管理器的应答的秒数。

相关概念

[第 11 页的『标准输入和输出』](#)

标准输入设备(也称为 **stdin**) 是从中获取系统输入的设备。通常，这是键盘，但您可以指定输入将来自串口或磁盘文件，例如。标准输出设备(也称为 **stdout**) 是将系统输出发送到的设备。通常这是一个显示器，但是您可以将输出重定向到串口或文件。

[第 12 页的『以交互方式使用 MQSC 命令』](#)

您可以使用命令窗口或 shell 以交互方式使用 MQSC 命令。

相关信息

[runmqsc](#)

从批处理文件运行 MQSC 命令

如果您有很长的命令，或者正在重复使用特定的命令序列，请考虑从批处理文件重定向 **stdin**。

要从批处理文件重定向 **stdin**，请首先使用常规文本编辑器创建包含 MQSC 命令的批处理文件。使用 **runmqsc** 命令时，请使用重定向运算符。以下示例：

1. 创建测试队列管理器 TESTQM
2. 创建匹配的 CLNTCONN 和侦听器集以使用 TCP/IP 端口 1600
3. 创建测试队列 TESTQ
4. 使用 amqsputc 样本程序将消息放入队列


```

export MYTEMPQM=TESTQM
export MYPOR=1600
export MQCHLLIB=/var/mqm/qmgrs/$MQTEMPQM/@ipcc

crtmqm $MYTEMPQM
stimqm $MYTEMPQM
runmqtsr -m $MYTEMPQM -t TCP -p $MYPOR &

runmqsc $MYTEMPQM << EOF
DEFINE CHANNEL(NTLM) CHLTYPE(SVRCONN) TRPTYPE(TCP)
DEFINE CHANNEL(NTLM) CHLTYPE(CLNTCONN) QMNAME('$MYTEMPQM') CONNAME('hostname($MYPOR)')
ALTER CHANNEL(NTLM) CHLTYPE(CLNTCONN)
DEFINE QLOCAL(TESTQ)
EOF

amqspucl TESTQ $MYTEMPQM << EOF
hello world
EOF

endmqm -i $MYTEMPQM

```

图 3: 用于从批处理文件运行 MQSC 命令的示例脚本

解决 MQSC 命令的问题

如果您无法运行 MQSC 命令，请使用本主题中的信息，了解这些常见问题中的任何问题是否适用于您。当您阅读命令生成的错误时，产生的问题并不总是很明显。

使用 runmqsc 命令时，请记住以下内容：

- 使用 < 运算符可从文件重定向输入。如果省略此运算符，那么队列管理器会将文件名解释为队列管理器名称，并发出以下错误消息：

```
AMQ8118E: IBM MQ queue manager does not exist.
```

- 如果将输出重定向到文件，请使用 > 重定向运算符。缺省情况下，在调用 runmqsc 时，将文件放入当前工作目录中。指定标准文件名，以将输出发送至特定文件和目录。
- 通过使用以下命令显示所有队列管理器，检查您是否已创建要运行这些命令的队列管理器：

```
dspmqs
```

- 该队列管理器必须正在运行。如果不是，请将其启动；(请参阅 [启动队列管理器](#))。如果尝试启动已在运行的队列管理器，那么会收到错误消息。
- 如果尚未定义缺省队列管理器，或者收到以下错误，请在 runmqsc 命令上指定队列管理器名称：

```
AMQ8146E: IBM MQ queue manager not available.
```

- 不能将 MQSC 命令指定为 runmqsc 命令的参数。例如，以下内容无效：

```
runmqsc DEFINE QLOCAL(FRED)
```

- 不能在发出 runmqsc 命令之前输入 MQSC 命令。
- 无法从 runmqsc 运行控制命令。例如，当您以交互方式运行 MQSC 命令时，不能发出 stimqm 命令来启动队列管理器。如果执行此操作，那么将收到类似于以下内容的错误消息：

```
runmqsc
.
.
Starting MQSC for queue manager jupiter.queue.manager.
1 : stimqm saturn.queue.manager
```

```
AMQ8405: Syntax error detected at or near end of cmd segment below:-s

AMQ8426: Valid MQSC commands are:
ALTER
CLEAR
DEFINE
DELETE
DISPLAY
END
PING
REFRESH
RESET
RESOLVE
RESUME
START
STOP
SUSPEND
2 : end
```

使用 PCF 命令自动执行 IBM MQ 管理

您可能决定自动执行某些管理和监视任务对您的安装是有益的。您可以使用可编程命令格式 (PCF) 命令自动执行本地和远程队列管理器的管理任务。本部分假定您具有管理 IBM MQ 对象的经验。

PCF 命令

IBM MQ 可编程命令格式 (PCF) 命令可用于将管理任务编程到管理程序中。这样，从程序中，您可以处理队列管理器对象 (队列，进程定义，名称列表，通道，客户机连接通道，侦听器，服务和认证信息对象)，甚至可以处理队列管理器本身。

PCF 命令涵盖 MQSC 命令提供的相同功能范围。您可以编写程序以从单个节点向网络中的任何队列管理器发出 PCF 命令。这样，您既可以集中管理任务，也可以自动执行管理任务。

每个 PCF 命令都是嵌入在 IBM MQ 消息的应用程序数据部分中的数据结构。使用 MQI 函数 MQPUT 以与任何其他消息相同的方式将每个命令发送到目标队列管理器。如果命令服务器正在接收消息的队列管理器上运行，那么命令服务器会将其解释为命令消息并运行该命令。要获取应答，应用程序会发出 MQGET 调用，并在另一数据结构中返回应答数据。然后，应用程序可以处理应答并相应地执行操作。

注: 与 MQSC 命令不同，PCF 命令及其回复不是您可以读取的文本格式。

简而言之，以下是创建 PCF 命令消息所需的部分内容：

消息描述符

这是标准 IBM MQ 消息描述符，其中：

- 消息类型 (*MsgType*) 为 MQMT_REQUEST。
- 消息格式 (*Format*) 为 MQFMT_ADMIN。

应用程序数据

包含包含 PCF 头的 PCF 消息，其中：

- PCF 消息类型 (*Type*) 指定 MQCFT_COMMAND。
- 命令标识指定命令，例如 *Change Queue* (MQCMD_CHANGE_Q)。

有关 PCF 数据结构及其实现方式的完整描述，请参阅第 19 页的『IBM MQ 可编程命令格式简介』。

PCF 对象属性

PCF 中的对象属性不限于 8 个字符，因为它们适用于 MQSC 命令。它们以斜体显示在本指南中。例如，RQMNAME 的 PCF 等效项为 *RemoteQMGrName*。






对 PCF 进行转义

转义 PCF 是在消息文本中包含 MQSC 命令的 PCF 命令。您可以使用 PCF 将命令发送到远程队列管理器。有关对 PCF 进行转义的更多信息，请参阅 [转义](#)。

IBM MQ 可编程命令格式简介

可编程命令格式 (PCF) 定义可在网络中的程序与任何队列管理器 (支持 PCF) 之间交换的命令和回复消息。PCF 简化了队列管理器管理和其他网络管理。它们可用于解决分布式网络的复杂管理问题, 尤其是随着网络规模和复杂性的增长。

可编程命令格式受以下支持:

-  IBM MQ for AIX
-  IBM MQ for HP-UX
-  IBM MQ for IBM i
-  IBM MQ for Linux
-  IBM MQ for Solaris
-  IBM MQ for Windows
-  IBM MQ for z/OS

问题 PCF 命令解决

分布式网络的管理可能会变得复杂。随着网络规模和复杂性的增加, 管理问题继续增加。

特定于消息传递和排队的管理示例包括:

- 资源管理。
例如, 队列创建和删除。
- 性能监视。
例如, 最大队列深度或消息速率。
- 控件。
例如, 调整队列参数, 例如, 最大队列深度, 最大消息长度以及启用和禁用队列。
- 消息路由。
定义通过网络的备用路由。

IBM MQ PCF 命令可用于简化队列管理器管理和其他网络管理。PCF 命令允许您使用单个应用程序从网络中的单个队列管理器执行网络管理。

什么是 PCF?

PCF 定义可在程序与网络中的任何队列管理器 (支持 PCF) 之间交换的命令和应答消息。您可以在系统管理应用程序中使用 PCF 命令来管理 IBM MQ 对象: 认证信息对象、通道、通道侦听器、名称列表、进程定义、队列管理器、队列、服务以及存储类。可从网络中的单一点运行应用程序, 以使用本地队列管理器与任何队列管理器 (本地或远程) 互通命令和回复信息。

每个队列管理器都有一个具有标准队列名称的管理队列, 应用程序可以将 PCF 命令消息发送到该队列。每个队列管理器还具有一个命令服务器, 用于处理来自管理队列的命令消息。因此, 网络中的任何队列管理器都可以处理 PCF 命令消息, 并且可以使用指定的应答队列将应答数据返回到应用程序。PCF 命令和应答消息是使用正常消息队列接口 (MQI) 发送和接收的。


有关可用 PCF 命令 (包括其参数) 的列表, 请参阅 [可编程命令格式的定义](#)。

使用 IBM MQ 可编程命令格式

您可以在系统管理程序中使用 PCF 进行 IBM MQ 远程管理。

本节包括:

- [第 20 页的『PCF 命令消息』](#)

- [第 22 页的『IBM MQ 中的 PCF 响应』](#)
-  [第 24 页的『扩展响应』](#)
- 用于命名 IBM MQ 对象的规则
- [第 25 页的『IBM MQ 中 PCF 命令的权限检查』](#)

PCF 命令消息

PCF 命令消息由 PCF 头，该头中标识的参数以及用户定义的消息数据组成。使用 "消息队列" 接口调用来发出消息。

每个命令及其参数都作为单独的命令消息发送，该命令消息中包含后跟多个参数结构的 PCF 头；有关 PCF 头的详细信息，请参阅 [MQCFH-PCF 头](#)，有关参数结构的示例，请参阅 [MQCFST-PCF 字符串参数](#)。PCF 头标识命令以及在同一消息中跟随的参数结构数。每个参数结构都为命令提供一个参数。

由命令服务器生成的对命令的应答具有类似的结构。有一个 PCF 头，后跟一些参数结构。应答可以由多条消息组成，但命令始终仅由一条消息组成。

 在 [多平台](#)上，将 PCF 命令发送到的队列始终称为 SYSTEM.ADMIN.COMMAND.QUEUE。

 在 z/OS 上，将命令发送到 SYSTEM.COMMAND.INPUT，但为 SYSTEM.ADMIN.COMMAND.QUEUE 可以是其别名。服务此队列的命令服务器将应答发送到由命令消息的消息描述符中的 *ReplyToQ* 和 *ReplyToQMGr* 字段定义的队列。

如何发出 PCF 命令消息

使用常规消息队列接口 (MQI) 调用，MQPUT 和 MQGET 等，将 PCF 命令和响应消息放入其队列以及从其队列中检索这些消息。

注：

确保命令服务器正在目标队列管理器上运行，以便 PCF 命令在该队列管理器上进行处理。

有关提供的头文件的列表，请参阅 [IBM MQ COPY](#)，头，包含和模块文件。

PCF 命令的消息描述符

IBM MQ 消息描述符完整记录在 [MQMD-消息描述符](#)中。

PCF 命令消息在消息描述符中包含以下字段：

报告

任何有效值 (根据需要)。

MsgType

此字段必须是 MQMT_REQUEST，以指示需要响应的消息。

到期

任何有效值 (根据需要)。

Feedback

设置为 MQFB_NONE





编码

如果要发送到下列其中一个系统，请将此字段设置为用于消息数据的编码；必要时将执行转换：

-  IBM i
-  Linux
-  UNIX
-  Windows

CodedCharSetId

如果要发送到下列其中一个系统，请将此字段设置为用于消息数据的编码字符集标识；必要时将执行转换：

-  IBM i
-  Linux
-  UNIX
-  Windows

格式

设置为 MQFMT_ADMIN。

优先级

任何有效值 (根据需要)。

持久

任何有效值 (根据需要)。

MsgId

发送应用程序可以指定任何值，或者可以指定 MQMI_NONE 以请求队列管理器生成唯一消息标识。

CorrelId

发送应用程序可以指定任何值，或者可以指定 MQCI_NONE 以指示无相关标识。

ReplyToQ

用于接收响应的队列的名称。

ReplyToQMgr

响应的队列管理器的名称 (或空白)。

消息上下文字段

可以根据需要将这些字段设置为任何有效值。通常，Put 消息选项 MQPMO_DEFAULT_CONTEXT 用于将消息上下文字段设置为缺省值。

如果您正在使用 version-2 MQMD 结构，那么必须设置以下其他字段：

GroupId

设置为 MQGI_NONE

MsgSeqNumber

设置为 1

偏移量

设置为 0

MsgFlags

设置为 MQMF_NONE

OriginalLength

设置为 MQOL_UNDEFINED

发送用户数据

PCF 结构还可用于发送用户定义的消息数据。在这种情况下，消息描述符 *Format* 字段必须设置为 MQFMT_PCF。

在指定队列中发送和接收 PCF 消息

将 PCF 消息发送到指定队列

要将消息发送到指定队列，mqPutBag 调用会将指定包的内容转换为 PCF 消息，并将消息发送到指定队列。通话后包内内容不变。

作为此调用的输入，必须提供：

- MQI 连接句柄。
- 要放置消息的队列的对象句柄。
- 消息描述符。有关消息描述符的更多信息，请参阅 [MQMD-消息描述符](#)。
- 使用 MQPMO 结构放置消息选项。有关 MQPMO 结构的更多信息，请参阅 [MQPMO-Put-message 选项](#)。
- 要转换为消息的包的句柄。

注: 如果包中包含管理消息，并且 mqAdd 查询调用用于将值插入到包中，那么 MQIASY_COMMAND 数据项的值必须是 MQAI 可识别的 INQUIRE 命令。

有关 mqPutBag 调用的完整描述，请参阅 [mqPutBag](#)。

从指定队列接收 PCF 消息

要从指定队列接收消息，mqGetBag 调用从指定队列获取 PCF 消息并将消息数据转换为数据包。

作为此调用的输入，必须提供：

- MQI 连接句柄。
- 要从中读取消息的队列的对象句柄。
- 消息描述符。在 MQMD 结构中，**Format** 参数必须是 MQFMT_ADMIN，MQFMT_EVENT 或 MQFMT_PCF。

注: 如果在工作单元中接收到消息 (即，使用 MQGMO_SYNCPOINT 选项)，并且消息具有不受支持的格式，那么可以回退工作单元。然后在队列上恢复该消息，并且可以使用 MQGET 调用 (而不是 mqGetBag 调用) 来检索该消息。有关消息描述符的更多信息，请参阅 [MQGMO-Get-message 选项](#)。

- 使用 MQGMO 结构获取消息选项。有关 MQGMO 结构的更多信息，请参阅 [MQMD-消息描述符](#)。
- 用于包含已转换消息的包的句柄。

有关 mqGetBag 调用的完整描述，请参阅 [mqGetBag](#)。

IBM MQ 中的 PCF 响应

作为对每个命令的响应，命令服务器会生成一条或多条响应消息。响应消息具有与命令消息相似的格式。

PCF 头与它作为响应的命令具有相同的命令标识值 (请参阅 [MQCFH-PCF 头](#) 以获取详细信息)。根据请求的报告选项设置消息标识和相关标识。

如果命令消息的 PCF 头类型为 MQCFT_COMMAND，那么仅生成标准响应。此类命令在除 z/OS 以外的所有平台上都受支持。较旧的应用程序不支持 z/OS 上的 PCF；IBM MQ Explorer on Windows 是一个此类应用程序 (但是，IBM WebSphere MQ 6.0 或更高版本 IBM MQ Explorer 在 z/OS 上支持 PCF)。

如果命令消息的 PCF 头类型为 MQCFT_COMMAND_XR，那么将生成扩展响应或标准响应。此类命令在 z/OS 和其他一些平台上受支持。在 z/OS 上发出的命令仅生成扩展响应。在其他平台上，可能会生成任一类型的响应。

如果单个命令指定通用对象名，那么将在其自己的消息中针对每个匹配对象返回单独的响应。对于响应生成，具有通用名称的单个命令将被视为多个单独的命令 (控制字段 MQCFC_LAST 或 MQCFC_NOT_LAST 除外)。否则，一条命令消息将生成一条响应消息。

某些 PCF 响应可能会返回结构，即使未请求该结构也是如此。此结构显示在响应的定义中 ([可编程命令格式的定义](#)) 始终返回。对于这些响应，需要对响应中的对象进行命名以标识应用数据的对象的原因。

响应的消息描述符

响应消息在消息描述符中具有以下字段：

MsgType

此字段为 MQMT_REPLY。

MsgId

此字段由队列管理器生成。

CorrelId

根据命令消息的报告选项生成此字段。

格式

此字段为 MQFMT_ADMIN。

编码

设置为 MQENC_NATIVE。

CodedCharSetId

设置为 MQCCSI_Q_MGR。

持久

与命令消息中的相同。

优先级

与命令消息中的相同。

将使用 MQPMO_PASS_IDENTITY_CONTEXT 生成响应。

标准响应

将生成头类型为 MQCFT_COMMAND 的命令消息，生成标准响应。此类命令在除 z/OS 以外的所有平台上都受支持。

有三种类型的标准响应：

- 确定响应
- 错误响应
- 数据响应

确定响应

此响应由以命令格式头开头的消息组成，*CompCode* 字段为 MQCC_OK 或 MQCC_WARNING。

对于 MQCC_OK，*Reason* 为 MQRC_NONE。

对于 MQCC_WARNING，*Reason* 标识警告的性质。在这种情况下，命令格式头后面可能跟有一个或多个适合于此原因码的警告参数结构。

在任一情况下，对于查询命令，可遵循以下部分中描述的进一步参数结构。

错误响应

如果该命令有错误，那么将发送一条或多条错误响应消息（即使对于通常只有一条响应消息的命令，也可能发送多条错误响应消息）。这些错误响应消息根据需要设置了 MQCFC_LAST 或 MQCFC_NOT_LAST。

每条此类消息都以响应格式头开头，*CompCode* 值为 MQCC_FAILED，*Reason* 字段用于标识特定错误。通常，每条消息都描述了不同的错误。此外，每条消息在头后面都有零个或一个（从不超过一个）错误参数结构。此参数结构（如果有）是 MQCFIN 结构，其 *Parameter* 字段包含下列其中一项：

- MQIACF_PARAMETER_ID

结构中的 *Value* 字段是出错的参数的参数标识（例如 MQCA_Q_NAME）。

- MQIACF_ERROR_ID

此值与 *Reason* 值（在命令格式头中）MQRC_UNEXPECTED_ERROR 配合使用。MQCFIN 结构中的 *Value* 字段是命令服务器接收到的意外原因码。

- MQIACF_SELECTOR

如果随命令发送的列表结构（MQCFIL）包含重复的选择器或无效的选择器，那么会发生此值。命令格式头中的 *Reason* 字段标识错误，MQCFIN 结构中的 *Value* 字段是出错的命令的 MQCFIL 结构中的参数值。

- MQIACF_ERROR_OFFSET

当 Ping 通道命令中存在数据比较错误时，会发生此值。结构中的 *Value* 字段是 Ping 通道比较错误的偏移量。

- MQIA_CODED_CHAR_SET_ID

当局 PCF 命令消息的消息描述符中的编码字符集标识与目标队列管理器的编码字符集标识不匹配时，将发生此值。结构中的 *Value* 字段是队列管理器的编码字符集标识。

最后 (或唯一) 错误响应消息是摘要响应，其 *CompCode* 字段为 MQCC_FAILED，*Reason* 字段为 MQRCCF_COMMAND_FAILED。此消息的头后面没有参数结构。

数据响应

此响应包含对查询命令的 OK 响应 (如前所述)。“确定”响应后跟包含请求的数据的其他结构，如 [可编程命令格式的定义](#) 中所述。

应用程序不得依赖于以任何特定顺序返回的这些附加参数结构。

扩展响应

在 z/OS 上发出的命令会生成扩展响应。

有三种类型的扩展响应：

- 消息响应，类型为 MQCFT_XR_MSG
- 项响应，类型为 MQCFT_XR_ITEM
- 摘要响应，类型为 MQCFT_XR_SUMMARY

每个命令都可以生成一组或多组响应。每组响应由一条或多条消息组成，从 PCF 头的 *MsgSeqNumber* 字段中的 1 按顺序编号。每个集合中最后一个 (或唯一) 响应的 *Control* 字段具有值 MQCFC_LAST。对于集合中的所有其他响应，此值为 MQCFC_NOT_LAST。

任何响应都可以包含一个或多个可选 MQCFBS 结构，其中 *Parameter* 字段设置为 MQBACF_RESPONSE_SET，值为响应集标识。标识是唯一的，标识包含响应的响应集。对于每一组响应，都有一个 MQCFBS 结构来标识它。

扩展响应具有至少两个参数结构：

- *Parameter* 字段设置为 MQBACF_RESPONSE_ID 的 MQCFBS 结构。此字段中的值是响应所属的响应集的标识。第一个集合中的标识是任意的。在后续集合中，标识是先前在 MQBACF_RESPONSE_SET 结构中通知的标识。
- 将 *Parameter* 字段设置为 MQCACF_RESPONSE_Q_MGR_NAME 的 MQCFST 结构，值是响应集来自的队列管理器的名称。

许多响应都具有其他参数结构，以下部分中描述了这些结构。

在找到具有 MQCFC_LAST 的响应之前，您无法预先确定除了获取响应之外的集合中有多少响应。您也无法提前确定有多少组响应，因为任何集合都可能包含 MQBACF_RESPONSE_SET 结构，以指示生成了其他集合。

对 "查询" 命令的扩展响应

查询命令通常为找到的每个与指定搜索条件匹配的项生成项响应 (类型 MQCFT_XR_ITEM)。项响应在头中具有值为 MQCC_OK 的 *CompCode* 字段和值为 MQRC_NONE 的 *Reason* 字段。它还包括描述该项及其请求的属性的其他参数结构，如 [可编程命令格式的定义](#) 中所述。

如果项出错，那么头中的 *CompCode* 字段的值为 MQCC_FAILED，而 *Reason* 字段标识特定错误。包含其他参数结构以标识项。

某些查询命令除了返回项响应外，还可能返回常规 (不特定于名称) 消息响应。这些响应是 MQCFT_XR_MSG 类型的参考响应或错误响应。

如果 Inquire 命令成功，那么可能存在 (可选) 摘要响应 (类型为 MQCFT_XR_SUMMARY)，*CompCode* 值为 MQCC_OK，*Reason* 字段值为 MQRC_NONE。

如果 "查询" 命令失败，那么可能会返回项响应，并且可能存在摘要响应 (类型为 MQCFT_XR_SUMMARY)，*CompCode* 值为 MQCC_FAILED，*Reason* 字段值为 MQRCCF_COMMAND_FAILED。

对除 "查询" 以外的命令的扩展响应

成功的命令会生成消息响应，其中头中的 *CompCode* 字段的值为 MQCC_OK，而 *Reason* 字段的值为 MQRC_NONE。始终至少有一条消息；它可能是参考消息 (MQCFT_XR_MSG) 或摘要消息 (MQCFT_XR_SUMMARY)。(可选) 可能有其他参考消息 (类型 MQCFT_XR_MSG)。每条参考消息可能包含许多其他参数结构以及有关该命令的信息；请参阅个别命令描述以了解可能发生的结构。

生成错误消息响应失败的命令 (类型为 MQCFT_XR_MSG)，其中头中的 *CompCode* 字段的值为 MQCC_FAILED，*Reason* 字段标识特定错误。每条消息都可能包含一些其他参数结构以及有关错误的信息；请参阅可能发生的结构的个别错误描述。可能会生成参考消息响应。可以选择是摘要响应 (MQCFT_XR_SUMMARY)，*CompCode* 值为 MQCC_FAILED，*Reason* 字段值为 MQRCCF_COMMAND_FAILED。

使用 CommandScope 对命令的扩展响应

如果命令使用 **CommandScope** 参数，或者导致生成使用 **CommandScope** 参数的命令，那么将从接收该命令的队列管理器设置初始响应。然后，将为命令所定向至的每个队列管理器生成单独的一组或多组响应 (就像发出了多个单独的命令一样)。最后，来自接收队列管理器的响应集包含整体摘要响应 (类型 MQCFT_XR_SUMMARY)。MQCACF_RESPONSE_Q_MGR_NAME 参数结构标识生成每个集合的队列管理器。


初始响应集具有以下附加参数结构：

- MQIACF_COMMAND_INFO (MQCFIN)。此结构中的可能值为 MQCMDI_CMDSCOPE_ACCEPTED 或 MQCMDI_CMDSCOPE_GENERATED。
- MQIACF_CMDSCOPE_Q_MGR_COUNT (MQCFIN)。此结构指示将命令发送到的队列管理器数。

IBM MQ 中 PCF 命令的权限检查

处理 PCF 命令时，命令消息中的消息描述符中的 *UserIdentifier* 将用于必需的 IBM MQ 对象权限检查。在每个平台上以不同方式实施权限检查，如本主题中所述。

将在正在处理该命令的系统上执行这些检查；因此，此用户标识必须存在于目标系统上，并且具有处理该命令所需的权限。如果消息来自远程系统，那么实现目标系统上存在的标识的一种方法是在本地和远程系统上都具有匹配的用户标识。

注：  有关 z/OS 上的权限检查的信息，请参阅 [任务 1: 标识 z/OS 系统参数](#)。

IBM MQ for IBM i



要处理任何 PCF 命令，用户标识必须对目标系统上的 IBM MQ 对象具有 *dsp* 权限。

此外，将对某些 PCF 命令执行 IBM MQ 对象权限检查，如 [第 26 页的表 1](#) 中所示。

在大多数情况下，这些检查与本地系统上发出的等效 IBM MQ CL 命令执行的检查相同。请参阅 [在 IBM i 上设置安全性](#)，以获取有关从 IBM MQ 权限到 IBM i 系统权限的映射以及 IBM MQ CL 命令的权限需求的更多信息。[使用安全出口的链接级别安全性](#) 文档中提供了有关出口的安全性的详细信息。

要处理以下任何命令，用户标识必须是组概要文件 QMQMADM 的成员：

- Ping 通道
- 更改通道
- 复制通道
- 创建通道
- 删除通道
- 重置通道
- 解析通道
- 启动通道
- 停止通道

- 启动通道启动程序
- 启动通道侦听器

IBM MQ for UNIX, Linux, and Windows

ULW

为了处理任何 PCF 命令，用户标识必须对目标系统上的队列管理器对象具有 *dsp* 权限。此外，将对某些 PCF 命令执行 IBM MQ 对象权限检查，如 [第 26 页的表 1](#) 中所示。

要处理以下任何命令，用户标识必须属于组 *mqm*。

注: 仅对于 Windows，用户标识可以属于组 *Administrators* 或组 *mqm*。

- 更改通道
- 复制通道
- 创建通道
- 删除通道
- Ping 通道
- 重置通道
- 启动通道
- 停止通道
- 启动通道启动程序
- 启动通道侦听器
- 解析通道
- Reset Cluster
- 刷新集群
- 暂挂队列管理器
- 恢复队列管理器

多平台的 IBM MQ 对象权限

Multi

| 命令 | IBM MQ 对象权限 | 类权限 (针对对象类型) |
|-----------|-------------------------|--------------|
| 更改认证信息 | dsp 和 chg | 不适用 |
| 更改通道 | dsp 和 chg | 不适用 |
| 更改通道侦听器 | dsp 和 chg | 不适用 |
| 更改客户机连接通道 | dsp 和 chg | 不适用 |
| 更改名称列表 | dsp 和 chg | 不适用 |
| 更改进程 | dsp 和 chg | 不适用 |
| 更改队列 | dsp 和 chg | 不适用 |
| 更改队列管理器 | chg 请参阅 "注释" 3 和 "注释" 5 | 不适用 |
| 更改服务 | dsp 和 chg | 不适用 |
| 清除队列 | clr | 不适用 |
| 复制认证信息 | dsp | crt |

| 表 1: 对象权限 (继续) | | |
|------------------------|-----------------------------|--------------|
| 命令 | IBM MQ 对象权限 | 类权限 (针对对象类型) |
| 复制认证信息 (替换) 请参阅注释 1 | 从: dsp 到: chg | crt |
| 复制通道 | dsp | crt |
| 复制通道 (替换) 请参阅注释 1 | 从: dsp 到: chg | crt |
| 复制通道侦听器 | dsp | crt |
| 复制通道侦听器 (替换) 请参阅注释 1 | 从: dsp 到: chg | crt |
| 复制客户机连接通道 | dsp | crt |
| 复制客户机连接通道 (替换) 请参阅注释 1 | 从: dsp 到: chg | crt |
| 复制名称列表 | dsp | crt |
| 复制名称列表 (替换) 请参阅注释 1 | 从 dsp 到 dsp 和 chg | crt |
| 复制进程 | dsp | crt |
| 复制过程 (替换) 请参阅注释 1 | 从: dsp 到: chg | crt |
| 复制队列 | dsp | crt |
| 复制队列 (替换) 请参阅注释 1 | 从 dsp 到 dsp 和 chg | crt |
| 创建认证信息 | (系统缺省认证信息) dsp | crt |
| 创建认证信息 (替换) 请参阅注释 1 | (系统缺省认证信息) dsp 到: chg | crt |
| 创建通道 | (系统缺省通道) dsp | crt |
| 创建通道 (替换) 请参阅注释 1 | (系统缺省通道) dsp 到: chg | crt |
| 创建通道侦听器 | (系统缺省侦听器) dsp | crt |
| 创建通道侦听器 (替换) 请参阅注释 1 | (系统缺省侦听器) dsp 到: chg | crt |
| 创建客户机连接通道 | (系统缺省通道) dsp | crt |
| 创建客户机连接通道 (替换) 请参阅注释 1 | (系统缺省通道) dsp 到: chg | crt |
| 创建名称列表 | (系统缺省名称列表) dsp | crt |
| 创建名称列表 (替换) 请参阅注释 1 | (系统缺省名称列表) dsp 到: dsp 和 chg | crt |
| 创建进程 | (系统缺省进程) dsp | crt |
| 创建过程 (替换) 请参阅注释 1 | (系统缺省进程) dsp 到: chg | crt |
| 创建队列 | (系统缺省队列) dsp | crt |
| 创建队列 (替换) 请参阅注释 1 | (系统缺省队列) dsp 到: dsp 和 chg | crt |
| 创建服务 | (系统缺省队列) dsp | crt |
| 创建服务 (替换) 请参阅注释 1 | (系统缺省队列) dsp 到: chg | crt |
| 删除认证信息 | dsp 和 dlt | 不适用 |
| 删除权限记录 | (队列管理器对象) chg 请参阅注释 4 | 请参阅注释 4 |
| 删除通道 | dsp 和 dlt | 不适用 |

| 表 1: 对象权限 (继续) | | |
|----------------------------------------------|-----------------------|--------------|
| 命令 | IBM MQ 对象权限 | 类权限 (针对对象类型) |
| 删除通道侦听器 | dsp 和 dlt | 不适用 |
| 删除客户机连接通道 | dsp 和 dlt | 不适用 |
| 删除名称列表 | dsp 和 dlt | 不适用 |
| 删除进程 | dsp 和 dlt | 不适用 |
| 删除队列 | dsp 和 dlt | 不适用 |
| 删除服务 | dsp 和 dlt | 不适用 |
| 查询认证信息 | dsp | 不适用 |
| 查询权限记录 | 请参阅注释 4 | 请参阅注释 4 |
| 查询通道 | dsp | 不适用 |
| 查询通道侦听器 | dsp | 不适用 |
| 查询通道状态 (针对 ChannelType MQCHT_CLSSDR) | inq | 不适用 |
| 查询客户机连接通道 | dsp | 不适用 |
| 查询名称列表 | dsp | 不适用 |
| 查询进程 | dsp | 不适用 |
| 查询队列 | dsp | 不适用 |
| 查询队列管理器 | 请参阅注释 3 | 不适用 |
| 查询队列状态 | dsp | 不适用 |
| 查询服务 | dsp | 不适用 |
| Ping 通道 | ctrl | 不适用 |
| Ping 队列管理器 | 请参阅注释 3 | 不适用 |
| 刷新队列管理器 | (队列管理器对象) chg | 不适用 |
| 刷新安全性 (针对 SecurityType MQSECTYPE_SSL) | (队列管理器对象) chg | 不适用 |
| 重置通道 | ctrlx | 不适用 |
| 重置队列管理器 | (队列管理器对象) chg | 不适用 |
| 重置队列统计信息 | dsp 和 chg | 不适用 |
| 解析通道 | ctrlx | 不适用 |
| 设置权限记录 | (队列管理器对象) chg 请参阅注释 4 | 请参阅注释 4 |
| 启动通道 | ctrl | 不适用 |
| 停止通道 | ctrl | 不适用 |
| 停止连接 | (队列管理器对象) chg | 不适用 |
| 启动侦听器 | ctrl | 不适用 |
| 停止侦听器 | ctrl | 不适用 |
| 启动服务 | ctrl | 不适用 |

| 表 1: 对象权限 (继续) | | |
|----------------|-------------|--------------|
| 命令 | IBM MQ 对象权限 | 类权限 (针对对象类型) |
| 停止服务 | ctrl | 不适用 |
| 转义 | 请参阅注释 2 | 请参阅注释 2 |

注意:

1. 如果要替换的对象存在, 那么此命令适用, 否则权限检查与 "创建" 或 "复制而不替换" 一样。
2. 所需权限由转义文本定义的 MQSC 命令确定, 它相当于之前的一个命令。
3. 为了处理任何 PCF 命令, 用户标识必须对目标系统上的队列管理器对象具有 dsp 权限。
4. 除非已使用 -a 参数启动命令服务器, 否则将授权此 PCF 命令。缺省情况下, 命令服务器在队列管理器启动时启动, 并且不使用 -a 参数。有关更多信息, 请参阅 [可编程命令格式参考](#)。
5. 授予队列管理器的用户标识 chg 权限使您能够为所有组和用户设置权限记录。请勿将此权限授予普通用户或应用程序。

IBM MQ 还提供了一些通道安全出口点, 以便您可以提供自己的用户出口程序以进行安全检查。有关更多信息, 请参阅 [显示通道](#)。

Multi 使用 MQAI 来简化 PCF 的使用

IBM MQ 管理接口 (MQAI) 是 IBM MQ 的编程接口, 在 AIX, HP-UX, IBM i, Linux, Solaris, 和 Windows 上提供。它使用数据包在 IBM MQ 队列管理器上执行管理任务, 以比使用可编程命令格式 (PCF) 更容易的方式处理对象的属性 (或参数)。

MQAI 通过使用数据包在队列管理器上执行管理任务。数据包允许您以比使用 PCF 更容易的方式处理对象的属性 (或参数)。

使用 MQAI 的优点如下所示:

简化 PCF 消息的使用

MQAI 是一种更易于管理 IBM MQ 的方法。如果使用 MQAI, 那么不必编写自己的 PCF 消息。这可避免与复杂数据结构相关联的问题。

要传递使用 MQI 调用编写的程序中的参数, PCF 消息必须包含命令以及字符串或整数数据的详细信息。要手动创建此配置, 必须在程序中为每个结构添加多个语句, 并且必须分配内存空间。这个任务可能漫长而费力。

使用 MQAI 编写的程序会将参数传递到相应的数据包中, 并且每个结构只需要一个语句。使用 MQAI 数据包可消除您处理阵列和分配存储器的需求, 并提供与 PCF 详细信息的某种程度的隔离。

更轻松地处理错误情况

很难从 PCF 命令中获取返回码。MQAI 使程序更容易处理错误情况。

在应用程序之间交换数据

应用程序数据以 PCF 格式发送, 并由 MQAI 打包和解包。如果消息数据由整数和字符串组成, 那么可以使用 MQAI 来利用 PCF 数据的 IBM MQ 内置数据转换。这将避免需要写入数据转换出口。

创建并填充数据包后, 可以使用 mqExecute 调用将管理命令消息发送到队列管理器的命令服务器。此调用将等待任何响应消息。mqExecute 调用处理与命令服务器的交换, 并在响应包中返回响应。

使用 MQAI 的示例

显示的列表提供了一些示例程序, 用于演示 MQAI 的使用。样本执行以下任务:

1. 创建本地队列。第 35 页的『[用于创建本地队列的样本 C 程序 \(amqsaicq.c\)](#)』
2. 使用简单事件监视器在屏幕上显示事件。第 38 页的『[用于使用事件监视器显示事件的样本 C 程序 \(amqsaiem.c\)](#)』
3. 打印所有本地队列及其当前深度的列表。第 50 页的『[用于查询队列和打印信息的样本 C 程序 \(amqsailq.c\)](#)』

4. 打印所有通道及其类型的列表。 [第 45 页的『用于查询通道对象的样本 C 程序 \(amqsaicl.c\)』](#)

构建 MQAI 应用程序

要使用 MQAI 构建应用程序，请链接到与 IBM MQ 相同的库。有关如何构建 IBM MQ 应用程序的信息，请参阅 [构建过程应用程序](#)。

有关使用 MQAI 配置 IBM MQ 的提示和技巧

MQAI 使用 PCF 消息将管理命令发送到命令服务器，而不是直接处理命令服务器本身。可在 [第 30 页的『有关使用 MQAI 配置 IBM MQ 的提示和技巧』](#) 中找到有关使用 MQAI 配置 IBM MQ 的提示。

相关信息

[IBM MQ 管理界面参考](#)

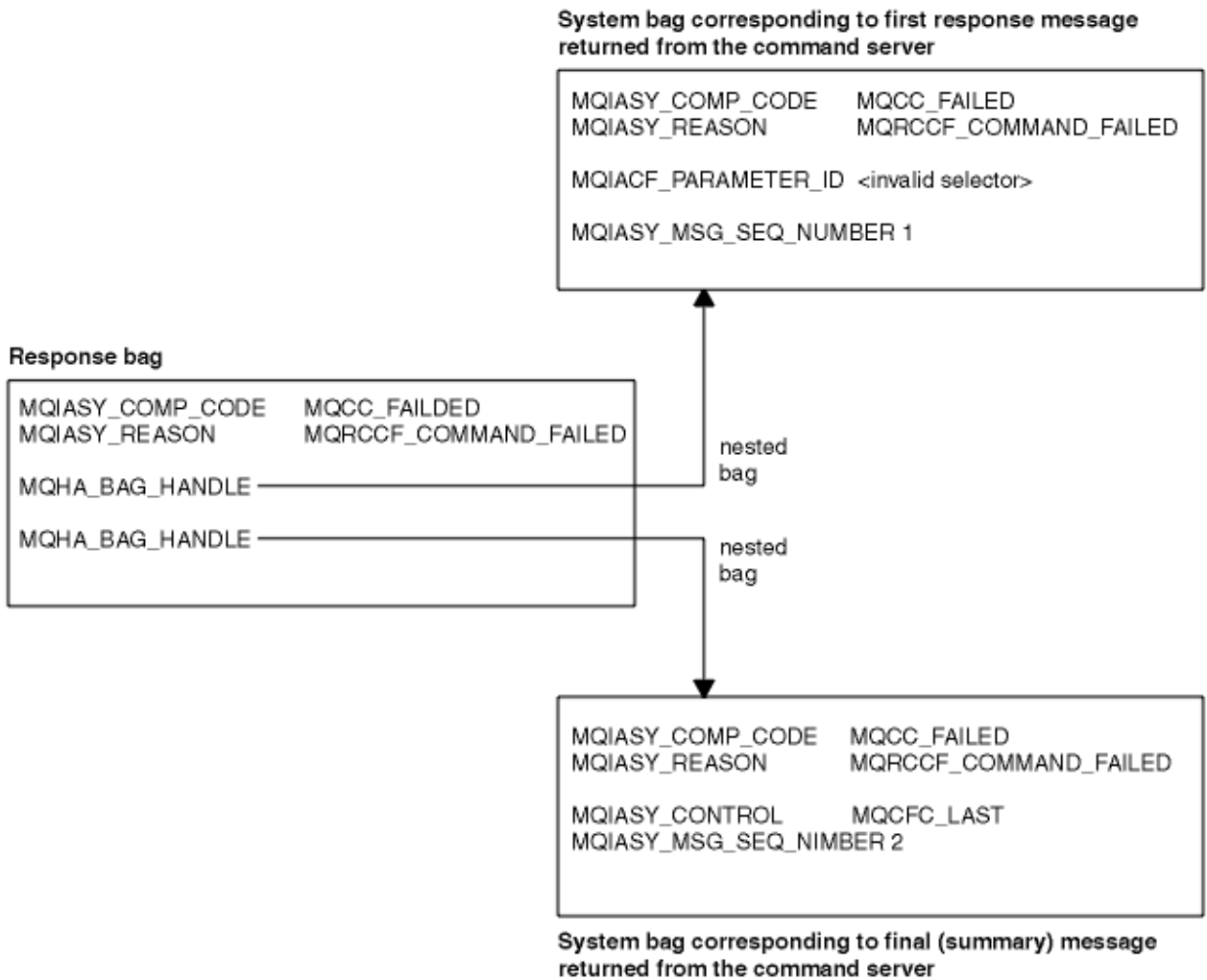
有关使用 MQAI 配置 IBM MQ 的提示和技巧

IBM MQ 管理接口 (MQAI) 使用 PCF 消息将管理命令发送到命令服务器，而不是直接处理命令服务器本身。以下是使用 MQAI 配置 IBM MQ 的一些提示。

- IBM MQ 中的字符串为空白，填充为固定长度。通过使用 C，通常可以将以 null 结束的字符串作为输入参数提供给 IBM MQ 编程接口。
- 要清除字符串属性的值，请将其设置为单个空白而不是空字符串。
- 请提前考虑要更改的属性，并仅查询这些属性。
- 无法更改某些属性，例如，队列名称或通道类型。请确保尝试仅更改可修改的那些属性。请参阅特定 PCF 更改对象的必需参数和可选参数的列表。请参阅 [可编程命令格式的定义](#)。
- 如果 MQAI 调用失败，那么会将失败的一些详细信息返回到响应包。然后，可以在可由选择器 MQHA_BAG_HANDLE 访问的嵌套包中找到更多详细信息。例如，如果 mqExecute 调用失败，并且原因码为 MQRCCF_COMMAND_FAILED，那么将在响应包中返回此信息。此原因码的可能原因是指定的选择器对于命令消息类型无效，并且在可由包句柄访问的嵌套包中找到此详细信息。

有关 MQExecute 的更多信息，请参阅 [第 62 页的『使用 mqExecute 调用将管理命令发送到 qm 命令服务器』](#)

下图显示了此场景：



高级 MQAI 主题

关于建立索引，数据转换和使用消息描述符的信息

- 建立索引

在从包中替换或删除现有数据项以保留插入顺序时，将使用索引。可在 [第 31 页的『在 MQAI 中建立索引』](#) 中找到有关建立索引的完整详细信息。

- 数据转换

MQAI 数据包中包含的字符串可以包含在各种编码字符集中，并且可以使用 mqSet 整数调用来转换这些字符串。可在 [第 32 页的『MQAI 中的数据转换处理』](#) 中找到有关数据转换的完整详细信息。

- 使用消息描述符

MQAI 生成在创建数据包时设置为初始值的消息描述符。可以在 [第 33 页的『在 MQAI 中使用消息描述符』](#) 中找到使用消息描述符的完整详细信息。

在 MQAI 中建立索引

当从包中替换或删除现有数据项时，将使用索引。有三种类型的索引，可以轻松检索数据项。

数据包中数据项内的每个选择器和值都有三个关联的索引号：

- 相对于具有相同选择器的其他项的索引。
- 相对于项所属的选择器 (用户或系统) 类别的索引。
- 相对于包中所有数据项 (用户和系统) 的索引。

这允许按用户选择器和/或系统选择器建立索引，如 [第 32 页的图 4](#) 中所示。

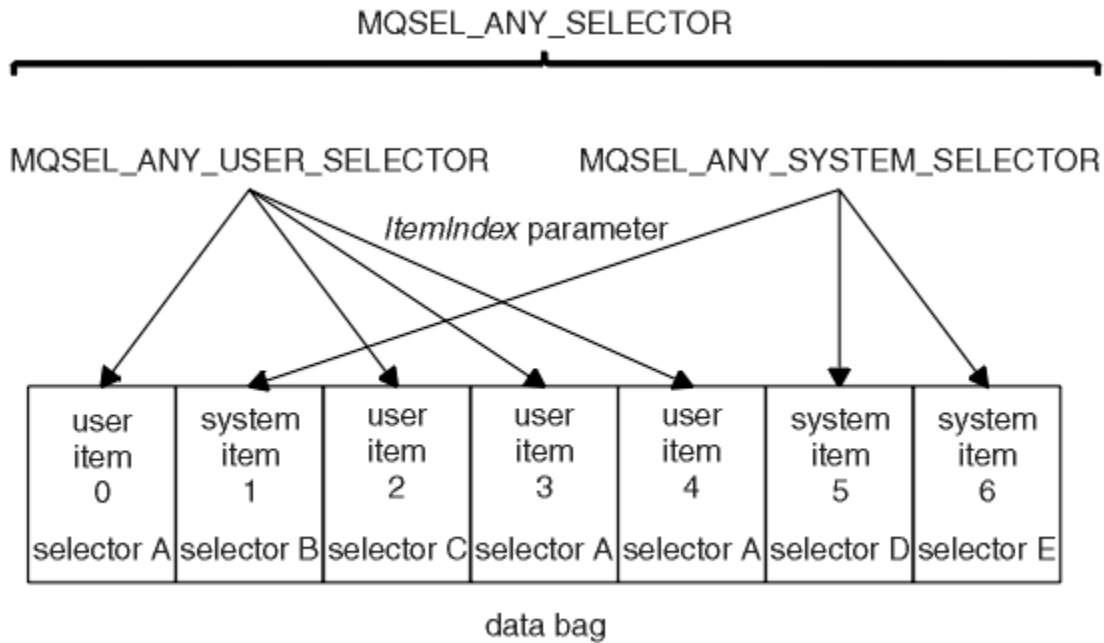


图 4: 建立索引

在图 第 32 页的图 4 中，用户项 3 (选择器 A) 可由以下索引对引用：

| Selector | ItemIndex |
|-------------------------|-----------|
| 选择器 A | 1 |
| MQSEL_ANY_USER_SELECTOR | 2 |
| MQSEL_ANY_SELECTOR | 3 |

索引以零为基础，类似于 C 中的数组；如果出现 "n"，那么索引的范围从 0 到 "n-1"，没有间隔。

当从包中替换或除去现有数据项时，将使用索引。以此方式使用时，将保留插入顺序，但可能会影响其他数据项的索引。有关此操作的示例，请参阅 [更改包中的信息](#) 和 [删除数据项](#)。

三种类型的索引允许轻松检索数据项。例如，如果包中有三个特定选择器实例，那么 mqCount 项调用可以计算该选择器的实例数，而 mqInquire* 调用可以同时指定选择器和索引以仅查询这些值。这对于具有值列表 (例如通道上的某些出口) 的属性很有用。

MQAI 中的数据转换处理

MQAI 数据包中包含的字符串可以包含在各种编码字符集中。可以使用 mqSet 整数调用来转换这些字符串。

与 PCF 消息一样，MQAI 数据包中包含的字符串可以包含在各种编码字符集中。通常，PCF 消息中的所有字符串都使用相同的编码字符集；即，与队列管理器相同的集合。

数据包中的每个字符串项都包含两个值：字符串本身和 CCSID。添加到包中的字符串可从 mqAddString 或 mqSetString 调用的 **Buffer** 参数获取。从包含 MQIASY_CODED_CHAR_SET_ID 选择器的系统项获取 CCSID。这称为包 CCSID，可以使用 mqSet 整数调用进行更改。

当查询数据包中包含的字符串的值时，CCSID 是来自调用的输出参数。

第 32 页的表 2 显示了将数据包转换为消息时应用的规则，反之亦然：

| 表 2: CCSID 处理 | | | |
|---------------|-------------|--------|--------|
| MQAI 调用 | CCSID | 要调用的输入 | 要调用的输出 |
| mqBagToBuffer | 包 CCSID (1) | 已忽略 | 无变化 |

| 表 2: CCSID 处理 (继续) | | | |
|--------------------|----------------------|---------------------------|-----------------------------|
| MQAI 调用 | CCSID | 要调用的输入 | 要调用的输出 |
| mqBagToBuffer | 包中的字符串 CCSID | 已使用 | 无变化 |
| mqBagToBuffer | 缓冲区中的字符串 CCSID | 不适用 | 从包中的字符串 CCSID 复制 |
| mqBufferToBag | 包 CCSID (<u>1</u>) | 已忽略 | 无变化 |
| mqBufferToBag | 缓冲区中的字符串 CCSID | 已使用 | 无变化 |
| mqBufferToBag | 包中的字符串 CCSID | 不适用 | 从缓冲区中的字符串 CCSID 复制 |
| mqPut 包 | MQMD CCSID | 已使用 | 未更改 (<u>2</u>) |
| mqPut 包 | 包 CCSID (<u>1</u>) | 已忽略 | 无变化 |
| mqPut 包 | 包中的字符串 CCSID | 已使用 | 无变化 |
| mqPut 包 | 发送的消息中的字符串 CCSID | 不适用 | 从包中的字符串 CCSID 复制 |
| mqGet 包 | MQMD CCSID | 用于消息的数据转换 | 设置为返回数据的 CCSID (<u>3</u>) |
| mqGet 包 | 包 CCSID (<u>1</u>) | 已忽略 | 无变化 |
| mqGet 包 | 消息中的字符串 CCSID | 已使用 | 无变化 |
| mqGet 包 | 包中的字符串 CCSID | 不适用 | 从消息中的字符串 CCSID 复制 |
| mqExecute | 请求包 CCSID | 用于请求消息的 MQMD (<u>4</u>) | 无变化 |
| mqExecute | 应答包 CCSID | 用于应答消息 (<u>4</u>) 的数据转换 | 设置为返回数据的 CCSID (<u>3</u>) |
| mqExecute | 请求包中的字符串 CCSID | 用于请求消息 | 无变化 |
| mqExecute | 应答包中的字符串 CCSID | 不适用 | 从应答消息中的字符串 CCSID 复制 |

注意:

1. 包 CCSID 是具有选择器 MQIASY_CODED_CHAR_SET_ID 的系统项。
2. MQCCSI_Q_MGR 更改为实际队列管理器 CCSID。
3. 如果请求数据转换，那么返回的数据的 CCSID 与输出值相同。如果未请求数据转换，那么返回的数据的 CCSID 与消息值相同。请注意，如果请求数据转换但失败，那么不会返回任何消息。
4. 如果 CCSID 是 MQCCSI_DEFAULT，那么将使用队列管理器的 CCSID。

相关信息

数据转换

[ccsid_part2.tbl 文件](#)

在 MQAI 中使用消息描述符

创建数据包时，MQAI 生成的消息描述符将设置为初始值。

从具有选择器 MQIASY_TYPE 的系统项获取 PCF 命令类型。创建数据包时，将根据您创建的数据包类型来设置此项的初始值:

| 袋子的类型 | MQIASY_TYPE 项的初始值 |
|-------------------|-------------------|
| MQCBO_ADMIN_BAG | MQCFT_COMMAND |
| MQCBO_COMMAND_BAG | MQCFT_COMMAND |
| MQCBO_* | MQCFT_USER |

MQAI 生成消息描述符时，**Format** 和 **MsgType** 参数中使用的值取决于具有选择器 MQIASY_TYPE 的系统项的值，如第 34 页的表 3 中所示。

| PCF 命令类型 | 格式 | MsgType |
|-------------------|-------------|---------------|
| MQCFT_COMMAND | MQFMT_ADMIN | MQMT_REQUEST |
| MQCFT_REPORT | MQFMT_ADMIN | MQMT_REPORT |
| MQCFT_RESPONSE | MQFMT_ADMIN | MQMT_REPLY |
| MQCFT_TRACE_ROUTE | MQFMT_ADMIN | MQMT_DATAGRAM |
| MQCFT_EVENT | MQFMT_EVENT | MQMT_DATAGRAM |
| MQCFT_* | MQFMT_PCF | MQMT_DATAGRAM |

第 34 页的表 4 显示了如果创建管理包或命令包，那么消息描述符的 *Format* 为 MQFMT_ADMIN，而 *MsgType* 为 MQMT_REQUEST。这适用于在期望返回响应时发送到命令服务器的 PCF 请求消息。

消息描述符中的其他参数采用第 34 页的表 5 中显示的值。

| 参数 | 值 |
|-----------------------|---------------------------|
| <i>StrucId</i> | MQMD_STRUC_ID |
| <i>Version</i> | MQMD_VERSION_1 |
| <i>Report</i> | MQRO_NONE |
| <i>MsgType</i> | 请参阅第 34 页的表 4 |
| <i>Expiry</i> | 30 秒 (注第 35 页的『1』) |
| <i>Feedback</i> | MQFB_NONE |
| <i>Encoding</i> | MQENC_NATIVE |
| <i>CodedCharSetId</i> | 依赖于包 CCSID (注释第 35 页的『2』) |
| <i>Format</i> | 请参阅第 34 页的表 4 |
| <i>Priority</i> | MQPRI_PRIORITY_AS_Q_DEF |
| <i>Persistence</i> | MQPER_NOT_PERSISTENT |
| <i>MsgId</i> | MQMI_NONE |
| <i>CorrelId</i> | MQCI_NONE |
| <i>BackoutCount</i> | 0 |
| <i>ReplyToQ</i> | 请参阅注释第 35 页的『3』 |
| <i>ReplyToQMgr</i> | 空白 |

注意:

1. 可以使用 **OptionsBag** 参数在 mqExecute 调用上覆盖此值。有关此操作的信息，请参阅 [mqExecute](#)。
2. 请参阅第 32 页的『MQAI 中的数据转换处理』。
3. 类型为 MQMT_REQUEST 的消息的用户指定的应答队列或 MQAI 生成的临时动态队列的名称。否则为空白。

用于创建本地队列的样本 C 程序 (amqsaicq.c)

样本 C 程序 amqsaicq.c 使用 MQAI 创建本地队列。

```
/*
/*
/* Program name: AMQSAICQ.C
/*
/* Description: Sample C program to create a local queue using the
/* IBM MQ Administration Interface (MQAI).
/*
/* Statement: Licensed Materials - Property of IBM
/*
/* 84H2000, 5765-B73
/* 84H2001, 5639-B42
/* 84H2002, 5765-B74
/* 84H2003, 5765-B75
/* 84H2004, 5639-B43
/*
/* (C) Copyright IBM Corp. 1999, 2023.
/*
/*
/*
/* Function:
/* AMQSAICQ is a sample C program that creates a local queue and is an
/* example of the use of the mqExecute call.
/*
/* - The name of the queue to be created is a parameter to the program.
/*
/* - A PCF command is built by placing items into an MQAI bag.
/* These are:-
/* - The name of the queue
/* - The type of queue required, which, in this case, is local.
/*
/* - The mqExecute call is executed with the command MQCMD_CREATE_Q.
/* The call generates the correct PCF structure.
/* The call receives the reply from the command server and formats into
/* the response bag.
/*
/* - The completion code from the mqExecute call is checked and if there
/* is a failure from the command server then the code returned by the
/* command server is retrieved from the system bag that is
/* embedded in the response bag to the mqExecute call.
/*
/* Note: The command server must be running.
/*
/*
/*
/*
/* AMQSAICQ has 2 parameters - the name of the local queue to be created
/* - the queue manager name (optional)
/*
/*
/* Includes
/*
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>

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

void CheckCallResult(MQCHAR *, MQLONG , MQLONG );
void CreateLocalQueue(MQHCONN, MQCHAR *);
```

```

int main(int argc, char *argv[])
{
    MQHCONN hConn; /* handle to IBM MQ connection */
    MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name */
    MQLONG connReason; /* MQCONN reason code */
    MQLONG compCode; /* completion code */
    MQLONG reason; /* reason code */

    /*****
    /* First check the required parameters */
    /*****
    printf("Sample Program to Create a Local Queue\n");
    if (argc < 2)
    {
        printf("Required parameter missing - local queue name\n");
        exit(99);
    }

    /*****
    /* Connect to the queue manager */
    /*****
    if (argc > 2)
        strncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
        MQCONN(QMName, &hConn, &compCode, &connReason);

    /*****
    /* Report reason and stop if connection failed */
    /*****
    if (compCode == MQCC_FAILED)
    {
        CheckCallResult("MQCONN", compCode, connReason);
        exit( (int)connReason);
    }

    /*****
    /* Call the routine to create a local queue, passing the handle to the
    /* queue manager and also passing the name of the queue to be created.
    /*****
    CreateLocalQueue(hConn, argv[1]);

    /*****
    /* Disconnect from the queue manager if not already connected */
    /*****
    if (connReason != MQRC_ALREADY_CONNECTED)
    {
        MQDISC(&hConn, &compCode, &reason);
        CheckCallResult("MQDISC", compCode, reason);
    }
    return 0;
}

/*****
/*
/* Function: CreateLocalQueue */
/* Description: Create a local queue by sending a PCF command to the command
/* server. */
/*
/*****
/*
/* Input Parameters: Handle to the queue manager */
/* Name of the queue to be created */
/*
/* Output Parameters: None */
/*
/* Logic: The mqExecute call is executed with the command MQCMD_CREATE_Q.
/* The call generates the correct PCF structure.
/* The default options to the call are used so that the command is sent
/* to the SYSTEM.ADMIN.COMMAND.QUEUE.
/* The reply from the command server is placed on a temporary dynamic
/* queue.
/* The reply is read from the temporary queue and formatted into the
/* response bag.
/*
/* The completion code from the mqExecute call is checked and if there
/* is a failure from the command server then the code returned by the
/* command server is retrieved from the system bag that is
/* embedded in the response bag to the mqExecute call.
/*
/*****
void CreateLocalQueue(MQHCONN hConn, MQCHAR *qName)
{

```

```

MQLONG reason; /* reason code */
MQLONG compCode; /* completion code */
MQHBAG commandBag = MQHB_UNUSABLE_HBAG; /* command bag for mqExecute */
MQHBAG responseBag = MQHB_UNUSABLE_HBAG; /* response bag for mqExecute */
MQHBAG resultBag; /* result bag from mqExecute */
MQLONG mqExecuteCC; /* mqExecute completion code */
MQLONG mqExecuteRC; /* mqExecute reason code */

printf("\nCreating Local Queue %s\n\n", qName);

/*****
/* Create a command Bag for the mqExecute call. Exit the function if the
/* create fails.
*****/
mqCreateBag(MQCBO_ADMIN_BAG, &commandBag, &compCode, &reason);
CheckCallResult("Create the command bag", compCode, reason);
if (compCode !=MQCC_OK)
    return;

/*****
/* Create a response Bag for the mqExecute call, exit the function if the
/* create fails.
*****/
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
CheckCallResult("Create the response bag", compCode, reason);
if (compCode !=MQCC_OK)
    return;

/*****
/* Put the name of the queue to be created into the command bag. This will
/* be used by the mqExecute call.
*****/
mqAddString(commandBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, qName, &compCode,
            &reason);
CheckCallResult("Add q name to command bag", compCode, reason);

/*****
/* Put queue type of local into the command bag. This will be used by the
/* mqExecute call.
*****/
mqAddInteger(commandBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
CheckCallResult("Add q type to command bag", compCode, reason);

/*****
/* Send the command to create the required local queue.
/* The mqExecute call will create the PCF structure required, send it to
/* the command server and receive the reply from the command server into
/* the response bag.
*****/
mqExecute(hConn, /* IBM MQ connection handle */
          MQCMD_CREATE_Q, /* Command to be executed */
          MQHB_NONE, /* No options bag */
          commandBag, /* Handle to bag containing commands */
          responseBag, /* Handle to bag to receive the response */
          MQHO_NONE, /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE */
          MQHO_NONE, /* Create a dynamic q for the response */
          &compCode, /* Completion code from the mqExecute */
          &reason); /* Reason code from mqExecute call */

if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
    printf("Please start the command server: <strmqcsv QMgrName>\n")
    MQDISC(&hConn, &compCode, &reason);
    CheckCallResult("MQDISC", compCode, reason);
    exit(98);
}

/*****
/* Check the result from mqExecute call and find the error if it failed.
*****/
if ( compCode == MQCC_OK )
    printf("Local queue %s successfully created\n", qName);
else
{
    printf("Creation of local queue %s failed: Completion Code = %d
           qName, compCode, reason);
    if (reason == MQRCCF_COMMAND_FAILED)
    {
        /*****
        /* Get the system bag handle out of the mqExecute response bag.
        *****/

```

```

/* This bag contains the reason from the command server why the      */
/* command failed.                                                  */
/*****
mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &resultBag, &compCode,
              &reason);
CheckCallResult("Get the result bag handle", compCode, reason);

/*****
/* Get the completion code and reason code, returned by the command */
/* server, from the embedded error bag.                               */
/*****
mqInquireInteger(resultBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                 &compCode, &reason);
CheckCallResult("Get the completion code from the result bag",
                compCode, reason);
mqInquireInteger(resultBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                 &compCode, &reason);
CheckCallResult("Get the reason code from the result bag", compCode,
                reason);
printf("Error returned by the command server: Completion code = %d :
       Reason = %d\n", mqExecuteCC, mqExecuteRC);
}
}
/*****
/* Delete the command bag if successfully created.                  */
/*****
if (commandBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&commandBag, &compCode, &reason);
    CheckCallResult("Delete the command bag", compCode, reason);
}
}

/*****
/* Delete the response bag if successfully created.                */
/*****
if (responseBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&responseBag, &compCode, &reason);
    CheckCallResult("Delete the response bag", compCode, reason);
}
}
} /* end of CreateLocalQueue */

/*****
/*
/* Function: CheckCallResult
/*
/*****
/*
/* Input Parameters:  Description of call
/*                   Completion code
/*                   Reason code
/*
/* Output Parameters: None
/*
/* Logic: Display the description of the call, the completion code and the
/*        reason code if the completion code is not successful
/*
/*****
void CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
    if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d :
              Reason = %d\n", callText, cc, rc);
}
}

```

用于使用事件监视器显示事件的样本 C 程序 (amqsaiem.c)

样本 C 程序 amqsaiem.c 演示了使用 MQAI 的基本事件监视器。

```

/*****
/*
/* Program name: AMQSAIEM.C
/*
/* Description: Sample C program to demonstrate a basic event monitor
/*              using the IBM MQ Admin Interface (MQAI).
/* Licensed Materials - Property of IBM
*/

```

```

/*
/* 63H9336
/* (c) Copyright IBM Corp. 1999, 2023. All Rights Reserved.
/*
/* US Government Users Restricted Rights - Use, duplication or
/* disclosure restricted by GSA ADP Schedule Contract with
/* IBM Corp.
/*****
/*
/* Function:
/* AMQSAIEM is a sample C program that demonstrates how to write a simple
/* event monitor using the mqGetBag call and other MQAI calls.
/*
/* The name of the event queue to be monitored is passed as a parameter
/* to the program. This would usually be one of the system event queues:-
/* SYSTEM.ADMIN.QMGR.EVENT Queue Manager events
/* SYSTEM.ADMIN.PERFM.EVENT Performance events
/* SYSTEM.ADMIN.CHANNEL.EVENT Channel events
/* SYSTEM.ADMIN.LOGGER.EVENT Logger events
/*
/* To monitor the queue manager event queue or the performance event queue,
/* the attributes of the queue manager need to be changed to enable
/* these events. For more information about this, see Part 1 of the
/* Programmable System Management book. The queue manager attributes can
/* be changed using either MQSC commands or the MQAI interface.
/* Channel events are enabled by default.
/*
/* Program logic
/* Connect to the Queue Manager.
/* Open the requested event queue with a wait interval of 30 seconds.
/* Wait for a message, and when it arrives get the message from the queue
/* and format it into an MQAI bag using the mqGetBag call.
/* There are many types of event messages and it is beyond the scope of
/* this sample to program for all event messages. Instead the program
/* prints out the contents of the formatted bag.
/* Loop around to wait for another message until either there is an error
/* or the wait interval of 30 seconds is reached.
/*
/*****
/*
/* AMQSAIEM has 2 parameters - the name of the event queue to be monitored
/* - the queue manager name (optional)
/*
/*****

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

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

/*****
/* Macros
/*****
#if MQAT_DEFAULT == MQAT_WINDOWS_NT
#define Int64 "I64"
#elif defined(MQ_64_BIT)
#define Int64 "l"
#else
#define Int64 "ll"
#endif

/*****
/* Function prototypes
/*****
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);
void GetQEvents(MQHCONN, MQCHAR *);
int PrintBag(MQHBAG);
int PrintBagContents(MQHBAG, int);

/*****
/* Function: main
/*****
int main(int argc, char *argv[])
{
MQHCONN hConn; /* handle to connection

```

```

MQCHAR QMName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QM name */
MQLONG reason; /* reason code */
MQLONG connReason; /* MQCONN reason code */
MQLONG compCode; /* completion code */

/*****
/* First check the required parameters */
/*****
printf("Sample Event Monitor (times out after 30 secs)\n");
if (argc < 2)
{
    printf("Required parameter missing - event queue to be monitored\n");
    exit(99);
}

/*****
/* Connect to the queue manager */
/*****
if (argc > 2)
    stncpy(QMName, argv[2], (size_t)MQ_Q_MGR_NAME_LENGTH);
MQCONN(QMName, &hConn, &compCode, &connReason);
/*****
/* Report the reason and stop if the connection failed */
/*****
if (compCode == MQCC_FAILED)
{
    CheckCallResult("MQCONN", compCode, connReason);
    exit( (int)connReason);
}

/*****
/* Call the routine to open the event queue and format any event messages */
/* read from the queue. */
/*****
GetQEvents(hConn, argv[1]);

/*****
/* Disconnect from the queue manager if not already connected */
/*****
if (connReason != MQRC_ALREADY_CONNECTED)
{
    MQDISC(&hConn, &compCode, &reason);
    CheckCallResult("MQDISC", compCode, reason);
}

return 0;
}

/*****
/*
/* Function: CheckCallResult */
/*
/*****
/*
/* Input Parameters: Description of call */
/* Completion code */
/* Reason code */
/*
/* Output Parameters: None */
/*
/* Logic: Display the description of the call, the completion code and the */
/* reason code if the completion code is not successful */
/*
/*
/*****
void CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
    if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %d : Reason = %d\n",
            callText, cc, rc);
}

/*****
/*
/* Function: GetQEvents */
/*
/*****
/*
/* Input Parameters: Handle to the queue manager */
/* Name of the event queue to be monitored */
/*

```



```

/* Output Parameters: None */
/*
/* Logic: Open the event queue. */
/* Get a message off the event queue and format the message into */
/* a bag. */
/* A real event monitor would need to be programmed to deal with */
/* each type of event that it receives from the queue. This is */
/* outside the scope of this sample, so instead, the contents of */
/* the bag are printed. */
/* The program waits for 30 seconds for an event message and then */
/* terminates if no more messages are available. */
/*
/*****
void GetQEvents(MQHCONN hConn, MQCHAR *qName)
{
    MQLONG openReason; /* MQOPEN reason code */
    MQLONG reason; /* reason code */
    MQLONG compCode; /* completion code */
    MQHOBJ eventQueue; /* handle to event queue */

    MQHBAG eventBag = MQHB_UNUSABLE_HBAG; /* event bag to receive event msg */
    MQOD od = {MQOD_DEFAULT}; /* Object Descriptor */
    MQMD md = {MQMD_DEFAULT}; /* Message Descriptor */
    MQGMO gmo = {MQGMO_DEFAULT}; /* get message options */
    MQLONG bQueueOK = 1; /* keep reading msgs while true */

    /*****
    /* Create an Event Bag in which to receive the event. */
    /* Exit the function if the create fails. */
    /*****
    mqCreateBag(MQCBO_USER_BAG, &eventBag, &compCode, &reason);
    CheckCallResult("Create event bag", compCode, reason);
    if (compCode != MQCC_OK)
        return;

    /*****
    /* Open the event queue chosen by the user */
    /*****
    strncpy(od.ObjectName, qName, (size_t)MQ_Q_NAME_LENGTH);
    MQOPEN(hConn, &od, MQOO_INPUT_AS_Q_DEF+MQOO_FAIL_IF_QUIESCING, &eventQueue,
        &compCode, &openReason);
    CheckCallResult("Open event queue", compCode, openReason);

    /*****
    /* Set the GMO options to control the action of the get message from the */
    /* queue. */
    /*****
    gmo.WaitInterval = 30000; /* 30 second wait for message */
    gmo.Options = MQGMO_WAIT + MQGMO_FAIL_IF_QUIESCING + MQGMO_CONVERT;
    gmo.Version = MQGMO_VERSION_2; /* Avoid need to reset Message ID */
    gmo.MatchOptions = MQMO_NONE; /* and Correlation ID after every */
    /* mqGetBag */

    /*****
    /* If open fails, we cannot access the queue and must stop the monitor. */
    /*****
    if (compCode != MQCC_OK)
        bQueueOK = 0;

    /*****
    /* Main loop to get an event message when it arrives */
    /*****
    while (bQueueOK)
    {
        printf("\nWaiting for an event\n");

        /*****
        /* Get the message from the event queue and convert it into the event */
        /* bag. */
        /*****
        mqGetBag(hConn, eventQueue, &md, &gmo, eventBag, &compCode, &reason);

        /*****
        /* If get fails, we cannot access the queue and must stop the monitor. */
        /*****
        if (compCode != MQCC_OK)
        {
            bQueueOK = 0;

            /*****
            /* If get fails because no message available then we have timed out, */
            /* so report this, otherwise report an error. */
            /*****

```

```

        if (reason == MQRC_NO_MSG_AVAILABLE)
        {
            printf("No more messages\n");
        }
        else
        {
            CheckCallResult("Get bag", compCode, reason);
        }
    }

    /*****
    /* Event message read - Print the contents of the event bag */
    /*****
else
{
    if ( PrintBag(eventBag) )
        printf("\nError found while printing bag contents\n");

    } /* end of msg found */
} /* end of main loop */
/*****
/* Close the event queue if successfully opened */
/*****
if (openReason == MQRC_NONE)
{
    MQCLOSE(hConn, &eventQueue, MQCO_NONE, &compCode, &reason);
    CheckCallResult("Close event queue", compCode, reason);
}

/*****
/* Delete the event bag if successfully created. */
/*****
if (eventBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&eventBag, &compCode, &reason);
    CheckCallResult("Delete the event bag", compCode, reason);
}

} /* end of GetQEvents */

/*****
/*
/* Function: PrintBag */
/*
/*****
/*
/* Input Parameters: Bag Handle */
/*
/* Output Parameters: None */
/*
/* Returns: Number of errors found */
/*
/* Logic: Calls PrintBagContents to display the contents of the bag. */
/*
/*****

int PrintBag(MQHBAG dataBag)
{
    int errors;

    printf("\n");
    errors = PrintBagContents(dataBag, 0);
    printf("\n");

    return errors;
}

/*****
/*
/* Function: PrintBagContents */
/*
/*****
/*
/* Input Parameters: Bag Handle */
/*
/* Indentation level of bag */
/*
/* Output Parameters: None */
/*
/* Returns: Number of errors found */
/*
/* Logic: Count the number of items in the bag */
/*
/* Obtain selector and item type for each item in the bag. */

```

```

/*      Obtain the value of the item depending on item type and display the */
/*      index of the item, the selector and the value.                        */
/*      If the item is an embedded bag handle then call this function again */
/*      to print the contents of the embedded bag increasing the           */
/*      indentation level.                                                 */
/*                                                                           */
/*      */
/*****
int PrintBagContents(MQHBAG dataBag, int indent)
{
    /*****
    /* Definitions
    /*****
    #define LENGTH 500                /* Max length of string to be read*/
    #define INDENT 4                  /* Number of spaces to indent  */
    /* embedded bag display          */

    /*****
    /* Variables
    /*****
    MQLONG  itemCount;                /* Number of items in the bag  */
    MQLONG  itemType;                /* Type of the item            */
    int     i;                        /* Index of item in the bag    */
    MQCHAR  stringVal[LENGTH+1];     /* Value if item is a string   */
    MQBYTE  byteStringVal[LENGTH];   /* Value if item is a byte string */
    MQLONG  stringLength;            /* Length of string value      */
    MQLONG  ccsid;                   /* CCSID of string value       */
    MQINT32 iValue;                  /* Value if item is an integer */
    MQINT64 i64Value;                /* Value if item is a 64-bit   */
    /* integer                        */
    MQLONG  selector;                /* Selector of item            */
    MQHBAG  bagHandle;               /* Value if item is a bag handle */
    MQLONG  reason;                  /* reason code                 */
    MQLONG  compCode;                /* completion code             */
    MQLONG  trimLength;              /* Length of string to be trimmed */
    int     errors = 0;               /* Count of errors found       */
    char    blanks[] = "            "; /* Blank string used to       */
    /* indent display                 */

    /*****
    /* Count the number of items in the bag
    /*****
    mqCountItems(dataBag, MQSEL_ALL_SELECTORS, &iItemCount, &compCode, &reason);

    if (compCode != MQCC_OK)
        errors++;
    else
    {
        printf("
        printf("
        printf("
    }

    /*****
    /* If no errors found, display each item in the bag
    /*****
    if (!errors)
    {
        for (i = 0; i < itemCount; i++)
        {

            /*****
            /* First inquire the type of the item for each item in the bag
            /*****
            mqInquireItemInfo(dataBag,                /* Bag handle
                               MQSEL_ANY_SELECTOR,    /* Item can have any selector*/
                               i,                    /* Index position in the bag */
                               &selector,           /* Actual value of selector  */
                               /* returned by call   */
                               &itemType,           /* Actual type of item       */
                               /* returned by call   */
                               &compCode,           /* Completion code           */
                               &reason);           /* Reason Code               */

            if (compCode != MQCC_OK)
                errors++;

            switch(itemType)
            {
            case MQITEM_INTEGER:
                /*****
                /* Item is an integer. Find its value and display its index,

```

```

/* selector and value. */
/******/
mqInquireInteger(dataBag, /* Bag handle */
                 MQSEL_ANY_SELECTOR, /* Allow any selector */
                 i, /* Index position in the bag */
                 &iValue, /* Returned integer value */
                 &compCode, /* Completion code */
                 &reason); /* Reason Code */

if (compCode != MQCC_OK)
    errors++;
else
    printf("%.s %-2d %-4d (%d)\n",
           indent, blanks, i, selector, iValue);
break

case MQITEM_INTEGER64:
/******/
/* Item is a 64-bit integer. Find its value and display its */
/* index, selector and value. */
/******/
mqInquireInteger64(dataBag, /* Bag handle */
                  MQSEL_ANY_SELECTOR, /* Allow any selector */
                  i, /* Index position in the bag */
                  &i64Value, /* Returned integer value */
                  &compCode, /* Completion code */
                  &reason); /* Reason Code */

if (compCode != MQCC_OK)
    errors++;
else
    printf("%.s %-2d %-4d (%"Int64"d)\n",
           indent, blanks, i, selector, i64Value);
break;

case MQITEM_STRING:
/******/
/* Item is a string. Obtain the string in a buffer, prepare */
/* the string for displaying and display the index, selector, */
/* string and Character Set ID. */
/******/
mqInquireString(dataBag, /* Bag handle */
               MQSEL_ANY_SELECTOR, /* Allow any selector */
               i, /* Index position in the bag */
               LENGTH, /* Maximum length of buffer */
               stringVal, /* Buffer to receive string */
               &stringLength, /* Actual length of string */
               &ccsid, /* Coded character set ID */
               &compCode, /* Completion code */
               &reason); /* Reason Code */

/******/
/* The call can return a warning if the string is too long for */
/* the output buffer and has been truncated, so only check */
/* explicitly for call failure. */
/******/
if (compCode == MQCC_FAILED)
    errors++;
else
    {
        /******/
        /* Remove trailing blanks from the string and terminate with */
        /* a null. First check that the string should not have been */
        /* longer than the maximum buffer size allowed. */
        /******/
        if (stringLength > LENGTH)
            trimLength = LENGTH;
        else
            trimLength = stringLength;
        mqTrim(trimLength, stringVal, stringVal, &compCode, &reason);
        printf("%.s %-2d %-4d '%s' %d\n",
               indent, blanks, i, selector, stringVal, ccsid);
    }
break;

case MQITEM_BYTE_STRING:
/******/
/* Item is a byte string. Obtain the byte string in a buffer, */
/* prepare the byte string for displaying and display the */
/* index, selector and string. */
/******/

```

```

mqInquireByteString(dataBag,      /* Bag handle      */
                    MQSEL_ANY_SELECTOR, /* Allow any selector */
                    i,           /* Index position in the bag */
                    LENGTH,     /* Maximum length of buffer */
                    byteStringVal, /* Buffer to receive string */
                    &stringLength, /* Actual length of string */
                    &compCode,  /* Completion code */
                    &reason);   /* Reason Code */

/*****
/* The call can return a warning if the string is too long for */
/* the output buffer and has been truncated, so only check */
/* explicitly for call failure. */
*****/
if (compCode == MQCC_FAILED)
    errors++;
else
{
    printf("%.s %-2d %-4d X'",
           indent, blanks, i, selector);

    for (i = 0 ; i < stringLength ; i++)
        printf("

    printf("\n");
}
break;

case MQITEM_BAG:
/*****
/* Item is an embedded bag handle, so call the PrintBagContents*/
/* function again to display the contents. */
*****/
mqInquireBag(dataBag,      /* Bag handle      */
             MQSEL_ANY_SELECTOR, /* Allow any selector */
             i,           /* Index position in the bag */
             &bagHandle, /* Returned embedded bag handle */
             &compCode,  /* Completion code */
             &reason);   /* Reason Code */

if (compCode != MQCC_OK)
    errors++;
else
{
    printf("%.s %-2d %-4d (%d)\n", indent, blanks, i,
           selector, bagHandle);
    if (selector == MQHA_BAG_HANDLE)
        printf("
    else
        printf("
        PrintBagContents(bagHandle, indent+INDENT);
}
break;

default:
    printf("
}
}
}
return errors;
}

```

用于查询通道对象的样本 C 程序 (amqsaicl.c)

样本 C 程序 amqsaicl.c 使用 MQAI 查询通道对象。

```

/*****
/*
/* Program name: AMQSAICL.C
/*
/* Description: Sample C program to inquire channel objects
/*              using the IBM MQ Administration Interface (MQAI)
/*
/* <N_OCO_COPYRIGHT>
/* Licensed Materials - Property of IBM
/*
/* 63H9336
/* (c) Copyright IBM Corp. 2008, 2023. All Rights Reserved.
*/

```

```

/*
/* US Government Users Restricted Rights - Use, duplication or
/* disclosure restricted by GSA ADP Schedule Contract with
/* IBM Corp.
/* <NOC_COPYRIGHT>
/*****
/*
/* Function:
/* AMQSAICL is a sample C program that demonstrates how to inquire
/* attributes of the local queue manager using the MQAI interface. In
/* particular, it inquires all channels and their types.
/*
/* - A PCF command is built from items placed into an MQAI administration
/* bag.
/* These are:-
/* - The generic channel name "*"
/* - The attributes to be inquired. In this sample we just want
/* name and type attributes
/*
/* - The mqExecute MQCMD_INQUIRE_CHANNEL call is executed.
/* The call generates the correct PCF structure.
/* The default options to the call are used so that the command is sent
/* to the SYSTEM.ADMIN.COMMAND.QUEUE.
/* The reply from the command server is placed on a temporary dynamic
/* queue.
/* The reply from the MQCMD_INQUIRE_CHANNEL is read from the
/* temporary queue and formatted into the response bag.
/*
/* - The completion code from the mqExecute call is checked and if there
/* is a failure from the command server, then the code returned by the
/* command server is retrieved from the system bag that has been
/* embedded in the response bag to the mqExecute call.
/*
/* Note: The command server must be running.
/*
/*****
/*
/* AMQSAICL has 2 parameter - the queue manager name (optional)
/* - output file (optional) default varies
/*****

/*****
/* Includes
/*****
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#if (MQAT_DEFAULT == MQAT_OS400)
#include <recio.h>
#endif

#include <cmqc.h> /* MQI
#include <cmqcfh.h> /* PCF
#include <cmqbc.h> /* MQAI
#include <cmqxc.h> /* MQCD

/*****
/* Function prototypes
/*****
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/*****
/* DataTypes
/*****
#if (MQAT_DEFAULT == MQAT_OS400)
typedef _RFILE OUTFILEHDL;
#else
typedef FILE OUTFILEHDL;
#endif

/*****
/* Constants
/*****
#if (MQAT_DEFAULT == MQAT_OS400)
const struct
{
char name[9];
} ChlTypeMap[9] =
{
" *SDR ", /* MQCHT_SENDER
" *SVR ", /* MQCHT_SERVER

```

```

"RCVR " , /* MQCHT_RECEIVER */
"RQSTR " , /* MQCHT_REQUESTER */
"ALL " , /* MQCHT_ALL */
"CLTCN " , /* MQCHT_CLNTCONN */
"SVRCONN " , /* MQCHT_SVRCONN */
"CLUSRCVR " , /* MQCHT_CLUSRCVR */
"CLUSSDR " , /* MQCHT_CLUSSDR */
};
#else
const struct
{
char name[9];
} ChlTypeMap[9] =
{
"sdr " , /* MQCHT_SENDER */
"svr " , /* MQCHT_SERVER */
"rcvr " , /* MQCHT_RECEIVER */
"rqstr " , /* MQCHT_REQUESTER */
"all " , /* MQCHT_ALL */
"cltconn " , /* MQCHT_CLNTCONN */
"svrcn " , /* MQCHT_SVRCONN */
"clusrcvr " , /* MQCHT_CLUSRCVR */
"clussdr " , /* MQCHT_CLUSSDR */
};
#endif

/*****
/* Macros */
/*****
#if (MQAT_DEFAULT == MQAT_OS400)
#define OUTFILE "QTEMP/AMQSAICL(AMQSAICL)"
#define OPENOUTFILE(hdl, fname) \
(hdl) = _Ropen((fname), "wr, rtncode=Y");
#define CLOSEOUTFILE(hdl) \
_Rclose((hdl));
#define WRITEOUTFILE(hdl, buf, buflen) \
_Rwrite((hdl), (buf), (buflen));

#elif (MQAT_DEFAULT == MQAT_UNIX)
#define OUTFILE "/tmp/amqsaicl.txt"
#define OPENOUTFILE(hdl, fname) \
(hdl) = fopen((fname), "w");
#define CLOSEOUTFILE(hdl) \
fclose((hdl));
#define WRITEOUTFILE(hdl, buf, buflen) \
fwrite((buf), (buflen), 1, (hdl)); fflush((hdl));

#else
#define OUTFILE "amqsaicl.txt"
#define OPENOUTFILE(fname) \
fopen((fname), "w");
#define CLOSEOUTFILE(hdl) \
fclose((hdl));
#define WRITEOUTFILE(hdl, buf, buflen) \
fwrite((buf), (buflen), 1, (hdl)); fflush((hdl));

#endif

#define ChlType2String(t) ChlTypeMap[(t)-1].name

/*****
/* Function: main */
/*****
int main(int argc, char *argv[])
{
/*****
/* MQAI variables */
/*****
MQHCONN hConn; /* handle to MQ connection */
MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name */
MQLONG reason; /* reason code */
MQLONG connReason; /* MQCONN reason code */
MQLONG compCode; /* completion code */
MQHBAG adminBag = MQHB_UNUSABLE_HBAG; /* admin bag for mqExecute */
MQHBAG responseBag = MQHB_UNUSABLE_HBAG; /* response bag for mqExecute */
MQHBAG cAttrsBag; /* bag containing chl attributes */
MQHBAG errorBag; /* bag containing cmd server error */
MQLONG mqExecuteCC; /* mqExecute completion code */
MQLONG mqExecuteRC; /* mqExecute reason code */
MQLONG chlNameLength; /* Actual length of chl name */
MQLONG chlType; /* Channel type */
MQLONG i; /* loop counter */

```

```

MQLONG numberOfBags; /* number of bags in response bag */
MQCHAR chlName[MQ_OBJECT_NAME_LENGTH+1]; /* name of chl extracted from bag */
MQCHAR OutputBuffer[100]; /* output data buffer */
OUTFILEHDL *outfp = NULL; /* output file handle */

/*****
/* Connect to the queue manager */
/*****
if (argc > 1)
    strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
MQCONN(qmName, &hConn, &compCode, &connReason);

/*****
/* Report the reason and stop if the connection failed. */
/*****
if (compCode == MQCC_FAILED)
{
    CheckCallResult("Queue Manager connection", compCode, connReason);
    exit( (int)connReason);
}

/*****
/* Open the output file */
/*****
if (argc > 2)
{
    OPENOUTFILE(outfp, argv[2]);
}
else
{
    OPENOUTFILE(outfp, OUTFILE);
}

if(outfp == NULL)
{
    printf("Could not open output file.\n");
    goto MOD_EXIT;
}

/*****
/* Create an admin bag for the mqExecute call */
/*****
mqCreateBag(MQCBO_ADMIN_BAG, &adminBag, &compCode, &reason);
CheckCallResult("Create admin bag", compCode, reason);

/*****
/* Create a response bag for the mqExecute call */
/*****
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
CheckCallResult("Create response bag", compCode, reason);

/*****
/* Put the generic channel name into the admin bag */
/*****
mqAddString(adminBag, MQCACH_CHANNEL_NAME, MQBL_NULL_TERMINATED, "*",
            &compCode, &reason);
CheckCallResult("Add channel name", compCode, reason);

/*****
/* Put the channel type into the admin bag */
/*****
mqAddInteger(adminBag, MQIACH_CHANNEL_TYPE, MQCHT_ALL, &compCode, &reason);
CheckCallResult("Add channel type", compCode, reason);

/*****
/* Add an inquiry for various attributes */
/*****
mqAddInquiry(adminBag, MQIACH_CHANNEL_TYPE, &compCode, &reason);
CheckCallResult("Add inquiry", compCode, reason);

/*****
/* Send the command to find all the channel names and channel types. */
/* The mqExecute call creates the PCF structure required, sends it to */
/* the command server, and receives the reply from the command server into */
/* the response bag. The attributes are contained in system bags that are */
/* embedded in the response bag, one set of attributes per bag. */
/*****
mqExecute(hConn, /* MQ connection handle */
          MQCMD_INQUIRE_CHANNEL, /* Command to be executed */
          MQHB_NONE, /* No options bag */
          adminBag, /* Handle to bag containing commands */
          responseBag, /* Handle to bag to receive the response */
          MQHO_NONE, /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE */

```



```

                MQHO_NONE,          /* Create a dynamic q for the response */
                &compCode;,        /* Completion code from the mqexecute */
                &reason;);        /* Reason code from mqexecute call */

/*****
/* Check the command server is started. If not exit.
*****/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
    printf("Please start the command server: <strmqcsv QMgrName="">\n");
    goto MOD_EXIT;
}

/*****
/* Check the result from mqExecute call. If successful find the channel
/* types for all the channels. If failed find the error.
*****/
if ( compCode == MQCC_OK )          /* Successful mqExecute */
{
    /*****
    /* Count the number of system bags embedded in the response bag from the
    /* mqExecute call. The attributes for each channel are in separate bags.
    *****/
    mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags;,
                &compCode;, &reason;);
    CheckCallResult("Count number of bag handles", compCode, reason);

    for ( i=0; i<numberOfBags; i++)
    {
        /*****
        /* Get the next system bag handle out of the mqExecute response bag.
        /* This bag contains the channel attributes
        *****/
        mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &cAttrsBag,
                    &compCode, &reason);
        CheckCallResult("Get the result bag handle", compCode, reason);

        /*****
        /* Get the channel name out of the channel attributes bag
        *****/
        mqInquireString(cAttrsBag, MQCACH_CHANNEL_NAME, 0, MQ_OBJECT_NAME_LENGTH,
                        chlName, &chlNameLength, NULL, &compCode, &reason);
        CheckCallResult("Get channel name", compCode, reason);

        /*****
        /* Get the channel type out of the channel attributes bag
        *****/
        mqInquireInteger(cAttrsBag, MQIACH_CHANNEL_TYPE, MQIND_NONE, &chlType,
                        &compCode, &reason);
        CheckCallResult("Get type", compCode, reason);

        /*****
        /* Use mqTrim to prepare the channel name for printing.
        /* Print the result.
        *****/
        mqTrim(MQ_CHANNEL_NAME_LENGTH, chlName, &compCode, &reason);
        sprintf(OutputBuffer, "%-20s%-9s", chlName, ChlType2String(chlType));
        WRITEOUTFILE(outfp, OutputBuffer, 29)
    }
}

else
{
    /*****
    /* Failed mqExecute
    *****/
    printf("Call to get channel attributes failed: Cc = %ld : Rc = %ld\n",
           compCode, reason);
    /*****
    /* If the command fails get the system bag handle out of the mqexecute
    /* response bag. This bag contains the reason from the command server
    /* why the command failed.
    *****/
    if (reason == MQRCCF_COMMAND_FAILED)
    {
        mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag,
                    &compCode, &reason);
        CheckCallResult("Get the result bag handle", compCode, reason);

        /*****
        /* Get the completion code and reason code, returned by the command
        /* server, from the embedded error bag.
        *****/
        mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,

```

```

        &compCode, &reason );
    CheckCallResult("Get the completion code from the result bag",
        compCode, reason);
    mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
        &compCode, &reason);
    CheckCallResult("Get the reason code from the result bag",
        compCode, reason);
    printf("Error returned by the command server: Cc = %ld : Rc = %ld\n",
        mqExecuteCC, mqExecuteRC);
}
}

MOD_EXIT:
/*****
/* Delete the admin bag if successfully created. */
*****/
if (adminBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&adminBag, &compCode, &reason);
    CheckCallResult("Delete the admin bag", compCode, reason);
}

/*****
/* Delete the response bag if successfully created. */
*****/
if (responseBag != MQHB_UNUSABLE_HBAG)
{
    mqDeleteBag(&responseBag, &compCode, &reason);
    CheckCallResult("Delete the response bag", compCode, reason);
}

/*****
/* Disconnect from the queue manager if not already connected */
*****/
if (connReason != MQRC_ALREADY_CONNECTED)
{
    MQDISC(&hConn, &compCode, &reason);
    CheckCallResult("Disconnect from Queue Manager", compCode, reason);
}

/*****
/* Close the output file if open */
*****/
if(outfp != NULL)
    CLOSEOUTFILE(outfp);

return 0;
}

/*****
/*
/* Function: CheckCallResult */
/*
*****/
/*
/* Input Parameters:  Description of call */
/*                   Completion code */
/*                   Reason code */
/*
/* Output Parameters: None */
/*
/* Logic: Display the description of the call, the completion code and the */
/*        reason code if the completion code is not successful */
/*
*****/
void CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
    if (cc != MQCC_OK)
        printf("%s failed: Completion Code = %ld : Reason = %ld\n", callText,
            cc, rc);
}
}

```

用于查询队列和打印信息的样本 C 程序 (amqsailq.c)

样本 C 程序 amqsailq.c 使用 MQAI 查询本地队列的当前深度。

```

/*****

```

```

/*
/* Program name: AMQSAILQ.C
/*
/* Description: Sample C program to inquire the current depth of the local
/* queues using the IBM MQ Administration Interface (MQAI)
/*
/* Statement: Licensed Materials - Property of IBM
/*
/* 84H2000, 5765-B73
/* 84H2001, 5639-B42
/* 84H2002, 5765-B74
/* 84H2003, 5765-B75
/* 84H2004, 5639-B43
/*
/* (C) Copyright IBM Corp. 1999, 2023.
/*
/*****
/*
/* Function:
/* AMQSAILQ is a sample C program that demonstrates how to inquire
/* attributes of the local queue manager using the MQAI interface. In
/* particular, it inquires the current depths of all the local queues.
/*
/* - A PCF command is built by placing items into an MQAI administration
/* bag.
/* These are:-
/* - The generic queue name "*"
/* - The type of queue required. In this sample we want to
/* inquire local queues.
/* - The attribute to be inquired. In this sample we want the
/* current depths.
/*
/* - The mqExecute call is executed with the command MQCMD_INQUIRE_Q.
/* The call generates the correct PCF structure.
/* The default options to the call are used so that the command is sent
/* to the SYSTEM.ADMIN.COMMAND.QUEUE.
/* The reply from the command server is placed on a temporary dynamic
/* queue.
/* The reply from the MQCMD_INQUIRE_Q command is read from the
/* temporary queue and formatted into the response bag.
/*
/* - The completion code from the mqExecute call is checked and if there
/* is a failure from the command server, then the code returned by
/* command server is retrieved from the system bag that has been
/* embedded in the response bag to the mqExecute call.
/*
/* - If the call is successful, the depth of each local queue is placed
/* in system bags embedded in the response bag of the mqExecute call.
/* The name and depth of each queue is obtained from each of the bags
/* and the result displayed on the screen.
/*
/* Note: The command server must be running.
/*
/*****
/*
/* AMQSAILQ has 1 parameter - the queue manager name (optional)
/*
/*****

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

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

/*****
/* Function prototypes
/*****
void CheckCallResult(MQCHAR *, MQLONG , MQLONG);

/*****
/* Function: main
/*****
int main(int argc, char *argv[])
{
/*****

```

```

/* MQAI variables */
/*****
MQHCONN hConn; /* handle to IBM MQ connection */
MQCHAR qmName[MQ_Q_MGR_NAME_LENGTH+1]=""; /* default QMgr name */
MQLONG reason; /* reason code */
MQLONG connReason; /* MQCONN reason code */
MQLONG compCode; /* completion code */
MQHBAG adminBag = MQHB_UNUSABLE_HBAG; /* admin bag for mqExecute */
MQHBAG responseBag = MQHB_UNUSABLE_HBAG; /* response bag for mqExecute */
MQHBAG qAttrsBag; /* bag containing q attributes */
MQHBAG errorBag; /* bag containing cmd server error */
MQLONG mqExecuteCC; /* mqExecute completion code */
MQLONG mqExecuteRC; /* mqExecute reason code */
MQLONG qNameLength; /* Actual length of q name */
MQLONG qDepth; /* depth of queue */
MQLONG i; /* loop counter */
MQLONG numberOfBags; /* number of bags in response bag */
MQCHAR qName[MQ_Q_NAME_LENGTH+1]; /* name of queue extracted from bag*/
*/

printf("Display current depths of local queues\n\n");

/*****
/* Connect to the queue manager */
/*****
if (argc > 1)
    strncpy(qmName, argv[1], (size_t)MQ_Q_MGR_NAME_LENGTH);
MQCONN(qmName, &hConn, &compCode, &connReason);

/*****
/* Report the reason and stop if the connection failed. */
/*****
if (compCode == MQCC_FAILED)
{
    CheckCallResult("Queue Manager connection", compCode, connReason);
    exit( (int)connReason);
}

/*****
/* Create an admin bag for the mqExecute call */
/*****
mqCreateBag(MQCBO_ADMIN_BAG, &adminBag, &compCode, &reason);
CheckCallResult("Create admin bag", compCode, reason);
/*****
/* Create a response bag for the mqExecute call */
/*****
mqCreateBag(MQCBO_ADMIN_BAG, &responseBag, &compCode, &reason);
CheckCallResult("Create response bag", compCode, reason);

/*****
/* Put the generic queue name into the admin bag */
/*****
mqAddString(adminBag, MQCA_Q_NAME, MQBL_NULL_TERMINATED, "*",
            &compCode, &reason);
CheckCallResult("Add q name", compCode, reason);

/*****
/* Put the local queue type into the admin bag */
/*****
mqAddInteger(adminBag, MQIA_Q_TYPE, MQQT_LOCAL, &compCode, &reason);
CheckCallResult("Add q type", compCode, reason);

/*****
/* Add an inquiry for current queue depths */
/*****
mqAddInquiry(adminBag, MQIA_CURRENT_Q_DEPTH, &compCode, &reason);
CheckCallResult("Add inquiry", compCode, reason);

/*****
/* Send the command to find all the local queue names and queue depths. */
/* The mqExecute call creates the PCF structure required, sends it to
/* the command server, and receives the reply from the command server into
/* the response bag. The attributes are contained in system bags that are
/* embedded in the response bag, one set of attributes per bag.
/*****
mqExecute(hConn, /* IBM MQ connection handle */
          MQCMD_INQUIRE_Q, /* Command to be executed */
          MQHB_NONE, /* No options bag */
          adminBag, /* Handle to bag containing commands */
          responseBag, /* Handle to bag to receive the response*/
          MQHO_NONE, /* Put msg on SYSTEM.ADMIN.COMMAND.QUEUE*/
          MQHO_NONE, /* Create a dynamic q for the response */

```

```

        &compCode,          /* Completion code from the mqExecute */
        &reason);          /* Reason code from mqExecute call */

/*****
/* Check the command server is started. If not exit. */
*****/
if (reason == MQRC_CMD_SERVER_NOT_AVAILABLE)
{
    printf("Please start the command server: <strmqcsv QMgrName>\n");
    MQDISC(&hConn, &compCode, &reason);
    CheckCallResult("Disconnect from Queue Manager", compCode, reason);
    exit(98);
}

/*****
/* Check the result from mqExecute call. If successful find the current */
/* depths of all the local queues. If failed find the error. */
*****/
if ( compCode == MQCC_OK )          /* Successful mqExecute */
{
    /*****
    /* Count the number of system bags embedded in the response bag from the */
    /* mqExecute call. The attributes for each queue are in a separate bag. */
    *****/
    mqCountItems(responseBag, MQHA_BAG_HANDLE, &numberOfBags, &compCode,
        &reason);
    CheckCallResult("Count number of bag handles", compCode, reason);

    for ( i=0; i<numberOfBags; i++)
    {
        /*****
        /* Get the next system bag handle out of the mqExecute response bag. */
        /* This bag contains the queue attributes */
        *****/
        mqInquireBag(responseBag, MQHA_BAG_HANDLE, i, &qAttrsBag, &compCode,
            &reason);
        CheckCallResult("Get the result bag handle", compCode, reason);

        /*****
        /* Get the queue name out of the queue attributes bag */
        *****/
        mqInquireString(qAttrsBag, MQCA_Q_NAME, 0, MQ_Q_NAME_LENGTH, qName,
            &qNameLength, NULL, &compCode, &reason);
        CheckCallResult("Get queue name", compCode, reason);

        /*****
        /* Get the depth out of the queue attributes bag */
        *****/
        mqInquireInteger(qAttrsBag, MQIA_CURRENT_Q_DEPTH, MQIND_NONE, &qDepth,
            &compCode, &reason);
        CheckCallResult("Get depth", compCode, reason);

        /*****
        /* Use mqTrim to prepare the queue name for printing. */
        /* Print the result. */
        *****/
        mqTrim(MQ_Q_NAME_LENGTH, qName, qName, &compCode, &reason)
        printf("%4d %-48s\n", qDepth, qName);
    }
}

else          /* Failed mqExecute */
{
    printf("Call to get queue attributes failed: Completion Code = %d :
        Reason = %d\n", compCode, reason);

    /*****
    /* If the command fails get the system bag handle out of the mqExecute */
    /* response bag. This bag contains the reason from the command server */
    /* why the command failed. */
    *****/
    if (reason == MQRCCF_COMMAND_FAILED)
    {
        mqInquireBag(responseBag, MQHA_BAG_HANDLE, 0, &errorBag, &compCode,
            &reason);
        CheckCallResult("Get the result bag handle", compCode, reason);

        /*****
        /* Get the completion code and reason code, returned by the command */
        /* server, from the embedded error bag. */
        *****/

```

```

mqInquireInteger(errorBag, MQIASY_COMP_CODE, MQIND_NONE, &mqExecuteCC,
                &compCode, &reason);
CheckCallResult("Get the completion code from the result bag",
                compCode, reason);
mqInquireInteger(errorBag, MQIASY_REASON, MQIND_NONE, &mqExecuteRC,
                &compCode, &reason);
CheckCallResult("Get the reason code from the result bag",
                compCode, reason);
printf("Error returned by the command server: Completion Code = %d :
        Reason = %d\n", mqExecuteCC, mqExecuteRC);
}
}

/*****
/* Delete the admin bag if successfully created.          */
*****/
if (adminBag != MQHB_UNUSABLE_HBAG)
{
mqDeleteBag(&adminBag, &compCode, &reason);
CheckCallResult("Delete the admin bag", compCode, reason);
}

/*****
/* Delete the response bag if successfully created.      */
*****/
if (responseBag != MQHB_UNUSABLE_HBAG)
{
mqDeleteBag(&responseBag, &compCode, &reason);
CheckCallResult("Delete the response bag", compCode, reason);
}

/*****
/* Disconnect from the queue manager if not already connected */
*****/
if (connReason != MQRC_ALREADY_CONNECTED)
{
MQDISC(&hConn, &compCode, &reason);
CheckCallResult("Disconnect from queue manager", compCode, reason);
}
return 0;
}

*****/
* Function: CheckCallResult                                */
*                                                         */
* Input Parameters:  Description of call                    */
*                   Completion code                        */
*                   Reason code                            */
* Output Parameters: None                                  */
* Logic: Display the description of the call, the completion code and the */
*        reason code if the completion code is not successful */
*                                                         */
*****/
void CheckCallResult(char *callText, MQLONG cc, MQLONG rc)
{
if (cc != MQCC_OK)
printf("%s failed: Completion Code = %d : Reason = %d\n",
        callText, cc, rc);
}

```

数据包和 MQAI

数据包是使用 IBM MQ 管理接口 (MQAI) 来处理对象的属性或参数的一种方法。

数据包

- 数据包包含零个或多个数据项。这些数据项在放入包中时在包中进行排序。这称为插入顺序。每个数据项都包含一个选择器，用于标识数据项以及该数据项的值，该数据项可以是整数，64 位整数，整数过滤器，字符串，字符串过滤器，字节字符串，字节字符串过滤器或另一个包的句柄。第 56 页的『MQAI 中可用的数据项类型』中的详细信息描述了数据项

有两种类型的选择器: 用户选择器和系统选择器。这些在 MQAI 选择器中进行了描述。选择器通常是唯一的, 但对于同一个选择器可以有多个值。在这种情况下, 索引标识所需的特定选择器实例。第 31 页的『在 MQAI 中建立索引』中描述了索引。

图 1 中显示了这些概念的层次结构。

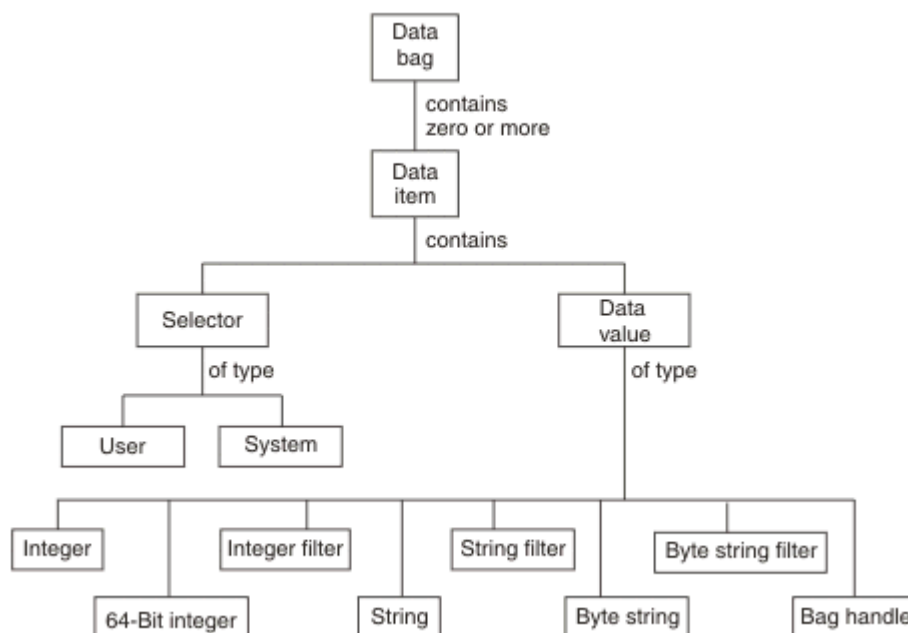


图 5: MQAI 概念的层次结构

在上一段中对层次结构进行了说明。

数据包类型

您可以根据要执行的任务来选择要创建的数据包类型:

用户包 (user bag)

用于用户数据的简单包。

管理包 (administration bag)

为用于通过向命令服务器发送管理消息来管理 IBM MQ 对象的数据创建的包。管理包自动暗示某些选项, 如第 56 页的『创建和删除数据包』中所述。

命令包 (command bag)

还为用于管理 IBM MQ 对象的命令创建了一个包。但是, 与管理包不同, 命令包不会自动暗示某些选项, 尽管这些选项可用。有关选项的更多信息, 请参阅第 56 页的『创建和删除数据包』。

组包

用于存放一组分组数据项的包。组包不能用于管理 IBM MQ 对象。

此外, 当从命令服务器返回应答消息并将其放入用户的输出包中时, MQAI 会创建 **系统包**。用户无法修改系统包。

使用数据包的不同使用方式在本主题中列出:

使用数据包

以下列表中显示了使用数据包的不同方法:

- 您可以创建和删除数据包 第 56 页的『创建和删除数据包』。
- 您可以使用数据包 第 56 页的『使用 MQAI 放置和接收数据包』在应用程序之间发送数据。
- 您可以将数据项添加到数据包 第 57 页的『使用 MQAI 将数据项添加到包』。

- 您可以在数据包 第 58 页的『[向包添加查询命令](#)』中添加查询命令。
- 您可以在数据包 第 59 页的『[在数据包中查询](#)』中进行查询。
- 您可以对数据包 第 61 页的『[对数据项进行计数](#)』中的数据项进行计数。
- 您可以在数据包 第 59 页的『[更改包中的信息](#)』中更改信息。
- 您可以清除数据包 第 60 页的『[使用 mqClearBag 调用清除包](#)』。
- 您可以截断数据包 第 60 页的『[使用 mqTruncateBag 调用截断包](#)』。
- 您可以转换包和缓冲区 第 60 页的『[转换包和缓冲区](#)』。

创建和删除数据包

创建数据包

要使用 MQAI，请首先使用 mqCreateBag 调用创建数据包。作为此调用的输入，您提供一个或多个选项来控制包的创建。

MQCreateBag 调用的 **Options** 参数允许您选择是创建用户包，命令包，组包还是管理包。

要创建用户包，命令包或组包，您可以选择一个或多个进一步的选项以：

- 当包中出现两个或更多个相邻的相同选择器时，请使用列表表单。
- 将数据项添加到 PCF 消息时对其进行重新排序，以确保参数的顺序正确。有关数据项的更多信息，请参阅 第 56 页的『[MQAI 中可用的数据项类型](#)』。
- 检查您添加到包中的项的用户选择器值。

管理包会自动暗示这些选项。

数据袋由其手柄标识。包句柄是从 mqCreate 包返回的，必须在使用该数据包的所有其他调用上提供。

有关 mqCreateBag 调用的完整描述，请参阅 [mqCreateBag](#)。

删除数据包

还必须使用 mqDeleteBag 调用删除用户创建的任何数据包。例如，如果在用户代码中创建了包，那么还必须在用户代码中删除该包。

系统包由 MQAI 自动创建和删除。有关此操作的更多信息，请参阅 第 62 页的『[使用 mqExecute 调用将管理命令发送到 qm 命令服务器](#)』。用户代码无法删除系统包。

有关 mqDeleteBag 调用的完整描述，请参阅 [mqDeleteBag](#)。

使用 MQAI 放置和接收数据包

还可以通过使用 mqPutBag 和 mqGetBag 调用来放置和获取数据包，在应用程序之间发送数据。这使 IBM MQ 管理接口 (MQAI) 能够处理缓冲区而不是应用程序。

mqPutBag 调用会将指定包的内容转换为 PCF 消息，并将该消息发送到指定队列，而 mqGetBag 调用会从指定队列中除去该消息并将其转换回数据包。因此，mqPutBag 调用等效于后跟 MQPUT 的 mqBagToBuffer 调用，而 mqGetBag 等效于后跟 mqBufferToBag 的 MQGET 调用。

有关在特定队列中发送和接收 PCF 消息的更多信息，请参阅 第 21 页的『[在指定队列中发送和接收 PCF 消息](#)』

注：如果选择使用 mqGetBag 调用，那么消息中的 PCF 详细信息必须正确；如果不正确，那么将不会返回相应的错误结果和 PCF 消息。

MQAI 中可用的数据项类型

IBM MQ 管理接口 (MQAI) 在创建数据项时使用这些数据项来填充数据包。这些数据项可以是用户或系统项。

这些用户项包含用户数据，例如要管理的对象的属性。系统项应用于对生成的消息进行更多控制：例如，生成消息头。有关系统项的更多信息，请参阅 [第 57 页的『系统项和 MQAI』](#)。

数据项的类型

创建数据包后，可以使用整数或字符串项填充该数据包。您可以查询所有三种类型的项。

数据项可以是整数或字符串项。以下是 MQAI 中可用的数据项类型：

- 整数
- 64 位整数
- 整数过滤器
- 字符串
- 字符串过滤器
- 字节字符串
- 字节字符串过滤器
- 袋柄

使用数据项

以下是使用数据项的方式：

- [第 61 页的『对数据项进行计数』](#)。
- [第 61 页的『删除数据项』](#)。
- [第 57 页的『使用 MQAI 将数据项添加到包』](#)。
- [第 58 页的『过滤和查询数据项』](#)。

系统项和 MQAI

系统项可由 IBM MQ 管理接口 (MQAI) 用于：

- 生成 PCF 头。系统项可以控制 PCF 命令标识，控制选项，消息序号和命令类型。
- 数据转换。系统项处理包中字符串项的字符集标识。

与所有数据项一样，系统项由选择器和值组成。有关这些选择器及其对象的信息，请参阅 [MQAI 选择器](#)。

系统项是唯一的。一个或多个系统项可由系统选择器标识。每个系统选择器只有一个实例。

可以修改大多数系统项 (请参阅 [第 59 页的『更改包中的信息』](#))，但用户无法更改包创建选项。无法删除系统项。(请参阅 [第 61 页的『删除数据项』](#)。)

使用 MQAI 将数据项添加到包

使用 IBM MQ 管理接口 (MQAI) 创建数据包时，可以使用数据项填充该数据包。这些数据项可以是用户或系统项。

有关数据项的更多信息，请参阅 [第 56 页的『MQAI 中可用的数据项类型』](#)。

MQAI 允许您将整数项，64 位整数项，整数过滤器项，字符串项，字符串过滤器，字节字符串项和字节字符串过滤器项添加到包中，如 [第 58 页的图 6](#) 中所示。这些项由选择器标识。通常，一个选择器仅标识一个项，但并非总是如此。如果包中已存在具有指定选择器的数据项，那么会将该选择器的其他实例添加到包的末尾。

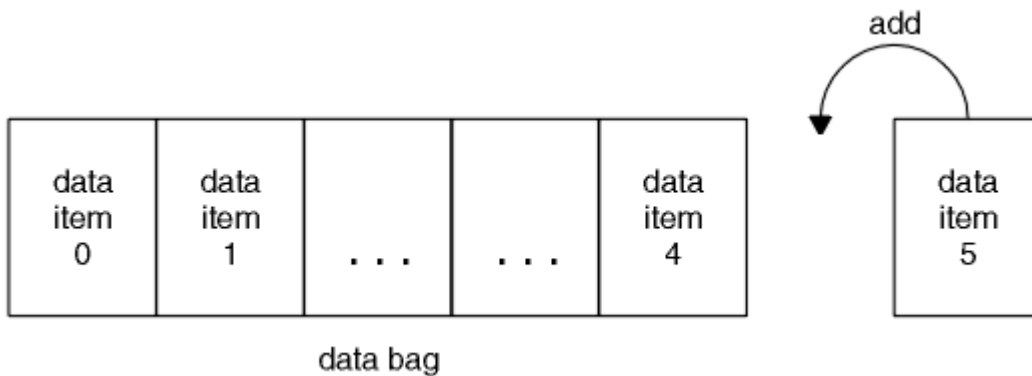


图 6: 添加数据项

使用 `mqAdd*` 调用将数据项添加到包中:

- 要添加整数项, 请使用 `mqAddInteger` 调用, 如 [mqAddInteger](#) 中所述
 - 要添加 64 位整数项, 请使用 `mqAddInteger64` 调用, 如 [mqAddInteger64](#) 中所述
 - 要添加整数过滤器项, 请使用 `mqAddIntegerFilter` 调用, 如 [mqAddIntegerFilter](#) 中所述
 - 要添加字符串项, 请使用 `mqAddString` 调用, 如 [mqAddString](#) 中所述
 - 要添加字符串过滤器项, 请使用 `mqAddStringFilter` 调用, 如 [mqAddStringFilter](#) 中所述
 - 要添加字节字符串项, 请使用 `mqAddByteString` 调用, 如 [mqAddByteString](#) 中所述
 - 要添加字节字符串过滤器项, 请使用 `mqAddByteString` 过滤器调用, 如 [mqAddByteString 过滤器](#) 中所述
- 有关将数据项添加到包的更多信息, 请参阅 [第 57 页的『系统项和 MQAI』](#)。

向包添加查询命令

`mqAdd` 查询调用用于将查询命令添加到包中。该调用专门用于管理目的, 因此只能与管理包一起使用。它允许您指定要从 IBM MQ 查询的属性的选择器。

有关 `mqAdd` 查询调用的完整描述, 请参阅 [mqAdd 查询](#)。

过滤和查询数据项

使用 MQAI 查询 IBM MQ 对象的属性时, 可以通过两种方式控制返回到程序的数据。

- 您可以 **过滤** 使用 `mqAddInteger` 和 `mqAddString` 调用返回的数据。此方法允许您指定 *Selector* 和 *ItemValue* 对, 例如:

```
mqAddInteger(inputbag, MQIA_Q_TYPE, MQQT_LOCAL)
```

此示例指定队列类型 (*Selector*) 必须是本地 (*ItemValue*), 并且此规范必须与您要查询的对象 (在本例中是队列) 的属性匹配。

可过滤的其他属性对应于可在 [第 19 页的『IBM MQ 可编程命令格式简介』](#) 中找到的 PCF Inquire * 命令。例如, 要查询通道的属性, 请参阅本产品文档中的 "查询通道" 命令。"查询通道" 命令的 "必需参数" 和 "可选参数" 标识可用于过滤的选择器。

- 您可以使用 `mqAdd` 查询调用来 **查询** 对象的特定属性。这将指定您感兴趣的选择器。如果未指定选择器, 那么将返回对象的所有属性。

以下是过滤和查询队列属性的示例:

```
/* Request information about all queues */
mqAddString(adminbag, MQCA_Q_NAME, "*")

/* Filter attributes so that local queues only are returned */
mqAddInteger(adminbag, MQIA_Q_TYPE, MQQT_LOCAL)

/* Query the names and current depths of the local queues */
```

```

mqAddInquiry(adminbag, MQCA_Q_NAME)
mqAddInquiry(adminbag, MQIA_CURRENT_Q_DEPTH)

/* Send inquiry to the command server and wait for reply */
mqExecute(MQCMD_INQUIRE_Q, ...)

```

在数据包中查询

您可以查询:

- 使用 mqInquire 整数调用的整数项的值。请参阅 [mqInquireInteger](#)。
- 使用 mqInquireInteger64 调用的 64 位整数项的值。请参阅 [mqInquireInteger64](#)。
- 使用 mqInquireIntegerFilter 调用的整数过滤器项的值。请参阅 [mqInquireIntegerFilter](#)。
- 使用 mqInquire 字符串调用的字符串项的值。请参阅 [mqInquire](#) 字符串。
- 使用 mqInquireStringFilter 调用的字符串过滤器项的值。请参阅 [mqInquireStringFilter](#)。
- 使用 mqInquireByteString 调用的字节字符串项的值。请参阅 [mqInquireByteString](#)。
- 使用 mqInquireByteString 过滤器调用的字节字符串过滤器项的值。请参阅 [mqInquireByteString 过滤器](#)。
- 使用 mqInquireBag 调用的包句柄的值。请参阅 [mqInquire](#) 包。

您还可以使用 mqInquireItemInfo 调用来查询特定项的类型 (整数, 64 位整数, 整数过滤器, 字符串, 字符串过滤器, 字节字符串, 字节字符串过滤器或包句柄)。请参阅 [mqInquireItemInfo](#)。

更改包中的信息

MQAI 允许您使用 mqSet* 调用来更改包中的信息。您可以:

1. 修改包中的数据项。索引允许通过标识要修改的项的出现来替换参数的单个实例 (请参阅 [第 59 页的图 7](#))。

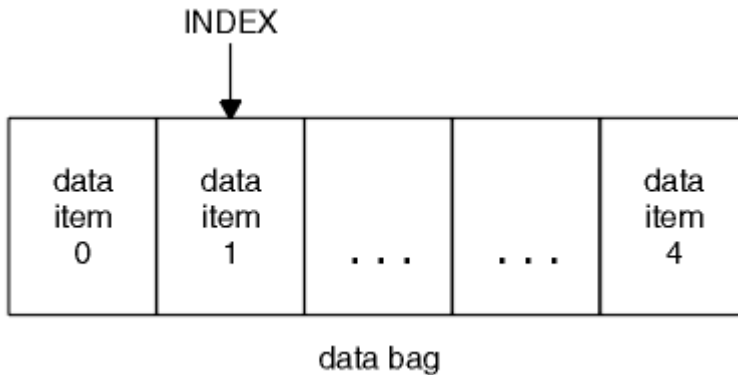


图 7: 修改单个数据项

2. 删除指定选择器的所有现有实例, 并将新实例添加到包的末尾。(请参阅 [第 59 页的图 8](#)。)特殊索引值允许替换参数的 **所有** 实例。

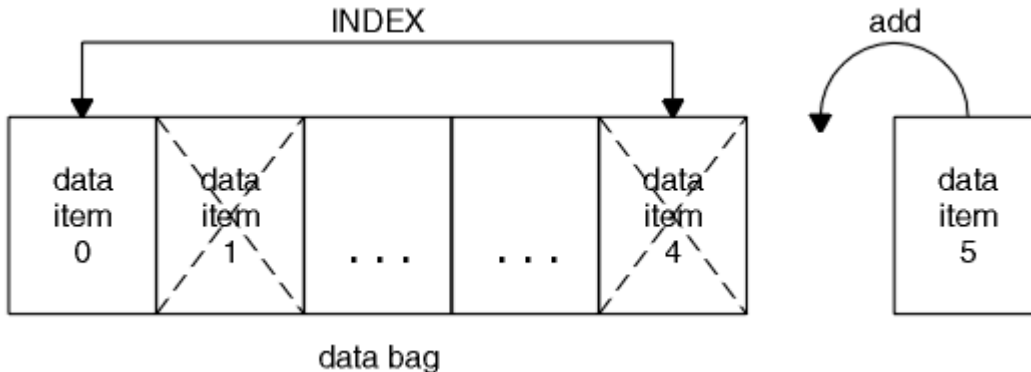


图 8: 修改所有数据项

注: 索引会保留包中的插入顺序, 但会影响其他数据项的索引。

mqSet 整数调用允许您修改包中的整数项。mqSetInteger64 调用允许您修改 64 位整数项。mqSetIntegerFilter 调用允许您修改整数过滤器项。mqSet 字符串调用允许您修改字符串项。mqSetStringFilter 调用允许您修改字符串过滤器项。mqSetByteString 调用允许您修改字节字符串项。mqSetByteString 过滤器调用允许您修改字节字符串过滤器项。或者, 您可以使用这些调用来删除指定选择器的所有现有实例, 并在包的末尾添加新实例。数据项可以是用户项或系统项。

有关这些调用的完整描述, 请参阅:

- [mqSet 整数](#)
- [mqSetInteger64](#)
- [mqSetIntegerFilter](#)
- [mqSet 字符串](#)
- [mqSetStringFilter](#)
- [mqSetByteString](#)
- [mqSetByteString 过滤器](#)

使用 `mqClearBag` 调用清除包

mqClearBag 调用从用户包中除去所有用户项, 并将系统项重置为其初始值。还会删除包中包含的系统包。

有关 mqClearBag 调用的完整描述, 请参阅 [mqClearBag](#)。

使用 `mqTruncateBag` 调用截断包

mqTruncateBag 调用通过从包末尾删除项 (从最近添加的项开始) 来减少用户包中的用户项数。例如, 当使用相同的头信息生成多条消息时, 可以使用它。

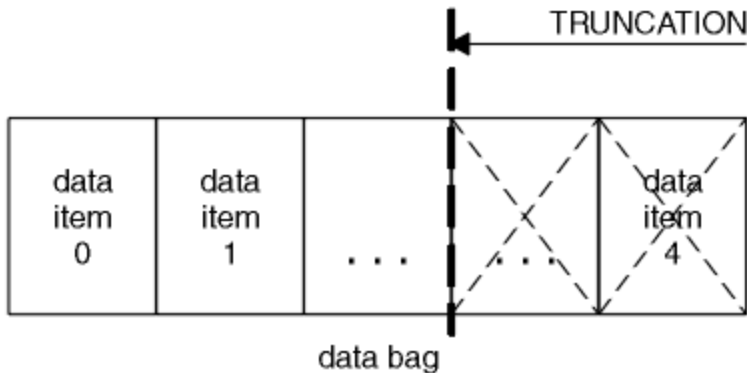


图 9: 截断包

有关 mqTruncateBag 调用的完整描述, 请参阅 [mqTruncateBag](#)。

转换包和缓冲区

要在应用程序之间发送数据, 首先将消息数据放在包中。然后, 使用 `mqBagToBuffer` 调用将包中的数据转换为 PCF 消息。使用 `MQPUT` 调用将 PCF 消息发送到所需队列。如图第 60 页的图 10 所示。有关 `mqBagToBuffer` 调用的完整描述, 请参阅 [mqBagToBuffer](#)。

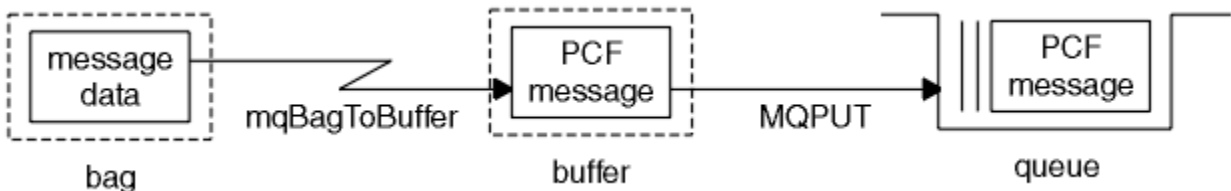


图 10: 将包转换为 PCF 消息

要接收数据，将使用 MQGET 调用将消息接收到缓冲区中。然后，使用 mqBufferToBag 调用将缓冲区中的数据转换为包，前提是缓冲区包含有效的 PCF 消息。如图 第 61 页的图 11 所示。有关 mqBufferToBag 调用的完整描述，请参阅 [mqBufferToBag](#)。

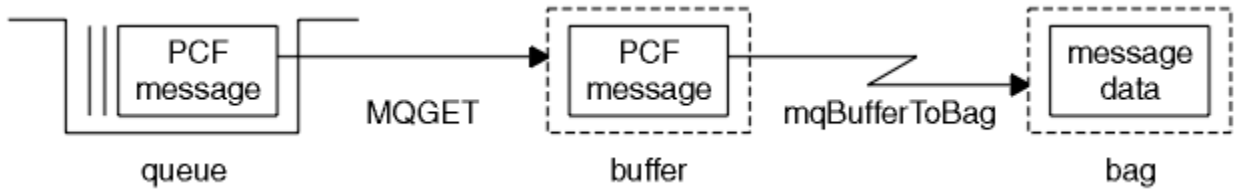


图 11: 将 PCF 消息转换为包格式

对数据项进行计数

mqCount 项调用对存储在数据包中的用户项数和/或系统项数进行计数，并返回此数字。例如，mqCountItems(Bag, 7, ...)返回包中选择器为 7 的项数。它可以按单个选择器，按用户选择器，按系统选择器或按所有选择器对项进行计数。

注: 此调用计算数据项的数量，而不是包中唯一选择器的数量。选择器可以多次出现，因此包中的唯一选择器可能少于数据项。

有关 mqCount 项调用的完整描述，请参阅 [mqCount](#) 项。

删除数据项

您可以通过多种方式从包中删除项目。您可以：

- 从包中除去一个或多个用户项。有关详细信息，请参阅 第 61 页的『使用 mqDeleteItem 调用从包中删除数据项』。
- 从包中删除 **所有** 用户项，即清除一个包。有关详细信息，请参阅 第 60 页的『使用 mqClearBag 调用清除包』。
- 从包的末尾删除用户项，即截断包。有关详细信息，请参阅 第 60 页的『使用 mqTruncateBag 调用截断包』。

使用 mqDeleteItem 调用从包中删除数据项

mqDelete 项调用从包中除去一个或多个用户项。索引用于删除以下任一项：

1. 一次出现指定的选择器。(请参阅 第 61 页的图 12。)

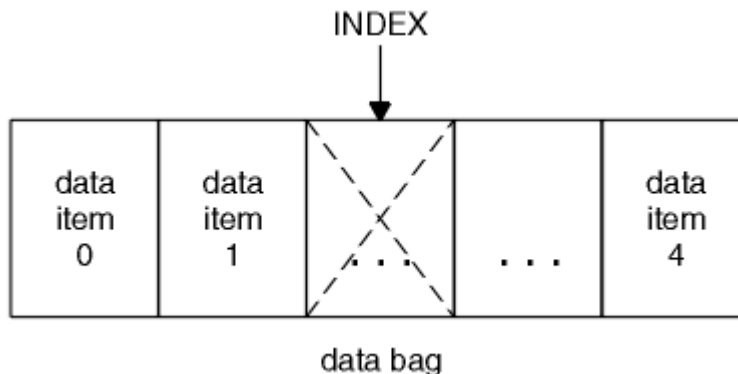


图 12: 删除单个数据项

或

2. 指定选择器的所有实例。(请参阅 第 62 页的图 13。)

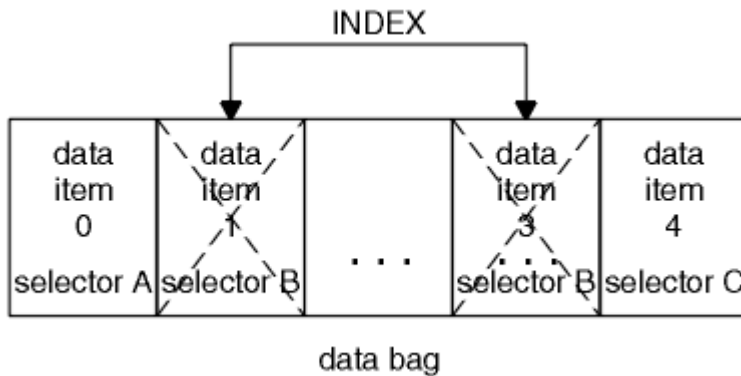


图 13: 删除所有数据项

注: 索引会保留包中的插入顺序, 但会影响其他数据项的索引。例如, `mqDeleteItem` 调用不会保留跟在已删除项后面的数据项的索引值, 因为将重组这些索引以填补从已删除项中保留的间隔。

有关 `mqDelete` 项调用的完整描述, 请参阅 [mqDelete](#) 项。

使用 `mqExecute` 调用将管理命令发送到 `qm` 命令服务器

创建并填充数据包后, 可以使用 `mqExecute` 调用将管理命令消息发送到队列管理器的命令服务器。这将处理与命令服务器的交换, 并在包中返回响应。

创建并填充数据包后, 可以将管理命令消息发送到队列管理器的命令服务器。执行此操作的最简单方法是使用 `mqExecute` 调用。`mqExecute` 调用将管理命令消息作为非持久消息发送, 并等待任何响应。响应在响应包中返回。例如, 这些属性可能包含与多个 IBM MQ 对象或一系列 PCF 错误响应消息相关的属性的相关信息。因此, 响应包只能包含返回码, 或者可以包含嵌套包。

响应消息将放入系统创建的系统包中。例如, 对于有关对象名称的查询, 将创建一个系统包以保存这些对象名称, 并将该包插入到用户包中。然后将这些包的句柄插入到响应包中, 选择器 `MQHA_BAG_HANDLE` 可访问嵌套包。如果未删除系统包, 那么该系统包将保留在存储器中, 直到删除响应包为止。

第 62 页的图 14 中显示了嵌套概念。

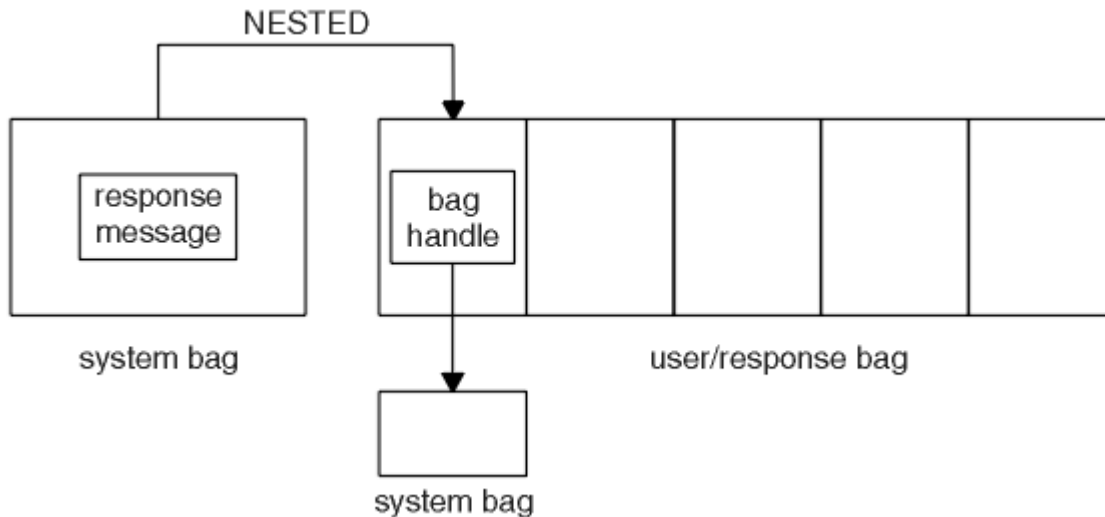


图 14: 嵌套

作为 `mqExecute` 调用的输入, 必须提供:

- MQI 连接句柄。
- 要执行的命令。这应该是 `MQCMD_*` 值之一。

注: 如果 `MQAI` 无法识别此值, 那么仍会接受该值。但是, 如果使用 `mqAdd` 查询调用将值插入到包中, 那么此参数必须是 `MQAI` 识别的 `INQUIRE` 命令。即, 参数的格式应为 `MQCMD_INQUIRE_*`。

- (可选) 包含控制呼叫处理的选项的包的手柄。这也是您可以指定 MQAI 应该等待每条应答消息的最大时间 (以毫秒为单位)。
- 管理包的句柄, 其中包含要发出的管理命令的详细信息。
- 接收应答消息的响应包的句柄。

以下句柄是可选的:

- 要放置管理命令的队列的对象句柄。

如果未指定对象句柄, 那么会将管理命令放在 SYSTEM.ADMIN.COMMAND.QUEUE 属于当前连接的队列管理器。这是缺省值。

- 要放置应答消息的队列的对象句柄。

您可以选择将应答消息放在 MQAI 自动创建的动态队列上。创建的队列仅在调用期间存在, 并且在退出 mqExecute 调用时被 MQAI 删除。

有关使用 mqExecute 调用的示例, 请参阅 [示例代码](#)

V 9.0.1 使用 REST API 进行管理

您可以使用 administrative REST API 来管理 IBM MQ 对象, 例如队列管理器和队列 **V 9.0.5** 以及 Managed File Transfer GET REST 服务。将以 JSON 格式向 administrative REST API 发送信息并从其接收信息。这些 RESTful API 可帮助您将 IBM MQ 管理嵌入到流行的 DevOps 和自动化工具中。

开始之前

有关可用 REST 资源的参考信息, 请参阅 [administrative REST API 参考](#)。

注:

在 IBM MQ 9.0.1 中, administrative REST API 未与 IBM MQ 安全性集成。因此, 缺省情况下会禁用 administrative REST API。必须先手动启用 administrative REST API, 然后才能使用。有关启用 administrative REST API 的更多信息, 请参阅第 63 页的『[开始使用 administrative REST API](#)』。

V 9.0.2 从 IBM MQ 9.0.2 开始, administrative REST API 与 IBM MQ 安全性集成。缺省情况下, 已启用 administrative REST API。但是, 必须先配置安全性, 然后才能使用 administrative REST API。有关更多信息, 请参阅 [IBM MQ 控制台和 REST API 安全性](#)。

过程

- [第 63 页的『开始使用 administrative REST API』](#)
- [第 69 页的『使用 administrative REST API』](#)
- [第 74 页的『使用 REST API 进行远程管理』](#)
- [第 77 页的『REST API 时间戳记』](#)
- [第 77 页的『REST API 错误处理』](#)
- [第 80 页的『REST API 发现』](#)
- [第 81 页的『REST API 本地语言支持』](#)

V 9.0.1 开始使用 administrative REST API

在可以启动 administrative REST API 之前, 必须安装正确的组件, 启用 REST API, 配置安全性并启动 mqweb 服务器。

开始之前

IBM i 在 IBM i 上, 命令应该在 QSHLL 中运行。

关于此任务

此任务的过程重点介绍了如何快速开始使用 administrative REST API。用于配置安全性的步骤概括了如何设置基本用户注册表，但也提供了用于配置用户和角色的其他选项。有关配置 administrative REST API 的安全性的更多信息，请参阅 [IBM MQ Console 和 REST API 安全性](#)。

注：您必须是 [特权用户](#) 才能访问 `mqwebuser.xml` 文件。

过程

1. 安装 IBM MQ Console 和 REST API 组件：

- **AIX V 9.0.4** 在 AIX 上，安装 `mqm.web.rte` 文件集。
- **Linux** 在 Linux 上，安装 MQSeriesWeb 组件。有关在 Linux 上安装组件和功能部件的更多信息，请参阅 [Linux 安装任务](#)。
- **Windows** 在 Windows 上，安装 Web Administration 功能部件。有关在 Windows 上安装组件和功能部件的更多信息，请参阅 [Windows 安装任务](#)。
- **z/OS** 在 z/OS 上，安装 IBM MQ for z/OS Unix System Services Web Components 功能部件。有关在 z/OS 上安装组件和功能部件的更多信息，请参阅 [z/OS 安装任务](#)。

2. 可选：从 IBM MQ 9.0.2 开始，缺省情况下会启用 administrative REST API。但是，在 IBM MQ 9.0.1 中，缺省情况下会禁用 administrative REST API。如果您正在使用 IBM MQ 9.0.1，请在 `<server>` 标记中通过将以下 XML 添加到 `mqwebuser.xml` 文件来启用 REST API：

```
<variable name="mqRestAutostart" value="true"/>
```

可以在下列其中一个目录中找到 `mqwebuser.xml`：

- **ULW** `MQ_DATA_DIRECTORY/web/installations/installationName/servers/mqweb`
- **z/OS** `WLP_user_directory/servers/mqweb`

其中，`WLP_user_directory` 是运行 `crtmqweb.sh` 脚本以创建 `mqweb` 服务器定义时指定的目录。

3. **V 9.0.2**

从 IBM MQ 9.0.2 开始，将为 administrative REST API 启用安全性。必须先配置用户和角色，然后才能使用 administrative REST API：

- a) 从 `MQ_INSTALLATION_PATH/web/mq/samp/configuration` 目录中复制 `basic_registry.xml` 文件。
- b) 将该样本 XML 文件放入相应目录中：

- **ULW** 在 UNIX, Linux, and Windows 上：`MQ_DATA_DIRECTORY/web/installations/installationName/servers/mqweb`
- **z/OS** 在 z/OS 上：`WLP_user_directory/servers/mqweb`

其中，`WLP_user_directory` 是在运行 `crtmqweb.sh` 脚本来创建 `mqweb` 服务器定义时指定的目录。

- c) 将该样本 XML 文件重命名为 `mqwebuser.xml`。

注：这个重命名的文件将替换 IBM MQ Console 也在使用的现有文件。因此，如果更改了 IBM MQ Console 的 `mqwebuser.xml` 文件，请将更改复制到新的 XML 文件，然后再将其重命名。

- d) 可选：编辑 `mqwebuser.xml` 文件以添加用户和组。为这些用户和组分配要授权使用 administrative REST API 的相应角色。您还可以更改缺省情况下所定义用户的密码，并对新密码进行编码。有关更多信息，请参阅 [配置用户和角色](#)。

4. 启用与 `mqweb` 服务器的远程连接：

- 从 V 9.0.4 开始，使用 `setmqweb` 命令：


```
setmqweb properties -k httpHost -v hostname
```

- 从 V 9.0.1 开始，将以下 XML 添加到 mqwebuser.xml 文件中的 <server> 标记内：

```
<variable name="httpHost" value="hostname"/>
```

其中，*hostname* 指定 IP 地址、带域名后缀的域名服务器 (DNS) 主机名或安装了 IBM MQ 的服务器的 DNS 主机名。使用星号 * 可指定所有可用的网络接口。



注意： **V 9.0.4** **z/OS**

在 z/OS 上发出 **setmqweb** 或 **dspmweb** 命令之前，必须设置 WLP_USER_DIR 环境变量，以便此变量指向 mqweb 服务器配置。

为此，请发出以下命令：

```
export WLP_USER_DIR=WLP_user_directory
```

其中 *WLP_user_directory* 是传递到 `crtmqweb.sh` 的目录的名称。例如：

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

有关更多信息，请参阅[创建 Liberty 服务器定义](#)。

5. 启动支持 REST API 的 mqweb 服务器：

- **Windows** **Linux** 在 Windows 和 Linux 上，作为 [特权用户](#)，在命令行上输入以下命令：
`strmqweb`
- **z/OS** 在 z/OS 上，启动在[任务 29：为 IBM WLP 服务器创建过程中创建的过程](#)。

下一步做什么

1. **V 9.0.2** 选择向 mqweb 服务器认证 administrative REST API 用户的方式。您不必对所有用户使用相同的方法。可选项有：
 - 使用 HTTP 基本认证来认证用户。在这种情况下，用户名和密码将进行编码但不会加密，并随每个 REST API 请求一起发送，以针对此请求对用户进行认证和授权。为保证此认证的安全，必须使用安全连接。即，必须使用 HTTPS。有关更多信息，请参阅[将 HTTP 基本认证用于 REST API](#)。
 - 使用令牌认证来认证用户。在这种情况下，用户使用 HTTP POST 方法向 REST API login 资源提供用户标识和密码。这会生成一个 LTPA 令牌，可让用户在一段时间内保持登录和授权状态。为保证此认证的安全，必须使用安全连接。即，必须使用 HTTPS。有关更多信息，请参阅[对 REST API 使用基于令牌的认证](#)。
 - 使用客户机证书来认证用户。在这种情况下，用户不会使用用户标识或密码来登录到 administrative REST API，而改为使用客户机证书。有关更多信息，请参阅[对 REST API 使用客户机证书认证](#)。
2. **V 9.0.1** 配置 REST API 设置，包括启用 HTTP 连接和更改端口号。有关更多信息，请参阅[配置 IBM MQ Console 和 REST API](#)。
3. **V 9.0.5** (可选) 为 MFT 配置 REST API。有关更多信息，请参阅第 66 页的『[MFT REST API 的必需配置](#)』。
4. **V 9.0.2** (可选) 为 REST API 配置跨源资源共享。缺省情况下，您无法通过与 REST API 不在相同域上托管的 Web 资源来访问 REST API。即，未启用跨源请求。您可以配置跨源资源共享 (CORS) 以允许来自指定 URL 的跨源请求。有关更多信息，请参阅[为 REST API 配置 CORS](#)。
5. 使用 REST API。有关更多信息，请参阅第 69 页的『[使用 administrative REST API](#)』，第 74 页的『[使用 REST API 进行远程管理](#)』，和 [管理 REST API 参考](#)。

注：您可以随时使用 **endmqweb** 命令来停止 mqweb 服务器。但是，如果 mqweb 服务器未在运行，那么无法使用 REST API 或 IBM MQ Console。

V 9.0.5 MFT REST API 的必需配置

将 Managed File Transfer 配置为使用 REST API 时需要执行的步骤

在 mqwebuser.xml 文件中, 设置:

- **mqRestMftEnabled** 属性必须为 *true*

注: 如果更改此属性的值, 那么必须重新启动 mqweb 服务器。

- **mqRestMftCoordinationQmgr** 属性到在运行 mqweb 服务器的机器上本地运行的相应 MFT 协调队列管理器。

要使队列管理器充当 REST API 的 MFT 协调队列管理器, 必须运行以下命令:

- 针对 mqwebuser.xml 文件中的 **mqRestMftCoordinationQmgr** 属性设置的同一本地队列管理器上的 **fteSetupCoordination** 命令。

此命令生成包含 IBM MQ 对象定义的 MQSC 文件。请注意, 该命令可以在安装了 MFT 的任何机器上运行。

- 应针对协调队列管理器运行上一步中生成的 MQSC 文件, 以创建所需的 IBM MQ 对象。

相关信息

[配置 MFTREST API 安全性](#)

[管理 MFT REST API](#)

[GET-传输列表](#)

[GET-传输状态](#)

V 9.0.1 确定 REST API URL

用于在 IBM MQ 9.0.4 和更高版本访问 administrative REST API 的缺省 URL 为: <https://localhost:9443/ibmmq/rest/v1/admin>。在 IBM MQ 9.0.3 和更低版本上, 缺省 URL 为: <https://localhost:9443/ibmmq/rest/v1>。用于访问 messaging REST API 的缺省 URL 为: <https://localhost:9443/ibmmq/rest/v1/messaging>。如果主机或端口已从缺省值更改, 或者如果启用了 HTTP, 那么可以使用 **dspmweb** 命令来确定 URL。

关于此任务

V 9.0.4 从 IBM MQ 9.0.4 开始, 您可以使用 **dspmweb status** 命令来确定 Windows, Linux 和 z/OS 上的 REST API URL。

对于 IBM MQ 9.0.3 和更低版本, 在 Windows 和 Linux 上, 可以使用 **dspmweb** 命令。在 z/OS 上, 必须搜索 messages.log 文件以确定 URL。

过程

V 9.0.4

请使用以下某种方法来确定该 URL:

- 从 IBM MQ 9.0.4 开始, 以 [特权用户](#) 身份使用 **dspmweb status** 命令:
 1. 通过在命令行处输入 **strmqweb** 命令来确保 mqweb 服务器正在运行。
 2. 通过在命令行上输入 **dspmweb status** 命令来显示 URL。
- 对于 IBM MQ 9.0.3 和更低版本, 请使用下列其中一种方法:

- **Windows** **Linux** 在 Windows 或 Linux 上, 以 [特权用户](#) 身份使用 **dspmweb** 命令:

1. 通过在命令行处输入 **strmqweb** 命令来确保 mqweb 服务器正在运行。
2. 通过在命令行上输入 **dspmweb** 命令来显示 URL。

- **z/OS** 在 z/OS 上, 使用 messages.log 文件来查找 URL:

1. 打开 `messages.log` 文件。

`messages.log` 文件位于 `WLP_user_directory/servers/mqweb/logs` 路径上，其中 `WLP_user_directory` 是运行 `crtmqweb.sh` 脚本以创建 mqweb 服务器定义时指定的目录。

2. 搜索以 `ibmmq/rest/v1` 结尾的最新 `CWWKT0016I` 消息代码。该 URL 包含在此消息中。

示例

以下示例显示了将此 URL 作为缺省 URL 的消息代码 `CWWKT0016I`：

```
A CWWKT0016I: Web application available (default_host): https://localhost:9443/ibmmq/rest/v1
```

V9.0.2 z/OS 正在检查是否在 z/OS 上正确配置了 mqweb 服务器

检查是否在 z/OS 上正确配置了 mqweb 服务器以及修复常见配置问题时需要执行的步骤。

过程

1. 检查 mqweb 服务器启动式任务。

您应该会看到类似于以下内容的作业日志消息：

```
+CWWKE0001I: The server mqweb has been launched.
+CWWKZ0001I: Application com.ibm.mq.rest started in 1.493 seconds.
+CWWKZ0001I: Application com.ibm.mq.console started in 0.885 seconds.
+CWWKF0011I: The server mqweb is ready to run a smarter planet.
```

STDERR 中不应有任何消息

STDOUT 中应该有类似如下的消息：

```
Launching mqweb (WebSphere Application Server 17.0.0.2, WAS FOR Z/OS 17.0.0.2,
MQM MVS/ESA V9 R0.4/wlp-1.0.17.c1170220170523-1818) on IBM J9 VM,
version pmz6480sr4fp7-20170627_02 (SR4 FP7) (en_US)
[AUDIT ] CWWKE0001I: The server mqweb has been
launched.
[AUDIT ] CWWKG0028A: Processing included configuration resource: /mqm/V9R0M4/web/mq/etc/
mqweb.xml
[AUDIT ] CWWKG0028A: Processing included configuration resource:
var/mqm/mqweb904/servers/mqweb/mqwebuser.xml
[AUDIT ] CWWKT0016I: Web application available (default_host):
yourhost:yourport/api/docs/
[AUDIT ] CWWKT0016I: Web application available (default_host):
yourhost:yourport/api/explorer/
[AUDIT ] CWWKT0016I: Web application available (default_host):
yourhost:yourport/ibm/api/
[AUDIT ] CWWKT0016I: Web application available (default_host):
yourhost:yourport/ibm/api/explorer/
[AUDIT ] CWWKT0016I: Web application available (default_host):
yourhost:yourport/ibm/api/docs/subscription/websocket/
[AUDIT ] CWWKT0016I: Web application available (default_host):
yourhost:yourport:9080/ibmmq/rest/v1/
[AUDIT ] CWWKZ0001I: Application com.ibm.mq.rest started in 1.493 seconds.
[AUDIT ] CWWKT0016I: Web application available (default_host):
yourhost:yourport/ibmmq/console/
[AUDIT ] CWWKT0016I: Web application available (default_host):
yourhost:yourport/ibmmq/console/internal/
[AUDIT ] CWWKZ0001I: Application com.ibm.mq.console started in 1.459 seconds.
[AUDIT ] CWWKF0012I: The server installed the following features:
[jsp-2.2, servlet-3.1, ssl-1.0, jndi-1.0, basicAuthenticationMQ-1.0,
apiDiscovery-1.0, localConnector-1.0, appSecurity-2.0, jaxrs-1.1,
concurrent-1.0, json-1.0, websocket-1.0, distributedMap-1.0,
applicationMonitorMQ-1.0].
[AUDIT ] CWWKF0011I: The server mqweb is ready to run a smarter planet.
[AUDIT ] MQWB2019I: MQ Console level: 9.0.4 - V904-GA904-L171016
[AUDIT ] MQWB0023I: MQ REST API level: 9.0.4 - V904-GA904-L171016
```

注意：

- a. 如果该过程无法启动, 请解决问题。
- b. 如果消息类似于:

```
Web application available (default_host):  
http://localhost:portnumber/ibmmq/console/
```

您尚未设置 **httpHost** 属性以允许与 mqweb 服务器的远程连接。

```
<variable name="httpHost" value="*" />
```

- c. 如果您已更改 **httpHost** 属性, 那么将收到类似如下的消息:

```
Web application available (default_host):  
yourhost:portnumber/ibmmq/console/
```

根据您的 TCP/IP 配置, 您可能无法使用显示为 URL 的地址。

- d. 对 XML 服务器配置文件的更改将在几秒钟后生效。无需重新启动 mqweb 服务器
2. 连接到 IBM MQ Console。

您应该会显示一个标题为 IBM MQ Console 的窗口。

注意:

- a. 如果看到具有 This site can't be reached 或 Context Root Not Found 的窗口, 那么 IBM MQ Console 处于不活动状态; 您应该等待。
- b. 如果未显示标题为 IBM MQ Console 的窗口, 那么在运行 **crtmqweb.sh** 脚本以创建 mqweb 服务器定义时, WLP_USER_DIRECTORY/servers/mqweb/logs/messages.log 中可能有其他诊断信息, 其中 WLP_USER_DIRECTORY 是您指定的目录。

请注意, 此文件采用 ASCII 格式, 要查看此文件, 您可以从 USS 命令行使用 **oedit**, 或者使用 ISPF 选项 3.17 并使用 **va** (视图 ASCII) 行命令。

- c. 您可能会在 STDOUT 中获取类似于以下内容的消息:

```
[WARNING ] SRVE0190E: File not found: /nls/en_GB/labels.json  
[WARNING ] SRVE0190E: File not found: /nls/en_GB/errors.json  
[WARNING ] SRVE0190E: File not found: /nls/en_GB/strings.json  
[WARNING ] SRVE0190E: File not found: /nls/en_GB/  
pcf.json
```

您可以忽略这些消息。

- d. 如果您在运行 mqweb 服务器的系统上定义了队列管理器, 并且这些队列管理器与 mqweb 服务器在同一级别运行, 那么这些队列管理器将显示在 MQ Console 中的 "本地队列管理器" 窗口小部件中。

如果未显示任何队列管理器, 那么您没有与 mqweb 服务器处于同一级别的任何队列管理器 (自上次 IPL 以来已启动)。

3. 如果收到消息:

```
Lost communication with the server Could not establish communication with  
the server.
```

- a) 在用于启动 mqweb 服务器的过程中:

- i) 检查 STEPLIB 库的级别是否正确以及是否已获得 APF 授权。
- ii) 请检查 PATH 和 LIBPATH 是否指向正确的路径。PATH 和 LIBPATH 是在基于样本 CSQ4WEBS 的 Web 服务器启动式任务过程中定义的。

- b) 在 USS 中, 使用命令 **ls -Eltr PathPrefix/web/bin/dspmq**, 其中 PathPrefix 是 IBM MQ Unix 系统服务组件安装路径。

这将显示类似于 **-rwxr-xr-t a-s- ... /mqm/V9R0M4/web/bin/dspmq** 的输出

检查是否已设置 **t** 和 **a** 标志。

使用以下命令:

- `chmod +t PathPrefix/web/bin/dspmq` 以设置粘性位 (t)
- 用于设置 APF 授权属性 (a) 的 `extattr +a PathPrefix/web/bin/dspmq`

V 9.0.1 使用 administrative REST API

使用 administrative REST API 时，将对表示各种 IBM MQ 对象 (例如队列管理器或队列) 的 URL 调用 HTTP 方法。HTTP 方法 (如 POST) 表示要在 URL 所表示的对象上执行的操作类型。有关操作的更多信息可以在 JSON 中作为 HTTP 方法有效内容的一部分提供，也可以在查询参数中进行编码。操作的执行结果信息可作为 HTTP 响应主体返回。

开始之前

在使用 administrative REST API 之前，请考虑以下事项：

- **V 9.0.2** 您必须向 mqweb 服务器进行认证才能使用 administrative REST API。您可以使用 HTTP 基本认证、客户机证书认证或基于令牌的认证来进行认证。有关如何使用这些认证方法的更多信息，请参阅 [IBM MQ Console](#) 和 [REST API 安全性](#)。
- REST API 区分大小写。例如，如果队列管理器名为 `qmgr1`，那么以下 URL 上的 HTTP GET 不会显示信息。

```
/ibmmq/rest/v1/admin/qmgr/QMGR1
```

- 并非 IBM MQ 对象名称中可使用的所有字符都能在 URL 中直接进行编码。要直接对这些字符进行编码，必须使用相应的 URL 编码：
 - 正斜杠 / 必须编码为 %2F。
 - 百分号 % 必须编码为 %25。
- 由于某些浏览器的行为，请勿仅使用句点或正斜杠字符来命名对象。

关于此任务

使用 REST API 对对象执行操作时，首先需要构造 URL 以表示该对象。每个 URL 都以一个前缀开头，此前缀描述了要向其发送请求的主机名和端口。URL 的其余部分描述特定对象或一组对象 (称为资源)。

要对资源执行的操作定义 URL 是否需要查询参数。它还定义了所使用的 HTTP 方法，以及是将其他信息以 JSON 格式发送到 URL 还是从 URL 返回其他信息。此附加信息可作为 HTTP 请求的一部分，或者作为 HTTP 响应的一部分返回。

构造 URL 并创建用于在 HTTP 请求中发送的可选 JSON 有效内容后，可以将 HTTP 请求发送到 IBM MQ。您可以使用所选编程语言内置的 HTTP 实现来发送请求。您还可以使用命令行工具 (例如 cURL)，Web 浏览器或 Web 浏览器附加组件来发送请求。

要点: 您必须至少执行步骤 [第 69 页的『1.a』](#) 和 [第 70 页的『1.b』](#)。

过程

1. 构造 URL:

- a) 开头为以下前缀 URL:

V 9.0.4 IBM MQ 9.0.4 和更高版本:

```
https://host:port/ibmmq/rest/v1/admin
```

IBM MQ 9.0.3 和更低版本:

```
https://host:port/ibmmq/rest/v1
```

主机

指定可使用 administrative REST API 的主机名或 IP 地址。

缺省值为 `localhost`。

port

指定 administrative REST API 使用的 HTTPS 端口号。

缺省值为 9443。

V 9.0.1 如果启用 HTTP 连接，那么可使用 HTTP 代替 HTTPS。有关启用 HTTP 的更多信息，请参阅配置 HTTP 和 HTTPS 端口。

有关如何确定前缀 URL 的更多信息，请参阅第 66 页的『确定 REST API URL』。

b) 将资源添加到 URL 路径。

以下资源可用：

- [安装](#)
- [QMGR](#)
- **V 9.0.2** [队列](#)
- **V 9.0.4** [预订](#)
- **V 9.0.4** [CHANNEL](#)

例如，要与队列管理器交互，请将 /qmgr 添加到前缀 URL 以创建以下 URL：

V 9.0.4 IBM MQ 9.0.4 和更高版本：

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr
```

IBM MQ 9.0.3 和更低版本：

```
https://localhost:9443/ibmmq/rest/v1/qmgr
```

c) 可选：将任何其他可选路径段添加到 URL。

在每种对象类型的参考信息中，可选段可以在 URL 中由环绕它的花括号 {} 来标识。

例如，将队列管理器名称 QM1 添加到 URL 以创建以下 URL：

V 9.0.4 IBM MQ 9.0.4 和更高版本：

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1
```

IBM MQ 9.0.3 和更低版本：

```
https://localhost:9443/ibmmq/rest/v1/qmgr/QM1
```

d) 可选：将可选查询参数添加到 URL 中。

向 URL 添加问号、变量名称，等号 = 以及 URL 的值或值列表。

例如，要请求队列管理器 QM1 的所有属性，请创建以下 URL：

V 9.0.4 IBM MQ 9.0.4 和更高版本：

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM1?attributes=*
```

IBM MQ 9.0.3 和更低版本：

```
https://localhost:9443/ibmmq/rest/v1/qmgr/QM1?attributes=*
```

e) 将更多的可选查询参数添加到 URL 中。

Add an ampersand, &, to the URL, and then repeat 步骤 d.

2. 在 URL 上调用相关的 HTTP 方法。指定任何可选 JSON 有效内容，并提供相应的安全凭证以认证。例如：

- 使用所选编程语言的 HTTP/REST 实现。
- 使用诸如 REST 客户机浏览器附加组件或 cURL 之类的工具。

V 9.0.5 获取 MFT 传输列表或传输状态

如何使用 REST API 获取 Managed File Transfer 传输列表

有关更多信息，请参阅 [使用 REST API 进行管理](#)。

要获取传输列表，请执行以下操作：

1. 将 [GET](#) 请求发送到托管 MFT 和 MFT Web 应用程序的服务器。在发送请求时，需要考虑 URL 的格式。
请参阅 [admin/mft/transfer](#) 以获取更多信息。
2. 当接收到请求时，将对该请求执行检查以查看该请求是否有效。
请参阅 [配置 MFT REST API 安全性](#) 以获取更多信息。
3. 如果请求有效，那么您将接收到成功响应以及响应主体。否则，将生成相应的错误代码和响应代码。
请参阅 [响应状态码](#) 以获取响应列表。

要获取传输状态，请执行以下操作：

1. 将 [GET](#) 请求发送到托管 MFT 和 MFT Web 应用程序的服务器。在发送请求时，需要考虑 URL 的格式。
请参阅 [admin/mft/transfer](#) 以获取更多信息。
2. 当接收到请求时，将对该请求执行检查以查看该请求是否有效。
请参阅 [配置 MFT REST API 安全性](#) 以获取更多信息。
3. 如果请求有效，那么您将接收到成功响应以及响应主体。否则，将生成相应的错误代码和响应代码。
请参阅 [响应状态码](#) 以获取响应列表。

响应主体示例

从内部存储器检索传输列表，并为每个传输生成 JSON 格式的响应主体。传输的响应主体类似于：

```
{
  "transfer": [
    {
      "destinationAgent": { "name": "AGENT.TRI.BANK" },
      "originator": {
        "host": "192.168.99.1",
        "userId": "johndoe"
      },
      "sourceAgent": { "name": "TESTAGENT" },
      "statistics": {
        "endTime": "2018-01-08T16:22:15.569Z",
        "numberOfFileFailures": 0,
        "numberOfFileSuccesses": 2,
        "numberOfFileWarnings": 0,
        "numberOfFiles": 2,
        "startTime": "2018-01-08T16:22:15.242Z"
      },
      "status": {
        "state": "successful"
      },
      "id": "414D51204D465444454D4F3320202020513E525A21109908"
    },
    {
      "destinationAgent": { "name": "AGENT.TRI.BANK" },
      "originator": {
        "host": "192.168.99.1",
        "userId": "ramsubbarao"
      },
      "sourceAgent": { "name": "TESTAGENT" },
      "statistics": {
        "endTime": "2018-01-08T16:22:13.573Z",
        "numberOfFileFailures": 0,
        "numberOfFileSuccesses": 2,
        "numberOfFileWarnings": 0,
        "numberOfFiles": 2,
      }
    }
  ]
}
```

```

    "startTime": "2018-01-08T16:22:13.167Z"
  },
  "status": {
    "state": "successful"
  },
  "id": "414D51204D465444454D4F33202020513E525A21109702"
}
]
}
]
}

```

请参阅 [REST API JSON 响应主体属性](#)，以获取有关这些属性的更多信息。

V 9.0.5 获取 MFT 代理程序状态

如何使用 REST API 获取 Managed File Transfer 代理程序状态

有关更多信息，请参阅 [使用 REST API 进行管理](#)。

要获取代理程序状态，请执行以下操作：

1. 将 GET 请求发送到使用 MFT 和 MFT Web 应用程序的缺省协调队列管理器下的代理。在发送请求时，需要考虑 URL 的格式。
请参阅 [admin/mft/agent](#) 以获取更多信息。
2. 当接收到请求时，将对该请求执行检查以查看该请求是否有效。
请参阅 [配置 MFT REST API 安全性](#) 以获取更多信息。
3. 如果请求有效，那么您将接收到成功响应以及响应主体。否则，将生成相应的错误代码和响应代码。
请参阅 [响应状态码](#) 以获取响应列表。

示例 URL

{baseURI}/ibmmq/rest/{version}/admin/mft/agent

输出协调队列管理器下所有代理的 **name**，**state** 和 **type** 的列表。

有关这些属性的更多信息，请参阅 [代理程序状态 REST API 属性](#)。

如果将 `?attributes=*` 添加到 URL 字符串的末尾，那么输出将包含协调队列管理器下每个代理的所有属性的列表。

如果将 `/{agentName}` 添加到 URL 字符串的末尾，那么输出将包含与 `agentName` 匹配的代理程序的 **name**，**state** 和 **type**。

同样，如果将 `/{agentName}?attributes=*` 添加到 URL 字符串的末尾，那么输出将包含与 `agentName` 匹配的代理程序的所有属性的列表。

{baseURI}/ibmmq/rest/{version}/admin/mft/agent?name={<prefix>*<suffix> }

例如，`{baseURI}/ibmmq/rest/{version}/admin/mft/agent?name=AGENT*TEST` 输出以名称 `AGENT` 开头并以名称 `TEST` 结尾的代理程序的缺省属性摘要。

如果从 URL 中省略 `*TEST`，那么输出将包含以名称 `AGENT` 开头的代理程序的缺省属性摘要。

{baseURI}/ibmmq/rest/{version}/admin/mft/agent?type={agentType}

例如，`{baseURI}/ibmmq/rest/{version}/admin/mft/agent?type=standard` 输出类型为 `standard` 的代理程序的缺省属性摘要。

同样，如果将 `type={agentType}` 替换为 `state={agentState}`，那么输出将显示指定状态的代理程序的缺省属性摘要。

{baseURI}/ibmmq/rest/{version}/admin/mft/agent?type={agentType}&attributes=*

例如，`{baseURI}/ibmmq/rest/{version}/admin/mft/agent?type=standard&attributes=*` 输出类型为 `standard` 的代理程序的所有属性。

您还可以按属性组合进行过滤。例如：

- `{baseURI}/ibmmq/rest/{version}/admin/mft/agent?name=*bob&type=standard&state=ready&attributes=*`
- `{baseURI}/ibmmq/rest/{version}/admin/mft/agent?type=standard&state=ready&attributes=*`

- `{baseURI}/ibmmq/rest/{version}/admin/mft/agent?name=agent*&state=ready&attributes=*`

响应主体示例

请参阅 [代理程序状态 REST API 属性](#) 和 [代理程序状态 REST API 响应主体属性](#)，以获取有关所描述属性的更多信息。

发出以下命令 `/ibmmq/rest/v1/admin/mft/agent/` 将列出代理程序名称，类型和状态的基本代理程序状态。例如：

```
{
  "agent": [ { "name": "AGENT1",
               "state": "ready",
               "type": "standard" },
             { "name": "AGENT2",
               "state": "ready",
               "type": "standard" },
             { "name": "BRIDGE_AGENT3",
               "type": "protocolBridge",
               "state": "ready" },
             { "name": "CD_AGENT",
               "type": "connectDirectBridge",
               "state": "ready" } ]
}
```

发出以下每个命令：

```
/ibmmq/rest/v1/admin/mft/agent?type=standard
/ibmmq/rest/v1/admin/mft/agent?state=stopped
/ibmmq/rest/v1/admin/mft/agent?name=AGENT*
```

列出与每个命令中的特定名称，状态和类型匹配的代理程序的基本信息。例如：

```
{ "agent": [ { "name": "AGENT1",
               "state": "ready",
               "type": "standard" },
             { "name": "AGENT2",
               "state": "ready",
               "type": "standard" } ]
}
{ "agent": [ { "name": "AGENT1",
               "state": "stopped",
               "type": "standard" },
             { "name": "AGENT2",
               "state": "stopped",
               "type": "standard" } ]
}
{ "agent": [ { "name": "AGENT1",
               "state": "ready",
               "type": "standard" },
             { "name": "AGENT2",
               "state": "ready",
               "type": "standard" } ]
}
```

发出以下命令，`/ibmmq/rest/v1/admin/mft/agent?attributes=general&type=standard` 将列出类型为 *standard* 的所有消息以及 *general* 属性。例如：

```
{
  "agent": [
    { "name": "AGENT1",
      "state": "ready",
      "type": "standard",
      "general": { "description": "Standard connected to the qmgr in client mode",
                  "statusAge": "06:31:00",
                  "version": "9.0.3.0",
                  "level": "p903-L170513",
                  "statusPublicationRate": 300,
                  "statusPublishTime": "2017-10-31T06:57:07.000Z",
                  "maximumQueuedTransfers": 1000,
                }
    }
  ]
}
```

```

        "maximumDestinationTransfers":25,
        "maximumSourceTransfers":25,
        "operatingSystem" : "Windows7" }
    },
    { "name": "AGENT2",
      "state" : "ready",
      "type": "standard"
      "general" : { "description" : "Standard connected to qmgr in Binding mode",
                    "statusAge" : "05:00:00",
                    "version" : "9.0.3.0",
                    "level" : "p903-L170513",
                    "statusPublicationRate" : 300,
                    "statusPublishTime" : "2017-09-13T09:10:09.000Z",
                    "maximumQueuedTransfers" : 1000,
                    "maximumDestinationTransfers":25,
                    "maximumSourceTransfers":25,
                    "operatingSystem" : "Windows7" }
    }
  ]
}

```

V 9.0.4 使用 REST API 进行远程管理

您可以使用 REST API 来管理远程队列管理器以及与这些队列管理器相关联的 IBM MQ 对象。此远程管理包含位于同一系统上但与 mqweb 服务器不在同一 IBM MQ 安装中的队列管理器。因此，您可以使用 REST API 仅通过一个运行 mqweb 服务器的安装来管理整个 IBM MQ 网络。要管理远程队列管理器，必须配置 administrative REST API 网关，以便在与 mqweb 服务器相同的安装中至少有一个队列管理器充当网关队列管理器。然后，可以在 REST API 资源 URL 中指定远程队列管理器以执行指定的管理操作。

开始之前

您可以通过禁用 administrative REST API 网关来阻止远程管理。有关更多信息，请参阅 [配置 administrative REST API 网关](#)。

要使用 administrative REST API 网关，必须满足以下条件：

- 必须配置并启动 mqweb 服务器。有关配置和启动 mqweb 服务器的更多信息，请参阅 [第 63 页的『开始使用 administrative REST API』](#)。
- 要配置为网关队列管理器的队列管理器必须与 mqweb 服务器位于同一安装中。
- 要管理的远程队列管理器必须位于 IBM MQ 8.0 或更高版本。
- 必须确保在请求中指定的任何属性对于要向其发送请求的系统有效。例如，如果网关队列管理器在 Windows 上，远程队列管理器在 z/OS 上，那么您无法请求针对 queue 资源上的 HTTP GET 请求返回 dataCollection.statistics 属性。
- 您必须确保在请求中指定的任何属性对于要向其发送请求的 IBM MQ 级别都有效。例如，如果远程队列管理器正在运行 IBM MQ 8.0，那么您无法请求针对 queue 资源上的 HTTP GET 请求返回 extended.enableMediaImageOperations 属性。
- 必须使用下列其中一个受支持的 REST 资源：

- /queue
- /subscription
- **V 9.0.5** /channel
- **V 9.0.5** /mqsc
- /qmgr

当您查询远程队列管理器时，/qmgr 资源仅返回属性的子集：name，status.started，status.channelInitiatorState，status.ldapConnectionState，status.connectionCount 和 status.publishSubscribeState。

关于此任务

要使用 administrative REST API 网关来管理远程队列管理器，必须准备队列管理器以进行远程管理。即，您必须配置网关队列管理器与远程队列管理器之间的传输队列，侦听器以及发送方和接收方通道。然后，可

以通过在资源 URL 中指定队列管理器来向远程队列管理器发送 REST 请求。通过使用 `setmqweb` 命令将 `mqRestGatewayQmgr` 属性设置为网关队列管理器的名称，或者在随请求一起发送的头中发送网关队列管理器的名称，来指定网关队列管理器。请求通过网关队列管理器发送到远程队列管理器。返回的响应的头指示用作网关队列管理器的队列管理器。

过程

1. 配置网关队列管理器与要管理的远程队列管理器之间的通信。这些配置步骤与通过 `runmqsc` 和 PCF 配置远程管理所需的步骤相同。
有关这些步骤的更多信息，请参阅第 169 页的『从本地队列管理器进行远程管理』。
2. 在远程队列管理器上配置安全性：
 - a) 确保运行远程队列管理器的系统上存在相关用户标识。远程系统上必须存在的用户标识取决于 REST API 用户的角色：
 - 如果 REST API 用户位于 `MQWebAdmin` 或 `MQWebAdminRO` 组中，那么启动 `mqweb` 服务器的用户标识必须存在于远程系统上。在 IBM MQ Appliance 上，启动 `mqweb` 服务器的用户为 `mqsystem`。
 - 如果 REST API 用户位于 `MQWebUser` 组中，那么该 REST API 用户标识必须存在于远程系统上。
 - b) 确保为相关用户标识授予必要的权限级别，以访问远程队列管理器上的相应 REST API 资源：
 - 将消息放入 `SYSTEM.ADMIN.COMMAND.QUEUE` 的权限。
 - 将消息放入 `SYSTEM.REST.REPLY.QUEUE` 的权限。
 - 访问为远程管理定义的传输队列的权限。
 - 显示队列管理器属性的权限。
 - 执行 REST 请求的权限。有关更多信息，请参阅 REST API 资源参考主题的“安全性需求”部分。
3. 配置将哪个本地队列管理器用作网关。您可以配置缺省网关队列管理器，在 HTTP 头中指定网关队列管理器，或者使用两种方法的组合：
 - 使用 `setmqweb` 命令配置缺省网关队列管理器：

```
setmqweb properties -k mqRestGatewayQmgr -v qmgrName
```

其中 `qmgrName` 是网关队列管理器的名称。

当以下两个语句都为 `true` 时，将使用此网关队列管理器：

- 未在 REST 请求的 `ibm-mq-rest-gateway-qmgr` 头中指定队列管理器。
 - REST API 资源 URL 中指定的队列管理器不是本地队列管理器。
- 通过将 HTTP 头 `ibm-mq-rest-gateway-qmgr` 设置为网关队列管理器的名称，在每个 REST 请求上配置网关队列管理器。
4. 在资源 URL 中包含要管理的远程队列管理器的名称。
例如，要从远程队列管理器 `remoteQM` 获取队列列表，请使用以下 URL：

```
https://localhost:9443/ibmmq/rest/v1/admin/qmgr/remoteQM/queue
```

结果

`ibm-mq-rest-gateway-qmgr` 头会随 REST 响应一起返回。此头指定哪个队列管理器用作网关队列管理器。

示例

在以下示例中，两台机器上有三个 IBM MQ 安装。在 Machine 1 上，有一个 Installation 1 和一个 Installation 2。在 Machine 2 上，存在 Installation 3。为 Installation 1 配置了 `mqweb` 服务器。每个安装中都有一个队列管理器，并且这些队列管理器配置为进行远程管理。即，配置并启动以下侦听器，通道和队列：

- 在队列管理器 QM1 上的 Installation 1 中的 Machine 1 上:
 - 发送方通道 QM1.to.QM2
 - 接收方通道 QM2.to.QM1
 - 发送方通道 QM1.to.QM3
 - 接收方通道 QM3.to.QM1
 - 传输队列 QM2
 - 传输队列 QM3
 - 在端口 1414 上配置的侦听器
- 在队列管理器 QM2 上的 Installation 2 中的 Machine 1 上:
 - 发送方通道 QM2.to.QM1
 - 接收方通道 QM1.to.QM2
 - 传输队列 QM1
 - 在端口 1415 上配置的侦听器
- 在队列管理器 QM3 上的 Installation 3 中的 Machine 2 上:
 - 发送方通道 QM3.to.QM1
 - 接收方通道 QM1.to.QM3
 - 传输队列 QM1
 - 缺省侦听器

在 QM2 上定义了队列 Qon2，在 QM3 上定义了队列 Qon3。

用户 mquser 在两台机器上定义，在 REST API 中被授予 MQWebAdmin 角色，并被授予访问每个队列管理器上相应队列的权限。

setmqweb 命令用于将队列管理器 QM1 配置为缺省网关队列管理器。

下图显示了此配置:

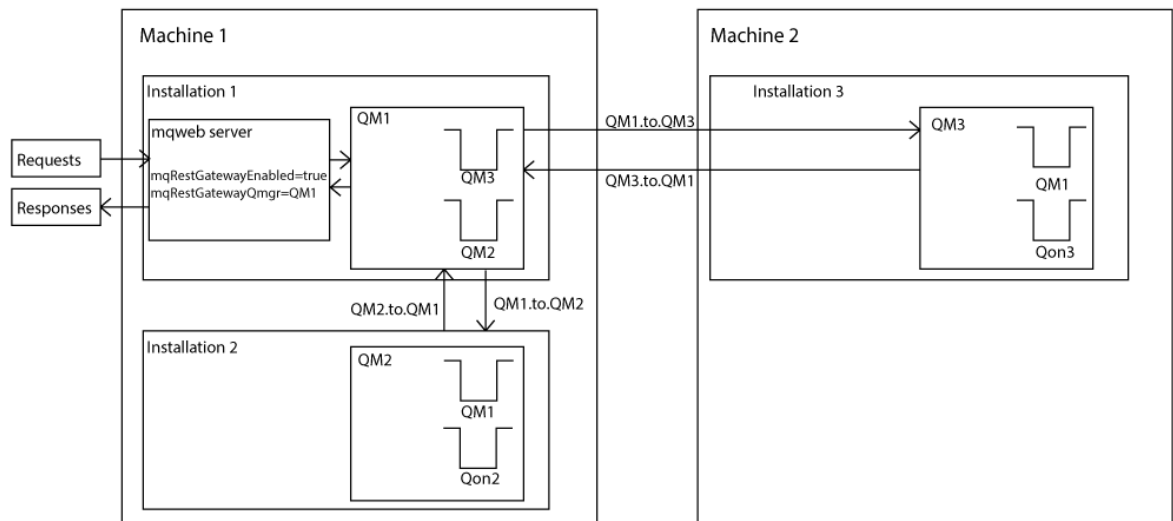


图 15: 使用 REST API 进行远程管理的示例配置的图。

以下 REST 请求将发送到 mqweb 服务器:

```
GET https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM2/queue?
attributes=general.isTransmissionQueue
```

收到以下响应:

```
{
  "queue" :
  [{
    "general": {
      "isTransmissionQueue": true
    },
    "name": "QM1",
    "type": "local"
  },
  {
    "general": {
      "isTransmissionQueue": false
    },
    "name": "Qon2",
    "type": "local"
  }
]
```

以下 REST 请求将发送到 mqweb 服务器:

```
GET https://localhost:9443/ibmmq/rest/v1/admin/qmgr/QM3/queue?
attributes=general.isTransmissionQueue,general.description
```

收到以下响应:

```
{
  "queue" :
  [{
    "general": {
      "isTransmissionQueue": true,
      "description": "Transmission queue for remote admin."
    },
    "name": "QM1",
    "type": "local"
  },
  {
    "general": {
      "isTransmissionQueue": false,
      "description": "A queue on QM3."
    },
    "name": "Qon3",
    "type": "local"
  }
]
```

V 9.0.2 REST API 时间戳记

当 administrative REST API 返回日期和时间信息时, 将以全球标准时间 (UTC) 格式以设置的格式返回该信息。

将以以下时间戳记格式返回日期和时间:

```
YYYY-MM-DDTHH:mm:ss:sssZ
```

例如, 2012-04-23T18:25:43.000Z, 其中 Z 指示时区为全球标准时间 (UTC)。

在 IBM MQ 9.0.2 中, 不保证此时间戳记的准确性。例如, 如果 mqweb 服务器未在与资源 URL 中指定的队列管理器相同的时区中启动, 那么时间戳记可能不准确。此外, 如果需要进行夏令时调整, 那么时间戳记可能不准确。

REST API 错误处理

REST API 通过返回相应的 HTTP 响应代码 (例如 404 (Not Found)) 和 JSON 响应来报告错误。任何不在 200-299 范围内的 HTTP 响应代码都被视为错误。

错误响应格式

响应采用 JSON 格式, 采用 UTF-8 编码。它包含嵌套的 JSON 对象:

- 包含名为 `error` 的单个 JSON 数组的外部 JSON 对象。
- 数组中的每个元素都是一个 JSON 对象，用于表示有关错误的信息。每个 JSON 对象都包含以下属性：

类型

字符串。

错误的类型。

messageId

字符串。

MQWBnnnnX 格式的消息的唯一标识。此标识具有以下元素：

MQWB

显示消息源自 MQ Rest API 的前缀。

nnnn

标识消息的唯一编号。

X

表示消息严重性的单个字母：

- I (如果消息完全是参考消息)。
- W (如果消息警告存在问题)。
- E 如果消息指示发生了错误。
- S 如果消息指示发生了严重错误。

message

字符串。

对错误的描述。

说明

字符串。

错误的说明。

操作

字符串。


可用来解决错误的步骤的描述。

qmgrName

 此字段仅适用于 z/OS，其中队列管理器是队列共享组的成员。您必须已指定 **commandScope** 可选查询参数或 **queueSharingGroupDisposition** 属性。

字符串。

迂到错误的队列管理器的名称。

 此字段不适用于 messaging REST API。

仅当 **type** 为 `pcf`、`java` 或 `rest` 时，此字段才可用。

数字。

与故障关联的 MQ 完成代码。

 **reasonCode**

仅当 **type** 为 `pcf`、`java` 或 `rest` 时，此字段才可用。

数字。

与故障关联的 MQ 原因码。

异常

仅当 **type** 为 `java` 时，此字段才可用。

阵列。

链 Java 或 JMS 异常的数组。异常数组的每个元素都包含一个 **stackTrace** 字符串数组。

stackTrace 字符串数组包含拆分为行的每个异常的详细信息。

队列共享组出错

z/OS

在队列共享组中，可以为某些命令指定可选查询参数 **commandScope**。此参数允许将命令传播到队列共享组中的其他队列管理器。这些命令中的任何一个都可以独立失败，导致队列共享组的某些命令成功，而某些命令失败。

在命令部分失败的情况下，将返回 HTTP 错误代码 500。对于生成故障的每个队列管理器，有关该故障的信息将作为 **error** JSON 数组中的元素返回。对于成功运行该命令的每个队列管理器，将以 **success** JSON 数组中的元素形式返回队列管理器的名称。

示例

- 以下示例显示了对尝试获取有关不存在的队列管理器的信息的错误响应:

```
"error": [
  {
    "type": "rest",
    "messageId": "MQWB0009E",
    "message": "MQWB0009E: Could not query the queue manager 'QM1'",
    "explanation": "The MQ REST API was invoked specifying a queue manager name which cannot be located.",
    "action": "Resubmit the request with a valid queue manager name or no queue manager name, to retrieve a list of queue managers. "
  }
]
```

- z/OS 以下示例显示尝试删除队列共享组中某些队列管理器不存在的队列的错误响应:

```
"error" : [
  {
    "type": "rest",
    "messageId": "MQWB0037E",
    "message": "MQWB0037E: Could not find the queue 'missingQueue' - the queue manager reason code is 3312 : 'MQRCCF_UNKNOWN_OBJECT_NAME'",
    "explanation": "The MQ REST API was invoked specifying a queue name which cannot be located.",
    "action": "Resubmit the request with the name of an existing queue, or with no queue name to retrieve a list of queues.",
    "qmgrName": "QM1"
  },
  {
    "type": "rest",
    "messageId": "MQWB0037E",
    "message": "MQWB0037E: Could not find the queue 'missingQueue' - the queue manager reason code is 3312 : 'MQRCCF_UNKNOWN_OBJECT_NAME'",
    "explanation": "The MQ REST API was invoked specifying a queue name which cannot be located.",
    "action": "Resubmit the request with the name of an existing queue, or with no queue name to retrieve a list of queues.",
    "qmgrName": "QM2"
  }
],
"success" : [{"qmgrName": "QM3"}, {"qmgrName": "QM4"}]
```

MFT 请求的错误

如果未启用 MFT REST API 服务，并且您调用 MFT REST API，那么会收到以下异常:

```
{"error": [{
  "action": "Enable the Managed File Transfer REST API and resubmit the request.",
  "completionCode": 0,
  "explanation": "Managed File Transfer REST calls are not permitted as the service is disabled.",
  "message": "MQWB0400E: Managed File Transfer REST API is not enabled.",
  "msgId": "MQWB0400E",
  "reasonCode": 0,
  "type": "rest"
}]}
```

如果已启用 MFT REST API 服务，并且未在 mqwebuser.xml 文件中设置协调队列管理器，那么您将收到以下异常：

```
{
  "error": [
    {
      "action": "Set the coordination queue manager name and restart the mqweb server.",
      "completionCode": 0,
      "explanation": "Coordination queue manager name must be set before using Managed File Transfer REST services.",
      "message": "MQWB0402E: Coordination queue manager name is not set.",
      "msgId": "MQWB0402E",
      "reasonCode": 0,
      "type": "rest"
    }
  ]
}
```

V 9.0.1 REST API 发现

REST API 的 Documentation 以 IBM Documentation 和 Swagger 格式提供。Swagger 是记录 REST API 的常用方法。可以通过在 mqweb 服务器上启用 API 发现功能来查看 REST API 的 Swagger 文档。

开始之前

必须对 mqweb 服务器启用安全性，才能使用 API 发现来查看 Swagger 文档。有关启用安全性所需的步骤的更多信息，请参阅 [配置 IBM MQ Console 安全性](#)。

过程

1. 在以下某个目录中找到 mqwebuser.xml 文件：

- **ULW** `MQ_DATA_DIRECTORY/web/installations/installationName/servers/mqweb`
- **z/OS** `WLP_user_directory/servers/mqweb`

其中，`WLP_user_directory` 是运行 `crtmqweb.sh` 脚本以创建 mqweb 服务器定义时指定的目录。

2. 将相应的 XML 添加到 mqwebuser.xml 文件：

- 如果 mqwebuser.xml 文件中存在 `<featureManager>` 标记，请在 `<featureManager>` 标记中添加以下 XML：

```
<feature>apiDiscovery-1.0</feature>
```

- 如果 mqwebuser.xml 文件中不存在 `<featureManager>` 标记，请在 `<server>` 标记中添加以下 XML：

```
<featureManager>
  <feature>apiDiscovery-1.0</feature>
</featureManager>
```

3. 使用下列其中一种方法来查看 Swagger 文档：

- 通过在浏览器中输入以下 URL，显示可浏览并试用 REST API 的 Web 页面：

```
https://host:port/ibm/api/explorer
```

除了认证每个请求外，还必须包含每个 POST，PATCH 或 DELETE 请求的 `ibm-mq-rest-csrf-token` 头。

V 9.0.5 此头的必需内容因 IBM MQ 的版本而异：

- 从 IBM MQ 9.0.5 开始，`ibm-mq-rest-csrf-token` HTTP 头需要存在于请求中；其值可以是包括空白在内的任何内容。

- 在 IBM MQ 9.0.5 之前，头的值是 csrfToken cookie 的内容。将 HTTP GET 方法与 REST API 配合使用时，将生成 csrfToken。您可以通过在浏览器的地址栏中输入以下文本来查看 cookie 的内容：

```
javascript:alert(document.cookie)
```

此请求头用于确认用于认证请求的凭证正由凭证的所有者使用。即，令牌用于防止跨站点请求伪造攻击。

- 通过向以下 URL 发出 HTTP GET 来检索描述整个 REST API 的单个 Swagger 2 文档：

```
https://host:port/ibm/api/docs
```

此文档可用于要以编程方式浏览可用 API 的应用程序。

主机

指定可使用 REST API 的主机名或 IP 地址。

缺省值为 localhost。

port

指定 administrative REST API 使用的 HTTPS 端口号。

缺省值为 9443。

如果主机名或端口号已从缺省值更改，那么您可以从 REST API URL 确定正确的值。有关确定 URL 的更多信息，请参阅第 66 页的『[确定 REST API URL](#)』。

V 9.0.4 REST API 本地语言支持

具有特定资格的 REST API 支持将本地语言指定为 HTTP 请求的一部分。

背景

HTTP 头允许对请求指定特定行为，并允许在响应中提供其他信息。

HTTP 头中包含请求以本地语言返回信息的能力。如果可能，REST API 将采用此头。

指定本地语言

在 ACCEPT-LANGUAGE HTTP 头中，可以提供一个或多个语言标记。您可以选择使列组与标记相关联，从而允许指定按首选项排序的列表。[此页面](#)对原则进行了有用的讨论。

REST API 采用此头，从 ACCEPT-LANGUAGE 头中选择语言并以该语言返回消息。当 ACCEPT-LANGUAGE 头不包含 REST API 可支持的语言时，将以缺省语言返回消息。此缺省语言对应于 REST API Web 服务器的缺省语言环境。

第 81 页的『[转换了哪些数据?](#)』部分说明了转换的数据。

指示响应的适用语言

来自 REST API 的响应上的 CONTENT-LANGUAGE HTTP 头指示返回消息的语言。

转换了哪些数据?

错误消息和参考消息已翻译，其他文本未翻译。

- 不会转换从队列管理器返回的数据-例如，在通过 REST API 执行 MQSC 命令的情况下，队列管理器的响应位于队列管理器的语言环境中。
- 为 REST API 生成的 (Swagger) 文档 (通过 apiDiscovery 功能部件公开) 采用英语。

支持哪些语言?

除英语外，REST API 错误和参考消息还会翻译为以下语言。

中文(简体)

由语言标记 zh_CN 表示

繁体中文

由语言标记 zh_TW 表示

捷克语

由语言标记 cs 表示

法语

由语言标记 fr 表示

匈牙利语

由语言标记 hu 表示

意大利语

由语言标记 it 表示

日语

由语言标记 ja 表示

韩语

由语言标记 ko 表示

波兰语

由语言标记 pl 表示

(巴西) 葡萄牙语

由语言标记 pt_BR 表示

俄语

由语言标记 ru 表示

西班牙语

由语言标记 es 表示

示例

在示例中，Web 服务器具有英语缺省语言环境。

指定单个受支持的语言

在请求头中，ACCEPT-LANGUAGE 设置为 fr。此设置指定法语是可翻译文本的首选语言。

在响应头中，CONTENT-LANGUAGE 设置为 fr。此设置指示响应中的错误和参考消息使用法语。

指定语言列表

在请求头中，ACCEPT-LANGUAGE 设置为 am, fr。此设置指定 Amharic 和 French 是可翻译文本的可接受语言，而 Amharic 是可翻译文本的首选语言。

在响应头中，CONTENT-LANGUAGE 设置为 fr。此设置指示响应中的错误和参考消息使用法语，因为 REST API 不支持 Amharic。

指定单个不受支持的语言

在请求头中，ACCEPT-LANGUAGE 设置为 am。此设置指定 Amharic 是可翻译文本的首选语言。

在响应头中，CONTENT-LANGUAGE 设置为 en。此设置指示响应中的错误和参考消息为英语，因为 REST API 不支持 Amharic。

V 9.0.1 使用 IBM MQ Console 进行管理

IBM MQ Console 是基于 Web 的用户界面，可用于执行一些常见管理任务。

开始之前

注:

- 使用 IBM MQ Console 时，请不要在任何队列管理器中禁用命令服务器。如果针对某个队列管理器禁用了命令服务器，那么 IBM MQ Console 会变得无响应，并且命令处理会有较长的延迟。向禁用了命令服务器的队列管理器发出的任何命令都会出现超时。

- 连接到 IBM MQ Console 后，浏览器会尝试使用自己设置的语言环境来查看 IBM MQ Console。系统会执行检查以确定 IBM MQ Console 是否支持指定的语言。如果未找到语言文件，那么将使用美国英语，并且 mqweb 服务器会记录“未找到文件”错误。因此，当浏览器设置的语言不受 IBM MQ Console 支持时，预期会出现以下消息：

```
SRVE0190E: File not found: /nls/en_GB/labels.json
SRVE0190E: File not found: /nls/en_GB/pcf.json
SRVE0190E: File not found: /nls/en_GB/errors.json
SRVE0190E: File not found: /nls/en_GB/strings.json
```

过程

- [使用本地队列管理器](#)
- [使用 IBM MQ 对象](#)
- [使用权限记录](#)
- [监视系统资源使用情况](#)
- [配置仪表板布局](#)

V 9.0.1 开始使用 IBM MQ Console

必须先安装正确的组件并启动 mqweb 服务器，然后才能启动 IBM MQ Console。然后，您才可以在浏览器中启动 IBM MQ Console。

开始之前

IBM i 在 IBM i 上，命令应该在 QSHELL 中运行。

关于此任务

此任务的过程主要包括快速开始使用 IBM MQ Console 的基本步骤。有关定制配置的更多信息，请参阅“后续操作”下列出的链接。

注：在安装期间，您可以选择在安全性没有安全性的情况下配置 IBM MQ Console。

过程

1. 安装 IBM MQ Console 和 REST API 组件：

- **AIX** **V 9.0.4** 在 AIX 上，安装 mqm.web.rte 文件集。
- **Linux** 在 Linux 上，安装 MQSeriesWeb 组件。有关在 Linux 上安装组件和功能部件的更多信息，请参阅 [Linux 安装任务](#)。
- **Windows** 在 Windows 上，安装 Web Administration 功能部件。有关在 Windows 上安装组件和功能部件的更多信息，请参阅 [Windows 安装任务](#)。
- **z/OS** 在 z/OS 上，安装 IBM MQ for z/OS Unix System Web Services Components 功能部件。有关在 z/OS 上安装组件和功能部件的更多信息，请参阅 [z/OS 安装任务](#)。

2. 配置基本安全性，以允许用户和组访问 IBM MQ Console。




- 将样本 basic_registry.xml 文件从 MQ_INSTALLATION_PATH/web/mq/samp/configuration 目录复制到：
 - **ULW** 在 UNIX, Linux, and Windows 上：MQ_DATA_DIRECTORY/web/installations/installationName/servers/mqweb
 - **z/OS** 在 z/OS 上：WLP_user_directory/servers/mqweb

其中，WLP_user_directory 是在运行 **crtmqweb.sh** 脚本来创建 mqweb 服务器定义时指定的目录。

b) 将该样本 XML 文件重命名为 `mqwebuser.xml`。

注: 这个重命名的文件将替换 administrative REST API 也在使用的现有文件。因此, 如果更改了 administrative REST API 的 `mqwebuser.xml` 文件, 请将更改复制到新的 XML 文件, 然后再将其重命名。

3. 根据平台, 启用与 mqweb 服务器的远程连接:

-  (可选) 在 Linux 上
-  (可选) 在 Windows 上
-  在 z/OS 上
- 从 IBM MQ 9.0.4, 使用 **setmqweb** 命令:
`setmqweb properties -k httpHost -v hostname`
- 从 IBM MQ 9.0.1 开始, 将以下 XML 添加到 `mqwebuser.xml` 文件中的 `<server>` 标记内:
`<variable name="httpHost" value="hostname"/>`

其中, `hostname` 指定 IP 地址、带域名后缀的域名服务器 (DNS) 主机名或安装了 IBM MQ 的服务器的 DNS 主机名。使用星号 * 可指定所有可用的网络接口。



注意:  

在 z/OS 上发出 **setmqweb** 或 **dspmqweb** 命令之前, 必须设置 `WLP_USER_DIR` 环境变量, 以便此变量指向 mqweb 服务器配置。

为此, 请发出以下命令:




```
export WLP_USER_DIR=WLP_user_directory
```

其中 `WLP_user_directory` 是传递到 `crtmqweb.sh` 的目录的名称。例如:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

有关更多信息, 请参阅[创建 Liberty 服务器定义](#)。

4. 启动支持 IBM MQ Console 的 mqweb 服务器:

-   在 Windows 和 Linux 上, 作为 [特权用户](#), 在命令行上输入以下命令:
`strmqweb`
-  在 z/OS 上, 启动在[任务 29: 为 IBM WLP 服务器创建过程中创建的过程](#)。

5. 通过在浏览器中输入以下 URL 来连接到 IBM MQ Console :

`https://hostname:9443/ibmmq/console`

其中 `hostname` 指定具有域名后缀的 IP 地址或域名服务器 (DNS) 主机名, 或者指定安装了 IBM MQ 的服务器的 DNS 主机名。如果未在步骤 3 中配置远程连接, 那么 `hostname` 的值为 `localhost`。

6. 登录到 IBM MQ Console。从 IBM MQ 9.0.2 开始, 使用用户名 `mqadmin` 和密码 `mqadmin`。在 IBM MQ 9.0.1 处, 使用用户名 `admin` 和密码 `admin`。

下一步做什么

- 有关配置 IBM MQ Console 安全性的更多信息 (包括如何配置用户和组、LDAP 和客户机证书), 请参阅[配置 IBM MQ Console 安全性](#)。
- 有关配置 IBM MQ Console 设置 (包括启用 HTTP 连接) 的更多信息, 请参阅[配置 IBM MQ 控制台](#)。
- 有关确定 URL 是否为非缺省 URL 的更多信息, 请参阅[第 85 页的『确定 IBM MQ Console URL』](#)。

V 9.0.1 确定 IBM MQ Console URL

用于访问 IBM MQ Console 的缺省 URL 是 `https://localhost:9443/ibmmq/console`。如果更改了缺省的 HTTP 主机或端口，或者已启用 HTTP 端口，那么可以使用 `dspmqweb` 命令来确定该 URL。

关于此任务

V 9.0.4 从 IBM MQ 9.0.4 开始，您可以使用 `dspmqweb status` 命令来确定 Windows、Linux 和 z/OS 上的 IBM MQ Console URL。对于 IBM MQ 9.0.3 和更低版本，在 Windows 和 Linux 上，可以使用 `dspmqweb` 命令。在 z/OS 上，必须搜索 `messages.log` 文件以确定 URL。

过程

V 9.0.4

请使用以下某种方法来确定该 URL：

- 从 IBM MQ 9.0.4 开始，以 特权用户 身份使用 `dspmqweb status` 命令：
 - 通过在命令行处输入 `strmqweb` 命令来确保 mqweb 服务器正在运行。
 - 通过在命令行上输入 `dspmqweb status` 命令来显示 URL。
- 对于 IBM MQ 9.0.3 和更低版本，请使用下列其中一种方法：

- **Windows** / **Linux** 在 Windows 或 Linux 上，以 特权用户 身份使用 `dspmqweb` 命令：

- 通过在命令行处输入 `strmqweb` 命令来确保 mqweb 服务器正在运行。
- 通过在命令行上输入 `dspmqweb` 命令来显示 URL。

- **z/OS** 在 z/OS 上，使用 `messages.log` 文件来查找 URL：

- 打开 `messages.log` 文件。

`messages.log` 文件位于 `WLP_user_directory/servers/mqweb/logs` 路径上，其中 `WLP_user_directory` 是运行 `crtmqweb.sh` 脚本以创建 mqweb 服务器定义时指定的目录。

- 搜索结尾为 `ibmmq/console` 的最新 CWWKT0016I 消息代码。该 URL 包含在此消息中。

注：IBM MQ Console 使用 `/ibmmq/console/internal` 的内部 URL。此 URL 出现在某些 CWWKT0016I 消息中，但仅供 IBM MQ Console 使用。

示例

以下示例显示了将此 URL 作为缺省 URL 的消息代码 CWWKT0016I：

```
A CWWKT0016I: Web application available (default_host): https://localhost:9443/ibmmq/console
```

V 9.0.1 z/OS 上的限制

使用 IBM MQ Console 管理 z/OS 上的队列管理器时，以下限制适用。

- 无法创建、删除、启动或停止 z/OS 上的队列管理器。
- 无法启动或停止 z/OS 上的通道启动程序，并且不显示通道启动程序状态。
- 无法显示或管理侦听器。
- 只能使用 CHLDISP(DEFAULT) 发出下列命令：启动、ping、解析和重置通道命令。
- 只能使用 QSGDISP(QMGR) 创建新对象。
- 无法显示或管理通过 QSGDISP(GROUP) 定义的对象。
- 无法管理队列管理器安全性。
- 无法监视系统资源使用情况。

相关任务

第 82 页的『使用 IBM MQ Console 进行管理』

IBM MQ Console 是基于 Web 的用户界面，可用于执行一些常见管理任务。

相关信息

使用本地队列管理器

V 9.0.1 使用本地队列管理器

您可以在 IBM MQ Console 中使用“本地队列管理器”窗口小部件来创建、配置和控制本地队列管理器。

关于此任务

“本地队列管理器”窗口小部件列出了添加到正在运行 IBM MQ Console 的 IBM MQ 安装。将不会列出与同一系统上不同 IBM MQ 安装相关联的队列管理器。可以从列表中选择要使用的单个队列管理器。

注: **V 9.0.4** IBM MQ Console 不支持复制的数据队列管理器 (RDQM)。

您可以通过单击 **添加窗口小部件** **V 9.0.5** **Add widget** 将本地队列管理器窗口小部件添加到仪表板。然后，选择**本地队列管理器**。

V 9.0.5 您可以在创建窗口小部件之后对其进行重命名。将鼠标指针悬停在标题栏中，将显示重命名图



标。单击该图标，在“重命名窗口小部件”窗口中输入新名称，然后单击**重命名**。

z/OS 无法在 z/OS 上创建、启动、停止或删除队列管理器。

过程

- 要创建新本地队列管理器：

- a) 单击本地队列管理器窗口小部件工具栏中的“创建”图标 **V 9.0.5** **Create +**。
- b) 输入新队列管理器的名称。该名称最多可以包含 48 个字符。有效字符包括字母和数字以及“。”，“/”，“_”和“%”个字符。
- c) 可选：输入可供队列管理器用于侦听的 TCP/IP 端口。此端口号不能超过 65535。
- d) 单击**创建**。此时会创建并启动新的队列管理器。

- 要启动本地队列管理器：

- a) 在“本地队列管理器”窗口小部件中，从列表中选择要启动的队列管理器。

- b) 单击本地队列管理器窗口小部件工具栏中的“启动”图标 **V 9.0.5** **Start**。

- 要停止本地队列管理器：

- a) 在“本地队列管理器”窗口小部件中，从列表中选择要停止的队列管理器。

- b) 单击本地队列管理器窗口小部件工具栏中的“停止”图标 **V 9.0.5** **Stop**。

- c) 单击**停止**以确认您要停止该队列管理器。

- 要删除本地队列管理器：

- a) 在“本地队列管理器”窗口小部件中，从列表中选择要删除的队列管理器。

- b) 如果队列管理器正在运行，请将其停止。

- c) 单击本地队列管理器窗口小部件工具栏中的“删除”图标 。
- d) 单击**删除**以确认要删除该队列管理器。此时会删除该队列管理器及所有相关对象。
- 要查看和编辑本地队列管理器的属性：
 - a) 确保队列管理器正在运行，并在队列管理器列表中将其选中。
 - b) 单击本地队列管理器窗口小部件工具栏中的属性图标 。或者，双击队列管理器。
 - c) 查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令行进行编辑。有关属性的信息，您可以在 MQ Explorer 文档中的 [队列管理器属性](#) 中查看属性信息。
- 要刷新本地队列管理器的安全性：
 - a) 确保本地队列管理器正在运行，并且在队列管理器列表中将其选中。
 - b) 选择 ... > **刷新安全**
 - c) 选择要刷新的队列管理器安全性：
 - 选择**授权服务**以刷新授权服务组件内部保存的权限列表。
 - 选择**连接认证**以刷新连接认证配置的高速缓存视图。
 - 选择**SSL**以刷新 SSL 或 TLS 密钥存储库的高速缓存视图。此选项还会刷新用于经认证的撤销列表的 LDAP 服务器位置，以及任何加密硬件参数。
- 要使用本地队列管理器的权限记录：
 - a) 确保本地队列管理器正在运行，并且在队列管理器列表中将其选中。
 - b) 请选择下列其中一个选项：
 - 选择 ... > **管理权限记录** 以处理队列管理器的权限记录，并指定用户组可以执行的操作。
 - 选择 ... > **管理创建权限记录** 以处理队列管理器的创建权限记录，并指定用户组可以在该队列管理器上创建的对象组。
- 要为本地队列管理器自动创建仪表板选项卡：
 - a) 在“本地队列管理器”窗口小部件中选择队列管理器。
 - b) 选择 ... > **添加新的仪表板选项卡**
此时会创建新仪表板选项卡。此选项卡包含队列管理器的名称。
- 要过滤本地队列管理器列表：
 - a) 在搜索框中输入过滤器文本。
 - b) 要停止过滤，请从搜索框中删除该文本。

使用 IBM MQ 对象

您可以使用 IBM MQ Console 中的 IBM MQ 对象窗口小部件来处理不同类型的 IBM MQ 对象。


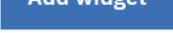
关于此任务

每个“IBM MQ 对象”窗口小部件包含与特定队列管理器相关联的对象。您可以将以下类型的“IBM MQ 对象”窗口小部件添加到仪表板中：

- “队列”窗口小部件
- “主题”窗口小部件
- “侦听器”窗口小部件
- “通道”窗口小部件
- “客户机连接通道”窗口小部件
- “认证信息”窗口小部件
- “预订”窗口小部件

过程


- 要创建“IBM MQ 对象”窗口小部件：

- 单击 **添加窗口小部件**  。
- 从列表中选择相应的队列管理器。
- 单击要创建的对象窗口小部件的类型名称。

- 


- 要重命名对象窗口小部件：



- 将鼠标指针悬停在标题栏中，将显示重命名图标 。单击该图标。
- 在“重命名”窗口小部件窗口中输入新名称，然后单击**重命名**。



- 



- 要配置“IBM MQ 对象”窗口小部件：

- 单击该窗口小部件标题栏中的配置图标 。
- 可选：指定要显示其 IBM MQ 对象的队列管理器。
- 可选：指定是显示还是隐藏系统对象。
- 单击**保存**。

- 要过滤该窗口小部件中显示的对象：

- 在搜索框中输入过滤器文本。
- 要停止过滤，请从搜索框中删除该文本。

- 要刷新窗口小部件的内容，请单击窗口小部件标题栏中的刷新图标  。

- 要移除窗口小部件，请单击窗口小部件标题栏中的移除图标  。

使用队列


您可以在 IBM MQ Console 中使用“队列”窗口小部件显示特定队列管理器存在的队列。然后，您可以添加和删除队列、添加和清除队列中的消息、浏览消息、查看和设置队列属性，以及管理队列的权限记录。

开始之前

必须先创建“队列”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。



关于此任务








“队列”窗口小部件将列出特定队列管理器存在的队列。可以从列表中选择要使用的单个队列。

 无法在 z/OS 上查看或编辑队列的权限记录。

过程

- 要添加队列：

- 单击“队列”窗口小部件工具栏中的“创建”图标  。
- 输入队列的名称。有效字符包括字母和数字以及“.”、“/”、“_”和“%”个字符。

- c) 选择要添加的队列类型。
- d) 单击**创建**。此时会创建新队列。
- 要将消息放入队列中：
 - a) 在“队列”窗口小部件中，从列表中选择要添加消息的队列。无法选择模型队列。
 - b) 单击“队列”窗口小部件工具栏中的“放入消息”图标 。
 - c) 输入要放入队列中的消息。
 - d) 单击**放入**。
- 要清除队列中的消息：
 - a) 在“队列”窗口小部件中，从列表中选择要清除消息的本地队列。
 - b) 选择  > **清除队列**。
 - c) 单击**清除队列**以确认您要清除队列。
- 要浏览队列中的消息：
 - a) 在“队列”窗口小部件中，从列表中选择要浏览的本地队列或别名队列。
 - b) 单击“队列”窗口小部件工具栏中的“浏览”图标 。此时会打开“浏览消息”窗口，其中会显示队列中的消息。
- 要删除队列：
 - a) 在“队列”窗口小部件中，从列表中选择要删除的队列。
 - b) 单击“队列”窗口小部件工具栏中的“删除”图标 。
 - c) 可选：如果队列中包含消息，那么可单击**清除队列**以确认可清除该队列。
 - d) 单击**删除**以确认您要删除该队列。此时会删除该队列。
- 要查看和编辑队列的属性：
 - a) 在“队列”窗口小部件中选择队列。
 - b) 单击“队列”窗口小部件工具栏中的“属性”图标 。或者，双击队列。
 - c)  查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令进行编辑。有关属性的信息，请查看 MQ Explorer 文档中的 [队列属性](#) 中的属性信息。
- 要查看和编辑队列的权限记录：
 - a) 在该窗口小部件中选择队列。
 - b) 单击  > **管理权限记录**。
权限记录会显示用户和管理员在所选队列上具有的许可权。

使用主题


您可以在 IBM MQ Console 中使用“主题”窗口小部件来添加和删除主题以及查看和设置主题的属性。

开始之前

必须先创建“主题”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。



关于此任务

“主题”窗口小部件将列出特定队列管理器存在的主题。可以从列表中选择要使用的单个主题。



 无法在 z/OS 上查看或编辑主题的权限记录。

过程

- 要添加主题：

- a) 单击 "主题" 窗口小部件工具栏中的 "创建" 图标  。
- b) 输入新主题的名称。有效字符包括字母和数字以及“。”，“/”，“_”和“%”个字符。
- c) 指定要将主题消息发布到的主题字符串。有关更多信息，请参阅 [主题属性](#)。
- d) 单击**创建**。此时会创建新主题。



- 要删除主题：

- a) 在“主题”窗口小部件中，从列表中选择要删除的主题。
- b) 单击 "主题" 窗口小部件工具栏中的 "删除" 图标  。
- c) 单击**删除**以确认您要删除该主题。此时会删除该主题。



- 要查看和编辑主题的属性：

- a) 在“主题”窗口小部件中选择主题。
- b) 单击 "主题" 窗口小部件工具栏中的 "属性" 图标  。或者，双击主题。
- c)  查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令行进行编辑。有关属性的信息，请查看 MQ Explorer 文档中的 [主题属性](#) 中的属性信息。


- 要发布有关主题的消息：

- a) 单击 "主题" 窗口小部件工具栏中的 "放入消息" 图标  。
- b) 在**消息**字段中输入消息。
- c) 在**主题字符串**字段中输入要将消息发布到的主题字符串。
- d) 单击**发布**。

- 要预订主题：

- a) 单击 "主题" 窗口小部件工具栏中的 "预订" 图标  。
- b) 在**主题字符串**字段中输入要预订的主题字符串。
- c) 单击**预订**。

- 要查看和编辑主题的权限记录：

- a) 在“主题”窗口小部件中选择主题。
- b) 单击  ... > **管理权限记录**。
权限记录会显示用户和管理员在所选主题上具有的许可权。

使用侦听器

您可以在 IBM MQ Console 中使用“侦听器”窗口小部件来添加和删除侦听器、启动和停止侦听器、查看和设置侦听器的属性，以及管理侦听器的权限记录。

开始之前

必须先创建“侦听器”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。

关于此任务

“侦听器”窗口小部件将列出特定队列管理器存在的侦听器。可以从列表中选择要使用的单个侦听器。

 无法在 z/OS 上使用“侦听器”窗口小部件。

过程

- 要添加 TCP/IP 侦听器：
 - a) 单击“侦听器”窗口小部件工具栏中的“创建”图标 。
 - b) 输入侦听器的名称。有效字符包括字母和数字以及“。”，“/”，“_”和“%”个字符。
 - c) 输入侦听器可用的 TCP/IP 端口。此端口号不能超过 65535。
 - d) 单击**创建**。此时会创建新侦听器。
- 要删除侦听器：
 - a) 在“侦听器”窗口小部件中，从列表中选择要删除的侦听器。
 - b) 单击“侦听器”窗口小部件工具栏中的“删除”图标 。
 - c) 单击**删除**以确认您要删除该侦听器。此时会删除该侦听器。
- 要启动侦听器：
 - a) 在“侦听器”窗口小部件中，从列表中选择要启动的侦听器。
 - b) 单击“侦听器”窗口小部件工具栏中的“启动”图标 。
- 要停止侦听器：
 - a) 在“侦听器”窗口小部件中，从列表中选择要停止的侦听器。
 - b) 单击“侦听器”窗口小部件工具栏中的“停止”图标 。
 - c) 单击**停止**以确认您要停止该侦听器。
- 要查看和编辑侦听器的属性：
 - a) 在“侦听器”窗口小部件中选择侦听器。
 - b) 单击“侦听器”窗口小部件工具栏中的“属性”图标 。或者，双击侦听器。
 - c)  查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令行进行编辑。有关属性的信息，请查看 MQ Explorer 文档中的 [侦听器属性](#) 中的属性信息。
- 要查看和编辑侦听器的权限记录：
 - a) 在“侦听器”窗口小部件中选择侦听器。
 - b) 单击  ... > **管理权限记录**。权限记录将显示用户和管理员在所选侦听器上具有的许可权。

使用通道


您可以在 IBM MQ Console 中使用“通道”窗口小部件来添加和删除通道、启动和停止通道、重置和解析通道，以及对通道执行 ping 操作。您还可以查看和设置通道的属性以及管理通道的权限记录。

开始之前

必须先创建“通道”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。


关于此任务

“通道”窗口小部件将列出特定队列管理器存在的通道。可以从列表中选择要使用的单个通道。


 无法在 z/OS 上查看或编辑通道的权限记录。

过程


- 要添加通道：

- a) 单击“通道”窗口小部件工具栏中的“创建”图标 。
- b) 输入通道的名称。有效字符包括字母和数字以及“。”，“/”，“_”和“%”个字符。
- c) 选择要添加的通道类型。
- d) 如果您正在创建发送方、集群发送方或请求者通道，请指定连接名称。此连接名称是用于托管目标队列管理器的计算机的名称。此名称的格式为 *computer_name*(*port_number*)。 *computer_name* 是用于托管目标队列管理器的计算机的名称或 IP 地址， *port_number* 是目标队列管理器的侦听器正在使用的端口。
- e) 如果您正在创建发送方通道或服务器通道，请指定与通道接收方一端的队列管理器相对应的传输队列。
- f) 单击**创建**。此时会创建新通道。


- 要删除通道：

- a) 在“通道”窗口小部件中，从列表中选择要删除的通道。
- b) 单击窗口小部件工具栏中的“删除”图标 。
- c) 单击**删除**以确认您要删除该通道。此时会删除该通道。



- 要启动通道：

- a) 在“通道”窗口小部件中，从列表中选择要启动的通道。
- b) 单击窗口小部件工具栏中的“开始”图标 。


- 要停止通道：



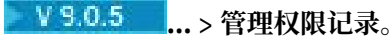
- a) 在“通道”窗口小部件中，从列表中选择要停止的通道。
- b) 单击窗口小部件工具栏中的“停止”图标 。
- c) 单击**停止**以确认您要停止该通道。

- 要查看通道的属性：

- a) 在“通道”窗口小部件中选择通道。
- b) 单击“通道”窗口小部件工具栏中的“属性”图标 。或者，双击通道。
- c) 
查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令行进行编辑。有关属性的信息，请查看 MQ Explorer 文档中的 [通道属性](#) 中的属性信息。

- 要重置通道：

- a) 在“通道”窗口小部件中选择通道。
- b) 单击  ... > **重置**。
- c) 指定消息序号。
如果由于两端对于要发送的下一条消息的序号未达成一致意见而导致未启动通道，那么需要重置通道。消息序号将指定该编号。
- d) 单击**重置通道**。

- 要解析通道：
 - a) 在“通道”窗口小部件中选择通道。
 - b) 单击 。
 - c) 单击**落实**或**回退**以选择是落实还是回退当前消息批次。
- 要对通道执行 ping 操作：
 - a) 在“通道”窗口小部件中选择通道。
 - b) 单击 。
- 要查看或编辑通道的权限记录：
 - a) 在该窗口小部件中选择通道。
 - b) 单击 。
权限记录将显示用户和管理员在所选通道上具有的许可权。

使用客户机连接通道


您可以在 IBM MQ Console 中使用“客户机连接通道”窗口小部件来添加和删除队列管理器中的客户机连接通道、查看和设置属性，以及管理通道的权限记录。

开始之前




必须先创建“客户机连接通道”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。

关于此任务

“客户机连接通道”窗口小部件将列出特定队列管理器存在的客户机连接通道。可以从列表中选择要使用的单个客户机连接通道。

 无法在 z/OS 上查看或编辑客户机连接通道的权限记录。

过程


- 要添加客户机连接通道：
 - a) 单击“客户机连接通道”窗口小部件工具栏中的“创建”图标 。
 - b) 输入客户机连接通道的名称。有效字符包括字母和数字以及“。”，“/”，“_”和“%”个字符。
 - c) 指定连接名称。此连接名称是用于托管目标队列管理器的计算机的名称。其格式为 `computer_name(port_number)`，其中 `computer_name` 是用于托管目标队列管理器的计算机的名称或 IP 地址，`port_number` 是目标队列管理器的侦听器正在使用的端口。
 - d) 单击**创建**。此时会创建新的客户机连接通道。
- 要删除客户机连接通道：
 - a) 在“客户机连接通道”窗口小部件中，从列表中选择要删除的客户机连接通道。
 - b) 单击窗口小部件工具栏中的“删除”图标 。
 - c) 单击**删除**以确认您要删除该客户机连接通道。此时会删除该客户机连接通道。
- 要查看和编辑客户机连接通道的属性：
 - a) 在“客户机连接通道”窗口小部件中选择客户机连接通道。
 - b) 单击“客户机连接通道”窗口小部件工具栏中的属性图标 。或者，双击客户机连接通道。

c)

查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令进行编辑。有关属性的信息，请查看 MQ Explorer 文档中的 [通道属性](#) 中的属性信息。

- 要查看和编辑客户机连接通道的权限记录：

a) 在“客户机连接通道”窗口小部件中选择客户机连接通道。

b) 单击  ... > **管理权限记录**。权限记录将显示用户和管理员在所选客户机连接通道上具有的许可权。

使用认证信息

您可以在 IBM MQ Console 中使用“认证信息”窗口小部件来添加和删除队列管理器中的认证信息对象。您还可以查看和设置属性以及管理对象的权限记录。

开始之前

必须先创建“认证信息”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。

关于此任务

“认证信息”窗口小部件将列出特定队列管理器存在的认证信息。可以从列表中选择要使用的单条认证信息。


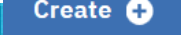
队列管理器认证信息是 IBM MQ 传输层安全性 (TLS) 支持的一部分。这些对象中包含了在 LDAP 服务器上使用 OCSP 或证书撤销列表 (CRL) 执行证书撤销检查所需的定义，以及启用用户标识和密码检查所需的定义。



无法在 z/OS 上使用 IDPW LDAP 或者查看/编辑认证信息对象的权限记录。

过程

- 要添加认证信息对象：

a) 单击“认证信息”窗口小部件工具栏中的“创建”图标  。

b) 指定认证信息对象的名称。有效字符包括字母和数字以及“。”，“/”，“_”和“%”个字符。

c) 指定认证信息对象的类型。



d) 指定该对象类型的其他适用信息：

- 针对 **CRL LDAP**，请指定 **LDAP 服务器名称**。此名称是正在运行 LDAP 服务器的主机的主机名、IPv4 点分十进制地址或 IPv6 十六进制表示法（带有可选端口号）。
- 针对 **OCSP**，请指定 **OCSP 响应程序 URL**。此 URL 是用于检查证书撤销的响应程序 URL。此值必须是包含 OCSP 响应程序主机名和端口号的 HTTP URL。如果 OCSP 响应程序正在使用端口 80（这是 HTTP 的缺省端口），那么可以省略端口号。在 RFC 1738 中定义了 HTTP URL。
- 对于 **IDPW OS**，没有额外的需求。
- 针对 **IDPW LDAP**，请指定 **LDAP 服务器名称和简短用户名**。LDAP 服务器名称是正在运行 LDAP 服务器的主机的主机名、IPv4 点分十进制地址或 IPv6 十六进制表示法（带有可选端口号）。简短用户名是 LDAP 用户记录中用作连接简短名称的字段。

e) 单击**创建**。

- 要删除认证信息对象：

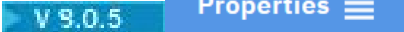

a) 在该窗口小部件中，从列表中选择要删除的认证信息对象。

b) 单击窗口小部件工具栏中的“删除”图标  。

c) 单击**删除**以确认您要删除该认证信息对象。此时会删除该对象。

- 要查看和编辑认证信息对象的属性：

a) 在该窗口小部件中选择认证信息对象。

- b) 单击窗口小部件工具栏中的属性图标 。或者，双击认证信息对象。
- c) 查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令行进行编辑。
- 要查看和编辑认证信息对象的权限记录：
 - a) 在“认证信息”窗口小部件中选择认证信息对象。
 - b) 单击  ... > **管理权限记录**。权限记录将显示用户和管理员在所选认证信息对象上具有的许可权。

使用预订

您可以在 IBM MQ Console 中使用“预订”窗口小部件来添加和删除队列管理器中的预订、查看和设置属性，以及管理预订的权限记录。

开始之前

必须先创建“预订”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。



关于此任务

预订将发布到队列管理器中，并包含订户希望接收的发布的相关信息：

- 订户感兴趣的主体字符串；如果使用了通配符，此主题可以解析为多个主体字符串。
- 要应用于已发布消息的可选的选择字符串。
- 要将所选发布放入的队列的名称。

有关预订的更多信息，请参阅订户和预订以及 [DEFINE SUB](#)。

过程

- 要添加预订对象：
 - a) 单击“预订”窗口小部件工具栏中的“创建”图标 。
 - b) 指定对象的名称。有效字符包括字母和数字以及“。”，“/”，“_”和“%”个字符。
 - c) 选择受管项或提供项作为目标类。如果选择受管项，那么会在本地队列管理器中创建目标。
 - d) 如果选择提供项作为目标类，请在目标字段中指定要将此预订的消息转发到的队列的名称。
 - e) 在主题字符串字段中，指定要预订的主题字符串。
 - f) 选择通配符使用设置。选择字符级通配符以指定通配符表示部分字符串。选择主题级通配符以指定通配符表示部分主题层次结构。
 - g) 选择范围。选择全部，即可将预订转发到通过发布/预订集合体或层次结构直接连接的所有队列管理器。选择队列管理器，即可使预订仅转发此队列管理器中就主题发布的信息。
 - h) 可选：指定选择器。选择字符串是一个表达式，它应用于发布中，用于确定其是否与某个预订匹配。
 - i) 单击创建。
- 要删除预订对象：
 - a) 在“预订”窗口小部件中，从列表中选择要删除的预订对象。
 - b) 单击窗口小部件工具栏中的“删除”图标 。
 - c) 单击删除以确认您要删除该预订对象。此时会删除该对象。
- 要查看和编辑预订对象的属性：

a) 在该窗口小部件中选择预订对象。

b) 单击窗口小部件工具栏中的属性图标 。或者，双击预订对象。

c) 查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令行进行编辑。

使用通道认证记录

您可以在 IBM MQ Console 中使用“通道认证记录”窗口小部件来添加和删除队列管理器中的通道认证记录。您还可以查看和设置通道认证记录的属性。

开始之前



必须先创建“通道认证记录”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。

关于此任务

要在通道级别更精确地控制向连接系统授予的访问权，可以使用通道认证记录。

为确保安全，您可以使用“阻止”通道认证记录来阻止访问通道。您还可以使用“地址映射”通道认证记录以允许访问指定的用户。要了解有关通道认证记录的更多信息，请参阅 [通道认证记录](#)。

过程

- 要添加具有 SSL/TLS 专有名称身份的通道认证记录，请参阅第 96 页的『创建具有 SSL/TLS 专有名称身份的通道认证记录』。
- 要添加具有客户机应用程序用户标识身份的通道认证记录，请参阅第 97 页的『创建具有客户机应用程序用户标识身份的通道认证记录』。
- 要添加具有远程队列管理器名称身份的通道认证记录，请参阅第 98 页的『创建具有远程队列管理器名称身份的通道认证记录』。
- 要添加具有地址身份的通道认证记录，请参阅第 99 页的『创建具有 IP 地址身份的通道认证记录』。
- 要删除通道认证记录：
 - a) 在“通道认证记录”窗口小部件中，从列表中选择要删除的通道认证记录。
 - b) 单击窗口小部件工具栏中的“删除”图标 。
 - c) 单击删除以确认您要删除该通道认证记录。此时会删除该通道认证记录。
- 要查看和编辑通道认证记录的属性：
 - a) 在“通道认证记录”窗口小部件中，从列表中选择要编辑的通道认证记录。
 - b) 单击窗口小部件工具栏中的属性图标 。或者，双击通道认证记录。
 - c) 查看属性并根据需要进行编辑。如果禁用了属性文本框，那么该属性为只读属性，或者只能通过命令行进行编辑。

创建具有 SSL/TLS 专有名称身份的通道认证记录


您可以使用“通道认证记录”窗口小部件来创建具有 SSL/TLS 专有名称身份的“允许”、“阻止”和“警告”通道认证记录。SSL/TLS 专有名称身份将与提供包含指定专有名称的 SSL 或 TLS 个人证书的用户进行匹配。

开始之前

必须先创建“通道认证记录”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。

过程

- 要添加通道认证记录:

- 单击 "通道认证记录" 窗口小部件工具栏中的 "创建" 图标 。
- 选择**规则类型**以指示通道认证记录所需的规则类型:
 - 选择**允许**可允许访问入站连接。
 - 选择**阻止**可阻止访问入站连接。
 - 选择**警告**可在访问要阻止访问的入站连接时发出警告。允许访问连接，但会报告一条错误消息。配置事件时，系统会创建一条错误消息来显示要阻止访问的内容的详细信息。仅报告匹配的规则。
- 从列表中选择 **SSL/TLS 专有名称** 身份类型。
- 单击**下一步**。
- 指定**通道概要文件**。

通道概要文件是要为其设置通道认证的通道或通道集的名称。概要文件可包含通配符，这样便可以阻止一系列通道。例如，概要文件 `alphadelta*` 将阻止名为 `alphadelta1`、`alphadelta2`、`alphadelta3` 等的通道。
- 指定**同级名称**。例如，`CN=John Smith, O=IBM ,OU=Test , C=GB`。有关对等名称的更多信息，请参阅 [SSLPEER 值的 WebSphere MQ 规则](#)。
- 可选：指定所使用的**地址过滤器**。此地址是通道另一端期望的 IP 地址。
- 可选：指定 **SSL 证书颁发者名称**。SSL 证书颁发者名称是颁发 SSL/TLS 证书的认证中心的名称。
- 可选：单击**下一步**。
- 可选：对于**允许**规则类型，可以选择指定通道认证记录的**用户源**。当入站连接与 SSL/TLS 专有名称匹配时，用户源指定所使用的用户标识的源。
 - **通道**选项指定与映射匹配的入站连接将使用所传输的用户标识或通道对象上定义的任何用户。
 - **映射**选项指定与映射匹配的入站连接将使用 **MCA 用户标识** 字段中指定的用户标识。
- 可选：单击**下一步**。
- 可选：指定通道认证记录的**描述**。
- 单击**创建**。此时会创建新的通道认证记录。

V 9.0.1 创建具有客户机应用程序用户标识身份的通道认证记录


您可以使用“通道认证记录”窗口小部件来创建具有客户机应用程序用户标识身份的“允许”、“阻止”和“警告”通道认证记录。客户机应用程序用户标识身份将与来自客户机连接通道的客户机应用程序标识进行匹配。

开始之前

必须先创建“通道认证记录”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『[使用 IBM MQ 对象](#)』。

过程

- 要添加通道认证记录:

- 单击 "通道认证记录" 窗口小部件工具栏中的 "创建" 图标 。
- 选择**规则类型**以指示通道认证记录所需的规则类型:
 - 选择**允许**可允许访问入站连接。
 - 选择**阻止**可阻止访问入站连接。
 - 选择**警告**可在访问要阻止访问的入站连接时发出警告。允许访问连接，但会报告一条错误消息。配置事件时，系统会创建一条错误消息来显示要阻止访问的内容的详细信息。仅报告匹配的规则。
- 从列表中选择**客户机应用程序用户标识** 身份类型。

- d) 单击**下一步**。
- e) 指定**通道概要文件**。
通道概要文件名称是要为其设置通道认证的通道或通道集的名称。概要文件可包含通配符，这样便可以阻止一系列通道。例如，概要文件 `alphadelta*` 将阻止名为 `alphadelta1`、`alphadelta2`、`alphadelta3` 等的通道。
- f) 指定**客户机用户标识**。客户机用户标识是您想要允许、阻止或发出警告的客户机的用户标识。
- g) 可选：指定所使用的**地址过滤器**。此地址是通道另一端期望的 IP 地址。
- h) 可选：单击**下一步**。
- i) 可选：对于**允许规则类型**，可以选择指定通道认证记录的**用户源**。当入站连接与客户机用户标识匹配时，用户源指定所使用的用户标识的源。
 - **通道**选项指定与映射匹配的入站连接将使用所传输的用户标识或通道对象上定义的任何用户。
 - **映射**选项指定与映射匹配的入站连接将使用 **MCA 用户标识**字段中指定的用户标识。
- j) 可选：单击**下一步**。
- k) 可选：指定通道认证记录的**描述**。
- l) 单击**创建**。此时会创建新的通道认证记录。

V 9.0.1 创建具有远程队列管理器名称身份的通道认证记录

您可以使用“通道认证记录”窗口小部件来创建具有远程队列管理器名称身份的“允许”、“阻止”和“警告”通道认证记录。远程队列管理器名称身份将与指定的队列管理器进行匹配。

开始之前

必须先创建“通道认证记录”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。

过程

- 要添加通道认证记录：

- a) 单击“通道认证记录”窗口小部件工具栏中的“创建”图标 。
- b) 选择**规则类型**以指示通道认证记录所需的规则类型：
 - 选择**允许**可允许访问入站连接。
 - 选择**阻止**可阻止访问入站连接。
 - 选择**警告**可在访问要阻止访问的入站连接时发出警告。允许访问连接，但会报告一条错误消息。配置事件时，系统会创建一条错误消息来显示要阻止访问的内容的详细信息。仅报告匹配的规则。
- c) 从列表中选择**远程队列管理器名称身份**类型。
- d) 单击**下一步**。
- e) 指定**概要文件名称**。
概要文件名称是要为其设置通道认证的通道或通道集的名称。概要文件可包含通配符，这样便可以阻止一系列通道。例如，概要文件 `alphadelta*` 将阻止名为 `alphadelta1`、`alphadelta2`、`alphadelta3` 等的通道。
- f) 指定**队列管理器名称**。队列管理器名称指定您想要允许、阻止或发出警告的远程队列管理器的名称。
- g) 可选：指定所使用的**地址过滤器**。此地址是通道另一端期望的 IP 地址。
- h) 可选：单击**下一步**。
- i) 可选：对于**允许规则类型**，可以选择指定通道认证记录的**用户源**。当入站连接与远程队列管理器名称匹配时，用户源指定所使用的用户标识的源。
 - **通道**选项指定与映射匹配的入站连接将使用所传输的用户标识或通道对象上定义的任何用户。
 - **映射**选项指定与映射匹配的入站连接将使用 **MCA 用户标识**字段中指定的用户标识。

- j) 可选：单击**下一步**。
- k) 可选：指定通道认证记录的**描述**。
- l) 单击**创建**。此时会创建新的通道认证记录。

V 9.0.1 创建具有 IP 地址身份的通道认证记录

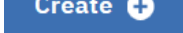
您可以使用“通道认证记录”窗口小部件来创建具有地址身份的“允许”、“阻止”和“警告”通道认证记录。地址身份将与特定的 IP 地址进行匹配。

开始之前

必须先创建“通道认证记录”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。

过程

- 要添加通道认证记录：

- a) 单击“通道认证记录”窗口小部件工具栏中的“创建”图标 。
- b) 选择**规则类型**以指示通道认证记录所需的规则类型：
 - 选择**允许**可允许访问入站连接。
 - 选择**阻止**可阻止访问入站连接。
 - 选择**警告**可在访问要阻止访问的入站连接时发出警告。允许访问连接，但会报告一条错误消息。配置事件时，系统会创建一条错误消息来显示要阻止访问的内容的详细信息。仅报告匹配的规则。
- c) 从列表中选择**地址**身份类型。
- d) 单击**下一步**。
- e) 可选：对于**阻止**或**警告**规则类型，请指定**何时匹配**。
可以从以下选项中进行选择：
 - **在侦听器上**。此选项尝试在侦听器上匹配规则。
 - **在通道上**。此选项尝试在通道上匹配规则。
- f) 指定**概要文件名称**。
概要文件名称是要为其设置通道认证的通道或通道集的名称。概要文件可包含通配符，这样便可以阻止一系列通道。例如，概要文件 `alphadelta*` 将阻止名为 `alphadelta1`、`alphadelta2`、`alphadelta3` 等的通道。
- g) 指定**地址**。地址是允许或阻止的 IP 地址或 IP 地址列表（用逗号分隔）。
- h) 可选：单击**下一步**。
- i) 可选：对于**允许**规则类型，可以选择指定通道认证记录的**用户源**。当入站连接与远程队列管理器名称匹配时，用户源指定所使用的用户标识的源。
 - **通道**选项指定与映射匹配的入站连接将使用所传输的用户标识或通道对象上定义的任何用户。
 - **映射**选项指定与映射匹配的入站连接将使用 **MCA 用户标识**字段中指定的用户标识。
- j) 可选：单击**下一步**。
- k) 可选：指定通道认证记录的**描述**。
- l) 单击**创建**。此时会创建新的通道认证记录。

V 9.0.1 创建具有最终分配的用户标识身份的通道认证记录


您可以使用“通道认证记录”窗口小部件来创建具有最终分配的用户标识身份的“阻止”和“警告”通道认证记录。最终分配的用户标识身份将与来自服务器通道的指定用户标识列表进行匹配。

开始之前

必须先创建“通道认证记录”窗口小部件，然后才能使用。有关创建“IBM MQ 对象”窗口小部件的更多信息，请参阅第 87 页的『使用 IBM MQ 对象』。

过程

- 要添加通道认证记录：


- a) 单击“通道认证记录”窗口小部件工具栏中的“创建”图标 。
- b) 选择**规则类型**以指示通道认证记录所需的规则类型：
 - 选择**阻止**可阻止访问进站连接。
 - 选择**警告**可在访问要阻止访问的进站连接时发出警告。允许访问连接，但会报告一条错误消息。配置事件时，系统会创建一条错误消息来显示要阻止访问的内容的详细信息。仅报告匹配的规则。
- c) 从列表中选择**最终分配的用户标识**身份类型。
- d) 单击**下一步**。
- e) 指定**概要文件名称**。
概要文件名称是要为其设置通道认证的通道或通道集的名称。概要文件可包含通配符，这样便可以阻止一系列通道。例如，概要文件 `alphadelta*` 将阻止名为 `alphadelta1`、`alphadelta2`、`alphadelta3` 等的通道。
- f) 指定**用户列表**。用户列表是通道阻止的用户标识列表（用逗号分隔）。
- g) 可选：单击**下一步**。
- h) 可选：指定通道认证记录的**描述**。
- i) 单击**创建**。此时会创建新的通道认证记录。

V 9.0.1 使用权限记录


您可以通过指定组的权限记录来控制组对于队列管理器和 IBM MQ 对象的访问权限。

关于此任务

您可以通过使用权限记录来调整消息传递用户组对于特定队列管理器或 IBM MQ 对象的访问权限。虽然可配置的实际许可权取决于对象类型，但可以使用相同过程，通过相同方式为所有对象类型配置权限记录。

例如，对比可用于队列管理器和队列的不同许可权，如下图所示：

Authority records for 'qm3'

| Delete  | | 1 item selected | Cancel |
|------------------------------------------------------------------------------------------|-------------|--------------------------|--------|
| ▲ Entity name | Entity type | | |
| mqm | Group | | |
| mqsystem | User | | |
| Total: 2 | | Last updated: 3:40:18 PM | |

Administration

- Change
- Delete
- Display
- Ctrl

Context

- Set all context
- Set identity context

MQI

- Alternate user authority
- Connect
- Inquire
- Set
- System

Check all


Uncheck all

Close

Save


V 9.0.5

Authority records for 'q1' on qm3

| Delete  | | 1 item selected | Cancel |
|------------------------------------------------------------------------------------------|-------------|-----------------|--------|
| ▲ Entity name | Entity type | | |
| mqm | Group | | |
| mqsystem | User | | |


| Administration | Context | MQI |
|---------------------------------------------|-----------------------------------------------------------|---------------------------------------------|
| <input checked="" type="checkbox"/> Change | <input checked="" type="checkbox"/> Pass all context | <input checked="" type="checkbox"/> Browse |
| <input checked="" type="checkbox"/> Clear | <input checked="" type="checkbox"/> Pass identity context | <input checked="" type="checkbox"/> Inquire |
| <input checked="" type="checkbox"/> Delete | <input checked="" type="checkbox"/> Set all context | <input checked="" type="checkbox"/> Get |
| <input checked="" type="checkbox"/> Display | <input checked="" type="checkbox"/> Set identity context | <input checked="" type="checkbox"/> Put |
| | | <input checked="" type="checkbox"/> Set |

Check all Uncheck all Close Save

 无法在 z/OS 上使用权限记录。

过程

- 要查看或编辑 IBM MQ 对象的权限记录：
 - a) 在仪表板上的窗口小部件中选择对象。相关队列管理器必须正在运行。
 - b) 从相应的窗口小部件工具栏中，选择 ... > **管理权限记录**。
 - c) 选择要查看其权限记录的组。此时会显示该组的权限。
 - d) 根据需要选中或清除权限。根据要为其创建权限记录的对象类型，可以使用不同的权限。
 - e) 单击**保存**。
- 要查看或编辑队列管理器的权限记录：
 - a) 在仪表板上的“队列管理器”窗口小部件中选择队列管理器。该队列管理器必须正在运行。
 - b) 从窗口小部件工具栏中，选择 ... > **管理创建权限记录**。
 - c) 选择要查看其创建权限记录的组。此时会显示该组的权限。
 - d) 根据需要选中或清除创建权限。
 - e) 单击**保存**。
- 要为 IBM MQ 对象创建权限记录：
 - a) 选择仪表板上的窗口小部件中的 IBM MQ 对象。相关队列管理器必须正在运行。
 - b) 从窗口小部件工具栏中，选择 ... > **管理权限记录**。

- c) 单击加号图标 。
 - d) 指定要为其创建权限记录的用户或组的名称。该用户或组必须存在。
 - e) 选择**实体类型**以指定实体是用户还是组。
 - f) 单击**创建**。
 - g) 选中或清除您希望该用户或组具有的权限。每种对象类型有不同的可用权限。
 - h) 单击**保存**。
- 要创建权限记录以便在队列管理器上创建对象：
 - a) 选择仪表板上的窗口小部件中的队列管理器。该队列管理器必须正在运行。
 - b) 从窗口小部件工具栏中，选择 ... > **管理创建权限记录**。
- c) 单击 "创建" 图标 。
 - d) 指定要为其创建权限记录的用户或组的名称。该用户或组必须存在。
 - e) 选择**实体类型**以指定实体是用户还是组。
 - f) 单击**创建**。
 - g) 选中或清除您希望该用户或组具有的创建权限。
 - h) 单击**保存**。

V 9.0.1 监视系统资源使用情况

您可以在 IBM MQ Console 中使用“图表”窗口小部件来查看队列管理器的监视数据。

关于此任务

可以将“图表”窗口小部件添加到仪表板中，然后将其配置为监视资源使用的特定方面。您可以创建“图表”窗口小部件的多个实例以用于显示不同的数据。将采用图表格式显示数据。

每 10 秒收集一次数据。图表的 X 轴显示时间线。Y 轴显示与所查看资源相对应的单位。Y 轴将根据返回的数据来动态调整大小。

您必须至少有一个正在运行的队列管理器，才能配置“图表”窗口小部件。


 无法在 z/OS 上监视系统资源使用情况。

过程

1. 将“图表”窗口小部件添加到仪表板中：

- a) 单击 "添加窗口小部件" 图标 。
- b) 选择**图表**。

2. 配置“图表”窗口小部件以显示数据：

- a) 单击“图表”窗口小部件标题栏中的配置图标 。
- b) 可选：输入**窗口小部件标题**。此标题会显示在该窗口小部件的标题栏中。
- c) 选择要监视的**资源类**：
 - 平台中央处理器**
监视 CPU 使用情况。
 - 平台持久数据存储**
监视磁盘资源使用情况。
 - API 使用情况统计信息**
监视 API 调用。

API 按队列使用情况统计信息

监视各个队列的 API 调用。选择此类后，请在**对象**字段中指定要监视的队列名称。

d) 选择要监视的**资源类型**。

可供选择的资源类型取决于所选的资源类。下表显示了资源类型：

| 表 6: 资源类型 | | |
|-----------------|--------------------|----------------------------------------------------------------------------------------------------|
| 类 | 类型 | 描述 |
| 平台中央处理器 | CPU 性能 - 平台级 | 选择此类型可查看 CPU 和内存的性能数据。 |
| | CPU 性能 - 运行中的队列管理器 | 选择此类型可查看与要监视的队列管理器相关的 CPU 和内存的性能数据。队列管理器必须正在运行，才能对其进行监视。如果您正在监视来自多个队列管理器的结果，那么会使用不同的颜色来区分图表中的性能数据。 |
| 平台持久数据存储 | 磁盘使用情况 - 平台级 | 选择此类型可查看全局磁盘使用情况的性能数据。 |
| | 磁盘使用情况 - 运行中的队列管理器 | 选择此类型可查看与要监视的队列管理器相关的磁盘使用情况的性能数据。队列管理器必须正在运行，才能对其进行监视。如果您正在监视来自多个队列管理器的结果，那么会使用不同的颜色来区分图表中的性能数据。 |
| | 磁盘使用情况 - 队列管理器恢复日志 | 选择此类型可查看有关如何对要监视的每个队列管理器恢复日志使用磁盘存储器的数据。 |
| API 使用情况统计信息 | MQCONN 和 MQDISC | 选择此类型可查看有关 MQCONN 和 MQDISC 调用的数据。 |
| | MQOPEN 和 MQCLOSE | 选择此类型可查看有关 MQOPEN 和 MQCLOSE 调用的数据。 |
| | MQINQ 和 MQSET | 选择此类型可查看有关 MQINQ 和 MQSET 调用的数据。 |
| | MQPUT | 选择此类型可查看有关 MQPUT 相关调用的数据。 |
| | MQGET | 选择此类型可查看有关 MQGET 相关调用的数据。 |
| | 落实和回滚 | 选择此类型可查看有关队列管理器如何使用同步点的信息。 |
| | 预订 | 选择此类型可查看有关 MQSUB 调用的数据。 |
| | 发布 | 选择此类型可查看有关已发布消息的数据。 |
| API 按队列使用情况统计信息 | MQOPEN 和 MQCLOSE | 选择此类型可查看有关指定队列的 MQOPEN 和 MQCLOSE 调用的数据。 |

| 表 6: 资源类型 (继续) | | |
|----------------|----------------|-----------------------------------------|
| 类 | 类型 | 描述 |
| | MQINQ 和 MQSET | 选择此类型可查看有关指定队列的 MQINQ 和 MQSET 调用的数据。 |
| | MQPUT 和 MQPUT1 | 选择此类型可查看有关指定队列的 MQPUT 和 MQPUT1 相关调用的数据。 |
| | MQGET | 选择此类型可查看有关指定队列的 MQGET 相关调用的数据。 |

e) 选择要监视的资源元素:

可供选择的资源元素取决于所选的资源类和资源类型。下表显示了资源元素:

| 表 7: 平台中央处理器资源的元素 | | |
|--------------------|--------------------------|-------------------------------------|
| 类型 | 元素 | 描述 |
| CPU 性能 - 平台级 | 用户 CPU 时间百分比 | 显示用户状态的 CPU 繁忙百分比。 |
| | 系统 CPU 时间百分比 | 显示系统状态的 CPU 繁忙百分比。 |
| | CPU 负载 - 1 分钟平均值 | 显示 1 分钟内的平均负载量。 |
| | CPU 负载 - 5 分钟平均值 | 显示 5 分钟内的平均负载量。 |
| | CPU 负载 - 15 分钟平均值 | 显示 15 分钟内的平均负载量。 |
| | 可用 RAM 百分比 | 显示可用 RAM 内存的百分比。 |
| | RAM 总字节数 | 显示已配置的 RAM 的总字节数。 |
| CPU 性能 - 运行中的队列管理器 | 用户 CPU 时间 - 队列管理器的百分比估算值 | 估算在与所监视的队列管理器相关的进程中用户状态的 CPU 使用百分比。 |
| | 系统 CPU 时间 - 队列管理器的百分比估算值 | 估算在与所监视的队列管理器相关的进程中系统状态的 CPU 使用百分比。 |
| | RAM 总字节数 - 队列管理器的估算值 | 估算所监视的队列管理器正在使用的 RAM 总字节数。 |

| 表 8: 平台持久数据存储资源元素的元素 | | |
|----------------------|--------------------|-----------------------|
| 类型 | 元素 | 描述 |
| 磁盘使用情况 - 平台级 | MQ 跟踪文件系统 - 使用的字节数 | 显示跟踪文件系统正在使用的磁盘存储字节数。 |
| | MQ 跟踪文件系统 - 可用空间 | 显示专为跟踪文件系统保留的可用磁盘存储量。 |
| | MQ 错误文件系统 - 使用的字节数 | 显示错误数据正在使用的磁盘存储字节数。 |
| | MQ 错误文件系统 - 可用空间 | 显示专为错误数据保留的可用磁盘存储量。 |

| 表 8: 平台持久数据存储资源元素 (继续) | | |
|------------------------|--------------------|----------------------------------|
| 类型 | 元素 | 描述 |
| | MQ FDC 文件数 | 显示 FDC 文件的当前数量。 |
| 磁盘使用情况 - 运行中的队列管理器 | 队列管理器文件系统 - 使用的字节数 | 显示所监视的队列管理器的队列管理器文件正在使用的磁盘存储字节数。 |
| | 队列管理器文件系统 - 可用空间 | 显示专为队列管理器文件保留的可用磁盘存储量。 |
| 磁盘使用情况 - 队列管理器恢复日志 | 日志 - 使用的字节数 | 显示所监视的队列管理器的恢复日志正在使用的磁盘存储字节数。 |
| | 日志 - 最大字节数 | 显示配置为用于队列管理器恢复日志的最大磁盘存储字节数。 |
| | 日志文件系统 - 使用的字节数 | 显示日志文件系统正在使用的磁盘总字节数。 |
| | 日志文件系统 - 最大字节数 | 显示针对日志文件系统配置的磁盘字节数。 |
| | 日志 - 写入的物理字节数 | 显示已写入恢复日志的字节数。 |
| | 日志 - 写入的逻辑字节数 | 显示已写入恢复日志的逻辑字节数。 |
| | 日志 - 写等待时间 | 显示同步写入队列管理器恢复日志时的等待时间度量方式。 |

| 表 9: API 使用情况统计信息资源元素 | | |
|-----------------------|----------------------|-----------------------------|
| 类型 | 元素 | 描述 |
| MQCONN 和 MQDISC | MQCONN/MQCONN 计数 | 显示 MQCONN 和 MQCONN 调用次数。 |
| | 失败的 MQCONN/MQCONN 计数 | 显示失败的 MQCONN 和 MQCONN 调用次数。 |
| | 并发连接 - 高水位标记 | 显示当前统计时间间隔内并发连接的最大数目。 |
| | MQDISC 计数 | 显示 MQDISC 调用次数。 |
| MQOPEN 和 MQCLOSE | MQOPEN 计数 | 显示 MQOPEN 调用次数。 |
| | 失败的 MQOPEN 计数 | 显示失败的 MQOPEN 调用次数。 |
| | MQCLOSE 计数 | 显示 MQCLOSE 调用次数。 |
| | 失败的 MQCLOSE 计数 | 显示失败的 MQCLOSE 调用次数。 |
| MQINQ 和 MQSET | MQINQ 计数 | 显示 MQINQ 调用次数。 |

| 表 9: API 使用情况统计信息资源的元素 (继续) | | |
|-----------------------------|-------------------------|---------------------------------|
| 类型 | 元素 | 描述 |
| | 失败的 MQINQ 计数 | 显示失败的 MQINQ 调用次数。 |
| | MQSET 计数 | 显示 MQSET 调用次数。 |
| | 失败的 MQSET 计数 | 显示失败的 MQSET 调用次数。 |
| MQPUT | 时间间隔总 MQPUT/MQPUT1 计数 | 显示 MQPUT 和 MQPUT1 调用次数。 |
| | 时间间隔总 MQPUT/MQPUT1 字节计数 | 显示由 MQPUT 和 MQPUT1 调用放入的数据总字节数。 |
| | 非持久消息 MQPUT 计数 | 显示 MQPUT 放入的非持久消息数。 |
| | 持久消息 MQPUT 计数 | 显示 MQPUT 放入的持久消息数。 |
| | 失败的 MQPUT 计数 | 显示失败的 MQPUT 调用次数。 |
| | 非持久消息 MQPUT1 计数 | 显示 MQPUT1 放入的非持久消息数。 |
| | 持久消息 MQPUT1 计数 | 显示 MQPUT1 放入的持久消息数。 |
| | 失败的 MQPUT1 计数 | 显示失败的 MQPUT1 调用次数。 |
| | 放入非持久消息 - 字节计数 | 显示放入非持久消息的字节数。 |
| | 放入持久消息 - 字节计数 | 显示放入持久消息的字节数。 |
| | MQSTAT 计数 | 显示 MQSTAT 调用次数。 |
| | 失败的 MQSTAT 计数 | 显示失败的 MQSTAT 调用次数。 |
| MQGET | 时间间隔总破坏性获取 - 计数 | 由 MQGET 从队列中移除的消息数。 |
| | 时间间隔总破坏性获取 - 字节计数 | 由 MQGET 从队列中移除的数据的字节数。 |
| | 非持久消息破坏性获取 - 计数 | 由 MQGET 从队列中移除的非持久消息数。 |
| | 持久消息破坏性获取 - 计数 | 由 MQGET 从队列中移除的持久消息数。 |
| | 失败的 MQGET - 计数 | 显示失败的 MQGET 调用次数。 |

| 表 9: API 使用情况统计信息资源的元素 (继续) | | |
|-----------------------------|----------------|--------------------------|
| 类型 | 元素 | 描述 |
| | 获取非持久消息 - 字节计数 | 显示返回给 MQGET 的非持久消息字节计数。 |
| | 获取持久消息 - 字节计数 | 显示返回给 MQGET 的持久消息字节计数。 |
| | 非持久消息浏览 - 计数 | 显示已浏览的非持久消息计数。 |
| | 持久消息浏览 - 计数 | 显示已浏览的持久消息计数。 |
| | 失败的浏览计数 | 显示失败的消息浏览计数。 |
| | 非持久消息浏览 - 字节计数 | 显示已浏览的非持久消息字节计数。 |
| | 持久消息浏览 - 字节计数 | 显示已浏览的持久消息字节计数。 |
| | 到期消息计数 | 显示到期的消息计数。 |
| | 队列清除计数 | 显示已清除的队列计数。 |
| | MQCB 计数 | 显示 MQCB 调用次数。 |
| | 失败的 MQCB 计数 | 显示失败的 MQCB 调用次数。 |
| | MQCTL 计数 | 显示 MQCTL 调用次数。 |
| | 失败的 MQCTL 计数 | 显示失败的 MQCTL 调用次数。 |
| 落实和回滚 | 落实计数 | 显示 MQCMIT 调用次数。 |
| | 失败的落实计数 | 显示失败的 MQCMIT 调用次数。 |
| | 回滚计数 | 显示 MQBACK 调用次数。 |
| 预订 | 创建持久预订计数 | 显示用于创建持久预订的 MQSUB 调用次数。 |
| | 更改持久预订计数 | 显示用于更改持久预订的 MQSUB 调用次数。 |
| | 恢复持久预订计数 | 显示用于恢复持久预订的 MQSUB 调用次数。 |
| | 创建非持久预订计数 | 显示用于创建非持久预订的 MQSUB 调用次数。 |
| | 更改非持久预订计数 | 显示用于更改非持久预订的 MQSUB 调用次数。 |
| | 恢复非持久预订计数 | 显示用于恢复非持久预订的 MQSUB 调用次数。 |

| 表 9: API 使用情况统计信息资源的元素 (继续) | | |
|-----------------------------|--------------------------|----------------------------------|
| 类型 | 元素 | 描述 |
| | 失败的创建/更改/恢复预订计数 | 显示用于创建、变更或恢复预订且失败的 MQSUBRQ 调用次数。 |
| | 删除持久预订计数 | 显示用于删除持久预订的 MQSUB 调用次数。 |
| | 删除非持久预订计数 | 显示用于删除非持久预订的 MQSUB 调用次数。 |
| | 预订删除失败计数 | 显示用于删除预订的 MQSUB 调用次数。 |
| | MQSUBRQ 计数 | 显示 MQSUBRQ 调用次数。 |
| | 失败的 MQSUBRQ 计数 | 显示失败的 MQSUBRQ 调用次数。 |
| | 持久订户 - 高水位标记 | 显示当前统计时间间隔内持久预订的最大数目。 |
| | 持久订户 - 低水位标记 | 显示当前统计时间间隔内持久预订的最小数目。 |
| | 非持久订户 - 高水位标记 | 显示当前统计时间间隔内非持久预订的最大数目。 |
| | 非持久订户 - 低水位标记 | 显示当前统计时间间隔内非持久预订的最小数目。 |
| 发布 | 主题 MQPUT/MQPUT1 总时间间隔 | 放入主题的消息数。 |
| | 时间间隔总放入主题字节数 | 放入主题的消息字节数。 |
| | 发布到订户 - 消息计数 | 显示已发布到订户的消息数。 |
| | 发布到订户 - 字节计数 | 显示已发布到订户的消息字节计数。 |
| | 非持久 - 主题 MQPUT/MQPUT1 计数 | 显示放入主题的非持久消息数。 |
| | 持久 - 主题 MQPUT/MQPUT1 计数 | 显示放入主题的持久消息数。 |
| | 失败的主题 MQPUT/MQPUT1 计数 | 显示放入主题的失败尝试次数。 |

| 表 10: 按队列划分的 API 使用情况统计信息资源的元素 | | |
|--------------------------------|------------|------------------|
| 类型 | 元素 | 描述 |
| MQOPEN 和 MQCLOSE | MQOPEN 计数 | 显示 MQOPEN 调用次数。 |
| | MQCLOSE 计数 | 显示 MQCLOSE 调用次数。 |

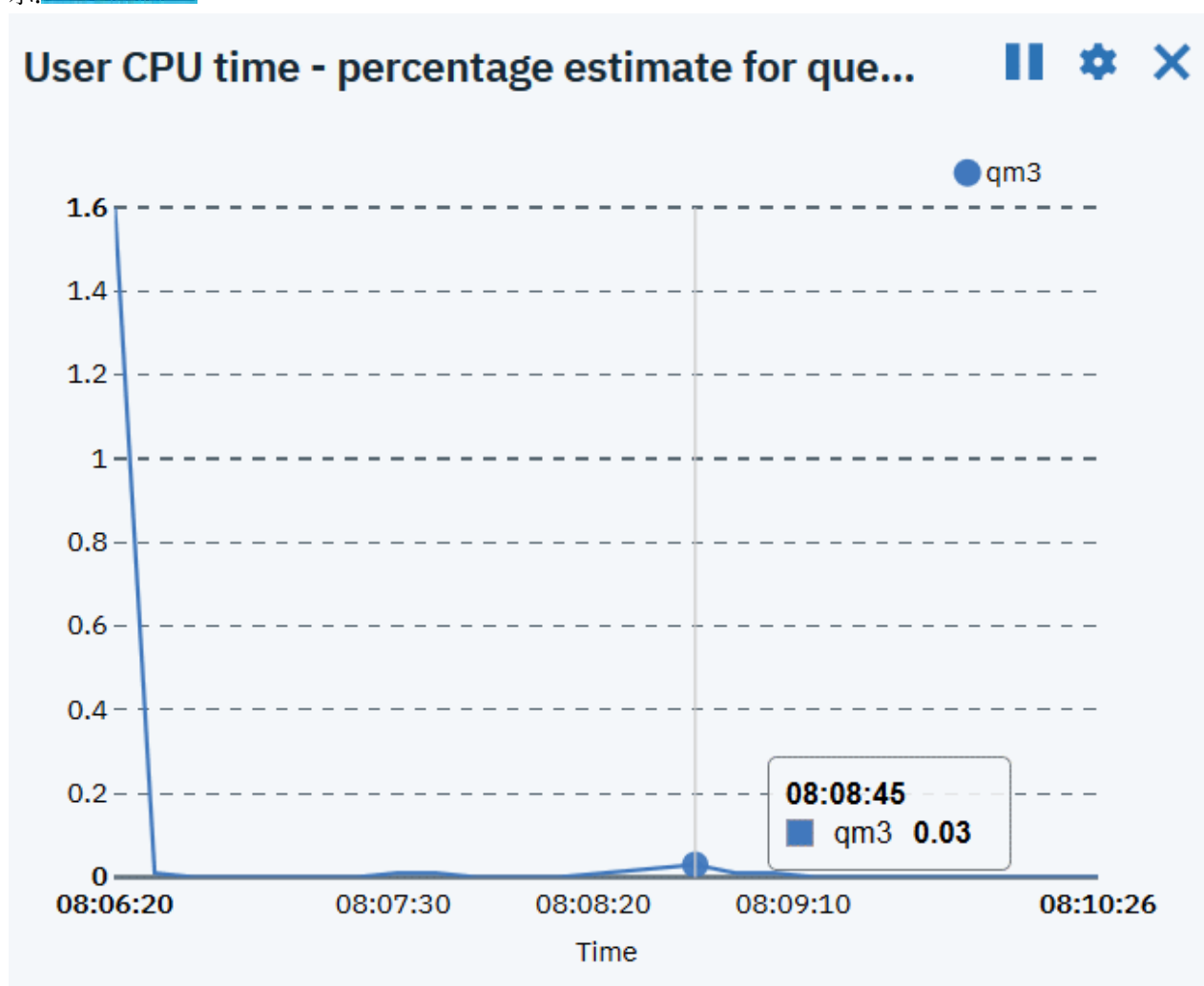
| 表 10: 按队列划分的 API 使用情况统计信息资源的元素 (继续) | | |
|-------------------------------------|-------------------|---------------------------------|
| 类型 | 元素 | 描述 |
| MQINQ 和 MQSET | MQINQ 计数 | 显示 MQINQ 调用次数。 |
| | MQSET 计数 | 显示 MQSET 调用次数。 |
| MQPUT 和 MQPUT1 | MQPUT/MQPUT1 计数 | 显示 MQPUT 和 MQPUT1 调用次数。 |
| | MQPUT 字节计数 | 显示由 MQPUT 和 MQPUT1 调用放入的数据总字节数。 |
| | MQPUT 非持久消息计数 | 显示 MQPUT 放入的非持久消息数。 |
| | MQPUT 持久消息计数 | 显示 MQPUT 放入的持久消息数。 |
| | MQPUT1 非持久消息计数 | 显示 MQPUT1 放入的非持久消息数。 |
| | MQPUT1 持久消息计数 | 显示 MQPUT1 放入的持久消息数。 |
| | 非持久字节计数 | 显示放入非持久消息的字节数。 |
| | 持久字节计数 | 显示放入持久消息的字节数。 |
| | 避开排队的放入次数 | |
| | 避开排队的字节数 | |
| | 锁定争用 | |
| MQGET | MQGET 计数 | |
| | MQGET 字节计数 | |
| | 破坏性 MQGET 非持久消息计数 | 由 MQGET 从队列中移除的非持久消息数。 |
| | 破坏性 MQGET 持久消息计数 | 由 MQGET 从队列中移除的持久消息数。 |
| | 破坏性 MQGET 非持久字节计数 | 显示返回给 MQGET 的非持久消息字节计数。 |
| | 破坏性 MQGET 持久字节计数 | 显示返回给 MQGET 的持久消息字节计数。 |
| | MQGET 浏览非持久消息计数 | 显示已浏览的非持久消息计数。 |
| | MQGET 浏览持久消息计数 | 显示已浏览的持久消息计数。 |
| | MQGET 浏览非持久字节计数 | 显示已浏览的非持久消息字节计数。 |
| | MQGET 浏览持久字节计数 | 显示已浏览的持久消息字节计数。 |
| | 到期消息数 | 显示到期的消息计数。 |

| 表 10: 按队列划分的 API 使用情况统计信息资源的元素 (继续) | | |
|-------------------------------------|---------|-------------|
| 类型 | 元素 | 描述 |
| | 清除的队列计数 | 显示已清除的队列计数。 |
| | 平均排队时间 | |
| | 队列时间 | |

- f) 选择要监视的队列管理器，并指定用于显示该队列管理器的信息的颜色。单击**添加**可添加更多队列管理器。可最多指定 5 个队列管理器。
- g) 单击**保存**。

结果


在配置该窗口小部件后但在图表中显示数据前，存在短暂的延迟。沿时间轴显示数据。每个数据点均表示 10 秒数据收集周期的结束点。您可以将鼠标悬停在图表中的数据点上，以查看详细信息，如以下示例中所示：**V 9.0.5**



V 9.0.1 配置仪表板布局

仪表板是 IBM MQ Console 中用于显示窗口小部件的容器。您可以创建多个仪表板选项卡以显示不同的信息选择。

关于此任务

您可以通过单击选项卡名称  旁边的箭头来配置每个仪表板选项卡。您可以更改选项卡名称以及为选项卡添加描述。您还可以配置选项卡具有的列数。

您可以通过拖放窗口小部件，在仪表板选项卡中配置窗口小部件布局。

过程

- [第 112 页的『创建和删除仪表板选项卡』](#)
- [第 113 页的『导入和导出仪表板布局』](#)

V 9.0.1 创建和删除仪表板选项卡

可以自动创建仪表板选项卡，以用于显示特定的本地队列管理器的相关信息。您可以手动创建和删除仪表板选项卡。

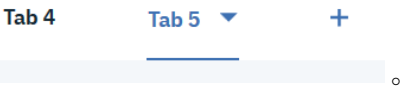

关于此任务

在自动创建仪表板选项卡以用于显示特定的本地队列管理器的相关信息时，会自动添加以下窗口小部件：

- “队列”窗口小部件
- “客户机连接通道”窗口小部件
- “通道”窗口小部件
- “侦听器”窗口小部件
- “预订”窗口小部件
- “主题”窗口小部件
- “认证信息”窗口小部件

过程


- 要创建仪表板选项卡：

- 
- 单击现有仪表板选项卡旁边的加号图标 。
 - 输入新选项卡的名称。
 - 可选：输入新选项卡的描述。
 - 单击**添加**。

- 要为特定队列管理器自动创建仪表板选项卡：

- 在“本地队列管理器”窗口小部件中选择队列管理器。
- 选择 **... > 添加新的仪表板选项卡**
此时会创建新仪表板选项卡。此选项卡包含队列管理器的名称。

- 要删除仪表板选项卡：

- 单击仪表板选项卡名称  旁边的箭头。
- 选择**删除选项卡**。
- 单击**删除**以确认您要删除该仪表板选项卡。此时会删除该选项卡。

V 9.0.1 导入和导出仪表板布局



您可以从 IBM MQ Console 中导出仪表板布局以进行保存。您可以将已保存的仪表板布局导入到 IBM MQ Console。

关于此任务



在导出仪表板时，会在本地磁盘上创建一个 .json 文件。随后，您可以将该 .json 文件导入到仪表板以重新创建布局。在导入仪表板布局时，您可以选择将已导入的选项卡添加到现有仪表板布局中。也可以将现有仪表板布局替换为导入的布局。

过程

- 要导出仪表板布局：

- a) 单击仪表板菜单图标  。
- b) 选择**导出仪表板**。
这样会将该文件保存到浏览器下载文件夹中。

- 要导入仪表板布局：

- a) 单击仪表板菜单图标  。
- b) 选择**导入仪表板**。
此时会打开“导入仪表板配置”窗口。
- c) 单击**浏览**并浏览至包含相关配置的文件所在的位置。
- d) 选择仪表板选项卡的导入方式：
可以从以下选项中进行选择：
 - 将导入的仪表板选项卡附加到现有仪表板上
 - 将现有仪表板替换为导入的仪表板选项卡
- e) 单击**导入**。
此时会导入仪表板选项卡。



V 9.0.1 仪表板控件

使用仪表板顶部的控件启用 IBM MQ Console 跟踪、访问联机帮助、查看 IBM MQ Console 相关信息，以及从 IBM MQ Console 注销。


关于此任务

仪表板控件适用于 IBM MQ Console 的常规功能。

过程

- 通过单击帮助图标  来访问 IBM Documentation for IBM MQ。
- 单击菜单图标  以导入、导出或重置仪表板。有关更多信息，请参阅第 111 页的『配置仪表板布局』。
- 单击设置图标  以对 IBM MQ Console 启用或禁用诊断跟踪。



- 通过单击人员图标  并选择 **关于**，可查看有关 IBM MQ Console 的信息，包括有关已登录用户的信息。
- 通过单击 **注销** 从 IBM MQ Console 注销。
如果通过客户机证书登录到 IBM MQ Console，或者如果使用了安全配置样本 no_security.xml，那么此选项不可用。有关安全配置的更多信息，请参阅[配置 IBM MQ Console 安全性](#)。

键盘快捷键

您可以通过键盘快捷键来使用 IBM MQ Console。

下表列出了可用快捷键。可在窗口小部件中使用所有快捷键。

| 键 | 操作 |
|---------|---------------|
| p | 查看所选对象的属性 |
| c | 创建新对象 |
| shift-d | 删除所选对象 |
| a | 选择窗口小部件中的所有对象 |
| shift-a | 取消选择对象 |

Windows

Linux

使用 IBM MQ Explorer 进行管理

IBM MQ Explorer 允许您从运行 Windows 或仅 Linux x86-64 的计算机对网络执行本地或远程管理。

IBM MQ for Windows 和 IBM MQ for Linux x86-64 提供了一个称为 IBM MQ Explorer 的管理接口，以执行管理任务作为使用控制或 MQSC 命令的替代方法。[比较命令集](#) 显示可以使用 IBM MQ Explorer 执行的操作。

IBM MQ Explorer 允许您通过将 IBM MQ Explorer 指向您感兴趣的队列管理器和集群，从运行 Windows 或 Linux x86-64 的计算机对网络执行本地或远程管理。它可以远程连接到在任何受支持平台（包括 z/OS）上运行的队列管理器，以便从控制台查看、探索和变更整个消息传递主干。

要配置远程 IBM MQ 队列管理器以便 IBM MQ Explorer 可以对其进行管理，请参阅 [第 116 页的『IBM MQ Explorer 的必备软件和定义』](#)。

它允许您执行任务，通常与在 Windows 或 Linux x86-64 系统域中本地或远程设置和微调 IBM MQ 的工作环境相关联。

在 Linux 上，如果有多个 Eclipse 安装，那么 IBM MQ Explorer 可能无法启动。如果发生这种情况，请使用与用于其他 Eclipse 安装的用户标识不同的用户标识来启动 IBM MQ Explorer。

在 Linux 上，要成功启动 IBM MQ Explorer，您必须能够将文件写入主目录，并且主目录必须存在。

Windows

Linux

您可以使用 IBM MQ Explorer 执行的操作

您可以使用 IBM MQ Explorer 通过一系列 "内容视图" 和 "属性" 对话框来执行管理任务。您还可以通过编写一个或多个 Eclipse 插件来扩展 IBM MQ Explorer。

IBM MQ Explorer 任务数

通过 IBM MQ Explorer，您可以执行以下任务：

- 创建和删除队列管理器 (仅在本地机器上)。
- 启动和停止队列管理器 (仅在本地机器上)。
- 定义，显示和变更 IBM MQ 对象 (例如，队列和通道) 的定义。

- 浏览队列上的消息。
- 启动和停止通道。
- 查看有关通道，侦听器，队列或服务对象的状态信息。
- 查看集群中的队列管理器。
- 检查以查看哪些应用程序，用户或通道打开了特定队列。
- 使用 "创建新集群" 向导创建新的队列管理器集群。
- 使用 "将队列管理器添加到集群" 向导将队列管理器添加到集群。
- 管理认证信息对象，与传输层安全性 (TLS) 通道安全性配合使用。
- 创建和删除通道启动程序，触发器监视器和侦听器。
- 启动或停止命令服务器，通道启动程序，触发器监视器和侦听器。
- 将特定服务设置为在队列管理器启动时自动启动。
- 修改队列管理器的属性。
- 更改本地缺省队列管理器。
- 调用 **strmqim** (ikeyMan) GUI 以管理 TLS 证书，使证书与队列管理器相关联，以及配置和设置证书库 (仅在本地机器上)。
- 从 IBM MQ 对象创建 JMS 对象，从 JMS 对象创建 IBM MQ 对象。
- 为当前支持的任何类型创建 JMS 连接工厂。
- 修改任何服务的参数，例如侦听器的 TCP 端口号或通道启动程序队列名称。
- 启动或停止服务跟踪。

"内容视图" 和 "属性" 对话框

您可以使用一系列 "内容视图" 和 "属性" 对话框来执行管理任务。

内容视图

"内容视图" 是可以显示以下内容的面板:

- 与 IBM MQ 本身相关的属性和管理选项。
- 与一个或多个相关对象相关的属性和管理选项。
- 集群的属性和管理选项。

属性对话框

属性对话框是一个面板，用于显示与一系列字段中的对象相关的属性，您可以对其中一些字段进行编辑。

您可以使用 "Navigator" 视图浏览 IBM MQ Explorer。Navigator 允许您选择所需的内容视图。

扩展 IBM MQ Explorer

IBM MQ Explorer 以与 Eclipse 框架以及 Eclipse 支持的其他插件应用程序的样式一致的方式提供信息。

通过扩展 IBM MQ Explorer，系统管理员能够定制 IBM MQ Explorer 以改进其管理 IBM MQ 的方式。

有关更多信息，请参阅 [扩展 MQ Explorer](#)。

决定是否使用 IBM MQ Explorer

在决定是否在安装时使用 IBM MQ Explorer 时，请考虑本主题中列出的信息。

你需要注意以下几点:

对象名称

如果将队列管理器和其他对象的小写名称与 IBM MQ Explorer 配合使用，那么在使用 MQSC 命令处理对象时，必须将对象名括在单引号中，否则 IBM MQ 无法识别这些对象名。

大型队列管理器

IBM MQ Explorer 最适用于小型队列管理器。如果您在单个队列管理器上具有大量对象，那么当 IBM MQ Explorer 抽取所需信息以显示在视图中时，可能会迂到延迟。

集群

IBM MQ 集群可能包含数百或数千个队列管理器。IBM MQ Explorer 使用树结构提供集群中的队列管理器。集群的物理大小不会显著影响 IBM MQ Explorer 的速度，因为 IBM MQ Explorer 直到您选择它们之后才会连接到集群中的队列管理器。

设置 IBM MQ Explorer

本部分概述了设置 IBM MQ Explorer 所需的步骤。

- [第 116 页的『IBM MQ Explorer 的必备软件和定义』](#)
- [第 116 页的『IBM MQ Explorer 的安全性』](#)
- [第 119 页的『在 IBM MQ Explorer 中显示和隐藏队列管理器和集群』](#)
- [第 120 页的『集群成员资格和 IBM MQ Explorer』](#)
- [第 120 页的『IBM MQ Explorer 的数据转换』](#)

IBM MQ Explorer 的必备软件和定义

在尝试使用 IBM MQ Explorer 之前，请确保满足以下需求。

IBM MQ Explorer 只能使用 TCP/IP 通信协议连接到远程队列管理器。

检查:

1. 命令服务器正在每个远程管理的队列管理器上运行。
2. 必须在每个远程队列管理器上运行合适的 TCP/IP 侦听器对象。此对象可以是 IBM MQ 侦听器，也可以是 UNIX and Linux 系统上的 inetd 守护程序。
3. 缺省情况下名为 SYSTEM.ADMIN.SVRCONN 存在于所有远程队列管理器上。

您可以使用以下 MQSC 命令创建通道:

```
DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)
```

此命令创建基本通道定义。如果需要更复杂的定义(例如，用于设置安全性)，那么需要其他参数。有关更多信息，请参阅 [DEFINE CHANNEL](#)。

4. 系统队列 SYSTEM.MQEXPLORER.REPLY.MODEL 必须存在。

IBM MQ Explorer 的安全性

如果在您控制用户对特定对象的访问权非常重要的环境中使用 IBM MQ，那么可能需要考虑使用 IBM MQ Explorer 的安全性方面。

授权使用 IBM MQ Explorer

任何用户都可以使用 IBM MQ Explorer，但需要某些权限才能连接，访问和管理队列管理器。

要使用 IBM MQ Explorer 执行本地管理任务，需要用户具有执行管理任务所需的权限。如果用户是 mqm 组的成员，那么该用户有权执行所有本地管理任务。

要连接到远程队列管理器并使用 IBM MQ Explorer 执行远程管理任务，执行 IBM MQ Explorer 的用户需要具有以下权限:

- 对目标队列管理器对象的 CONNECT 权限
- 对目标队列管理器对象的 INQUIRE 权限
- 对目标队列管理器对象的 DISPLAY 权限
- 对队列 SYSTEM.MQEXPLORER.REPLY.MODEL
- 对队列 SYSTEM.MQEXPLORER.REPLY.MODEL

- 对队列 SYSTEM.MQEXPLORER.REPLY.MODEL
- 对队列 SYSTEM.MQEXPLORER.REPLY.MODEL
- 对队列 SYSTEM.ADMIN.COMMAND.QUEUE
- 队列 SYSTEM.ADMIN.COMMAND.QUEUE
- 执行所选操作的权限

注: INPUT 权限与来自队列 (获取操作) 的用户输入相关。 OUTPUT 权限与从用户到队列的输出相关 (放置操作)。

要连接到 IBM MQ for z/OS 上的远程队列管理器并使用 IBM MQ Explorer 执行远程管理任务, 必须提供以下内容:

- 系统队列 SYSTEM.MQEXPLORER.REPLY.MODEL 的 RACF 概要文件
- 队列的 RACF 概要文件 AMQ.MQEXPLORER.*

此外, 执行 IBM MQ Explorer 的用户需要具有以下权限:

- RACF 对系统队列 SYSTEM.MQEXPLORER.REPLY.MODEL
- RACF 对队列的 UPDATE 权限, AMQ.MQEXPLORER.*
- 对目标队列管理器对象的 CONNECT 权限
- 执行所选操作的权限
- 对 MQCMDSD 类中所有 hlq.DISPLAY.object 概要文件的 READ 权限

有关如何授予 IBM MQ 对象权限的信息, 请参阅 [授予对 UNIX 或 Linux 系统和 Windows 上的 IBM MQ 对象的访问权](#)。

如果用户尝试执行他们无权执行的操作, 那么目标队列管理器将调用授权失败过程, 并且该操作将失败。

IBM MQ Explorer 中的缺省过滤器是显示所有 IBM MQ 对象。 如果存在用户没有 DISPLAY 权限的任何 IBM MQ 对象, 那么将生成授权失败。 如果正在记录权限事件, 请将显示的对象范围限制为用户具有 DISPLAY 权限的对象。

用于从 *IBM MQ Explorer* 连接到远程队列管理器的安全性

您必须保护 IBM MQ Explorer 与每个远程队列管理器之间的通道。

IBM MQ Explorer 作为 MQI 客户机应用程序连接到远程队列管理器。 这意味着每个远程队列管理器都必须具有服务器连接通道的定义和合适的 TCP/IP 侦听器。 如果您不保护服务器连接通道, 那么恶意应用程序可能会连接到同一服务器连接通道, 并获得对具有无限权限的队列管理器对象的访问权。 为了保护服务器连接通道, 请为通道的 MCAUSER 属性指定非空白值, 使用通道认证记录或使用安全出口。

MCAUSER 属性的缺省值是本地用户标识。 如果指定非空白用户名作为服务器连接通道的 MCAUSER 属性, 那么使用此通道连接到队列管理器的所有程序将以指定用户的身份运行, 并且具有相同的权限级别。 如果使用通道认证记录, 那么不会发生此情况。

将安全出口与 *IBM MQ Explorer* 配合使用

您可以使用 IBM MQ Explorer 指定缺省安全出口和特定于队列管理器的安全出口。

您可以定义缺省安全出口, 该出口可用于来自 IBM MQ Explorer 的所有新客户机连接。 在建立连接时, 可以覆盖此缺省出口。 您还可以为单个队列管理器或一组队列管理器定义安全出口, 这将在建立连接时生效。 使用 IBM MQ Explorer 指定出口。 有关详细信息, 请参阅 [IBM MQ Explorer 帮助](#)。

使用 *IBM MQ Explorer* 通过支持 TLS 的 MQI 通道连接到远程队列管理器

IBM MQ Explorer 使用 MQI 通道连接到远程队列管理器。 如果要使用 TLS 安全性来保护 MQI 通道, 那么必须使用客户机通道定义表来建立通道。

有关如何使用客户机通道定义表建立 MQI 通道的信息, 请参阅 [IBM MQ MQI clients 概述](#)。

使用客户机通道定义表建立通道后, 可以使用 IBM MQ Explorer 通过支持 TLS 的 MQI 通道连接到远程队列管理器, 如第 118 页的『[托管远程队列管理器的系统上的任务](#)』和第 118 页的『[托管 IBM MQ Explorer 的系统上的任务](#)』中所述。

托管远程队列管理器的系统上的任务

在托管远程队列管理器的系统上，执行以下任务：

1. 定义通道的服务器连接和客户机连接对，并为这两个通道上的服务器连接上的 `SSLCPH` 属性指定相应的值。有关 `SSLCPH` 属性的更多信息，请参阅 [使用 TLS 保护通道](#)
2. 将在队列管理器的 `@ipcc` 目录中找到的通道定义表 `AMQCLCHL.TAB` 发送到托管 IBM MQ Explorer 的系统。
3. 在指定的端口上启动 TCP/IP 侦听器。
4. 将 CA 和个人 TLS 证书放入队列管理器的 SSL 目录中：
 - UNIX and Linux 系统的 `/var/mqm/qmgrs/+QMNAME+/SSL`
 - Windows 系统的 `C:\Program Files\IBM\MQ\qmgrs\+QMNAME+\SSL`其中 `+QMNAME+` 是表示队列管理器名称的令牌。
5. 创建名为 `key.kdb` 的 CMS 类型的密钥数据库文件。通过选中 `strmqikm` (iKeyman) 中的选项，将密码隐藏在文件中 GUI，或者将 `-stash` 选项与 `runmqckm` 命令配合使用。
6. 将 CA 证书添加到在上一步中创建的密钥数据库。
7. 将队列管理器的个人证书导入到密钥数据库中。

有关在 Windows 系统上使用 TLS 的更多详细信息，请参阅 [在 UNIX, Linux, and Windows 上使用 TLS](#)。

托管 IBM MQ Explorer 的系统上的任务

在托管 IBM MQ Explorer 的系统上，执行以下任务：

1. 创建名为 `key.jks` 的 JKS 类型的密钥数据库文件。设置此密钥数据库文件的密码。

IBM MQ Explorer 将 Java 密钥库文件 (JKS) 用于 TLS 安全性，因此要为 IBM MQ Explorer 配置 TLS 而创建的密钥库文件必须与此匹配。
2. 将 CA 证书添加到在上一步中创建的密钥数据库。
3. 将队列管理器的个人证书导入到密钥数据库中。
4. 在 Windows 和 Linux 系统上，使用系统菜单，MQ Explorer 可执行文件或 `strmqcfg` 命令启动 IBM MQ Explorer。
5. 从 IBM MQ Explorer 工具栏中，单击 **窗口->首选项**，然后展开 **IBM MQ Explorer**，然后单击 **SSL 客户机证书库**。在 "可信证书库" 和 "个人证书库" 中输入在 [第 118 页](#) 的『托管 IBM MQ Explorer 的系统上的任务』的步骤 1 中创建的 JKS 文件的名称和密码，然后单击 **确定**。
6. 关闭 "首选项" 窗口，然后右键单击 **队列管理器**。单击 **显示/隐藏队列管理器**，然后在 "显示/隐藏队列管理器" 屏幕上单击 **添加**。
7. 输入队列管理器的名称，然后选择 **直接连接** 选项。单击 "下一步"。
8. 选择 **使用客户机通道定义表 (CCDT)**，并在托管远程队列管理器的系统上指定您在步骤 2 中的 [第 118 页](#) 的『托管远程队列管理器的系统上的任务』中从远程队列管理器传输的通道表文件的位置。
9. 单击 **完成**。现在可以从 IBM MQ Explorer 访问远程队列管理器。

使用 IBM MQ Explorer 通过另一个队列管理器进行连接

IBM MQ Explorer 允许您通过中间队列管理器连接到队列管理器，IBM MQ Explorer 已连接到该中间队列管理器。

在这种情况下，IBM MQ Explorer 会将 PCF 命令消息放入中间队列管理器，并指定以下内容：

- 对象描述符 (MQOD) 中的 `ObjectQMGrName` 参数作为目标队列管理器的名称。有关队列名称解析的更多信息，请参阅 [名称解析](#)。
- 消息描述符 (MQMD) 中的 `UserIdentifier` 参数作为本地 `userId`。

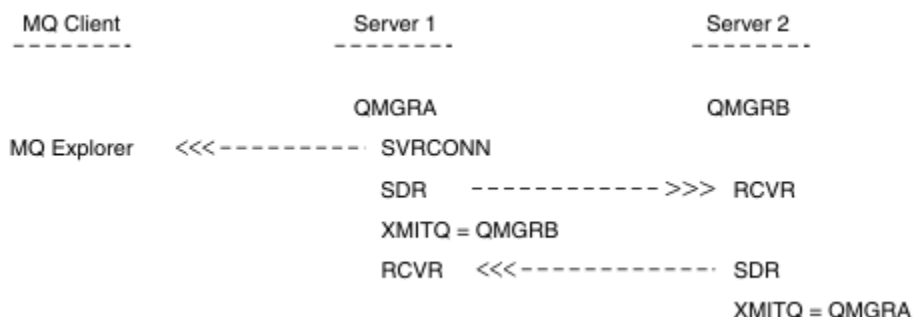
如果连接随后用于通过中间队列管理器连接到目标队列管理器，那么 `userId` 将再次在消息描述符 (MQMD) 的 `UserIdentifier` 参数中流动。为了使目标队列管理器上的 MCA 侦听器接受此消息，必须设置 `MCAUSER` 属性，或者必须已存在具有 `put` 权限的 `userId`。

目标队列管理器上的命令服务器将消息放入传输队列，并在消息描述符 (MQMD) 中的 *UserIdentifier* 参数中指定 *userId*。要使此 put 操作成功，具有 put 权限的目标队列管理器上必须已存在 *userId*。

以下示例显示如何通过中间队列管理器将队列管理器连接到 IBM MQ Explorer。

建立与队列管理器的远程管理连接。验证：

- 服务器上的队列管理器处于活动状态，并且定义了服务器连接通道 (SVRCONN)。
- 侦听器处于活动状态。
- 命令服务器处于活动状态。
- SYSTEM.MQ EXPLORER.REPLY.MODEL 队列，并且您具有足够的权限。
- 队列管理器侦听器，命令服务器和发送方通道已启动。



在本示例中：

- IBM MQ Explorer 已使用客户机连接连接到队列管理器 QMGRA (在 Server1 上运行)。
- Server2 上的队列管理器 QMGRB 现在可以通过中间队列管理器 (QMGRA) 连接到 IBM MQ Explorer
- 使用 IBM MQ Explorer 连接到 QMGRB 时，选择 QMGRA 作为中间队列管理器

在此情况下，不存在从 IBM MQ Explorer 到 QMGRB 的直接连接；与 QMGRB 的连接是通过 QMGRA 进行的。

Server2 上的队列管理器 QMGRB 使用发送方/接收方通道连接到 Server1 上的 QMGRA。必须以可进行远程管理的方式设置 QMGRA 与 QMGRB 之间的通道；请参阅第 170 页的『为远程管理准备通道和传输队列』。

在 IBM MQ Explorer 中显示和隐藏队列管理器和集群

IBM MQ Explorer 一次可以显示多个队列管理器。从 "显示/隐藏队列管理器" 面板 (可从 "队列管理器" 树节点的菜单中选择)，您可以选择是否显示有关另一台 (远程) 机器的信息。将自动检测本地队列管理器。

要显示远程队列管理器：

1. 右键单击 **队列管理器** 树节点，然后选择 **显示/隐藏队列管理器**。
2. 单击**添加**。将显示 "显示/隐藏队列管理器" 面板。
3. 在提供的字段中输入远程队列管理器的名称和主机名或 IP 地址。

主机名或 IP 地址用于使用其缺省服务器连接通道 SYSTEM.ADMIN.SVRCONN 或用户定义的服务器连接通道。

4. 单击**完成**。

"显示/隐藏队列管理器" 面板还显示所有可视队列管理器的列表。您可以使用此面板在导航视图中隐藏队列管理器。

如果 IBM MQ Explorer 显示作为集群成员的队列管理器，那么将检测到该集群并自动显示该集群。

要从此面板导出远程队列管理器的列表，请执行以下操作：

1. 关闭 "显示/隐藏队列管理器" 面板。
2. 右键单击 IBM MQ Explorer 的 "导航" 窗格中的最高 **IBM MQ** 树节点，然后选择 **导出 IBM MQ Explorer 设置**

3. 单击 **IBM MQ Explorer > IBM MQ Explorer 设置**
4. 选择 **连接信息 > 远程队列管理器**。
5. 选择要在其中存储导出的设置的文件。
6. 最后，单击 **完成** 以将远程队列管理器连接信息导出到指定文件。

要导入远程队列管理器列表，请执行以下操作：

1. 右键单击 IBM MQ Explorer 的 "导航" 窗格中的最高 **IBM MQ** 树节点，然后选择 **导入 IBM MQ Explorer 设置**
2. 单击 **IBM MQ Explorer > IBM MQ Explorer 设置**
3. 单击 **浏览**，然后浏览到包含远程队列管理器连接信息的文件的路径。
4. 单击 **打开**。如果该文件包含远程队列管理器列表，那么将选中 **连接信息 > 远程队列管理器** 框。
5. 最后，单击 **完成** 以将远程队列管理器连接信息导入到 IBM MQ Explorer 中。

集群成员资格和 IBM MQ Explorer

IBM MQ Explorer 需要有关作为集群成员的队列管理器的信息。

如果队列管理器是集群的成员，那么将自动填充集群树节点。

如果队列管理器在 IBM MQ Explorer 运行时成为集群的成员，那么必须使用有关集群的最新管理数据来维护 IBM MQ Explorer，以便它可以与这些集群进行有效通信，并在请求时显示正确的集群信息。要执行此操作，IBM MQ Explorer 需要以下信息：

- 存储库队列管理器的名称
- 存储库队列管理器的连接名称 (如果它在远程队列管理器上)

通过此信息，IBM MQ Explorer 可以：

- 使用存储库队列管理器来获取集群中队列管理器的列表。
- 管理属于集群成员且处于受支持平台和命令级别的队列管理器。

在下列情况下，无法进行管理：

- 所选存储库变为不可用。IBM MQ Explorer 不会自动切换到备用存储库。
- 无法通过 TCP/IP 联系所选存储库。
- 所选存储库正在运行在 IBM MQ Explorer 不支持的平台和命令级别上运行的队列管理器上。

可以管理的集群成员可以是本地成员，也可以是远程成员 (如果可以使用 TCP/IP)。IBM MQ Explorer 直接连接到作为集群成员的本地队列管理器，而不使用客户机连接。

IBM MQ Explorer 的数据转换

IBM MQ Explorer 以 CCSID 1208 (UTF-8) 工作。这使 IBM MQ Explorer 能够正确显示来自远程队列管理器的数据。无论是直接连接到队列管理器，还是使用中间队列管理器，IBM MQ Explorer 都要求将所有入局消息转换为 CCSID 1208 (UTF-8)。

如果尝试在 IBM MQ Explorer 与具有 IBM MQ Explorer 无法识别的 CCSID 的队列管理器之间建立连接，那么将发出错误消息。

[代码页转换](#)中描述了受支持的转换。

Windows 使用 IBM MQ 任务栏应用程序 (仅限 Windows)

IBM MQ 任务栏应用程序在服务器上的 Windows 系统托盘中显示图标。该图标为您提供 IBM MQ 的当前状态以及一个菜单，您可以从该菜单执行一些简单操作。

在 Windows 上，IBM MQ 图标位于服务器上的系统托盘中，并以颜色编码的状态符号覆盖，这可能具有下列其中一个含义：

绿色

工作正常; 目前无警报

蓝色

不确定; IBM MQ 正在启动或关闭

黄色

警报; 一个或多个服务发生故障或已发生故障

要显示菜单, 请右键单击 IBM MQ 图标。从菜单中, 可以执行以下操作:

- 单击 **打开** 以打开 IBM MQ 警报监视器。
- 单击 **退出** 以退出 IBM MQ 任务栏应用程序。
- 单击 **IBM MQ Explorer** 以启动 IBM MQ Explorer。
- 单击 **停止 IBM MQ** 以停止 IBM MQ。
- 单击关于 **IBM MQ** 以显示有关 IBM MQ 警报监视器的信息。

Windows IBM MQ 警报监视器应用程序 (仅限 Windows)

IBM MQ 警报监视器是一个错误检测工具, 用于识别和记录本地机器上的 IBM MQ 问题。

警报监视器显示有关 IBM MQ 服务器本地安装的当前状态的信息。它还监视 Windows 高级配置和电源接口 (ACPI), 并确保强制实施 ACPI 设置。

从 IBM MQ 警报监视器中, 可以执行以下操作:

- 直接访问 IBM MQ Explorer
- 查看与所有未完成的警报相关的信息
- 关闭本地机器上的 IBM MQ 服务
- 通过网络将警报消息路由到可配置的用户帐户, 或者路由到 Windows 工作站或服务器

管理本地 IBM MQ 对象

您可以管理本地 IBM MQ 对象以支持使用消息队列接口 (MQI) 的应用程序。

关于此任务

在此上下文中, 本地管理意味着创建, 显示, 更改, 复制和删除 IBM MQ 对象。

除了本节中描述的方法外, 您还可以使用 IBM MQ Explorer 来管理本地 IBM MQ 对象。有关更多信息, 请参阅第 114 页的『[使用 IBM MQ Explorer 进行管理](#)』。

过程

- 使用以下主题中的信息来帮助您管理本地 IBM MQ 对象。
 - [使用 MQI 的应用程序](#)
 - [第 9 页的『使用 MQSC 命令管理 MQ』](#)
 - [第 125 页的『使用队列管理器』](#)
 - [第 127 页的『使用本地队列』](#)
 - [第 132 页的『使用别名队列』](#)
 - [第 149 页的『使用模型队列』](#)
 - [第 156 页的『使用服务』](#)
 - [第 162 页的『管理用于触发的对象』](#)

启动和停止队列管理器

使用指向进一步信息的链接来停止和启动队列管理器的简介。

关于此任务

您可以使用以下命令来启动和停止队列管理器:

- 要启动队列管理器, 请使用 **strmqm** 命令。
- 要停止队列管理器, 请使用 **endmqm** 命令, 该命令提供了三种方法来停止队列管理器: 受控或停顿关闭, 立即关闭和先发制人关闭。

Windows **Linux** 或者, 在 Windows 和 Linux 上, 可以使用 IBM MQ Explorer 来启动和停止队列管理器。

Windows 在 Windows 上, 您可以将队列管理器配置为在系统使用 IBM MQ Explorer 启动时自动启动。

过程

1. 有关如何启动队列管理器的更多信息, 请参阅 [启动队列管理器](#)。
2. 有关如何停止队列管理器的更多信息, 请参阅 [停止队列管理器](#)。

ULW 手动停止队列管理器

如果用于停止和除去队列管理器的标准方法失败, 那么您可以尝试手动停止队列管理器。

关于此任务

停止队列管理器的标准方法是使用 **endmqm** 命令, 如 [停止队列管理器](#) 中所述。如果无法以标准方式停止队列管理器, 那么可以尝试手动停止队列管理器。执行此操作的方式取决于您正在使用的平台。

过程

- **Windows**
要在 Windows 上停止队列管理器, 请参阅 [第 122 页的『在 Windows 上手动停止队列管理器』](#)。
- **Linux** **UNIX**
要在 UNIX 或 Linux 上停止队列管理器, 请参阅 [第 123 页的『在 UNIX 和 Linux 上手动停止队列管理器』](#)。

相关信息

[在 Multiplatforms 版上创建和管理队列管理器](#)
恩德姆

Windows 在 Windows 上手动停止队列管理器

如果无法使用 **endmqm** 命令在 Windows 上停止队列管理器, 那么可以尝试通过结束正在运行的任何进程并停止 IBM MQ 服务来手动停止队列管理器。

关于此任务

提示: Windows 任务管理器和 **tasklist** 命令提供有关任务的有限信息。有关帮助确定哪些进程与特定队列管理器相关的更多信息, 请考虑使用可从 Microsoft Web 站点 (<https://www.microsoft.com>) 下载的工具, 例如 *Process Explorer* (procexp.exe)。

要在 Windows 上停止队列管理器, 请完成以下步骤。

过程

1. 使用 Windows 任务管理器列出正在运行的进程的名称 (标识)。
2. 使用 Windows 任务管理器或 **taskkill** 命令按以下顺序结束进程 (如果这些进程正在运行):

| 表 12: 要在运行时停止的 Windows 进程 | |
|---------------------------|------------------|
| 进程名称 | 描述 |
| AMQZMUC0 | 关键流程管理器 |
| AMQZXMA0 | 执行控制器 |
| AMQZFUMA | OAM 进程 |
| AMQZLAA0 | LQM 代理程序 |
| AMQZLSA0 | LQM 代理程序 |
| AMQZMUFO | 实用程序管理器 |
| AMQZMGR0 | 进程控制器 |
| AMQZMUR0 | 可重新启动的进程管理器 |
| AMQFQPUB | 发布预订流程 |
| AMQFCXBA | 代理工作程序进程 |
| AMQRMPPA | 进程池进程 |
| AMQCRSTA | 非线程响应程序作业进程 |
| AMQCRS6B | LU62 接收方通道和客户机连接 |
| AMQRRMFA | 存储库进程 (针对集群) |
| AMQPCSEA | 命令服务器 |
| runmqtrm | 调用服务器的触发器监视器 |
| RUNMQDLQ | 调用死信队列处理程序 |
| 运行 MQCHI | 通道启动程序进程 |
| 运行 MQLSR | 通道侦听器进程 |
| AMQXSSVN | 共享内存服务器 |

3. 从 Windows 控制面板上的 **管理工具 > 服务** 停止 IBM MQ 服务。
4. 如果尝试了所有方法并且队列管理器尚未停止，请重新引导系统。

Linux > UNIX 在 UNIX 和 Linux 上手动停止队列管理器

如果无法使用 `endmqm` 命令在 UNIX 和 Linux 上停止队列管理器，那么可以尝试通过结束正在运行的任何进程并停止 IBM MQ 服务来手动停止队列管理器。

关于此任务

要在 UNIX 和 Linux 上停止队列管理器，请完成以下步骤。

如果手动停止队列管理器，那么可能会采用 FFST，并将 FDC 文件放在 `/var/mqm/errors` 中。这不应视为队列管理器中的缺陷。

即使在您使用手动停止队列管理器的方法将其停止之后，该队列管理器也将正常重新启动。

过程

1. 使用 `ps` 命令查找仍在运行的队列管理器程序的进程标识。
例如，如果队列管理器名为 `QMNAME`，请使用以下命令：

```
ps -ef | grep QMNAME
```

2. 使用 **kill** 命令结束仍在运行的任何队列管理器进程，并指定使用 **ps** 命令发现的 PID。

要结束进程，请使用 **kill -KILL <pid>** 或等效的 **kill -9 <pid>** 命令。

您必须逐个处理要杀死的 PID，每次都发出该命令。

要点: 如果使用除 **9(SIGKILL)** 以外的任何信号，那么进程可能不会停止，并且您将获得不可预测的结果。

按以下顺序结束进程:

| 进程名称 | 描述 |
|----------|------------------|
| amqzmuc0 | 关键流程管理器 |
| amqzma0 | 执行控制器 |
| 阿姆格兹富马 | OAM 进程 |
| amqzlaa0 | LQM 代理程序 |
| amqzlsa0 | LQM 代理程序 |
| amqzmuf0 | 实用程序管理器 |
| amqzmur0 | 可重新启动的进程管理器 |
| amqzmgr0 | 进程控制器 |
| 阿姆格夫格普卜 | 发布预订流程 |
| 阿姆格夫奇巴 | 代理工作程序进程 |
| 阿姆克姆帕 | 进程池进程 |
| 阿姆克斯塔 | 非线程响应程序作业进程 |
| amqcrs6b | LU62 接收方通道和客户机连接 |
| 阿姆格勒姆法 | 存储库进程 (针对集群) |
| 安格普切西 | 命令服务器 |
| runmqtrm | 调用服务器的触发器监视器 |
| runmqdlq | 调用死信队列处理程序 |
| 鲁姆基 | 通道启动程序进程 |
| 运行 mqlsr | 通道侦听器进程 |

注: 您可以使用 **kill -9** 命令来结束未能停止的进程。

停止 MQI 通道

对服务器连接通道发出 STOP CHANNEL 命令时，可以选择使用何种方法来停止客户机连接通道。这意味着可以控制发出 MQGET 等待调用的客户机通道，并且您可以决定如何以及何时停止该通道。

可以使用三种方式发出 STOP CHANNEL 命令，指示如何停止通道:

停顿

在处理任何当前消息后停止通道。

如果启用了共享对话，那么 IBM MQ MQI client 将及时了解停止请求; 此时间取决于网络的速度。由于发出对 IBM MQ 的后续调用，客户机应用程序将知道停止请求。

Force

立即停止通道。

终止

立即停止通道。如果通道作为进程运行，那么它可以终止通道的进程，或者如果通道作为线程运行，那么它的线程。

这是一个多阶段的过程。如果使用方式终止，那么会尝试停止服务器连接通道，首先使用方式停顿，然后使用方式强制，如果需要使用方式终止。在终止的不同阶段，客户机可以接收不同的返回码。如果进程或线程已终止，那么客户机将接收到通信错误。

返回到应用程序的返回码根据发出的 MQI 调用和发出的 STOP CHANNEL 命令而有所不同。客户机将接收到 MQRC_CONNECTION_QUIESCING 或 MQRC_CONNECTION_BROKEN 返回码。如果客户机检测到 MQRC_CONNECTION_QUIESCING，那么应尝试完成当前事务并终止。对于 MQRC_CONNECTION_BROKEN，这是不可能的。如果客户机未完成事务并以足够快的速度终止，那么将在几秒后获取 CONNECTION_BROKEN。使用 MODE (FORCE) 或 MODE (TERMINATE) 的 STOP CHANNEL 命令比使用 MODE (QUIESCE) 更有可能导致 CONNECTION_BROKEN。

相关信息

[通道](#)

使用队列管理器

可用于显示或更改队列管理器属性的 MQSC 命令的示例。

显示队列管理器属性

要显示在 **runmqsc** 命令上指定的队列管理器的属性，请使用 **DISPLAY QMGR MQSC** 命令：

```
DISPLAY QMGR
```

此命令的典型输出显示在 [第 126 页的图 16](#) 中

```

DISPLAY QMGR
  1 : DISPLAY QMGR
AMQ8408: Display Queue Manager details.
QMNAME(QM1)
ACCTINT(1800)
ACCTQ(OFF)
ACTVCONO (DISABLED)
ALTDATE(2012-05-27)
AUTHOREV(DISABLED)
CHAD(DISABLED)
CHADEXIT( )
CLWLDATA( )
CLWLEN(100)
CLWLUSEQ(LOCAL)
CMDLEVEL(800)
CONFIGEV(DISABLED)
CRTIME(16.14.01)
DEFXMITQ( )
DISTL(YES)
IPADDRV(IPV4)
LOGGERSV(DISABLED)
MAXHANDS(256)
MAXPROPL(NOLIMIT)
MAXUMSGS(10000)
MONCHL(OFF)
PARENT( )
PLATFORM(WINDOWSNT)
PSNPMMSG(DISCARD)
PSSYNCP(1FPER)
PSMODE(ENABLED)
REPOS( )
ROUTEREC(MSG)
SCMDSERV(QMGR)
SSLCRYP( )
SSLFIPS(NO)
MQ\Data\qmgrs\QM1\ssl\key)
SSLKEYC(0)
STATCHL(OFF)
STATMQI(OFF)
STRSTPEV(ENABLED)
TREELIFE(1800)
ACCTCONO(DISABLED)
ACCTMQI(OFF)
ACTIVREC(MSG)
ACTVTRC(OFF)
ALTTIME(16.14.01)
CCSID(850)
CHADEV(DISABLED)
CHLEV(DISABLED)
CLWLXIT( )
CLWLMRUC(999999999)
CMDEV(DISABLED)
COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)
CRDATE(2011-05-27)
DEADQ( )
DESCR( )
INHIBTEV(DISABLED)
LOCALEV(DISABLED)
MARKINT(5000)
MAXMSGL(4194304)
MAXPRTY(9)
MONACLS(QMGR)
MONQ(OFF)
PERFMEV(DISABLED)
  PSRTYCNT(5)
PSNPRES(NORMAL)
QMID(QM1_2011-05-27_16.14.01)
REMOETEVEV(DISABLED)
REPOSNL( )
SCHINIT(QMGR)
SSLCRLNL( )
SSLEV(DISABLED)
SSLKEYR(C:\Program Files\IBM\WebSphere
STATACLS(QMGR)
STATINT(1800)
STATQ(OFF)
SYNCP
TRIGINT(999999999)

```

图 16: DISPLAY QMGR 命令的典型输出

注: SYNCPT 是只读队列管理器属性。

ALL 参数是 **DISPLAY QMGR** 命令的缺省值。它显示所有队列管理器属性。特别是，输出会告诉您缺省队列管理器名称，死信队列名称和命令队列名称。

您可以通过输入以下命令来确认这些队列是否存在:

```
DISPLAY QUEUE (SYSTEM.*)
```

这将显示与词干 SYSTEM.* 匹配的队列的列表。括号是必需的。

更改队列管理器属性

要更改在 **runmqsc** 命令上指定的队列管理器的属性，请使用 **MQSC** 命令 **ALTER QMGR**，指定要更改的属性和值。例如，使用以下命令来变更 **jupiter.queue.manager** 的属性:

```
runmqsc jupiter.queue.manager
ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

ALTER QMGR 命令更改使用的死信队列，并启用禁止事件。

相关信息

[在 Multiplatforms 版上创建和管理队列管理器](#)

[队列管理器的属性](#)

[runmqsc \(运行 MQSC 命令\)](#)

[显示队列管理器](#)

[ALTER QMGR](#)

使用本地队列

本部分包含可用于管理本地队列，模型队列和别名队列的一些 MQSC 命令的示例。

有关这些命令的详细信息，请参阅 [MQSC 命令](#)。

相关信息

[队列的命名限制](#)

[其他对象的命名限制](#)

定义本地队列

对于应用程序，本地队列管理器是应用程序所连接的队列管理器。由本地队列管理器管理的队列据说是该队列管理器的本地队列。

使用 MQSC 命令 **DEFINE QLOCAL** 来创建本地队列。您还可以使用缺省本地队列定义中定义的缺省值，也可以修改缺省本地队列的队列特征。

注: 缺省本地队列为 SYSTEM.DEFAULT.LOCAL.QUEUE，它是在系统安装时创建的。

例如，后面的 **DEFINE QLOCAL** 命令定义为 ORANGE.LOCAL.QUEUE：

- 它针对获取启用，针对放置启用，并按优先级顺序运行。
- 它是普通队列；它不是启动队列或传输队列，并且不会生成触发器消息。
- 最大队列深度为 5000 条消息；最大消息长度为 4194304 字节。

```
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) +
  DESCR('Queue for messages from other systems') +
  PUT(ENABLED) +
  GET(ENABLED) +
  NOTRIGGER +
  MSGDLVSQ(PRIORITY) +
  MAXDEPTH(5000) +
  MAXMSGL(4194304) +
  USAGE(NORMAL)
```

注:

1. 除了描述的值以外，示例中显示的所有属性值都是缺省值。这些示例是为了明确说明目的而包含的。如果您确定缺省值是您想要的或尚未更改的值，那么可以省略这些值。另请参阅第 127 页的『[显示缺省对象属性](#)』。
2. **USAGE(NORMAL)** 指示此队列不是传输队列。
3. 如果已在同一队列管理器上具有名为 ORANGE.LOCAL.QUEUE，此命令失败。如果要覆盖队列的现有定义，请使用 **REPLACE** 属性，但另请参阅第 128 页的『[更改本地队列属性](#)』。

相关信息

[DEFINE QLOCAL](#)

显示缺省对象属性

您可以使用 **DISPLAY QUEUE** 命令来显示在定义 IBM MQ 对象时从缺省对象获取的属性。

当您定义 IBM MQ 对象时，它将采用您未从缺省对象中指定的任何属性。例如，当您定义本地队列时，该队列将从缺省本地队列（称为 SYSTEM.DEFAULT.LOCAL.QUEUE。要确切查看这些属性的内容，请使用以下命令：

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE)
```

此命令的语法与相应 **DEFINE** 命令的语法不同。在 **DISPLAY** 命令上，只能提供队列名称，而在 **DEFINE** 命令上，必须指定队列类型，即 **QLOCAL**，**QALIAS**，**QMODEL** 或 **QREMOTE**。

您可以通过单独指定属性来选择性地显示这些属性。例如：

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
MAXDEPTH +
MAXMSGL +
CURDEPTH;
```

此命令显示三个指定的属性，如下所示：

```
AMQ8409: Display Queue details.
QUEUE(ORANGE.LOCAL.QUEUE)      TYPE(QLOCAL)
CURDEPTH(0)                     MAXDEPTH(5000)
MAXMSGL(4194304)
```

CURDEPTH 是当前队列深度，即队列上的消息数。这是要显示的有用属性，因为通过监视队列深度，可以确保队列不会变满。

相关信息

[DISPLAY QUEUE](#)

[DEFINE 队列](#)

复制本地队列定义

您可以使用 **DEFINE** 命令上的 **LIKE** 属性来复制队列定义。

例如：

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
LIKE (ORANGE.LOCAL.QUEUE)
```

此命令将创建与原始队列 **ORANGE.LOCAL.QUEUE**，而不是系统缺省本地队列的队列。输入要复制的队列的名称，与创建队列时输入的名称完全相同。如果名称包含小写字母，请将名称括在单引号中。

您还可以使用此格式的 **DEFINE** 命令来复制队列定义，但将一个或多个更改替换为原始属性。例如：

```
DEFINE QLOCAL (THIRD.QUEUE) +
LIKE (ORANGE.LOCAL.QUEUE) +
MAXMSGL(1024);
```

此命令将复制队列 **ORANGE.LOCAL.QUEUE** 到队列 **THIRD.QUEUE**，但指定新队列上的最大消息长度为 1024 字节，而不是 4194304 字节。

注：

1. 在 **DEFINE** 命令上使用 **LIKE** 属性时，仅复制队列属性。您未在复制队列上的消息。
2. 如果在未指定 **LIKE** 的情况下定义本地队列，那么它与 **DEFINE LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE)**。

相关信息

[DEFINE 队列](#)

更改本地队列属性

您可以通过两种方式更改队列属性，即使用 **ALTER QLOCAL** 命令或带有 **REPLACE** 属性的 **DEFINE QLOCAL** 命令。

在第 127 页的『定义本地队列』中，队列为 **ORANGE.LOCAL.QUEUE**。例如，假设您希望将此队列上的最大消息长度减少到 10,000 字节。

- 使用 **ALTER** 命令:

```
ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
```

此命令会更改单个属性，即最大消息长度的属性; 所有其他属性保持不变。

- 将 **DEFINE** 命令与 **REPLACE** 选项配合使用，例如:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
```

此命令不仅会更改最大消息长度，还会更改所有其他属性 (给定它们的缺省值)。现在已启用该队列，而先前已禁止将其放入。启用放置是缺省值，由队列 `SYSTEM.DEFAULT.LOCAL.QUEUE`。

如果减小现有队列上的最大消息长度，那么现有消息不受影响。但是，任何新消息都必须满足新条件。

相关信息

[ALTER 队列](#)

[ALTER QLOCAL](#)

[DEFINE 队列](#)

[DEFINE QLOCAL](#)

清除本地队列

您可以使用 **CLEAR** 命令来清除本地队列。

例如，要从名为 `MAGENTA.QUEUE`，使用以下命令:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

注: 没有使您能够改变主意的提示; 当您按 **Enter** 键时，消息将丢失。

在下列情况下，无法清除队列:

- 有未落实的消息已放在同步点下的队列上。
- 某个应用程序当前打开了该队列。

相关信息

[清除 QLocal](#)

删除本地队列

可以使用 `MQSC` 命令 **DELETE QLOCAL** 来删除本地队列。

如果队列上有未落实的消息，那么无法删除该队列。但是，如果队列具有一条或多条已落实的消息，并且没有未落实的消息，那么仅当指定 **PURGE** 选项时，才能将其删除。例如:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

指定 **NOPURGE** 而不是 **PURGE** 可确保如果队列包含任何已落实的消息，那么不会将其删除。

相关信息

[删除 QLOCAL](#)

浏览队列

IBM MQ 提供了样本队列浏览器，可用于查看队列中消息的内容。浏览器以源格式和可执行格式提供。

`MQ_INSTALLATION_PATH` 表示 IBM MQ 安装所在的高级目录。

Windows

在 Windows 上，样本队列浏览器的文件名和路径如下所示:

来源

`MQ_INSTALLATION_PATH\tools\c\samples\`

可执行文件

`MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe`

 在 UNIX 和 Linux 上，文件名和路径如下所示:

来源

`MQ_INSTALLATION_PATH/samp/amqsbcg0.c`

可执行文件

`MQ_INSTALLATION_PATH/samp/bin/amqsbcg`

样本需要两个输入参数，即队列名称和队列管理器名称。例如：

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

此命令的典型结果显示在 [第 131 页的图 17](#) 中。

```

AMQSBCG0 - starts here
*****

MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'

MQGET of message number 1
****Message descriptor****

  StrucId : 'MD ' Version : 2
  Report  : 0 MsgType : 8
  Expiry  : -1 Feedback : 0
  Encoding : 546 CodedCharSetId : 850
  Format   : 'MQEVENT '
  Priority : 0 Persistence : 0
  MsgId   : X'414D512073617475726E2E71756575650005D30033563DB8'
  CorrelId : X'0000000000000000000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ      : '
  ReplyToQMgr   : 'saturn.queue.manager'
  ** Identity Context
  UserIdentifier : '
  AccountingToken :
  X'0000000000000000000000000000000000000000000000000000000000000000'
  ApplIdentityData : '
  ** Origin Context
  PutApplType : '7'
  PutApplName : 'saturn.queue.manager'
  PutDate : '19970417' PutTime : '15115208'
  ApplOriginData : '

  GroupId : X'0000000000000000000000000000000000000000000000000000000000000000'
  MsgSeqNumber : '1'
  Offset : '0'
  MsgFlags : '0'
  OriginalLength : '104'

**** Message ****

length - 104 bytes

00000000: 0700 0000 2400 0000 0100 0000 2C00 0000 '.....->.....'
00000010: 0100 0000 0100 0000 0100 0000 AE08 0000 '.....'
00000020: 0100 0000 0400 0000 4400 0000 DF07 0000 '.....D.....'
00000030: 0000 0000 3000 0000 7361 7475 726E 2E71 '...0...saturn.q'
00000040: 7565 7565 2E6D 616E 6167 6572 2020 2020 'ueue.manager'
00000050: 2020 2020 2020 2020 2020 2020 2020 2020 '
00000060: 2020 2020 2020 2020 '

No more messages
MQCLOSE
MQDISC

```

图 17: 来自队列浏览器的典型结果

相关信息

[Browser 样本程序](#)

启用大型队列

IBM MQ 支持大于 2 GB 的队列。

Windows 在 Windows 系统上，提供了对大型文件的支持，而无需任何其他支持。

Linux **UNIX** 在 UNIX 和 Linux 系统上，您需要显式启用大型文件支持，然后才能创建大于 2 GB 的队列文件。请参阅操作系统文档，以获取有关如何执行此操作的信息。

某些实用程序 (例如 tar) 无法处理大于 2 GB 的文件。在启用大型文件支持之前，请查看操作系统文档以获取有关使用的实用程序限制的信息。

有关规划队列所需的存储量的信息，请参阅特定于平台的性能报告的 [IBM MQ 系列-性能报告](#)。

使用别名队列

您可以定义别名队列来间接引用其他队列或主题。

V9.0.0.1 V9.0.1



注意: 分发列表不支持使用指向主题对象的别名队列。从 IBM MQ 9.0.1 和 IBM MQ 9.0.0 Fix Pack 1 开始, 如果别名队列指向分发列表中的主题对象, 那么 IBM MQ 会返回 MQRC_ALIAS_BASE_Q_TYPE_ERROR。

别名队列引用的队列可以是以下任意项:

- 本地队列 (请参阅第 127 页的『定义本地队列』)。
- 远程队列的本地定义 (请参阅第 174 页的『创建远程队列的本地定义』)。
- 主题。

别名队列不是真实的队列, 而是在运行时解析为真实 (或目标) 队列的定义。别名队列定义指定目标队列。当应用程序对别名队列进行 MQOPEN 调用时, 队列管理器会将别名解析为目标队列名称。

别名队列无法解析为本地定义的其他别名队列。然而, 别名队列可解析为在本地队列管理器所属集群中的其他位置定义的别名队列。请参阅[名称解析](#), 以获取进一步的信息。

别名队列有助于:

- 为不同的应用程序提供对目标队列的不同访问权限级别。
- 允许不同的应用程序以不同方式使用相同的队列。(或许您想指定不同的缺省优先级或不同的缺省持久性值。)
- 简化维护、迁移和工作负载均衡。(或许您想在无须更改应用程序的情况下更改目标队列名称, 此时应用程序继续使用别名。)

例如, 假设开发了一个应用程序, 用于将消息放入队列 MY.ALIAS.QUEUE。它在发出 MQOPEN 请求时指定此队列的名称, 如果它将消息放入此队列, 那么将间接指定此队列的名称。此应用程序不知道此队列是别名队列。对于使用此别名的每个 MQI 调用, 此队列管理器会解析实际队列名称 (可以是在此队列管理器上定义的本地队列或远程队列)。

通过更改 TARGQ 属性的值, 您可以将 MQI 调用重定向至另一个队列 (可能在另一个队列管理器上)。这有助于维护、迁移和负载均衡。

定义别名队列

以下命令创建别名队列:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGET (YELLOW.QUEUE)
```

此命令会将指定 MY.ALIAS.QUEUE 的 MQI 调用重定向至队列 YELLOW.QUEUE。此命令不创建目标队列; 如果队列 YELLOW.QUEUE 在运行时不存在, 那么 MQI 调用会失败。

如果您更改此别名定义, 可以将 MQI 调用重定向至另一个队列。例如:

```
ALTER QALIAS (MY.ALIAS.QUEUE) TARGET (MAGENTA.QUEUE)
```

此命令将 MQI 调用重定向至另一个队列 MAGENTA.QUEUE。

您也可以使用别名队列, 使单个队列 (目标队列) 对于不同的应用程序看起来具有不同的属性。可通过定义两个别名 (对每个应用程序各定义一个) 来实现这一点。假设有两个应用程序:

- 应用程序 ALPHA 可以将消息放入 YELLOW.QUEUE, 但不允许它从此队列取出消息。
- 应用程序 BETA 可以从 YELLOW.QUEUE 取出消息, 但不允许它将消息放入此队列。

以下命令定义对应用程序 ALPHA 启用放入和禁用取出的别名:

```
DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
```

```
TARGET (YELLOW.QUEUE) +
PUT (ENABLED) +
GET (DISABLED)
```

以下命令定义对应用程序 BETA 禁用放入和启用取出的别名：

```
DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
TARGET (YELLOW.QUEUE) +
PUT (DISABLED) +
GET (ENABLED)
```

ALPHA 在其 MQI 调用中使用队列名称 ALPHAS.ALIAS.QUEUE；BETA 使用队列名称 BETAS.ALIAS.QUEUE。它们都访问相同的队列，但以不同的方式访问。

您可以在定义队列别名时使用 LIKE 和 REPLACE 属性，与您将这些属性用于本地队列的方式相同。

对别名队列使用其他命令

您可以使用相应的 MQSC 命令来显示或更改别名队列属性，或删除此别名队列对象。例如：

使用 **DISPLAY QALIAS** 命令显示别名队列的属性：

```
DISPLAY QUEUE (ALPHAS.ALIAS.QUEUE)
```

使用 **ALTER QALIAS** 命令可更改别名解析为的基本队列名称，其中 force 选项强制进行更改，即使队列已打开也是如此：

```
ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGQ(ORANGE.LOCAL.QUEUE) FORCE
```

使用 **DELETE QALIAS** 命令可删除此队列别名：

```
DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

如果应用程序当前打开了该队列，那么您无法删除别名队列。

相关信息

[ALTER QALIAS](#)

[定义 Qalias](#)

[删除 Qalias](#)

[分发列表](#)

使用死信队列

每个队列管理器通常都有一个本地队列用作死信队列，这样就可以存储无法送达正确目标的消息，以供之后检索。告诉队列管理器有关死信队列的信息，并指定如何处理在死信队列中找到的消息。使用死信队列可能会影响传递消息的顺序，因此您可以选择不使用这些消息。

要告知队列管理器有关死信队列的信息，请在 **crtmqm** 命令 (例如 **crtmqm -u DEAD.LETTER.QUEUE**) 上指定死信队列名称，或者使用 **ALTER QMGR** 命令上的 **DEADQ** 属性稍后指定一个。必须先定义死信队列，然后才能使用该队列。

名为 SYSTEM.DEAD.LETTER.QUEUE 随产品一起提供。创建队列管理器时，将自动创建此队列。如果需要，可以修改此定义，并将其重命名。

死信队列没有特殊要求，只是：

- 它必须是本地队列
- 其 MAXMSGL (最大消息长度) 属性必须使队列能够容纳队列管理器必须处理的最大消息 **加上** 死信头 (MQDLH) 的大小

使用死信队列可能会影响传递消息的顺序，因此您可以选择不使用这些消息。设置 `USEDLQ` 通道属性以确定在无法传递消息时是否使用死信队列。可以配置此属性，以便队列管理器的某些功能使用死信队列，而其他功能不使用死信队列。有关在不同 MQSC 命令上使用 `USEDLQ` 通道属性的更多信息，请参阅 [定义通道](#)，[显示通道](#)，[变更通道](#)和 [DISPLAY CLUSQMGR](#)。

IBM MQ 提供了一个死信队列处理程序，用于指定如何处理或除去在死信队列中找到的消息。请参阅第 134 页的『[处理 IBM MQ 死信队列上的消息](#)』。

相关信息

[死信队列](#)

[未送达消息故障诊断](#)

[ALTER QMGR](#)

[crtmqm \(创建队列管理器\)](#)

处理 IBM MQ 死信队列上的消息

为了处理死信队列 (DLQ) 上的消息，IBM MQ 提供了缺省 DLQ 处理程序。该处理程序将 DLQ 上的消息与您定义的规则表中的条目相匹配。

消息可由队列管理器，消息通道代理程序 (MCA) 和应用程序放在 DLQ 上。死信队列上的所有消息都必须以 *dead-letter header* 结构 MQDLH 作为前缀。由队列管理器或消息通道代理程序放在 DLQ 上的消息始终具有此头；将消息放在 DLQ 上的应用程序必须提供此头。MQDLH 结构的原因字段包含用于标识消息在 DLQ 上的原因的原因码。

所有 IBM MQ 环境都需要一个例程来定期处理 DLQ 上的消息。IBM MQ 提供了一个称为死信队列处理程序 (DLQ 处理程序) 的缺省例程，您可以使用 `runmqdlq` 命令来调用该例程。

通过用户编写的规则表向 DLQ 处理程序提供有关在 DLQ 上处理消息的指示信息。即，DLQ 处理程序将 DLQ 上的消息与规则表中的条目相匹配；当 DLQ 消息与规则表中的条目相匹配时，DLQ 处理程序将执行与该条目相关联的操作。

相关信息

[死信队列](#)

[未送达消息故障诊断](#)

IBM i 上的死信队列处理程序

什么是 IBM i 死信队列处理程序，以及如何调用它？

死信队列 (DLQ) (有时称为未传递的消息队列) 是无法传递到其目标队列的消息的保留队列。网络中的每个队列管理器都应该具有关联的 DLQ。

注：通常最好避免将消息放在 DLQ 上。有关 DLQ 的使用和避免的信息，请参阅第 133 页的『[使用死信队列](#)』。

队列管理器，消息通道代理程序和应用程序可以将消息放在 DLQ 上。死信队列上的所有消息都必须以 *dead-letter header* 结构 MQDLH 作为前缀。队列管理器或消息通道代理程序在 DLQ 上放置的消息始终具有 MQDLH。始终向在 DLQ 上放置消息的应用程序提供 MQDLH。MQDLH 结构的原因字段包含用于标识消息在 DLQ 上的原因的原因码。

在所有 IBM MQ 环境中，必须有一个定期运行的例程来处理 DLQ 上的消息。IBM MQ 提供了一个称为死信队列处理程序 (DLQ 处理程序) 的缺省例程，您可以使用 `STRMQMDLQ` 命令来调用该例程。用户编写的规则表向 DLQ 处理程序提供用于处理 DLQ 上的消息的指令。即，DLQ 处理程序将 DLQ 上的消息与规则表中的条目相匹配。当 DLQ 消息与规则表中的条目匹配时，DLQ 处理程序将执行与该条目关联的操作。

调用 DLQ 处理程序

使用 `STRMQMDLQ` 命令来调用 DLQ 处理程序。您可以通过两种方式来命名要处理的 DLQ 和要使用的队列管理器：

- 从命令提示符作为 `STRMQMDLQ` 的参数。例如：

```
STRMQMDLQ UDLMSGQ(ABC1.DEAD.LETTER.QUEUE) SRCMBR(QRULE) SRCFILE(library/QTXTSRC)
MQMNAME(MY.QUEUE.MANAGER)
```

- 在规则表中。例如：

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE)
```

注：规则表是源物理文件中可以使用任何名称的成员。

这些示例适用于名为 ABC1.DEAD.LETTER.QUEUE 的 DLQ，由缺省队列管理器拥有。

如果未如所示指定 DLQ 或队列管理器，那么会将安装的缺省队列管理器与属于该队列管理器的 DLQ 一起使用。

SRMQMDLQ 命令从规则表中获取其输入。

您必须有权访问 DLQ 本身以及 DLQ 上的消息转发到的任何消息队列，以便运行 DLQ 处理程序。您还必须有权采用其他用户的身份，以便 DLQ 使用消息上下文中用户标识的权限将消息放入队列中。

相关信息

[死信队列](#)

[未送达消息故障诊断](#)

IBM i 上的 DLQ 处理程序规则表

死信队列处理程序规则表定义 DLQ 处理程序如何处理到达 IBM i DLQ 的消息。

DLQ 处理程序规则表定义 DLQ 处理程序如何处理到达 DLQ 的消息。规则表中有两种类型的条目：

- 表中的第一个条目(可选)包含控制数据。
- 表中的所有其他条目都是供 DLQ 处理程序遵循的规则。每个规则都包含一个模式(一组消息特征)，该模式与消息匹配，以及当 DLQ 上的消息与指定模式匹配时要执行的操作。规则表中必须至少有一个规则。

规则表中的每个条目都包含一个或多个关键字。

控制数据

本部分描述了可以包含在 DLQ 处理程序规则表中的控制数据条目中的关键字。请注意下列事项：

- 关键字的缺省值(如果有)带有下划线。
- 垂直线 (|) 分隔替代项。只能指定其中之一。
- 所有关键字都是可选的。

INPUTQ (*QueueName* | " ")

要处理的 DLQ 的名称：

1. 您指定为 **STRMQMDLQ** 命令的参数的任何 UDLMSGQ 值(或 *DFT) 将覆盖规则表中的任何 INPUTQ 值。
2. 如果指定空白 UDLMSGQ 值作为 **STRMQMDLQ** 命令的参数，那么将使用规则表中的 INPUTQ 值。
3. 如果指定空白 UDLMSGQ 值作为 **STRMQMDLQ** 命令的参数，并在规则表中指定空白 INPUTQ 值，那么将使用系统缺省死信队列。

INPUTQM (*QueueManager* 名称 | " ")

拥有 INPUTQ 关键字上指定的 DLQ 的队列管理器的名称。

如果未指定队列管理器，或者在规则表中指定 INPUTQM (")，那么系统将使用缺省队列管理器进行安装。

RETRYINT (时间间隔|60)

这是一个时间间隔(以秒为单位)，DLQ 处理程序应在此时间间隔内尝试重新处理在第一次尝试时无法处理的 DLQ 上的消息，并且已请求进行重复尝试。缺省情况下，重试时间间隔为 60 秒。

WAIT (YES|NO|nnn)

当 DLQ 处理程序检测到没有可处理的其他消息时，是否应等待更多消息到达 DLQ。

YES

使 DLQ 处理程序无限期待。

否

当检测到 DLQ 为空或不包含可处理的消息时，导致 DLQ 处理程序终止。

nnn

导致 DLQ 处理程序在检测到队列为空或不包含可处理的消息后，等待 *nnn* 秒，以便新工作到达，然后终止。

对繁忙 DLQ 和 WAIT (NO) 或 WAIT (*nnn*) 指定 WAIT (YES) 对于活动级别较低的 DLQ。如果允许 DLQ 处理程序终止，请使用触发来重新调用该处理程序。

您可以提供 DLQ 的名称作为 **STRMQMDLQ** 命令的输入参数，作为在规则表中包含控制数据的替代方法。如果在规则表和 **STRMQMDLQ** 命令的输入中都指定了任何值，那么 **STRMQMDLQ** 命令上指定的值优先。

注: 如果控制数据条目包含在规则表中，那么它必须是表中的第一个条目。

IBM i IBM i 上的 DLQ 规则 (模式和操作)

每个 IBM i 死信队列规则的模式和操作的描述。

以下是来自 DLQ 处理程序规则表的示例规则:

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +  
ACTION (RETRY) RETRY (3)
```

此规则指示 DLQ 处理程序 3 尝试将由于禁止 MQPUT 和 MQPUT1 而放置在 DLQ 上的任何持久消息传递到其目标队列。

本部分描述了可以包含在规则中的关键字。请注意下列事项:

- 关键字的缺省值 (如果有) 带有下划线。对于大多数关键字，缺省值为 * (星号)，这与任何值匹配。
- 垂直线 (|) 分隔替代项。只能指定其中之一。
- 除 ACTION 以外的所有关键字都是可选的。

此部分以模式匹配关键字 (与 DLQ 上的消息匹配的关键字) 的描述开头。然后，它描述操作关键字 (用于确定 DLQ 处理程序如何处理匹配消息的操作关键字)。

IBM i IBM i 上的 DLQ 模式匹配关键字

在示例中描述了模式匹配关键字。使用这些关键字来指定与 IBM i 死信队列上的消息匹配的值。所有模式匹配关键字都是可选的。

APPLIDAT (*ApplIdentity* 数据|*)

DLQ 上消息的 *ApplIdentity* 数据值，在消息描述符 MQMD 中指定。

APPLNAME (*PutAppl* 名称|*)

发出 MQPUT 或 MQPUT1 调用的应用程序的名称，如 DLQ 上消息的消息描述符 MQMD 的 *PutApplName* 字段中所指定。

APPLTYPE (*PutAppl* 类型|*)

在 DLQ 上的消息的消息描述符 MQMD 中指定的 *PutAppl* 类型值。

DESTQ (*QueueName*|*)

以消息为目标的消息队列的名称。

DESTQM (*QueueManager* 名称|*)

以消息为目标的消息队列的队列管理器名称。

FEEDBACK (反馈|*)

当 *MsgType* 值为 MQMT_REPORT 时，反馈 描述报告的性质。

您可以使用符号名称。例如，可以使用符号名称 MQFB_COA 来标识 DLQ 上需要确认其到达其目标队列的那些消息。

格式 (格式|*)

消息发送方用于描述消息数据格式的名称。

MSGTYPE (MsgType|*)

DLQ 上消息的消息类型。

您可以使用符号名称。例如，可以使用符号名称 MQMT_REQUEST 来标识 DLQ 上需要应答的那些消息。

PERSIST (持久性|*)

消息的持久性值。(消息的持久性决定了它是否在队列管理器重新启动后仍然存在。)

您可以使用符号名称。例如，可以使用符号名称 MQPER_PERSISTENT 来标识 DLQ 上持久的那些消息。

原因 (ReasonCode|*)

用于描述将消息放入 DLQ 的原因码。

您可以使用符号名称。例如，您可以使用符号名称 MQRC_Q_FULL 来标识放置在 DLQ 上的那些消息，因为它们的目标队列已满。

REPLYQ (QueueName|*)

在 DLQ 上消息的消息描述符 MQMD 中指定的应答队列名称。

REPLYQM (QueueManager 名称|*)

REPLYQ 关键字中指定的应答队列的队列管理器名称。

USERID (UserIdentifier|*)

在 DLQ 上生成消息的用户的用户标识，如消息描述符 MQMD 中所指定。

IBM i IBM i 上的 DLQ 操作关键字

使用这些死信队列操作关键字来确定如何处理 IBM i 死信队列上的匹配消息。

ACTION (DISCARD | IGNORE | RETRY | FWD)

对 DLQ 上与此规则中定义的模式匹配的任何消息执行的操作。

丢弃

导致从 DLQ 中删除消息。

IGNORE

使消息保留在 DLQ 上。

重试

使 DLQ 处理程序再次尝试将消息放在其目标队列上。

转发

使 DLQ 处理程序将消息转发到 FWDQ 关键字上指定的队列。

必须指定 ACTION 关键字。实现操作的尝试次数由 RETRY 关键字控制。控制数据的 RETRYINT 关键字控制尝试之间的时间间隔。

FWDQ (QueueName|&DESTQ|&REPLYQ)

选择 ACTION 关键字时要将消息转发到的消息队列的名称。

QueueName

消息队列的名称。FWDQ (") 无效。

&DESTQ

从 MQDLH 结构中的 DestQName 字段获取队列名称。

&REPLYQ

从消息描述符 MQMD 中的 ReplyToQ 字段获取队列名称。

You can specify REPLYQ (?*) in the message pattern to avoid error messages, when a rule specifying FWDQ (&REPLYQ) matches a message with a blank ReplyToQ field.

FWDQM (*QueueManager* 名称|&DESTQM|&REPLYQM|')

将消息转发到的队列的队列管理器。

队列管理器名称

选择 ACTION (FWD) 关键字时将消息转发到的队列的队列管理器名称。

&DESTQM

从 MQDLH 结构中的 *DestQMGrName* 字段获取队列管理器名称。

&REPLYQM

从消息描述符 MQMD 中的 *ReplyToQMGr* 字段获取队列管理器名称。

''

FWDQM ("") (缺省值) 标识本地队列管理器。

HEADER (YES|NO)

MQDLH 是否应保留在请求 ACTION (FWD) 的消息上。缺省情况下, MQDLH 保留在消息上。HEADER 关键字对于 FWD 以外的操作无效。

PUTAUT (DEF| CTX)

应由 DLQ 处理程序放置消息的权限:

DEF

使用 DLQ 处理程序本身的权限放置消息。

CTX

使消息在消息上下文中具有用户标识的权限。如果指定了 PUTAUT (CTX), 那么您必须有权采用其他用户的身份。

RETRY (*RetryCount*|1)

尝试操作 (在控制数据的 RETRYINT 关键字上指定的时间间隔) 的次数, 范围为 1-999,999,999。

注: DLQ 处理程序为实现任何特定规则而进行的尝试计数特定于 DLQ 处理程序的当前实例; 该计数不会在重新启动之间持久存在。如果重新启动 DLQ 处理程序, 那么应用规则的尝试计数将重置为零。

IBM i IBM i 上的 DLQ 规则表约定

IBM i 死信队列规则表必须遵循有关其语法, 结构和内容的特定约定。

- 规则表必须至少包含一个规则。
- 关键字可以按任意顺序出现。
- 关键字只能包含在任何规则中一次。
- 关键字不区分大小写。
- 关键字及其参数值必须与其他关键字至少用一个空格或逗号分隔。
- 可以在规则的开头或结尾以及关键字, 标点和值之间出现任意数目的空格。
- 每个规则都必须在新行上开始。
- 对于可移植性, 行的有效长度不得大于 72 个字符。
- 使用加号 (+) 作为行上的最后一个非空白字符, 以指示规则从下一行中的第一个非空白字符继续。使用减号 (-) 作为行上的最后一个非空白字符, 以指示规则从下一行的开头开始继续。可以在关键字和参数中出现连续字符。

例如:

```
APPLNAME('ABC+  
D')
```

结果为 "ABCD"。

```
APPLNAME('ABC-  
D')
```

结果为 " ABC D"。

- 以星号 (*) 开头的注释行可以出现在规则表中的任何位置。
- 空白行予以忽略。
- DLQ 处理程序规则表中的每个条目都包含一个或多个关键字及其关联参数。这些参数必须遵循以下语法规则:

- 每个参数值必须至少包含一个有效字符。括在引号内的值中的定界引号被视为不重要。例如, 以下参数有效:

| | |
|---------------|---------|
| FORMAT('ABC') | 3 个有效字符 |
| FORMAT(ABC) | 3 个有效字符 |
| FORMAT('A') | 1 有效字符 |
| FORMAT(A) | 1 有效字符 |
| FORMAT(' ') | 1 有效字符 |

这些参数无效, 因为它们不包含重要字符:

```
FORMAT(' ')  
FORMAT( )  
FORMAT()  
FORMAT
```

- 支持通配符。您可以使用问号 (?) 代替任何单个字符, 但尾部空格除外。可以使用星号 (*) 代替零个或多个相邻字符。星号 (*) 和问号 (?) **始终** 解释为参数值中的通配符。
- 不能在下列关键字的参数中包含通配符 :ACTION, HEADER, RETRY, FWDQ, FWDQM 和 PUTAUT。
- 在执行通配符匹配时, 参数值中的尾部空格以及 DLQ 上消息中的相应字段中的尾部空格并不重要。但是, 在引号中的字符串内的前导和嵌入空格对于通配符匹配很重要。
- 数字参数不能包含问号 (?) 通配符。可以包含星号 (*) 来代替整个数字参数, 但不能将星号包含为数字参数的一部分。例如, 以下是有效的数字参数:

| | |
|--------------|-------------|
| MSGTYPE(2) | 只有应答消息才符合条件 |
| MSGTYPE(*) | 任何消息类型都符合条件 |
| MSGTYPE('*') | 任何消息类型都符合条件 |

但是, MSGTYPE('2*') 无效, 因为它包含星号 (*) 作为数字参数的一部分。

- 数字参数必须在范围 0-999 999 999 之间。如果参数值在此范围内, 那么将接受该参数值, 即使该参数值在与关键字相关的字段中当前无效也是如此。可以将符号名称用于数字参数。
- 如果字符串值比与关键字相关的 MQDLH 或 MQMD 中的字段短, 那么该值将用空白填充到字段的长度。如果值 (不包括星号) 比字段长, 那么会诊断错误。例如, 以下是 8 字符字段的所有有效字符串值:

| | |
|---------------------|---------------|
| 'ABCDEFGH' | 8 个字符 |
| 'A*C*E*G*I' | 5 个字符 (不包括星号) |
| '*A*C*E*G*I*K*M*O*' | 8 个字符 (不包括星号) |

- 包含空格, 小写字母或除句点 (.), 正斜杠 (?), 下划线 (_) 和百分号 (%) 以外的特殊字符的字符串必须括在单引号中。未括在引号内的小写字母将转换为大写。如果字符串包含引号, 那么必须使用两个单引号来表示引号的开头和结尾。计算字符串的长度时, 每次出现的双引号都算作单个字符。

 在 IBM i 上处理 DLQ 规则表
死信队列处理程序在规则表中搜索具有与 IBM i 死信队列上的消息匹配的模式规则。

搜索以表中的第一个规则开始，并在表中按顺序继续。找到具有匹配模式的规则时，规则表会尝试该规则中的操作。每当 DLQ 处理程序尝试应用某个规则时，它会将该规则的重试计数递增 1。如果第一次尝试失败，那么将重复该尝试，直到尝试次数与 **RETRY** 关键字上指定的数字匹配为止。如果所有尝试都失败，那么 DLQ 处理程序将搜索表中的下一个匹配规则。

将针对后续匹配规则重复此过程，直到操作成功为止。当尝试了每个匹配规则时，在其 **RETRY** 关键字上指定的次数，并且所有尝试都失败了，那么将采用 **ACTION (IGNORE)**。如果找不到匹配的规则，那么也将采用 **ACTION (IGNORE)**。

注:

1. 仅针对以 **MQDLH** 开头的 DLQ 上的消息查找匹配的规则模式。不以 **MQDLH** 开头的消息会定期报告为出错，并无限期地保留在 DLQ 上。
2. 所有模式关键字都可以是缺省值，因此规则只能由操作组成。但是，请注意，仅操作规则将应用于队列上具有 **MQDLH** 且尚未根据表中的其他规则进行处理的所有消息。
3. 将在 DLQ 处理程序启动时验证规则表，并在该时间标记错误。(在消息和原因码中描述了 DLQ 处理程序发出的错误消息。)您可以随时对规则表进行更改，但这些更改直到 DLQ 处理程序重新启动后才会生效。
4. DLQ 处理程序不会改变消息，**MQDLH** 或消息描述符的内容。DLQ 处理程序始终使用消息选项 **MQPMO_PASS_ALL_CONTEXT** 将消息放入其他队列。
5. 可能无法识别规则表中的连续语法错误，因为规则表的验证将消除重复错误的生成。
6. DLQ 处理程序使用 **MQOO_INPUT_AS_Q_DEF** 选项打开 DLQ。
7. 可使用相同的规则表对同一队列并发运行 DLQ 处理程序的多个实例。但是，通常情况下，DLQ 与 DLQ 处理程序之间存在一对一关系。

IBM i 确保在 *IBM i* 上处理所有 DLQ 消息

死信队列处理程序保留 IBM i DLQ 上所有已看到但未除去的消息的记录。

如果使用 DLQ 处理程序作为过滤器从 DLQ 中抽取一小部分消息，那么 DLQ 处理程序仍会在 DLQ 上保留其未处理的那些消息的记录。此外，DLQ 处理程序无法保证将看到到达 DLQ 的新消息，即使将 DLQ 定义为先进先出 (FIFO) 也是如此。如果队列不为空，那么将定期重新扫描 DLQ 以检查所有消息。

出于这些原因，请尝试确保 DLQ 包含尽可能少的消息。如果允许无法丢弃或转发到其他队列的消息 (无论出于何种原因) 在队列上累积，那么 DLQ 处理程序的工作负载会增加，并且 DLQ 本身有填满的危险。

您可以采取特定措施，使 DLQ 处理程序能够清空 DLQ。例如，尝试不使用 **ACTION (IGNORE)**，这会将消息保留在 DLQ 上。(请记住，对于表中的其他规则未显式处理的消息，将采用 **ACTION (IGNORE)**。)相反，对于那些要忽略的消息，请使用将消息移至另一个队列的操作。例如：

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

同样，使表中的最终规则成为用于处理表中先前规则未处理的消息的 **catchall**。例如，表中的最终规则可能如下所示：

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

这会导致传递到表中最终规则的消息被转发到队列 **REALLY.DEAD.QUEUE**，在该队列中可以手动处理这些消息。如果您没有这样的规则，那么消息可能会无限期地保留在 DLQ 上。

IBM i *IBM i* 上的 DLQ 处理程序规则表示例

IBM i 上的死信队列处理程序规则表的示例代码。此示例规则表包含单个控制数据条目和多个规则。

```
*****
*   An example rules table for the STRMQDLQ command   *
*****
* Control data entry
* -----
* If no queue manager name is supplied as an explicit parameter to
```

```

* STRMQMDLQ, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to STRMQMDLQ,
* use the DLQ defined for the local queue manager.
*
inputqm(' ') inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.

* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation is always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never does things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it must be able
* to cope with the message being lost, so we can afford to
* discard the message.

PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)

* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We do not have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)

```

调用 DLQ 处理程序

使用 `runmqdlq` 命令调用死信队列处理程序。您可以通过两种方式来命名要处理的 DLQ 和要使用的队列管理器。

这两种方式如下：

- 作为命令提示符中 `runmqdlq` 的参数。例如：

```
runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER <qrulc.rul
```

- 在规则表中。例如：

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
```

这些示例适用于名为 `ABC1.DEAD.LETTER.QUEUE`，由队列管理器 `ABC1.QUEUE.MANAGER`。

如果未如所示指定 DLQ 或队列管理器，那么会将安装的缺省队列管理器与属于该队列管理器的 DLQ 一起使用。

`runmqdlq` 命令从 `stdin` 获取其输入；通过从规则表重定向 `stdin` 将规则表与 `runmqdlq` 相关联。

要运行 DLQ 处理程序，您必须有权访问 DLQ 本身以及将 DLQ 上的消息转发到的任何消息队列。要使 DLQ 处理程序将消息放在消息上下文中具有用户标识权限的队列上，您还必须有权采用其他用户的身份。

有关 `runmqdlq` 命令的更多信息，请参阅 [runmqdlq](#)。

相关信息

[死信队列](#)

[未送达消息故障诊断](#)

样本 DLQ 处理程序 `amqsdldq`

除了使用 `runmqdlq` 命令调用的死信队列处理程序外，IBM MQ 还为样本 DLQ 处理程序 `amqsdldq` 的源提供类似于 `runmqdlq` 提供的函数。

您可以定制 `amqsdldq` 以提供符合您需求的 DLQ 处理程序。例如，您可能决定需要一个可处理不带死信头的消息的 DLQ 处理程序。（缺省 DLQ 处理程序和样本 `amqsdldq` 都仅处理 DLQ 上以死信头 `MQDLH` 开头的那些消息。未以 `MQDLH` 开头的消息被标识为出错，并且无限期地保留在 DLQ 上。）

`MQ_INSTALLATION_PATH` 表示 IBM MQ 安装所在的高级目录。

在 IBM MQ for Windows 中，在以下目录中提供了 `amqsdldq` 的源：

```
MQ_INSTALLATION_PATH\tools\c\samples\dlq
```

并且在目录中提供了编译版本：

```
MQ_INSTALLATION_PATH\tools\c\samples\bin
```

在 IBM MQ for UNIX 和 Linux 系统中，在以下目录中提供了 `amqsdldq` 的源：

```
MQ_INSTALLATION_PATH/samp/dlq
```

并且在目录中提供了编译版本：

```
MQ_INSTALLATION_PATH/samp/bin
```

您还可以在客户机方式下编译 `amqsdldq`。有关更多信息，请参阅 [编写客户机过程应用程序](#)，[为 IBM MQ MQI clients 构建应用程序](#)以及 [在 IBM MQ MQI client 环境中运行应用程序](#)。

DLQ 处理程序规则表

死信队列处理程序规则表定义 DLQ 处理程序如何处理到达 DLQ 的消息。

规则表中有两种类型的条目：

- 表中的第一个条目 (可选) 包含 控制数据。
- 表中的所有其他条目都是供 DLQ 处理程序遵循的规则。每个规则都包含一个 模式 (一组消息特征)，该模式与消息匹配，以及当 DLQ 上的消息与指定模式匹配时要执行的操作。规则表中必须至少有一个规则。

规则表中的每个条目都包含一个或多个关键字。

相关信息

[死信队列](#)

[未送达消息故障诊断](#)

DLQ 控制数据

可以在死信队列处理程序规则表中的控制数据条目中包含关键字。

注:

- 垂直线 (|) 分隔替代项，只能指定其中一个替代项。
- 所有关键字都是可选的。

INPUTQ (*QueueName* | " ")

要处理的 DLQ 的名称:

1. 作为参数提供给 runmqdlq 命令的任何 INPUTQ 值都将覆盖规则表中的任何 INPUTQ 值。
2. 如果未将 INPUTQ 值指定为 runmqdlq 命令的参数，但 **确实** 在规则表中指定值，那么将使用规则表中的 INPUTQ 值。
3. 如果未指定 DLQ 或在规则表中指定 INPUTQ (")，那么将使用属于队列管理器的 DLQ 的名称以及作为 runmqdlq 命令的参数提供的名称。
4. 如果未将 INPUTQ 值指定为 runmqdlq 命令的参数或指定为规则表中的值，那么将使用属于规则表中 INPUTQM 关键字上指定的队列管理器的 DLQ。

INPUTQM (*QueueManager* 名称 | " ")

拥有 INPUTQ 关键字上指定的 DLQ 的队列管理器的名称:

1. 作为参数提供给 runmqdlq 命令的任何 INPUTQM 值都将覆盖规则表中的任何 INPUTQM 值。
2. 如果未将 INPUTQM 值指定为 runmqdlq 命令的参数，那么将使用规则表中的 INPUTQM 值。
3. 如果未指定队列管理器，或者您在规则表中指定 INPUTQM (")，那么将使用安装的缺省队列管理器。

RETRYINT (时间间隔 | 60)

这是一个时间间隔 (以秒为单位)，DLQ 处理程序应在此时间间隔内重新处理第一次尝试时无法处理的 DLQ 上的消息，并且已请求重复尝试。缺省情况下，重试时间间隔为 60 秒。

WAIT (YES | NO | *nnn*)

当 DLQ 处理程序检测到没有可处理的其他消息时，是否应等待更多消息到达 DLQ。

YES

DLQ 处理程序无限期等待。

否

当检测到 DLQ 为空或不包含可处理的消息时，DLQ 处理程序结束。

nnn

在 DLQ 处理程序检测到队列为空或不包含它可以处理的消息后，它将等待 *nnn* 秒以等待新工作到达，然后再结束。

对繁忙 DLQ 和 WAIT (NO) 或 WAIT (*nnn*) 指定 WAIT (YES) 对于活动级别较低的 DLQ。如果允许 DLQ 处理程序终止，请使用触发再次调用该处理程序。有关触发的更多信息，请参阅 [使用触发器启动 IBM MQ 应用程序](#)。

在规则表中包含控制数据的替代方法是提供 DLQ 及其队列管理器的名称作为 runmqdlq 命令的输入参数。如果在规则表中同时指定值并将其作为 runmqdlq 命令的输入，那么在 runmqdlq 命令上指定的值优先。

如果在规则表中包含控制数据条目，那么它必须是表中的 **第一个** 条目。

DLQ 规则 (模式和操作)

模式匹配关键字 (与死信队列上的消息匹配的关键字) 和操作关键字 (确定 DLQ 处理程序如何处理匹配消息的关键字) 的描述。还提供了示例规则。

模式匹配关键字

用于指定与 DLQ 上的消息匹配的值的模式匹配关键字如下所示。(所有模式匹配关键字都是可选的):

APPLIDAT (*ApplIdentity* 数据|*)

在 DLQ 上的消息的消息描述符 MQMD 中指定的 *ApplIdentity* 数据值。

APPLNAME (*PutAppl* 名称|*)

发出 MQPUT 或 MQPUT1 调用的应用程序的名称, 如 DLQ 上消息的消息描述符 MQMD 的 *PutApplName* 字段中所指定。

APPLTYPE (*PutAppl* 类型|*)

在 DLQ 上的消息的消息描述符 MQMD 中指定的 *PutAppl* 类型值。

DESTQ (*QueueName*|*)

以消息为目标的消息队列的名称。

DESTQM (*QueueManager* 名称|*)

要为其发送消息的消息队列的队列管理器的名称。

FEEDBACK (反馈|*)

当 *MsgType* 值为 MQFB_REPORT 时, 反馈 将描述报告的性质。

您可以使用符号名称。例如, 可以使用符号名称 MQFB_COA 来标识 DLQ 上需要确认其到达其目标队列的那些消息。

格式 (格式|*)

消息发送方用于描述消息数据格式的名称。

MSGTYPE (*MsgType*|*)

DLQ 上消息的消息类型。

您可以使用符号名称。例如, 可以使用符号名称 MQMT_REQUEST 来标识 DLQ 上需要应答的那些消息。

PERSIST (持久性|*)

消息的持久性值。(消息的持久性决定了它是否在队列管理器重新启动后仍然存在。)

您可以使用符号名称。例如, 可以使用符号名称 MQPER_PERSISTENT 来标识 DLQ 上持久的消息。

原因 (*ReasonCode*|*)

用于描述将消息放入 DLQ 的原因码。

您可以使用符号名称。例如, 您可以使用符号名称 MQRC_Q_FULL 来标识放置在 DLQ 上的那些消息, 因为它们的目标队列已满。

REPLYQ (*QueueName*|*)

在 DLQ 上消息的消息描述符 MQMD 中指定的应答队列的名称。

REPLYQM (*QueueManager* 名称|*)

应答队列的队列管理器的名称, 如 DLQ 上消息的消息描述符 MQMD 中所指定。

USERID (*UserIdentifier*|*)

在 DLQ 上生成消息的用户的用户标识, 如 DLQ 上消息的消息描述符 MQMD 中所指定。

操作关键字

用于描述如何处理匹配消息的操作关键字如下所示:

ACTION (DISCARD | IGNORE | RETRY | FWD)

要对 DLQ 上与此规则中定义的模式匹配的任何消息执行的操作。

丢弃

从 DLQ 中删除消息。

IGNORE

将消息保留在 DLQ 上。

重试

如果第一次尝试将消息放在其目标队列上失败，请重试。RETRY 关键字设置为实现操作而进行的尝试次数。控制数据的 RETRYINT 关键字控制尝试之间的时间间隔。

转发

将消息转发到 FWDQ 关键字上指定的队列。

必须指定 ACTION 关键字。

FWDQ (*QueueName* | &DESTQ | &REPLYQ)

请求 ACTION (FWD) 时要将消息转发到的消息队列的名称。

QueueName

消息队列的名称。FWDQ (") 无效。

&DESTQ

从 MQDLH 结构中的 *DestQName* 字段获取队列名称。

&REPLYQ

从消息描述符 MQMD 中的 *ReplyToQ* 字段获取队列名称。

To avoid error messages when a rule specifying FWDQ (&REPLYQ) matches a message with a blank *ReplyToQ* field, specify REPLYQ (?*) in the message pattern.

FWDQM (*QueueManager* 名称 | &DESTQM | &REPLYQM | ')

要将消息转发到的队列的队列管理器。

队列管理器名称

请求 ACTION (FWD) 时要将消息转发到的队列的队列管理器的名称。

&DESTQM

从 MQDLH 结构中的 *DestQMName* 字段获取队列管理器名称。

&REPLYQM

从消息描述符 MQMD 中的 *ReplyToQMName* 字段获取队列管理器名称。

..

FWDQM ("") (缺省值) 标识本地队列管理器。

HEADER (YES | NO)

MQDLH 是否应保留在请求 ACTION (FWD) 的消息上。缺省情况下，MQDLH 保留在消息上。HEADER 关键字对于 FWD 以外的操作无效。

PUTAUT (DEF | CTX)

应由 DLQ 处理程序放置消息的权限:

DEF

使用 DLQ 处理程序本身的权限放入消息。

CTX

将具有用户标识权限的消息放在消息上下文中。如果指定 PUTAUT (CTX)，那么您必须有权采用其他用户的身份。

RETRY (*RetryCount* | 1)

尝试操作 (在控制数据的 RETRYINT 关键字上指定的时间间隔) 的次数，范围为 1-999,999,999。DLQ 处理程序为实现任何特定规则而进行的尝试计数特定于 DLQ 处理程序的当前实例；该计数不会在重新启动之间持久存在。如果重新启动 DLQ 处理程序，那么应用规则的尝试计数将重置为零。

示例规则

以下是来自 DLQ 处理程序规则表的示例规则:

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +  
ACTION (RETRY) RETRY (3)
```

此规则指示 DLQ 处理程序进行三次尝试以将由于 MQPUT 和 MQPUT1 被禁止而放入 DLQ 上的任何持久消息传递到其目标队列。

在此部分的其余部分中描述了可用于规则的所有关键字。请注意下列事项：

- 关键字的缺省值 (如果有) 带有下划线。对于大多数关键字, 缺省值为 * (星号), 这与任何值匹配。
- 垂直线 (|) 分隔替代项, 只能指定其中一个替代项。
- 除 ACTION 以外的所有关键字都是可选的。

DLQ 规则表约定

死信队列处理程序规则表的语法, 结构和内容必须遵循这些约定。

规则表必须遵循以下约定:

- 规则表必须至少包含一个规则。
- 关键字可以按任意顺序出现。
- 关键字只能包含在任何规则中一次。
- 关键字不区分大小写。
- 关键字及其参数值必须与其他关键字至少用一个空格或逗号分隔。
- 在规则的开头或结尾以及关键字, 标点和值之间可以有任意数目的空格。
- 每个规则都必须在新行上开始。
- 在 Windows 系统上, 表中的最后一条规则必须以回车符/换行符结尾。您可以通过确保在规则末尾按 Enter 键来实现此目的, 以便表的最后一行是空白行。
- 出于可移植性的原因, 行的显着长度不得大于 72 个字符。
- 使用加号 (+) 作为行上的最后一个非空白字符, 以指示规则从下一行中的第一个非空白字符继续。使用减号 (-) 作为行上的最后一个非空白字符, 以指示规则从下一行的开头开始继续。可以在关键字和参数中出现连续字符。

例如:

```
APPLNAME('ABC+  
D')
```

生成 "ABCD", 并且

```
APPLNAME('ABC-  
D')
```

结果为 " ABC D"。

- 以星号 (*) 开头的注释行可以出现在规则表中的任何位置。
- 空白行予以忽略。
- DLQ 处理程序规则表中的每个条目都包含一个或多个关键字及其关联参数。这些参数必须遵循以下语法规则:
 - 每个参数值必须至少包含一个有效字符。用引号括起的值中的定界单引号被认为不重要。例如, 以下参数有效:

| | |
|---------------|---------|
| FORMAT('ABC') | 3 个有效字符 |
| FORMAT(ABC) | 3 个有效字符 |
| FORMAT('A') | 1 有效字符 |
| FORMAT(A) | 1 有效字符 |
| FORMAT(' ') | 1 有效字符 |

这些参数无效, 因为它们不包含重要字符:

FORMAT('')

FORMAT()

FORMAT()

FORMAT

- 支持通配符。可以使用问号 (?) 代替任何单个字符，但尾部空格除外；可以使用星号 (*) 代替零个或多个相邻字符。星号 (*) 和问号 (?) **始终** 解释为参数值中的通配符。
- 通配符不能包含在下列关键字的参数中: ACTION, HEADER, RETRY, FWDQ, FWDQM 和 PUTAUT。
- 在执行通配符匹配时，参数值中的尾部空格以及 DLQ 上消息中的相应字段中的尾部空格并不重要。但是，以单引号括起的字符串中的前导空格和嵌入空格对于通配符匹配很重要。
- 数字参数不能包含问号 (?) 通配符。可以使用星号 (*) 代替整个数字参数，但不能将其作为数字参数的一部分。例如，以下是有效的数字参数:

| | |
|--------------|-------------|
| MSGTYPE(2) | 只有应答消息才符合条件 |
| MSGTYPE(*) | 任何消息类型都符合条件 |
| MSGTYPE('*') | 任何消息类型都符合条件 |

但是，MSGTYPE('2*') 无效，因为它包含星号 (*) 作为数字参数的一部分。

- 数字参数必须在范围 0-999 999 999 之间。如果参数值在此范围内，那么将接受该参数值，即使该参数值在与关键字相关的字段中当前无效也是如此。可以将符号名称用于数字参数。
- 如果字符串值比与关键字相关的 MQDLH 或 MQMD 中的字段短，那么该值将用空白填充到字段的长度。如果值 (不包括星号) 比字段长，那么会诊断错误。例如，以下是 8 字符字段的所有有效字符串值:

| | |
|---------------------|---------------|
| 'ABCDEFGH' | 8 个字符 |
| 'A*C*E*G*I' | 5 个字符 (不包括星号) |
| '*A*C*E*G*I*K*M*O*' | 8 个字符 (不包括星号) |

- 将包含空格，小写字符或除句点 (.)，正斜杠 (?), 下划线 (_) 和百分号 (%) 以外的特殊字符的字符串括在单引号中。未括在单引号内的小写字符将折叠为大写。如果字符串包含引号，请使用两个单引号来表示引号的开头和结尾。计算字符串的长度时，每次出现的双引号都算作单个字符。

如何处理 DLQ 规则表

死信队列处理程序在规则表中搜索模式与 DLQ 上的消息匹配的规则。

搜索以表中的第一个规则开始，并在表中按顺序继续。当 DLQ 处理程序找到具有匹配模式的规则时，它将从该规则执行操作。无论何时应用规则，DLQ 处理程序都会将规则的重试计数递增 1。如果第一次尝试失败，那么 DLQ 处理程序将再次尝试，直到尝试次数与 RETRY 关键字上指定的次数匹配为止。如果所有尝试都失败，那么 DLQ 处理程序将搜索表中的下一个匹配规则。

将针对后续匹配规则重复此过程，直到操作成功为止。当尝试了每个匹配规则时，在其 RETRY 关键字上指定的次数，并且所有尝试都失败了，那么将采用 ACTION (IGNORE)。如果找不到匹配的规则，那么也将采用 ACTION (IGNORE)。

注:

1. 仅针对以 MQDLH 开头的 DLQ 上的消息查找匹配的规则模式。不以 MQDLH 开头的消息会定期报告为出错，并无限期地保留在 DLQ 上。
2. 可以允许所有模式关键字为缺省值，以便规则只能由操作组成。但是，请注意，仅操作规则将应用于队列上具有 MQDLH 且尚未根据表中的其他规则进行处理的所有消息。
3. 当 DLQ 处理程序启动时，将验证规则表，并在该时间标记错误。您可以随时对规则表进行更改，但这些更改直到 DLQ 处理程序重新启动后才会生效。

- DLQ 处理程序不会更改消息，MQDLH 或消息描述符的内容。DLQ 处理程序始终使用消息选项 MQPMO_PASS_ALL_CONTEXT 将消息放入其他队列。
- 可能无法识别规则表中的连续语法错误，因为规则表旨在避免在验证期间生成重复错误。
- DLQ 处理程序使用 MQOO_INPUT_AS_Q_DEF 选项打开 DLQ。
- 可使用相同的规则表对同一队列并发运行 DLQ 处理程序的多个实例。但是，通常情况下，DLQ 与 DLQ 处理程序之间存在一对一关系。

相关信息

[死信队列](#)

[未送达消息故障诊断](#)

确保处理所有 DLQ 消息

死信队列处理程序保留 DLQ 上所有已看到但未除去的消息的记录。

如果使用 DLQ 处理程序作为过滤器从 DLQ 中抽取一小部分消息，那么 DLQ 处理程序仍必须在它未处理的 DLQ 上保留这些消息的记录。此外，DLQ 处理程序无法保证会看到到达 DLQ 的新消息，即使将 DLQ 定义为先进先出 (FIFO) 也是如此。如果队列不为空，那么将定期重新扫描 DLQ 以检查所有消息。

出于这些原因，请尝试确保 DLQ 包含尽可能少的消息；如果允许无法丢弃或转发到其他队列的消息 (无论出于何种原因) 在队列上累积，那么 DLQ 处理程序的工作负载会增加，并且 DLQ 本身会填满。

您可以采取特定措施，使 DLQ 处理程序能够清空 DLQ。例如，尝试不使用 ACTION (IGNORE)，这会在 DLQ 上留下消息。(请记住，对于表中其他规则未显式处理的消息，将采用 ACTION (IGNORE)。) 相反，对于那些要忽略的消息，请使用将消息移至另一个队列的操作。例如：

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

同样，使表中的最终规则成为一个 catchall，以处理表中先前规则未处理的消息。例如，表中的最终规则可能如下所示：

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

这会将落到表中最终规则的消息转发到队列 REALLY.DEAD.QUEUE，在该队列中可以手动处理这些消息。如果您没有这样的规则，那么消息可能会无限期地保留在 DLQ 上。

一个示例 DLQ 处理程序规则表

runmqdlq 命令的示例死信队列规则表，其中包含单个控制数据条目和多个规则。

```
*****
*   An example rules table for the runmqdlq command   *
*****
* Control data entry
* -----
* If no queue manager name is supplied as an explicit parameter to
* runmqdlq, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to runmqdlq,
* use the DLQ defined for the local queue manager.
*
inputqm(' ') inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.
* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).
```

```

REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation are always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never do things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

```

```

* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it should be able
* to cope with the message being lost, so we can afford to
* discard the message. PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)
* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We don't have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)

```

相关信息

[死信队列](#)

[未送达消息故障诊断](#)

[runmqdlq \(运行死信队列处理程序\)](#)

使用模型队列

如果队列管理器从指定已定义为模型队列的队列名称的应用程序接收 MQI 调用，那么它将创建 动态队列。新动态队列的名称由队列管理器在创建队列时生成。模型队列 是一个模板，用于指定从中创建的任何动态队列的属性。模型队列为应用程序提供了一种方便的方法来根据需要创建队列。

定义模型队列

使用 **DEFINE QMODEL** 命令以与定义本地队列相同的方式定义具有一组属性的模型队列。模型队列和本地队列具有相同的属性集，但在模型队列上，您可以指定创建的动态队列是临时的还是永久的。(在队列管理器重新启动之间保留永久队列，但不保留临时队列。) 例如：

```
DEFINE QMODEL (GREEN.MODEL.QUEUE) +
DESCR('Queue for messages from application X') +
PUT (DISABLED) +
GET (ENABLED) +
NOTRIGGER +
MSGDLVSQ (FIFO) +
MAXDEPTH (1000) +
MAXMSGL (2000) +
USAGE (NORMAL) +
DEFTYPE (PERMDYN)
```

此命令创建模型队列定义。从 **DEFTYPE** 属性中，您可以看到从此模板创建的实际队列是永久动态队列。将自动从 `SYSYSTEM.DEFAULT.MODEL.QUEUE` 缺省队列。

定义模型队列时，可以使用 **LIKE** 和 **REPLACE** 属性，方式与将它们用于本地队列的方式相同。

将其他命令与模型队列配合使用

您可以使用相应的 MQSC 命令来显示或更改模型队列的属性，或者删除模型队列对象。例如：

使用 **DISPLAY QUEUE** 命令可显示模型队列的属性：

```
DISPLAY QUEUE (GREEN.MODEL.QUEUE)
```

使用 **ALTER QMODEL** 命令可更改模型，以启用从此模型创建的任何动态队列上的放置：

```
ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)
```

使用 **DELETE QMODEL** 命令来删除此模型队列：

```
DELETE QMODEL (RED.MODEL.QUEUE)
```

相关信息

[变更 QMODEL](#)

[定义 QModel](#)

[删除 QMODEL](#)

[DISPLAY QUEUE](#)

处理管理主题

使用 MQSC 命令来管理管理主题。

有关这些命令的详细信息，请参阅 [MQSC 命令](#)。

相关概念

[第 151 页的『定义管理主题』](#)

使用 MQSC 命令 **DEFINE TOPIC** 来创建管理主题。定义管理主题时，可以选择设置每个主题属性。

[第 151 页的『显示管理主题对象属性』](#)

使用 MQSC 命令 **DISPLAY TOPIC** 显示管理主题对象。

[第 152 页的『更改管理主题属性』](#)

您可以通过两种方式更改主题属性，即使用 **ALTER TOPIC** 命令或带有 **REPLACE** 属性的 **DEFINE TOPIC** 命令。

[第 152 页的『复制管理主题定义』](#)

您可以在 **DEFINE** 命令上使用 LIKE 属性来复制主题定义。

第 153 页的『删除管理主题定义』

您可以使用 MQSC 命令 **DELETE TOPIC** 来删除管理主题。

相关信息

[管理主题对象](#)

定义管理主题

使用 MQSC 命令 **DEFINE TOPIC** 来创建管理主题。定义管理主题时，可以选择设置每个主题属性。

未显式设置的主题的任何属性都将从缺省管理主题 SYSTEM.DEFAULT.TOPIC，在安装系统安装时创建。

例如，下面的 **DEFINE TOPIC** 命令定义了具有以下特征的名为 ORANGE.TOPIC 的主题：

- 解析为主题字符串 ORANGE。有关如何使用主题字符串的信息，请参阅 [组合主题字符串](#)。
- 设置为 ASPARENT 的任何属性都将使用本主题的父主题所定义的属性。此操作在主题树上重复到根主题 SYSTEM.BASE.TOPIC。有关更多信息，请参阅 [主题树](#)。

```
DEFINE TOPIC (ORANGE.TOPIC) +
TOPICSTR (ORANGE) +
DEFPRTY(ASPARENT) +
NPMSGDLV(ASPARENT)
```

注：

- 除主题字符串的值外，显示的所有属性值都是缺省值。此处仅显示为示例。如果您确定缺省值是您想要的值或未更改的值，那么可以省略这些值。另请参阅第 151 页的『显示管理主题对象属性』。
- 如果已在同一队列管理器上具有名为 ORANGE.TOPIC，此命令失败。如果要覆盖主题的现有定义，请使用 REPLACE 属性，但另请参阅第 152 页的『更改管理主题属性』。

相关信息

[DEFINE TOPIC](#)

显示管理主题对象属性

使用 MQSC 命令 **DISPLAY TOPIC** 显示管理主题对象。

要显示所有主题，请使用：

```
DISPLAY TOPIC (ORANGE.TOPIC)
```

您可以通过使用 **DISPLAY TOPIC** 命令单独指定属性来选择性地显示这些属性。例如：

```
DISPLAY TOPIC (ORANGE.TOPIC) +
TOPICSTR +
DEFPRTY +
NPMSGDLV
```

此命令显示三个指定的属性，如下所示：

```
AMQ8633: Display topic details.
TOPIC (ORANGE.TOPIC)
TOPICSTR (ORANGE)
NPMSGDLV (ASPARENT)
TYPE (LOCAL)
DEFPRTY (ASPARENT)
```

要在运行时使用主题 ASPARENT 值时显示这些值，请使用 **DISPLAY TPSTATUS** 命令。例如，使用：

```
DISPLAY TPSTATUS (ORANGE) DEFPRTY NPMSGDLV
```

该命令显示以下详细信息：

```
AMQ8754: Display topic status details.  
TOPICSTR(ORANGE)          DEFPRTY(0)  
NPMSGDLV(ALLAVAIL)
```

定义管理主题时，它将采用您未从缺省管理主题 (称为 SYSTEM.DEFAULT.TOPIC。要查看这些缺省属性是什么，请使用以下命令：

```
DISPLAY TOPIC (SYSTEM.DEFAULT.TOPIC)
```

相关信息

[DISPLAY TOPIC](#)

[DISPLAY TPSTATUS](#)

更改管理主题属性

您可以通过两种方式更改主题属性，即使用 **ALTER TOPIC** 命令或带有 **REPLACE** 属性的 **DEFINE TOPIC** 命令。

例如，如果要更改传递到名为 ORANGE.TOPIC 为 5，请使用以下任一命令。

- 使用 **ALTER** 命令：

```
ALTER TOPIC(ORANGE.TOPIC) DEFPRTY(5)
```

此命令将传递到此主题的消息的缺省优先级的单个属性更改为 5；所有其他属性保持不变。

- 使用 **DEFINE** 命令：

```
DEFINE TOPIC(ORANGE.TOPIC) DEFPRTY(5) REPLACE
```

此命令将更改传递到此主题的消息的缺省优先级。 将为所有其他属性提供其缺省值。

如果更改发送到此主题的消息的优先级，那么现有消息不受影响。但是，如果发布应用程序未提供任何新消息，那么将使用指定的优先级。

相关信息

[ALTER TOPIC](#)

[DISPLAY TOPIC](#)

复制管理主题定义

您可以在 **DEFINE** 命令上使用 **LIKE** 属性来复制主题定义。

例如：

```
DEFINE TOPIC (MAGENTA.TOPIC) +  
LIKE (ORANGE.TOPIC)
```

此命令创建主题 MAGENTA.TOPIC，具有与原始主题 ORANGE.TOPIC，而不是系统缺省管理主题的主题。输入要复制的主题的名称，与创建主题时输入的名称完全相同。如果名称包含小写字母，请将名称括在单引号中。

您还可以使用此格式的 **DEFINE** 命令来复制主题定义，但对原始属性进行更改。例如：

```
DEFINE TOPIC(BLUE.TOPIC) +  
TOPICSTR(BLUE) +  
LIKE(ORANGE.TOPIC)
```

您还可以将主题 BLUE.TOPIC 的属性复制到主题 GREEN.TOPIC，并指定当发布无法传递到其正确的订户队列时，不会将其放入死信队列中。例如：


```
DEFINE TOPIC(GREEN.TOPIC) +
TOPICSTR(GREEN) +
LIKE(BLUE.TOPIC) +
USEDLQ(NO)
```

相关信息

[DEFINE TOPIC](#)

删除管理主题定义

您可以使用 MQSC 命令 **DELETE TOPIC** 来删除管理主题。

例如：

```
DELETE TOPIC(ORANGE.TOPIC)
```

应用程序将无法再打开主题以进行发布或使用对象名 **ORANGE.TOPIC**。发布打开了主题的应用程序能够继续发布已解析的主题字符串。已对此主题进行的任何预订都将在删除此主题后继续接收发布。

未引用此主题对象但正在使用此主题对象所表示的已解析主题字符串 (在此示例中为 "ORANGE") 的应用程序将继续工作。在这种情况下，它们将从主题树中更高的主题对象继承属性。有关更多信息，请参阅 [主题树](#)。

相关信息

[删除主题](#)

使用预订

使用 MQSC 命令来管理预订。

预订可以是 **SUBTYPE** 属性中定义的三种类型之一：

管理

由用户以管理方式定义。

PROXY

用于在队列管理器之间路由发布的内部创建的预订。

API

以编程方式创建，例如，使用 MQI MQSUB 调用。

有关这些命令的详细信息，请参阅 [MQSC 命令](#)。

相关概念

[第 153 页的『定义管理预订』](#)

使用 MQSC 命令 **DEFINE SUB** 来创建管理预订。您还可以使用缺省本地预订定义中定义的缺省值。或者，您可以从缺省本地预订 **SYSTEM.DEFAULT.SUB**。

[第 154 页的『显示预订的属性』](#)

您可以使用 **DISPLAY SUB** 命令来显示队列管理器已知的任何预订的已配置属性。

[第 155 页的『更改本地预订属性』](#)

可以通过两种方式更改预订属性，使用 **ALTER SUB** 命令或带有 **REPLACE** 属性的 **DEFINE SUB** 命令。

[第 155 页的『复制本地预订定义』](#)

您可以使用 **DEFINE** 命令上的 **LIKE** 属性来复制预订定义。

[第 156 页的『删除本地预订』](#)

可以使用 MQSC 命令 **DELETE SUB** 来删除本地预订。

定义管理预订

使用 MQSC 命令 **DEFINE SUB** 来创建管理预订。您还可以使用缺省本地预订定义中定义的缺省值。或者，您可以从缺省本地预订 **SYSTEM.DEFAULT.SUB**。

例如，以下 **DEFINE SUB** 命令定义了具有以下特征的名为 ORANGE 的预订：

- 持久预订，表示它在队列管理器重新启动后持续存在，并且到期时间不限。
- 接收对 ORANGE 主题字符串进行的发布，消息优先级由发布应用程序设置。
- 为此预订交付的发布将发送到本地队列 SUBQ，必须在定义预订之前定义此队列。

```
DEFINE SUB (ORANGE) +
TOPICSTR (ORANGE) +
DESTCLAS (PROVIDED) +
DEST (SUBQ) +
EXPIRY (UNLIMITED) +
PUBPRTY (AS PUB)
```

注：

- 预订和主题字符串名称不必匹配。
- 除了目标和主题字符串的值以外，显示的所有属性值都是缺省值。此处仅显示为示例。如果您确定缺省值是您想要的值或未更改的值，那么可以省略这些值。另请参阅第 154 页的『显示预订的属性』。
- 如果已在同一队列管理器上具有名为 ORANGE 的本地预订，那么此命令将失败。如果要覆盖队列的现有定义，请使用 **REPLACE** 属性，但另请参阅第 155 页的『更改本地预订属性』。
- 如果队列 SUBQ 不存在，那么此命令将失败。

相关信息

[DEFINE SUB](#)

显示预订的属性

您可以使用 **DISPLAY SUB** 命令来显示队列管理器已知的任何预订的已配置属性。

例如，使用：

```
DISPLAY SUB(ORANGE)
```

您可以通过单独指定属性来选择性地显示这些属性。例如：

```
DISPLAY SUB(ORANGE) +
SUBID +
TOPICSTR +
DURABLE
```

此命令显示三个指定的属性，如下所示：

```
AMQ8096: IBM MQ subscription inquired.
SUBID(414D51204141412020202020202020EE921E4E20002A03)
SUB(ORANGE)
DURABLE(YES)
TOPICSTR(ORANGE)
```

TOPICSTR 是此订户正在运行的已解析主题字符串。当定义预订以使用主题对象时，来自该对象的主题字符串将用作进行预订时提供的主题字符串的前缀。SUBID 是创建预订时由队列管理器指定的唯一标识。这是要显示的有用属性，因为某些预订名称可能是长的，或者位于可能变得不切实际的其他字符集中。

显示预订的备用方法是使用 SUBID：

```
DISPLAY SUB +
SUBID(414D51204141412020202020202020EE921E4E20002A03) +
TOPICSTR +
DURABLE
```

此命令提供与之前相同的输出：

```
AMQ8096: IBM MQ subscription inquired.
```

```
SUBID(414D5120414141202020202020202020EE921E4E20002A03)
SUB(ORANGE)
DURABLE(YES)
```

```
TOPICSTR(ORANGE)
```

缺省情况下，不会显示队列管理器上的代理预订。要显示它们，请指定 **SUBTYPE PROXY** 或 **ALL**。
您可以使用 `DISPLAY SBSTATUS` 命令来显示 "运行时" 属性。例如，使用以下命令：

```
DISPLAY SBSTATUS(ORANGE) NUMMSGs
```

将显示以下输出：

```
AMQ8099: IBM MQ subscription status inquired.
SUB(ORANGE)
SUBID(414D5120414141202020202020202020EE921E4E20002A03)
NUMMSGs(0)
```

定义管理预订时，它将采用您未从缺省预订 (称为 `SYSTEM.DEFAULT.SUB`)。要查看这些缺省属性是什么，请使用以下命令：

```
DISPLAY SUB (SYSTEM.DEFAULT.SUB)
```

相关信息

[显示子项](#)

更改本地预订属性

可以通过两种方式更改预订属性，使用 **ALTER SUB** 命令或带有 **REPLACE** 属性的 **DEFINE SUB** 命令。
例如，如果要将传递到名为 `ORANGE` 的预订的消息的优先级更改为 5，请使用以下任一命令：

- 使用 **ALTER** 命令：

```
ALTER SUB(ORANGE) PUBPRTY(5)
```

此命令将传递到此预订的消息的优先级的单个属性更改为 5；所有其他属性保持不变。

- 使用 **DEFINE** 命令：

```
DEFINE SUB(ORANGE) PUBPRTY(5) REPLACE
```

此命令不仅会更改传递到此预订的消息的优先级，还会更改为其缺省值提供的所有其他属性。

如果更改发送到此预订的消息的优先级，那么现有消息不受影响。但是，任何新消息都具有指定的优先级。

相关信息

[变更 SUB](#)

[DEFINE SUB](#)

复制本地预订定义

您可以使用 **DEFINE** 命令上的 **LIKE** 属性来复制预订定义。

例如：

```
DEFINE SUB(BLUE) +
  LIKE(ORANGE)
```

您还可以将子 `REAL` 的属性复制到子 `THIRD.SUB`，并指定已交付的发布的 `correlID` 为 `THIRD`，而不是发布程序 `correlID`。例如：

```
DEFINE SUB(THIRD.SUB) +
  LIKE(BLUE) +
  DESTCORL(ORANGE)
```

相关信息

[DEFINE SUB](#)

删除本地预订

可以使用 MQSC 命令 **DELETE SUB** 来删除本地预订。

```
DELETE SUB(ORANGE)
```

您还可以使用 SUBID 删除预订:

```
DELETE SUB SUBID(414D51204141412020202020202020EE921E4E20002A03)
```

相关信息

[删除 SUB](#)

检查预订上的消息

定义预订时，它与队列相关联。与此预订匹配的已发布消息将放入此队列。

关于此任务

请注意，以下 **runmqsc** 命令仅显示接收消息的那些预订。

要检查当前排队等待预订的消息，请执行以下步骤:

过程

1. 要检查排队等待预订类型 **DISPLAY SBSTATUS(sub_name) NUMMSGS** 的消息，请参阅 [第 154 页的『显示预订的属性』](#)。
2. 如果 **NUMMSGS** 值大于零，请通过输入 **DISPLAY SUB(sub_name) DEST** 来标识与预订关联的队列。
3. 通过使用返回的队列名称，您可以遵循 [第 129 页的『浏览队列』](#) 中描述的方法来查看消息。

相关信息

[显示 SBSTATUS](#)

使用服务

服务对象是将其他进程作为队列管理器的一部分进行管理的一种方法。通过服务，您可以定义在队列管理器启动和结束时启动和停止的程序。IBM MQ 服务始终以启动队列管理器的用户的用户标识启动。

要定义新的 IBM MQ 服务定义，请使用 MQSC 命令 **DEFINE SERVICE**。

服务对象可以是下列其中一种类型:

服务器

服务器是将参数 **SERVTYPE** 指定为 **SERVER** 的服务对象。服务器服务对象是在启动指定队列管理器时执行的程序的定义。服务器服务对象定义通常运行很长时间的程序。例如，可以使用服务器服务对象来执行触发器监视器进程，例如 **runmqtrm**。

只能同时运行服务器服务对象的一个实例。可以使用 MQSC 命令 **DISPLAY SVSTATUS** 来监视正在运行的服务器服务对象的状态。

命令

命令是将参数 **SERVTYPE** 指定为 **COMMAND** 的服务对象。命令服务对象类似于服务器服务对象，但是命令服务对象的多个实例可以并发运行，并且无法使用 MQSC 命令 **DISPLAY SVSTATUS** 来监视其状态。

如果执行 MQSC 命令 **STOP SERVICE**，那么在停止程序之前不会进行检查以确定 MQSC 命令 **START SERVICE** 启动的程序是否仍处于活动状态。

相关信息

[显示 SVSTATUS](#)

[启动服务](#)

[停止服务](#)

定义服务对象

使用 MQSC 命令 **DEFINE SERVICE** 定义服务对象。

您需要定义的属性如下所示：

SERVTYPE

定义服务对象的类型。可能的值如下所示：

服务器

服务器服务对象。

一次只能执行服务器服务对象的一个实例。可以使用 MQSC 命令 **DISPLAY SVSTATUS** 来监视服务器服务对象的状态。

COMMAND

命令服务对象。

可以同时执行命令服务对象的多个实例。无法监视命令服务对象的状态。

STARTCMD

为启动服务而执行的程序。必须指定程序的标准路径。

STARTARG

传递到启动程序的自变量。

STDERR

指定应该将服务程序的标准错误 (stderr) 重定向到的文件的路径。

STDOUT

指定应该将服务程序的标准输出 (stdout) 重定向到的文件的路径。

STOPCMD

为停止服务而执行的程序。必须指定程序的标准路径。

STOPARG

传递到停止程序的参数。

控制

指定如何启动和停止服务：

手动

服务不会自动启动或自动停止。它通过使用 **START SERVICE** 和 **STOP SERVICE** 命令进行控制。这是缺省值。

QMGR

要定义的服务将在启动和停止队列管理器的同时启动和停止。

STARTONLY

该服务将在队列管理器启动的同时启动，但不会在队列管理器停止时被请求停止。

相关概念

[第 158 页的『管理服务』](#)

通过使用 **CONTROL** 参数，服务对象的实例可以由队列管理器自动启动和停止，也可以使用 MQSC 命令 **START SERVICE** 和 **STOP SERVICE** 启动和停止。

相关信息

[定义服务](#)

[显示 SVSTATUS](#)

[启动服务](#)

停止服务

管理服务

通过使用 **CONTROL** 参数，服务对象的实例可以由队列管理器自动启动和停止，也可以使用 MQSC 命令 **START SERVICE** 和 **STOP SERVICE** 启动和停止。

启动服务对象的实例时，将向队列管理器错误日志写入一条消息，其中包含服务对象的名称和已启动进程的进程标识。以下是服务器服务对象启动的示例日志条目：

```
02/15/2005 11:54:24 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5028: The Server 'S1' has started. ProcessId(13031).

EXPLANATION:
The Server process has started.
ACTION:
None.
```

以下是命令服务对象启动的示例日志条目：

```
02/15/2005 11:53:55 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5030: The Command 'C1' has started. ProcessId(13030).

EXPLANATION:
The Command has started.
ACTION:
None.
```

当实例服务器服务停止时，将向队列管理器错误日志写入一条消息，其中包含服务的名称和结束进程的进程标识。以下是服务器服务对象停止的示例日志条目：

```
02/15/2005 11:54:54 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr(A.B.C)
AMQ5029: The Server 'S1' has ended. ProcessId(13031).

EXPLANATION:
The Server process has ended.
ACTION:
None.
```

相关参考

第 158 页的『其他环境变量』

启动某项服务时，服务进程的启动环境将继承自队列管理器的环境。可以通过将要定义的变量添加到其中一个 `service.env` 环境覆盖文件，来定义要在服务进程的环境中设置的其他环境变量。

相关信息

[停止服务](#)

[启动服务](#)

其他环境变量




启动某项服务时，服务进程的启动环境将继承自队列管理器的环境。可以通过将要定义的变量添加到其中一个 `service.env` 环境覆盖文件，来定义要在服务进程的环境中设置的其他环境变量。

可以向其添加环境变量的文件

有两个可能的文件可供您添加环境变量：




机器作用域 `service.env` 文件

此文件位于：

-   UNIX and Linux 系统上的 /var/mqm。
-  在 Windows 系统上安装期间选择的数据目录。

队列管理器作用域 **service.env** 文件

此文件位于队列管理器数据目录中。例如，名为 QMNAME 的队列管理器的环境覆盖文件的位置为：

-   在 UNIX and Linux 系统上， /var/mqm/qmgrs/QMNAME/service.env
-  在 Windows 系统上， C:\ProgramData\IBM\MQ\qmgrs\QMNAME\service.env

将处理这两个文件 (如果可用)，队列管理器作用域文件中的定义优先于机器作用域文件中的定义。

可以在 **service.env** 中指定的环境变量。

可以在 **service.env** 中指定任何环境变量。例如，如果 IBM MQ 服务运行大量命令，那么在 **service.env** 文件中设置 **PATH** 用户变量可能很有用。将变量设置为的值不能是环境变量；例如，**CLASSPATH= %CLASSPATH%** 不正确。同样，在 Linux **PATH= \$PATH : /opt/mqm/bin** 上会给出意外的结果。

CLASSPATH 必须大写，并且类路径语句只能包含文字。某些服务 (例如，Telemetry) 设置自己的类路径。将 **service.env** 中所定义的 **CLASSPATH** 添加至此类路径。

文件 **service.env** 中定义的变量的格式是名称/值变量对的列表。必须在新行上定义每个变量，并且将每个变量作为显式定义的变量 (包括空格)。

service.env 示例

```

#*****#
#*                                           *#
#* <N_OCO_COPYRIGHT>                         *#
#* Licensed Materials - Property of IBM      *#
#*                                           *#
#* 63H9336                                    *#
#* (C) Copyright IBM Corporation 2005, 2023. *#
#*                                           *#
#* <NOC_COPYRIGHT>                          *#
#*                                           *#
#*****#
#* Module Name: service.env                  *#
#* Type       : IBM MQ service environment file *#
#* Function   : Define additional environment variables to be set *#
#*           : for SERVICE programs.         *#
#* Usage     : <VARIABLE>=<VALUE>          *#
#*                                           *#
#*****#
MYLOC=/opt/myloc/bin
MYTMP=/tmp
TRACEDIR=/tmp/trace
MYINITQ=ACCOUNTS.INITIATION.QUEUE

```

相关参考

第 159 页的『服务定义上的可替换插入』

在服务对象的定义中，可以替换令牌。当执行服务程序时，被替换的标记将自动替换为其展开的文本。可以从以下公共令牌列表表中获取替代令牌，也可以从文件 **service.env** 中定义的任何变量中获取替代令牌。

服务定义上的可替换插入

在服务对象的定义中，可以替换令牌。当执行服务程序时，被替换的标记将自动替换为其展开的文本。可以从以下公共令牌列表表中获取替代令牌，也可以从文件 **service.env** 中定义的任何变量中获取替代令牌。



以下是可用于替换服务对象定义中的令牌的公共令牌：

MQ_INSTALL_PATH

IBM MQ 的安装位置。

MQ_DATA_PATH

IBM MQ 数据目录的位置:

-  在 UNIX and Linux 系统上, IBM MQ 数据目录位置为 /var/mqm/
-  在 Windows 系统上, IBM MQ 数据目录的位置是在安装 IBM MQ 期间选择的数据目录

QMNAME

当前队列管理器名称。

MQ_SERVICE_NAME

服务名称。

MQ_SERVER_PID

此令牌只能由 **STOPARG** 和 **STOPCMD** 参数使用。

对于服务器服务对象, 此令牌将替换为由 **STARTCMD** 和 **STARTARG** 参数启动的进程的进程标识。否则, 此令牌将替换为 0。

MQ_Q_MGR_DATA_PATH

队列管理器数据目录的位置。

MQ_Q_MGR_DATA_NAME

队列管理器的变换名称。有关名称变换的更多信息, 请参阅 [了解 IBM MQ 文件名](#)。

要使用可替换插入, 请将 + 字符内的标记插入到任何 **STARTCMD**, **STARTARG**, **STOPCMD**, **STOPARG**, **STDOUT** 或 **STDERR** 字符串中。有关此操作的示例, 请参阅 [第 160 页的『使用服务对象的示例』](#)。

使用服务对象的示例

此部分中的服务使用 UNIX 样式路径分隔符进行编写, 除非另有声明。

使用服务器服务对象

此示例显示如何定义, 使用和变更服务器服务对象以启动触发器监视器。

1. 使用 **DEFINE SERVICE** MQSC 命令定义了服务器服务对象:

```
DEFINE SERVICE(S1) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+bin/runmqtrm') +
STARTARG('-m +QMNAME+ -q ACCOUNTS.INITIATION.QUEUE') +
STOPCMD('+MQ_INSTALL_PATH+bin/amqsstop') +
STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+')
```

其中:

+MQ_INSTALL_PATH+ 是表示安装目录的令牌。

+QMNAME+ 是表示队列管理器名称的令牌。

ACCOUNTS.INITIATION.QUEUE 是启动队列。

amqsstop 是随 IBM MQ 提供的样本程序, 它请求队列管理器中断进程标识的所有连接。amqsstop 生成 PCF 命令, 因此命令服务器必须正在运行。

+MQ_SERVER_PID+ 是表示传递到停止程序的进程标识的令牌。

请参阅 [第 159 页的『服务定义上的可替换插入』](#) 以获取公共令牌的列表。

2. 下次启动队列管理器时, 将执行服务器服务对象的实例。但是, 我们将使用 **START SERVICE** MQSC 命令立即启动服务器服务对象的实例:

```
START SERVICE(S1)
```

3. 将使用 **DISPLAY SVSTATUS** MQSC 命令显示服务器服务进程的状态:

```
DISPLAY SVSTATUS(S1)
```


4. 此示例现在显示如何更改服务器服务对象并通过手动重新启动服务器服务进程来获取更新。将更改服务器服务对象，以便将启动队列指定为 JUPITER.INITIATION.QUEUE。使用 **ALTER SERVICE MQSC** 命令：

```
ALTER SERVICE(S1) +
STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

注：正在运行的服务在重新启动之前不会获取对其服务定义的任何更新。

5. 将重新启动服务器服务进程，以便使用 **STOP SERVICE** 和 **START SERVICE MQSC** 命令进行更改：

```
STOP SERVICE(S1)
```

接下来发出以下命令：

```
START SERVICE(S1)
```

服务器服务进程将重新启动，并选取在 [第 161 页的『4』](#) 中进行的更改。

注：仅当在服务定义中指定了 **STOPCMD** 参数时，才能使用 MQSC 命令 **STOP SERVICE**。

相关信息

[变更服务](#)

[定义服务](#)

[显示 SVSTATUS](#)

[启动服务](#)

[停止服务](#)

使用命令服务对象

此示例显示如何定义命令服务对象以在队列管理器启动或停止时启动将条目写入操作系统的系统日志的程序。

1. 使用 **DEFINE SERVICE MQSC** 命令定义了命令服务对象：

```
DEFINE SERVICE(S2) +
CONTROL(QMGR) +
SERVTYPE(COMMAND) +
STARTCMD('/usr/bin/logger') +
STARTARG('Queue manager +QMNAME+ starting') +
STOPCMD('/usr/bin/logger') +
STOPARG('Queue manager +QMNAME+ stopping')
```

其中：

logger 是 UNIX and Linux 系统提供的用于写入系统日志的命令。

+QMNAME+ 是表示队列管理器名称的令牌。

相关信息

[定义服务](#)

在队列管理器仅结束时使用命令服务对象

此示例显示如何定义命令服务对象以在仅当队列管理器停止时启动将条目写入操作系统的系统日志的程序。

1. 使用 **DEFINE SERVICE MQSC** 命令定义了命令服务对象：

```
DEFINE SERVICE(S3) +
CONTROL(QMGR) +
SERVTYPE(COMMAND) +
STOPCMD('/usr/bin/logger') +
STOPARG('Queue manager +QMNAME+ stopping')
```

其中：

logger 是随 IBM MQ 提供的样本程序，可将条目写入操作系统的系统日志。
+QMNAME+ 是表示队列管理器名称的令牌。

相关信息

[定义服务](#)

有关传递参数的更多信息

此示例显示如何定义服务器服务对象以在启动队列管理器时启动名为 runserv 的程序。

此示例使用 Windows 样式路径分隔符进行编写。

要传递到起始程序的其中一个自变量是包含空格的字符串。此自变量需要作为单个字符串传递。要实现此目的，请使用双引号，如以下命令中所示来定义命令服务对象：

1. 使用 **DEFINE SERVICE** MQSC 命令定义了服务器服务对象：

```
DEFINE SERVICE(S1) SERVTYPE(SERVER) CONTROL(QMGR) +
STARTCMD('C:\Program Files\Tools\runserv.exe') +
STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\'') +
STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')

DEFINE SERVICE(S4) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('C:\Program Files\Tools\runserv.exe') +
STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\'') +
STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')
```

其中：

+QMNAME+ 是表示队列管理器名称的令牌。

"C:\Program Files\Tools\'" 是包含空格的字符串，将作为单个字符串传递。

相关信息

[定义服务](#)

自动启动服务

此示例显示如何定义可用于在队列管理器启动时自动启动触发器监视器的服务器服务对象。

1. 使用 **DEFINE SERVICE** MQSC 命令定义了服务器服务对象：

```
DEFINE SERVICE(TRIG_MON_START) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('runmqtrm') +
STARTARG('-m +QMNAME+ -q +IQNAME+')
```

其中：

+QMNAME+ 是表示队列管理器名称的令牌。

+IQNAME+ 是用户在表示启动队列名称的其中一个 service.env 文件中定义的环境变量。

相关信息

[定义服务](#)

管理用于触发的对象

IBM MQ 使您能够在满足队列上的特定条件时自动启动应用程序。例如，当队列上的消息数达到指定的数目时，您可能想要启动应用程序。此工具称为触发。您必须定义支持触发的对象。

在 [使用触发器启动 IBM MQ 应用程序](#) 中详细描述了触发。

定义用于触发的应用程序队列

应用程序队列是应用程序通过 MQI 用于消息传递的本地队列。触发需要在应用程序队列上定义多个队列属性。

触发本身由 **Trigger** 属性 (MQSC 命令中的 TRIGGER) 启用。在此示例中, 当本地队列 MOTOR.INSURANCE.QUEUE, 如下所示:

```
DEFINE QLOCAL (MOTOR.INSURANCE.QUEUE) +
PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
MAXMSGL (2000) +
DEFPSIST (YES) +
INITQ (MOTOR.INS.INIT.QUEUE) +
TRIGGER +
TRIGTYPE (DEPTH) +
TRIGDPTH (100)+
TRIGMPRI (5)
```

其中:

QLOCAL (MOTOR.INSURANCE.QUEUE)

正在定义的应用程序队列的名称。

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)

定义要由触发器监视器程序启动的应用程序的进程定义的名称。

MAXMSGL (2000)

队列中消息的最大长度。

DEFPSIST (YES)

指定缺省情况下此队列上的消息是持久的。

INITQ (MOTOR.INS.INIT.QUEUE)

队列管理器要在其上放置触发器消息的启动队列的名称。

TRIGGER

是触发器属性值。

TRIGTYPE (DEPTH)

指定当所需优先级 (TRIGMPRI) 的消息数达到 TRIGDPTH 中指定的数目时, 将生成触发器事件。

TRIGDPTH (100)

生成触发器事件所需的消息数。

TRIGMPRI (5)

队列管理器在决定是否生成触发器事件时要计算的消息的优先级。仅计算优先级为 5 或更高的消息。

定义启动队列

发生触发器事件时, 队列管理器会将触发器消息放在应用程序队列定义中指定的启动队列上。启动队列没有特殊设置, 但您可以使用本地队列 MOTOR.INS.INIT.QUEUE, 用于指导:

```
DEFINE QLOCAL (MOTOR.INS.INIT.QUEUE) +
GET (ENABLED) +
NOSHARE +
NOTRIGGER +
MAXMSGL (2000) +
MAXDEPTH (1000)
```

定义流程

使用 DEFINE PROCESS 命令来创建进程定义。进程定义定义要用于处理来自应用程序队列的消息的应用程序。应用程序队列定义对要使用的进程进行命名, 从而使应用程序队列与要用于处理其消息的应用程序相关联。这是通过应用程序队列 MOTOR.INSURANCE.QUEUE。以下 MQSC 命令定义必需的进程 MOTOR.INSURANCE.QUOTE.PROCESS, 在此示例中标识:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
```

```
DESCR ('Insurance request message processing') +
APPLTYPE (UNIX) +
APPLICID ('/u/admin/test/IRMP01') +
USERDATA ('open, close, 235')
```

其中:

MOTOR.INSURANCE.QUOTE.PROCESS

是进程定义的名称。

DESCR ('Insurance request message processing')

描述与此定义相关的应用程序。此文本在您使用 DISPLAY PROCESS 命令时显示。这可以帮助您确定流程的作用。如果在字符串中使用空格,那么必须用单引号将字符串括起来。

APPLTYPE (UNIX)

要启动的应用程序的类型。

APPLICID ('/u/admin/test/IRMP01')

应用程序可执行文件的名称,指定为标准文件名。在 Windows 系统中,典型的 APPLICID 值将为 c:\appl\test\irmp01.exe。

USERDATA ('open, close, 235')

是用户定义的数据,可供应用程序使用。

显示进程定义的属性

使用 DISPLAY PROCESS 命令来检查定义的结果。例如:

```
DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)

24 : DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) ALL
AMQ8407: Display Process details.
DESCR ('Insurance request message processing')
APPLICID ('/u/admin/test/IRMP01')
USERDATA (open, close, 235)
PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
APPLTYPE (UNIX)
```

您还可以使用 MQSC 命令 ALTER PROCESS 来变更现有进程定义,使用 DELETE PROCESS 命令来删除进程定义。

在两个系统之间使用 dmpmqmsg 实用程序

dmpmqmsg 实用程序 (以前称为 **qload**) 从 IBM MQ 8.0 合并到产品中。以前, **qload** 实用程序已作为 SupportPac MO03 提供。

概述

dmpmqmsg 实用程序允许您将队列内容或其消息复制或移动到文件。可以根据需要保存此文件,稍后将其用于将消息重新装入到队列中。

要点: 该文件具有实用程序可理解的特定格式。但是,该文件是人类可读的,因此您可以在重新装入该文件之前在编辑器中对其进行更新。如果编辑该文件,那么不得更改其格式。

可能的用途包括:

- 将队列上的消息保存到文件中。可能出于归档目的,稍后将其重新装入到队列中。
- 使用先前保存到文件的消息重新装入队列。
- 从队列中除去旧消息。
- "重放"来自存储位置的测试消息,甚至在需要时保持消息之间的正确时间。



注意: SupportPac MO03 使用 **-l** 参数来指定本地或客户机绑定。**-l** 已替换为 **-c** 参数。

-P 现在用于代码页信息,而不是 **-c**。

有关命令和可用参数的更多信息,请参阅 [dmpmqmsg](#)。

在 Linux 上使用 dmpmqmsg 实用程序的示例，使用 Windows 机器

在 Linux 机器上，您有一个队列管理器，它在队列 (Q1) 上具有要移动到同一队列管理器中的另一个队列 (Q2) 中的消息。您希望从 Windows 机器启动 **dmpmqmsg** 实用程序。

队列 (Q1) 有四条消息是使用样本 **amqsput** (本地队列管理器) 或 **amqsputc** (远程队列管理器) 应用程序添加的。

在 Linux 机器上，您会看到：

```
display ql(Q1) CURDEPTH
      2 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(Q1)
      TYPE(QLOCAL)
      CURDEPTH(4)
```

设置 MQSERVER 环境变量以指向 Linux 中的队列管理器。例如：

```
set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/veracruz.x.com(1414)
```

其中 *veracruz* 是机器的名称。

运行 **dmpmqmsg** 实用程序以从队列 Q1 中读取，并将输出存储在 `c:\temp\mqqlload.txt` 中。

作为远程客户机连接到在 MQSERVER 所建立的 Linux 主机和端口中运行的队列管理器 *QM_VER*。您可以使用以下属性作为远程客户机来实现连接：-c。

```
dmpmqmsg -m QM_VER -i Q1 -f c:\temp\mqqlload.txt -c
Read      - Files:    0  Messages:    4  Bytes:          22
Written - Files:    1  Messages:    4  Bytes:          22
```

输出文件 `c:\temp\mqqlload.txt` 包含文本，使用 **dmpmqmsg** 实用程序理解的格式。

在 Windows 机器上，发出 **dmpmqmsg** 命令 (使用 -o 选项而不是 -i 选项) 以从 Windows 机器上的文件装入 Linux 机器上的队列 (Q2)：

```
dmpmqmsg -m QM_VER -o Q2 -f c:\temp\mqqlload.txt -c
Read      - Files:    1  Messages:    4  Bytes:          22
Written - Files:    0  Messages:    4  Bytes:          22
```

在 Linux 机器上，请注意队列中现在有四条已从文件复原的消息。

```
display ql(Q2) CURDEPTH
      6 : display ql(Q2) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(Q2)
      TYPE(QLOCAL)
      CURDEPTH(4)
```

在 Linux 机器上，

从原始队列中删除消息。

```
clear qllocal(Q1)
      4 : clear qllocal(Q1)
AMQ8022: IBM MQ queue cleared.
```

确认原始队列上没有更多消息：

```
display ql(Q1) CURDEPTH
      5 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
      QUEUE(Q1)
```

```
TYPE(QLOCAL)
CURDEPTH(0)
```

请参阅 [dmpmqmsg](#) 以获取命令及其参数的描述。

相关概念

第 166 页的『[使用 dmpmqmsg 实用程序的示例](#)』

可以使用 **dmpmqmsg** 实用程序 (以前称为 **qload**) 的简单方法。此实用程序已从 IBM MQ 8.0 合并到产品中。

使用 dmpmqmsg 实用程序的示例

可以使用 **dmpmqmsg** 实用程序 (以前称为 **qload**) 的简单方法。此实用程序已从 IBM MQ 8.0 合并到产品中。

以前, **qload** 实用程序可用作 SupportPac MO03。

将队列卸载到文件

在命令行上使用以下选项将队列上的消息保存到文件中:

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile
```

此命令从队列中获取消息的副本, 并将其保存在指定的文件中。

将队列卸载到一系列文件

您可以使用文件名中的 **insert** 字符将队列卸载到一系列文件。在此方式下, 会将每条消息写入新文件:

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile%n
```

此命令将队列卸载到文件 **myfile1**, **myfile2** 和 **myfile3** 等。

从文件装入队列

要使用您在 [第 166 页的『将队列卸载到文件』](#) 中保存的消息重新装入队列, 请在命令行上使用以下选项:

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

此命令将队列卸载到文件 **myfile1**, **myfile2** 和 **myfile3** 等。

从一系列文件装入队列

您可以使用文件名中的 **insert** 字符从一系列文件装入队列。在此方式下, 会将每条消息写入新文件:

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

此命令将队列装入到文件 **myfile1**, **myfile2** 和 **myfile3** 等。

将消息从一个队列复制到另一个队列

将 [第 166 页的『将队列卸载到文件』](#) 中的文件参数替换为另一个队列名称, 并使用以下选项:

```
dmpmqmsg -m QM1 -i Q1 -o Q2
```

此命令允许将来自一个队列的消息复制到另一个队列。

将前 100 条消息从一个队列复制到另一个队列

使用上一个示例中的命令并添加 `-r#100` 选项:

```
dmpmqmsg -m QM1 -i Q1 -o Q2 -r#100
```

将消息从一个队列移至另一个队列

第 166 页的『从文件装入队列』上的变体。 请注意使用仅浏览队列的 `-i` (小写) 与从队列破坏性获取的 `-I` (大写) 之间的区别:

```
dmpmqmsg -m QM1 -I Q1 -o Q2
```

将超过一天的消息从一个队列移至另一个队列

此示例显示使用年龄选择。 可以选择年龄大于, 小于或在一定年龄范围内的消息。

```
dmpmqmsg -m QM1 -I Q1 -o Q2 -T1440
```

显示当前在队列上的消息的时间长度

在命令行上使用以下选项:

```
dmpmqmsg -m QM1 -i Q1 -f stdout -dT
```

使用消息文件

从队列中卸载了消息, 如第 166 页的『将队列卸载到文件』中所示, 您可能想要编辑该文件。

您可能还希望更改文件的格式, 以使用在卸载队列时未指定的其中一个显示选项。

即使在卸载队列之后, 您也可以使用 **dmpmqmsg** 实用程序将文件重新处理为所需格式。 在命令行上使用以下选项。

```
dmpmqmsg -f c:\oldfile -f c:\newfile -dA
```

请参阅 [dmpmqmsg](#) 以获取命令及其参数的描述。

管理远程 IBM MQ 对象

本节说明如何使用 MQSC 命令来管理远程队列管理器上的 IBM MQ 对象, 以及如何使用远程队列对象来控制消息和应答消息的目标。

过程

- 有关如何管理远程 IBM MQ 对象的信息, 请参阅以下子主题:
 - 第 168 页的『通道和远程排队』
 - 第 169 页的『从本地队列管理器进行远程管理』
 - 第 174 页的『创建远程队列的本地定义』
 - 第 176 页的『检查是否已完成用于分布式网络的异步命令』
 - 第 178 页的『使用远程队列定义作为别名』
 - 第 179 页的『数据转换』

通道和远程排队

您可以设置通道以在本地队列管理器与远程队列管理器之间提供单向通信链路。通道可以将消息传送到远程队列管理器中的任意数目的队列。如果您希望远程队列管理器对本地队列管理器发送的消息作出响应，那么可以设置第二个通道以将响应从远程队列管理器发送回本地队列管理器。

队列管理器通过发送消息与另一个队列管理器通信，并在需要时接收响应。接收队列管理器可以是：

- 在同一机器上
- 在同一位置的另一台机器上 (甚至在世界的另一边)
- 在与本地队列管理器相同的平台上运行
- 在 IBM MQ 支持的其他平台上运行

这些消息可能源自：

- 用户编写的应用程序，用于将数据从一个节点传输到另一个节点
- 使用 PCF 命令或 MQAI 的用户编写的管理应用程序
- IBM MQ Explorer。
- 队列管理器正在发送：
 - 检测事件消息到另一个队列管理器
 - 从 **runmqsc** 命令以间接方式发出的 MQSC 命令 (其中命令在另一个队列管理器上运行)

在将消息发送到远程队列管理器之前，本地队列管理器需要一种机制来检测消息的到达并传输这些消息，包括：

- 至少一个通道
- 传输队列
- 通道启动程序

要使远程队列管理器接收消息，需要侦听器。

通道是两个队列管理器之间的单向通信链路，可以将消息发送到远程队列管理器上的任意数目的队列。

通道的每一端都有单独的定义。例如，如果一端是发送方或服务方，那么另一端必须是接收方或请求者。简单通道由本地队列管理器端的发送方通道定义和远程队列管理器端的接收方通道定义组成。这两个定义必须具有相同的名称，并且共同构成单个消息通道。

如果您希望远程队列管理器响应本地队列管理器发送的消息，请设置第二个通道以将响应发送回本地队列管理器。

使用 MQSC 命令 **DEFINE CHANNEL** 来定义通道。在此部分中，除非另有指定，否则与通道相关的示例将使用缺省通道属性。

在通道的每一端都有一个消息通道代理 (MCA)，用于控制消息的发送和接收。MCA 从传输队列中获取消息，并将它们放在队列管理器之间的通信链路上。

传输队列是专门的本地队列，它在 MCA 选取消息并将其发送到远程队列管理器之前临时保存这些消息。在远程队列定义上指定传输队列的名称。

您可以允许 MCA 使用多个线程来传输消息。此过程称为流水线。Pipelining 使 MCA 能够更高效地传输消息，从而提高通道性能。有关如何配置通道以使用管道的详细信息，请参阅 [通道属性](#)。

第 170 页的『为远程管理准备通道和传输队列』告诉您如何使用这些定义来设置远程管理。

有关通常设置分布式排队的更多信息，请参阅 [分布式排队组件](#)。

相关信息

[runmqsc \(运行 MQSC 命令\)](#)

使用集群的远程管理

在使用分布式排队的 IBM MQ 网络中，每个队列管理器都是独立的。如果一个队列管理器需要将消息发送到另一个队列管理器，那么它必须为要向其发送消息的每个队列定义传输队列，到远程队列管理器的通道以及远程队列定义。

集群是一组队列管理器，其设置方式使队列管理器可以通过单个网络直接相互通信，而无需复杂的传输队列、通道和队列定义。可以轻松设置集群，并且通常包含以某种方式逻辑相关且需要共享数据或应用程序的队列管理器。即使是最小的集群也会降低系统管理成本。

在集群中建立队列管理器网络涉及的定义少于建立传统分布式排队环境。通过更少的定义，您可以更快速轻松地设置或更改网络，并降低在定义中发生错误的风险。

要设置集群，每个队列管理器需要一个集群发送方 (CLUSDR) 和一个集群接收方 (CLUSRCVR) 定义。您不需要任何传输队列定义或远程队列定义。在集群中使用时，远程管理的原则相同，但定义本身已大大简化。

从本地队列管理器进行远程管理

您可以使用 MQSC 和 PCF 命令从本地队列管理器管理远程队列管理器。

对于 MQSC 和 PCF 命令，准备队列和通道基本上是相同的。在此部分中，示例显示 MQSC 命令，因为它们更易于理解。有关使用 PCF 命令编写管理程序的更多信息，请参阅第 19 页的『使用 IBM MQ 可编程命令格式』。

将 MQSC 命令以交互方式或从包含这些命令的文本文件发送到远程队列管理器。远程队列管理器可能位于同一台机器上，或者更通常位于另一台机器上。您可以在其他 IBM MQ 环境中远程管理队列管理器，包括：

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

要实现远程管理，必须创建特定对象。除非您有专门的需求，否则缺省值 (例如，最大消息长度) 已足够。

准备队列管理器以进行远程管理

如何使用 MQSC 命令来准备队列管理器以进行远程管理。

第 170 页的图 18 显示了使用 `runmqsc` 命令进行远程管理所需的队列管理器和通道的配置。对象 `source.queue.manager` 是可以从中发出 MQSC 命令的源队列管理器，这些命令 (操作员消息) 的结果将返回到该源队列管理器。对象 `target.queue.manager` 是目标队列管理器的名称，用于处理命令并生成任何操作员消息。

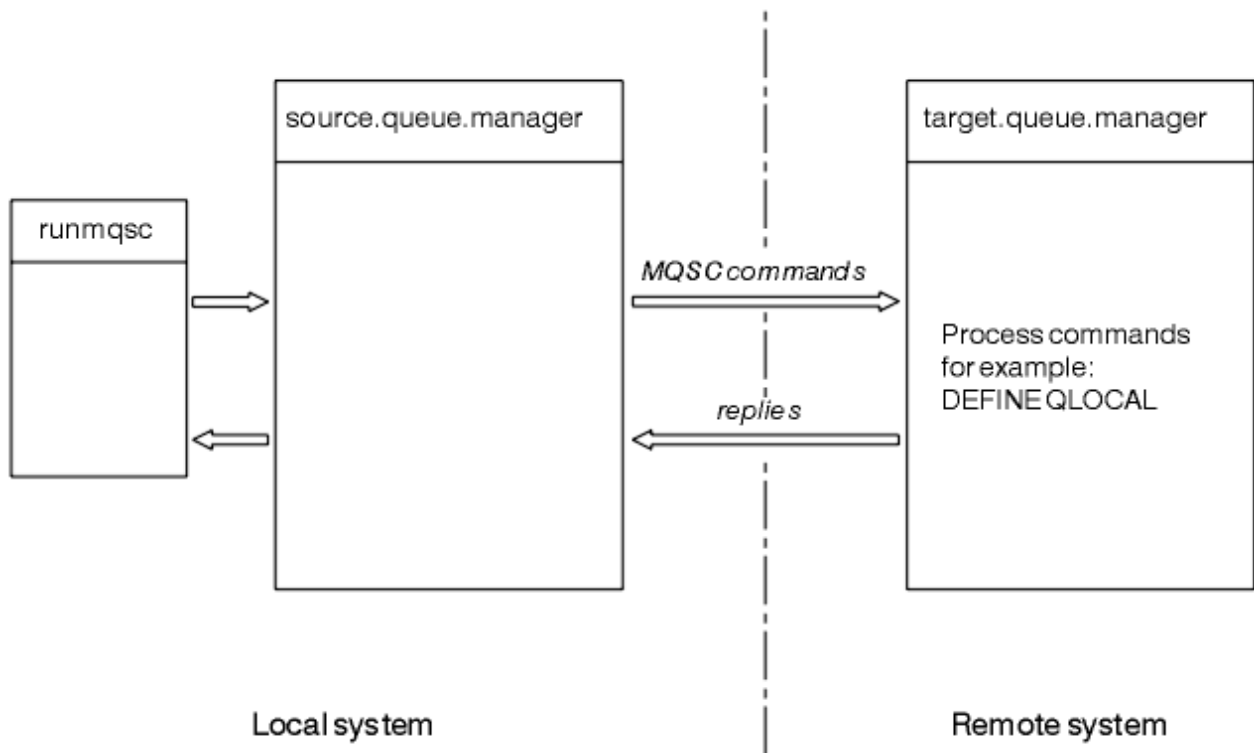


图 18: 使用 MQSC 命令进行远程管理

在这两个系统上，如果尚未执行此操作：

- 使用 `crtmqm` 命令创建队列管理器和缺省对象。有关更多信息，请参阅 [crtmqm](#)。
- 使用 `strmqm` 命令启动队列管理器。有关更多信息，请参阅 [strmqm](#)。

在目标队列管理器上：

- 命令队列 `SYSTEM.ADMIN.COMMAND.QUEUE` 必须存在。缺省情况下，将在创建队列管理器时创建此队列。

您必须在本地或通过 Telnet 之类的网络设施运行这些命令。

为远程管理准备通道和传输队列

如何使用 MQSC 命令来准备用于远程管理的通道和传输队列。

要远程运行 MQSC 命令，请设置两个通道 (每个方向一个通道) 及其关联的传输队列。此示例假定您使用 TCP/IP 作为传输类型，并且您知道所涉及的 TCP/IP 地址。

通道 `source.to.target` 用于将 MQSC 命令从源队列管理器发送到目标队列管理器。其发送方位于 `source.queue.manager`，其接收方位于 `target.queue.manager`。通道 `target.to.source` 用于返回来自命令的输出以及生成到源队列管理器的任何操作员消息。您还必须为每个通道定义一个传输队列。此队列是给出了接收队列管理器的名称的本地队列。除非使用队列管理器别名，否则 XMITQ 名称必须与远程队列管理器名称匹配才能使远程管理工作。第 171 页的图 19 汇总了此配置。

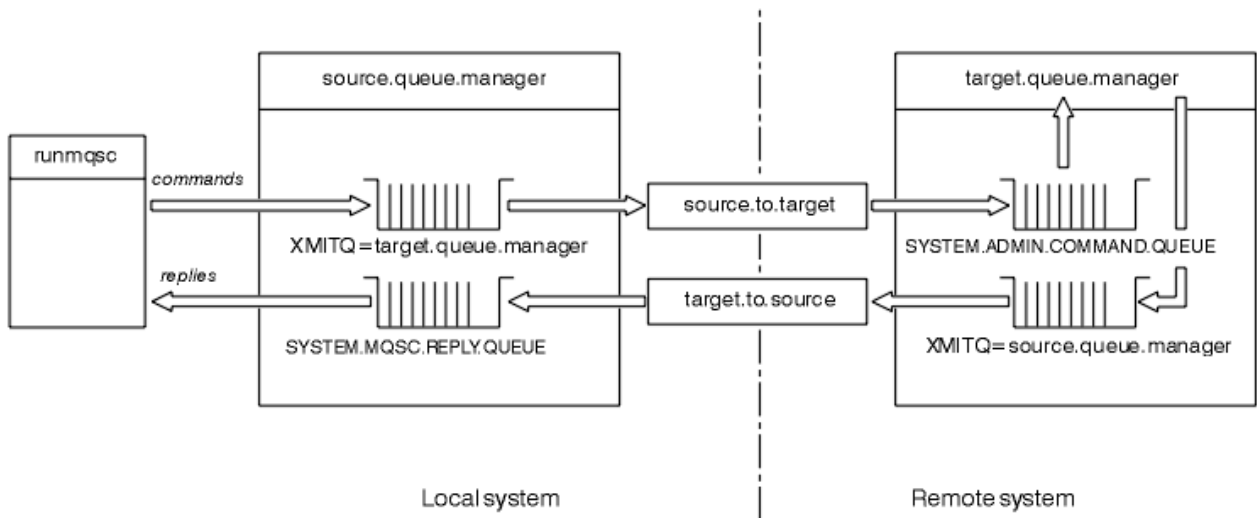


图 19: 设置用于远程管理的通道和队列

有关设置通道的更多信息，请参阅 [配置分布式排队](#)。

定义通道，侦听器 and 传输队列

在源队列管理器 (source.queue.manager) 上，发出以下 MQSC 命令以定义通道，侦听器 and 传输队列：

1. 在源队列管理器上定义发送方通道：

```
DEFINE CHANNEL ('source.to.target') +
CHLTYPE(SDR) +
CONNNAME (RHX5498) +
XMITQ ('target.queue.manager') +
TRPTYPE(TCP)
```

2. 在源队列管理器上定义接收方通道：

```
DEFINE CHANNEL ('target.to.source') +
CHLTYPE(RCVR) +
TRPTYPE(TCP)
```

3. 在源队列管理器上定义侦听器：

```
DEFINE LISTENER ('source.queue.manager') +
TRPTYPE (TCP)
```

4. 在源队列管理器上定义传输队列：

```
DEFINE QLOCAL ('target.queue.manager') +
USAGE (XMITQ)
```

在目标队列管理器 (target.queue.manager) 上发出以下命令以创建通道，侦听器 and 传输队列：

1. 在目标队列管理器上定义发送方通道：

```
DEFINE CHANNEL ('target.to.source') +
CHLTYPE(SDR) +
CONNNAME (RHX7721) +
XMITQ ('source.queue.manager') +
TRPTYPE(TCP)
```

2. 在目标队列管理器上定义接收方通道：

```
DEFINE CHANNEL ('source.to.target') +
  CHLTYPE(RCVR) +
  TRPTYPE(TCP)
```

3. 在目标队列管理器上定义侦听器:

```
DEFINE LISTENER ('target.queue.manager') +
  TRPTYPE (TCP)
```

4. 在目标队列管理器上定义传输队列:

```
DEFINE QLOCAL ('source.queue.manager') +
  USAGE (XMITQ)
```

注: 在发送方通道定义中为 CONNAME 属性指定的 TCP/IP 连接名称仅用于说明。这是连接的其他端的机器的网络名。使用适合于您的网络的值。

启动侦听器和通道

如何使用 MQSC 命令来启动侦听器和通道。

使用以下 MQSC 命令启动这两个侦听器:

1. 通过发出以下 MQSC 命令在源队列管理器 `source.queue.manager` 上启动侦听器:

```
START LISTENER ('source.queue.manager')
```

2. 通过发出以下 MQSC 命令在目标队列管理器 `target.queue.manager` 上启动侦听器:

```
START LISTENER ('target.queue.manager')
```

使用以下 MQSC 命令启动两个发送方通道:

1. 通过发出以下 MQSC 命令在源队列管理器 `source.queue.manager` 上启动发送方通道:

```
START CHANNEL ('source.to.target')
```

2. 通过发出以下 MQSC 命令在目标队列管理器 `target.queue.manager` 上启动发送方通道:

```
START CHANNEL ('target.to.source')
```

通道的自动定义

通过使用 MQSC 命令, ALTER QMGR (或 PCF 命令 "更改队列管理器") 更新队列管理器对象来启用接收方和服务器连接定义的自动定义。

如果 IBM MQ 接收到入站连接请求, 但找不到相应的接收方或服务器连接通道, 那么它会自动创建通道。自动定义基于 IBM MQ 随附的两个缺省定义: SYSTEM.AUTO.RECEIVER 和 SYSTEM.AUTO.SVRCONN。

有关自动创建通道定义的更多信息, 请参阅 [准备通道](#)。有关自动定义集群通道的信息, 请参阅 [使用自动定义的通道](#)。

管理命令服务器以进行远程管理

如何启动, 停止和显示命令服务器的状态。对于涉及 PCF 命令, MQAII 以及远程管理的所有管理, 都必须使用命令服务器。

每个队列管理器都可以有一个与其关联的命令服务器。命令服务器处理来自远程队列管理器的任何入局命令或来自应用程序的 PCF 命令。它向队列管理器提供用于处理的命令, 并根据命令的来源返回完成代码或操作员消息。

注: 对于远程管理, 请确保目标队列管理器正在运行。否则, 包含命令的消息无法离开从中发出这些命令的队列管理器。而是在为远程队列管理器提供服务的本地传输队列中对这些消息进行排队。避免这种情况。

有单独的控制命令用于启动和停止命令服务器。如果命令服务器正在运行, 那么 IBM MQ for Windows 或 IBM MQ for Linux (x86 和 x86-64 平台) 的用户可以使用 IBM MQ Explorer 来执行以下部分中描述的操作。有关更多信息, 请参阅第 114 页的『使用 IBM MQ Explorer 进行管理』。

启动命令服务器

根据队列管理器属性 `SCMDSERV` 的值, 命令服务器将在队列管理器启动时自动启动, 或者必须手动启动。可以使用 MQSC 命令 `ALTER QMGR` 指定参数 `SCMDSERV` 来更改队列管理器属性的值。缺省情况下, 将自动启动命令服务器。

如果 `SCMDSERV` 设置为 `MANUAL`, 请使用以下命令启动命令服务器:

```
stirmqcsv saturn.queue.manager
```

其中, `saturn.queue.manager` 是要对其启动命令服务器的队列管理器。

显示命令服务器的状态

对于远程管理, 请确保目标队列管理器上的命令服务器正在运行。如果它未在运行, 那么无法处理远程命令。包含命令的任何消息都将在目标队列管理器的命令队列中排队。

要显示队列管理器的命令服务器的状态, 请发出以下 MQSC 命令:

```
DISPLAY QMSTATUS CMDSERV
```

停止命令服务器

要结束上一个示例启动的命令服务器, 请使用以下命令:

```
endmqcsv saturn.queue.manager
```

可以通过两种方式停止命令服务器:

- 对于受控停止, 请使用带有 `-c` 标志 (缺省值) 的 `endmqcsv` 命令。
- 对于立即停止, 请使用带有 `-i` 标志的 `endmqcsv` 命令。

注: 停止队列管理器还会结束与其关联的命令服务器。

在远程队列管理器上发出 MQSC 命令

您可以使用特定形式的 `runmqsc` 命令在远程队列管理器上运行 MQSC 命令。

如果要远程处理 MQSC 命令, 那么命令服务器必须正在目标队列管理器上运行。(这在源队列管理器上不是必需的)。有关如何在队列管理器上启动命令服务器的信息, 请参阅第 172 页的『管理命令服务器以进行远程管理』。

然后, 在源队列管理器上, 可以通过输入以下命令以间接方式以交互方式运行 MQSC 命令:

```
runmqsc -w 30 -m source.queue.manager target.queue.manager
```

此形式的 `runmqsc` 命令 (带有 `-w` 标志) 以间接方式运行 MQSC 命令, 其中命令 (以修改后的形式) 放在命令服务器输入队列上并按顺序执行。

输入 MQSC 命令时，会将其重定向到远程队列管理器，在本例中为 `target.queue.manager`。超时设置为 30 秒；如果在 30 秒内未收到应答，那么将在本地（源）队列管理器上生成以下消息：

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

当您停止发出 MQSC 命令时，本地队列管理器将显示已到达的任何超时响应，并废弃任何进一步的响应。

源队列管理器缺省为缺省本地队列管理器。如果在 `runmqsc` 命令中指定 `-m LocalQmgrName` 选项，那么可以指示要通过任何本地队列管理器发出的命令。

在间接方式下，您还可以在远程队列管理器上运行 MQSC 命令文件。例如：

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

其中 `mycomds.in` 是包含 MQSC 命令的文件，`report.out` 是报告文件。

远程发出命令的建议方法

在远程队列管理器上发出命令时，请考虑使用以下方法：

1. 将要在远程系统上运行的 MQSC 命令放在命令文件中。
2. 通过在 `runmqsc` 命令上指定 `-v` 标志，在本地验证 MQSC 命令。
不能使用 `runmqsc` 在另一个队列管理器上验证 MQSC 命令。
3. 检查命令文件是否在本地运行而没有错误。
4. 在远程系统上运行命令文件。

如果您在远程使用 MQSC 命令时迁到问题

如果您在远程运行 MQSC 命令时迁到困难，请确保您具有：

- 已在目标队列管理器上启动命令服务器。
- 定义了有效的传输队列。
- 定义了两个消息通道的两端：
 - 用于发送命令的通道。
 - 要返回应答的通道。
- 在通道定义中指定了正确的连接名称 (CONNAME)。
- 在启动消息通道之前，已启动侦听器。
- 检查断开连接时间间隔是否未到期，例如，如果通道已启动但在一段时间后关闭。如果手动启动通道，那么这尤其重要。
- 从源队列管理器发送对目标队列管理器没有意义的请求（例如，包含在远程队列管理器上不受支持的参数的请求）。

另请参阅第 17 页的『[解决 MQSC 命令的问题](#)』。

在 z/OS 上使用队列管理器

您可以从本指南中描述的平台上的队列管理器向 z/OS 队列管理器发出 MQSC 命令。但是，要执行此操作，必须修改 `runmqsc` 命令以及发送方的通道定义。

特别是，将 `-x` 标志添加到源节点上的 `runmqsc` 命令，以指定目标队列管理器正在 z/OS 下运行：

```
runmqsc -w 30 -x target.queue.manager
```

创建远程队列的本地定义

远程队列的本地定义是本地队列管理器上引用远程队列管理器上的队列的定义。

您不必从本地位置定义远程队列，但这样做的优点是应用程序可以通过其本地定义的名称来引用远程队列，而不必指定由远程队列所在队列管理器的标识限定的名称。

了解远程队列的本地定义如何工作

应用程序连接到本地队列管理器，然后发出 MQOPEN 调用。在打开的调用中，指定的队列名称是本地队列管理器上的远程队列定义的名称。远程队列定义提供目标队列，目标队列管理器和 (可选) 传输队列的名称。要将消息放在远程队列上，应用程序会发出 MQPUT 调用，并指定从 MQOPEN 调用返回的句柄。队列管理器在消息开头的传输头中使用远程队列名称和远程队列管理器名称。此信息用于将消息路由到其在网络中的正确目标。

作为管理员，您可以通过改变远程队列定义来控制消息的目标。

以下示例显示应用程序如何将消息放入由远程队列管理器拥有的队列中。应用程序连接到队列管理器，例如 saturn.queue.manager。目标队列由另一个队列管理器拥有。

在 MQOPEN 调用上，应用程序指定以下字段：

| 字段值 | 描述 |
|------------------------------------------------|-------------------------------------------------------|
| <i>ObjectName</i> CYAN.REMOTE.QUEUE | 指定远程队列对象的局部名。这将定义目标队列和目标队列管理器。 |
| <i>ObjectType</i> (队列) | 将此对象标识为队列。 |
| <i>ObjectQmgrName</i> 空白或 saturn.queue.manager | 此字段是可选字段。 如果为空，那么将采用本地队列管理器的名称。(这是远程队列定义所在的队列管理器。) |

在此之后，应用程序发出 MQPUT 调用以将消息放入此队列。

在本地队列管理器上，可以使用以下 MQSC 命令创建远程队列的本地定义：

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
DESCR ('Queue for auto insurance requests from the branches') +
RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
RQMNAME (jupiter.queue.manager) +
XMITQ (INQUOTE.XMIT.QUEUE)
```

其中：

QREMOTE (CYAN.REMOTE.QUEUE)

指定远程队列对象的局部名。这是连接到此队列管理器的应用程序必须在 MQOPEN 调用中指定的名称，以打开远程队列管理器 jupiter.queue.manager 上的队列 AUTOMOBILE.INSURANCE.QUOTE.QUEUE。

DESCR ('Queue for auto insurance requests from the branches')

提供用于描述队列使用情况的其他文本。

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)

指定远程队列管理器上的目标队列的名称。这是由指定队列名称 CYAN.REMOTE.QUEUE。队列 AUTOMOBILE.INSURANCE.QUOTE.QUEUE 必须定义为远程队列管理器上的本地队列。

RQMNAME (jupiter.queue.manager)

指定拥有目标队列 AUTOMOBILE.INSURANCE.QUOTE.QUEUE。

XMITQ (INQUOTE.XMIT.QUEUE)

指定传输队列的名称。这是可选的；如果未指定传输队列的名称，那么将使用与远程队列管理器同名的队列。

在任一情况下，都必须将相应的传输队列定义为具有 **Usage** 属性的本地队列，该属性指定它是 MQSC 命令中的传输队列 (USAGE (XMITQ))。

将消息放入远程队列的替代方法

使用远程队列的本地定义不是将消息放入远程队列的唯一方法。应用程序可以在 MQOPEN 调用中指定完整队列名称 (包括远程队列管理器名称)。在这种情况下, 您不需要远程队列的本地定义。但是, 这意味着应用程序必须在运行时知道或有权访问远程队列管理器的名称。

将其他命令与远程队列配合使用

您可以使用 MQSC 命令来显示或更改远程队列对象的属性, 也可以删除远程队列对象。例如:

- 要显示远程队列的属性:

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- 更改远程队列以启用放置。这不会影响目标队列, 仅影响指定此远程队列的应用程序:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- 删除此远程队列。这不会影响目标队列, 仅影响其本地定义:

```
DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

注: 删除远程队列时, 仅删除远程队列的本地表示。您不会删除远程队列本身或其上的任何消息。

定义传输队列

传输队列是当队列管理器通过消息通道将消息转发到远程队列管理器时使用的本地队列。

通道提供到远程队列管理器的单向链接。消息将在传输队列中排队, 直到通道可以接受这些消息为止。定义通道时, 必须在消息通道的发送端指定传输队列名称。

MQSC 命令属性 USAGE 定义队列是传输队列还是普通队列。

缺省传输队列

当队列管理器向远程队列管理器发送消息时, 它将使用以下顺序来标识传输队列:

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

例如, 以下 MQSC 命令在 `source.queue.manager` 上为转至 `target.queue.manager` 的消息创建缺省传输队列:

```
DEFINE QLOCAL ('target.queue.manager') +  
DESCR ('Default transmission queue for target qm') +  
USAGE (XMITQ)
```

应用程序可以将消息直接放置在传输队列上, 也可以通过远程队列定义间接放置。另请参阅第 174 页的『创建远程队列的本地定义』。

检查是否已完成用于分布式网络的异步命令

在分布式网络中使用时, 许多命令是异步的。根据命令, 以及发出时的网络状态, 可能需要大量时间才能完成。队列管理器在完成时不会发出消息, 因此您需要其他方法来检查命令是否已完成。

关于此任务

您对集群进行的几乎任何配置更改都可能以异步方式完成。这是因为在集群中运行的内部管理和更新周期。对于发布/预订层次结构，任何影响预订的配置更改都可能以异步方式完成。这在命令的名称中并不总是很明显。

以下 MQSC 命令可能都以异步方式完成。这些命令中的每个命令都具有等效的 PCF，并且大多数命令也可从 IBM MQ Explorer 中获取。在没有工作负载的小型网络上运行时，这些命令通常在几秒钟内完成。但是，对于更大和更繁忙的网络，情况并非如此。此外，**REFRESH CLUSTER** 命令可能需要更长时间，尤其是在同时在多个队列管理器上发出时。

要确信这些命令已完成，请检查远程队列管理器上是否存在期望的对象。

过程

- [ALTER QMGR](#)

对于 [ALTER QMGR PARENT](#) 命令，请使用 `DISPLAY PUBSUB TYPE(PARENT) ALL` 来跟踪请求的父关系的状态。

对于 [ALTER QMGR REPOS](#) 和 [ALTER QMGR REPOSNL](#) 命令，请使用 `DISPLAY CLUSQMGR QMTYPE` 来确认完成。

- [DEFINE CHANNEL](#)，[ALTER CHANNEL](#) 和 [DELETE CHANNEL](#)

对于表 [ALTER CHANNEL parameters](#) 中列出的所有参数，请使用 `DISPLAY CLUSQMGR` 命令来监视何时已将更改传播到集群。

- [DEFINE NAMELIST](#)，[ALTER NAMELIST](#) 和 [DELETE NAMELIST](#)。

如果在 **QMGR** 对象的 **CLUSNL** 属性上使用 **NAMELIST**，那么队列或集群通道可能会影响该对象。根据需要监视受影响的对象。

对 `SYSTEM.QPUBSUB.QUEUE.NAMELIST` 的更改可能会影响在发布/预订层次结构中创建或取消代理预订。使用 `DISPLAY SUB SUBTYPE(PROXY)` 命令来监视此情况。

- [DEFINE 队列](#)，[ALTER 队列](#) 和 [DELETE 队列](#)。

对于表 可由 `DISPLAY QUEUE` 命令返回的参数中列出的所有参数，请使用 `DISPLAY QCLUSTER` 命令来监视何时已将更改传播到集群。

- [DEFINE SUB](#) 和 [DELETE SUB](#)

在主题字符串上定义第一个预订时，可以在发布/预订层次结构或发布/预订集群中创建代理预订。同样，当您删除主题字符串上的最后一个预订时，可能会取消发布/预订层次结构或发布/预订集群中的代理预订。

要检查定义或删除预订的命令是否已完成，请检查分布式网络中的其他队列管理器上是否存在期望的代理预订。如果要在集群中使用直接路由，请检查集群中的其他部分存储库上是否存在期望的代理预订。如果要在集群中使用主题主机路由，请检查匹配的主题主机上是否存在期望的代理预订。使用以下 MQSC 命令：

```
DISPLAY SUB(*) SUBTYPE(PROXY)
```

在集群或层次结构中发出以下等效预订和取消预订 MQI 调用时，请使用相同的检查：

- 使用 [MQSUB](#) 进行预订。
- 通过将 [MQCLOSE](#) 与 [MQCO_REMOVE_SUB](#) 配合使用来取消预订。

- [DEFINE TOPIC](#)，[ALTER TOPIC](#) 和 [DELETE TOPIC](#)

要检查定义，更改或删除集群主题的命令是否已完成，请在集群中的其他部分存储库中显示该主题 (如果您正在使用直接路由) 或在其他主题主机上 (如果您正在使用主题主机路由)。

对于表 可由 `DISPLAY TOPIC` 命令返回的参数中列出的所有参数，请使用 `DISPLAY TCLUSTER` 命令来监视何时已将更改传播到集群。

注：

- **CLUSTER** 参数可能会影响在发布/预订集群中创建或取消代理预订。
- **PROXYSUB** 和 **SUBSCOPE** 参数可能会影响在发布/预订层次结构或发布/预订集群中创建或取消代理预订。
- 使用 **DISPLAY SUB SUBTYPE(PROXYSUB)** 命令来监视此情况。
- **刷新集群**

如果您正在运行 **REFRESH CLUSTER** 命令，请轮询集群命令队列深度。等待它到达零，并保持为零，然后再查找对象。

1. 使用以下 MQSC 命令来检查集群命令队列深度是否为零。

```
DISPLAY QL(SYSTEM.CLUSTER.COMMAND.QUEUE) CURDEPTH
```

2. 重复检查，直到队列深度达到零，并在后续检查中保持为零。

REFRESH CLUSTER 命令会除去并重新创建对象，并且在大型配置中可能需要大量时间才能完成。请参阅 [针对发布/预订集群的 REFRESH CLUSTER 注意事项](#)。

- **REFRESH QMGR 类型 (PROXYSUB)**

要检查 **REFRESH QMGR TYPE(PROXYSUB)** 命令是否已完成，请检查是否已在分布式网络中的其他队列管理器上更正代理预订。如果要在集群中使用直接路由，请检查是否已在集群中的其他部分存储库上更正代理预订。如果要在集群中使用主题主机路由，请检查是否已在匹配的主题主机上更正期望的代理预订。使用以下 MQSC 命令：

```
DISPLAY SUB(*) SUBTYPE(PROXYSUB)
```

- **Reset Cluster**

要检查 **RESET CLUSTER** 命令是否已完成，请使用 **DISPLAY CLUSQMGR**。

- **RESET QMGR 类型 (PUBSUB)**

要检查 **RESET QMGR** 命令是否已完成，请使用 **DISPLAY PUBSUB TYPE(PARENT|CHILD)**。

注: **RESET QMGR** 命令可能会导致取消发布/预订层次结构或发布/预订集群中的代理预订。使用 **DISPLAY SUB SUBTYPE(PROXYSUB)** 命令来监视此情况。

- 您可能还希望监视其他系统队列，这些队列在命令完成时趋向于队列深度为零。

例如，您可能想要监视 **SYSTEM.INTER.QMGR.CONTROL** 队列和 **SYSTEM.INTER.QMGR.FANREQ** 队列。请参阅 [监视集群中的代理预订流量](#) 和 [均衡发布/预订网络中的生产者 and 使用者](#)。

下一步做什么

如果这些检查未确认异步命令已完成，那么可能发生了错误。要进行调查，请先检查发出该命令的队列管理器的日志，然后 (对于集群) 检查集群完整存储库日志。

相关信息

 [z/OS 上 CLUSTER 命令的异步行为](#)

使用远程队列定义作为别名

除了在另一个队列管理器上查找队列外，还可以将远程队列的本地定义用于队列管理器别名和应答队列别名。这两种类型的别名都通过远程队列的本地定义进行解析。您必须设置相应的通道以使消息到达其目标。

队列管理器别名

别名是一个进程，消息中指定的目标队列管理器的名称由消息路由上的队列管理器修改。队列管理器别名很重要，因为您可以使用它们来控制队列管理器网络中的消息目标。

您可以通过在控制点更改队列管理器上的远程队列定义来执行此操作。发送应用程序不知道指定的队列管理器名称是别名。

有关队列管理器别名的更多信息，请参阅 [什么是别名?](#)

应答队列别名

(可选) 应用程序可以在将请求消息放入队列时指定应答队列的名称。

如果处理消息的应用程序抽取了应答队列的名称，那么它知道在哪里发送应答消息(如果需要)。

应答队列别名是由消息路由上的队列管理器改变请求消息中指定的应答队列的过程。发送应用程序不知道指定的应答队列名称是别名。

应答队列别名允许您更改应答队列的名称及其队列管理器(可选)。这反过来使您能够控制用于应答消息的路由。

有关请求消息，应答消息和应答队列的更多信息，请参阅 [消息类型](#) 和 [应答队列和队列管理器](#)。

有关应答队列别名的更多信息，请参阅 [应答队列别名和集群](#)。

数据转换

IBM MQ 定义的格式(也称为内置格式)中的消息数据可以由队列管理器从一个编码字符集转换为另一个编码字符集，前提是这两个字符集都与单个语言或一组类似语言相关。

例如，支持在具有标识(CCSID) 850 和 500 的编码字符集之间进行转换，因为两者都适用于西欧语言。

对于 EBCDIC 换行符(NL) 字符到 ASCII 的转换，请参阅 [所有队列管理器](#)。

在 [数据转换处理](#) 中定义了受支持的转换。

当队列管理器无法转换内置格式的消息时

如果消息的 CCSID 表示不同的本地语言组，那么队列管理器无法自动转换内置格式的消息。例如，不支持 CCSID 850 与 CCSID 1025 (这是使用西里尔文脚本的语言的 EBCDIC 编码字符集) 之间的转换，因为一个编码字符集中的许多字符不能在另一个编码字符集中表示。如果您具有以不同本地语言工作的队列管理器网络，并且不支持在某些编码字符集之间进行数据转换，那么可以启用缺省转换。

V 9.0.0 对于 `ccsid_part2.tbl` 适用的平台，请参阅第 182 页的『指定缺省数据转换』使用 `ccsid_part2.tbl` 以获取更多信息。第 180 页的『缺省数据转换』中描述了除 `ccsid_part2.tbl` 文件适用的平台以外的平台上的缺省数据转换。

IBM MQ 9.0 中增强的 Unicode 数据转换支持

V 9.0.0

在 IBM MQ 9.0 之前，产品的先前版本不支持将包含 Unicode 代码点的数据转换到基本多语言平面(高于 U+FFFF 的代码点)之外。Unicode 数据转换支持仅限于 Unicode 3.0 标准中定义的代码点，以 UTF-8 或 UCS-2 编码，这是 UTF-16 的 2 字节固定宽度子集。

从 IBM MQ 9.0 开始，IBM MQ 在数据转换中支持 Unicode 8.0 标准中定义的所有 Unicode 字符。这包括对 UTF-16 的完全支持，包括代理对(一对 2-byte UTF-16 字符，范围为 X'D800' 到 X'DFFF'，表示高于 U+FFFF 的 Unicode 代码点)。

在一个 CCSID 中的预组成字符映射到另一个 CCSID 中的组合字符序列的情况下，也支持组合字符序列。

在某些平台上，扩展了与 Unicode 和 CCSID 1388，1390，1399，4933，5488 和 16884 之间的数据转换，以支持当前为这些 CCSID 定义的所有代码点，包括那些映射到 Unicode 补充平面中的代码点的代码点。

对于 CCSID 1390,1399 和 16884，这包括在 JIS X 0213 (JIS2004) 标准中定义的字符。

还增加了对 Unicode 和 6 个新的 CCSID (1374 到 1379) 之间的转换的支持。

ccsid_part2.tbl 文件

V 9.0.0


从 IBM MQ 9.0 提供了另一个文件 `ccsid_part2.tbl`。

`ccsid_part2.tbl` 文件优先于 `ccsid.tbl` 文件，并且：

- 允许添加或修改 CCSID 条目
- 指定缺省数据转换
- 指定其他命令级别的数据

`ccsid_part2.tbl` 仅适用于以下平台：

-  Linux - 所有版本
-  Solaris
-  Windows


 从 IBM MQ 9.0 开始，在 IBM MQ for Windows 上，缺省情况下 `ccsid_part2.tbl` 位于目录 `MQDataRoot\conv\table` 中。此外，在 IBM MQ for Windows 上，它会记录所有受支持的代码集。



  从 IBM MQ 9.0 开始，在 IBM MQ for Linux 和 Solaris 平台上，`ccsid_part2.tbl` 位于目录 `MQDataRoot/conv/table` 中。对于所有 Linux 和 Solaris 平台，受支持的代码集保存在 IBM MQ 提供的转换表中。

虽然 `ccsid_part2.tbl` 文件将替换先前版本的 IBM MQ 中用于提供其他 CCSID 信息的现有 `ccsid.tbl` 文件，但 `ccsid.tbl` 文件将继续由 IBM MQ 解析，因此不得删除。

有关更多信息，请参阅第 181 页的『[ccsid_part2.tbl 文件](#)』。


ccsid.tbl 文件

 在 `ccsid_part2.tbl` 应用的平台以外的平台上，文件 `ccsid.tbl` 用于以下目的：

-   在 AIX 和 HP-UX 平台上，受支持的代码集由操作系统在内部保存。
- 它指定任何其他代码集。要指定其他代码集，需要编辑 `ccsid.tbl` (在文件中提供了有关如何执行此操作的指导)。
- 它指定任何缺省数据转换。

您可以更新 `ccsid.tbl` 中记录的信息；例如，如果操作系统的未来发行版支持其他编码字符集，那么可能需要执行此操作。


缺省数据转换

 从 IBM MQ 9.0 开始，缺省数据转换方法在以下平台上已更改：

- Linux - 所有版本
- Solaris
- Windows

请参阅第 182 页的『[指定缺省数据转换](#)』使用 `ccsid_part2.tbl` 以获取更多信息。

如果在通常不支持数据转换的两台机器之间设置通道，那么必须启用缺省数据转换才能使通道正常工作。

 在 `ccsid_part2.tbl` 适用的平台以外的平台上，要启用缺省数据转换，请编辑 `ccsid.tbl` 文件以指定缺省 EBCDIC CCSID 和缺省 ASCII CCSID。有关如何执行此操作的指示信息包含在文件中。必须在将使用通道连接的所有机器上执行此操作。重新启动队列管理器以使更改生效。

缺省数据转换过程如下所示：

- 如果源 CCSID 和目标 CCSID 之间的转换不受支持，但源环境和目标环境的 CCSID 都是 EBCDIC 或 ASCII，那么字符数据将传递到目标应用程序而不进行转换。

- 如果一个 CCSID 表示 ASCII 编码字符集，另一个表示 EBCDIC 编码字符集，那么 IBM MQ 使用 `ccsid.tbl` 中定义的缺省数据转换 CCSID 来转换数据。

注: 尝试将要转换的字符限制为那些在为消息指定的编码字符集和缺省编码字符集中具有相同代码值的字符。如果仅使用对 IBM MQ 对象名有效的字符集 (如命名 IBM MQ 对象中所定义) 通常，您将满足此要求。在日本使用 EBCDIC CCSID 290, 930, 1279 和 5026 时发生异常，其中小写字符与其他 EBCDIC CCSID 中使用的代码不同。

以用户定义的格式转换消息

队列管理器无法将用户定义格式的消息从一个编码字符集转换为另一个编码字符集。如果需要以用户定义的格式转换数据，那么必须为每种此类格式提供数据转换出口。请勿使用缺省 CCSID 来转换用户定义格式的字符数据。有关以用户定义格式转换数据和写入数据转换出口的更多信息，请参阅 [编写数据转换出口](#)。

更改队列管理器 CCSID

当您使用 `ALTER QMGR` 命令的 `CCSID` 属性来更改队列管理器的 `CCSID` 时，请停止并重新启动队列管理器，以确保停止并重新启动所有正在运行的应用程序 (包括命令服务器和通道程序)。

这是必需的，因为更改队列管理器 `CCSID` 时正在运行的任何应用程序都将继续使用现有 `CCSID`。

V 9.0.0 `ccsid_part2.tbl` 文件

从 IBM MQ 9.0 开始，`ccsid_part2.tbl` 文件将替换产品的先前版本中使用的现有 `ccsid.tbl` 文件，以提供其他 `CCSID` 信息。请注意，`ccsid.tbl` 文件继续由 IBM MQ 解析，不应删除，并且 `ccsid_part2.tbl` 中的条目优先于 `ccsid.tbl` 中的其他条目。

`ccsid_part2.tbl` 位于以下位置的 `MQDataRoot/conv/table` 目录中:

- Linux - 所有版本
- Solaris

以及 Windows 上的 `MQDataRoot\conv\table` 目录

请注意，`ccsid_part2.tbl` 文件在任何其他平台上都不可用。该文件用于执行以下操作:

- 添加或修改现有 `CCSID` 项以在 IBM MQ 数据转换中使用
- 指定缺省数据转换

您应该使用 `ccsid_part2.tbl` 作为此文件:

- 包含对新的 Unicode 编码值的支持。
- 允许您指定 `CCSID` 项的版本，以便这些项仅适用于所选命令级别。

添加或修改 CCSID 项

`ccsid_part2.tbl` 文件中的条目具有以下格式:

```
<CCSID number> <Base CCSID> <DBCS CodePage> <SBCS CodePage>
<Type> <Encoding> <ACRI> <Name>
```

`CCSID 1200 (UTF-16)` 的示例条目为:

```
1200 1200 1200 1200 3 8 0 UTF-16
```

注: 有关 `ACRI` 值的更多详细信息，请参阅 `ccsid_part2.tbl` 文件中的注释。

在 `ccsid_part2.tbl` 格式中:

类型可以等于:

- 1=SBCS
- 2=DBCS

3=MBCS

编码可以等于:

1=EBCDIC

2 = ASCII

3 = ISO

4 = UCS-2

5 = UTF-8

6 = Euc

7 = GB18030

8 = UTF-16

9 = UTF-32

编辑文件时:

- 可以使用行开头的 # 符号来指定注释。这将阻止 IBM MQ 尝试解析该行。
- 无法提供直接插入注释。
- 必须确保不创建空行。
- 不得在文件末尾添加新条目。

应该在 ACRI 表信息之前添加新的 CCSID 项。

指定缺省数据转换

如果不支持在两个 CCSID 之间进行转换，那么可以定义缺省转换 CCSID，这些 CCSID 用于在 ASCII 或类似和 EBCDIC CCSID 之间进行转换。

如果启用此功能，那么缺省转换用于传输和消息头，也可用于用户数据转换。

通过创建类似于以下内容的两行来启用缺省转换:

```
default      0      500      1      1      0
default      0      850      1      2      0
```

第一行将 EBCDIC CCSID 的缺省值设置为 500，第二行将 ASCII 和类似 CCSID 的缺省值设置为 850。

指定不同命令级别的数据

要为 IBM MQ 的不同命令级别指定 CCSID 条目，请使用冒号，后跟您希望下一节适用的命令级别 (或命令级别) IBM MQ。

该数字表示队列管理器或客户机必须在其中运行的最低命令级别。例如，如果当前队列管理器是命令级别 900，并且迁到 800 或 900 命令级别标志，那么将读取 CCSID。

但是，级别为 800 的队列管理器将忽略 900 部分中的任何 CCSID。

指定的命令级别适用于在命令级别标志之后迁到的所有 CCSID 项，直到找到新的命令级别标志为止。

如果需要将命令级别设置为所有命令级别，请指定数字零。

首次解析 ccsid_part2.tbl 时，IBM MQ 会将迁到的所有 CCSID 视为对 IBM MQ 的所有命令级别都有效。

仅当 IBM MQ 迁到第一个命令级别标志时，才开始使用版本控制。

以下代码片段显示了使用版本控制的示例:

```
# Comment Block
# End of Comment Block
# Because no command level flag is specified and we're at the start of the file
# the following CCSIDs will be read on all versions
  819  819  0      819  1  3  0  ISO8859-1
```

```

923 923 0 923 1 3 0 IS08859-15
1051 1051 0 1051 1 3 0 IBM-1051
# The colon :900 below shows that the CCSIDs after will only be for MQ cmd level 900 and above
:900
8629 437 0 437 1 2 0 IBM-437
12725 437 0 437 1 2 0 IBM-437
16821 437 0 437 1 2 0 IBM-437
20917 437 0 437 1 2 0 IBM-437
# The colon :0 below shows that the CCSIDs after will be for all version of MQ
:0
4946 850 0 850 1 2 0 IBM-850
33618 850 0 850 1 2 0 IBM-850
61697 850 0 850 1 2 0 IBM-850
61698 850 0 850 1 2 0 IBM-850

```

Windows

Linux

AIX

管理 MQ Telemetry

MQ Telemetry 使用 IBM MQ Explorer 或在命令行上进行管理。使用资源管理器来配置遥测通道，控制遥测服务以及监视连接到 IBM MQ 的 MQTT 客户机。使用 JAAS、TLS 和 IBM MQ 对象权限管理器来配置 MQ Telemetry 的安全性。

使用 IBM MQ Explorer 进行管理

使用资源管理器来配置遥测通道，控制遥测服务以及监视连接到 IBM MQ 的 MQTT 客户机。使用 JAAS、TLS 和 IBM MQ 对象权限管理器来配置 MQ Telemetry 的安全性。

使用命令行进行管理

可以使用 IBM MQ MQSC 命令在命令行上完全管理 MQ Telemetry。

MQ Telemetry 文档还具有样本脚本，用于演示 IBM MQ Telemetry Transport v3 客户机应用程序的基本用法。

在使用 [IBM MQ Telemetry Transport 样本程序](#) 中的样本之前，请先阅读并了解这些样本。

相关信息

[MQ Telemetry](#)

[MQXR 属性](#)

Linux

AIX

在 Linux 和 AIX 上配置队列管理器以进行遥测

遵循以下手动步骤来配置队列管理器以运行 MQ Telemetry。您可以运行自动化过程，以使用 IBM MQ Explorer 的 MQ Telemetry 支持来设置更简单的配置。

开始之前

1. 有关如何安装 IBM MQ 和 MQ Telemetry 功能部件的信息，请参阅 [安装 MQ Telemetry](#)。
2. 创建并启动队列管理器。在此任务中，队列管理器称为 *qMgr*。
3. 在此任务中，您将配置遥测 (MQXR) 服务。MQXR 属性设置存储在特定于平台的属性文件中：`mqxr_unix.properties`。您通常不需要直接编辑 MQXR 属性文件，因为几乎所有设置都可以通过 MQSC 管理命令或 IBM MQ Explorer 进行配置。如果确实决定直接编辑该文件，请先停止队列管理器，然后再进行更改。请参阅 [MQXR 属性](#)。

关于此任务

IBM MQ Explorer 的 MQ Telemetry 支持包含向导和样本命令过程 `sampleMQM`。他们使用访客用户标识设置初始配置；请参阅 [使用 IBM MQ Explorer 验证 MQ Telemetry 的安装](#) 和 [IBM MQ Telemetry Transport 样本程序](#)。

遵循本任务中的步骤，使用不同的授权方案手动配置 MQ Telemetry。

过程

1. 打开遥测样本目录中的命令窗口。

遥测样本目录为 `/opt/mqm/mqxr/samples`。

2. 创建遥测传输队列。

```
echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc qMgr
```

首次启动遥测 (MQXR) 服务时，它会创建 `SYSTEM.MQTT.TRANSMIT.QUEUE`。

它是在此任务中手动创建的，因为在启动遥测 (MQXR) 服务之前，`SYSTEM.MQTT.TRANSMIT.QUEUE` 必须存在，以授权对其进行访问。

3. 设置缺省传输队列

首次启动遥测 (MQXR) 服务时，它不会更改队列管理器以使 `SYSTEM.MQTT.TRANSMIT.QUEUE` 成为缺省传输队列。

要使 `SYSTEM.MQTT.TRANSMIT.QUEUE` 成为缺省传输队列，请更改缺省传输队列属性。使用 IBM MQ Explorer 或以下示例中的命令来变更属性：

```
echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
```

更改缺省传输队列可能会干扰现有配置。将缺省传输队列更改为 `SYSTEM.MQTT.TRANSMIT.QUEUE` 的原因是为了更轻松地将消息直接发送到 MQTT 客户机。在不改变缺省传输队列的情况下，必须为接收 IBM MQ Explorer 消息的每个客户机添加远程队列定义；请参阅第 187 页的『直接向客户机发送消息』。

4. 遵循第 189 页的『授权 MQTT 客户机访问 IBM MQ 对象』中的过程来创建一个或多个用户标识。用户标识具有向 MQTT 客户机发布，预订和发送发布的权限。

5. 安装遥测 (MQXR) 服务

```
cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
```

另请参阅第 185 页的图 20 中的示例代码。

6. 启动服务

```
echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
```

启动队列管理器时，将自动启动遥测 (MQXR) 服务。

它在此任务中手动启动，因为队列管理器已在运行。

7. 使用 IBM MQ Explorer，配置遥测通道以接受来自 MQTT 客户机的连接。

必须配置遥测通道，以使其身份是步骤 4 中定义的用户标识之一。

另请参阅 `DEFINE CHANNEL (MQTT)`。

8. 通过运行样本客户机来验证配置。

要让样本客户机使用遥测通道，该通道必须授权客户机发布，预订和接收发布。缺省情况下，样本客户机连接到端口 1883 上的遥测通道。另请参阅 [IBM MQ Telemetry Transport 样本程序](#)。

示例

第 185 页的图 20 显示了用于在 Linux 上手动创建 `SYSTEM.MQXR.SERVICE` 的 `runmqsc` 命令。


```

DEF SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+/mqxr/bin/runMQXRService.sh') +
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+" -g "+MQ_DATA_PATH+"') +
STOPCMD('+MQ_INSTALL_PATH+/mqxr/bin/endMQXRService.sh') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+/mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+/mqxr.stderr')

```

图 20: `installMQXRService_unix.mqsc`

Windows 在 Windows 上配置队列管理器以进行遥测

遵循以下手动步骤来配置队列管理器以运行 MQ Telemetry。您可以运行自动化过程，以使用 IBM MQ Explorer 的 MQ Telemetry 支持来设置更简单的配置。

开始之前

1. 有关如何安装 IBM MQ 和 MQ Telemetry 功能部件的信息，请参阅 [安装 MQ Telemetry](#)。
2. 创建并启动队列管理器。在此任务中，队列管理器称为 `qMgr`。
3. 在此任务中，您将配置遥测 (MQXR) 服务。MQXR 属性设置存储在特定于平台的属性文件中：`mqxr_win.properties`。您通常不需要直接编辑 MQXR 属性文件，因为几乎所有设置都可以通过 MQSC 管理命令或 IBM MQ Explorer 进行配置。如果确实决定直接编辑该文件，请先停止队列管理器，然后再进行更改。请参阅 [MQXR 属性](#)。

关于此任务

IBM MQ Explorer 的 MQ Telemetry 支持包含向导和样本命令过程 `sampleMQM`。他们使用访客用户标识设置初始配置；请参阅 [使用 IBM MQ Explorer 验证 MQ Telemetry 的安装](#) 和 [IBM MQ Telemetry Transport 样本程序](#)。

遵循本任务中的步骤，使用不同的授权方案手动配置 MQ Telemetry。

过程

1. 打开遥测样本目录中的命令窗口。
遥测样本目录为 `WMQ program installation directory\mqxr\samples`。
2. 创建遥测传输队列。

```
echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
```

首次启动遥测 (MQXR) 服务时，它会创建 `SYSTEM.MQTT.TRANSMIT.QUEUE`。

它是在此任务中手动创建的，因为在启动遥测 (MQXR) 服务之前，`SYSTEM.MQTT.TRANSMIT.QUEUE` 必须存在，以授权对其进行访问。

3. 设置 `qMgr` 的缺省传输队列

```
echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
```

图 21: 设置缺省传输队列

首次启动遥测 (MQXR) 服务时，它不会更改队列管理器以使 `SYSTEM.MQTT.TRANSMIT.QUEUE` 成为缺省传输队列。

要使 `SYSTEM.MQTT.TRANSMIT.QUEUE` 成为缺省传输队列，请更改缺省传输队列属性。使用 IBM MQ Explorer 或 [第 185 页的图 21](#) 中的命令来变更属性。

更改缺省传输队列可能会干扰现有配置。将缺省传输队列更改为 `SYSTEM.MQTT.TRANSMIT.QUEUE` 的原因是为了更轻松地将消息直接发送到 MQTT 客户机。在不改变缺省传输队列的情况下，必须为接收 IBM MQ 消息的每个客户机添加远程队列定义；请参阅第 187 页的『直接向客户机发送消息』。

4. 遵循第 189 页的『授权 MQTT 客户机访问 IBM MQ 对象』中的过程来创建一个或多个用户标识。用户标识具有向 MQTT 客户机发布、预订和发送发布的权限。
5. 安装遥测 (MQXR) 服务

```
type
installMQXRService_win.mqsc | runmqsc qMgr
```

6. 启动服务

```
echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
```

启动队列管理器时，将自动启动遥测 (MQXR) 服务。

它在此任务中手动启动，因为队列管理器已在运行。

7. 使用 IBM MQ Explorer，配置遥测通道以接受来自 MQTT 客户机的连接。

必须配置遥测通道，以使其身份是步骤 4 中定义的用户标识之一。

另请参阅 [DEFINE CHANNEL \(MQTT\)](#)。

8. 通过运行样本客户机来验证配置。

要让样本客户机使用遥测通道，该通道必须授权客户机发布、预订和接收发布。缺省情况下，样本客户机连接到端口 1883 上的遥测通道。另请参阅 [IBM MQ Telemetry Transport 样本程序](#)。

手动创建 SYSTEM.MQXR.SERVICE

第 186 页的图 22 显示了用于在 Windows 上手动创建 `SYSTEM.MQXR.SERVICE` 的 `runmqsc` 命令。

```
DEF SERVICE(SYSTEM.MQXR.SERVICE) +
CONTROL(QMGR) +
DESCR('Manages clients using MQXR protocols such as MQTT') +
SERVTYPE(SERVER) +
STARTCMD('+MQ_INSTALL_PATH+\mqxr\bin\runMQXRService.bat') +
STARTARG('-m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+\" -g "+MQ_DATA_PATH+\"') +
STOPCMD('+MQ_INSTALL_PATH+\mqxr\bin\endMQXRService.bat') +
STOPARG('-m +QMNAME+') +
STDOUT('+MQ_Q_MGR_DATA_PATH+\mqxr.stdout') +
STDERR('+MQ_Q_MGR_DATA_PATH+\mqxr.stderr')
```

图 22: `installMQXRService_win.mqsc`

Windows

Linux

AIX

配置分布式排队以将消息发送到 MQTT 客户机

IBM MQ 应用程序可以通过发布到客户机创建的预订或直接发送消息来发送 MQTT v3 客户机消息。无论使用哪种方法，都将消息放在 `SYSTEM.MQTT.TRANSMIT.QUEUE` 上，并通过遥测 (MQXR) 服务发送到客户机。可以通过多种方法在 `SYSTEM.MQTT.TRANSMIT.QUEUE` 上放置消息。

发布消息以响应 MQTT 客户机预订

遥测 (MQXR) 服务代表 MQTT 客户机创建预订。客户机是与客户机发送的预订匹配的任何发布的目标。遥测服务会将匹配的发布转发回客户机。

MQTT 客户机作为队列管理器连接到 IBM MQ，其队列管理器名称设置为其 `ClientIdentifier`。要发送到客户机的发布的目标是传输队列 `SYSTEM.MQTT.TRANSMIT.QUEUE`。遥测服务使用目标队列管理器名称作为特定客户机的密钥，将 `SYSTEM.MQTT.TRANSMIT.QUEUE` 上的消息转发到 MQTT 客户机。

遥测 (MQXR) 服务使用 `ClientIdentifier` 作为队列管理器名称来打开传输队列。遥测 (MQXR) 服务将队列的对象句柄传递到 `MQSUB` 调用，以转发与客户机预订匹配的发布。在对象名解析中，将创建 `ClientIdentifier` 作为远程队列管理器名称，并且传输队列必须解析为

SYSTEM.MQTT.TRANSMIT.QUEUE。通过使用标准 IBM MQ 对象名解析，将按如下所示解析 *ClientIdentifier*；请参阅第 187 页的表 14。

1. *ClientIdentifier* 与任何内容都不匹配。

ClientIdentifier 是远程队列管理器名称。它与本地队列管理器名称，队列管理器别名或传输队列名称不匹配。

未定义队列名称。目前，遥测 (MQXR) 服务将 SYSTEM.MQTT.PUBLICATION.QUEUE 设置为队列的名称。MQTT v3 客户机不支持队列，因此客户机将忽略已解析的队列名称。

必须将本地队列管理器属性 **缺省传输队列** 的名称设置为 SYSTEM.MQTT.TRANSMIT.QUEUE，以便将发布放在 SYSTEM.MQTT.TRANSMIT.QUEUE 上以发送到客户机。

2. *ClientIdentifier* 与名为 *ClientIdentifier* 的队列管理器别名匹配。

ClientIdentifier 是远程队列管理器名称。它与队列管理器别名的名称相匹配。

必须使用 *ClientIdentifier* 作为远程队列管理器名称来定义队列管理器别名。

通过在队列管理器别名定义中设置传输队列名称，不必将缺省传输设置为 SYSTEM.MQTT.TRANSMIT.QUEUE。

| <i>ClientIdentifier</i> | Input | | Output | | |
|----------------------------------------|-------------------------|------|-------------------------|------|---------------------------------------|
| | 队列管理器名称 | 队列名称 | 队列管理器名称 | 队列名称 | 传输队列 |
| 不匹配任何内容 | <i>ClientIdentifier</i> | 未定义 | <i>ClientIdentifier</i> | 未定义 | 缺省传输队列。 SYSTEM.MQTT.TRANSMIT.QUEUE |
| 与名为 <i>ClientIdentifier</i> 的队列管理器别名匹配 | <i>ClientIdentifier</i> | 未定义 | <i>ClientIdentifier</i> | 未定义 | SYSTEM.MQTT.TRANSMIT.QUEUE |

有关名称解析的更多信息，请参阅 [名称解析](#)。

任何 IBM MQ 程序都可以发布到同一主题。该出版物将发送到其订户，包括预订该主题的 MQTT v3 客户机。

如果在集群中创建了具有属性 CLUSTER(*clusterName*) 的管理主题，那么集群中的任何应用程序都可以发布到客户机；例如：

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

图 23: 在 Windows 上定义集群主题

注: 请勿为 SYSTEM.MQTT.TRANSMIT.QUEUE 提供集群属性。

MQTT 客户机订户和发布程序可以连接到不同的队列管理器。订户和发布者可以是同一集群的一部分，也可以通过发布/预订层次结构进行连接。发布将使用 IBM MQ 从发布程序交付到订户。

直接向客户机发送消息

作为创建预订并接收与预订主题相匹配的发布的客户机的替代方法，直接将消息发送到 MQTT v3 客户机。MQTT V3 客户机应用程序无法直接发送消息，但其他应用程序 (例如 IBM MQ 应用程序) 可以发送消息。

IBM MQ 应用程序必须知道 MQTT v3 客户机的 `ClientIdentifier`。由于 MQTT v3 客户机没有队列，因此会将目标队列名称作为主题名称传递到 MQTT v3 应用程序客户机 `messageArrived` 方法。例如，在 MQI 程序中，使用客户机作为 `ObjectQmgr` 名称创建对象描述符：

```
MQOD.ObjectQmgrName = ClientIdentifier ;
MQOD.ObjectName = name ;
```

图 24: 用于将消息发送到 MQTT v3 客户机目标的 MQI 对象描述符

如果应用程序是使用 JMS 编写的，请创建点到点目标；例如：

```
javax.jms.Destination jmsDestination =
(javax.jms.Destination)jmsFactory.createQueue
("queue://ClientIdentifier/name");
```

图 25: 用于将消息发送到 MQTT v3 客户机的 JMS 目标

要向 MQTT 客户机发送非请求消息，请使用远程队列定义。远程队列管理器名称必须解析为客户机的 `ClientIdentifier`。传输队列必须解析为 `SYSTEM.MQTT.TRANSMIT.QUEUE`；请参阅第 188 页的表 15。远程队列名称可以是任何内容。客户机将其作为主题字符串接收。

| Input | | Output | | |
|-----------|--------------|----------------|-------------------------------|-----------------------------------------|
| 队列名称 | 队列管理器名称 | 队列名称 | 队列管理器名称 | 传输队列 |
| 远程队列定义的名称 | 空白或本地队列管理器名称 | 用作主题字符串的远程队列名称 | <code>ClientIdentifier</code> | <code>SYSTEM.MQTT.TRANSMIT.QUEUE</code> |

如果客户机已连接，那么会将消息直接发送到 MQTT 客户机，这将调用 `messageArrived` 方法；请参阅 `messageArrived` 方法。

如果客户机已与持久会话断开连接，那么消息将存储在 `SYSTEM.MQTT.TRANSMIT.QUEUE` 中；请参阅 `MQTT` 无状态和有状态会话。当客户机再次重新连接到会话时，会将其转发到客户机。

如果发送非持久消息，那么将通过至多一次服务质量 `QoS=0` 将其发送到客户机。如果将持久消息直接发送到客户机，那么缺省情况下，将随正好一次服务质量 `QoS=2` 一起发送该消息。由于客户机可能没有持久性机制，因此客户机可以降低直接发送的消息所接受的服务质量。要降低直接发送到客户机的消息的服务质量，请预订主题 `DEFAULT.QoS`。指定客户机可支持的最大服务质量。

Windows

Linux

AIX

MQTT 客户机标识、授权和认证

遥测 (MQXR) 服务使用 MQTT 通道代表 MQTT 客户机发布或预订 IBM MQ 主题。IBM MQ 管理员配置用于 IBM MQ 授权的 MQTT 通道身份。管理员可以为通道定义一个公共身份，也可以使用与此通道相连的客户机的用户名或客户机标识。

遥测 (MQXR) 服务可以使用客户机所提供的 `Username` 或者使用客户机证书来认证客户机。使用客户机所提供的密码来认证用户名。

总结：客户机标识就是选择的客户机身份。根据上下文不同，可通过客户机标识、用户名、由管理员创建的公共客户机身份或者客户机证书来标识客户机。用于检查真实性的客户机标识不必是用于授权的同一标识。

MQTT 客户机程序设置使用 MQTT 通道发送到服务器的用户名和密码。他们还可以设置加密和认证连接所需的 TLS 属性。管理员决定是否认证 MQTT 通道以及如何认证通道。

要授权 MQTT 客户机访问 IBM MQ 对象，请授权 `ClientIdentifier` 或客户机的用户名，或授权公共客户机身份。要允许客户机连接到 IBM MQ，请对用户名进行认证或使用客户机证书。配置 JAAS 以认证用户名，并配置 TLS 以认证客户机证书。

如果在客户机上设置了密码，请使用 VPN 对连接进行加密，或者将 MQTT 通道配置为使用 TLS，以保持密码为私有。

难以管理客户机证书。正因为如此，如果可以接受进行密码认证存在的风险，那么通常使用密码认证对客户机进行认证。

如果有一种安全的方法来管理和存储客户机证书，那么可以依赖于证书认证来实现。但是，在使用 Telemetry 的各种类型的环境中，很少能够安全地管理证书。然而，通过在服务器中对客户机密码进行认证，可以作为“使用客户机证书对客户机进行认证”的补充。由于还存在其他复杂性，因此，仅限于对高度敏感的应用程序使用客户机证书。使用两种形式的认证被称为双因素认证。您必须知道其中一个因素（例如，密码），并且具有另一个因素（例如，证书）。

在高度敏感的应用程序中（例如，chip-and-pin 设备），在制造期间将锁定此设备，以防止篡改内部硬件和软件。会将一个可信的、具有时间限制的客户机证书复制到此设备中。将此设备部署到要使用它的位置。每当使用此设备时，都会使用密码或者来自智能卡的另一个证书对它执行进一步认证。

Windows Linux AIX MQTT 客户机身份和权限

使用客户机标识 `Username` 或公共客户机身份来授权访问 IBM MQ 对象。

IBM MQ 管理员有三个选项用于选择 MQTT 通道的身份。管理员在定义或修改客户机所使用的 MQTT 通道时进行选择。身份用来授予对于 IBM MQ 主题访问权的访问权。按以下顺序进行选择：

1. 客户机标识 (请参阅 [USECLNTID](#))。
2. 管理员为通道 (通道的 `MCAUSER`) 提供的身份。请参阅 [MCAUSER](#)。
3. 如果上述两个选项都不适用，那么从 MQTT 客户机传递的用户名 (`Username` 是 `MqttConnectOptions` 类的属性)。它必须在客户机连接至服务之前进行设置。其缺省值为空。

避免故障: 此后，此进程选择的身份被称为客户机的 `MCAUSER`，例如由 `DISPLAY CHSTATUS (MQTT)` 命令引用。请注意，这不一定与选项 (2) 中引用的通道的 `MCAUSER` 相同。

使用 IBM MQ `setmqaut` 命令来选择哪些对象以及哪些操作有权由与 MQTT 通道关联的身份使用。例如，以下代码授权由队列管理器 `QM1` 的管理员提供的通道身份 `MQTTClient`：

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

Windows Linux AIX 授权 MQTT 客户机访问 IBM MQ 对象

执行以下步骤以授权 MQTT 客户机发布和预订 IBM MQ 对象。这些步骤遵循四种备用访问控制模式。

开始之前

通过在 MQTT 客户机连接到遥测通道时为其分配身份，这些客户机有权访问 IBM MQ 中的对象。IBM MQ 管理员使用 IBM MQ Explorer 配置遥测通道，为客户机提供三种类型的身份之一：

1. `ClientIdentifier`
2. 用户名
3. 管理员分配给通道的名称。

无论使用哪种类型，身份都必须由已安装的授权服务将 IBM MQ 定义为主体。Windows 或 Linux 上的缺省授权服务称为对象权限管理器 (OAM)。如果您正在使用 OAM，那么必须将身份定义为用户标识。

使用身份为客户机或客户机集合授予发布或预订 IBM MQ 中定义的主题的许可权。如果 MQTT 客户机已预订主题，请使用该身份为其授予接收生成的发布的许可权。

很难管理具有数万个 MQTT 客户机的系统，每个客户机都需要单独的访问许可权。一种解决方案是定义公共身份，并将单个 MQTT 客户机与其中一个公共身份相关联。定义所需数量的公共身份，以定义不同的许可权组合。另一个解决方案是编写您自己的授权服务，该服务可以比操作系统更轻松处理数以千计的用户。

您可以使用 OAM 以两种方式将 MQTT 客户机组合为公共身份：

1. 定义多个遥测通道，每个通道具有管理员使用 IBM MQ Explorer 分配的不同用户标识。使用不同 TCP/IP 端口号进行连接的客户机与不同的遥测通道相关联，并分配不同的身份。
2. 定义单个遥测通道，但让每个客户机从一小组用户标识中选择一个 **用户名**。管理员配置遥测通道以选择客户机 **用户名** 作为其身份。

在此任务中，遥测通道的标识称为 *mqttUser*，而不考虑其设置方式。如果客户机集合使用不同的身份，请使用多个 *mqttUsers*，每个客户机集合一个。由于任务使用 OAM，因此每个 *mqttUser* 都必须为用户标识。

关于此任务

在此任务中，您可以选择可根据特定需求定制四个访问控制模式。这些模式的访问控制粒度不同。

- [第 190 页的『无访问控制』](#)
- [第 190 页的『粗颗粒度访问控制』](#)
- [第 190 页的『中颗粒度访问控制』](#)
- [第 191 页的『细颗粒度访问控制』](#)

模型的结果是分配 *mqttUsers* 组许可权以发布和预订 IBM MQ，并从 IBM MQ 接收发布。

无访问控制

MQTT 客户机被授予 IBM MQ 管理权限，并且可以对任何对象执行任何操作。

过程

1. 创建用户标识 *mqttUser* 以充当所有 MQTT 客户机的身份。
2. 将 *mqttUser* 添加到 *qmqm* 组；请参阅 [向 Windows 上的组添加用户](#) 或 [向 Linux 上的组添加用户](#)

粗颗粒度访问控制

MQTT 客户机具有发布和预订以及向 MQTT 客户机发送消息的权限。他们无权执行其他操作或访问其他对象。

过程

1. 创建用户标识 *mqttUser* 以充当所有 MQTT 客户机的身份。
2. 授权 *mqttUser* 发布和预订所有主题以及向 MQTT 客户机发送发布。

```
setmqaut -m qMgr -t topic -n SYSTEM.BASE.TOPIC -p mqttUser -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqttUser -all +put
```

中颗粒度访问控制

MQTT 客户机分为不同的组以发布和预订不同的主题集，并将消息发送到 MQTT 客户机。

过程

1. 在发布/预订主题树中创建多个用户标识 *mqttUsers* 和多个管理主题。
2. 将不同的 *mqttUsers* 授权给不同的主题。

```
setmqaut -m qMgr -t topic -n topic1 -p mqttUserA -all +pub +sub
setmqaut -m qMgr -t topic -n topic2 -p mqttUserB -all +pub +sub
```

3. 创建组 *mqtt*，并将所有 *mqttUsers* 添加到该组。

4. 授权 *mqtt* 将主题发送到 MQTT 客户机。

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

细颗粒度访问控制

MQTT 客户机已合并到现有访问控制系统中，该系统授权组对对象执行操作。

关于此任务

根据用户标识所需的权限，将该用户标识分配给一个或多个操作系统组。如果 IBM MQ 应用程序正在发布和预订与 MQTT 客户机相同的主题空间，请使用此模型。这些组称为 Publish X，Subscribe Y 和 mqtt

Publish X

Publish X 组的成员可以发布到 *topicX*。

Subscribe Y

Subscribe Y 组的成员可以预订 *topicY*。

mqtt

mqtt 组的成员可以向 MQTT 客户机发送发布。

过程

1. 创建分配给发布/预订主题树中的多个管理主题的多个组 Publish X 和 Subscribe Y。
2. 创建组 *mqtt*。
3. 创建多个用户标识 *mqttUsers*，并将这些用户添加到任何组，具体取决于他们有权执行的操作。
4. 将不同的 Publish X 和 Subscribe X 组授权给不同的主题，并授权 *mqtt* 组向 MQTT 客户机发送消息。

```
setmqaut -m qMgr -t topic -n topic1 -p Publish X -all +pub
setmqaut -m qMgr -t topic -n topic1 -p Subscribe X -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

Windows

Linux

AIX

MQTT 使用密码进行客户机认证

使用客户机密码来认证用户名。使用与用来授权客户机发布和预订主题的身份不同的身份对客户机进行认证。

遥测 (MQXR) 服务使用 JAAS 来认证客户机 Username。JAAS 使用 MQTT 客户机提供的密码。

IBM MQ 管理员通过配置客户机连接到的 MQTT 通道来决定是认证用户名，还是根本不认证。可以将客户机分配给不同的通道，并且可以配置每个通道以采用不同方式对它的客户机进行认证。通过使用 JAAS，可以配置哪些方法必须对客户机进行认证，哪些方法可以有选择地对客户机进行认证。

为认证选择的身份并不会影响为授权选择的身份。为了便于管理，您可能想为授权设置一个公共身份，但是对每个用户进行认证以使用该身份。以下过程概述了对各个用户进行认证以使用公共身份时要执行的步骤：

1. IBM MQ 管理员使用 IBM MQ Explorer 将 MQTT 通道身份设置为任何名称，例如 *MQTTClientUser*。
2. IBM MQ 管理员授权 *MQTTClient* 发布和预订任何主题：

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

3. MQTT 客户机应用程序开发者在连接到服务器之前创建 *MqttConnectOptions* 对象并设置用户名和密码。
4. 安全性开发者创建一个 JAAS LoginModule，以使用密码来认证用户名，并将此模块包括在 JAAS 配置文件。
5. IBM MQ 管理员配置 MQTT 通道以使用 JAAS 来认证客户机的 Username。

MQTT 客户机与队列管理器之间的连接始终由 MQTT 客户机发起。MQTT 客户机始终是 SSL 客户机。服务器的客户机认证和 MQTT 客户机的服务器认证都是可选的。

通过向客户机提供专用签名数字证书，可以向 WebSphere MQ 认证 MQTT 客户机。WebSphere MQ 管理员可以强制 MQTT 客户机使用 TLS 向队列管理器认证它们本身。只能在相互认证的过程中请求客户机认证。

除了使用 SSL 以外，某些种类的虚拟专用网 (VPN) (例如，IPSec) 将对 TCP/IP 连接的端点进行认证。VPN 将对流经网络的每个 IP 包进行加密。一旦建立了这样一个 VPN 连接，您就已经建立了一个可信网络。您可以使用 VPN 网络上的 TCP/IP 将 MQTT 客户机连接至遥测通道。

使用 TLS 的客户机认证依赖于具有私钥的客户机。私钥是客户机的专用密钥 (对于自签名证书来说)，或者是由认证中心提供的密钥。密钥用来为客户机的数字证书签名。拥有相应的公用密钥的任何人都可以验证数字证书。证书可能是可信的，如果它们是链式证书，那么追溯整个证书链以找到可信根证书。客户机验证将客户机所提供的证书链中的所有证书发送至服务器。服务器将检查证书链，直到找到它信任的证书为止。可信证书是根据自签名证书生成的公用证书，或者是通常由认证中心发放的根证书。作为最后的可选步骤，可以将可信证书与“实时”的证书撤销列表进行比较。

可信证书可能已由认证中心颁发并已经包含在 JRE 证书库中。它可以是自签名证书，也可以是已经作为可信证书添加至遥测通道密钥库的任何证书。

注：遥测通道具有组合密钥库/信任密钥库，该库中包含一个或多个遥测通道的专用密钥以及对客户机进行认证所需的所有公用证书。由于 SSL 通道必须具有密钥库且它是与通道信任库相同的文件，因此绝不会引用 JRE 证书库。其含义为，如果客户机认证需要认证中心根证书，那么即使该证书已存在于 JRE 证书库中，也必须将其放入该通道的密钥库中。绝不会引用 JRE 证书库。

请考虑进行客户机认证是为了对付威胁，以及客户机在服务器对付威胁时所起的作用。只是对客户机证书进行认证还不足以防止对系统进行未经授权的访问。如果其他人控制了客户机设备，那么该客户机设备不一定采用证书拥有者的权限来运作。决不能依赖单一防护措施来对付不希望出现的攻击。至少应使用双重认证方法，并使用私有信息来补充证书的内容。例如，使用 JAAS，以及使用由服务器发放的密码来认证客户机。

客户机证书面对的主要威胁是被不适当的人拥有。证书保存在客户机上的一个受密码保护的密钥库中。它是如何保存在密钥库中的？MQTT 客户机如何将密码保存到密钥库中？密码保护的安全程度如何？遥测设备通常容易被移除，然后可以私下修改。设备硬件必须防篡改吗？分发和保护客户机端证书被认为是一项艰巨的任务；它被称为密钥管理问题。

次要的威胁是，无意识地误用了设备来访问服务器。例如，如果篡改了 MQTT 应用程序，那么有可能在使用已认证的客户机身份的服务器配置中使用薄弱环节。

要使用 SSL 对 MQTT 客户机进行认证，请配置遥测通道和此客户机。

相关概念

第 192 页的『使用 TLS 进行 MQTT 客户机认证的遥测通道配置』

IBM MQ 管理员在服务器上配置遥测通道。每个通道被配置为接受不同端口号上的 TCP/IP 连接。配置了 TLS 通道，以对密钥文件进行受口令保护的访问。如果没有为 TLS 通道定义口令或密钥文件，那么此通道将不接受 TLS 连接。

相关信息

使用 TLS 进行客户机认证的 MQTT 客户机配置

IBM MQ 管理员在服务器上配置遥测通道。每个通道被配置为接受不同端口号上的 TCP/IP 连接。配置了 TLS 通道，以对密钥文件进行受口令保护的访问。如果没有为 TLS 通道定义口令或密钥文件，那么此通道将不接受 TLS 连接。

将 TLS 遥测通道的属性 `com.ibm.mq.MQTT.ClientAuth` 设置为 `REQUIRED`，以强制所有在该通道上连接的客户机提供已验证数字证书的证明。使用来自认证中心的证书对客户机证书进行认证，从而生成可信根证书。如果客户机证书是自签名证书，或者由来自认证中心的证书签署，那么客户机或认证中心的公用签名证书必须安全地存储在服务器上。

将公用签名的客户机证书或来自认证中心的证书放在遥测通道密钥库中。在服务器上，公用签名证书与专用签名证书存储在同一密钥文件中，而不是存储在单独的信任库中。

服务器使用其拥有的所有公用证书和密码套件来验证其发送的任何客户机证书的签名。服务器验证密钥链。可以配置队列管理器以根据证书撤销列表测试证书。队列管理器撤销名称列表属性为 SSLCRLNL。

如果客户机发送的任何证书由服务器密钥库中的证书验证，那么将对客户机进行认证。

IBM MQ 管理员可以配置相同的遥测通道，以使用 JAAS 来检查具有客户机密码的客户机的 `UserName` 或 `ClientIdentifier`。

您可以将同一密钥库用于多个遥测通道。

验证设备上受密码保护的客户机密钥库中的至少一个数字证书会向服务器认证客户机。数字证书仅用于 IBM MQ 的认证。它不用于验证客户机的 TCP/IP 地址，也不用于设置客户机的身份以进行授权或记帐。服务器采用的客户机的身份是客户机的 `用户名` 或 `ClientIdentifier`，或者是由 IBM MQ 管理员创建的身份。

您还可以使用 TLS 密码套件进行客户机认证。如果您计划使用 SHA-2 密码套件，请参阅第 194 页的『关于将 SHA-2 密码套件用于 MQTT 通道的系统需求』。

相关概念

第 193 页的『使用 TLS 进行通道认证的遥测通道配置』

IBM MQ 管理员在服务器上配置遥测通道。每个通道被配置为接受不同端口号上的 TCP/IP 连接。配置了 TLS 通道，以对密钥文件进行受口令保护的访问。如果没有为 TLS 通道定义口令或密钥文件，那么此通道将不接受 TLS 连接。

相关信息

[DEFINE CHANNEL \(MQTT\)](#)

[ALTER CHANNEL \(MQTT\)](#)

[CipherSpecs 和 CipherSuites](#)

Windows

Linux

AIX

使用 TLS 的遥测通道认证

MQTT 客户机与队列管理器之间的连接始终由 MQTT 客户机发起。MQTT 客户机始终是 SSL 客户机。服务器的客户机认证和 MQTT 客户机的服务器认证都是可选的。

除非客户机被配置为使用支持匿名连接的 CipherSpec，否则客户机始终会尝试对服务器进行认证。如果认证失败，那么不会建立连接。

除了使用 SSL 以外，某些种类的虚拟专用网 (VPN)（例如，IPSec）将对 TCP/IP 连接的端点进行认证。VPN 将对流经网络的每个 IP 包进行加密。一旦建立了这样一个 VPN 连接，您就已经建立了一个可信网络。您可以使用 VPN 网络上的 TCP/IP 将 MQTT 客户机连接至遥测通道。

使用 SSL 的服务器认证将对您要发送的保密信息发送至的服务器进行认证。客户机根据其信任库或 JRE `cacerts` 库中的证书执行与从服务器发送的证书相匹配的检查。

JRE 证书库是 JKS 文件 `cacerts`。它位于 `JRE InstallPath\lib\security\` 中。它是使用缺省密码 `changeit` 进行安装的。您可以将信任的证书存储在 JRE 证书库或客户机信任库中。不能同时使用这两个库。如果要使客户机信任的公用证书独立于其他 Java 应用程序使用的证书，请使用客户机信任库。如果要针对客户机上正在运行的所有 Java 应用程序使用公共证书库，请使用 JRE 证书库。如果确定要使用 JRE 证书库，请查看其所含证书以确保您信任这些证书。

可以通过提供另外的信任提供程序来修改 JSSE 配置。可以定制信任提供程序以对证书执行另外的检查。在一些使用了 MQTT 客户机的 OGSi 环境中，该环境提供了不同的信任提供程序。

要使用 TLS 对遥测通道进行认证，请配置服务器和客户机。

Windows

Linux

AIX

使用 TLS 进行通道认证的遥测通道配置

IBM MQ 管理员在服务器上配置遥测通道。每个通道被配置为接受不同端口号上的 TCP/IP 连接。配置了 TLS 通道，以对密钥文件进行受口令保护的访问。如果没有为 TLS 通道定义口令或密钥文件，那么此通道将不接受 TLS 连接。

将使用其专用密钥签名的服务器的数字证书存储在遥测通道将在服务器上使用的密钥库中。如果要将密钥链传输到客户机，请将其密钥链中的任何证书存储在密钥库中。使用 IBM MQ 资源管理器配置遥测通道以使用 TLS。为其提供密钥库的路径以及用于访问密钥库的口令。如果未设置通道的 TCP/IP 端口号，那么 TLS 遥测通道端口号缺省为 8883。

您还可以使用 TLS 密码套件进行通道认证。如果您计划使用 SHA-2 密码套件，请参阅第 194 页的『关于将 SHA-2 密码套件用于 MQTT 通道的系统需求』。

相关概念

第 192 页的『使用 TLS 进行 MQTT 客户机认证的遥测通道配置』

IBM MQ 管理员在服务器上配置遥测通道。每个通道被配置为接受不同端口号上的 TCP/IP 连接。配置了 TLS 通道，以对密钥文件进行受口令保护的访问。如果没有为 TLS 通道定义口令或密钥文件，那么此通道将不接受 TLS 连接。

相关信息

[DEFINE CHANNEL \(MQTT\)](#)

[ALTER CHANNEL \(MQTT\)](#)

[CipherSpecs 和 CipherSuites](#)

Windows Linux AIX 关于将 SHA-2 密码套件用于 MQTT 通道的系统需求

如果您使用的 Java 版本支持 SHA-2 密码套件，那么可以使用这些套件来保护 MQTT（遥测）通道和客户机应用程序。

对于包含遥测 (MQXR) 服务的 IBM MQ 8.0，最低 Java 版本是 Java 7（从 IBM SR6 起）。从 IBM SR4 起，缺省情况下在 Java 7 中支持 SHA-2 密码套件。因此，可以使用具有遥测 (MQXR) 服务的 SHA-2 密码套件来保护 MQTT（遥测）通道。

如果您是使用其他 JRE 来运行 MQTT 客户机，那么需要确保它还支持 SHA-2 密码套件。

相关概念

第 193 页的『使用 TLS 进行通道认证的遥测通道配置』

IBM MQ 管理员在服务器上配置遥测通道。每个通道被配置为接受不同端口号上的 TCP/IP 连接。配置了 TLS 通道，以对密钥文件进行受口令保护的访问。如果没有为 TLS 通道定义口令或密钥文件，那么此通道将不接受 TLS 连接。

相关信息

[遥测 \(MQXR\) 服务](#)

[DEFINE CHANNEL \(MQTT\)](#)

[ALTER CHANNEL \(MQTT\)](#)

Windows Linux AIX 发布在遥测通道上的隐私

通过使用 TLS 对通过连接的传输进行加密，可保护通过遥测通道向任一方向发送的 MQTT 发布的隐私。

连接到遥测通道的 MQTT 客户机使用 TLS 来保护使用对称密钥密码术在通道上传输的发布的隐私。由于未对端点进行认证，因此您不能只信任通道加密。应将保护隐私与服务器认证或相互认证结合起来。

除了使用 SSL 以外，某些种类的虚拟专用网 (VPN)（例如，IPSec）将对 TCP/IP 连接的端点进行认证。VPN 将对流经网络的每个 IP 包进行加密。一旦建立了这样一个 VPN 连接，您就已经建立了一个可信网络。您可以使用 VPN 网络上的 TCP/IP 将 MQTT 客户机连接至遥测通道。

对于典型配置（它会对通道进行加密和对服务器进行认证），请查阅第 193 页的『使用 TLS 的遥测通道认证』。

在不认证服务器的情况下对 TLS 连接进行加密会使连接暴露于中间人攻击。尽管可以保护您交换的信息不被窃听，但是您并不知道是与谁在交换信息。除非您控制整个网络，否则其他人就有可能拦截您的 IP 传输并且伪装成端点。

您可以使用支持匿名 TLS 的 Diffie-Hellman 密钥交换 CipherSpec 来创建加密 TLS 连接，而无需认证服务器。在客户机和服务器之间共享并用于加密 TLS 传输的主密钥是在不交换专用签名服务器证书的情况下建立的。

由于匿名连接不安全，因此大多数 TLS 实现都不会缺省使用匿名 CipherSpecs。如果遥测通道接受针对 TLS 连接的客户机请求，那么该通道必须具有受口令保护的密钥库。缺省情况下，由于 TLS 实现不使用匿名 CipherSpecs，因此密钥库必须包含客户机可认证的专用签名证书。

如果您使用匿名 CipherSpec，那么服务器密钥库必须存在，但是它不需要包含任何私下签名的证书。

另一种建立加密连接的方法是，将客户机中的信任提供程序替换为您自己的实现。您的信任提供程序将不对服务器证书进行认证，但是连接将加密。



注意：将 TLS 与 MQTT 配合使用时，可以使用大型消息，但是这样做可能会影响性能。MQTT 针对处理小消息进行了优化（通常大小在 1KB 到 1MB 之间）。

Windows

Linux

AIX

MQTT Java 客户机和遥测通道的 TLS 配置

配置 TLS 以认证遥测通道和 MQTT Java 客户机，并对它们之间的消息传输进行加密。MQTT Java 客户机使用 Java 安全套接字扩展 (JSSE) 通过 TLS 连接遥测通道。除了使用 SSL 以外，某些种类的虚拟专用网 (VPN)（例如，IPSec）将对 TCP/IP 连接的端点进行认证。VPN 将对流经网络的每个 IP 包进行加密。一旦建立了这样一个 VPN 连接，您就已经建立了一个可信网络。您可以使用 VPN 网络上的 TCP/IP 将 MQTT 客户机连接至遥测通道。

您可以配置 Java MQTT 客户机与遥测通道之间的连接，以使用基于 TCP/IP 的 TLS 协议。受保护的内容取决于您如何配置 TLS 以使用 JSSE。从最安全的配置开始，您可以配置三种不同级别的安全性：

1. 仅允许可信 MQTT 客户机进行连接。仅将 MQTT 客户机连接到可信遥测通道。对客户机与队列管理器之间的消息进行加密；请参阅第 192 页的『使用 TLS 的 MQTT 客户机认证』。
2. 仅将 MQTT 客户机连接到可信遥测通道。对客户机与队列管理器之间传递的消息进行加密；请参阅第 193 页的『使用 TLS 的遥测通道认证』。
3. 对客户机与队列管理器之间传递的消息进行加密；请参阅第 194 页的『发布在遥测通道上的隐私』。

JSSE 配置参数

修改 JSSE 参数以更改 TLS 连接的配置方式。JSSE 配置参数分为三个集合：

1. [MQ Telemetry 通道](#)
2. [MQTT Java 客户机](#)
3. [JRE](#)

使用 IBM MQ Explorer 配置遥测通道参数。在 `MqttConnectionOptions.SSLProperties` 属性中设置 MQTT Java 客户机参数。通过编辑客户机和服务器中的 JRE 安全性目录中的文件来修改 JRE 安全性参数。

MQ Telemetry 通道

使用 IBM MQ Explorer 设置所有遥测通道 TLS 参数。

ChannelName

`ChannelName` 是所有通道的一个必需参数。

通道名称标识与特定端口号相关联的通道。用于帮助您管理 MQTT 客户机集的名称通道。

PortNumber

`PortNumber` 是所有通道的一个可选参数。对于 TCP 通道，缺省值为 1883，对于 TLS 通道，缺省值为 8883。

TCP/IP 端口号与此通道相关联。MQTT 客户机通过指定为通道定义的端口来连接到通道。如果通道具有 TLS 属性，那么客户机必须使用 TLS 协议进行连接；例如：

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileName

KeyFileName 是 TLS 通道的必需参数。对于 TCP 通道，必须省略此参数。

KeyFileName 是包含您提供的数字证书的 Java 密钥库的路径。使用 JKS、JCEKS 或 PKCS12 作为服务器上的密钥库类型。

可使用以下某个文件扩展名来确定密钥库类型：

- .jks
- .jceks
- .p12
- .pkcs12

具有其他任何文件扩展名的密钥库均视为 JKS 密钥库。

您可以将服务器上的一类密钥库与客户机上其他类型的密钥库相合并。

将服务器的私有证书放在密钥库中。证书被称为服务器证书。证书可以是自签名证书，也可以是签署机构所签署的证书链的一部分。

如果您要使用证书链，请将关联的证书放在服务器密钥库中。

服务器证书及其证书链中的任何证书被发送至客户机以认证服务器的身份。

如果您已将 ClientAuth 设置为 Required，那么密钥库中必须包含对客户机进行认证时所需要的任何证书。客户机发送一个自签名证书，或一个证书链，这样会对照密钥库中的证书，通过首次验证此材料，对客户机进行认证。通过使用证书链，一个证书可以验证许多客户机，即使向它们颁发了不同的客户机证书，也是如此。

PassPhrase

PassPhrase 是 TLS 通道的必需参数。对于 TCP 通道，必须省略此参数。

口令用来保护密钥库。

ClientAuth

ClientAuth 是可选的 TLS 参数。它缺省设置为不进行客户机认证。对于 TCP 通道，必须省略此参数。

如果您想要遥测 (MQXR) 服务在允许客户机连接至遥测通道之前认证客户机，那么请设置 ClientAuth。

如果设置 ClientAuth，那么客户机必须使用 TLS 连接到服务器，并对服务器进行认证。作为对设置 ClientAuth 的响应，客户机将它的数字证书及其密钥库中的任何其他证书发送至服务器。它的数字证书被称为客户机证书。将针对保存在通道密钥库以及 JRE cacerts 库中的证书来认证这些证书。

CipherSuite

CipherSuite 是可选 TLS 参数。它缺省设置为尝试所有已启用的 CipherSpec。对于 TCP 通道，必须省略此参数。

如果要使用特定 CipherSpec，请将 CipherSuite 设置为必须用于建立 TLS 连接的 CipherSpec 的名称。

遥测服务和 MQTT 客户机通过在每一端启用的所有 CipherSpecs 协商公共 CipherSpec。如果在连接的其中一端或者两端指定了一个特定 CipherSpec，那么此 CipherSpec 必须与另一端的 CipherSpec 相匹配。

通过将其提供程序添加至 JSSE 来安装其他密码。

联邦信息处理标准 (FIPS)

FIPS 是一个可选设置。缺省情况下，不会设置此项。

在队列管理器的“属性”面板中设置 SSLFIPS，或者使用 **runmqsc** 来设置。SSLFIPS 指定是否仅使用经过 FIPS 证明的算法。

撤销名称列表

“撤销名称列表”是一个可选设置。缺省情况下，不会设置此项。

在队列管理器的“属性”面板中设置 `SSLCRLNL`，或者使用 `runmqsc` 来设置。`SSLCRLNL` 指定用来提供证书撤销位置的认证信息对象的名称列表。

不使用其他用于设置 TLS 属性的队列管理器参数。

MQTT Java 客户机

在 `MqttConnectionOptions.SSLProperties` 中设置 Java 客户机的 TLS 属性; 例如:

```
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.keyStoreType", "JKS");
com.ibm.micro.client.mqttv3.MqttConnectOptions conOptions = new MqttConnectOptions();
conOptions.setSSLProperties(sslClientProperties);
```

`MqttConnectOptions` 类中描述了特定属性的名称和值。有关 MQTT 客户机库的客户机 API 文档的链接，请参阅 [MQTT 客户机编程参考](#)。

协议

`Protocol` 是可选的。

协议是在与遥测服务器协商时选择的。如果您需要特定协议，那么可以选择一种协议。如果遥测服务器不支持此协议，那么连接将失败。

ContextProvider

`ContextProvider` 是可选的。

KeyStore

`KeyStore` 是可选的。如果在服务器中设置了 `ClientAuth` 以强制对客户机进行认证，那么请配置此项。

将使用客户机的专用密钥签名的数字证书放入密钥库中。指定密钥库的路径和密码。类型和提供程序是可选的。`JKS` 是缺省类型，`IBMJCE` 是缺省提供程序。

指定另外的密钥库提供程序，以引用用于添加新的密钥库提供程序的类。传递密钥库提供程序所使用的算法的名称，以通过设置密钥管理器名称来实例化 `KeyManagerFactory`。

TrustStore

`TrustStore` 是可选的。您可以将您信任的所有证书都放在 `JRE cacerts` 存储库中。

如果要为客户机设置不同的信任库，请配置信任库。如果服务器正在使用已在 `cacerts` 中存储其根证书的众所周知的认证中心发放的证书，那么可能无法配置信任库。

将服务器的公开签名的证书或者根证书添加至信任库，并指定信任库的路径和密码。`JKS` 是缺省类型，`IBMJCE` 是缺省提供程序。

指定另外的信任库提供程序，以引用用于添加新的信任库提供程序的类。传递信任库提供程序所使用的算法的名称，以通过设置信任管理器名称来实例化 `TrustManagerFactory`。

JRE

在 JRE 中配置了影响客户机和服务器上 TLS 行为的 Java 安全性的其他方面。Windows 上的配置文件位于 `Java Installation Directory\jre\lib\security` 中。如果使用 IBM MQ 随附的 JRE，那么该路径如下表中所示:

| 平台 | 菲尔帕特 |
|---------|---------------------------------------------------------------|
| Windows | <code>WMQ Installation Directory\java\jre\lib\security</code> |

| 表 16: JRE TLS 配置文件的文件路径 (按平台排列) (继续) | |
|--------------------------------------|----------------------------------------------------------------|
| 平台 | 菲尔帕特 |
| UNIX and Linux 平台 | <i>WMQ Installation Directory</i> /java/jre64/jre/lib/security |

众所周知的认证中心

`cacerts` 文件中包含众所周知的认证中心的根证书。除非您指定信任库，否则缺省情况下将使用 `cacerts`。如果使用 `cacerts` 存储库或未提供信任库，那么必须查看和编辑 `cacerts` 中的签署者列表以满足安全性需求。

您可以使用运行 IBM Key Management 实用程序的 IBM MQ 命令 `strmqikm` 来打开 `cacerts`。使用密码 `changeit` 将 `cacerts` 作为 JKS 文件打开。修改密码以保护此文件的安全。

配置安全性类

使用 `java.security` 文件来注册其他安全性提供程序和其他缺省安全性属性。

许可权

使用 `java.policy` 文件来修改为资源授予的许可权。`javaws.policy` 为 `javaws.jar` 授予许可权

加密强度

某些 JRE 提供了强度降低的加密。如果您无法将密钥导入密钥库中，可能是由于加密强度降低引起的。请尝试使用 `strmqikm` 命令启动 `ikeyman`，或者从 [IBM Developer Kit, Security information](#) 下载强大但有限的管辖区域文件。

要点: 您的产地国对加密软件的进口、拥有、使用或再次出口到其他国家可能有一些限制。在下载或使用不受限制的策略文件之前，必须针对您所在国家或地区的法律进行检查。检查相应的规章以及对加密软件进行进口、拥有、使用和再次出口的相关政策，从而确定是否允许下载或使用这些文件。

修改信任提供程序以允许客户机连接至任何服务器

此示例说明如何添加信任提供程序并从 MQTT 客户机代码对其进行引用。该示例对客户机或服务器不执行认证。将对生成的 TLS 连接进行加密，而不进行认证。

第 198 页的图 26 中的代码片段为 MQTT 客户机设置 `AcceptAllProviders` 信任提供程序和信任管理器。

```
java.security.Security.addProvider(new AcceptAllProvider());
java.util.Properties sslClientProperties = new Properties();
sslClientProperties.setProperty("com.ibm.ssl.trustManager", "TrustAllCertificates");
sslClientProperties.setProperty("com.ibm.ssl.trustStoreProvider", "AcceptAllProvider");
conOptions.setSSLProperties(sslClientProperties);
```

图 26: MQTT 客户机代码片段

```
package com.ibm.mq.id;
public class AcceptAllProvider extends java.security.Provider {
private static final long serialVersionUID = 1L;
public AcceptAllProvider() {
super("AcceptAllProvider", 1.0, "Trust all X509 certificates");
put("TrustManagerFactory.TrustAllCertificates",
AcceptAllTrustManagerFactory.class.getName());
}
}
```

图 27: `AcceptAllProvider.java`

```

protected static class AcceptAllTrustManagerFactory extends
javax.net.ssl.TrustManagerFactorySpi {
public AcceptAllTrustManagerFactory() {}
protected void engineInit(java.security.KeyStore keystore) {}
protected void engineInit(
javax.net.ssl.ManagerFactoryParameters parameters) {}
protected javax.net.ssl.TrustManager[] engineGetTrustManagers() {
return new javax.net.ssl.TrustManager[] { new AcceptAllX509TrustManager() };
}
}

```

图 28: *AcceptAllTrustManagerFactory.java*

```

protected static class AcceptAllX509TrustManager implements
javax.net.ssl.X509TrustManager {
public void checkClientTrusted(
java.security.cert.X509Certificate[] certificateChain,
String authType) throws java.security.cert.CertificateException {
report("Client authtype=" + authType);
for (java.security.cert.X509Certificate certificate : certificateChain) {
report("Accepting:" + certificate);
}
}
public void checkServerTrusted(
java.security.cert.X509Certificate[] certificateChain,
String authType) throws java.security.cert.CertificateException {
report("Server authtype=" + authType);
for (java.security.cert.X509Certificate certificate : certificateChain) {
report("Accepting:" + certificate);
}
}
public java.security.cert.X509Certificate[] getAcceptedIssuers() {
return new java.security.cert.X509Certificate[0];
}
private static void report(String string) {
System.out.println(string);
}
}
}

```

图 29: *AcceptAllX509TrustManager.java*

Windows Linux AIX 遥测通道 JAAS 配置

配置 JAAS 以认证由客户机发送的用户名。

IBM MQ 管理员使用 JAAS 配置哪些 MQTT 通道需要客户机认证。指定每个要执行 JAAS 认证的通道的 JAAS 配置名称。这些通道可以全部使用同一个 JAAS 配置，也可以使用不同的 JAAS 配置。这些配置是在 *WMQData directory\qmgrs\qMgrName\mqxr\jaas.config* 中定义的。

jaas.config 文件按 JAAS 配置名称进行组织。在每个配置名称下是一个登录配置列表；请参阅第 200 页的图 30。

JAAS 提供了四个标准登录模块。标准 NT 和 UNIX 登录模块的值有限。

JndiLoginModule

针对在 JNDI (Java 命名和目录接口) 下配置的目录服务进行认证。

Krb5LoginModule

使用 Kerberos 协议进行认证。

NTLoginModule

使用当前用户的 NT 安全性信息来进行认证。

UnixLoginModule

使用当前用户的 UNIX 安全信息进行认证。

使用 *NTLoginModule* 或 *UnixLoginModule* 时迁到的问题是遥测 (MQXR) 服务使用 *mqm* 身份而不是 MQTT 通道的身份运行。*mqm* 是传递给 *NTLoginModule* 或 *UnixLoginModule* 以进行认证的身份，而不是客户机的身份。

要解决此问题，可编写您自己的登录模块，或者使用其他标准登录模块。MQ Telemetry 随附了样本 `JAASLoginModule.java`。它是 `javax.security.auth.spi.LoginModule` 接口的实现。使用它来开发您自己的认证方法。

您提供的任何新 `LoginModule` 类都必须在遥测 (MQXR) 服务的类路径上。请勿将类放在类路径中的 IBM MQ 目录中。创建您自己的目录，并定义遥测 (MQXR) 服务的整个类路径。

您可以通过在 `service.env` 文件中设置类路径来扩充遥测 (MQXR) 服务所使用的类路径。`CLASSPATH` 必须使用大写字母，`class path` 语句只能包含文字。不能在 `CLASSPATH` 中使用变量；例如，`CLASSPATH=%CLASSPATH%` 就不正确。遥测 (MQXR) 服务设置其自己的类路径。将 `service.env` 中所定义的 `CLASSPATH` 添加至此类路径。

遥测 (MQXR) 服务提供两个回调，用于返回连接到 MQTT 通道的客户机的用户名和密码。用户名和密码在 `MqttConnectOptions` 对象中设置。请参阅第 201 页的图 31 以了解如何访问用户名和密码的示例。

示例

具有一个已命名的配置 `MQXRConfig` 的 JAAS 配置文件示例。

```
MQXRConfig {
samples.JAASLoginModule required debug=true;
//com.ibm.security.auth.module.NTLoginModule required;
//com.ibm.security.auth.module.Krb5LoginModule required
//    principal=principal@your_realm
//    useDefaultCcache=TRUE
//    renewTGT=true;
//com.sun.security.auth.module.NTLoginModule required;
//com.sun.security.auth.module.UnixLoginModule required;
//com.sun.security.auth.module.Krb5LoginModule required
//    useTicketCache="true"
//    ticketCache="${user.home}/${}tickets";
};
```

图 30: 样本 `jaas.config` 文件

编码为接收 MQTT 客户机提供的用户名和密码的 JAAS 登录模块的示例。


```

public boolean login()
throws javax.security.auth.login.LoginException {
    javax.security.auth.callback.Callback[] callbacks =
    new javax.security.auth.callback.Callback[2];
    callbacks[0] = new javax.security.auth.callback.NameCallback("NameCallback");
    callbacks[1] = new javax.security.auth.callback.PasswordCallback(
    "PasswordCallback", false);
    try {
        callbackHandler.handle(callbacks);
        String username = ((javax.security.auth.callback.NameCallback) callbacks[0])
        .getName();
        char[] password = ((javax.security.auth.callback.PasswordCallback) callbacks[1])
        .getPassword();
        // Accept everything.
        if (true) {
            loggedIn = true;
        } else
        throw new javax.security.auth.login.FailedLoginException("Login failed");

        principal= new JAASPrincipal(username);

    } catch (java.io.IOException exception) {
        throw new javax.security.auth.login.LoginException(exception.toString());
    } catch (javax.security.auth.callback.UnsupportedCallbackException exception) {
        throw new javax.security.auth.login.LoginException(exception.toString());
    }

    return loggedIn;
}

```

图 31: 样本 `JAASLoginModule.Login()` 方法

相关信息

AuthCallback MQXR 类

解决问题: 遥测服务未调用 JAAS 登录模块

V 9.0.0 管理 IBM MQ Light

您可以使用 IBM MQ Explorer 或命令行来管理 MQ Light。使用资源管理器来配置通道并监视连接到 IBM MQ 的 MQ Light 客户机。使用 TLS 和 JAAS 配置 MQ Light 的安全性。

在启动前

有关在平台上安装 AMQP 的信息, 请参阅选择安装内容。使用 IBM MQ 8.0.0.4 制造更新 (而不是 8.0.0.4 修订包) 来安装 AMQP 服务组件。不能在低于 8.0.0.4 的队列管理器版本上安装 AMQP 组件。

使用 IBM MQ Explorer 进行管理

使用资源管理器来配置 AMQP 通道并监视连接到 IBM MQ 的 MQ Light 客户机。您可以使用 TLS 和 JAAS 来配置 MQ Light 的安全性。

使用命令行进行管理

您可以使用 IBM MQ `MQSC` 命令在命令行上管理 MQ Light。

V 9.0.0 查看 MQ Light 客户机正在使用的 IBM MQ 对象

您可以查看 MQ Light 客户机正在使用的不同 IBM MQ 资源, 例如连接和预订。

3. 对于每个句柄，查看打开了句柄的 MQ Light 客户机标识:

```
DISPLAY CONN(707E0A56642B0020) CLIENTID
23 : DISPLAY CONN(707E0A56642B0020) CLIENTID

AMQ8276: Display Connection details.
CONN(707E0A56642B0020)
EXTCONN(414D5143514D31202020202020202020)
TYPE(CONN)
CLIENTID(recv_8f02c9d)
DISPLAY CONN(707E0A565F290020) CLIENTID
24 : DISPLAY CONN(707E0A565F290020) CLIENTID
AMQ8276: Display Connection details.
CONN(707E0A565F290020)
EXTCONN(414D5143514D31202020202020202020)
TYPE(CONN)
CLIENTID(recv_86d8888)
```

V 9.0.0 MQ Light 客户机标识、授权和认证

与其他 IBM MQ 客户机应用程序一样，您可以通过多种方式保护 AMQP 连接。

您可以使用以下安全功能来保护与 IBM MQ 的 AMQP 连接:

- [通道认证记录](#)
- [连接认证](#)
- [通道 MCA 用户配置](#)
- [IBM MQ 权限定义](#)
- [TLS 连接](#)

从安全角度来看，建立连接包括以下两个步骤:

- 决定连接是否应继续
- 决定应用程序为以后的权限检查所采用的 IBM MQ 身份

以下信息概述了不同的 IBM MQ 配置以及 AMQP 客户机尝试建立连接时执行的步骤。并非所有 IBM MQ 配置都使用所描述的所有步骤。例如，某些配置不将 TLS 用于公司防火墙内的连接，而某些配置使用 TLS 但不使用客户机证书进行认证。许多环境不使用定制或定制 JAAS 模块。

建立连接

以下步骤描述了 AMQP 客户机建立连接时发生的情况。这些步骤确定连接是否继续以及应用程序对权限检查所采用的 IBM MQ 身份:

1. 如果客户机打开到 IBM MQ 的 TLS 连接并提供证书，那么队列管理器会尝试验证客户机证书。
2. 如果客户机提供了用户名和密码凭证，那么队列管理器会接收到 AMQP SASL 帧，并且会检查 MQ CONNAUTH 配置。
3. 检查 MQ 通道认证规则 (例如，IP 地址和 TLS 证书 DN 是否有效)
4. 除非通道认证规则另有决定，否则将声明通道 MCAUSER。
5. 如果已配置 JAAS 模块，那么将调用该模块
6. MQ CONNECT 权限检查应用于生成的 MQ 用户标识。
7. 已使用假定的 IBM MQ 身份建立连接。

发布消息

以下步骤描述 AMQP 客户机发布消息时发生的情况。这些步骤确定连接是否继续以及应用程序对权限检查所采用的 IBM MQ 身份:

1. AMQP 链接连接框架到达队列管理器。将针对连接期间建立的 MQ 用户身份检查指定主题字符串的 IBM MQ 发布权限。

2. 将消息发布到指定的主题字符串。

预订主题模式

以下步骤描述 AMQP 客户机预订主题模式时发生的情况。这些步骤确定连接是否继续以及应用程序对权限检查所采用的 IBM MQ 身份:

1. AMQP 链接连接框架到达队列管理器。将针对连接期间建立的 MQ 用户身份检查指定主题模式的 IBM MQ 预订权限。
2. 已创建预订。

V 9.0.0 MQ Light 客户机身份和权限

使用 MQ Light 客户机标识, MQ Light 用户名或在通道或通道认证规则中定义的公共客户机身份来授权访问 IBM MQ 对象。

管理员在定义或修改 AMQP 通道时, 通过配置队列管理器 CONNAUTH 设置或通过定义通道认证规则进行选择。身份用来授予对于 IBM MQ 主题的访问权。根据以下情况进行选择:

1. 通道 USECLNTID 属性。
2. 队列管理器 CONNAUTH 规则的 ADOPTCTX 属性。
3. 在通道上定义的 MCAUSER 属性。
4. 匹配通道认证规则的 USERSRC 属性。

避免故障: 此后, 此进程选择的身份被称为客户机的 MCAUSER, 例如 DISPLAY CHSTATUS (AMQP) 命令。请注意, 这不一定与选项 (2) 中引用的通道的 MCAUSER 相同。

使用 IBM MQ `setmqaut` 命令来选择哪些对象以及哪些操作有权由与 AMQP 通道关联的身份使用。例如, 以下命令授权通道标识 AMQPClient, 由队列管理器 QM1 的管理员提供:

```
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPClient -all +pub +sub
```

和

```
setmqaut -m QM1 -t qmgr -p AMQPClient -all +connect
```

V 9.0.0 MQ Light 使用密码进行客户机认证

使用客户机密码认证 MQ Light 用户名。您可以使用与用于授权客户机发布和预订主题的身份不同的身份来认证客户机。

AMQP 服务可以使用 MQ CONNAUTH 或 JAAS 来认证客户机用户名。如果配置了其中之一, 那么客户机提供的密码将由 MQ CONNAUTH 配置或 JAAS 模块验证。

以下过程概述了针对本地操作系统用户和密码对个别用户进行认证的示例步骤, 如果成功, 请采用公共身份 AMQPUser:

1. IBM MQ 管理员使用 IBM MQ Explorer 将 AMQP 通道 MCAUSER 身份设置为任何名称, 例如 AMQPUser。
2. IBM MQ 管理员授权 AMQPUser 发布和预订任何主题:

```
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPUser -all +pub +sub +connect
```

3. IBM MQ 管理员配置 IDPWOS CONNAUTH 规则以检查客户机提供的用户名和密码。CONNAUTH 规则应设置 CHCKCLNT (必需) 和 ADOPTCTX (NO)。

注: 建议您使用通道认证规则, 并将 MCAUSER 通道属性设置为没有特权的用户, 以允许对与队列管理器的连接进行更多控制。

V 9.0.0 在通道上发布隐私

通过 AMQP 通道中任一方向发送的 AMQP 发布的隐私通过使用 TLS 对通过连接的传输进行加密来进行保护。

连接到 AMQP 通道的 AMQP 客户机使用 TLS 来保护使用对称密钥密码术在通道上传输的发布的隐私。由于未对端点进行认证，因此您不能只信任通道加密。应将保护隐私与服务器认证或相互认证结合起来。

作为使用 TLS 的替代方法，某些类型的虚拟专用网 (VPN) (例如 IPsec) 对 TCP/IP 连接的端点进行认证。VPN 将对流经网络的每个 IP 包进行加密。一旦建立了这样一个 VPN 连接，您就已经建立了一个可信网络。您可以通过 VPN 网络使用 TCP/IP 将 AMQP 客户机连接到 AMQP 通道。

在不认证服务器的情况下对 TLS 连接进行加密会使连接暴露于中间人攻击。尽管可以保护您交换的信息不被窃听，但是您并不知道是与谁在交换信息。除非您控制整个网络，否则其他人就有可能拦截您的 IP 传输并且伪装成端点。

您可以使用支持匿名 TLS 的 Diffie-Hellman 密钥交换 CipherSpec 来创建加密 TLS 连接，而无需认证服务器。在客户机和服务器之间共享并用于加密 TLS 传输的主密钥是在不交换专用签名服务器证书的情况下建立的。

由于匿名连接不安全，因此大多数 TLS 实现都不会缺省使用匿名 CipherSpecs。如果 AMQP 通道接受针对 TLS 连接的客户机请求，那么该通道必须具有受口令保护的密钥库。缺省情况下，由于 TLS 实现不使用匿名 CipherSpecs，因此密钥库必须包含客户机可认证的专用签名证书。

如果您使用匿名 CipherSpec，那么服务器密钥库必须存在，但是它不需要包含任何私下签名的证书。

另一种建立加密连接的方法是，将客户机中的信任提供程序替换为您自己的实现。您的信任提供程序将不对服务器证书进行认证，但是连接将加密。

V 9.0.0 使用 TLS 配置 MQ Light 客户机

您可以配置 MQ Light 客户机以使用 TLS 来保护流经网络的数据，并认证客户机所连接到的队列管理器的身份。

要将 TLS 用于从 MQ Light 客户机到 AMQP 通道的连接，必须确保已将队列管理器配置为 TLS。[在队列管理器上配置 TLS](#) 描述了如何配置队列管理器从中读取 TLS 证书的密钥库。

使用密钥库配置队列管理器后，必须在客户机将连接到的 AMQP 通道上配置 TLS 属性。AMQP 通道具有四个与 TLS 配置相关的属性，如下所示：

SSLCAUTH

SSLCAUTH 属性用于指定队列管理器是否应要求 MQ Light 客户机提供客户机证书以验证其身份。

SSLCIPH

SSLCIPH 属性指定通道应该用于对 TLS 流中的数据进行编码的密码。

SSLPEER

SSLPEER 属性用于指定客户机证书必须匹配的专有名称 (DN) (如果要允许连接)。

CERTLABL

CERTLABL 指定队列管理器应该向客户机提供的证书。队列管理器的密钥库可以包含多个证书。此属性允许您指定要用于与此通道的连接的证书。如果未指定 CERTLABL，那么将使用队列管理器密钥存储库中具有与队列管理器 CERTLABL 属性对应的的标签的证书。

使用 TLS 属性配置 AMQP 通道后，必须使用以下命令重新启动 AMQP 服务：

```
STOP SERVICE(SYSTEM.AMQP.SERVICE) START SERVICE(SYSTEM.AMQP.SERVICE)
```

当 MQ Light 客户机连接到受 TLS 保护的 AMQP 通道时，客户机将验证队列管理器提供的证书的身份。为此，必须使用包含队列管理器证书的信任库来配置 MQ Light 客户机。执行此操作的步骤因您所使用的 MQ Light 客户机而异。

- 有关 MQ Light Client for Node JS API 文档，请参阅 <https://www.npmjs.com/package/mqlight>
- 有关 MQ Light Client for Java API 文档，请参阅 <https://mqlight.github.io/java-mqlight/>
- 有关 MQ Light Client for Ruby 文档，请参阅 <https://www.rubydoc.info/github/mqlight/ruby-mqlight/>

- 有关 MQ Light client for Python 文档, 请参阅 <https://python-mqlight.readthedocs.org/en/latest/>

V 9.0.0 将 MQ Light 客户机与队列管理器断开连接

如果要断开 MQ Light 与队列管理器的连接, 请运行 PURGE CHANNEL 命令或停止与 MQ Light 客户机的连接。

- 运行 **PURGE CHANNEL** 命令。例如:

```
PURGE CHANNEL(MYAMQP) CLIENTID('recv_28dbb7e')
```

- 或者, 通过完成以下步骤, 停止 MQ Light 客户机用于断开客户机连接的连接:

1. 通过运行 **DISPLAY CONN** 命令来查找客户机正在使用的连接。例如:

```
DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_28dbb7e')
```

命令输出如下所示:

```
DISPLAY CONN(*) TYPE(CONN) WHERE(CLIENTID EQ 'recv_28dbb7e')
  40 : DISPLAY CONN(*) TYPE(CONN) WHERE(CLIENTID EQ 'recv_28dbb7e')
AMQ8276: Display Connection details.
CONN(707E0A565F2D0020)
EXTCONN(414D5143514D31202020202020202020)
TYPE(CONN)
CLIENTID(recv_28dbb7e)
```

2. 停止连接。例如:

```
STOP CONN(707E0A565F2D0020)
```

管理多点广播

使用此信息可了解 IBM MQ 多点广播管理任务, 例如减少多点广播消息的大小和启用数据转换。

多点广播入门

使用此信息可开始使用 IBM MQ 多点广播主题和通信信息对象。

关于此任务

IBM MQ 多点广播消息传递使用网络通过将主题映射到组地址来传递消息。以下任务是测试是否为多点广播消息传递正确配置了必需的 IP 地址和端口的快速方法。

为多点广播创建 **COMMINFO** 对象

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

使用以下命令行示例来定义用于多点广播的 COMMINFO 对象:

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

其中 MC1 是 COMMINFO 对象的名称, 组地址 是组多点广播 IP 地址或 DNS 名称, 端口号 是要传输的端口 (缺省值为 1414)。

将创建名为 MC1 的新 COMMINFO 对象; 此名称是在下一个示例中定义 TOPIC 对象时必须指定的名称。

为多点广播创建 **TOPIC** 对象

主题是发布/预订消息中发布的信息的主题, 通过创建 TOPIC 对象来定义主题。TOPIC 对象有两个参数, 用于定义它们是否可以与多点广播配合使用。这些参数为: **COMMINFO** 和 **MCAST**。

- **COMMINFO** 此参数指定多点广播通信信息对象的名称。有关 COMMINFO 对象参数的更多信息，请参阅 [DEFINE COMMINFO](#)。
- **MCAST** 此参数指定在主题树中的此位置是否允许多点广播。

使用以下命令行示例来定义用于多点广播的 TOPIC 对象：

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

将创建名为 *ALLSPORTS* 的新 TOPIC 对象。它具有主题字符串 *Sports*，其相关通信信息对象称为 *MC1* (这是您在先前示例中定义 COMMINFO 对象时指定的名称)，并且已启用多点广播。

测试多点广播发布/预订

创建 TOPIC 和 COMMINFO 对象后，可以使用 `amqspubc` 样本和 `amqssubc` 样本对其进行测试。有关这些样本的更多信息，请参阅 [发布/预订样本程序](#)。

1. 打开两个命令行窗口；第一个命令行用于 `amqspubc` 发布样本，第二个命令行用于 `amqssubc` 预订样本。
2. 在命令行 1 上输入以下命令：

```
amqspubc Sports QM1
```

其中 *Sports* 是先前示例中定义的 TOPIC 对象的主题字符串，*QM1* 是队列管理器的名称。

3. 在命令行 2 上输入以下命令：

```
amqssubc Sports QM1
```

其中 *Sports* 和 *QM1* 与步骤 [第 207 页的『2』](#) 中使用的相同。

4. 在命令行 1 输入 `Hello world`。如果正确配置了 COMMINFO 对象中指定的端口和 IP 地址；那么 `amqssubc` 样本 (在端口上侦听来自指定地址的发布) 将在命令行 2 输出 `Hello world`。

IBM MQ 多点广播主题拓扑

使用此示例来了解 IBM MQ 多点广播主题拓扑。

IBM MQ 多点广播支持要求每个子树在总层次结构中都有自己的多点广播组和数据流。

有类网络 IP 寻址方案针对多点广播地址指定了地址空间。完整的多点广播 IP 地址范围是 224.0.0.0 到 239.255.255.255，但其中一些地址是保留的地址。要获取保留地址的列表，请联系您的系统管理员，或访问 <https://www.iana.org/assignments/multicast-addresses> 以获取更多信息。建议您使用 239.0.0.0 到 239.255.255.255 之间的本地作用域多点广播地址。

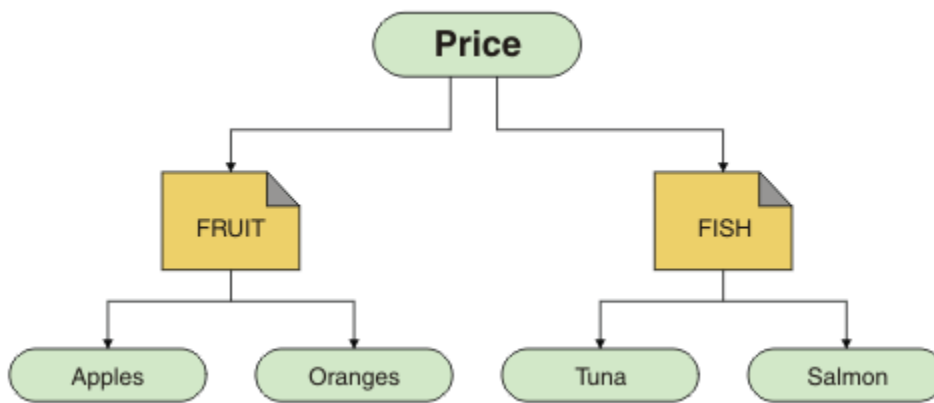
在下图中，有两个可能的多点广播数据流：

```
DEF COMMINFO(MC1) GRPADDR(239.XXX.XXX.XXX
)
DEF COMMINFO(MC2) GRPADDR(239.YYY.YYY.YYY)
```

其中 239.XXX.XXX.XXX 和 239.YYY.YYY.YYY 是有效的多点广播地址。

这些主题定义用于创建主题树，如下图所示：

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)
DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



每个多点广播通信信息 (COMMINFO) 对象都表示一条不同的数据流，因为其组地址不同。在此示例中，FRUIT 主题定义为使用 COMMINFO 对象 MC1，FISH 主题定义为使用 COMMINFO 对象 MC2，并且 Price 节点没有多点广播定义。

IBM MQ 多点广播针对主题字符串具有 255 字符限制。此限制意味着必须注意树中节点和叶节点的名称；如果节点和叶节点的名称太长，主题字符串可能会超过 255 个字符，并返回 [2425 \(0979\) \(RC2425\)](#): [MQRC_TOPIC_STRING_ERROR](#) 原因码。建议尽可能保持主题字符串简短，因为较长的主题字符串可能对性能造成不利影响。

控制多点广播消息的大小

使用此信息可了解 IBM MQ 消息格式，并减小 IBM MQ 消息的大小。

IBM MQ 消息具有许多与它们相关联的属性，这些属性包含在消息描述符中。对于小型消息，这些属性可能表示大部分数据流量，并可能对传输速率产生重大不利影响。IBM MQ 多点广播使用户能够配置随消息一起传输的属性 (如果有)。

消息属性 (主题字符串除外) 的存在取决于 COMMINFO 对象是否声明必须发送这些属性。如果未传输属性，那么接收应用程序将应用缺省值。缺省 MQMD 值不一定与 MQMD_DEFAULT 值相同，稍后在 [第 209 页的表 17](#) 中进行了描述。

COMMINFO 对象包含 MCPROP 属性，用于控制随消息一起流动的 MQMD 字段和用户属性的数量。通过将此属性的值设置为适当的级别，可以控制 IBM MQ 多点广播消息的大小：

MCPROP

多点广播属性控制随消息一起流动的 MQMD 属性和用户属性数量。

ALL

传输 MQMD 的所有用户属性和所有字段。

REPLY

将仅传输用户属性和处理消息应答的 MQMD 字段。这些属性包括：

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

USER

将仅传输用户属性。

无

将不会传输任何用户属性或 MQMD 字段。

COMPAT

此值导致以兼容方式将消息传输到 RMM，这允许与当前 XMS 应用程序和 IBM Integration Bus RMM 应用程序进行某些互操作。

多点广播消息属性

消息属性可以来自各种位置，例如 MQMD，MQRFH2 中的字段以及消息属性。

下表显示了根据 MCPROP 值和未发送属性时使用的缺省值发送消息时发生的情况。

| 属性 | 使用多点广播时的操作 | 缺省值 (如果未传输) |
|-----------------|--------------|----------------------|
| TopicString | 始终包含 | 不适用 |
| MQMQ StrucId | 未传输 | 不适用 |
| MQMD 版本 | 未传输 | 不适用 |
| 报告 | 包含 (如果不是缺省值) | 0 |
| MsgType | 包含 (如果不是缺省值) | MQMT_DATAGRAM |
| 到期 | 包含 (如果不是缺省值) | 0 |
| Feedback | 包含 (如果不是缺省值) | 0 |
| 编码 | 包含 (如果不是缺省值) | MQENC_NORMAL (单元) |
| CodedCharSetId | 包含 (如果不是缺省值) | 1208 |
| 格式 | 包含 (如果不是缺省值) | MQRFH2 |
| 优先级 | 包含 (如果不是缺省值) | 4 |
| 持久 | 包含 (如果不是缺省值) | MQPER_NOT_PERSISTENT |
| MsgId | 包含 (如果不是缺省值) | Null |
| CorrelId | 包含 (如果不是缺省值) | Null |
| BackoutCount | 包含 (如果不是缺省值) | 0 |
| ReplyToQ | 包含 (如果不是缺省值) | Blank |
| ReplyToQMgr | 包含 (如果不是缺省值) | Blank |
| UserIdentifier | 包含 (如果不是缺省值) | Blank |
| AccountingToken | 包含 (如果不是缺省值) | Null |
| PutAppIType | 包含 (如果不是缺省值) | MQAT_JAVA |
| PutAppIName | 包含 (如果不是缺省值) | Blank |
| PutDate | 包含 (如果不是缺省值) | Blank |
| PutTime | 包含 (如果不是缺省值) | Blank |
| ApplOriginData | 包含 (如果不是缺省值) | Blank |
| GroupID | 已排除 | 不适用 |
| MsgSeqNumber | 已排除 | 不适用 |
| 偏移量 | 已排除 | 不适用 |
| MsgFlags | 已排除 | 不适用 |
| OriginalLength | 已排除 | 不适用 |
| UserProperties | 已包含 | 不适用 |

相关参考

[变更命令信息](#)

相关信息

[定义命令信息](#)

为多点广播消息传递启用数据转换

本信息用于了解针对 IBM MQ 多点广播消息传递的数据转换如何工作。

IBM MQ 多点广播是一种共享的无连接协议，因此对于每个客户机来说不可能针对数据转换发出特定的请求。预订同一多点广播流的每个客户机都会收到相同的二进制数据；因此，如果需要进行 IBM MQ 数据转换，都会在每个客户机本地执行转换。

在混合平台安装中，可能是大多数客户机需要的数据格式不是传输应用程序的本机格式。在此情况下，可以使用多点广播 COMMINFO 对象的 **CCSID** 和 **ENCODING** 值来定义消息传输的编码以提高效率。

IBM MQ 多点广播支持以下内置格式的消息有效内容的数据转换：

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

除了这些格式外，您还可以定义自己的格式并使用 [MQDXP-数据转换出口参数](#) 数据转换出口。

有关对数据转换进行编程的信息，请参阅 [用于多点广播消息传递的 MQI 中的数据转换](#)。

有关数据转换的更多信息，请参阅 [数据转换](#)。

有关数据转换出口和 [ClientExitPath](#) 的更多信息，请参阅客户机配置文件的 [ClientExitPath](#) 节。

多点广播应用程序监视

使用此信息可了解有关管理和监视 IBM MQ 多点广播的信息。

多点广播流量的当前发布者和订户的状态 (例如，发送和接收的消息数或丢失的消息数) 会定期从客户机传输到服务器。接收到状态时，COMMINFO 对象的 COMMEV 属性指定队列管理器是否将事件消息放在 SYSTEM.ADMIN.PUBSUB.EVENT。事件消息包含接收到的状态信息。此信息是查找问题根源的宝贵诊断帮助。

使用 MQSC 命令 **DISPLAY CONN** 来显示有关连接到队列管理器的应用程序的连接信息。有关 **DISPLAY CONN** 命令的更多信息，请参阅 [DISPLAY CONN](#)。

使用 MQSC 命令 **DISPLAY TPSTATUS** 可显示发布程序和订户的状态。有关 **DISPLAY TPSTATUS** 命令的更多信息，请参阅 [DISPLAY TPSTATUS](#)。

COMMEV 和多点广播消息可靠性指示符

可靠性指示符与 COMMINFO 对象的 **COMMEV** 属性结合使用，是监视 IBM MQ 多点广播发布程序和订户的关键元素。可靠性指示符 (在 "发布" 或 "预订" 状态命令上返回的 **MSGREL** 字段) 是一个 IBM MQ 指示符，用于说明没有错误的传输百分比。有时，由于传输错误 (反映在 **MSGREL** 的值中)，必须重新传输消息。传输错误的潜在原因包括订户缓慢，网络繁忙和网络中断。**COMMEV** 控制是否为使用 COMMINFO 对象创建的多点广播句柄生成事件消息，并设置为以下三个可能值之一：

DISABLED

不写入事件消息。

ENABLED

事件消息始终以 COMMINFO **MONINT** 参数中定义的频率写入。

异常

如果消息可靠性低于可靠性阈值，那么将写入事件消息。90% 或更低的消息可靠性级别指示网络配置可能存在问题，或者一个或多个发布/预订应用程序运行速度太慢：

- 值 **MSGREL (100, 100)** 指示在短期或长期时间范围内都没有问题。
- 值 **MSGREL (80, 60)** 指示 20% 的消息当前存在问题，但这也是对长期值 60 的改进。

即使到队列管理器的单点广播连接中断，客户机也可能继续传输和接收多点广播流量，因此数据可能已过时。

多点广播消息可靠性

使用此信息可了解如何设置 IBM MQ 多点广播预订和消息历史记录。

通过多点广播克服传输故障的关键要素是 IBM MQ 对传输的数据 (要在链路的传输端保留的消息的历史记录) 进行缓冲。此过程意味着在放入应用程序过程中不需要对消息进行缓冲，因为 IBM MQ 提供了可靠性。此历史记录的大小是通过通信信息 (COMMINFO) 对象配置的，如以下信息中所述。更大的传输缓冲区意味着有更多的传输历史记录需要重新传输 (如果需要)，但由于组播的性质，无法支持 100% 有保证的传送。

IBM MQ 多点广播消息历史记录在通信信息 (COMMINFO) 对象中由 **MSGHIST** 属性控制：

MSGHIST

此值是系统为处理 NACKs (否定应答) 情况下的重新传输而保留的消息历史记录量 (以千字节为单位)。

值 0 表示可靠性级别最低。缺省值为 100 KB。

IBM MQ 多点广播新预订历史记录在通信信息 (COMMINFO) 对象中由 **NSUBHIST** 属性控制：

NSUBHIST

新订户历史记录控制加入发布流的订户是接收当前可用的所有数据，还是仅接收预订以来进行的发布。

无

值 NONE 将导致发送方仅传输从预订时进行的发布。NONE 是缺省值。

ALL

值 ALL 将导致发送设备重新发送已知的主题历史记录。在某些情况下，这种情况会给保留的发布提供类似的行为。

注：如果由于重新传输了所有主题历史记录，因此存在较大的主题历史记录，那么使用 ALL 的值可能会对性能产生不利影响。

相关信息

[定义命令信息](#)

[变更命令信息](#)

高级多点广播任务

使用此信息可了解高级 IBM MQ 多点广播管理任务，例如，配置 .ini 文件以及与 IBM MQ LLM 的互操作性。

有关多点广播安装中的安全性的注意事项，请参阅 [多点广播安全性](#)。

多点广播和非多点广播发布/预订域之间的桥接

使用此信息可了解当非多点广播发布程序发布到 IBM MQ 已启用多点广播的主题时发生的情况。

如果非多点广播发布程序发布到定义为 **MCAST enabled** 和 **BRIDGE enabled** 的主题，那么队列管理器会将消息通过多点广播直接传输到可能正在侦听的任何订户。多点广播发布程序无法发布到未启用多点广播的主题。

可以通过设置主题对象的 **MCAST** 和 **COMMINFO** 参数来启用现有主题多点广播。有关这些参数的更多信息，请参阅 [初始多点广播概念](#)。

COMMINFO 对象 **BRIDGE** 属性控制来自未使用多点广播的应用程序的发布。如果 **BRIDGE** 设置为 **ENABLED**，并且主题的 **MCAST** 参数也设置为 **ENABLED**，那么来自未使用多点广播的应用程序的出版物将桥接到执行此操作的应用程序。有关 **BRIDGE** 参数的更多信息，请参阅 [DEFINE COMMINFO](#)。

为多点广播配置 .ini 文件

使用此信息可了解 .ini 文件中的 IBM MQ 多点广播字段。

可以在 ini 文件中进行其他 IBM MQ 多点广播配置。必须使用的特定 ini 文件取决于应用程序类型:

- 客户机: 配置 `MQ_DATA_PATH/mqclient.ini` 文件。
- 队列管理器: 配置 `MQ_DATA_PATH/qmgrs/QMNAME/qm.ini` 文件。

其中 `MQ_DATA_PATH` 是 IBM MQ 数据目录 (`/var/mqm/mqclient.ini`) 的位置, `QMNAME` 是应用 .ini 文件的队列管理器的名称。

.ini 文件包含用于微调 IBM MQ 多点广播: 的行为的字段

```
Multicast:
Protocol      = IP | UDP
IPVersion     = IPv4 | IPv6 | ANY | BOTH
LimitTransRate = DISABLED | STATIC | DYNAMIC
TransRateLimit = 100000
SocketTTL     = 1
Batch         = NO
Loop          = 1
Interface     = <IPAddress>
FeedbackMode  = ACK | NACK | WAIT1
HeartbeatTimeout = 20000
HeartbeatInterval = 2000
```

协议

UDP

在此方式下, 将使用 UDP 协议发送包。但是, 网络元素无法像在 IP 方式下一样在多点广播分发中提供帮助。包格式与 PGM 保持兼容。这是缺省值。

IP

在此方式下, 发送方发送原始 IP 包。具有 PGM 支持的网络元素有助于可靠的多点广播包分发。此方式与 PGM 标准完全兼容。

IPVersion

IPv4

仅使用 IPv4 协议进行通信。这是缺省值。

IPv6

仅使用 IPv6 协议进行通信。

ANY

使用 IPv4 和/或 IPv6 进行通信, 具体取决于可用的协议。

BOTH

支持使用 IPv4 和 IPv6 进行通信。

LimitTrans 速率

DISABLED

没有传输速率控制。这是缺省值。

静态

实现静态传输速率控制。发送方的传输速率不会超过 `TransRateLimit` 参数指定的速率。

动态

发射机根据从接收机获得的反馈来调整其传输速率。在这种情况下, 传输速率限制不能超过 `TransRateLimit` 参数指定的值。发射机试图达到最佳传输速率。

TransRate 限制

传输速率限制 (以 Kbps 为单位)。

SocketTTL

SocketTTL 的值确定多点广播流量是可以通过路由器, 还是可以通过路由器的数目。

批处理

控制是立即对消息进行批处理还是发送。有 2 个可能的值:

- NO 未对消息进行批处理，将立即发送这些消息。
- YES 将对消息进行批处理。

循环

将值设置为 1 以启用多点广播循环。多点广播循环定义发送的数据是否回送到主机。

接口

多点广播流量在其上流动的接口的 IP 地址。有关更多信息和故障诊断，请参阅：[在非多点广播网络上测试多点广播应用程序](#) 和 [为多点广播流量设置相应的网络](#)

FeedbackMode

NACK

通过否定应答进行反馈。这是缺省值。

ACK

肯定的反馈。

WAIT1

由肯定应答进行的反馈，其中发送方仅等待来自任何接收方的 1 ACK。

HeartbeatTimeout

脉动信号超时(以毫秒计)。值 0 指示主题的一个或多个接收方未引发脉动信号超时事件。缺省值为 20000。

HeartbeatInterval

脉动信号间隔(以毫秒为单位)。值 0 指示未发送任何脉动信号。脉动信号间隔必须远小于 **HeartbeatTimeout** 值，以避免发生错误的脉动信号超时事件。缺省值为 2000。

与 IBM MQ 低等待时间消息传递的多点广播互操作性

使用此信息可了解 IBM MQ 多点广播与 IBM MQ 低等待时间消息传递 (LLM) 之间的互操作性。

对于使用 LLM 的应用程序，可以进行基本有效内容传输，而另一个应用程序使用多点广播来双向交换消息。虽然多点广播使用 LLM 技术，但 LLM 产品本身并不是嵌入式的。因此，可以同时安装 LLM 和 IBM MQ 多点广播，并分别对这两个产品进行操作和服务。

与多点广播通信的 LLM 应用程序可能需要发送和接收消息属性。IBM MQ 消息属性和 MQMD 字段作为具有特定 LLM 消息属性代码的 LLM 消息属性进行传输，如下表中所示：

| IBM MQ 属性 | IBM MQ LLM 属性类型 | LLM 属性类型 | LLM 属性代码 |
|---------------------|--------------------|-------------------------|----------|
| MQMD.Report | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1001 |
| MQMD.MsgType | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1002 |
| MQMD.Expiry | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1003 |
| MQMD.Feedback | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1004 |
| MQMD.Encoding | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1005 |
| MQMD.CodedCharSetId | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1006 |
| MQMD.Format | RMM_MSG_PROP_BYTES | LLM_PROP_KIND_String | -1007 |
| MQMD.Priority | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1008 |
| MQMD.Persistence | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1009 |
| MQMD.MsgId | RMM_MSG_PROP_BYTES | LLM_PROP_KIND_ByteArray | -1010 |
| MQMD.BackoutCount | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_Int32 | -1012 |
| MQMD.ReplyToQ | RMM_MSG_PROP_BYTES | LLM_PROP_KIND_String | -1013 |
| MQMD.ReplyToQMger | RMM_MSG_PROP_BYTES | LLM_PROP_KIND_String | -1014 |

| 表 18: IBM MQ 消息属性到 IBM MQ LLM 属性映射 (继续) | | | |
|-----------------------------------------|--------------------|----------------------|----------|
| IBM MQ 属性 | IBM MQ LLM 属性类型 | LLM 属性类型 | LLM 属性代码 |
| MQMD.PutDate | RMM_MSG_PROP_BYTES | LLM_PROP_KIND_String | -1020 |
| MQMD.PutTime | RMM_MSG_PROP_BYTES | LLM_PROP_KIND_String | -1021 |
| MQMD.ApplOriginData | RMM_MSG_PROP_BYTES | LLM_PROP_KIND_String | -1022 |
| MQPubOptions | RMM_MSG_PROP_INT32 | LLM_PROP_KIND_int32 | -1053 |

有关 LLM 的更多信息，请参阅 LLM 产品文档: [IBM MQ Low Latency Messaging](#)。

IBM i 管理 IBM MQ for IBM i

介绍可用于在 IBM i 上管理 IBM MQ 的方法。

管理任务包括创建，启动，改变，查看，停止和删除集群，进程和 IBM MQ 对象 (队列管理器，队列，名称列表，进程定义，通道，客户机连接通道，侦听器，服务和认证信息对象)。

请参阅以下链接以获取有关如何管理 IBM MQ for IBM i 的详细信息:

- [第 214 页的『使用 CL 命令管理 IBM MQ for IBM i』](#)
- [第 227 页的『管理 IBM MQ for IBM i 的替代方法』](#)
- [第 231 页的『IBM i 的工作管理』](#)

相关概念

[第 237 页的『IBM i 上的可用性，备份，恢复和重新启动』](#)

使用此信息来了解 IBM MQ for IBM i 如何使用 IBM i 日志记录支持来帮助其备份和复原策略。

相关参考

[第 273 页的『暂停 IBM MQ for IBM i』](#)

本节说明如何暂停 (正常结束) IBM MQ for IBM i。

相关信息

[更改 IBM i 上的配置信息](#)

[了解 IBM MQ for IBM i 队列管理器库名](#)

[在 IBM i 上设置安全性](#)

[IBM i 上的死信队列处理程序](#)

[使用 IBM MQ for IBM i 应用程序确定问题](#)

[IBM i 上的可安装服务和组件](#)

[IBM i 上的系统对象和缺省对象](#)

IBM i 使用 CL 命令管理 IBM MQ for IBM i

使用此信息来了解 IBM MQ IBM i 命令。

可以使用相关 **WRK*** 命令访问大多数 IBM MQ 命令组，包括与队列管理器，队列，主题，通道，名称列表，进程定义和认证信息对象相关联的命令组。

集合中的主体命令为 **WRKMQM**。例如，此命令允许您显示系统上所有队列管理器的列表以及状态信息。或者，您可以针对每个条目使用各种选项来处理所有特定于队列管理器的命令。

从 **WRKMQM** 命令中，可以选择每个队列管理器的特定区域，例如，使用通道，主题或队列，然后从其中选择个别对象。

记录 IBM MQ 应用程序定义

创建或定制 IBM MQ 应用程序时，保留所创建的所有 IBM MQ 定义的记录很有用。此记录可用于:

- 恢复目的

- 维护
- 推出 IBM MQ 个应用程序

您可以通过 1 (共 2) 方式记录 IBM MQ 应用程序定义:

1. 创建 CL 程序以生成服务器的 IBM MQ 定义。
2. 创建 MQSC 文本文件作为 SRC 成员, 以使用跨平台 IBM MQ 命令语言生成 IBM MQ 定义。

有关定义队列对象的更多详细信息, 请参阅 [第 10 页的『脚本 \(MQSC\) 命令』](#) 和 [第 19 页的『使用 IBM MQ 可编程命令格式』](#)。

相关信息

[IBM MQ for IBM i CL 命令参考](#)

IBM i 在使用 CL 命令开始使用 IBM MQ for IBM i 之前

使用此信息来启动 IBM MQ 子系统并创建本地队列管理器。

开始之前

确保 IBM MQ 子系统正在运行 (使用命令 STRSBS QMQM/QMQM), 并且未挂起与该子系统关联的作业队列。缺省情况下, IBM MQ 子系统和作业队列都在库 QMQM 中命名为 QMQM。

关于此任务

使用 IBM i 命令行来启动队列管理器

过程

1. 通过从 IBM i 命令行发出 CRTMQM 命令来创建本地队列管理器。
创建队列管理器时, 可以选择使该队列管理器成为缺省队列管理器。如果省略了队列管理器名称参数 (MQMNAME), 那么缺省队列管理器 (只能有一个) 是 CL 命令所应用于的队列管理器。
2. 通过从 IBM i 命令行发出 STRMQM 命令来启动本地队列管理器。
如果队列管理器启动时间超过几秒, 那么 IBM MQ 将间歇地显示详细说明启动进度的状态消息。有关这些消息的更多信息, 请参阅 [消息和原因码](#)。

下一步做什么

您可以通过从 IBM i 命令行发出 ENDMQM 命令来停止队列管理器, 并通过从 IBM i 命令行发出其他 IBM MQ 命令来控制队列管理器。

远程队列管理器不能远程启动, 但必须由本地操作员在其系统中创建和启动。此操作的例外情况是存在远程操作设施 (在 IBM MQ for IBM i 外部) 以启用此类操作。

本地队列管理员无法停止远程队列管理器。

注: 作为停顿 IBM MQ 系统的一部分, 您必须停顿活动队列管理器。在 [第 273 页的『停顿 IBM MQ for IBM i』](#) 中对此进行了描述。

IBM i 创建 IBM MQ for IBM i 对象

使用此信息来了解为 IBM i 创建 IBM MQ 对象的方法。

开始之前

以下任务建议了可以从命令行使用 IBM MQ for IBM i 的各种方法。

关于此任务

有两种用于创建 IBM MQ 对象的联机方法, 它们是:

过程

1. 使用 Create 命令，例如：**Create MQM Queue** 命令：**CRTMQMQ**
2. 使用后跟 F6 的 "使用 MQM 对象" 命令，例如：**Work with MQM Queues** 命令：**WRKMQM**

下一步做什么

有关所有命令的列表，请参阅 [IBM MQ for IBM i CL 命令](#)。

注：可以从 "消息队列管理器命令" 菜单提交所有 MQM 命令。要显示此菜单，请在命令行上输入 GO CMDMQM，然后按 Enter 键。

当您从此菜单中选择命令时，系统会自动显示提示面板。要显示直接在命令行上输入的命令的提示面板，请先按 F4 键，然后再按 Enter 键。

使用 CRTMQMQ 命令创建本地队列

过程

1. 在命令行上输入 CHGMQM，然后按 F4 键。
2. 在 "创建 MQM 队列" 面板上，在 Queue name 字段中输入要创建的队列的名称。要指定混合大小写的名称，请将该名称括在单引号中。
3. 在 Queue type 字段中输入 *LCL。
4. 除非您正在使用缺省队列管理器，否则请指定队列管理器名称，然后按 Enter 键。您可以使用新值覆盖任何值。向前滚动以查看更多字段。用于集群的选项位于选项列表的末尾。
5. 更改任何值后，按 Enter 键以创建队列。

使用 WRKMQM 命令创建本地队列

过程

1. 在命令行上输入 WRKMQM。
2. 输入队列管理器的名称。
3. 如果要显示提示面板，请按 F4。通过指定通用队列名称或队列类型，提示面板对于减少显示的队列数很有用。
4. 按 Enter，将显示 "使用 MQM 队列" 面板。您可以使用新值覆盖任何值。向前滚动以查看更多字段。用于集群的选项位于选项列表的末尾。
5. 按 F6 以创建新队列；这将转至 CRTMQMQ 面板。有关如何创建队列的指示信息，请参阅第 216 页的『使用 CRTMQMQ 命令创建本地队列』。创建队列后，将再次显示使用 MQM 队列 面板。当您按 F5=Refresh 键时，会将新队列添加到列表中。

更改队列管理器属性

关于此任务

要更改在 CHGMQM 命令上指定的队列管理器的属性，请指定要更改的属性和值。例如，使用以下选项来变更 jupiter.queue.manager 的属性：

过程

在命令行上输入 CHGMQM，然后按 F4 键。

结果

此命令将更改所使用的死信队列，并启用禁止事件。

IBM i 在 IBM i 上使用本地队列

本部分包含可用于管理本地队列的一些命令的示例。还可以使用 **WRKMQMQ** 命令面板中的选项来显示所有命令。

定义本地队列

对于应用程序，本地队列管理器是应用程序所连接的队列管理器。由本地队列管理器管理的队列被认为是该队列管理器的本地队列。

使用命令 **CRTMQMQ QTYPE *LCL** 可创建本地队列的定义，还可创建称为队列的数据结构。您还可以从缺省本地队列的队列特征中修改队列特征。

在此示例中，将我们定义的队列 `orange.local.queue` 指定为具有以下特征：

- 它针对获取启用，针对放置禁用，并以先进先出 (FIFO) 为基础进行操作。
- 它是普通队列，即，它不是启动队列或传输队列，并且不会生成触发器消息。
- 最大队列深度为 1000 条消息；最大消息长度为 2000 字节。

以下命令在缺省队列管理器上执行此操作：

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL)
TEXT('Queue for messages from other systems')
PUTENBL(*NO)
GETENBL(*YES)
TRGENBL(*NO)
MSGDLYSEQ(*FIFO)
MAXDEPTH(1000)
MAXMSGL(2000)
USAGE(*NORMAL)
```

注：

1. USAGE *NORMAL 指示此队列不是传输队列。
2. 如果在同一队列管理器上已有名为 `orange.local.queue` 的本地队列，那么此命令将失败。如果要覆盖队列的现有定义，请使用 REPLACE *YES 属性，但另请参阅 [第 218 页的『更改本地队列属性』](#)。

定义死信队列

每个队列管理器都必须具有要用作死信队列的本地队列，以便可以存储无法传递到其正确目标的消息以供以后检索。您必须明确告知队列管理器有关死信队列的信息。您可以通过在 **CRTMQM** 命令上指定死信队列来执行此操作，也可以稍后使用 **CHGMQM** 命令来指定一个死信队列。您还必须先定义死信队列，然后才能使用该队列。

产品随附了名为 `SYSTEM.DEAD.LETTER.QUEUE` 的样本死信队列。创建队列管理器时，将自动创建此队列。如果需要，您可以修改此定义。无需对其进行重命名，尽管您可以根据需要进行重命名。

死信队列没有特殊要求，只是：

- 它必须是本地队列。
- 其 MAXMSGL (最大消息长度) 属性必须使队列能够容纳队列管理器必须处理的最大消息加上死信头 (MQDLH) 的大小。

IBM MQ 提供了一个死信队列处理程序，用于指定如何处理或除去在死信队列中找到的消息。有关更多信息，请参阅 [IBM MQ for IBM i 死信队列处理程序](#)。

显示缺省对象属性

当您定义 IBM MQ 对象时，它将采用您未从缺省对象中指定的任何属性。例如，当您定义本地队列时，该队列会从缺省本地队列（称为 `SYSTEM.DEFAULT.LOCAL.QUEUE`）继承您在定义中省略的任何属性。要确切查看这些属性的内容，请使用以下命令：

```
DSPMQMQ QNAME(SYSTEM.DEFAULT.LOCAL.QUEUE) MQMNAME(MYQUEUEMANAGER)
```

复制本地队列定义

您可以使用 `CPYMQMQ` 命令复制队列定义。例如：

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

此命令创建的队列具有与原始队列 `orange.local.queue` 相同的属性，而不是系统缺省本地队列的属性。您还可以使用 `CPYMQMQ` 命令来复制队列定义，但将一个或多个更改替换为原始属性。例如：

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('third.queue') MQMNAME(MYQUEUEMANAGER)  
MAXMSGLEN(1024)
```

此命令将队列 `orange.local.queue` 的属性复制到队列 `third.queue`，但指定新队列上的最大消息长度为 1024 字节，而不是 2000 字节。

注：使用 `CPYMQMQ` 命令时，仅复制队列属性，而不复制队列上的消息。

更改本地队列属性

可以通过两种方式更改队列属性，使用带有 `REPLACE *YES` 属性的 `CHGMQMQ` 命令或 `CPYMQMQ` 命令。在第 217 页的『定义本地队列』中，您定义了队列 `orange.local.queue`。例如，如果需要将此队列上的最大消息长度增加到 10,000 字节。

- 使用 `CHGMQMQ` 命令：

```
CHGMQMQ QNAME('orange.local.queue') MQMNAME(MYQUEUEMANAGER) MAXMSGLEN(10000)
```

此命令会更改单个属性，即最大消息长度的属性；所有其他属性保持不变。

- 将 `CRTMQMQ` 命令与 `REPLACE *YES` 选项配合使用，例如：

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL) MQMNAME(MYQUEUEMANAGER)  
MAXMSGLEN(10000) REPLACE(*YES)
```

此命令不仅会更改最大消息长度，还会更改所有其他属性（给定它们的缺省值）。现在已启用该队列，而先前已禁止将其放入。启用 `PUT` 是缺省值，由队列 `SYSTEM.DEFAULT.LOCAL.QUEUE` 指定，除非您对其进行了更改。

如果减小现有队列上的最大消息长度，那么现有消息不受影响。但是，任何新消息都必须满足新条件。

清除本地队列

要从名为 `magenta.queue` 的本地队列中删除所有消息，请使用以下命令：

```
CLRMQMQ QNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

在下列情况下，无法清除队列：

- 在同步点下的队列上已放入未落实的消息。
- 某个应用程序当前打开了该队列。

删除本地队列

使用命令 **DLTMQM** 来删除本地队列。

如果队列上有未落实的消息，或者队列正在使用中，那么无法将其删除。

启用大型队列

IBM MQ 支持大于 2 GB 的队列。请参阅操作系统文档，以获取有关如何启用 IBM i 以支持大型文件的信息。

可以在 [IBM Documentation](#) 中找到 IBM i 产品文档。

某些实用程序可能无法处理大于 2 GB 的文件。在启用大型文件支持之前，请查看操作系统文档以获取有关此类支持的限制的信息。

IBM i 在 IBM i 上使用别名队列

本部分包含可用于管理别名队列的一些命令的示例。还可以使用 **WRKMQMQ** 命令面板中的选项来显示所有命令。

别名队列 (有时称为队列别名) 提供了重定向 MQI 调用的方法。别名队列不是实际队列，而是解析为实际队列的定义。别名队列定义包含由 TGTQNAME 属性指定的目标队列名称。

当应用程序在 MQI 调用中指定别名队列时，队列管理器会在运行时解析实际队列名称。

例如，已开发应用程序以将消息放入名为 `my.alias.queue` 的队列中。它在发出 **MQOPEN** 请求时指定此队列的名称，并在将消息放入此队列时指定此队列的名称。此应用程序不知道此队列是别名队列。对于使用此别名的每个 MQI 调用，此队列管理器会解析实际队列名称 (可以是在此队列管理器上定义的本地队列或远程队列)。

通过更改 TGTQNAME 属性的值，可以将 MQI 调用重定向到另一个队列 (可能在另一个队列管理器上)。这有助于维护、迁移和负载均衡。

定义别名队列

以下命令创建别名队列：

```
CRTMQM QNAME('my.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
MQMNAME(MYQUEUEMANAGER)
```

此命令将指定 `my.alias.queue` 的 MQI 调用重定向到队列 `yellow.queue`。此命令不会创建目标队列；如果队列 `yellow.queue` 在运行时不存在，那么 MQI 调用将失败。

如果您更改此别名定义，可以将 MQI 调用重定向至另一个队列。例如：

```
CHGMQM QNAME('my.alias.queue') TGTQNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

此命令将 MQI 调用重定向到另一个队列 `magenta.queue`。

您也可以使用别名队列，使单个队列 (目标队列) 对于不同的应用程序看起来具有不同的属性。可通过定义两个别名 (对每个应用程序各定义一个) 来实现这一点。假设有两个应用程序：

- 应用程序 ALPHA 可以将消息放在 `yellow.queue` 上，但不允许从它获取消息。
- 应用程序 BETA 可以从 `yellow.queue` 获取消息，但不允许将消息放在该消息上。

您可以使用以下命令来执行此操作：

```
/* This alias is put enabled and get disabled for application ALPHA */
CRTMQM QNAME('alphas.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
PUTENBL(*YES) GETENBL(*NO) MQMNAME(MYQUEUEMANAGER)

/* This alias is put disabled and get enabled for application BETA */
```

```
CRTMQMQ QNAME('betas.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
PUTENBL(*NO) GETENBL(*YES) MQMNAME(MYQUEUEMANAGER)
```

ALPHA 在其 MQI 调用中使用队列名称 `alphas.alias.queue` ; BETA 使用队列名称 `betas.alias.queue`。它们都访问相同的队列，但以不同的方式访问。

您可以在定义别名队列时使用 `REPLACE *YES` 属性，其方式与将这些属性与本地队列配合使用的方式相同。

对别名队列使用其他命令

您可以使用相应的命令来显示或更改别名队列属性。例如：

```
* Display the alias queue's attributes */
DSPMQMQ QNAME('alphas.alias.queue') MQMNAME(MYQUEUEMANAGER)
/* ALTER the base queue name, to which the alias resolves. */
/* FORCE = Force the change even if the queue is open. */
CHQMCMQ QNAME('alphas.alias.queue') TGTQNAME('orange.local.queue') FORCE(*YES)
MQMNAME(MYQUEUEMANAGER)
```

IBM i 在 IBM i 上使用模型队列

本部分包含可用于管理模型队列的一些命令的示例。还可以使用 **WRKMQMQ 命令** 面板中的选项来显示所有命令。

如果队列管理器从指定已定义为模型队列的队列名称的应用程序接收到 MQI 调用，那么队列管理器将创建动态队列。新动态队列的名称由队列管理器在创建队列时生成。模型队列是一个模板，用于指定从中创建的任何动态队列的属性。

模型队列为应用程序提供了一种方便的方法来根据需要创建队列。

定义模型队列

使用一组属性定义模型队列的方式与定义本地队列的方式相同。模型队列和本地队列具有相同的属性集，但在模型队列上，您可以指定创建的动态队列是临时的还是永久的。(在队列管理器重新启动之间保留永久队列，而不保留临时队列)。例如：

```
CRTMQMQ QNAME('green.model.queue') QTYPE(*MDL) DFNTYPE(*PERMDYN)
```

此命令创建模型队列定义。在 `DFNTYPE` 属性中，从此模板创建的实际队列是永久动态队列。未指定的属性将自动从 `SYSYSTEM.DEFAULT.MODEL.QUEUE` 缺省队列复制。

您可以在定义模型队列时使用 `REPLACE *YES` 属性，其方式与将它们与本地队列配合使用的方式相同。

将其他命令与模型队列配合使用

您可以使用相应的命令来显示或更改模型队列的属性。例如：

```
/* Display the model queue's attributes */
DSPMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('green.model.queue')
/* ALTER the model queue to enable puts on any */
/* dynamic queue created from this model. */
CHGMCMQ MQMNAME(MYQUEUEMANAGER) QNAME('blue.model.queue') PUTENBL(*YES)
```

IBM i 在 IBM i 上使用触发

使用此信息可了解有关触发和进程定义的信息。

IBM MQ 提供了用于在满足队列上的特定条件时自动启动应用程序的工具。条件的一个示例是当队列中的消息数达到指定的数目时。此工具称为 **触发**，在 [触发通道](#) 中进行了详细描述。

什么是触发？

队列管理器将某些条件定义为构成触发器事件。如果对队列启用了触发并且发生了触发器事件，那么队列管理器会将触发器消息发送到称为启动队列的队列。启动队列上出现触发器消息表示发生了触发器事件。

队列管理器生成的触发器消息不是持久的。这样可以减少日志记录 (从而提高性能)，并在重新启动期间最大限度减少重复项，从而缩短重新启动时间。

什么是触发器监视器？

处理启动队列的程序称为触发器监视器应用程序，其功能是根据触发器消息中包含的信息读取触发器消息并采取相应的操作。通常，此操作将启动其他应用程序以处理导致生成触发器消息的队列。从队列管理器的角度来看，`trigger-monitor` 应用程序没有什么特殊之处-它是从队列 (启动队列) 读取消息的另一个应用程序。

更改触发器监视器的作业提交属性

作为命令 **STRMQMTRM** 提供的触发器监视器使用系统缺省作业描述 `QDFTJOB` 为每条触发器消息提交作业。这有一些限制，即提交的作业始终称为 `QDFTJOB`，并且具有缺省作业描述的属性 (包括库列表 `*SYSVAL`)。IBM MQ 提供了用于覆盖这些属性的方法。例如，可以定制提交的作业以具有更有意义的作业名，如下所示：

1. 在作业描述中，指定所需的描述，例如日志记录值。
2. 指定触发进程中使用的进程定义的环境数据：

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOB(MYLIB/TRIGJOB)')
```

触发器监视器使用指定的描述执行 `SBMJOB`。

可以通过在进程定义的环境数据中指定相应的关键字和值来覆盖 `SBMJOB` 的其他属性。唯一的例外是 `CMD` 关键字，因为此属性由触发器监视器填充。以下是用于指定进程定义的环境数据的命令示例，其中将更改作业名和描述：

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOB(MYLIB/TRIGJOB)
JOB(TRIGGER)')
```

定义用于触发的应用程序队列

应用程序队列是应用程序通过 MQI 用于消息传递的本地队列。触发需要在应用程序队列上定义多个队列属性。触发本身由 `TRGENBL` 属性启用。

在此示例中，当本地队列 `motor.insurance.queue` 上有 100 条优先级为 5 或更高的消息时，将生成触发器事件，如下所示：

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.insurance.queue') QTYPE(*LCL)
PRCNAME('motor.insurance.quote.process') MAXMSGLEN(2000)
DFTMSGPST(*YES) INITQNAME('motor.ins.init.queue')
TRGENBL(*YES) TRGTYP(*DEPTH) TRGDEPTH(100) TRGMSGPTY(5)
```

其中，参数为：

MQMNAME(MYQUEUEMANAGER)

队列管理器的名称。

QNAME('motor.insurance.queue')

正在定义的应用程序队列的名称。

PRCNAME('motor.insurance.quote.process')

要由触发器监视器程序启动的应用程序的名称。

MAXMSGLEN(2000)

队列上消息的最大长度。

DFTMSGPST(*YES)

缺省情况下，此队列上的消息是持久的。

INITQNAME('motor.ins.init.queue')

队列管理器要在其上放置触发器消息的启动队列的名称。

TRGENBL(*YES)

触发器属性值。

TRGTYPE(*DEPTH)

当具有所需优先级 (**TRGMSGPTY**) 的消息数时，将生成触发器事件达到 **TRGDEPTH** 中指定的数字。

TRGDEPTH(100)

生成触发器事件所需的消息数。

TRGMSGPTY(5)

队列管理器在决定是否生成触发器事件时要计算的消息的优先级。仅计算优先级为 5 或更高的消息。

定义启动队列

发生触发器事件时，队列管理器会将触发器消息放在应用程序队列定义中指定的启动队列上。启动队列没有特殊设置，但您可以使用本地队列 `motor.ins.init.queue` 的以下定义来获取指导：

```

CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.ins.init.queue') QTYPE(*LCL)
GETENBL(*YES) SHARE(*NO) TRGTYPE(*NONE)
MAXMSGL(2000)
MAXDEPTH(1000)

```

创建流程定义

使用 **CRTMQMPRC** 命令来创建进程定义。进程定义将应用程序队列与要处理来自该队列的消息的应用程序相关联。这是通过应用程序队列 `motor.insurance.queue` 上的 **PRCNAME** 属性完成的。以下命令将创建此示例中标识的必需进程 `motor.insurance.quote.process`：

```

CRTMQMPRC MQMNAME(MYQUEUEMANAGER) PRCNAME('motor.insurance.quote.process')
TEXT('Insurance request message processing')
APPCTYPE(*OS400) APPID(MQTEST/TESTPROG)
USRDATA('open, close, 235')

```

其中，参数为：

MQMNAME(MYQUEUEMANAGER)

队列管理器的名称。

PRCNAME('motor.insurance.quote.process')

进程定义的名称。

TEXT('Insurance request message processing')

与此定义相关的应用程序的描述。此文本在您使用 **DSPMQMPRC** 命令时显示。这可以帮助您确定流程的作用。如果在字符串中使用空格，那么必须用单引号将字符串括起来。

APPCTYPE(*OS400)

要启动的应用程序的类型。

APPID(MQTEST/TESTPROG)

应用程序可执行文件的名称，指定为标准文件名。

USRDATA('open, close, 235')
用户定义的数据，可供应用程序使用。

显示流程定义

使用 **DSPMQPRC** 命令来检查定义的结果。例如：

```
MQMNAME(MYQUEUEMANAGER) DSPMQPRC('motor.insurance.quote.process')
```

您还可以使用 **CHGMQPRC** 命令来变更现有进程定义，并使用 **DLTMQPRC** 命令来删除进程定义。

在 IBM i 上的两个 IBM MQ 系统之间进行通信

此编码示例说明如何使用 CL 命令设置两个 IBM MQ for IBM i 系统，以便它们可以相互通信。

这些系统称为 SYSTEMA 和 SYSTEMB，使用的通信协议是 TCP/IP。

执行以下过程：

1. 在 SYSTEMA 上创建队列管理器，并将其称为 QMGRA1。

```
CRTMQM  MQMNAME(QMGRA1) TEXT('System A - Queue +  
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

2. 启动此队列管理器。

```
STRMQM  MQMNAME(QMGRA1)
```

3. 定义 SYSTEMA 上需要向 SYSTEMB 上的队列管理器发送消息的 IBM MQ 对象。

```
/* Transmission queue */  
CRTMQMQ  QNAME(XMITQ.TO.QMGRB1) QTYPE(*LCL) +  
MQMNAME(QMGRA1) TEXT('Transmission Queue +  
to QMGRB1') MAXDEPTH(5000) USAGE(*TMQ)  
  
/* Remote queue that points to a queue called TARGETB */  
/* TARGETB belongs to queue manager QMGRB1 on SYSTEMB */  
CRTMQMQ  QNAME(TARGETB.ON.QMGRB1) QTYPE(*RMT) +  
MQMNAME(QMGRA1) TEXT('Remote Q pointing +  
at Q TARGETB on QMGRB1 on Remote System +  
SYSTEMB') RMTQNAME(TARGETB) +  
RMTMQNAME(QMGRB1) TMQNAME(XMITQ.TO.QMGRB1)  
  
/* TCP/IP sender channel to send messages to the queue manager on SYSTEMB*/  
CRTMQMCHL  CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*SDR) +  
MQMNAME(QMGRA1) TRPTYPE(*TCP) +  
TEXT('Sender Channel From QMGRA1 on +  
SYSTEMA to QMGRB1 on SYSTEMB') +  
CONNNAME(SYSTEMB) TMQNAME(XMITQ.TO.QMGRB1)
```

4. 在 SYSTEMB 上创建队列管理器，并将其称为 QMGRB1。

```
CRTMQM  MQMNAME(QMGRB1) TEXT('System B - Queue +  
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

5. 在 SYSTEMB 上启动队列管理器。

```
STRMQM  MQMNAME(QMGRB1)
```

6. 定义从 SYSTEMA 上的队列管理器接收消息所需的 IBM MQ 对象。

```
/* Local queue to receive messages on */  
CRTMQMQ  QNAME(TARGETB) QTYPE(*LCL) MQMNAME(QMGRB1) +  
TEXT('Sample Local Queue for QMGRB1')
```

```

/* Receiver channel of the same name as the sender channel on SYSTEMA */
CRTMQMCHL CHLNAME(QMGRA1.TO.QMGRB1) CHLTYPE(*RCVR) +
MQMNAME(QMGRB1) TRPTYPE(*TCP) +
TEXT('Receiver Channel from QMGRA1 to +
QMGRB1')

```

7. 最后，在 SYSTEMB 上启动 TCP/IP 侦听器，以便可以启动通道。此示例使用缺省端口 1414。

```
STRMQMLSR MQMNAME(QMGRB1)
```

现在，您已准备好在 SYSTEMA 和 SYSTEMB 之间发送测试消息。使用提供的其中一个样本，将一系列消息放入 SYSTEMA 上的远程队列。

通过使用命令 **STRMQMCHL** 或使用命令 **WRKMQMCHL** 并针对发送方通道输入启动请求 (选项 14)，在 SYSTEMA 上启动通道。

通道应进入 RUNNING 状态，并将消息发送到 SYSTEMB 上的队列 TARGETB。

通过发出以下命令来检查消息：

```
WRKMQMMMSG QNAME(TARGETB) MQMNAME(QMGRB1).
```

IBM i 上的样本资源定义

此样本包含 AMQSAMP4 样本 IBM i CL 程序。

```

/*****/
/*
/* Program name: AMQSAMP4
/*
/* Description: Sample CL program defining MQM queues
/* to use with the sample programs
/* Can be run, with changes as needed, after
/* starting the MQM
/*
/* <N_OCO_COPYRIGHT> */
/* Licensed Materials - Property of IBM
/*
/* 63H9336
/* (c) Copyright IBM Corp. 1993, 2023. All Rights Reserved.
/*
/* US Government Users Restricted Rights - Use, duplication or
/* disclosure restricted by GSA ADP Schedule Contract with
/* IBM Corp.
/* <NOC_COPYRIGHT> */
/*
/*
/*****/
/* Function:
/*
/* AMQSAMP4 is a sample CL program to create or reset the
/* MQI resources to use with the sample programs.
/*
/* This program, or a similar one, can be run when the MQM
/* is started - it creates the objects if missing, or resets
/* their attributes to the prescribed values.
/*
/*
/*
/* Exceptions signaled: none
/* Exceptions monitored: none
/*
/* AMQSAMP4 takes a single parameter, the Queue Manager name
/*
/*****/
QSYS/PGM PARM(&QMGRNAME)

/*****/
/* Queue Manager Name Parameter
/*****/

```



```

QSYS/DCL VAR(&QMGRNAME) TYPE(*CHAR)

/*****/
/*   EXAMPLES OF DIFFERENT QUEUE TYPES                               */
/*   */                                                              */
/*   Create local, alias and remote queues                          */
/*   */                                                              */
/*   Uses system defaults for most attributes                      */
/*   */                                                              */
/*****/
/* Create a local queue */
CRTMQMQ QNAME('SYSTEM.SAMPLE.LOCAL')      +
MQMNAME(&QMGRNAME)                        +
QTYPE(*LCL) REPLACE(*YES)                +
+
TEXT('Sample local queue') /* description */+
SHARE(*YES)                /* Shareable */+
DFTMSGPST(*YES) /* Persistent messages OK */

/* Create an alias queue */
CRTMQMQ QNAME('SYSTEM.SAMPLE.ALIAS')      +
MQMNAME(&QMGRNAME)                        +
QTYPE(*ALS) REPLACE(*YES)                +
+
TEXT('Sample alias queue')                +
DFTMSGPST(*YES) /* Persistent messages OK */+
TGTOQNAME('SYSTEM.SAMPLE.LOCAL')

/* Create a remote queue - in this case, an indirect reference */
/* is made to the sample local queue on OTHER queue manager */
CRTMQMQ QNAME('SYSTEM.SAMPLE.REMOTE')      +
MQMNAME(&QMGRNAME)                        +
QTYPE(*RMT) REPLACE(*YES)                +
+
TEXT('Sample remote queue') /* description */+
DFTMSGPST(*YES) /* Persistent messages OK */+
RMTQNAME('SYSTEM.SAMPLE.LOCAL')          +
RMTMQMNAME(OTHER) /* Queue is on OTHER */

/* Create a transmission queue for messages to queues at OTHER */
/* By default, use remote node name */
CRTMQMQ QNAME('OTHER') /* transmission queue name */+
MQMNAME(&QMGRNAME)      +
QTYPE(*LCL) REPLACE(*YES) +
TEXT('Transmission queue to OTHER') +
USAGE(*TMQ) /* transmission queue */

/*****/
/*   SPECIFIC QUEUES AND PROCESS USED BY SAMPLE PROGRAMS          */
/*   */                                                              */
/*   Create local queues used by sample programs                  */
/*   Create MQI process associated with sample initiation queue    */
/*   */                                                              */
/*****/
/* General reply queue */
CRTMQMQ QNAME('SYSTEM.SAMPLE.REPLY')      +
MQMNAME(&QMGRNAME)                        +
QTYPE(*LCL) REPLACE(*YES)                +
+
TEXT('General reply queue')                +
DFTMSGPST(*NO) /* Not Persistent */

/* Queue used by AMQSINQ4 */
CRTMQMQ QNAME('SYSTEM.SAMPLE.INQ')        +
MQMNAME(&QMGRNAME)                        +
QTYPE(*LCL) REPLACE(*YES)                +
+
TEXT('Queue for AMQSINQ4')                +
SHARE(*YES)                /* Shareable */+
DFTMSGPST(*NO) /* Not Persistent */+
+
TRGENBL(*YES) /* Trigger control on */+
TRGTYPE(*FIRST) /* Trigger on first message */+
PRCNAME('SYSTEM.SAMPLE.INQPROCESS')      +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/* Queue used by AMQSSET4 */
CRTMQMQ QNAME('SYSTEM.SAMPLE.SET')        +
MQMNAME(&QMGRNAME)                        +
QTYPE(*LCL) REPLACE(*YES)                +
+
TEXT('Queue for AMQSSET4')                +

```

```

SHARE(*YES)          /* Shareable */ +
DFTMSGPST(*NO)/* Not Persistent */ +
+
TRGENBL(*YES) /* Trigger control on */ +
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.SETPROCESS') +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/* Queue used by AMQSECH4 */
CRTMQMQ QNAME('SYSTEM.SAMPLE.ECHO') +
MQMNAME(&QMGRNAME) +
QTYPE(*LCL) REPLACE(*YES) +
+
TEXT('Queue for AMQSECH4') +
SHARE(*YES)          /* Shareable */ +
DFTMSGPST(*NO)/* Not Persistent */ +
+
TRGENBL(*YES) /* Trigger control on */ +
TRGTYPE(*FIRST)/* Trigger on first message*/+
PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS') +
INITQNAME('SYSTEM.SAMPLE.TRIGGER')

/* Initiation Queue used by AMQSTRG4, sample trigger process */
CRTMQMQ QNAME('SYSTEM.SAMPLE.TRIGGER') +
MQMNAME(&QMGRNAME) +
QTYPE(*LCL) REPLACE(*YES) +
TEXT('Trigger queue for sample programs')

/* MQI Processes associated with triggered sample programs */
/* */
/***** Note - there are versions of the triggered samples *****/
/***** in different languages - set APPID for these *****/
/***** process to the variation you want to trigger *****/
/* */
CRTMQMPRC PRCNAME('SYSTEM.SAMPLE.INQPROCESS') +
MQMNAME(&QMGRNAME) +
REPLACE(*YES) +
+
TEXT('Trigger process for AMQSINQ4') +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here **/ +
APPID('QMOM/AMQSINQ4') /* C */ +
/* APPID('QMOM/AMQOINQ4') /* COBOL */ +
/* APPID('QMOM/AMQ3INQ4') /* RPG - ILE */

CRTMQMPRC PRCNAME('SYSTEM.SAMPLE.SETPROCESS') +
MQMNAME(&QMGRNAME) +
REPLACE(*YES) +
+
TEXT('Trigger process for AMQSSET4') +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here **/ +
APPID('QMOM/AMQSSET4') /* C */ +
/* APPID('QMOM/AMQOSET4') /* COBOL */ +
/* APPID('QMOM/AMQ3SET4') /* RPG - ILE */

CRTMQMPRC PRCNAME('SYSTEM.SAMPLE.ECHOPROCESS') +
MQMNAME(&QMGRNAME) +
REPLACE(*YES) +
+
TEXT('Trigger process for AMQSECH4') +
ENVDATA('JOBPTY(3)') /* Submit parameter */ +
/** Select the triggered program here **/ +
APPID('QMOM/AMQSECH4') /* C */ +
/* APPID('QMOM/AMQOECH4') /* COBOL */ +
/* APPID('QMOM/AMQ3ECH4') /* RPG - ILE */

/*****
/*
/* Normal return.
/*
/*****
SNDPGMMMSG MSG('AMQSAMP4 Completed creating sample +
objects for ' *CAT &QMGRNAME)
RETURN
ENDPGM

/*****
/*
/* END OF AMQSAMP4

```

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

IBM i 管理 IBM MQ for IBM i 的替代方法

使用 CL 命令是管理 IBM MQ for IBM i 的首选方法。但是，您可以使用各种其他管理方法，包括 MQSC 命令，PCF 命令和远程管理。

通常使用 IBM i CL 命令来管理 IBM MQ for IBM i。有关这些命令的概述，请参阅第 214 页的『使用 CL 命令管理 IBM MQ for IBM i』。

您可以使用 IBM MQ 检测事件来监视队列管理器的操作。请参阅 [检测事件](#)，以获取有关 IBM MQ 检测事件以及如何使用这些事件的信息。

您可以使用以下子主题中描述的任何管理方法作为使用 IBM i CL 命令的替代方法：

IBM i IBM i 上的本地和远程管理

您可以在本地或远程管理 IBM MQ for IBM i 对象。

本地管理表示在本地系统上定义的任何队列管理器上执行管理任务。在 IBM MQ 中，您可以将此视为本地管理，因为不涉及任何 IBM MQ 通道，即，通信由操作系统管理。要执行此类型的任务，必须登录到远程系统并从该系统发出命令，或者创建可以为您发出命令的进程。

IBM MQ 支持从单点到称为远程管理的管理。远程管理包括向目标队列管理器上的 SYSTEM.ADMIN.COMMAND.QUEUE 发送可编程命令格式 (PCF) 控制消息。

有多种生成 PCF 消息的方法。这些字段为：

1. 使用 PCF 消息编写程序。请参阅第 228 页的『在 IBM i 上使用 PCF 命令进行管理』。
2. 使用 MQAI 编写程序，这将发送 PCF 消息。请参阅第 29 页的『使用 MQAI 来简化 PCF 的使用』。
3. 使用 IBM MQ for Windows 提供的 IBM MQ Explorer，这允许您使用图形用户界面 (GUI) 并生成正确的 PCF 消息。请参阅第 229 页的『将 IBM MQ Explorer 与 IBM MQ for IBM i 一起使用』。
4. 使用 **STRMQMQSC** 将命令间接发送到远程队列管理器。请参阅第 227 页的『在 IBM i 上使用 MQSC 命令进行管理』。

例如，可以发出远程命令来更改远程队列管理器上的队列定义。

某些命令不能以此方式发出，尤其是创建或启动队列管理器以及启动命令服务器。要执行此类型的任务，必须登录到远程系统并从该系统发出命令，或者创建一个可以为您发出命令的进程。

IBM i 在 IBM i 上使用 MQSC 命令进行管理

使用此信息可了解 MQSC 命令以及如何使用这些命令来管理 IBM MQ for IBM i。

IBM MQ 脚本 (MQSC) 命令以人类可读格式 (即，EBCDIC 文本) 编写。您可以使用 MQSC 命令来管理队列管理器对象，包括队列管理器本身，队列，进程定义，名称列表，通道，客户机连接通道，侦听器，服务，主题和认证信息对象。

使用 **STRMQMQSC** IBM MQ CL 命令向队列管理器发出 MQSC 命令。此方法仅是批处理方法，从服务器库系统中的源物理文件获取其输入。此源物理文件的缺省名称为 QMQSC。



注意: 请勿将 QTEMP 库用作 STRMQMQSC 的源库，因为 QTEMP 库的使用受到限制。必须使用其他库作为命令的输入文件。

IBM MQ for IBM i 未提供名为 QMQSC 的源文件。要处理 MQSC 命令，必须通过发出以下命令在所选库中创建 QMQSC 源文件：

```
CRTSRCPF FILE(MYLIB/QMQSC) RCDLEN(240) TEXT('IBM MQ - MQSC Source')
```

MQSC 源保存在此源文件中的成员中。要使用成员，请输入以下命令：

现在，您可以添加新成员并维护现有成员

您还可以通过发出 RUNMQSC 或以下命令以交互方式输入 MQSC 命令：

1. 输入队列管理器名称并按 Enter 键以访问 **WRKMQM** 结果面板。
2. 在此面板上选择 F23=More options。
3. 针对第 228 页的图 32 中显示的面板上的活动队列管理器选择选项 26。

要结束此类 MQSC 会话，请输入 end。

第 228 页的图 32 是从 MQSC 命令文件中抽取的，其中显示 MQSC 命令 (DEFINE QLOCAL) 及其属性。

```
.
.
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) REPLACE +
DESCR(' ') +
PUT(ENABLED) +
DEFPRTY(0) +
DEFPSIST(NO) +
GET(ENABLED) +
MAXDEPTH(5000) +
MAXMSGL(1024) +
DEFSOPT(SHARED) +
NOHARDENBO +
USAGE(NORMAL) +
NOTRIGGER;
.
.
```

图 32: 从 MQSC 命令文件 *myprog.in* 中抽取

为了在 IBM MQ 环境中实现可移植性，请将 MQSC 命令文件中的行长度限制为 72 个字符。加号指示命令在下一行继续执行。

MQSC 中指定的对象属性在本节中以大写形式显示 (例如，RQMNAME)，尽管它们不区分大小写。

注：

1. MQSC 文件的格式不取决于其在文件系统中的位置。
2. MQSC 属性名称限制为 8 个字符。
3. MQSC 命令在其他平台 (包括 z/OS) 上可用。

有关每个 MQSC 命令及其语法的描述，请参阅第 10 页的『脚本 (MQSC) 命令』。

IBM i 在 IBM i 上使用 PCF 命令进行管理

IBM MQ 可编程命令格式 (PCF) 命令的目的是允许将管理任务编程到管理程序中。通过这种方式，您可以从程序创建队列和进程定义，以及更改队列管理器。

PCF 命令涵盖 MQSC 命令提供的相同功能范围。但是，与 MQSC 命令不同，PCF 命令及其应答不是您可以读取的文本格式。

您可以编写程序以从单个节点向网络中的任何队列管理器发出 PCF 命令。这样，您既可以集中管理任务，也可以自动执行管理任务。

每个 PCF 命令都是嵌入在 IBM MQ 消息的应用程序数据部分中的数据结构。使用 MQI 函数 MQPUT 以与任何其他消息相同的方式将每个命令发送到目标队列管理器。接收消息的队列管理器上的命令服务器将其解释为命令消息并运行命令。要获取应答，应用程序发出 MQGET 调用，并在另一数据结构中返回应答数据。然后，应用程序可以处理应答并相应地执行操作。

简而言之，这些是应用程序员为创建 PCF 命令消息而必须指定的一些内容：

消息描述符

这是标准 IBM MQ 消息描述符，其中：

- 消息类型 (*MsgType*) 为 MQMT_REQUEST。
- 消息格式 (*Format*) 为 MQFMT_ADMIN。

应用程序数据

包含包含 PCF 头的 PCF 消息，其中：

- PCF 消息类型 (*Type*) 指定 MQCFT_COMMAND。
- 命令标识指定命令，例如 *Change Queue* (MQCMD_CHANGE_Q)。

转义 PCF 是在消息文本中包含 MQSC 命令的 PCF 命令。您可以使用 PCF 将命令发送到远程队列管理器。请参阅第 29 页的『使用 MQAI 来简化 PCF 的使用』，以了解更多信息。

有关 PCF 数据结构及其实现方式的完整描述，请参阅 [命令和响应的结构](#)。

IBM i 将 IBM MQ Explorer 与 IBM MQ for IBM i 一起使用

使用此信息可通过 IBM MQ Explorer 来管理 IBM MQ for IBM i。

IBM MQ for Windows (x86 平台) 和 IBM MQ for Linux (x86 和 x86-64 平台) 提供称为 IBM MQ Explorer 的管理接口，以执行管理任务作为使用 CL，控制或 MQSC 命令的替代方法。

IBM MQ Explorer 允许您通过将 IBM MQ Explorer 指向您感兴趣的队列管理器和集群，从运行 Windows (x86 平台) 或 Linux (x86 和 x86-64 平台) 的计算机对网络执行本地或远程管理。

通过 IBM MQ Explorer，您可以：

- 启动和停止队列管理器 (仅在本地机器上)。
- 定义，显示和变更 IBM MQ 对象 (例如，队列，主题和通道) 的定义。
- 浏览队列上的消息。
- 启动和停止通道。
- 查看有关通道的状态信息。
- 查看集群中的队列管理器。
- 检查以查看哪些应用程序，用户或通道打开了特定队列。
- 使用 "新建集群" 向导创建新的队列管理器集群。
- 使用 "将队列管理器添加到集群" 向导将队列管理器添加到集群。
- 管理认证信息对象，与传输层安全性 (TLS) 通道安全性配合使用。

通过使用在线指导，您可以：

- 定义和控制各种资源，包括队列管理器，队列，通道，进程定义，客户机连接通道，侦听器，主题，服务，名称列表和集群。
- 启动或停止队列管理器及其关联的进程。
- 在您的工作站上或从其他工作站查看队列管理器及其关联的对象。
- 检查队列管理器、集群和通道的状态。

在尝试使用 IBM MQ Explorer 来管理服务器上的 IBM MQ 之前，请确保满足以下需求。检查：

1. 正在对正在管理的任何队列管理器运行由 CL 命令 **STRMQMSVR** 在服务器上启动的命令服务器。
2. 对于每个远程队列管理器，都存在合适的 TCP/IP 侦听器。这是 **STRMQLSR** 命令启动的 IBM MQ 侦听器。
3. 每个远程队列管理器上都存在名为 SYSTEM.ADMIN.SVRCONN 的服务器连接通道。您必须自行创建此通道。对于正在管理的每个远程队列管理器，它都是必需的。没有它，远程管理是不可能的。
4. 验证 SYSTEM.MQEXPLORER.REPLY.MODEL 队列是否存在。

IBM i 在 IBM i 上管理命令服务器以进行远程管理

使用此信息可了解 IBM MQ for IBM i 命令服务器的远程管理。

每个队列管理器都可以有一个与其关联的命令服务器。命令服务器处理来自远程队列管理器的任何入局命令或来自应用程序的 PCF 命令。它向队列管理器提供用于处理的命令，并根据命令的来源返回完成代码或操作员消息。

对于涉及 PCF，MQAI 以及远程管理的所有管理，命令服务器都是必需的。

注：对于远程管理，必须确保目标队列管理器正在运行。否则，包含命令的消息无法离开从中发出这些命令的队列管理器。而是在为远程队列管理器提供服务的本地传输队列中对这些消息进行排队。如果可能，请避免此情况。

有单独的控制命令用于启动和停止命令服务器。您可以使用 IBM MQ Explorer 执行以下部分中描述的操作。

启动和停止命令服务器

要启动命令服务器，请使用以下 CL 命令：

```
STRMQMSVR MQMNAME('saturn.queue.manager')
```

其中，`saturn.queue.manager` 是要对其启动命令服务器的队列管理器。

要停止命令服务器，请使用下列其中一个 CL 命令：

1.

```
ENDMQMSVR MQMNAME('saturn.queue.manager') OPTION(*CNTRLD)
```

执行受控停止，其中 `saturn.queue.manager` 是要停止命令服务器的队列管理器。这是缺省选项，这意味着可以省略 `OPTION(*CNTRLD)`。

2.

```
ENDMQMSVR MQMNAME('saturn.queue.manager') OPTION(*IMMED)
```

执行立即停止，其中 `saturn.queue.manager` 是要停止命令服务器的队列管理器。

显示命令服务器的状态

对于远程管理，请确保目标队列管理器上的命令服务器正在运行。如果它未在运行，那么无法处理远程命令。包含命令的任何消息都将在目标队列管理器的命令队列 `SYSTEM.ADMIN.COMMAND.QUEUE` 中排队。

要显示在此处调用 `saturn.queue.manager` 的队列管理器的命令服务器的状态，CL 命令为：

```
DSPMQMSVR MQMNAME('saturn.queue.manager')
```

在目标机器上发出此命令。如果命令服务器正在运行，那么将显示 [第 231 页的图 33](#) 中显示的面板：

```
Display MQM Command Server (DSPMQMCSVR)

Queue manager name . . . . . > saturn.queue.manager
MQM Command Server Status. . . . > RUNNING

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys
```

图 33: 显示 MQM 命令服务器面板

IBM i IBM i 的工作管理

此信息描述了 IBM MQ 处理工作请求的方式，并详细说明了可用于对与 IBM MQ 关联的作业进行优先级划分和控制的选项。

警告

除非您完全了解 IBM i 和 IBM MQ 工作管理的概念，否则请勿变更 IBM MQ 工作管理对象。

可以在 IBM i 产品文档中的 [工作管理](#) 下找到有关子系统和作业描述的其他信息。请特别注意有关 [启动作业](#) 和 [批处理作业](#) 的部分。

IBM MQ for IBM i 合并了 IBM i UNIX 环境和 IBM i 线程。请勿对 Integrated File System (IFS) 中的对象进行任何更改。

在正常操作期间，IBM MQ 队列管理器会启动许多批处理作业以执行不同的任务。缺省情况下，这些批处理作业在安装 IBM MQ 时创建的 QMQM 子系统中运行。

工作管理是指定制 IBM MQ 任务以从系统获取最佳性能或使管理更简单的过程。

例如，您可以：

- 更改作业的运行优先级，使一个队列管理器比另一个队列管理器更具响应性。
- 将多个作业的输出重定向到特定输出队列。
- 使特定类型的所有作业在特定子系统中运行。
- 将错误隔离到子系统。

通过创建或更改与 IBM MQ 作业关联的作业描述来执行工作管理。您可以为以下对象配置工作管理：

- 整个 IBM MQ 安装。
- 个别队列管理器。
- 个别队列管理器的个别作业。

IBM i IBM i 的 IBM MQ 任务

这是 IBM MQ for IBM i 作业的表以及每个作业的简要描述。

当队列管理器正在运行时，您会看到以下部分或全部批处理作业在 IBM MQ 系统中的 QMQM 用户概要文件下运行。在 [第 232 页的表 19](#) 中简要描述了这些作业。

您可以使用 [使用队列管理器 \(WRKMQM\)](#) 面板上的选项 22 来查看连接到队列管理器的所有作业。您可以使用 WRKMQMLSR 命令来查看侦听器。

| 表 19: IBM MQ 个任务。 | |
|-------------------|---------------------------------------------------------------|
| 作业名 | 函数 |
| AMQALMPX | 定期获取日志检查点的检查点处理器。 |
| AMQZMUC0 | 实用程序管理器。此作业执行关键队列管理器实用程序，例如日志链管理器。 |
| AMQZXMA0 | 这是队列管理器启动的第一个作业的执行控制器。它处理 MQCONN 请求，并启动代理进程以处理 IBM MQ API 调用。 |
| AMQZFUMA | 对象权限管理器 (OAM)。 |
| AMQZLAA0 | 为使用 MQCNO_STANDARD_BINDING 连接到队列管理器的应用程序执行大部分工作的队列管理器代理程序。 |
| AMQZLSA0 | 队列管理器代理程序。 |
| AMQZMUFO | 实用程序管理器 |
| AMQZMGRO | 进程控制器。此作业用于启动和管理侦听器和服务。 |
| AMQZMUR0 | 实用程序管理器。此作业执行关键队列管理器实用程序，例如日志链管理器。 |
| AMQFQPUB | 已排队的发布/预订守护程序。 |
| AMQFCXBA | 代理工作程序作业。 |
| 运行 MQBRK | 代理控制作业。 |
| AMQRMPPA | 通道进程池作业。 |
| AMQCRSTA | TCP/IP 调用的通道响应程序。 |
| AMQCRS6B | LU62 接收方通道和客户机连接 (请参阅注释)。 |
| AMQRRMFA | 集群的存储库管理器。 |
| AMQCLMAA | 非线程 TCP/IP 侦听器。 |
| AMQPCSEA | PCF 命令处理器，用于处理 PCF 和远程管理请求。 |
| runmqtrm | 触发监视器。 |
| RUNMQDLQ | 死信队列处理程序。 |
| 运行 MQCHI | 通道启动程序。 |
| 运行 MQCHL | 为每个发送方通道启动的发送方通道作业。 |
| 运行 MQLSR | 线程 TCP/IP 侦听器。 |
| AMQRCMLA | 通道 MQSC 和 PCF 命令处理器。 |

注: LU62 接收方作业在通信子系统中运行，并从用于启动作业的路由和通信条目中获取其运行时属性。请参阅 [启动的结束 \(接收方\)](#) 以获取更多信息。

IBM i 上的工作管理对象

安装 IBM MQ 时，将在 QMQM 库中提供各种对象以帮助进行工作管理。这些对象是 IBM MQ 作业在其自己的子系统中运行所必需的对象。

为两个 IBM MQ 批处理作业提供了样本作业描述。如果没有为 IBM MQ 作业提供特定作业描述，那么它将使用缺省作业描述 QMQMJOBDD 运行。

安装 IBM MQ 时提供的工作管理对象在 [第 233 页的表 20](#) 中列出，为队列管理器创建的对象在 [第 233 页的表 21](#) 中列出

注: 可以在 QMQM 库中找到工作管理对象，也可以在队列管理器库中找到队列管理器对象。

| 名称 | 类型 | 描述 |
|-----------|------|---------------------------------|
| AMQALMPX | *JOB | 检查点进程使用的作业描述 |
| AMQZLAA0 | *JOB | IBM MQ 代理程序进程使用的作业描述 |
| AMQZLSA0 | *JOB | 隔离的绑定队列管理器代理程序 |
| AMQZXMA0 | *JOB | IBM MQ 执行控制器使用的作业描述 |
| QMQM | *SBS | 运行所有 IBM MQ 作业的子系统 |
| QMQM | *JOB | 连接到提供的子系统的作业队列 |
| QMQMJOB | *JOB | 缺省 IBM MQ 作业描述, 在没有作业的特定作业描述时使用 |
| QMQMMSG | *MSG | IBM MQ 作业的缺省消息队列。 |
| QMQMRUN20 | *CLS | 高优先级 IBM MQ 作业的类描述 |
| QMQMRUN35 | *CLS | 中等优先级 IBM MQ 作业的类描述 |
| QMQMRUN50 | *CLS | 低优先级 IBM MQ 作业的类描述 |

| 名称 | 类型 | 描述 |
|------------|---------|---------------------------------------------------------------------------------|
| AMQA000000 | *JRNRCV | 本地日志接收器 |
| AMQAJRN | *JRN | 本地日志 |
| AMQJRNINF | *USRSPC | 使用队列管理器的启动和介质恢复所需的最新日志接收器更新的用户空间。应用程序可以查询此用户空间, 以确定哪些日志接收器需要归档以及哪些日志接收器可以安全地删除。 |
| AMQAJRNMSG | *MSG | 本地日志消息队列 |
| AMQCRC6B | *PGM | 启动 LU6.2 连接的程序 |
| AMQRFOLD | *FILE | 已迁移队列管理器通道定义文件 |
| QMQMMSG | *MSG | 队列管理器消息队列 |

IBM i IBM MQ 如何在 IBM i 上使用工作管理对象

本信息描述了 IBM MQ 使用工作管理对象的方式, 并提供了配置示例。



注意: 请勿更改 QMQM 子系统内的作业队列条目设置, 以按优先级限制子系统中允许的作业数。如果您尝试执行此操作, 那么可以在提交基本 IBM MQ 作业后将其停止运行, 并导致队列管理器启动失败。

要了解如何配置工作管理, 必须首先了解 IBM MQ 如何使用作业描述。

用于启动作业的作业描述控制作业的许多属性。例如:

- 作业在其中排队的作业队列以及作业在其中运行的子系统。
- 用于启动作业和作业用于其运行时参数的类的路由数据。
- 作业用于打印文件的输出队列。

可以通过三个步骤来考虑启动 IBM MQ 作业的过程:

1. IBM MQ 选择作业描述。

IBM MQ 使用以下方法来确定要用于批处理作业的作业描述:

- a. 在队列管理器库中查找与作业同名的作业描述。有关队列管理器库的更多详细信息，请参阅 [了解 IBM MQ for IBM i 队列管理器库名](#)。
- b. 在队列管理器库中查找缺省作业描述 QMQMJOB。D。
- c. 在 QMQM 库中查找与作业同名的作业描述。
- d. 在 QMQM 库中使用缺省作业描述 QMQMJOB。D。

2. 将作业提交到作业队列。

缺省情况下，已设置 IBM MQ 随附的作业描述，以将作业放入库 QMQM 中的作业队列 QMQM。QMQM 作业队列连接到提供的 QMQM 子系统，因此缺省情况下，作业开始在 QMQM 子系统中运行。

3. 作业进入子系统并完成路由步骤。

当作业进入子系统时，作业描述上指定的路由数据用于查找作业的路由项。

路由数据必须与 QMQM 子系统中定义的其中一个路由条目匹配，这将定义作业所使用的所提供类 (QMQRUN20, QMQRUN35 或 QMQRUN50)。

注：如果 IBM MQ 作业未显示为正在启动，请确保子系统正在运行并且未挂起作业队列，

如果已修改 IBM MQ 工作管理对象，请确保所有内容都正确关联。例如，如果在作业描述上指定 QMQM/QMQM 以外的作业队列，请确保对子系统 (即 QMQM) 执行 ADDJOBQE。

您可以使用以下工作表作为示例，为 [第 232 页的表 19](#) 中记录的每个作业创建作业描述：

```

What is the queue manager library name? -----
Does job description AMQZXMA0 exist in the queue manager library? Yes   No
Does job description QMQMJOB exist in the queue manager library? Yes   No
Does job description AMQZXMA0 exist in the QMQM library?           Yes   No
Does job description QMQMJOB exist in the QMQM library?           Yes   No
  
```

如果对所有这些问题回答 "否"，请在 QMQM 库中创建全局作业描述 QMQMJOB。D。

IBM MQ 消息队列

将在每个队列管理器库中创建 IBM MQ 消息队列 QMQMMSG。当队列管理器作业结束时，操作系统消息将发送到此队列，而 IBM MQ 将消息发送到此队列。例如，报告启动时需要哪些日志接收器。将此消息队列中的消息数保持在可管理的大小，以便更易于监视。

IBM i 的缺省系统示例

这些示例显示在队列管理器启动时提交某些标准作业时，未修改的 IBM MQ 安装如何工作。

首先，启动 AMQZXMA0 执行控制器作业。

1. 针对队列管理器 TESTQM 发出 **STRMQM** 命令。
2. IBM MQ 搜索队列管理器库 QMTESTQM，首先搜索作业描述 AMQZXMA0，然后搜索作业描述 QMQMJOB。D。

这些作业描述都不存在，因此 IBM MQ 将在产品库 QMQM 中查找作业描述 AMQZXMA0。此作业描述存在，因此用于提交作业。

3. 作业描述使用 IBM MQ 缺省作业队列，因此作业将提交到作业队列 QMQM/QMQM。
4. AMQZXMA0 作业描述上的路由数据为 QMQRUN20，因此系统会在子系统路由条目中搜索与该数据匹配的路由条目。

缺省情况下，序号为 9900 的路由条目具有与 QMQRUN20 匹配的比较数据，因此将使用该路由条目上定义的类型 (也称为 QMQRUN20) 来启动作业。

5. QMQM/QMQRUN20 类的运行优先级设置为 20，因此 AMQZXMA0 作业在子系统 QMQM 中以与系统上大多数交互式作业相同的优先级运行。

接下来，AMQALMPX 检查点进程作业将启动。

1. IBM MQ 搜索队列管理器库 QMTESTQM，首先搜索作业描述 AMQALPMX，然后搜索作业描述 QMQMJOB。D。

这些作业描述都不存在，因此 IBM MQ 会在产品库 QMQM 中查找作业描述 AMQALMPX 和 QMQMJOB。D。

作业描述 AMQALMPX 不存在，但 QMQMJOB。D 不存在，因此 QMQMJOB。D 用于提交作业。

注: QMQMJOB。D 作业描述始终用于没有其自己的作业描述的 IBM MQ 作业。

2. 作业描述使用 IBM MQ 缺省作业队列，因此作业将提交到作业队列 QMQM/QMQM。
3. QMQMJOB。D 作业描述上的路由数据为 QMQMRUN35，因此系统将在子系统路由条目中搜索与该数据匹配的路由条目。
缺省情况下，序号为 9910 的路由条目具有与 QMQMRUN35 匹配的比较数据，因此将使用该路由条目上定义的类 (也称为 QMQMRUN35) 来启动作业。
4. QMQM/QMQMRUN35 类的运行优先级设置为 35，因此 AMQALMPX 作业在子系统 QMQM 中运行，其优先级低于系统上的大多数交互式作业，但高于大多数批处理作业。

IBM i 在 IBM i 上配置工作管理示例

使用此信息可了解如何更改和创建 IBM MQ 作业描述以更改 IBM MQ 作业的运行属性。

IBM MQ 工作管理灵活性的关键在于 IBM MQ 搜索作业描述的两层方式:

- 如果在队列管理器库中创建或更改作业描述，那么这些更改将覆盖 QMQM 中的全局作业描述，但这些更改是本地的，并且仅影响该特定队列管理器。
- 如果在 QMQM 库中创建或更改全局作业描述，那么这些作业描述将影响系统上的所有队列管理器，除非针对个别队列管理器在本地覆盖。

1. 以下示例提高了单个队列管理器的通道控制作业的优先级。

要使存储库管理器和通道启动程序作业 AMQRRMFA 和 RUNMQCHI 尽快针对队列管理器 TESTQM 运行，请执行以下步骤:

- a. 使用要在队列管理器库中控制的 IBM MQ 进程的名称创建 QMQM/QMQMJOB。D 作业描述的本地重复项。例如:

```
CRTDUPOBJ OBJ(QMQMJOB。D) FROMLIB(QMQM) OBJTYPE(*JOB) TOLIB(QMTESTQM)
NEWOBJ (RUNMQCHI)
CRTDUPOBJ OBJ(QMQMJOB。D) FROMLIB(QMQM) OBJTYPE(*JOB) TOLIB(QMTESTQM)
NEWOBJ (AMQRRMFA)
```

- b. 更改作业描述上的路由数据参数，以确保作业使用 QMQMRUN20 类。

```
CHGJOB JOB(QMTESTQM/RUNMQCHI) RTGDTA('QMQMRUN20')
CHGJOB JOB(QMTESTQM/AMQRRMFA) RTGDTA('QMQMRUN20')
```

现在，队列管理器 TESTQM 的 AMQRRMFA 和 RUNMQCHI 作业:

- 在队列管理器库中使用新的本地作业描述
 - 以优先级 20 运行，因为当作业进入子系统时将使用 QMQMRUN20 类。
2. 以下示例为 QMQM 子系统定义新的运行优先级类。

- a. 通过发出以下命令，在 QMQM 库中创建重复的类，以允许其他队列管理器访问该类:

```
CRTDUPOBJ OBJ(QMQMRUN20) FROMLIB(QMQM) OBJTYPE(*CLS) TOLIB(QMQM)
NEWOBJ (QMQMRUN10)
```

- b. 通过发出以下命令，更改类以具有新的运行优先级:

```
CHGCLS CLS(QMQM/QMQMRUN10) RUNPTY(10)
```

- c. 通过发出以下命令将新的类定义添加到子系统:

```
ADDRTGE SBSD(QMQM/QMQM) SEQNBR(8999) CMPVAL('QMQRUN10') PGM(QSYS/QCMD)
CLS(QMQM/QMQRUN10)
```

注: 您可以为路由序号指定任何数字值, 但这些值必须按顺序排列。此序号告诉子系统搜索路由项以查找路由数据匹配的顺序。

- d. 通过发出以下命令, 更改本地或全局作业描述以使用新的优先级类:

```
CHGJOB JOB(QMQLibname/QMQMJOB) RTGDTA('QMQRUN10')
```

现在, 与 QMLibraryname 关联的所有队列管理器作业都使用运行优先级 10。

3. 以下示例在其自己的子系统中运行队列管理器

要使队列管理器 TESTQM 的所有作业在 QBATCH 子系统中运行, 请执行以下步骤:

- a. 使用命令在队列管理器库中创建 QMQM/QMQMJOB 作业描述的本地副本

```
CRTDUPOBJ OBJ(QMQMJOB) FROMLIB(QMQM) OBJTYPE(*JOB) TOLIB(QMTESTQM)
```

- b. 更改作业描述上的作业队列参数以确保作业使用 QBATCH 作业队列。

```
CHGJOB JOB(QMTESTQM/QMQMJOB) JOBQ(*LIBL/QBATCH)
```

注: 作业队列与子系统描述相关联。如果发现作业停留在作业队列上, 请验证是否在 SBSD 上定义了作业队列定义。对子系统使用 DSPSBSD 命令, 并采用选项 6, 作业队列项。

现在, 队列管理器 TESTQM 的所有作业:

- 在队列管理器库中使用新的本地缺省作业描述
- 提交到作业队列 QBATCH。

要确保正确路由作业并对其划分优先级, 请执行以下操作:

- 为子系统 QBATCH 中的 IBM MQ 作业创建路由项, 或者
- 依赖于调用 QCMD 的全捕获路由条目, 而不考虑使用的路由数据。

仅当作业队列 QBATCH 的最大活动作业选项设置为 *NOMAX 时, 此选项才有效。系统缺省值为 1。

4. 以下示例创建另一个 IBM MQ 子系统

- a. 通过发出以下命令在 QMQM 库中创建重复的子系统:

```
CRTDUPOBJ OBJ(QMQM) FROMLIB(QMQM) OBJTYPE(*SBSD) TOLIB(QMQM) NEWOBJ(QMQM2)
```

- b. 通过发出以下命令来除去 QMQM 作业队列:

```
RMVJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM)
```

- c. 通过发出以下命令为子系统创建新的作业队列:

```
CRTJOBQ JOBQ(QMQM/QMQM2) TEXT('Job queue for IBM MQ Queue Manager')
```

- d. 通过发出以下命令向子系统添加作业队列项:

```
ADDJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM2) MAXACT(*NOMAX)
```

- e. 通过发出以下命令在队列管理器库中创建重复的 QMQMJOB:

```
CRTDUPOBJ OBJ(QMQMJOB) FROMLIB(QMQM) OBJTYPE(*JOB) TOLIB(QMlibraryname)
```

- f. 通过发出以下命令，更改作业描述以使用新的作业队列：

```
CHGJOB JOB(QMlibraryname/QMQMJOB) JOBQ(QMQM/QMQM2)
```

- g. 通过发出以下命令来启动子系统：

```
STRSBS SBS(QMQM/QMQM2)
```

注：

- 可以在任何库中指定子系统。如果由于任何原因重新安装了产品，或者替换了 QMQM 库，那么将除去您所做的任何更改。
 - 与 QMlibraryname 关联的所有队列管理器作业现在都在子系统 QMQM2 下运行。
5. 以下示例收集作业类型的所有输出。

要将多个队列管理器的所有检查点进程 AMQALMPX 作业日志收集到单个输出队列上，请执行以下步骤：

- a. 创建输出队列，例如

```
CRTOUTQ OUTQ(MYLIB/CHKPTLOGS)
```

- b. 使用要控制的 IBM MQ 进程的名称创建 QMQM/QMQMJOB 作业描述的全局副本，例如

```
CRTDUPOBJ OBJ(QMQMJOB) FROMLIB(QMQM) OBJTYPE(*JOB) NEWOBJ(AMQALMPX)
```

- c. 更改作业描述上的输出队列参数以指向新的输出队列，并更改作业记录级别，以便将所有消息写入作业记录。

```
CHGJOB JOB(QMQM/AMQALMPX) OUTQ(MYLIB/CHKPTLOGS) LOG(4 00 *SECLVL)
```

对于所有队列管理器，所有 IBM MQ AMQALMPX 作业都使用新的全局 AMQALMPX 作业描述，前提是本地队列管理器库中没有本地覆盖作业描述。

现在，这些作业的所有作业记录假脱机文件都将写入库 MYLIB 中的输出队列 CHKPTLOGS。

注：

- 仅当 QPJOBLOG 或任何打印文件的输出队列参数值为 *JOB 时，上述示例才有效。在前面的示例中，QSYS/QPDJOBLOG 文件需要 OUTQ 设置为 *JOB。
- 要更改系统打印文件，请使用 CHGPRTF 命令。例如：

```
CHGPRTF PRTF(QJOBLOG) OUTQ(*JOB)
```

*JOB 选项指示必须使用您的作业描述。

- 您可以将与 IBM MQ 作业关联的任何假脱机文件发送到特定输出队列。但是，请验证所使用的打印文件是否具有适合于 OUTQ 参数的值。

IBM i 上的可用性，备份，恢复和重新启动

使用此信息来了解 IBM MQ for IBM i 如何使用 IBM i 日志记录支持来帮助其备份和复原策略。

在阅读本部分之前，您必须熟悉标准 IBM i 备份和恢复方法，以及在 IBM i 上使用日志及其关联的日志接收器。有关这些主题的信息，请参阅 [备份和恢复](#)。

要了解备份和恢复策略，首先需要了解 IBM MQ for IBM i 如何在 IBM i 文件系统和集成文件系统 (IFS) 中组织其数据。

IBM MQ for IBM i 将其数据保存在每个队列管理器实例的单个库中，并保存在 IFS 文件系统中的流文件中。

特定于队列管理器的库包含日志，日志接收器和控制队列管理器的工作管理所需的对象。IFS 目录和文件包含 IBM MQ 配置文件，IBM MQ 对象的描述及其包含的数据。

对这些对象的每个更改(可跨系统故障恢复)都将记录在日志中，之前会将其应用于相应的对象。这就产生了这样的变化可以通过重放日志中记录的信息来恢复。

您可以将 IBM MQ for IBM i 配置为使用不同服务器上的多个队列管理器实例，以提供更高的队列管理器可用性，并在发生服务器或队列管理器故障时加快恢复速度。

IBM i 上的队列管理器日志

使用此信息可了解 IBM MQ for IBM i 如何在其操作中使用日志来控制对本地对象的更新。

每个队列管理器库都包含该队列管理器的日志，该日志的名称为 QM GRLIB/AMQ A JRN，其中 QM GRLIB 是队列管理器库的名称，而 A 是单个实例队列管理器的唯一字母 A。

QM GRLIB 采用名称 QM，后跟唯一格式的队列管理器名称。例如，名为 TEST 的队列管理器具有名为 QMTEST 的队列管理器库。使用 **CRTMQM** 命令创建队列管理器时，可以指定队列管理器库。

日志具有包含要记录的信息的关联日志接收器。接收器是只能附加信息的对象，最终将填充这些对象。

日志接收器使用具有过时信息的宝贵磁盘空间。但是，您可以将信息放在永久存储器中，以最大程度地减少此问题。在任何特定时间，都有一个日志接收器连接到日志。如果日志接收器达到其预定阈值大小，那么将拆离该日志接收器并将其替换为新的日志接收器。使用 **CRTMQM** 和 **THRESHOLD** 参数创建队列管理器时，可以指定日志接收器的阈值。

与本地 IBM MQ for IBM i 日志关联的日志接收器存在于每个队列管理器库中，并采用如下命名约定：

```
AMQ Arnnnnn
```

其中：

a

是字母 A-Z。对于单实例队列管理器，它是 A。它因多实例队列管理器的不同实例而异。

nnnnn

是序列中下一个日志的十进制 00000 to 99999，按 1 递增。

r

是十进制 0 to 9，每次恢复接收器时按 1 递增。

日记帐的顺序基于日期。但是，下一个期刊的命名是基于以下规则：

1. AMQA_rnnnnn 转至 AMQA_r(nnnnn+1)，nnnnn 在到达 99999 时回绕。例如，AMQA099999 转至 AMQA000000，AMQA999999 转至 AMQA900000。
2. 如果已存在具有规则 1 生成的名称的日志，那么会将消息 CPI70E3 发送到 QSYSOPR 消息队列，并且自动接收方切换将停止。
将继续使用当前连接的接收器，直到您调查问题并手动连接新的接收器为止。
3. 如果序列中没有新名称可用(即，所有可能的日志名称都在系统上)，那么需要执行以下两个操作：
 - a. 不再需要删除日记帐(请参阅第 242 页的『IBM i 上的日志管理』)。
 - b. 使用 (**RCDMQMIMG**) 将日志更改记录到最新的日志接收器中 然后重复上一步。这允许复用旧的日志接收器名称。

AMQAJRN 日志使用 MNGRCV(*SYSTEM) 选项使操作系统能够在达到阈值时自动更改日志接收器。有关系统如何管理接收器的更多信息，请参阅 IBM i 备份和恢复。

日志接收器的缺省阈值为 100,000 KB。您可以在创建队列管理器时将此值设置为更大的值。

LogReceiverSize 属性的初始值将写入 mqs.ini 文件的 LogDefaults 节。

当日志接收器超出其指定阈值时，将拆离该接收器并创建新的日志接收器，从而继承先前接收器的属性。当系统自动连接新的日志接收器时，将忽略创建队列管理器后对 LogReceiverSize 或 LogASP 属性所作的更改

请参阅 [更改 IBM i 上的配置信息](#)，以获取有关配置系统的更多详细信息。

如果需要在创建队列管理器后更改日志接收器的大小，请创建新的日志接收器，并使用以下命令将其所有者设置为 QMQM：

```
CRTJRNRCV JRNRCV(QM GRLIB/AMQ Arnnnnn) THRESHOLD(xxxxxx) +
TEXT('MQM LOCAL JOURNAL RECEIVER')
CHGOBJOWN OBJ(QM GRLIB/AMQ Arnnnnn) OBJTYPE(*JRNRCV) NEWOWN(QMQM)
```

其中：

QMGRLIB

是队列管理器库的名称

a

是实例标识 (通常为 A)。

rnrrrrr

是先前描述的命名序列中的下一个日志接收器

xxxxxxx

新接收方阈值 (以 KB 为单位)

注：接收器的最大大小由操作系统控制。要检查此值，请查看 **CRTJRNRCV** 命令上的 THRESHOLD 关键字。

现在，使用以下命令将新接收器连接到 AMQAJRN 日志：

```
CHGJRN JRN(QMGRLIB/AMQ A JRN) JRNRCV(QMGRLIB/AMQ Anrrrrr)
```

有关如何管理这些日志接收器的详细信息，请参阅 [第 242 页的『IBM i 上的日志管理』](#)。

IBM i IBM i 上的队列管理器日志使用情况

使用此信息可了解 IBM MQ for IBM i 如何在其操作中使用日志来控制对本地对象的更新。

对消息队列的持久更新分两个阶段进行。首先将表示更新的记录写入日志，然后更新队列文件。

因此，日志接收器可能比队列文件更新更新。为确保重新启动处理从一致点开始，IBM MQ 使用检查点。

检查点是日志中描述的记录与队列中的记录相同的时间点。检查点本身由重新启动队列管理器所需的一系列日志记录组成。例如，检查点时处于活动状态的所有事务 (即，工作单元) 的状态。

检查点由 IBM MQ 自动生成。当队列管理器启动和关闭时，以及在记录了一定数量的操作之后，将会执行这些操作。

通过对队列管理器上的所有对象发出 RCDMQMIMG 命令并显示结果，可以强制队列管理器采用检查点，如下所示：

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME) DSPJRNDTA(*YES)
```

当队列处理更多消息时，检查点记录将与队列的当前状态不一致。

重新启动 IBM MQ 时，它将在日志中查找最新的检查点记录。此信息保存在每个检查点结束时更新的检查点文件中。检查点记录表示日志与数据之间的最新一致性点。此检查点的数据用于重建在检查点时存在的队列。当重新创建队列时，将向前播放日志以将队列恢复到系统故障或关闭之前的状态。

要了解 IBM MQ 如何使用日志，请考虑队列管理器 TEST 中名为 TESTQ 的本地队列的情况。这由 IFS 文件表示：

```
/QIBM/UserData/mqm/qmgrs/TEST/queues
```

如果将指定的消息放在此队列上，然后从队列中检索，那么所执行的操作将显示在图 [第 240 页的图 34](#) 中。

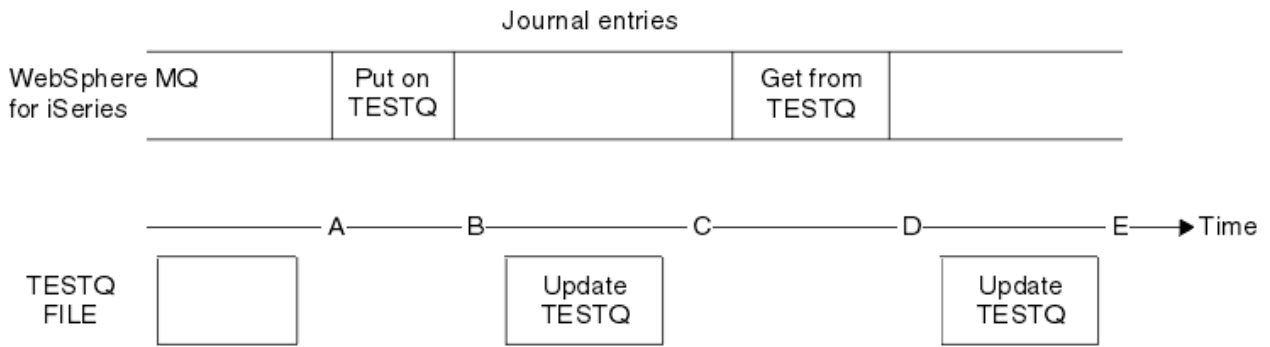


图 34: 更新 MQM 对象时的事件序列

图中显示的五点 A 到 E 表示定义以下状态的时间点:

- A** 队列的 IFS 文件表示与日志中包含的信息一致。
- B** 日志项将写入定义队列上的 Put 操作的日志。
- C** 对队列进行相应的更新。
- D** 日志项将写入定义队列中的 Get 操作的日志。
- E** 对队列进行相应的更新。

IBM MQ for IBM i 恢复功能的关键是，用户可以将 TESTQ 的 IFS 文件表示保存在时间 A，然后恢复 TESTQ 的 IFS 文件表示在时间 E，方法是复原保存的对象并从时间 A 开始重放日志中的条目。

此策略由 IBM MQ for IBM i 用于在系统故障后恢复持久消息。IBM MQ 会记住日志接收器中的特定条目，并确保在启动时从此时开始重放日志中的条目。将定期重新计算此启动条目，以便 IBM MQ 只需在下次启动时执行最低必要重放。

IBM MQ 提供对象的单独恢复。与对象相关的所有持久信息都记录在本地 IBM MQ for IBM i 日志中。任何已损坏或损坏的 IBM MQ 对象都可以根据日志中保存的信息进行完全重建。

有关系统如何管理接收方的更多信息，请参阅第 237 页的『IBM i 上的可用性，备份，恢复和重新启动』。

IBM i 上的介质映像

在 IBM i 上，介质映像是记录在日志中的 IBM MQ 对象的完整副本。某些损坏或损坏的对象可以从其介质映像中自动恢复。

持续时间较长的 IBM MQ 对象可以表示大量日志项，返回到创建该对象的位置。为避免此情况，IBM MQ for IBM i 具有对象的介质映像概念。

此介质映像是日志中记录的 IBM MQ 对象的完整副本。如果获取对象的图像，那么可以通过从此图像开始重放日志项来重建该对象。表示每个 IBM MQ 对象的重放点的日志中的条目称为其介质恢复条目。IBM MQ 跟踪以下内容:

- 每个队列管理器对象的介质恢复条目。
- 此集中最早的条目 (请参阅第 242 页的『IBM i 上的日志管理』中的错误消息 AMQ7462 以获取详细信息)。

将定期获取 *CTLG 对象和 *MQM 对象的图像，因为这些对象对于队列管理器重新启动至关重要。

在方便时拍摄其他对象的图像。缺省情况下，当使用带有参数 ENDCCTJOB (*YES) 的 ENDMQM 命令关闭队列管理器时，将获取所有对象的映像。对于非常大的队列管理器，此操作可能需要相当长的时间。如果需

要快速关闭，请使用 ENDCCTJOB (*YES) 指定参数 RCDMQMIMG (*NO)。在这种情况下，建议您在重新启动队列管理器后，使用以下命令在日志中记录完整的介质映像：

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME)
```

IBM MQ 会自动记录对象的图像，如果它找到一个方便的点，在该点可以通过日志中的一个小条目来紧凑地描述对象。但是，对于某些对象 (例如，始终包含大量消息的队列)，可能永远不会发生此情况。

使用 IBM MQ 命令 RCDMQMIMG (使您能够手动获取所选对象的映像)，而不是允许最旧的介质恢复条目的日期继续一段不必要的长周期。

从介质映像恢复

如果发现某些对象损坏或损坏，那么 IBM MQ 会自动从其介质映像中恢复这些对象。这尤其适用于作为正常队列管理器启动的一部分的特殊 *MQM 和 *CTLG 对象。如果在上次关闭队列管理器时有任何同步点事务未完成，那么受影响的任何队列也会自动恢复，以便完成启动操作。

必须使用 IBM MQ 命令 RCRMQM0BJ 手动恢复其他对象。此命令将重放日志中的条目以重新创建 IBM MQ 对象。如果 IBM MQ 对象已损坏，那么唯一有效的操作是删除该对象或通过此方法重新创建该对象。但是，请注意，无法以此方式恢复非持久消息。

IBM i IBM MQ for IBM i 上的检查点

在不同时间设置检查点，以提供已知的一致恢复起始点。

检查点进程 AMQALMPX 负责在以下位置获取检查点：

- 队列管理器启动 (STRMQM)。
- 队列管理器关闭 (ENDMQM)。
- 自上次检查点以来经过一段时间 (缺省时间段为 30 分钟)，并且自上次检查点以来已写入的日志记录的最小数目 (缺省值为 100)。
- 在写入大量日志记录之后。缺省值为 10 000。
- 在超过日志阈值大小并且已自动创建新的日志接收器之后。
- 当使用以下内容拍摄完整媒体图像时：

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(Q_MGR_NAME) DSPJRNDTA(*YES)
```

IBM i IBM MQ for IBM i 数据的备份

使用此信息来了解每个队列管理器的两种类型的 IBM MQ 备份。

对于每个队列管理器，要考虑两种类型的 IBM MQ 备份：

- 数据和日志备份。

要确保这两组数据一致，请仅在关闭队列管理器后执行此操作。

- 日志备份。

您可以在队列管理器处于活动状态时执行此操作。

对于这两种方法，都需要查找队列管理器 IFS 目录和队列管理器库的名称。您可以在 IBM MQ 配置文件 (mq.ini) 中找到这些内容。有关更多信息，请参阅 [QueueManager](#) 节。

使用以下过程来执行两种类型的备份：

特定队列管理器的数据和日志备份

注：当队列管理器正在运行时，请勿使用 "活动时保存" 请求。除非落实或回滚具有暂挂更改的所有落实定义，否则此类请求无法完成。如果在队列管理器处于活动状态时使用此命令，那么通道连接可能不会正常结束。请始终使用以下过程。

1. 使用以下命令创建空日志接收器：

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. 使用 **RCDMQMIMG** 命令为所有 IBM MQ 对象记录 MQM 映像，然后使用以下命令强制检查点:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 结束通道并确保队列管理器未在运行。如果队列管理器正在运行，请使用 **ENDMQM** 命令将其停止。
4. 通过发出以下命令来备份队列管理器库:

```
SAVLIB LIB(QMTEST)
```

5. 通过发出以下命令来备份队列管理器 IFS 目录:

```
SAV DEV(...) OBJ((' /QIBM/UserData/mqm/qmgrs/test'))
```

特定队列管理器的日志备份

因为所有相关信息都保存在日志中，只要在某个时间执行完全保存，就可以通过保存日志接收器来执行部分备份。这些记录自完全备份以来的所有更改，并通过发出以下命令来执行:

1. 使用以下命令创建空日志接收器:

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. 使用 **RCDMQMIMG** 命令为所有 IBM MQ 对象记录 MQM 映像，然后使用以下命令强制检查点:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 使用以下命令保存日志接收器:

```
SAVOBJ OBJ(AMQ*) LIB(QMTEST) OBJTYPE(*JRNRCV) .....
```

简单的备份策略是每周执行 IBM MQ 库的完全备份，并执行每日日志备份。当然，这取决于您如何为企业设置备份策略。

IBM i 上的日志管理

作为备份策略的一部分，请注意日志接收器。由于各种原因，从 IBM MQ 库中除去日志接收器很有用:

- 要释放空间; 这适用于所有日志接收器
- 提高启动时的性能 (STRMQM)
- 提高重新创建对象的性能 (RCRMQMOBJ)

在删除日志接收器之前，必须注意您具有备份副本并且不再需要日志接收器。

日志接收器在从日志中拆离并保存后，可以从队列管理器库中除去日志接收器，前提是它们可用于恢复操作(如果需要)。

第 243 页的图 35 中显示了日志管理的概念。

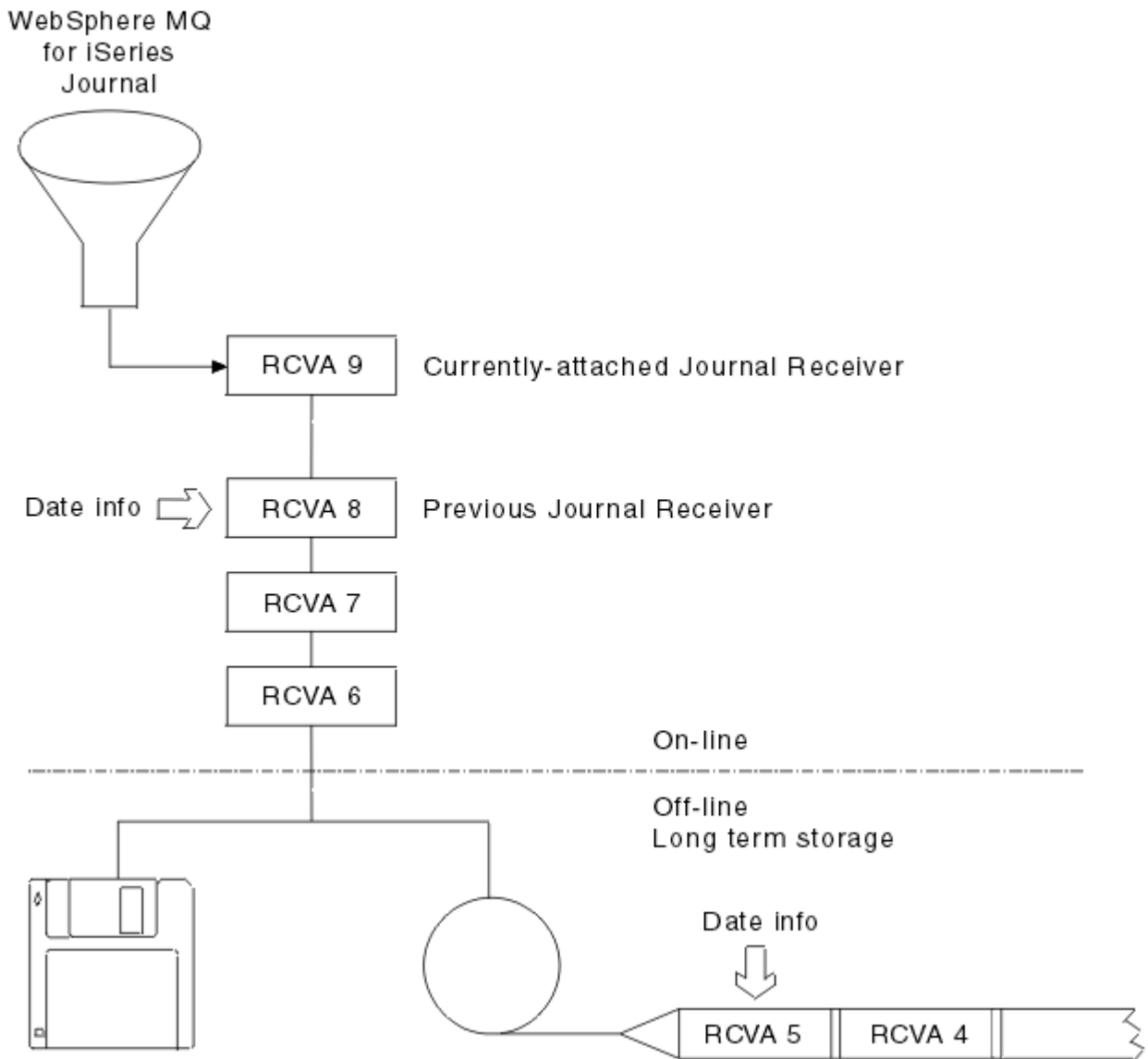


图 35: 在 IBM i 上进行日志记录

要确定何时可以从队列管理器库中除去已备份的日志接收器，以及何时可以废弃备份本身，请务必了解日志 IBM MQ 中可能需要返回的距离。

IBM MQ 向队列管理器消息队列 (队列管理器库中的 QMQMMSG) 发出两条消息以帮助确定此时间。当它启动时，当它更改本地日志接收器时，并且您使用 RCDMQIMG 来强制检查点时，将发出这些消息。这两条消息是：

AMQ7460

启动恢复点。此消息定义启动条目的日期和时间，在启动恢复通过时，IBM MQ 将从该启动条目重放日志。如果包含此记录的日志接收器在 IBM MQ 库中可用，那么此消息还包含包含此记录的日志接收器的名称。

AMQ7462

最早的介质恢复条目。此消息定义用于从其介质映像重新创建对象的最旧条目的日期和时间。

标识的日志接收器是所需的最旧的日志接收器。不再需要具有较早创建日期的任何其他 IBM MQ 日志接收器。如果仅显示星表，那么您需要从指示的日期恢复备份，以确定哪个是最早的日志接收器。

记录这些消息时，IBM MQ 还会将用户空间对象写入仅包含一个条目的队列管理器库：需要保留在系统上的最旧日志接收器的名称。此用户空间称为 AMQJRNINF，数据以如下格式写入：

```
JJJJJJJJJLLLLLLLLLLLLYYYYMMDDHHMMSSmmm
```

其中：

JJJJJJJJJ

是 IBM MQ 仍需要的最旧的接收方名称。

LLLLLLLLLLL

是日志接收器库名。

YYYY

是 IBM MQ 需要的最早日志项的年份。

MM

是 IBM MQ 需要的最早日志项的月份。

DD

是 IBM MQ 需要的最早日志项的日期。

HH

是 IBM MQ 需要的最早日志项的小时。

SS

IBM MQ 需要的最早日志项的秒数。

mmm

是 IBM MQ 需要的最早日志项的毫秒数。

从系统中删除最旧的日志接收器时，此用户空间包含日志接收器名称的星号 (*)。

注：定期执行 RCDQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) 可以为 IBM MQ 节省启动时间，并减少需要保存和恢复以进行恢复的本地日志接收器数。

IBM MQ for IBM i 不会引用日志接收器，除非它正在执行恢复传递以启动或重新创建对象。如果发现它需要的日志不存在，那么它会向队列管理器消息队列 (QMOMMSG) 发出消息 AMQ7432，报告完成恢复传递所需的日志项的时间和日期。

如果发生此情况，请从备份复原在此日期之后拆离的所有日志接收器，以允许恢复传递成功。

使包含启动项和任何后续日志接收器的日志接收器在队列管理器库中可用。

使包含最旧的 Media Recovery Entry 和任何后续日志接收器的日志接收器始终可用，并在队列管理器库中提供或备份。

强制检查点时：

- 如果未高级 AMQ7460 中指定的日志接收器，那么这指示存在需要落实或回滚的不完整工作单元。
- 如果 AMQ7462 中指定的日志接收器不是高级的，那么这指示存在一个或多个受损对象。

IBM i 在 IBM i 上复原完整队列管理器 (数据和日志)

使用此信息可从备份或远程机器复原一个或多个队列管理器。

如果要从备份中恢复一个或多个 IBM MQ 队列管理器，请执行以下步骤。

1. 停顿 IBM MQ 队列管理器。
2. 找到最新的备份集，由最新的完全备份和随后备份的日志接收器组成。
3. 通过发出以下命令，从完全备份执行 RSTLIB 操作，以将 IBM MQ 数据库恢复到完全备份时的状态：

```
RSTLIB LIB(QMQLIB1) .....  
RSTLIB LIB(QMQLIB2) .....
```

如果日志接收器部分保存在一个日志备份中，并且完全保存在后续备份中，那么仅复原完全保存的日志接收器。按时间顺序逐个复原日志。

4. 使用以下命令执行 RST 操作以将 IBM MQ IFS 目录复原到 IFS 文件系统:

```
RST DEV(...) OBJ('/QIBM/UserData/mqm/qmgrs/testqm') ...
```

5. 启动消息队列管理器。这将重放自完全备份以来写入的所有日志记录, 并将所有 IBM MQ 对象复原到日志备份时的一致状态。

如果要在另一台机器上复原完整的队列管理器, 请使用以下过程从队列管理器库复原所有内容。(我们使用 TEST 作为样本队列管理器名称。)

1. CRTMQM TEST

2. DLTLIB LIB(QMTEST)

3. RSTLIB SAVLIB(QMTEST) DEV(*SAVF) SAVF(QMGRLIBSAV)

4. 删除以下 IFS 文件:

```
/QIBM/UserData/mqm/qmgrs/TEST/QMQMCHKPT
/QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMQMOBJCAT
/QIBM/UserData/mqm/qmgrs/TEST/qmanager/QMANAGER
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.AUTH.DATA.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CHANNEL.INITQ/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.COMMAND.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.REPOSITORY.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.CLUSTER.TRANSMIT.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.PENDING.DATA.QUEUE/q
/QIBM/UserData/mqm/qmgrs/TEST/queues/SYSTEM.ADMIN.COMMAND.QUEUE/q
```

5. STRMQM TEST

6. RCRMQMOBJ OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(TEST)

IBM i 在 IBM i 上复原特定队列管理器的日志接收器

使用此信息来了解恢复日志接收器的不同方法。

最常见的操作是将已备份的日志接收器复原到队列管理器库中, 如果后续恢复功能再次需要已除去的接收器。

这是一个简单的任务, 需要使用标准 IBM i RSTOBJ 命令来恢复日志接收器:

```
RSTOBJ OBJ(QMQMDATA/AMQA000005) OBJTYPE(*JRNRCV) .....
```

可能需要恢复一系列日志接收器, 而不是单个接收器。例如, AMQA000007 是 IBM MQ 库中最早的接收方, 需要同时复原 AMQA000005 和 AMQA000006。

在这种情况下, 以相反的时间顺序分别恢复接收器。这并非总是必要的, 而是好的做法。在严重情况下, 您可能需要使用 IBM i 命令 WRKJRNA 将复原的日志接收器与日志相关联。

恢复日志时, 系统会自动创建一个在日志接收器序列中具有新名称的连接日志接收器。但是, 生成的新名称可能与需要复原的日志接收器相同。需要手动干预以克服此问题; 在恢复日志接收器之前, 按顺序创建新的名称日志接收器和新的日志。

例如, 考虑保存的日志 AMQAJRN 和以下日志接收器的问题:

- AMQA000000
- AMQA100000
- AMQA200000
- AMQA300000
- AMQA400000
- AMQA500000

- AMQA600000
- AMQA700000
- AMQA800000
- AMQA900000

将日志 AMQAJRN 复原到队列管理器库时，系统会自动创建日志接收器 AMQA000000。此自动生成的接收器与您要复原的其中一个现有日志接收器 (AMQA000000) 冲突，您无法复原这些接收器。

解法是：

1. 手动创建下一个日志接收器 (请参阅 [第 238 页的『IBM i 上的队列管理器日志』](#))：

```
CRTJRNRCV JRNRCV(QMQRLIB/AMQA9000001) THRESHOLD(XXXXX)
```

2. 使用日志接收器手动创建日志：

```
CRTJRN JRN(QMGRLIB/AMQAJRN) MNGRCV(*SYSTEM) +
JRNRCV(QMGRLIB/AMQA9000001) MSGQ(QMGRLIB/AMQAJRNMSG)
```

3. 将本地日志接收器 AMQA000000 复原到 AMQA900000。

IBM i IBM i 上的多实例队列管理器

多实例队列管理器通过在活动服务器发生故障时自动切换到备用服务器来提高可用性。活动服务器和备用服务器是相同队列管理器的多个实例；它们共享相同的队列管理器数据。如果活动实例失败，您需要将其日志传输到接管的备用数据库，以便队列管理器可以重建其队列。

配置正在其上运行多实例队列管理器的 IBM i 系统，以便在活动队列管理器实例发生故障时，它所使用的日志可供接管的备用实例使用。您可以设计自己的配置和管理任务，以使活动实例中的日志可供接管的实例使用。如果您不想丢失消息，那么您的设计必须确保备用日志与故障点的活动日志一致。您可以根据通过后续主题中的示例描述的两个配置之一来调整设计，以保持一致性。

1. 将日志从正在运行活动队列管理器实例的系统镜像到正在运行备用实例的系统。
2. 将日志放在可从运行活动实例的系统传输到备用实例的独立辅助存储池 (IASP) 中。

第一个解决方案不需要额外的硬件或软件，因为它使用基本 ASP。第二种解决方案需要可切换的 IASP，这些 IASP 需要 IBM i 集群支持，可作为单独定价的 IBM i 许可证产品 5761-SS1 选项 41 提供。

IBM i IBM i 上的可靠性和可用性

多实例队列管理器旨在提高应用程序的可用性。技术和物理约束意味着您需要不同的解决方案来满足灾难恢复，备份队列管理器和持续操作的需求。

在配置可靠性和可用性时，您会取舍大量因素，从而产生四个不同的设计点：

灾难恢复

针对破坏所有本地资产的重大灾难后的恢复进行了优化。

IBM i 上的灾难恢复通常基于 IASP 的地理镜像。

备份

针对本地化故障 (通常是人为错误或某些不可预见的技术问题) 后的恢复进行了优化。

IBM MQ 提供备份队列管理器以定期备份队列管理器。您还可以使用队列管理器日志的异步复制来提高备份的货币。

可用性

针对恢复操作进行了优化，在可预见的技术故障 (例如服务器或磁盘故障) 后快速提供几乎不间断服务的外观。

恢复通常以分钟为单位进行测量，检测有时需要比恢复过程更长的时间。多实例队列管理器可帮助您配置可用性。

连续操作

针对提供不间断服务进行了优化。

连续操作解决方案必须解决检测问题，几乎总是涉及通过多个系统提交相同的工作，并要么使用第一个结果，要么如果正确性是主要考虑因素，比较至少两个结果。

多实例队列管理器可帮助您配置可用性。队列管理器的一个实例一次处于活动状态。切换到备用实例需要 10 多秒到 15 分钟或更长时间，具体取决于系统的配置，装入和调整方式。

如果与可重新连接的 IBM MQ MQI clients 配合使用，那么多实例队列管理器可以提供近乎不间断的服务的外观，这些服务能够在应用程序不必知道队列管理器中断的情况下继续处理；请参阅主题 [自动化客户机重新连接](#)。

IBM i 上高可用性解决方案的组件

通过为配置为可重新启动队列管理器服务的应用程序的队列管理器数据，日志复制或队列管理器日志的强大 IASP 存储器提供强大的网络存储器，并使用可重新连接的客户机，构造使用多实例队列管理器的高可用性解决方案。

多实例队列管理器通过在另一个服务器上恢复另一个队列管理器实例的启动，对队列管理器故障的检测作出反应。要完成其启动，实例需要访问联网存储器中的共享队列管理器数据及其本地队列管理器日志的副本。

要创建高可用性解决方案，您需要管理队列管理器数据的可用性，本地队列管理器日志的货币，以及构建可重新连接的客户机应用程序，或者将应用程序部署为队列管理器服务以在队列管理器恢复时自动重新启动。IBM MQ classes for Java 不支持客户机自动重新连接。

队列管理器数据

将队列管理器数据放置到共享，高可用性和可靠的网络存储器上，可能使用 RAID 级别 1 磁盘或更高版本。文件系统需要满足多实例队列管理器的共享文件系统的需求；有关共享文件系统的需求的更多信息，请参阅 [共享文件系统的需求](#)。网络文件系统版本 4 (NFS4) 是满足这些需求的协议。

队列管理器日志

您还需要配置队列管理器实例所使用的 IBM i 日志，以便备用实例能够将其队列管理器数据复原到一致状态。对于不间断服务，这意味着您必须在活动实例失败时将日志复原到其状态。与备份或灾难恢复解决方案不同，将日志复原到较早的检查点是不够的。

无法在联网存储器上的多个 IBM i 系统之间物理共享日志。要在发生故障时将队列管理器日志复原到一致状态，您需要将在发生故障时位于活动队列管理器实例本地的物理日志传输到已激活的新实例，或者将日志的维护镜像传输到正在运行的备用实例。镜像日志是与属于失败实例的本地日志保持完全同步的远程日志副本。

三种配置是设计如何管理多实例队列管理器的日志的起点，

1. 使用从活动实例 ASP 到备用实例 ASP 的同步日志复制 (日志镜像)。
2. 正在将配置为保存队列管理器日志的 IASP 从活动实例传输到作为活动实例接管的备用实例。
3. 使用同步的辅助 IASP 镜像。

请参阅 IBM MQ IBM i CRTMQM 命令中的 [ASP](#) 选项，以获取有关将队列管理器数据放到 iASP 上的更多信息。

另请参阅 IBM Documentation 中的 [高可用性](#)。

应用

要构建客户机以在备用队列管理器恢复时自动重新连接到队列管理器，请使用 MQCONNX 将应用程序连接到队列管理器，并在 **MQCNO** 选项 字段中指定 MQCNO_RECONNECT_Q_MGR。请参阅 [高可用性样本程序](#)，以获取使用可重新连接的客户机的三个样本程序，以及 [应用程序恢复](#)，以获取有关设计客户机应用程序以进行恢复的信息。

在 IBM i 服务器上创建网络共享以存储队列管理器数据。设置来自两个服务器 (将托管队列管理器实例) 的连接以访问网络共享。

开始之前

- 此任务需要三个 IBM i 服务器。网络共享是在其中一个服务器 GAMMA 上定义的。另外两个服务器 (ALPHA 和 BETA) 将连接到 GAMMA。
- 在所有三个服务器上安装 IBM MQ。
- 安装 System i Navigator; 请参阅 [System i Navigator](#)。

关于此任务

- 在 GAMMA 上创建队列管理器目录, 并为用户概要文件 QMQM 和 QMQMADM 设置正确的所有权和许可权。通过在 GAMMA 上安装 IBM MQ, 可轻松创建目录和许可权。
- 使用 System i Navigator 来创建与 GAMMA 上的队列管理器数据目录的共享。
- 在 ALPHA 和 BETA 上创建指向共享的目录。

过程

1. 在 GAMMA 上, 创建以 QMQM 用户概要文件作为所有者并以 QMQMADM 作为主组来托管队列管理器数据的目录。

提示:

创建具有正确许可权的目录的快速可靠方法是在 GAMMA 上安装 IBM MQ。

稍后, 如果您不希望在 GAMMA 上运行 IBM MQ, 请卸载 IBM MQ。卸载后, 目录 /QIBM/UserData/mqm/qmgrs 将保留在 GAMMA 上, 并具有所有者 QMQM 用户概要文件和 QMQMADM 主组。

该任务使用 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 目录进行共享。

2. 启动 System i Navigator **添加连接** 向导并连接到 GAMMA 系统。
 - a) 双击 Windows 桌面上的 **System i Navigator** 图标。
 - b) 单击 **是** 以创建连接。
 - c) 遵循 "添加连接" 向导中的指示信息, 并创建从 IBM i 系统到 GAMMA 的连接。

与 GAMMA 的连接将添加到 **我的连接**。
3. 在 GAMMA 上添加新的文件共享。
 - a) 在 "System i Navigator" 窗口中, 单击 My Connections/GAMMA/File Systems 中的 File Shares 文件夹。
 - b) 在 "我的任务" 窗口中, 单击 **管理 IBM i NetServer 共享**。

新窗口 **IBM i NetServer -GAMMA** 将在桌面上打开并显示共享对象。
 - c) 右键单击 Shared Objects 文件夹 > **文件** > **新建** > **文件**。

将打开新窗口 **IBM i NetServer File Share-GAMMA**。
 - d) 为共享提供名称, 例如 WMQ。
 - e) 将访问控制设置为 Read/Write。
 - f) 通过浏览到先前创建的 /QIBM/UserData/mqm/qmgrs 目录来选择 **路径名**, 然后单击 **确定**。

IBM i NetServer File Share-GAMMA 窗口将关闭, 并且 WMQ 将列示在共享对象窗口中。

4. 在共享对象窗口中右键单击 **WMQ**。单击 **文件** > **许可权**。

将打开对象 /QIBM/UserData/mqm/qmgrs 的窗口 **Qmgrs 许可权-GAMMA**。

- a) 检查 QMQM 的以下许可权 (如果尚未设置):

Read

Write
Execute
Management
Existence
Alter
Reference

b) 检查 QMQMADM 的以下许可权 (如果尚未设置):

Read
Write
Execute
Reference

c) 添加要授予 /QIBM/UserData/mqm/qmgrs 许可权的其他用户概要文件。

例如, 您可以将缺省用户概要文件 (公共) Read 和 Execute 许可权授予 /QIBM/UserData/mqm/qmgrs。

5. 检查在 GAMMA 上授予对 /QIBM/UserData/mqm/qmgrs 的访问权的所有用户概要文件是否与在访问 GAMMA 的服务器上的用户概要文件具有相同的密码。

特别是, 请确保要访问共享的其他服务器上的 QMQM 用户概要文件与 GAMMA 上的 QMQM 用户概要文件具有相同的密码。

提示: 单击 System i Navigator 中的 My Connections/GAMMA/Users and Groups 文件夹以设置密码。或者, 使用 **CHFUSRPRF** 和 **CHGPWD** 命令。

结果

请检查您是否可以使用共享从其他服务器访问 GAMMA。如果要执行其他任务, 请检查是否可以使用路径 /QNTC/GAMMA/WMQ 从 ALPHA 和 BETA 访问 GAMMA。如果 /QNTC/GAMMA 目录在 ALPHA 或 BETA 上不存在, 那么必须创建该目录。根据 NetServer 域, 您可能必须在创建目录之前对 ALPHA 或 BETA 进行 IPL。

```
CRTDIR DIR('/QNTC/GAMMA')
```

When you have checked that you have access to /QNTC/GAMMA/WMQ from ALPHA or BETA, issuing the command, CRTMQM MQMNAME('QM1') MQMDIRP('/QNTC/GAMMA/WMQ') creates /QIBM/UserData/mqm/qmgrs/QM1 on GAMMA.

下一步做什么

通过执行任务 第 258 页的『在 IBM i 上使用日志镜像和 NetServer 创建多实例队列管理器』或 第 261 页的『在 IBM i 上使用 NetServer 和日志镜像将单实例队列管理器转换为多实例队列管理器』中的步骤来创建多实例队列管理器。

IBM i 上的故障转移性能

检测队列管理器实例失败的时间, 然后在备用数据库上恢复处理的时间可能在数十秒到十五分钟或更长时间之间, 具体取决于配置。在设计和测试高可用性解决方案时, 性能需要成为主要考虑因素。

在决定是将多实例队列管理器配置为使用日志复制还是使用 IASP 时, 需要权衡一些优点和缺点。镜像要求队列管理器同步写入远程日志。从硬件角度来看, 这不需要影响性能, 但从软件角度来看, 写入远程日志所涉及的路径长度大于仅写入本地日志所涉及的路径长度, 这可能在一定程度上会降低正在运行的队列管理器的性能。但是, 当备用队列管理器接管时, 与 IBM i 检测 IASP 并将其传输到运行队列管理器备用实例的服务器所花费的时间相比, 将其本地日志从活动实例维护的远程日志同步失败之前的延迟通常较小。IASP 传输时间可能高达 10 到 15 分钟, 而不是以秒为单位完成。IASP 传输时间取决于将 IASP 传输到备用系统时需要联机的对象数以及需要合并的访问路径或索引的大小。

当备用队列管理器接管时, 与 IBM i 检测独立 ASP 并将其传输到运行队列管理器的备用实例的服务器所花费的时间相比, 将其本地日志与活动实例维护的远程日志同步失败之前的延迟通常较小。独立 ASP 传输时间

可能高达 10 到 15 分钟，而不是以秒为单位完成。独立 ASP 传输时间取决于将独立 ASP 传输到备用系统时需要联机的对象数以及需要合并的访问路径或索引的大小。

但是，转移日志并不是影响备用实例完全恢复所需的时间的唯一因素。您还需要考虑网络文件系统释放对队列管理器数据的锁定所花费的时间，这些数据向备用实例发出信号以尝试继续其启动，还需要考虑从日志中恢复队列以便实例能够再次开始处理消息所花费的时间。这些其他延迟源都增加了启动备用实例所需的时间。切换的总时间由以下组件组成：

故障检测时间

NFS 释放对队列管理器数据的锁定所需的时间，以及备用实例继续其启动过程所需的时间。

传输时间

对于 HA 集群，这是 IBM i 将 IASP 从主管活动实例的系统传输到备用实例所花费的时间，对于日志复制，这是使用远程副本中的数据更新备用数据库上的本地日志所花费的时间。

重新启动时间

新活动队列管理器实例从其复原的日志中的最新检查点重建其队列并恢复处理消息所需的时间。

注：

如果已接管的备用实例配置为同步复制到先前活动的实例，那么可能会延迟启动。如果远程日志位于托管先前活动实例的服务器上，并且服务器发生故障，那么新的已激活实例可能无法复制到其远程日志。

等待同步响应的缺省时间为 1 分钟。您可以在复制超时之前配置最大延迟。或者，您可以配置备用实例以开始对失败的活动实例使用异步复制。稍后，当发生故障的实例再次在备用数据库上运行时，将切换到同步复制。同一注意事项适用于使用同步独立 ASP 镜像。

您可以对这些组件进行单独的基线测量，以帮助评估故障转移的总体时间，并在决策中考虑要使用的配置方法。在做出最佳配置决策时，您还需要考虑同一服务器上的其他应用程序将如何进行故障转移，以及是否存在已使用 IASP 的备份或灾难恢复进程。

可以通过调整集群配置来缩短 IASP 传输时间：

1. 集群中跨系统的用户概要文件应该具有相同的 GID 和 UID，以消除联机过程更改 UID 和 GID 的需求。
2. 最小化系统和基本用户磁盘池中的数据库对象数，因为需要合并这些对象以创建磁盘池组的交叉引用表。
3. 可以在 IBM 红皮书 *实现 PowerHA for IBM i, SG24-7405* 中找到更多性能提示。

使用基本 ASP，日志镜像和小型配置的配置应该以数十秒的顺序切换。

IBM i 将 IBM i 集群功能与 IBM MQ 集群相结合的概述

与仅使用 IBM MQ 集群相比，在 IBM i 上运行 IBM MQ 并利用 IBM i 集群功能可提供更全面的高可用性解决方案。

要具有此功能，您需要设置：

1. IBM i 机器上的集群；请参阅第 250 页的『[IBM i 集群](#)』
2. 将队列管理器移动到其中的独立辅助存储池 (IASP)；请参阅第 251 页的『[独立辅助存储池 \(IASP\)](#)』
3. 集群资源组 (CRG)；请参阅第 251 页的『[设备集群资源组](#)』，在其中定义：
 - 恢复域
 - IASP
 - 出口程序；请参阅第 251 页的『[设备 CRG 出口程序](#)』

IBM i 集群

IBM i 集群是逻辑上链接在一起的实例 (即 IBM i 计算机或分区) 的集合。

此分组的目的是允许备份每个实例，从而消除单点故障并提高应用程序和数据弹性。通过创建集群，可以配置各种集群资源组 (CRG) 类型来管理集群中的应用程序，数据和设备。

请参阅 [创建集群](#) 和 [创建集群 \(CRTCLU\)](#) 命令以获取更多信息。

独立辅助存储池 (IASP)

IASP 是充当单级存储器扩展的用户 ASP 类型。它是一个存储器，由于它独立于系统存储器，可以轻松地进行操作，而不必对系统进行 IPL。

IASP 可以轻松切换到另一个操作系统实例，或者复制到另一个操作系统实例上的目标 IASP。可以使用两种方法在实例之间切换 IASP:

- 第一种方法要求使用高速链路 (HSL) 环路连接集群中的所有计算机以及包含 IASP 的可切换磁盘塔。
- 第二种方法要求操作系统实例是可以在分区之间切换输入/输出处理器 (IOP) 的同一 IBM i 计算机上的分区。无需特殊硬件即可复制 IASP。通过网络使用 TCP/IP 执行复制。

有关更多信息，请参阅 [配置设备 ASP \(CFGDEVASP\) 命令](#)。

设备集群资源组

有多种类型的集群资源组 (CRG)。有关不同类型的可用 CRG 的更多信息，请参阅 [集群资源组](#)。

本主题集中讨论设备 CRG。设备 CRG:

- 描述和管理设备资源，例如独立辅助存储池 (IASP)。
- 定义集群节点的恢复域
- 分配设备，以及
- 分配将处理集群事件的出口程序。

恢复域表示哪个集群节点将被视为主节点。其余节点被视为备份。备份节点也在恢复域中排序，指定哪个节点是第一个备份，第二个备份，依此类推，具体取决于恢复域中有多少个节点。

如果主节点发生故障，那么将在恢复域中的所有节点上运行出口程序。然后，在第一个备份上运行的出口程序可以进行必要的初始化，以使此节点成为新的主节点。

有关更多信息，请参阅 [创建设备 CRG 和 创建集群资源组 \(CRTCRG\) 命令](#)。

设备 CRG 出口程序

当在恢复域定义的某个节点中发生事件时，操作系统集群资源服务会调用设备 CRG 出口程序; 例如，故障转移或切换事件。

当集群的主节点发生故障，并且 CRG 使用其管理的所有资源进行切换时，将发生故障转移事件，当手动将特定 CRG 从主节点切换到备份节点时，将发生切换事件。

无论哪种方式，出口程序都负责初始化和启动在先前主节点上运行的所有程序，这会将第一个备份节点转换为新的主节点。

例如，对于 IBM MQ，出口程序应负责启动 IBM MQ 子系统 (QMQM) 和队列管理器。队列管理器应配置为自动启动侦听器和服务，例如触发器监视器。

可切换 IASP 配置

可以设置 IBM MQ 以利用 IBM i 的集群功能。要执行此操作:

1. 在数据中心系统之间创建 IBM i 集群
2. 将队列管理器移至 IASP。

第 252 页的『[将队列管理器移动到独立辅助存储池或从独立辅助存储池中除去队列管理器](#)』包含一些样本代码以帮助您执行此操作。

3. 您需要创建定义恢复域，IASP 和出口程序的 CRG。

第 252 页的『[配置设备集群资源组](#)』包含一些样本代码以帮助您执行此操作。

相关概念

第 269 页的『[独立 ASP 和高可用性](#)』

独立 ASP 支持在服务器之间移动应用程序和数据。独立 ASP 的灵活性意味着它们是某些 IBM i 高可用性解决方案的基础。在考虑是将 ASP 还是独立 ASP 用于队列管理器日志时，应考虑基于独立 ASP 的其他高可用性配置。

IBM i 配置设备集群资源组

用于设置设备集群资源组 (CRG) 的示例程序。

关于此任务

在以下示例中，请注意：

- [PRIMARY SITE NAME] 和 [BACKUP SITE NAME] 可以是任意两个长度不超过 8 个字符的不同字符串。
- [PRIMARY IP] 和 [BACKUP IP] 是要用于镜像的 IP。

过程

1. 标识集群的名称。
2. 标识 CRG 出口程序名和库。
3. 确定要由此 CRG 定义的主节点和备份节点的名称。
4. 标识要由此 CRG 管理的 IASP，并确保已在主节点下创建该 IASP。
5. 使用以下命令在备份节点中创建设备描述：

```
CRTDEVASP DEVD([IASP NAME]) RSRcname([IASP NAME])
```

6. 使用以下命令将接管 IP 地址添加到所有节点：

```
ADDTCPIFC INTNETADR(' [TAKEOVER IP]') LIND([LINE DESC])  
SUBNETMASK(' [SUBNET MASK]') AUTOSTART(*NO)
```

7. 使用以下命令仅在主节点中启动接管 IP 地址：

```
STRTCPIFC INTNETADR(' [TAKEOVER IP]')
```

8. 可选：如果 IASP 可切换，请调用以下命令：

```
CRTCRG CLUSTER([CLUSTER NAME]) CRG( [CRG NAME]) CRGTYPE(*DEV) EXITPGM([EXIT LIB]/[EXIT  
NAME])  
USRPRF([EXIT PROFILE]) RCYDMN(([PRIMARY NODE] *PRIMARY) ([BACKUP NAME] *BACKUP))  
EXITPGMFMT(EXTP0200) CFGOBJ(([IAPS NAME] *DEVD *ONLINE '[TAKEOVER IP]')
```

9. 可选：如果要对 IASP 进行镜像，请调用以下命令：

```
CRTCRG CLUSTER([CLUSTER NAME]) CRG([CRG NAME]) CRGTYPE(*DEV) EXITPGM([EXIT LIB]/[EXIT NAME])  
USRPRF([EXIT PROFILE]) RCYDMN(([PRIMARY NODE] *PRIMARY *LAST [PRIMARY SITE NAME] (' [PRIMARY  
IP]'))  
[BACKUP NAME] *BACKUP *LAST [BACKUP SITE NAME] (' [BACKUP IP]')) EXITPGMFMT(EXTP0200)  
CFGOBJ(([IAPS NAME] *DEVD *ONLINE '[TAKEOVER IP]'))
```

IBM i 将队列管理器移动到独立辅助存储池或从独立辅助存储池中除去队列管理器

用于将队列管理器移动到独立辅助存储池 (IASP) 的示例程序以及用于从 IASP 中除去队列管理器的命令。

关于此任务

在以下示例中，请注意：

- [MANAGER NAME] 是队列管理器的名称。
- [IASP NAME] 是 IASP 的名称。
- [MANAGER LIBRARY] 是队列管理器库的名称。
- [MANAGER DIRECTORY] 是队列管理器目录的名称。

过程

1. 确定主节点和备份节点。
2. 在主节点上执行以下过程:
 - a) 确保队列管理器已结束。
 - b) 通过使用以下命令确保您的 IASP 为 vary on

```
VRYCFG CFGOBJ([IASP NAME]) CFGTYPE(*DEV) STATUS(*ON)
```

- c) 在 IASP 下创建队列管理器目录。
在根目录下将有一个名为 IASP 的目录，它是:

```
QSH CMD('mkdir -p /[IASP_NAME]/QIBM/UserData/mqm/qmgrs/')
```

- d) 使用以下命令将管理器的 IFS 对象移至刚刚在 IASP 下创建的队列管理器目录:

```
QSH CMD('mv /QIBM/UserData/mqm/qmgrs/[MANAGER NAME]  
/[IASP NAME]/QIBM/UserData/mqm/qmgrs/')
```

- e) 使用以下命令创建名为 MGRLIB 的临时保存文件:

```
CRTSAVF QGPL/MGRLIB
```

- f) 使用以下命令将队列管理器库保存到 MGRLIB 保存文件:

```
SAVLIB LIB([MANGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
```

- g) 使用以下命令删除队列管理器库，并忽略所有查询消息:

```
DLTLIB [MANAGER LIBRARY]
```

- h) 使用以下命令将队列管理器库复原到 IASP:

```
RSTLIB SAVLIB([MANAGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)  
RSTASPDEV([IASP NAME])
```

- i) 使用以下命令删除临时保存文件:

```
DLTF FILE(QGPL/MGRLIB)
```

- j) 使用以下命令创建指向 IASP 下的队列管理器 IFS 对象的符号链接:

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')  
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

- k) 使用以下命令连接到 IASP:

```
SETASPGRP [IASP NAME]
```

- l) 使用以下命令启动队列管理器:

```
STRMQM [MANAGER NAME]
```

3. 在备份节点上执行以下过程:

- a) 使用以下命令创建临时队列管理器目录:

```
QSH CMD('mkdir -p /[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

- b) 使用以下命令创建指向队列管理器临时目录的符号链接:

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

c) 使用以下命令删除临时目录:

```
QSH CMD('rm -r /[IASP NAME]')
```

d) 在文件 /QIBM/UserData/mqm/mqs.ini 末尾添加以下内容:

```
QueueManager:
Name=[MANAGER NAME]
Prefix=/QIBM/UserData/mqm
Library=[MANAGER LIBRARY]
Directory=[MANAGER DIRECTORY]
```

4. 要从 IASP 中除去队列管理器, 请发出以下命令:

- a) VRYCFG CFGOBJ ([IASP NAME]) CFGTYPE (*DEV) 状态 (*ON)
- b) SETASPGRP [IASP 名称]
- c) ENDMQM [管理器名称]
- d) DLTMQM [管理器名称]

IBM i 上 ASP 的镜像日志配置

使用镜像日志之间的同步复制来配置稳健的多实例队列管理器。

镜像队列管理器配置使用在基本或独立辅助存储池 (ASP) 中创建的日志。

在 IBM i 上, 队列管理器数据将写入日志和文件系统。日志包含队列管理器数据的主副本。使用同步或异步日志复制在系统之间共享日志。需要混合使用本地和远程日志才能重新启动队列管理器实例。队列管理器重新启动会从服务器上的本地和远程日志以及共享网络文件系统上的队列管理器数据的混合中读取日志记录。文件系统中的数据会加快重新启动队列管理器的速度。检查点存储在文件系统中, 用于标记文件系统与日志之间的同步点。典型队列管理器重新启动不需要在检查点之前存储的日志记录。但是, 文件系统中的数据可能不是最新的, 并且使用检查点之后的日志记录来完成队列管理器重新启动。附加到实例的日志中的数据保持最新, 以便可以成功完成重新启动。

但是, 如果正在异步复制备用服务器上的远程日志, 并且在同步之前发生了故障, 那么即使日志记录也可能不是最新的。如果您决定使用未同步的远程日志重新启动队列管理器, 那么备用队列管理器实例可能重新处理在活动实例失败之前删除的消息, 或者不处理在活动实例失败之前接收到的消息。

另一种罕见的可能性是, 文件系统包含最新的检查点记录, 而备用数据库上未同步的远程日志不包含。在这种情况下, 队列管理器不会自动重新启动。您可以选择等到远程日志同步, 或者从文件系统冷启动备用队列管理器。尽管在这种情况下, 文件系统包含比远程日志更新的队列管理器数据检查点, 但它可能不包含在活动实例失败之前处理的所有消息。在与日志不同步的冷重新启动后, 可能会重新处理某些消息, 但某些消息未处理。

对于多实例队列管理器, 文件系统还用于控制队列管理器的哪个实例处于活动状态以及哪个是备用实例。活动实例获取对队列管理器数据的锁定。备用数据库等待获取锁定, 并且在执行此操作时, 它将成为活动实例。如果锁定正常结束, 那么该锁定将由活动实例释放。如果文件系统检测到活动实例发生故障或无法访问文件系统, 那么该锁定将由文件系统释放。文件系统必须满足检测故障的要求; 请参阅 [共享文件系统的要求](#)。

IBM i 上多实例队列管理器的体系结构在服务器或队列管理器发生故障后提供自动重新启动。它还支持在存储队列管理器数据的文件系统发生故障后恢复队列管理器数据。

在第 255 页的图 36 中, 如果 ALPHA 失败, 那么可以使用镜像日志在 Beta 上手动重新启动 QM1。通过将多实例队列管理器功能添加到 QM1, 如果 ALPHA 上的活动实例失败, 那么 QM1 的备用实例将在 BETA 上自动恢复。如果是服务器 ALPHA 失败 (而不仅仅是 QM1 的活动实例), 那么 QM1 也可以自动恢复。一旦 BETA 成为活动队列管理器实例的主机, 就可以在 ALPHA 上启动备用实例。

第 255 页的图 36 显示了使用 NetServer 来存储队列管理器数据的两个队列管理器实例之间的日志镜像配置。您可以展开该模式以包含更多日记帐, 从而包含更多实例。遵循主题 [第 238 页的『IBM i 上的队列管](#)

理器日志』中说明的日志命名规则。目前，队列管理器的运行实例数限制为 2 个，1 个处于活动状态，1 个处于备用状态。

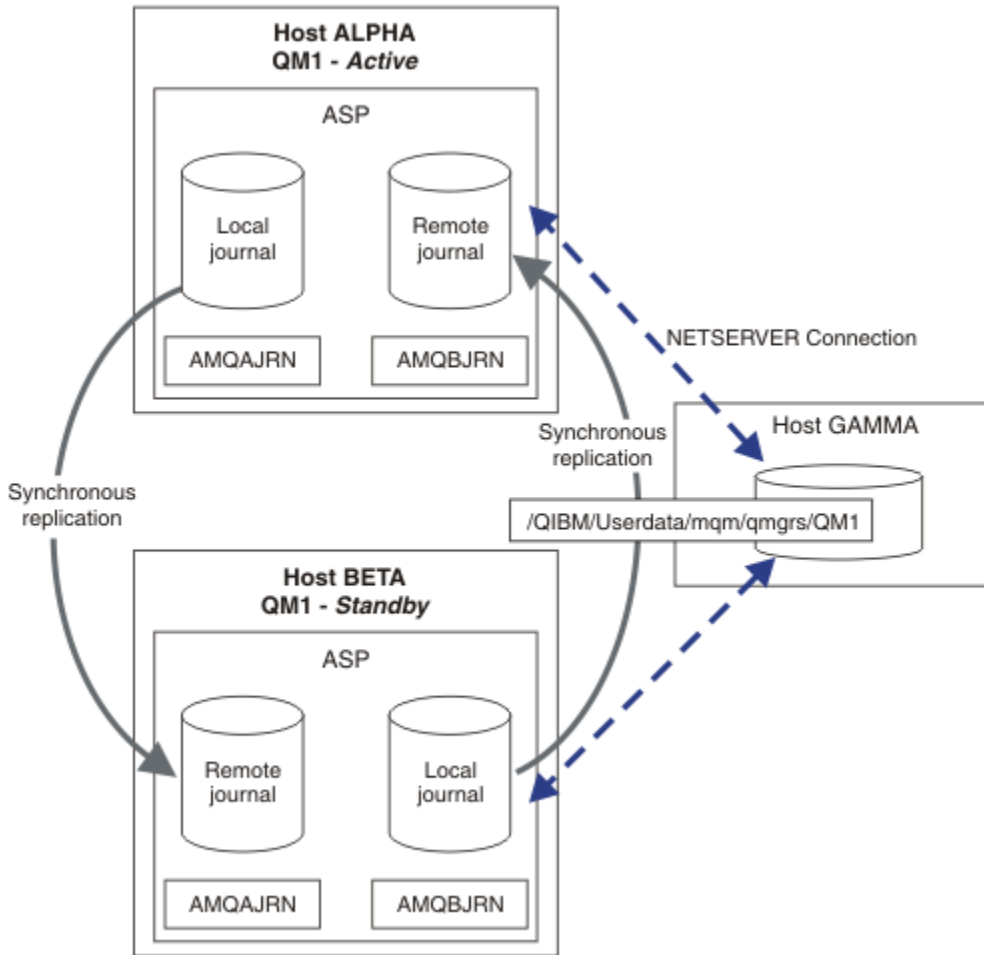


图 36: 镜像队列管理器日志

主机 ALPHA 上的 QM1 的本地日志称为 AMQAJRN (或更完整的 QMQM1/AMQAJRN)，而在 BETA 上的日志为 QMQM1/AMQBJRN。每个本地日志都会复制到队列管理器的所有其他实例上的远程日志。如果使用两个实例配置队列管理器，那么会将本地日志复制到一个远程日志。

*SYNC 或 *ASync 远程日志复制

使用同步 (*SYNC) 对 IBM i 日志进行镜像 或异步 (*ASync) 日志记录; 请参阅 [远程日志管理](#)。

第 255 页的图 36 中的复制方式为 *SYNC，而不是 *ASync。*ASync 更快，但如果远程日志状态为 *ASyncPEND 时发生故障，那么本地日志和远程日志不一致。远程日志必须与本地日志同步。如果选择 *SYNC，那么本地系统在从需要完成写的调用返回之前等待远程日志。本地和远程日志通常保持一致。仅当 *SYNC 操作花费的时间超过指定的时间时¹，并且远程日志记录已取消激活，请执行日志脱离同步。将错误记录到日志消息队列和 QSYSOPR。队列管理器检测到此消息，将错误写入队列管理器错误日志，并取消激活队列管理器日志的远程复制。活动队列管理器实例将在没有远程日志记录到此日志的情况下恢复。当远程服务器再次可用时，必须手动重新激活同步远程日志复制。然后再同步这些日志。

第 255 页的图 36 中说明的 *SYNC / *ASync 配置问题是 BETA 上的备用队列管理器实例如何进行控制。一旦 BETA 上的队列管理器实例写入其第一条持久消息，它就会尝试更新 ALPHA 上的远程日志。如果控制从 ALPHA 传递到 BETA 的原因是 ALPHA 失败，并且 ALPHA 仍处于关闭状态，那么远程日志记录到 ALPHA 将失败。BETA 等待 ALPHA 响应，然后取消激活远程日志记录，并仅使用本地日志记录来恢复处理消息。BETA 要等一段时间才能检测到 ALPHA 下降，造成一段时间的不活动。

¹ 指定的时间为 60 秒 (在 IBM i 5 上)，范围为 1-3600 秒 (在 IBM i 6.1 上)。

选择将远程日志记录设置为 *SYNC 或 *ASYN 是一种权宜之选。第 256 页的表 22 总结了在一对队列管理器之间使用 *SYNC 和 *ASYN 日志记录之间的权衡：

| Active | 备用 | *SYNC | *ASYN |
|--------|----|-----------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| *SYNC | | <ol style="list-style-type: none"> 1. 一致的转换和故障转移 2. 故障转移后，备用实例不会立即恢复。 3. 远程日志记录必须始终可用 4. 队列管理器性能取决于远程日志记录 | <ol style="list-style-type: none"> 1. 一致的转换和故障转移 2. 当备用服务器可用时，必须将远程日志记录切换到 *SYNC 3. 远程日志记录在重新启动后必须保持可用 4. 队列管理器性能取决于远程日志记录 |
| *ASYN | | <ol style="list-style-type: none"> 1. 不是明智的组合 | <ol style="list-style-type: none"> 1. 在故障转移或转换后，某些消息可能丢失或重复 2. 备用实例不需要一直可用，活动实例才能毫无延迟地继续。 3. 性能不依赖于远程日志记录 |

*SYNC / *SYNC

活动队列管理器实例使用 *SYNC 日志记录，当备用队列管理器实例启动时，它立即尝试使用 *SYNC 日志记录。

1. 远程日志在事务上与活动队列管理器的本地日志一致。如果队列管理器切换到备用实例，那么它可以立即恢复。备用实例通常会恢复，而不会丢失或重复任何消息。仅当自上次检查点以来远程日志记录失败，并且无法重新启动先前处于活动状态的队列管理器时，才会丢失或复制消息。
2. 如果队列管理器故障转移到备用实例，那么它可能无法立即启动。使用 *SYNC 日志记录来激活备用队列管理器实例。故障转移的原因可能阻止远程日志记录到托管备用实例的服务器。队列管理器等待直到检测到问题，然后再处理任何持久消息。将错误记录到日志消息队列和 QSYSOPR。队列管理器检测到此消息，将错误写入队列管理器错误日志，并取消激活队列管理器日志的远程复制。活动队列管理器实例将在没有远程日志记录到此日志的情况下恢复。当远程服务器再次可用时，必须手动重新激活同步远程日志复制。然后再同步这些日志。
3. 远程日志复制到的服务器必须始终可用于维护远程日志。通常会将远程日志复制到托管备用队列管理器的同一服务器。服务器可能变得不可用。将错误记录到日志消息队列和 QSYSOPR。队列管理器检测到此消息，将错误写入队列管理器错误日志，并取消激活队列管理器日志的远程复制。活动队列管理器实例将在没有远程日志记录到此日志的情况下恢复。当远程服务器再次可用时，必须手动重新激活同步远程日志复制。然后再同步这些日志。
4. 远程日志记录比本地日志记录慢，如果服务器相隔很大的距离，那么远程日志记录将慢得多。队列管理器必须等待远程日志记录，这将降低队列管理器性能。

一对服务器之间的 *SYNC / *SYNC 配置具有在故障转移后恢复备用实例的延迟的缺点。*SYNC / *ASYN 配置没有此问题。

*SYNC / *SYNC 保证转换或故障转移后不会丢失消息，只要远程日志可用。如果要降低故障转移或转换后消息丢失的风险，您有两个选择。如果远程日志变为不活动状态，请停止活动实例，或者在多个服务器上创建远程日志。

*SYNC / *ASYN

活动队列管理器实例使用 *SYNC 日志记录，当备用队列管理器实例启动时，它使用 *ASYN 日志记录。在主管新备用实例的服务器变为可用后不久，系统操作员必须将活动实例上的远程日志切换到 *SYNC。当操作员将远程日志记录从 *ASYN 切换到 *SYNC 时，如果远程日志的状态为 *ASYNPEND，那么活动实例将暂停。活动队列管理器实例将等待剩余日志项传输到远程日志。当远程日志与本地日志同步时，新的备用数据库与新的活动实例在事务上再次保持一致。从多实例队列管理器管理的角度来看，在 *SYNC / *ASYN 配置中，IBM i 系统操作员具有其他任务。除了重新启动失败的队列管理器实例外，操作员还必须将远程日志记录切换到 *SYNC。

1. 远程日志在事务上与活动队列管理器的本地日志一致。如果活动队列管理器实例已切换或故障转移到备用实例，那么备用实例可以立即恢复。备用实例通常会恢复，而不会丢失或重复任何消息。仅当自上次检查点以来远程日志记录失败，并且无法重新启动先前处于活动状态的队列管理器时，才会丢失或复制消息。
2. 在主管活动实例的系统再次变为可用后，系统操作员必须立即将远程日志从 *ASYNCR 切换到 *SYNCR。在将远程日志切换到 *SYNCR 之前，操作员可能等待远程日志追赶。或者，操作员可以立即将远程实例切换到 *SYNCR，并强制活动实例等待直到备用实例日志已捕获。当远程日志记录设置为 *SYNCR 时，备用实例通常在事务上与活动实例一致。仅当自上次检查点以来远程日志记录失败，并且无法重新启动先前处于活动状态的队列管理器时，才会丢失或复制消息。
3. 从转换或故障转移恢复配置后，托管远程日志的服务器必须始终可用。

当您希望备用队列管理器在故障转移后快速恢复时，请选择 *SYNCR / *ASYNCR。必须手动将新活动实例上的远程日志设置恢复为 *SYNCR。*SYNCR / *ASYNCR 配置与管理一对多实例队列管理器的正常模式相匹配。一个实例发生故障后，在重新启动备用实例之前会有一段时间，在此期间，活动实例无法进行故障转移。

***ASYNCR / *ASYNCR**

托管活动队列管理器和备用队列管理器的服务器都配置为使用 *ASYNCR 远程日志记录。

1. 发生转换或故障转移时，队列管理器将继续处理新服务器上的日志。发生转换或故障转移时，可能无法同步日志。因此，消息可能会丢失或重复。
2. 即使主管备用队列管理器的服务器不可用，活动实例也会运行。本地日志在可用时与备用服务器异步复制。
3. 远程日志记录不会影响本地队列管理器的性能。

如果性能是您的主要需求，请选择 *ASYNCR / *ASYNCR，并且您已准备好在故障转移或转换后松散或复制某些消息。

***ASYNCR / *SYNCR**

没有理由使用此选项组合。

从远程日志激活队列管理器

日志可以同步复制，也可以异步复制。远程日志可能未处于活动状态，或者它可能正在赶上本地日志。远程日志可能正在追赶，即使它是同步复制的，因为它可能是最近激活的。队列管理器对其在启动期间使用的远程日志的状态应用的规则如下所示。

1. 如果备用数据库必须从备用数据库上的远程日志重放并且日志状态为 *FAILED 或 *INACTPEND，那么备用数据库启动将失败。
2. 当开始激活备用数据库时，备用数据库上的远程日志状态必须为 *ACTIVE 或 *INACTIVE。如果状态为 *INACTIVE，那么激活可能失败 (如果尚未复制所有日志数据)。

如果网络文件系统上的队列管理器数据具有比远程日志中存在的更新的检查点记录，那么将发生此故障。只要远程日志在检查点之间的缺省 30 分钟最大时间间隔内激活良好，就不太可能发生故障。如果备用队列管理器确实从文件系统读取了较新的检查点记录，那么它不会启动。

您可以选择：等到可以复原活动服务器上的本地日志，或者冷启动备用队列管理器。如果选择冷启动，那么队列管理器启动时没有日志数据，并且依赖于文件系统中队列管理器数据的一致性和完整性。

注：如果冷启动队列管理器，那么在最后一个检查点之后会有丢失或复制消息的风险。消息事务已写入日志，但某些事务可能未写入文件系统中的队列管理器数据。冷启动队列管理器时，将启动新日志，并且未写入文件系统中的队列管理器数据的事务将丢失。

3. 备用队列管理器激活将等待备用数据库上的远程日志状态从 *ASYNCPEND 或 *SYNCPEND 更改为 *ASYNCR 或 *SYNCR。定期将消息写入执行控制器的作业记录。

注：在这种情况下，激活正在等待正在激活的备用队列管理器的本地远程日志。在没有远程日志的情况下继续之前，队列管理器还会等待一段时间。当它尝试以同步方式写入其远程日志 (或日志) 并且该日志不可用时，它将等待。

4. 如果日志状态更改为 *FAILED 或 *INACTPEND，那么激活将停止。

要在激活中使用的本地和远程日志的名称和状态将写入队列管理器错误日志。

创建要在两个 IBM i 服务器上运行的多实例队列管理器。队列管理器数据使用 NetServer 存储在第三个 IBM i 服务器上。使用远程日志记录在两个服务器之间对队列管理器日志进行镜像。 **ADDQMJRN** 命令用于简化远程日志的创建。

开始之前

1. 该任务需要三个 IBM i 服务器。在其中两个上安装 IBM MQ，即示例中的 ALPHA 和 BETA。IBM MQ 必须至少为 V 7.0.1.1。
2. 第三个服务器是 IBM i 服务器，由 NetServer 连接到 ALPHA 和 BETA。它用于共享队列管理器数据。它不必具有 IBM MQ 安装。作为临时步骤在服务器上安装 IBM MQ 以设置队列管理器目录和许可权非常有用。
3. 确保 QMQM 用户概要文件在所有三个服务器上都具有相同的密码。
4. 安装 IBM i NetServer; 请参阅 [i5/OS NetServer](#)。

关于此任务

执行以下步骤以创建 [第 260 页](#) 的图 37 中显示的配置。使用 IBM i NetServer 连接队列管理器数据。

- 创建从 ALPHA 和 BETA 到 GAMMA 上用于存储队列管理器数据的目录共享的连接。此任务还设置必需的许可权，用户概要文件和密码。
- 将关系数据库条目 (RDBE) 添加到要运行队列管理器实例的 IBM i 系统。RDBE 条目用于连接到用于远程日志记录的 IBM i 系统。
- 在 IBM i 服务器 ALPHA 上创建队列管理器 QM1。
- 在另一个 IBM i 服务器 BETA 上添加 QM1 的队列管理器控制信息。
- 在两个 IBM i 服务器上为两个队列管理器实例创建远程日志。每个队列管理器都写入本地日志。本地日志将复制到远程日志。命令 **ADDQMJRN** 简化了添加日志和连接的过程。
- 启动队列管理器，允许备用实例。

过程

1. 执行任务 [第 248 页](#) 的『在 IBM i 上使用 NetServer 为队列管理器数据创建网络共享』。

因此，ALPHA 和 BETA 具有指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 的共享 /QNTC/GAMMA/WMQ。用户概要文件 QMQM 和 QMQMADM 具有必需的许可权，并且 QMQM 在所有三个系统上都具有匹配的密码。

2. 将关系数据库条目 (RDBE) 添加到将要托管队列管理器实例的 IBM i 系统。
 - a) 在 ALPHA 上，创建与 BETA 的连接。

```
ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
```

- b) 在 BETA 上，创建与 ALPHA 的连接。

```
ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
```

3. 在 ALPHA 上创建队列管理器 QM1，在 GAMMA 上保存队列管理器数据。

```
CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
MQMDIRP(' /QNTC/GAMMA/WMQ')
```

路径使用 NetServer 在其中创建队列管理器数据。

4. 在 ALPHA 上运行。该命令在 BETA 上为其添加远程日志。

```
ADDQMQRN MQMNAME(QM1) RMTJNRDB(BETA)
```

当 ALPHA 上的活动实例处于 ALPHA 时，在 ALPHA 上的本地日志中创建日志项。将 ALPHA 上的本地日志复制到 BETA 上的远程日志。

5. 使用此命令可检查针对 ALPHA 创建的 IBM MQ 配置数据。

下一步需要这些信息。

在此示例中，将在 ALPHA 上为以下对象创建以下配置：

```
Name=QM1
Prefix=/QIBM/UserData/mqm
Library=QMOM1
Directory=QM1
DataPath= /QNTC/GAMMA/WMQ /QM1
```

6. 使用该命令在 BETA 上创建 QM1 的队列管理器实例。在 BETA 上运行以下命令以修改 BETA 上的队列管理器控制信息。

```
ADDQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMOM1)
DATAPATH('/QNTC/GAMMA/WMQ /QM1')
```

提示：复制并粘贴配置信息。队列管理器节在 ALPHA 和 BETA 上相同。

7. 在 BETA 上运行。此命令在 BETA 上添加本地日志，并在 ALPHA 上添加远程日志。

```
ADDQMQRN MQMNAME(QM1) RMTJNRDB(ALPHA)
```

当 BETA 上的活动实例时，在 BETA 上的本地日志中创建日志项。BETA 上的本地日志将复制到 ALPHA 上的远程日志。

注：作为替代方法，您可能希望使用异步日志记录来设置从 BETA 到 ALPHA 的远程日志记录。

使用此命令来设置从 BETA 到 ALPHA 的异步日志记录，而不是步骤第 259 页的『7』中的命令。

```
ADDQMQRN MQMNAME(QM1) RMTJNRDB(ALPHA) RMTJNDLV(*ASYNC)
```

如果服务器或 ALPHA 上的日志记录是故障的根源，那么 BETA 将在不等待将新日志项复制到 ALPHA 的情况下启动。

当 ALPHA 再次联机时，使用该命令将复制方式切换到 *SYNC。

使用第 254 页的『IBM i 上 ASP 的镜像日志配置』中的信息来决定是同步，异步还是同时对日记帐进行镜像。缺省情况是同步复制，远程日志的响应等待时间为 60 秒。

8. 验证 ALPHA 和 BETA 上的日志是否已启用，以及远程日志复制的状态是否为。

- a) 在 ALPHA 上：

```
WRKMQRN MQMNAME(QM1)
```

- b) 在 BETA 上：

```
WRKMQRN MQMNAME(QM1)
```

9. 在 ALPHA 和 BETA 上启动队列管理器实例。

- a) 在 ALPHA 上启动第一个实例，使其成为活动实例。允许切换到备用实例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

b) 在 BETA 上启动第二个实例，使其成为备用实例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

结果

用于检查队列管理器状态:

1. 应该是 ALPHA 上的队列管理器实例的状态。
2. BETA 上的队列管理器实例的状态应该为。

示例

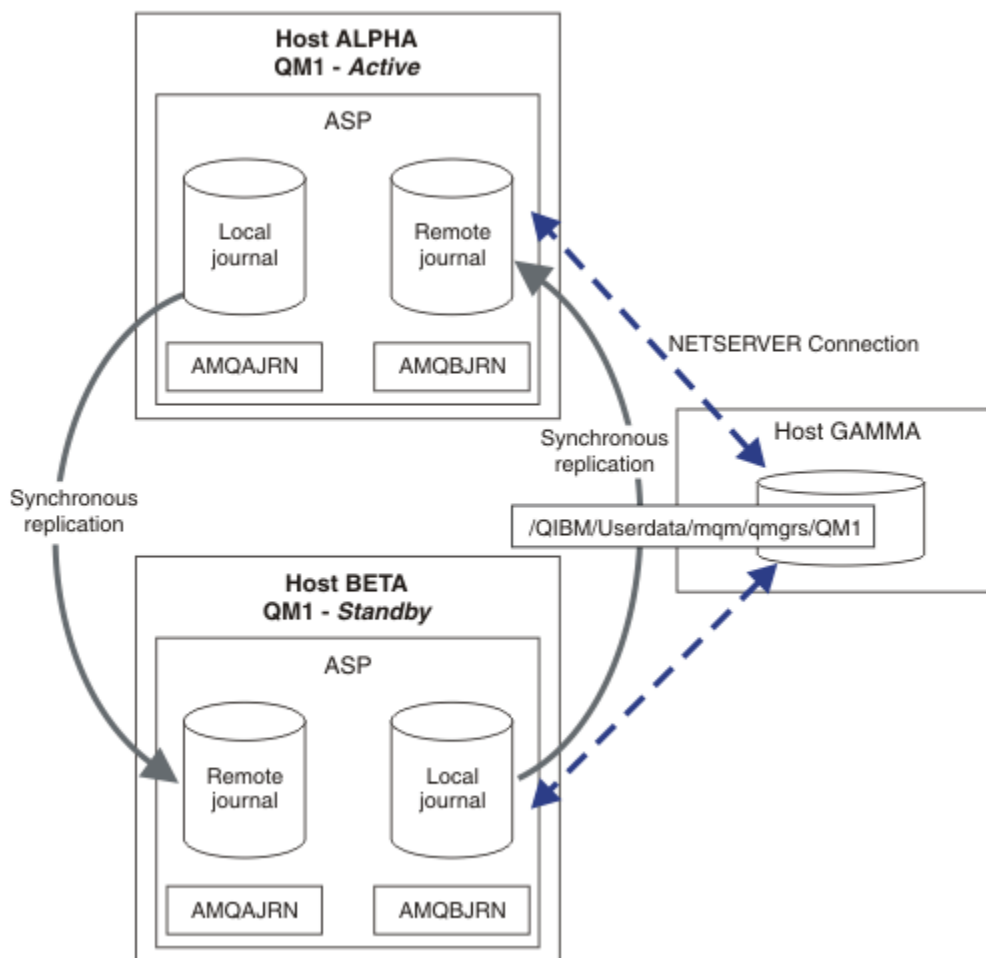


图 37: 镜像日志配置

下一步做什么

- 验证活动实例和备用实例是否自动切换。您可以运行样本高可用性样本程序以测试切换; 请参阅 [高可用性样本程序](#)。样本程序是 "C" 客户机。您可以从 Windows 或 Unix 平台运行这些命令。
 1. 启动高可用性样本程序。

2. 在 ALPHA 上，结束请求切换的队列管理器:

```
ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
```

3. 检查 BETA 上的实例是否处于活动状态。
4. 在 ALPHA 上重新启动

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

- 查看备用高可用性配置:

1. 使用 NetServer 将队列管理器数据放在 Windows 服务器上。
2. 将日志存储在独立 ASP 上，而不是使用远程日志记录来对队列管理器日志进行镜像。使用 IBM i 集群将独立 ASP 从 ALPHA 传输到 BETA。

IBM i 在 IBM i 上使用 NetServer 和日志镜像将单实例队列管理器转换为多实例队列管理器。将单个实例队列管理器转换为多实例队列管理器。将队列管理器数据移至由 NetServer 连接的网络共享。使用远程日志记录将队列管理器日志镜像到第二个 IBM i 服务器。

开始之前

1. 该任务需要三个 IBM i 服务器。示例中的服务器 ALPHA 上的现有 IBM MQ 安装必须至少为 IBM WebSphere MQ 7.0.1 Fix Pack 1。ALPHA 正在运行示例中名为 QM1 的队列管理器。
2. 在示例中的第二个 IBM i 服务器 BETA 上安装 IBM MQ。
3. 第三个服务器是 IBM i 服务器，由 NetServer 连接到 ALPHA 和 BETA。它用于共享队列管理器数据。它不必具有 IBM MQ 安装。作为临时步骤在服务器上安装 IBM MQ 以设置队列管理器目录和许可权非常有用。
4. 确保 QMQM 用户概要文件在所有三个服务器上都具有相同的密码。
5. 安装 IBM i NetServer; 请参阅 [i5/OS NetServer](#)。

关于此任务

执行以下步骤以将单个实例队列管理器转换为第 264 页的图 38 中所示的多实例队列管理器。将在任务中删除单实例队列管理器，然后重新创建，将队列管理器数据存储在由 NetServer 连接的网络共享上。此过程比使用 CPY 命令将队列管理器目录和文件移动到网络共享更可靠。

- 创建从 ALPHA 和 BETA 到 GAMMA 上用于存储队列管理器数据的目录共享的连接。此任务还设置必需的许可权，用户概要文件和密码。
- 将关系数据库条目 (RDBE) 添加到要运行队列管理器实例的 IBM i 系统。RDBE 条目用于连接到用于远程日志记录的 IBM i 系统。
- 保存队列管理器日志和定义，停止队列管理器，然后将其删除。
- 重新创建队列管理器，将队列管理器数据存储在 GAMMA 上的网络共享上。
- 将队列管理器的第二个实例添加到另一个服务器。
- 在两个 IBM i 服务器上为两个队列管理器实例创建远程日志。每个队列管理器都写入本地日志。本地日志将复制到远程日志。命令 **ADDQMJRN** 简化了添加日志和连接的过程。
- 启动队列管理器，允许备用实例。

注:

在任务的步骤第 262 页的『4』中，您将删除单个实例队列管理器 QM1。删除队列管理器将删除队列上的所有持久消息。因此，在转换队列管理器之前，请完成处理队列管理器存储的所有消息。如果无法处理所有消息，请在步骤第 262 页的『4』之前备份队列管理器库。在步骤第 262 页的『5』之后复原队列管理器库。

注:

在任务的步骤 第 262 页的『5』中，您将重新创建 QM1。虽然队列管理器具有相同的名称，但它具有不同的队列管理器标识。队列管理器集群使用队列管理器标识。要在集群中删除并重新创建队列管理器，必须首先从集群中除去队列管理器；请参阅 [从集群中除去队列管理器: 备用方法](#) 或 [从集群中除去队列管理器](#)。重新创建队列管理器后，将其添加到集群。虽然它具有与以前相同的名称，但它似乎是集群中其他队列管理器的新队列管理器。

过程

1. 执行任务 第 248 页的『在 IBM i 上使用 NetServer 为队列管理器数据创建网络共享』。

因此，ALPHA 和 BETA 具有指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 的共享 /QNTC/GAMMA/WMQ。用户概要文件 QMQM 和 QMQMADM 具有必需的许可权，并且 QMQM 在所有三个系统上都具有匹配的密码。

2. 将关系数据库条目 (RDBE) 添加到将要托管队列管理器实例的 IBM i 系统。

- a) 在 ALPHA 上，创建与 BETA 的连接。

```
ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
```

- b) 在 BETA 上，创建与 ALPHA 的连接。

```
ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
```

3. 创建用于重新创建队列管理器对象的脚本。

```
QSAVEQMGR LCLQMGRNAM(QM1) FILENAME('*CURLIB/QMQSC(QM1)')
OUTPUT(*REPLACE) MAKEAUTH(*YES) AUTHFN('*CURLIB/QMAUT(QM1)')
```

4. 停止队列管理器并将其删除。

```
ENDMQM MQMNAME(QM1) OPTION(*IMMED) ENDCCTJOB(*YES) RCDMQMIMG(*YES) TIMEOUT(15)
DLTMQM MQMNAME(QM1)
```

5. 在 ALPHA 上创建队列管理器 QM1，在 GAMMA 上保存队列管理器数据。

```
CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
MQMDIRP('/QNTC/GAMMA/WMQ')
```

路径使用 NetServer 在其中创建队列管理器数据。

6. 根据保存的定义为 QM1 重新创建队列管理器对象。

```
STRMQMQSC SRCMBR(QM1) SRCFILE(*CURLIB/QMQSC) MQMNAME(QM1)
```

7. 应用来自已保存信息的权限。

- a) 编译已保存的授权程序。

```
CRTCLPGM PGM(*CURLIB/QM1) SRCFILE(*CURLIB/QMAUT)
SRCMBR(QM1) REPLACE(*YES)
```

- b) 运行程序以应用授权。

```
CALL PGM(*CURLIB/QM1)
```

- c) 刷新 QM1 的安全性信息。

```
RFRMQMAUT MQMNAME(QM1)
```

- 在 ALPHA 上运行。该命令在 BETA 上为其添加远程日志。

```
ADDQMJRNRN MQMNAME(QM1) RMTJRNRDB(BETA)
```

当 ALPHA 上的活动实例处于 ALPHA 时，在 ALPHA 上的本地日志中创建日志项。将 ALPHA 上的本地日志复制到 BETA 上的远程日志。

- 使用此命令可检查针对 ALPHA 创建的 IBM MQ 配置数据。

下一步需要这些信息。

在此示例中，将在 ALPHA 上为以下对象创建以下配置：

```
Name=QM1
Prefix=/QIBM/UserData/mqm
Library=QM1
Directory=QM1
DataPath= /QNTC/GAMMA/WMQ /QM1
```

- 使用该命令在 BETA 上创建 QM1 的队列管理器实例。在 BETA 上运行以下命令以修改 BETA 上的队列管理器控制信息。

```
ADDQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QM1)
DATAPATH('/QNTC/GAMMA/WMQ /QM1')
```

提示：复制并粘贴配置信息。队列管理器节在 ALPHA 和 BETA 上相同。

- 在 BETA 上运行。此命令在 BETA 上添加本地日志，并在 ALPHA 上添加远程日志。

```
ADDQMJRNRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
```

当 BETA 上的活动实例时，在 BETA 上的本地日志中创建日志项。BETA 上的本地日志将复制到 ALPHA 上的远程日志。

注：作为替代方法，您可能希望使用异步日志记录来设置从 BETA 到 ALPHA 的远程日志记录。

使用此命令来设置从 BETA 到 ALPHA 的异步日志记录，而不是步骤 [第 259 页的『7』](#) 中的命令。

```
ADDQMJRNRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNCR)
```

如果服务器或 ALPHA 上的日志记录是故障的根源，那么 BETA 将在不等待将新日志项复制到 ALPHA 的情况下启动。

当 ALPHA 再次联机时，使用该命令将复制方式切换到 *SYNCR。

使用 [第 254 页的『IBM i 上 ASP 的镜像日志配置』](#) 中的信息来决定是同步，异步还是同时对日记帐进行镜像。缺省情况是同步复制，远程日志的响应等待时间为 60 秒。

- 验证 ALPHA 和 BETA 上的日志是否已启用，以及远程日志复制的状态是否为。

- 在 ALPHA 上：

```
WRKMQJRNRN MQMNAME(QM1)
```

- 在 BETA 上：

```
WRKMQJRNRN MQMNAME(QM1)
```

- 在 ALPHA 和 BETA 上启动队列管理器实例。

- 在 ALPHA 上启动第一个实例，使其成为活动实例。允许切换到备用实例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

b) 在 BETA 上启动第二个实例，使其成为备用实例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

结果

用于检查队列管理器状态:

1. 应该是 ALPHA 上的队列管理器实例的状态。
2. BETA 上的队列管理器实例的状态应该为。

示例

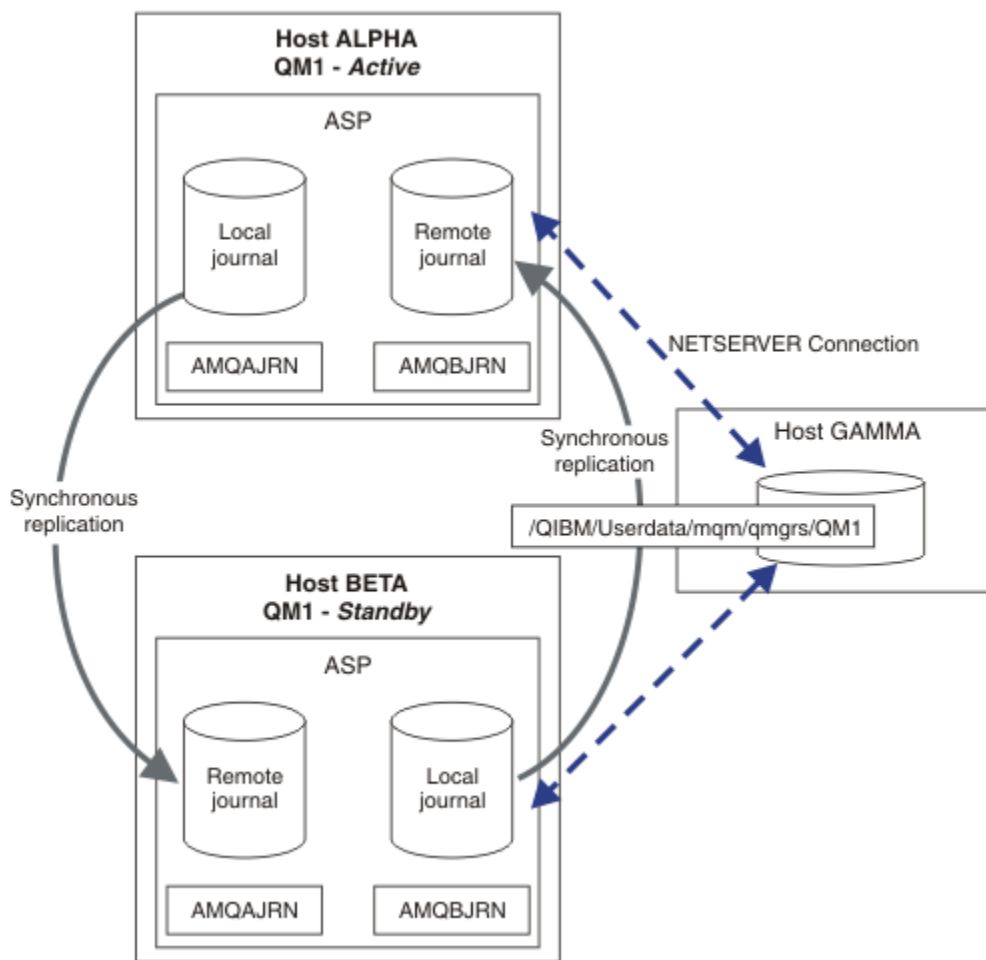


图 38: 镜像日志配置

下一步做什么

- 验证活动实例和备用实例是否自动切换。您可以运行样本高可用性样本程序以测试切换; 请参阅 [高可用性样本程序](#)。样本程序是 "C" 客户机。您可以从 Windows 或 Unix 平台运行这些命令。
 1. 启动高可用性样本程序。

2. 在 ALPHA 上，结束请求切换的队列管理器:

```
ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
```

3. 检查 BETA 上的实例是否处于活动状态。

4. 在 ALPHA 上重新启动

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

• 查看备用高可用性配置:

1. 使用 NetServer 将队列管理器数据放在 Windows 服务器上。
2. 将日志存储在独立 ASP 上，而不是使用远程日志记录来对队列管理器日志进行镜像。使用 IBM i 集群将独立 ASP 从 ALPHA 传输到 BETA。

IBM i 上的交换式独立 ASP 日志配置

您不需要复制独立 ASP 日志来创建多实例队列管理器配置。您需要自动执行将独立 ASP 从活动队列管理器传输到备用队列管理器的方法。存在可使用独立 ASP 的备用高可用性解决方案，并非所有这些解决方案都需要使用多实例队列管理器。

使用独立 ASP 时，不需要对队列管理器日志进行镜像。如果已安装集群管理，并且主管队列管理器实例的服务器位于同一集群资源组中，那么在运行活动实例的主机发生故障时，可以在活动服务器的短距离内将队列管理器日志自动传输到另一个服务器。您还可以手动传输日志 (作为计划的开关的一部分)，也可以编写命令过程以通过编程方式传输独立 ASP。

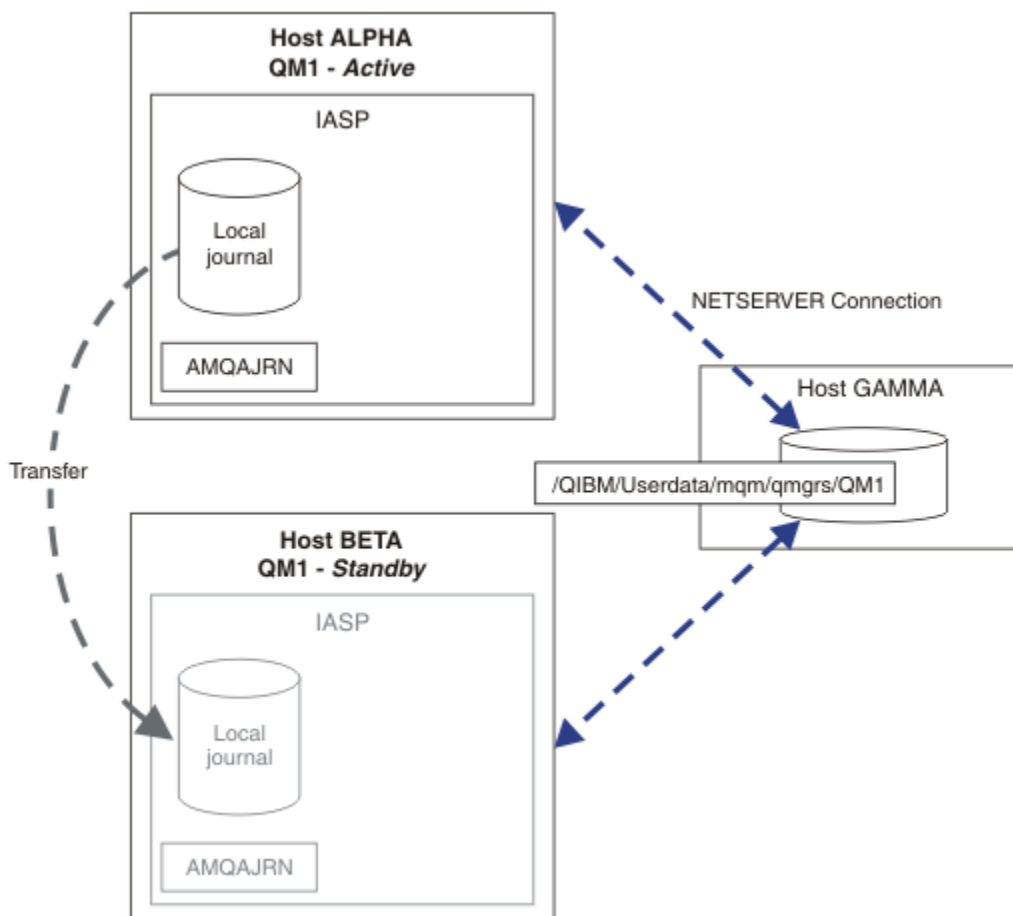


图 39: 使用独立 ASP 传输队列管理器日志

对于多实例队列管理器操作，队列管理器数据必须存储在共享文件系统上。文件系统可以托管在各种不同的平台上。不能将多实例队列管理器数据存储在 ASP 或独立 ASP 上。

共享文件系统在配置中执行两个角色：在队列管理器的所有实例之间共享相同的队列管理器数据。文件系统必须具有强大的锁定协议，以确保队列管理器的一个实例在启动后能够访问队列管理器数据。如果队列管理器发生故障，或者与文件服务器的通信中断，那么文件系统必须释放对不再与文件系统通信的活动实例所持有的队列管理器数据的锁定。然后，备用队列管理器实例可以获得对队列管理器数据的读/写访问权。文件系统协议必须符合一组规则才能正确使用多实例队列管理器；请参阅第 247 页的『IBM i 上高可用性解决方案的组件』。

锁定机制对启动队列管理器命令进行序列化，并控制队列管理器的哪个实例处于活动状态。一旦队列管理器处于活动状态，它就会从您或 HA 集群已传输到备用服务器的本地日志重建其队列。正在等待重新连接到同一队列管理器的可重新连接客户机将重新连接，并且将回退任何进行中的事务。将启动配置为作为队列管理器服务启动的应用程序。

您需要确保通过配置集群资源管理器或手动传输独立 ASP，将独立 ASP 上失败的活动队列管理器实例中的本地日志传输到托管新激活的备用队列管理器实例的服务器。如果您决定将独立 ASP 用于备份和灾难恢复，并将远程日志镜像用于多实例队列管理器配置，那么使用独立 ASP 并不排除配置远程日志和镜像。

如果您已选择使用独立 ASP，那么可以考虑使用备用高可用性配置。第 269 页的『独立 ASP 和高可用性』中描述了这些解决方案的背景。

1. 完全在独立 ASP 上安装和配置单个实例队列管理器，并使用 IBM i 高可用性服务使队列管理器故障转移，而不是使用多实例队列管理器。您可能需要使用队列管理器监视器来扩充解决方案，以检测队列管理器是否独立于服务器发生故障。这是 *Supportpac MC41: 为 iSeries 配置 IBM MQ 以实现高可用性*。
2. 使用独立 ASP 和跨站点镜像 (XSM) 来镜像独立 ASP，而不是在本地总线上切换独立 ASP。这将独立 ASP 解决方案的地理范围扩展到长距离写入日志记录所需要的时间。

IBM i 在 IBM i 上使用独立 ASP 和 NetServer 创建多实例队列管理器

创建要在两个 IBM i 服务器上运行的多实例队列管理器。队列管理器数据使用 NetServer 存储在 IBM i 服务器上。队列管理器日志存储在独立 ASP 上。使用 IBM i 集群或手动过程将包含队列管理器日志的独立 ASP 传输到其他 IBM i 服务器。

开始之前

1. 该任务需要三个 IBM i 服务器。在其中两个上安装 IBM MQ，即示例中的 ALPHA 和 BETA。IBM MQ 必须至少为 V 7.0.1.1。
2. 第三个服务器是 IBM i 服务器，由 NetServer 连接到 ALPHA 和 BETA。它用于共享队列管理器数据。它不必具有 IBM MQ 安装。作为临时步骤在服务器上安装 IBM MQ 以设置队列管理器目录和许可权非常有用。
3. 确保 QMQM 用户概要文件在所有三个服务器上都具有相同的密码。
4. 安装 IBM i NetServer；请参阅 *i5/OS NetServer*。
5. 创建过程将以独立 ASP 从发生故障的队列管理器传输到正在接管的备用数据库。您可能在 *SupportPac MC41: 为 iSeries 配置 IBM MQ 以实现高可用性* 有助于设计独立 ASP 传输过程。

关于此任务

执行以下步骤以创建第 268 页的图 40 中显示的配置。使用 IBM i NetServer 连接队列管理器数据。

- 创建从 ALPHA 和 BETA 到 GAMMA 上用于存储队列管理器数据的目录共享的连接。此任务还设置必需的许可权，用户概要文件和密码。
- 在 IBM i 服务器 ALPHA 上创建队列管理器 QM1。
- 在另一个 IBM i 服务器 BETA 上添加 QM1 的队列管理器控制信息。
- 启动队列管理器，允许备用实例。

过程

1. 执行任务第 248 页的『在 IBM i 上使用 NetServer 为队列管理器数据创建网络共享』。

因此，ALPHA 和 BETA 具有指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 的共享 /QNTC/GAMMA/WMQ。用户概要文件 QMQM 和 QMQMADM 具有必需的许可权，并且 QMQM 在所有三个系统上都具有匹配的密码。

2. 在 ALPHA 上创建队列管理器 QM1，在 GAMMA 上保存队列管理器数据。

```
CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
MQMDIRP(' /QNTC/GAMMA/WMQ')
```

路径使用 NetServer 在其中创建队列管理器数据。

3. 使用此命令可检查针对 ALPHA 创建的 IBM MQ 配置数据。

下一步需要这些信息。

在此示例中，将在 ALPHA 上为以下对象创建以下配置：

```
Name=QM1
Prefix=/QIBM/UserData/mqm
Library=QMQM1
Directory=QM1
DataPath= /QNTC/GAMMA/WMQ /QM1
```

4. 使用该命令在 BETA 上创建 QM1 的队列管理器实例。在 BETA 上运行以下命令以修改 BETA 上的队列管理器控制信息。

```
ADDQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH('/QNTC/GAMMA/WMQ /QM1')
```

提示：复制并粘贴配置信息。队列管理器节在 ALPHA 和 BETA 上相同。

5. 在 ALPHA 和 BETA 上启动队列管理器实例。

- a) 在 ALPHA 上启动第一个实例，使其成为活动实例。允许切换到备用实例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

- b) 在 BETA 上启动第二个实例，使其成为备用实例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

结果

用于检查队列管理器状态：

1. 应该是 ALPHA 上的队列管理器实例的状态。
2. BETA 上的队列管理器实例的状态应该为。

示例

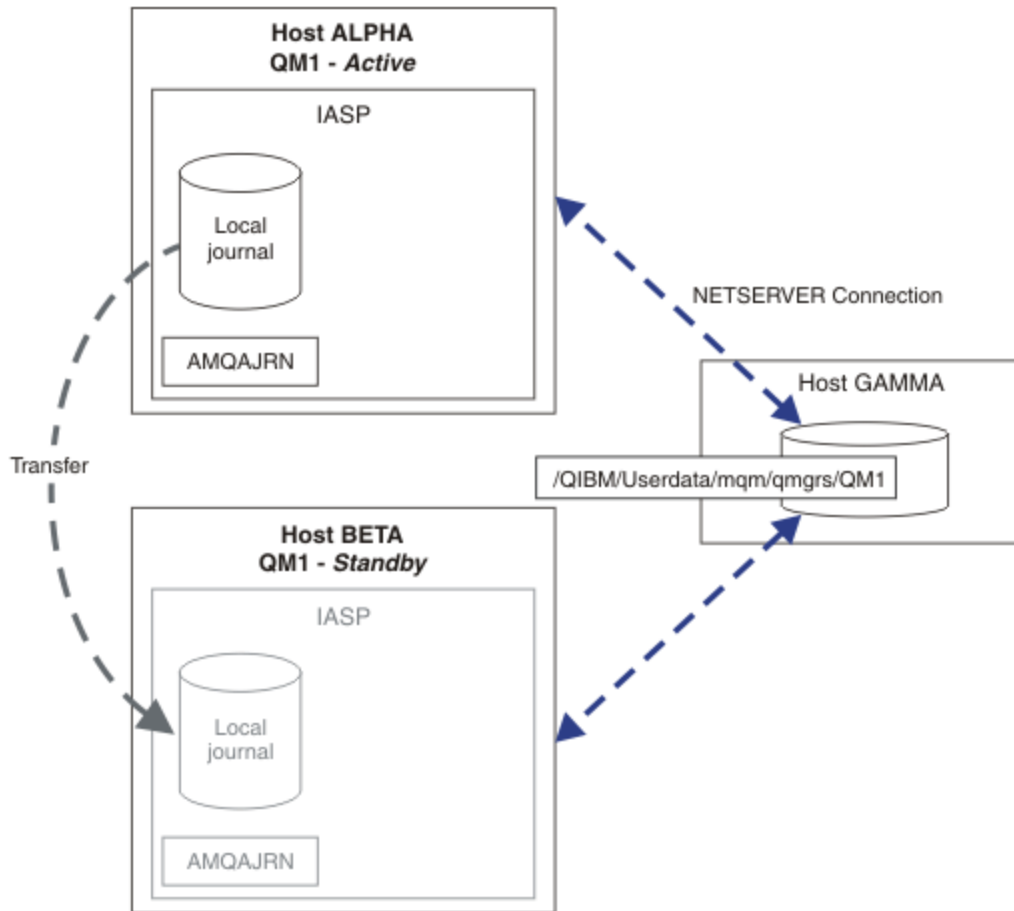


图 40: 使用独立 ASP 传输队列管理器日志

下一步做什么

- 验证活动实例和备用实例是否自动切换。您可以运行样本高可用性样本程序以测试切换; 请参阅 [高可用性样本程序](#)。样本程序是 "C" 客户机。您可以从 Windows 或 Unix 平台运行这些命令。

1. 启动高可用性样本程序。
2. 在 ALPHA 上, 结束请求切换的队列管理器:

```
ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
```

3. 检查 BETA 上的实例是否处于活动状态。
4. 在 ALPHA 上重新启动

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

- 查看备用高可用性配置:
 1. 使用 NetServer 将队列管理器数据放在 IBM i 服务器上。
 2. 使用远程日志记录将日志镜像到备用服务器, 而不是使用独立 ASP 将队列管理器日志传输到备用服务器。

IBM i 独立 ASP 和高可用性

独立 ASP 支持在服务器之间移动应用程序和数据。独立 ASP 的灵活性意味着它们是某些 IBM i 高可用性解决方案的基础。在考虑是将 ASP 还是独立 ASP 用于队列管理器日志时，应考虑基于独立 ASP 的其他高可用性配置。

辅助存储池 (ASP) 是 IBM i 体系结构的构建块。磁盘单元分组在一起以形成单个 ASP。通过将对象放置在不同的 ASP 中，可以保护一个 ASP 中的数据不受另一个 ASP 中磁盘故障的影响。

每个 IBM i 服务器都至少有一个基本 ASP (称为系统 ASP)。它被指定为 ASP1，有时被称为 *SYSBAS。您最多可以配置 31 个其他基本用户 ASP，这些 ASP 无法从应用程序的角度与系统 ASP 区分开来，因为它们共享相同的名称空间。通过使用多个基本 ASP 在多个磁盘上分发应用程序，可以提高性能并缩短恢复时间。使用多个基本 ASP 还可以对磁盘故障提供某种程度的隔离，但它不会整体提高可靠性。

独立 ASP 是特殊类型的 ASP。它们通常称为独立磁盘池。独立磁盘池是 IBM i 高可用性的关键组件。您可以存储数据和应用程序，这些数据和应用程序将自己视为独立于它们在独立磁盘存储单元上连接到的当前系统。您可以配置可切换或不可切换的独立 ASP。从可用性角度来看，您通常只关心可切换的独立 ASP，这些 ASP 可以自动从服务器传输到服务器。因此，您可以将独立 ASP 上的应用程序和数据从服务器移至服务器。

与基本用户 ASP 不同，独立 ASP 与系统 ASP 不共享同一名称空间。使用用户 ASP 的应用程序需要更改才能使用独立 ASP。您需要验证您的软件以及您使用的第三方软件在独立 ASP 环境中工作。

当独立 ASP 连接到另一服务器时，独立 ASP 的名称空间必须与系统 ASP 的名称空间组合。此过程称为独立 ASP 联机。您可以使独立 ASP 联机，而无需对服务器进行 IPL。需要集群支持才能自动将独立 ASP 从一个服务器传输到另一个服务器。

使用独立 ASP 构建可靠解决方案

将日志记录到独立 ASP，而不是将日志记录到 ASP 并使用日志复制，提供了另一种方法来为备用队列管理器提供来自失败队列管理器实例的本地日志副本。要自动将独立 ASP 传输到另一个服务器，您需要已安装并配置集群支持。有许多基于集群支持和低级别磁盘镜像的独立 ASP 的高可用性解决方案，您可以使用多实例队列管理器来组合或替代这些解决方案。

以下列表描述了基于独立 ASP 构建可靠解决方案所需的组件。

日志记录

队列管理器和其他应用程序使用本地日志将持久数据安全地写入磁盘，以防止由于服务器故障而丢失内存中的数据。这有时被称为时间点一致性。它不保证一段时间内发生的多个更新的一致性。

Commitment Control

通过使用全局事务，您可以协调对消息和数据库的更新，以便写入日志的数据一致。它通过使用两阶段落实协议来提供一段时间内的一致性。

交换磁盘

交换磁盘由 HA 集群中的设备集群资源组 (CRG) 管理。在发生意外中断的情况下，CRG 会自动将独立 ASP 切换到新服务器。CRG 在地理上受限于本地 IO 总线的范围。

通过在可切换的独立 ASP 上配置本地日志，可以将日志传输到其他服务器，并恢复处理消息。除非独立 ASP 发生故障，否则不会丢失对没有同步点控制或通过同步点控制落实的持久消息所作的更改。

如果在可切换的独立 ASP 上同时使用日志记录和落实控制，那么可以将数据库日志和队列管理器日志传输到其他服务器，并恢复处理事务，而不会丢失一致性或已落实的事务。

跨站点镜像 (XSM)

XSM 通过 TCP/IP 网络将主独立 ASP 镜像到地理上远程的辅助独立 ASP，并在发生故障时自动传输控制。您可以选择配置同步或异步镜像。同步镜像会降低队列管理器的性能，因为在生产系统上的写操作完成之前会对数据进行镜像，但它确实保证辅助独立 ASP 是最新的。而如果使用异步镜像，那么不能保证辅助独立 ASP 是最新的。异步镜像可保持辅助独立 ASP 的一致性。

有三种 XSM 技术。

地理镜像

地理镜像是集群的扩展，使您能够跨广泛区域切换独立 ASP。它同时具有同步和异步方式。只能在同步方式下保证高可用性，但独立 ASP 的分离可能会对性能产生太大影响。您可以将地理镜像与交换磁盘相结合，以提供本地高可用性和远程灾难恢复。

高速镜像

Metro Mirror 是设备级别的服务，它提供比本地总线更长距离的快速本地同步镜像。您可以将其与多实例队列管理器组合，以提供队列管理器的高可用性，并通过具有独立 ASP 的两个副本来实现队列管理器日志的高可用性。

全局镜像

全局镜像提供异步镜像的设备级别服务，适用于更长距离的备份和灾难恢复，但并非高可用性的正常选择，因为它仅维护时间点一致性而不是货币。

您应该考虑的关键决策点是，

ASP 还是独立 ASP?

您无需运行 IBM i HA 集群即可使用多实例队列管理器。如果您已在使用独立 ASP，或者对于需要独立 ASP 的其他应用程序有可用性要求，那么可以选择独立 ASP。可能值得将独立 ASP 与多实例队列管理器组合，以替换队列管理器监视，作为检测队列管理器故障的方法。

可用性?

什么是恢复时间目标 (RTO)? 如果您需要出现近乎不间断的行为，那么哪个解决方案具有最快的恢复时间?

日志可用性?

如何消除期刊作为单一的故障点。您可以采用硬件解决方案 (使用 RAID 1 设备或更高版本)，也可以使用副本日志或磁盘镜像来组合或使用软件解决方案。

距离?

活动队列管理器实例与备用队列管理器实例之间的距离。您的用户是否可以容忍在超过约 250 米的距离上同步复制的性能下降?

技能?

需要完成一些工作，以自动执行定期维护和实施解决方案所涉及的管理任务。对于基于 ASP 和独立 ASP 的解决方案，执行自动化所需的技能有所不同。

在 IBM i 上删除多实例队列管理器

在删除多实例队列管理器之前，请停止远程日志记录，并删除队列管理器实例。

开始之前

1. 在此示例中，在服务器 ALPHA 和 BETA 上定义了 QM1 队列管理器的两个实例。ALPHA 是活动实例，BETA 是备用实例。与队列管理器 QM1 关联的队列管理器数据使用 NetServer 存储在 IBM i 服务器 GAMMA 上。请参阅第 258 页的『在 IBM i 上使用日志镜像和 NetServer 创建多实例队列管理器』。
2. 必须连接 ALPHA 和 BETA，以便 IBM MQ 可以删除定义的任何远程日志。
3. 使用系统命令 **EDTF** 或 **WRKLNK** 验证是否可以访问 /QNTC 目录和服务器目录文件共享

关于此任务

在使用 **DLTMQM** 命令从服务器中删除多实例队列管理器之前，请使用 **RMVMQMINF** 命令除去其他服务器上的任何队列管理器实例。

使用 **RMVMQMINF** 命令除去队列管理器实例时，将删除以 AMQ 为前缀并与该实例关联的本地和远程日志。还会删除有关服务器本地队列管理器实例的配置信息。

请勿在保存队列管理器的其余实例的服务器上运行 **RMVMQMINF** 命令。这样做会阻止 **DLTMQM** 正常工作。

使用 **DLTMQM** 命令删除队列管理器。将从网络共享中删除队列管理器数据。将删除以 AMQ 为前缀并与实例关联的本地和远程日志。**DLTMQM** 还会删除有关服务器本地队列管理器实例的配置信息。

在此示例中，只有两个队列管理器实例。IBM MQ 支持正在运行的多实例配置，该配置具有一个活动队列管理器实例和一个备用实例。如果已创建要在运行配置中使用的其他队列管理器实例，请先使用 **RMVMQMINF** 命令将其除去，然后再删除其余实例。

过程

1. 在每个服务器上运行 **CHGMQMJRN RMTJRNSTS (*INACTIVE)** 命令，以使队列管理器实例之间的远程日志记录处于不活动状态。

a) 在 ALPHA 上:

```
CHGMQMJRN MQMNAME('QM1')
RMTJRN RDB('BETA') RMTJRNSTS(*INACTIVE)
```

b) 在 BETA 上:

```
CHGMQMJRN MQMNAME('QM1')
RMTJRN RDB('ALPHA') RMTJRNSTS(*INACTIVE)
```

2. 在活动队列管理器实例 ALPHA 上运行 **ENDMQM** 命令，以停止 QM1 的两个实例。

```
ENDMQM MQMNAME(QM1) OPTION(*IMMED) INSTANCE(*ALL) ENDCCTJOB(*YES)
```

3. Run the **RMVMQMINF** command on ALPHA to remove the queue manager resources for the instance from ALPHA and BETA.

```
RMVMQMINF MQMNAME(QM1)
```

RMVMQMINF 从 ALPHA 中除去 QM1 的队列管理器配置信息。如果日志名称以 AMQ 为前缀，那么将从 ALPHA 中删除与 QM1 关联的本地日志。如果日志名称以 AMQ 作为前缀，并且已创建远程日志，那么它还会从 BETA 中除去远程日志。

4. 在 BETA 上运行 **DLTMQM** 命令以删除 QM1。

```
DLTMQM MQMNAME(QM1)
```

DLTMQM 从 GAMMA 上的网络共享中删除队列管理器数据。它将从 BETA 中除去 QM1 的队列管理器配置信息。如果日志名称以 AMQ 为前缀，那么将从 BETA 中删除与 QM1 关联的本地日志。如果日志名称以 AMQ 为前缀，并且已创建远程日志，那么它还会从 ALPHA 中除去远程日志。

结果

DLTMQM 和 **RMVMQMINF** 删除由 **CRTMQM** 和 **ADDMQJRN** 创建的本地和远程日志。这些命令还会删除日志接收器。日志和日志接收器必须遵循以 AMQ 开头的名称的命名约定。**DLTMQM** 和 **RMVMQMINF** 从 `mqs.ini` 中除去队列管理器对象，队列管理器数据和队列管理器配置信息。

下一步做什么

另一种方法是在步骤 第 271 页的『1』中取消激活日志记录之后并在结束队列管理器实例之前发出以下命令。或者，如果未遵循命名约定，那么必须按名称删除日志和日志接收器。

1. 在 ALPHA 上:

```
RMVMQMJRN MQMNAME('QM1') RMTJRN RDB('BETA')
```

2. 在 BETA 上:

```
RMVMQMJRN MQMNAME('QM1') RMTJRN RDB('ALPHA')
```

删除日记帐后，继续执行其余步骤。

IBM i 在 IBM i 上备份多实例队列管理器

此过程说明如何备份本地服务器上的队列管理器对象以及网络文件服务器上的队列管理器数据。调整示例以备份其他队列管理器的数据。

开始之前

在此示例中，与队列管理器 QM1 关联的队列管理器数据使用 NetServer 存储在名为 GAMMA 的 IBM i 服务器上。请参阅第 258 页的『在 IBM i 上使用日志镜像和 NetServer 创建多实例队列管理器』。IBM MQ 安装在服务器 ALPHA 和 BETA 上。队列管理器 QM1 是在 ALPHA 和 BETA 上配置的。

关于此任务

IBM i 不支持从远程目录保存数据。使用文件系统服务器本地的备份过程将队列管理器数据保存在远程文件系统中。在此任务中，网络文件系统位于 IBM i 服务器 GAMMA 上。队列管理器数据将备份在 GAMMA 上的保存文件中。

如果网络文件系统在 Windows 或 Linux 上，那么可以将队列管理器数据存储在压缩文件中，然后将其保存。如果您有备份系统（例如 Tivoli Storage Manager），请使用它来备份队列管理器数据。

过程

1. 在 ALPHA 上为与 QM1 关联的队列管理器库创建保存文件。

使用队列管理器库名来命名保存文件。

```
CRTSAVF FILE(QGPL/QMQM1)
```

2. 将队列管理器库保存在 ALPHA 上的保存文件中。

```
SAVLIB LIB(QMQM1) DEV(*SAVF) SAVF(QGPL/QMQM1)
```

3. 在 GAMMA 上为队列管理器数据目录创建保存文件。

使用队列管理器名称来命名保存文件。

```
CRTSAVF FILE(QGPL/QMDQM1)
```

4. 从 GAMMA 上的本地目录保存队列管理器数据的副本。

```
SAV DEV('/QSYS.LIB/QGPL.LIB/QMDQM1.FILE') OBJ('/QIBM/Userdata/mqm/qmgrs/QM1')
```

IBM i 用于设置多实例队列管理器的命令

IBM MQ 具有用于简化配置日志复制，添加新队列管理器实例以及配置队列管理器以使用独立 ASP 的命令。

用于创建和管理本地和远程日志的日志命令如下：

ADDMQMJRN

通过此命令，您可以为队列管理器实例创建指定的本地和远程日志，并配置复制是同步还是异步，同步超时是什么，以及如果要立即激活远程日志。

CHGMQMJRN

此命令将修改影响副本日志的超时，状态和交付参数。

RMVMQMJRN

从队列管理器实例中除去指定的远程日志。

WRKMQMJRN

列出本地队列管理器实例的本地和远程日志的状态。

使用以下命令添加和管理其他队列管理器实例，这些命令用于修改 `mqs.ini` 文件。

地址 MQMINF

该命令使用您使用 DSPMQMINF 命令从 mqs.ini 文件中抽取的信息在另一 IBM i 服务器上添加新的队列管理器实例。

RMVMQMINF

除去队列管理器实例。使用此命令可以除去现有队列管理器的实例，也可以除去已从其他服务器中删除的队列管理器的配置信息。

CRTMQM 命令有三个参数可帮助配置多实例队列管理器，

MQMDIRP (*DFT | *directory-prefix*)

使用此参数来选择映射到联网存储器上的队列管理器数据的安装点。

ASP (*SYSTEM|*ASPDEV| 辅助存储池编号)

指定 *SYSTEM 或 辅助存储池号 以将队列管理器日志放在系统或基本用户 ASP 上。选择 *ASPDEV 选项，同时使用 **ASPDEV** 参数设置设备名，以将队列管理器日志放在独立 ASP 上。

ASPDEV (*ASP|*device-name*)

指定主或辅助独立 ASP 设备的 *device-name*。选择 *ASP 与指定 **ASP (*SYSTEM)** 具有相同的结果。

IBM i IBM i 上的性能和磁盘故障转移注意事项

使用不同的辅助存储池来提高性能和可靠性。

如果在应用程序中使用大量持久消息或大型消息，那么将这些消息写入磁盘所花费的时间将成为系统性能的重要因素。

确保有足够的磁盘激活来应对这种可能性，或者考虑单独的辅助存储池 (ASP) 来存放队列管理器日志接收器。

您可以在使用 ASP 参数 **CRTMQM** 创建队列管理器时指定存储队列管理器库和日志的 ASP。缺省情况下，队列管理器库和日志以及 IFS 数据存储在系统 ASP 中。

ASP 允许隔离一个或多个特定磁盘机上的对象。这还可以减少由于磁盘介质故障而丢失数据的情况。在大多数情况下，只有存储在受影响 ASP 中的磁盘机上的数据会丢失。

建议将队列管理器库和日志数据存储在与根 IFS 文件系统不同的用户 ASP 中，以提供故障转移和减少磁盘争用。

有关更多信息，请参阅 [备份和恢复](#)。

IBM i 使用 SAVLIB 在 IBM i 上保存 IBM MQ 库

无法使用 SAVLIB LIB(*ALLUSR) 来保存 IBM MQ 库，因为这些库具有以 Q 开头的名称。

您可以使用 SAVLIB LIB(QM*) 来保存所有队列管理器库，但仅当您使用的是 *SAVF 以外的保存设备时。对于 DEV(*SAVF)，必须对系统上的每个队列管理器库使用 SAVLIB 命令。

IBM i 停顿 IBM MQ for IBM i

本节说明如何停顿 (正常结束) IBM MQ for IBM i。

要停顿 IBM MQ for IBM i:

1. 登录到新的交互式 IBM MQ for IBM i 会话，确保您未访问任何对象。
2. 确保您具有：
 - *ALLOBJ 权限或 QMQM 库的对象管理权限
 - 有足够权限使用 ENDSBS 命令
3. 建议所有用户停止 IBM MQ for IBM i。
4. 然后如何继续取决于是否要关闭 (停顿) 单个队列管理器 (其中可能存在其他队列管理器) (请参阅第 274 页的『正在关闭 IBM MQ for IBM i 的单个队列管理器』) 或所有队列管理器 (请参阅第 275 页的『正在关闭 IBM MQ for IBM i 的所有队列管理器』)。

ENDMQM 参数 ENDCCTJOB (*YES)

ENDMQM 参数 ENDCCTJOB (*YES) 在 IBM MQ for IBM i 6.0 和更高版本中的工作方式与先前版本不同。

在先前版本上, 当您指定 ENDCCTJOB (*YES) 时, MQ 会强制终止您的应用程序。

在 IBM MQ for IBM i 6.0 或更高版本上, 当您指定 ENDCCTJOB (*YES) 时, 应用程序不会终止, 而是与队列管理器断开连接。

如果指定 ENDCCTJOB (*YES), 并且有未写入的应用程序检测到队列管理器正在结束, 那么下次发出新的 MQI 调用时, 该调用将返回 MQRC_CONNECTION_BROKEN (2009) 错误。

作为使用 ENDCCTJOB (*YES) 的替代方法, 使用参数 ENDCCTJOB (*NO) 并使用 WRKMQM 选项 22 (使用作业) 来手动结束将阻止队列管理器重新启动的任何应用程序作业。

IBM i 正在关闭 IBM MQ for IBM i 的单个队列管理器

使用此信息可了解三种类型的关闭。

在后续过程中, 我们使用样本队列管理器名称 QMgr1 和样本子系统名称 SUBX。如果需要, 请将这些名称替换为您自己的值。

计划中关机

计划在 IBM i 上关闭队列管理器

1. 在关闭之前, 执行:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
```

2. 要关闭队列管理器, 请执行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

如果 QMgr1 未结束, 那么通道或应用程序可能正忙。

3. 如果必须立即关闭 QMgr1, 请执行以下操作:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)  
ENDCCTJOB(*YES) TIMEOUT(15)
```

计划外关闭

1. 要关闭队列管理器, 请执行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

如果 QMgr1 未结束, 那么通道或应用程序可能正忙。

2. 如果需要立即关闭 QMgr1, 请执行以下操作:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)  
ENDCCTJOB(*YES) TIMEOUT(15)
```

在异常情况下关闭

1. 要关闭队列管理器, 请执行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

如果 QMgr1 未结束, 请继续执行步骤 3, 前提是:

- QMgr1 位于其自己的子系统中，或者
 - 可以结束与 QMgr1 共享同一子系统的所有队列管理器。对所有此类队列管理器使用计划外关闭过程。
2. 对共享子系统的所有队列管理器 (在我们的示例中为 SUBX) 执行该过程中的所有步骤后，请执行：

```
ENDSBS SUBX *IMMED
```

如果此命令无法完成，请使用未规划的关闭过程来关闭所有队列管理器，并在您的机器上执行 IPL。

警告：请勿将 ENDJOBABN 用于由于 ENDJOB 或 ENDSBS 而无法结束的 IBM MQ 作业，除非您准备在之后立即在机器上执行 IPL。

3. 通过执行以下命令来启动子系统：

```
STRSBS SUBX
```

4. 通过执行以下命令，立即关闭队列管理器：

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)  
ENDCCTJOB(*YES) TIMEOUT(10)
```

5. 通过执行以下命令重新启动队列管理器：

```
STRMQM MQMNAME(QMgr1)
```

如果此操作失败，那么您将执行以下操作：

- 已通过执行 IPL 来重新启动机器，或者
- 只有一个队列管理器

通过执行以下命令来整理 IBM MQ 共享内存：

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)  
ENDCCTJOB(*YES) TIMEOUT(15)
```

在重复步骤 5 之前。

如果队列管理器重新启动时间超过几秒，那么 IBM MQ 会将状态消息间歇性地添加到详细描述启动进度的作业日志中。

如果您在重新启动队列管理器时仍遇到问题，请联系 IBM 支持人员。您可能执行的任何进一步操作都会损坏队列管理器，导致 IBM MQ 无法恢复。

IBM i 正在关闭 IBM MQ for IBM i 的所有队列管理器

使用此信息可了解三种类型的关闭。

这些过程几乎与针对单个队列管理器的过程相同，但在可能的情况下使用 *ALL 而不是队列管理器名称，否则，请依次使用重复使用每个队列管理器名称的命令。在整个过程中，我们使用样本队列管理器名称 QMgr1 和样本子系统名称 SUBX。请将这些替换为您自己的。

计划中关机

1. 在关闭前一小时执行：

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNTA(*YES)
```

对要关闭的每个队列管理器重复此操作。

2. 要关闭队列管理器，请执行：

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

对要关闭的每个队列管理器重复此操作; 可以并行运行单独的命令。

如果任何队列管理器未在合理时间 (例如 10 分钟) 内结束, 请继续执行步骤 3。

3. 要立即关闭所有队列管理器, 请执行以下操作:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)  
ENDCCTJOB(*YES) TIMEOUT(15)
```

计划外关闭

1. 要关闭队列管理器, 请执行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

对要关闭的每个队列管理器重复此操作; 可以并行运行单独的命令。

如果队列管理器未结束, 那么通道或应用程序可能正忙。

2. 如果需要立即关闭队列管理器, 请执行以下操作:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)  
ENDCCTJOB(*YES) TIMEOUT(15)
```

在异常情况下关闭

1. 要关闭队列管理器, 请执行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

对要关闭的每个队列管理器重复此操作; 可以并行运行单独的命令。

2. 通过执行以下命令结束子系统 (示例中的 SUBX):

```
ENDSBS SUBX *IMMED
```

对要关闭的每个子系统重复此操作; 可以并行运行单独的命令。

如果此命令无法完成, 请在系统上执行 IPL。

警告: 请勿将 ENDJOBABN 用于由于 ENDJOB 或 ENDSBS 而无法结束的作业, 除非您准备在之后立即在系统上执行 IPL。

3. 通过执行以下命令来启动子系统:

```
STRSBS SUBX
```

对要启动的每个子系统重复此操作。

4. 通过执行以下命令, 立即关闭队列管理器:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)  
ENDCCTJOB(*YES) TIMEOUT(15)
```

5. 通过执行以下命令来重新启动队列管理器:

```
STRMQM MQMNAME(QMgr1)
```

对要启动的每个队列管理器重复此操作。

如果任何队列管理器重新启动时间超过几秒，那么 IBM MQ 将间歇地显示详细说明启动进度的状态消息。

如果您在重新启动任何队列管理器时仍迁到问题，请联系 IBM 支持人员。您可能执行的任何进一步操作都可能损坏队列管理器，使 MQSeries 或 IBM MQ 无法恢复。

z/OS 管理 IBM MQ for z/OS

管理队列管理器和关联资源包含您为了激活和管理这些资源而经常执行的任务。选择您首选的方法来管理队列管理器和关联的资源。

IBM MQ for z/OS 可由随产品提供的一组实用程序和程序控制和管理。可以使用 IBM MQ 脚本 (MQSC) 命令或可编程命令格式 (PCF) 来管理 IBM MQ for z/OS。有关对 IBM MQ for z/OS 使用命令的信息，请参阅第 277 页的『向 IBM MQ for z/OS 发出命令』。

IBM MQ for z/OS 还提供了一组实用程序来帮助您进行系统管理。有关不同实用程序以及如何使用它们的信息，请参阅第 284 页的『IBM MQ for z/OS 实用程序』。

有关如何管理 IBM MQ for z/OS 以及您可能需要执行的不同管理任务的详细信息，请参阅以下链接：

相关概念

第 5 页的『管理 IBM MQ』

要管理 IBM MQ 队列管理器和关联资源，请从可用于激活和管理这些资源的一组任务中选择首选方法。

相关任务

第 121 页的『管理本地 IBM MQ 对象』

您可以管理本地 IBM MQ 对象以支持使用消息队列接口 (MQI) 的应用程序。

第 167 页的『管理远程 IBM MQ 对象』

本节说明如何使用 MQSC 命令来管理远程队列管理器上的 IBM MQ 对象，以及如何使用远程队列对象来控制消息和应答消息的目标。

相关信息

[IBM MQ for z/OS 概念](#)

[规划](#)

[在 z/OS 上规划 IBM MQ 环境](#)

[配置](#)

[配置 z/OS](#)

[可编程命令格式参考](#)

[MQSC 引用](#)

[使用 IBM MQ for z/OS 实用程序](#)

z/OS 向 IBM MQ for z/OS 发出命令

您可以在批处理或交互方式下使用 IBM MQ 脚本命令 (MQSC) 来控制队列管理器。

IBM MQ for z/OS 支持 MQSC 命令，可以从以下源发出这些命令：

- z/OS 控制台或同等版本 (例如 SDSF/TSO)。
- 初始化输入数据集。
- 提供的批处理实用程序 CSQUTIL 处理顺序数据集中的命令列表。
- 适当授权的应用程序，通过将命令作为消息发送到命令输入队列。应用程序可以是下列任何一项：
 - 批处理区域程序
 - CICS 应用程序
 - IMS 应用程序
 - TSO 应用程序
 - 另一个 IBM MQ 系统上的应用程序或实用程序

第 280 页的表 24 汇总了 MQSC 命令以及可从中发出这些命令的源。

IBM MQ for z/OS 操作和控制面板以方便的方式提供了这些命令的许多功能。

使用命令 (直接或间接) 对队列管理器的资源定义所作的更改在重新启动 IBM MQ 子系统时保留。

IBM MQ for z/OS 还支持可编程命令格式 (PCF) 命令。这些操作简化了用于管理 IBM MQ 的应用程序的创建。MQSC 命令采用人类可读的文本格式, 而 PCF 使应用程序能够创建请求并读取应答, 而不必解析文本字符串。与 MQSC 命令一样, 应用程序通过将 PCF 命令作为消息发送到命令输入队列来发出这些命令。有关使用 PCF 命令的更多信息以及这些命令的详细信息, 请参阅 [可编程命令格式参考](#) 文档。

z/OS IBM MQ for z/OS 上的专用和全局定义

在 IBM MQ for z/OS 上定义对象时, 可以选择是要与其他队列管理器共享该定义 (全局定义), 还是仅由一个队列管理器 (专用定义) 使用该对象定义。这称为对象处置。

全局定义

如果队列管理器属于队列共享组, 那么您可以选择与该组的其他成员共享您所创建的任何对象定义。这意味着您必须只定义一次对象, 从而减少整个系统所需的定义总数。

全局对象定义保存在共享存储库 (Db2 共享数据库) 中, 并且可供队列共享组中的所有队列管理器使用。这些对象的处置为 GROUP。

专用定义

如果要创建仅一个队列管理器所需的对象定义, 或者如果队列管理器不是队列共享组的成员, 那么可以创建未与队列共享组的其他成员共享的对象定义。

专用对象定义保留在定义队列管理器的页集零上。这些对象的处置为 QMGR。

您可以为除 CF 结构以外的所有类型的 IBM MQ 对象 (即, 通道, 名称列表, 进程定义, 队列, 队列管理器, 存储类定义和认证信息对象) 以及除队列管理器以外的所有类型的对象的全局定义创建专用定义。

IBM MQ 会自动将组对象的定义复制到使用该组对象的每个队列管理器的页集零。如果需要, 您可以临时更改定义的副本, 并且 IBM MQ 允许您根据需要从存储库副本刷新页集副本。

IBM MQ 始终尝试在启动时从存储库副本刷新页集副本 (对于通道命令, 这是在通道启动程序重新启动时完成的), 或者如果组对象已更改。

注: 仅当创建该定义的副本后组的定义已更改时, 才会从该组的定义中刷新该定义的副本。

这将确保页集副本反映存储库上的版本, 包括队列管理器处于不活动状态时所做的任何更改。通过生成 DEFINE REPLACE 命令来刷新副本, 因此在某些情况下不会执行刷新, 例如:

- 如果打开了队列的副本, 那么更改队列使用情况的刷新将失败。
- 如果队列的副本包含消息, 那么删除该队列的刷新将失败。
- 如果队列的副本需要带有 FORCE 的 ALTER 来对其进行更改。

在这些情况下, 不会对该副本执行刷新, 而是对所有其他队列管理器上的副本执行刷新。

如果关闭队列管理器, 然后单独重新启动, 那么将删除对象的任何本地副本, 除非例如, 队列具有关联的消息。

存在仅适用于本地队列的第三个对象处置。这允许您创建共享队列。共享队列的定义保存在共享存储库中, 并且可供队列共享组中的所有队列管理器使用。此外, 共享队列上的消息也可供队列共享组中的所有队列管理器使用。这在 [共享队列和队列共享组](#) 中进行了描述。共享队列的对象处置为 SHARED。

下表总结了作为队列共享组的成员单独启动的队列管理器的对象处置选项的影响。

| 处置 | 独立队列管理器 | 队列共享组的成员 |
|-------|----------------|-----------------------------------------|
| QMGR | 对象定义保留在页集 0 上。 | 对象定义保留在页集 0 上。 |
| GROUP | 不允许。 | 共享存储库中保留的对象定义。保留在组中每个队列管理器的页集 0 上的本地副本。 |

| 处置 | 独立队列管理器 | 队列共享组的成员 |
|--------|---------|--------------------------------|
| SHARED | 不允许。 | 共享存储库中保留的队列定义。可用于组中任何队列管理器的消息。 |

处理全局定义

如果要更改共享存储库中保存的对象的定义，那么需要指定是要更改存储库上的版本，还是更改页集 0 上的本地副本。使用对象处置作为命令的一部分来执行此操作。

z/OS 将命令定向到 z/OS 上的不同队列管理器

您可以使用 命令作用域 来控制命令在哪个队列管理器上运行。

您可以选择在输入该命令的队列管理器上执行该命令，或者在队列共享组中的另一个队列管理器上执行该命令。您还可以选择在队列共享组中的所有队列管理器上并行发出特定命令。这适用于 MQSC 命令和 PCF 命令。

这由 命令作用域 确定。命令作用域与对象处置一起使用，以确定要使用的对象版本。

例如，您可能想要更改对象的某些属性，该对象的定义保存在共享存储库中。

- 您可能只想更改一个队列管理器上的版本，而不更改存储库上的版本或其他队列管理器正在使用的版本。
- 您可能希望为将来的用户更改共享存储库中的版本，但保留现有副本不变。
- 您可能希望更改共享存储库中的版本，但也希望您的更改立即反映在队列共享组中的所有队列管理器上，这些队列管理器在其页集 0 上保存对象的副本。

使用命令作用域来指定是在此队列管理器，另一个队列管理器还是所有队列管理器上执行该命令。使用对象处置来指定您正在处理的对象是在共享存储库中(组对象)，还是在页集 0 上是本地副本(队列管理器对象)。

您不必指定用于处理共享队列的命令作用域和对象处置，因为队列共享组中的每个队列管理器都将共享队列作为单个队列处理。

z/OS IBM MQ for z/OS 的命令摘要

使用本主题作为主 MQSC 和 PCF 命令的参考。

第 279 页的表 23 汇总了 IBM MQ for z/OS 上可用于变更，定义，删除和显示 IBM MQ 对象的 MQSC 和 PCF 命令。

| MQSC 命令 | 更改 | 定义 | DISPLAY | DELETE |
|----------|----|-------|---------|--------|
| PCF 命令 | 更改 | 创建/复制 | 查询 | 删除 |
| AUTHINFO | X | X | X | X |
| CFSTATUS | | | X | |
| CFSTRUCT | X | X | X | X |
| 通道 | X | X | X | X |
| CHSTATUS | | | X | |
| NAMELIST | X | X | X | X |
| PROCESS | X | X | X | X |
| QALIAS | M | M | M | M |
| QCLUSTER | | | M | |

表 23: 按对象类型列出的主 MQSC 和 PCF 命令的摘要 (继续)

| MQSC 命令 | 更改 | 定义 | DISPLAY | DELETE |
|----------|----|----|---------|--------|
| QLOCAL | M | M | M | M |
| QMGR | X | | X | |
| QMODEL | M | M | M | M |
| QREMOTE | M | M | M | M |
| 队列 | P | P | X | P |
| 队列状态 | | | X | |
| STGCLASS | X | X | X | X |

表符号的键:

- M = 仅 MQSC
- 仅 P = PCF
- X = 两者

有许多其他 MQSC 和 PCF 命令允许您管理其他 IBM MQ 资源，并执行除 [第 279 页的表 23](#) 中概述的操作之外的其他操作。

[第 280 页的表 24](#) 显示了每个 MQSC 命令，以及可以从其中发出每个命令的位置:

- CSQINP1 初始化输入数据集
- CSQINP2 初始化输入数据集
- z/OS 控制台 (或等效控制台)
- SYSTEM.COMMAND.INPUT 队列和命令服务器 (来自应用程序， CSQUTIL 或 CSQINPX 初始化输入数据集)

表 24: 从中运行 MQSC 命令的源

| 命令 | CSQINP1 | CSQINP2 | z/OS 控制台 | 命令输入队列和服务器 |
|----------------|---------|---------|----------|------------|
| 变更授权信息 | | X | X | X |
| ALTER BUFFPOOL | | X | X | X |
| 变更 CFSTRUCT | | X | X | X |
| ALTER CHANNEL | | X | X | X |
| ALTER NAMELIST | | X | X | X |
| 变更 PSID | | | X | X |
| 变更过程 | | X | X | X |
| ALTER QALIAS | | X | X | X |
| ALTER QLOCAL | | X | X | X |
| ALTER QMGR | | X | X | X |
| 变更 QMODEL | | X | X | X |
| 变更 QREMOTE | | X | X | X |
| 变更安全性 | X | X | X | X |
| 变更 STGCLASS | | X | X | X |

表 24: 从中运行 MQSC 命令的源 (继续)

| 命令 | CSQINP1 | CSQINP2 | z/OS 控制台 | 命令输入队列和服务器 |
|-----------------|---------|---------|----------|------------|
| 变更 SUB | | X | X | X |
| ALTER TOPIC | | X | X | X |
| ALTER TRACE | X | X | X | X |
| 归档日志 | X | X | X | X |
| 备份 cfstruct | | | X | X |
| 清除 QLocal | | X | X | X |
| 定义授权信息 | | X | X | X |
| 定义缓冲池 | X | X | | |
| 定义 CFSTRUCT | | X | X | X |
| DEFINE CHANNEL | | X | X | X |
| 定义日志 | | | X | X |
| DEFINE NAMELIST | | X | X | X |
| 定义流程 | | X | X | X |
| DEFINE PSID | X | | X | X |
| 定义 Qalias | | X | X | X |
| DEFINE QLOCAL | | X | X | X |
| 定义 QModel | | X | X | X |
| 定义 QRemote | | X | X | X |
| DEFINE STGCLASS | | X | X | X |
| DEFINE SUB | | | X | X |
| DEFINE TOPIC | | X | X | X |
| 删除授权信息 | | X | X | X |
| 删除 BUFFPOOL | | | X | X |
| 删除 CFSTRUCT | | X | X | X |
| 删除通道 | | | X | X |
| 删除名称列表 | | X | X | X |
| 删除进程 | | X | X | X |
| 删除 PSID | | | X | X |
| 删除 Qalias | | X | X | X |
| 删除 QLOCAL | | X | X | X |
| 删除 QMODEL | | X | X | X |
| 删除 QREMOTE | | X | X | X |
| 删除 STGCLASS | | X | X | X |
| 删除 SUB | | X | X | X |

表 24: 从中运行 MQSC 命令的源 (继续)

| 命令 | CSQINP1 | CSQINP2 | z/OS 控制台 | 命令输入队列和服务器 |
|------------------|---------|---------|----------|------------|
| 删除主题 | | X | X | X |
| 显示归档 | X | X | X | X |
| 显示授权信息 | | X | X | X |
| 显示 CFSTATUS | | | X | X |
| 显示 CFSTRUCT | | X | X | X |
| 显示通道 | | X | X | X |
| DISPLAY CHSTATUS | | | X | X |
| DISPLAY CLUSQMGR | | | X | X |
| DISPLAY CMDSERV | X | X | X | X |
| DISPLAY CONN | | X | X | X |
| 显示中国 | | X | X | X |
| 显示组 | | X | X | X |
| 显示日志 | X | X | X | X |
| 显示 NAMELIST | | X | X | X |
| 显示过程 | | X | X | X |
| 显示 Qalias | | X | X | X |
| 显示 Qcluster | | X | X | X |
| DISPLAY QLOCAL | | X | X | X |
| 显示队列管理器 | | X | X | X |
| 显示 QModel | | X | X | X |
| 显示 QRemote | | X | X | X |
| 显示 QSTATUS | | X | X | X |
| DISPLAY QUEUE | | X | X | X |
| 显示安全性 | | | X | X |
| 显示 STGCLASS | | X | X | X |
| 显示子项 | | X | X | X |
| DISPLAY TOPIC | | X | X | X |
| 显示系统 | X | X | X | X |
| DISPLAY THREAD | | X | X | X |
| DISPLAY TRACE | X | X | X | X |
| 显示使用情况 | | X | X | X |
| 移动 QLocal | | X | X | X |
| Ping 通道 | | | X | X |
| 恢复 BSDS | X | X | X | X |

表 24: 从中运行 MQSC 命令的源 (继续)

| 命令 | CSQINP1 | CSQINP2 | z/OS 控制台 | 命令输入队列和服务器 |
|------------------|---------|---------|----------|------------|
| 恢复 CFSTRUCT | | | X | X |
| 刷新集群 | | X | X | X |
| 刷新队列管理器 | | X | X | X |
| REFRESH SECURITY | | X | X | X |
| 重置通道 | | | X | X |
| Reset Cluster | | X | X | X |
| 重置 QSTATS | | X | X | X |
| 复位 TPIPE | | | X | X |
| 解析通道 | | | X | X |
| 解析不确定 | | X | X | X |
| 恢复队列管理器 | | | X | X |
| Rverify 安全性 | | X | X | X |
| 设置归档 | X | X | X | X |
| 设置日志 | X | X | X | X |
| 设置系统 | X | X | X | X |
| 启动通道 | | | X | X |
| START 中国 | | X | X | X |
| START CMDSERV | X | X | X | |
| 启动侦听器 | | | X | X |
| 开始 QMGR | | | X | |
| 启动跟踪 | X | X | X | X |
| 停止通道 | | | X | X |
| STOP 中国 | | | X | X |
| 停止 CMDSERV | X | X | X | |
| 停止侦听器 | | | X | X |
| STOP QMGR | | | X | X |
| STOP TRACE | X | X | X | X |
| 已暂挂的队列管理器 | | | X | X |

在 MQSC 命令中，每个命令描述都标识可运行该命令的源。

IBM MQ for z/OS 的初始化命令

可以使用初始化命令来控制队列管理器启动。

在队列管理器启动时初始化 IBM MQ 时，将处理初始化输入数据集中的命令。可以从初始化输入数据集发出三种类型的命令：

- 用于定义无法在其他位置定义的 IBM MQ 实体的命令，例如 DEFINE BUFFPOOL。

这些命令必须位于 DD 名称 CSQINP1 所标识的数据集中。它们在初始化的重新启动阶段之前进行处理。它们不能通过控制台，操作和控制面板或应用程序发出。这些命令的响应将写入您在启动式任务过程的 CSQOUT1 语句中引用的连续数据集。

- 用于定义重新启动后可恢复的 IBM MQ 对象的命令。必须在 DD 名称 CSQINP2 所标识的数据集中指定这些定义。它们存储在页集零中。CSQINP2 在初始化的重新启动阶段之后处理。这些命令的响应将写入您在启动式任务过程的 CSQOUT2 语句中引用的连续数据集。
- 用于处理 IBM MQ 对象的命令。还必须在 DD 名称 CSQINP2 所标识的数据集中指定这些命令。例如，IBM MQ 提供的样本包含 ALTER QMGR 命令，用于指定子系统的死信队列。对这些命令的响应将写入 CSQOUT2 输出数据集。

注：如果在 CSQINP2 中定义了 IBM MQ 对象，那么每次启动队列管理器时，IBM MQ 都会尝试重新定义这些对象。如果对象已存在，那么尝试定义这些对象将失败。如果需要在 CSQINP2 中定义对象，那么可以通过使用 DEFINE 命令的 REPLACE 参数来避免此问题，但是，这将覆盖先前运行队列管理器期间进行的任何更改。

IBM MQ for z/OS 随附了样本初始化数据集成员。它们在 [IBM MQ](#) 随附的样本定义中进行了描述。

用于分布式排队的初始化命令

您还可以将 CSQINP2 初始化数据集用于 START CHINIT 命令。如果需要一系列其他命令来定义分布式排队环境（例如，启动侦听器），那么 IBM MQ 将提供称为 CSQINPX 的第三个初始化输入数据集，作为通道启动程序启动式任务过程的一部分进行处理。

在通道启动程序初始化结束时执行数据集中包含的 MQSC 命令，并将输出写入由 CSQOUTX DD 语句指定的数据集。例如，可以使用 CSQINPX 初始化数据集来启动侦听器。

IBM MQ for z/OS 随附了样本通道启动程序初始化数据集成员。在 [随 IBM MQ](#) 提供的样本定义中进行了描述。

用于发布/预订的初始化命令

如果需要一系列命令来定义发布/预订环境（例如，定义预订时），那么 IBM MQ 会提供称为 CSQINPT 的第四个初始化输入数据集。

在发布/预订初始化结束时执行数据集中包含的 MQSC 命令，并将输出写入 CSQOUTT DD 语句指定的数据集。例如，您可以使用 CSQINPT 初始化数据集来定义预订。

IBM MQ for z/OS 随附了样本发布/预订初始化数据集成员。在 [随 IBM MQ](#) 提供的样本定义中进行了描述。

IBM MQ for z/OS 实用程序

IBM MQ for z/OS 提供了一组实用程序，您可以使用这些程序来帮助进行系统管理。

IBM MQ for z/OS 提供了一组实用程序来帮助您执行各种管理任务，包括：

- 管理消息安全策略。
- 执行备份，复原和重组任务。
- 发出命令和进程对象定义。
- 生成数据转换出口。
- 修改引导数据集。
- 列出有关日志的信息。
- 打印日志。
- 设置 Db2 表和其他 Db2 实用程序。

- 处理死信队列上的消息。

消息安全策略实用程序

消息安全策略实用程序 (CSQ0UTIL) 作为独立实用程序运行，以管理消息安全策略。请参阅 [消息安全策略实用程序 \(CSQ0UTIL\)](#) 以获取更多信息。

CSQUTIL 实用程序

这是一个实用程序，用于帮助您执行备份，复原和重组任务。请参阅 [CSQUTIL 实用程序](#) 以获取更多信息。

数据转换出口实用程序

IBM MQ for z/OS 数据转换出口实用程序 (CSQUCVX) 作为独立实用程序运行以创建数据转换出口例程。

更改日志库存实用程序

IBM MQ for z/OS 更改日志库存实用程序 (CSQJU003) 作为独立实用程序运行以更改引导数据集 (BSDS)。您可以使用该实用程序来执行以下功能：

- 添加或删除活动或归档日志数据集。
- 提供归档日志的密码。

打印日志映射实用程序

IBM MQ for z/OS 打印日志映射实用程序 (CSQJU004) 作为独立实用程序运行以列出以下信息：

- 所有活动日志数据集和归档日志数据集的副本的日志数据集名称和日志 RBA 关联。如果双日志记录未处于活动状态，那么只有一个数据集副本。
- 可用于新日志数据的活动日志数据集。
- 引导数据集 (BSDS) 中检查点记录队列的内容。
- 归档日志命令历史记录的内容。
- 系统和实用程序时间戳记。

日志打印实用程序

日志打印实用程序 (CSQ1LOGP) 作为独立实用程序运行。您可以在指定以下内容的情况下运行实用程序：

- 引导数据集 (BSDS)
- 活动日志 (无 BSDS)
- 归档日志 (无 BSDS)

队列共享组实用程序

队列共享组实用程序 (CSQ5PQSG) 作为独立实用程序运行，以设置 Db2 表并执行队列共享组所需的其他 Db2 任务。

活动日志预格式化实用程序

活动日志预格式化实用程序 (CSQJUFMT) 在队列管理器使用活动日志数据集之前对其进行格式化。如果活动日志数据集由实用程序预先格式化，那么在队列管理器首次通过活动日志时将提高日志写入性能。

死信队列处理程序实用程序

死信队列处理程序实用程序 (**CSQUDLQH**) 作为独立实用程序运行。它会检查死信队列上的消息, 并根据您提供给实用程序的一组规则来处理这些消息。

IBM MQ for z/OS 的 CSQUTIL 实用程序

随 IBM MQ for z/OS 提供了 CSQUTIL 实用程序, 以帮助您执行备份, 复原和重组任务以及发出命令和处理对象定义。

有关 CSQUTIL 实用程序的更多信息, 请参阅 [IBM MQ 实用程序 \(CSQUTIL\)](#)。通过使用此实用程序, 可以调用以下函数:

COMMAND

发出 MQSC 命令, 记录对象定义以及创建客户机通道定义文件。

COPY

读取指定 IBM MQ for z/OS 消息队列的内容或指定页集的所有队列的内容, 并将其放入顺序文件中并保留原始队列。

复制页面

将整个页集复制到更大的页集。

空

要删除指定 IBM MQ for z/OS 消息队列的内容或指定页集的所有队列的内容, 请保留这些队列的定义。

FORMAT

用于格式化 IBM MQ for z/OS 页集。

LOAD

从 COPY 函数创建的顺序文件复原指定 IBM MQ for z/OS 消息队列的内容或指定页集的所有队列的内容。

页信息

从一个或多个页集抽取页集信息。

RESETPAGE

将整个页集复制到其他页集数据集并重置副本中的日志信息。

SCOPY

在队列管理器脱机时将队列内容复制到数据集。

SDEFS

在队列管理器脱机时为对象生成一组定义命令。

SLOAD

从先前 COPY 或 SCOPY 操作的目标数据集复原消息。SLOAD 处理单个队列。

开关

切换或查询与集群发送方通道关联的传输队列。

XPARM

将通道启动程序参数装入模块转换为队列管理器属性 (用于迁移)。

操作 IBM MQ for z/OS

使用这些基本过程来操作 IBM MQ for z/OS。

您还可以使用随 IBM MQ for Windows, IBM MQ for Linux (x86 和 x86-64 平台) 和 SupportPac MS0T 一起分发的 IBM MQ Explorer 来执行本节中描述的操作。有关更多信息, 请参阅 [第 114 页的『使用 IBM MQ Explorer 进行管理』](#) 和 [IBM 支持与下载](#)。

本节包含有关以下主题的信息:

您可以从 z/OS 控制台或使用实用程序 CSQUTIL 发出 IBM MQ 控制命令。命令可以使用命令前缀字符串 (CPF) 来指示哪个 IBM MQ 子系统处理该命令。

您可以使用 IBM MQ 命令来控制 IBM MQ 的大部分操作环境。IBM MQ for z/OS 支持这些命令的 MQSC 和 PCF 类型。本主题描述了如何使用 MQSC 命令指定属性，因此它引用了那些使用其 MQSC 命令名 (而不是其 PCF 名称) 的命令和属性。有关 MQSC 命令语法的详细信息，请参阅 [MQSC 命令](#)。有关 PCF 命令语法的详细信息，请参阅第 19 页的『[使用 IBM MQ 可编程命令格式](#)』。如果您是适当的授权用户，那么可以从以下位置发出 IBM MQ 命令：

- 初始化输入数据集 (如第 283 页的『[IBM MQ for z/OS 的初始化命令](#)』中所述)。
- z/OS 控制台或等效控制台，例如 SDSF
- z/OS 主 get 命令例程 MGCRE (SVC 34)
- IBM MQ 实用程序 CSQUTIL (在 [IBM MQ 实用程序](#)中描述)。
- 用户应用程序，可以是：
 - CICS 程序
 - TSO 程序
 - z/OS 批处理程序
 - IMS 程序

请参阅第 304 页的『[编写程序以管理 IBM MQ for z/OS](#)』以获取有关此问题的信息。

这些命令的大部分功能由操作和控制面板以方便的方式提供，可从 TSO 和 ISPF 访问，并在第 292 页的『[IBM MQ for z/OS 的操作和控制面板](#)』中进行了描述。

有关更多信息，请参阅

- [第 287 页的『从 z/OS 控制台或其等效控制台发出命令』](#)
 - [命令前缀字符串](#)
 - [使用 z/OS 控制台发出命令](#)
 - [命令响应](#)
- [从实用程序 CSQUTIL 发出命令](#)

从 z/OS 控制台或其等效控制台发出命令

您可以从 z/OS 控制台或其等效命令发出所有 IBM MQ 命令。您还可以从可以发出 z/OS 命令的任何位置 (例如 SDSF) 或通过使用 MGCRE 宏的程序发出 IBM MQ 命令。

在控制台输入的命令可显示的最大数据量为 32 KB。

注：

1. 不能从 IMS 终端发出使用 IMS/SSR 命令格式的 IBM MQ 命令。IMS 适配器不支持此功能。
2. SDSF 提供的输入字段对于某些命令 (尤其是那些用于通道的命令) 可能不够长。

命令前缀字符串

每个 IBM MQ 命令都必须以命令前缀字符串 (CPF) 作为前缀，如第 288 页的图 41 中所示。

由于多个 IBM MQ 子系统可以在 z/OS 下运行，因此 CPF 用于指示哪个 IBM MQ 子系统处理该命令。例如，要为名为 CSQ1 的子系统启动队列管理器，其中 CPF 为 '+CSQ1'，从操作员控制台发出命令 +CSQ1 START QMGR。必须在子系统名称表中定义此 CPF (对于子系统 CSQ1)。这在 [定义命令前缀字符串 \(CPF\)](#) 中进行了描述。在示例中，字符串 "+CSQ1" 用作命令前缀。

使用 z/OS 控制台发出命令

您可以从 z/OS 控制台输入简单命令，例如第 288 页的图 41 中的 DISPLAY 命令。但是，对于复杂命令或您频繁发出的命令集，发出命令的其他方法更好。

```
+CSQ1 DISPLAY QUEUE(TRANSMIT.QUEUE.PROD) TYPE(QLLOCAL)
```

图 41: 从 z/OS 控制台发出 DISPLAY 命令

命令响应

对命令的直接响应将发送到发出该命令的控制台。IBM MQ 支持 z/OS 中提供的扩展控制台支持 (EMCS) 功能，因此可以使用具有 4 字节标识的控制台。此外，当使用 MGCRC 宏的程序发出命令时，除 START QMGR 和 STOP QMGR 以外的所有命令都支持使用命令和响应令牌 (CART)。

从实用程序 CSQUTIL 发出命令

您可以使用实用程序 CSQUTIL 的 COMMAND 函数从连续数据集发出命令。此实用程序将命令作为消息传输到 *system-command input queue*，并等待响应 (与 SYSPRINT 中的原始命令一起打印)。有关详细信息，请参阅 [IBM MQ 实用程序](#)。

在 z/OS 上启动和停止队列管理器

使用本主题作为停止和启动队列管理器的简介。

本节描述如何启动和停止队列管理器。它包含有关以下主题的信息：

- [第 288 页的『开始之前 IBM MQ』](#)
- [第 288 页的『启动队列管理器』](#)
- [第 290 页的『停止队列管理器』](#)

启动和停止队列管理器相对简单。当队列管理器在正常情况下停止时，其最后一项操作是执行终止检查点。此检查点和日志为队列管理器提供重新启动所需的信息。

本部分包含有关 START 和 STOP 命令的信息，并包含发生异常终止后启动的简要概述。

开始之前 IBM MQ

安装 IBM MQ 后，它将定义为正式的 z/OS 子系统。此消息在 z/OS 的任何初始程序装入 (IPL) 期间出现：

```
CSQ3110I +CSQ1 CSQ3UR00 - SUBSYSTEM ssnm INITIALIZATION COMPLETE
```

其中 *ssnm* 是 IBM MQ 子系统名称。

从现在开始，您可以从已授权发出系统控制命令的任何 z/OS 控制台启动该子系统的队列管理器；即，z/OS SYS 命令组。必须从授权控制台发出 START 命令，不能通过 JES 或 TSO 发出。

如果您正在使用队列共享组，那么必须先启动 RRS，然后启动 Db2，然后再启动队列管理器。

启动队列管理器

通过发出 `START QMGR` 命令来启动队列管理器。但是，除非您具有相应的权限，否则无法成功使用 `START` 命令。请参阅在 `z/OS` 上设置安全性，以获取有关 `IBM MQ` 安全性的信息。第 289 页的图 42 显示了 `START` 命令的示例。(请记住，必须使用命令前缀字符串 (CPF) 作为 `IBM MQ` 命令的前缀。)

```
+CSQ1 START QMGR
+CSQ1 START QMGR PARM(NEWLOG)
```

图 42: 从 `z/OS` 控制台启动队列管理器

有关 `START QMGR` 命令的语法信息，请参阅 [START QMGR](#)。

不能将队列管理器作为批处理作业运行，也不能使用 `z/OS` 命令 `START` 将其启动。这些方法可能会启动 `IBM MQ` 的地址空间，然后异常结束。也不能从 `CSQUTIL` 实用程序或类似的用户应用程序启动队列管理器。

但是，您可以通过将 `START QMGR` 命令传递到 `z/OS` `MGCRC` (SVC 34) 服务，从 `APF` 授权的程序启动队列管理器。

如果您正在使用队列共享组，那么在启动队列管理器时，关联的 `Db2` 系统和 `RRS` 必须处于活动状态。

启动选项

启动队列管理器时，将装入系统参数模块。可以通过以下两种方法之一指定系统参数模块的名称：

- 使用 `/cpf START QMGR` 命令的 `PARM` 参数，例如

```
/cpf START QMGR PARM(CSQ1ZPRM)
```

- 使用启动过程中的参数，例如，将 `JCL EXEC` 语句编码为

```
//MQM EXEC PGM=CSQYASCP,PARM='ZPARAM(CSQ1ZPRM)'
```

系统参数模块提供定制队列管理器时指定的信息。

V 9.0.3 从 `IBM MQ 9.0.3` 开始，可以使用 `QMGRPROD` 选项来指定要记录队列管理器使用情况的的产品，并使用 `AMSPROD` 选项来指定 `AMS` 的等效项 (如果已使用)。有关允许的值的详细信息，请参阅 `MQSC START QMGR` 命令。

V 9.0.3 以下是示例 `JCL EXEC` 语句：

```
//MQM EXEC PGM=CSQYASCP,PARM='QMGRPROD(MQ)'
```

请参阅 [z/OS MVS 产品管理](#) 以获取有关产品用法记录的更多信息。

您还可以使用 `ENVPARM` 选项来替换队列管理器的 `JCL` 过程中的一个或多个参数。

例如，您可以更新队列管理器启动过程，以便 `DDname CSQINP2` 是变量。这意味着您可以在不更改启动过程的情况下更改 `CSQINP2 DDname`。这对于实现更改，为操作程序提供回退以及队列管理器操作很有用。

假设队列管理器 `CSQ1` 的启动过程类似于第 290 页的图 43。

```

//CSQ1MSTR PROC INP2=NORM
//MQMESA EXEC PGM=CSQYASCP
//STEPLIB DD DISP=SHR,DSN=thlqual.SCSQANLE
//          DD DISP=SHR,DSN=thlqual.SCSQAUTH
//          DD DISP=SHR,DSN=db2qual.SDSNLOAD
//BSDS1 DD DISP=SHR,DSN=myqual.BSDS01
//BSDS2 DD DISP=SHR,DSN=myqual.BSDS02
//CSQP0000 DD DISP=SHR,DSN=myqual.PSID00
//CSQP0001 DD DISP=SHR,DSN=myqual.PSID01
//CSQP0002 DD DISP=SHR,DSN=myqual.PSID02
//CSQP0003 DD DISP=SHR,DSN=myqual.PSID03
//CSQINP1 DD DISP=SHR,DSN=myqual.CSQINP(CSQ1INP1)
//CSQINP2 DD DISP=SHR,DSN=myqual.CSQINP(CSQ1&INP2.)
//CSQOUT1 DD SYSOUT=*
//CSQOUT2 DD SYSOUT=*

```

图 43: 样本启动过程

如果然后使用以下命令启动队列管理器:

```
+CSQ1 START QMGR
```

使用的 CSQINP2 是名为 CSQ1NORM 的成员。

但是, 假定您正在将一组新的程序投入生产, 以便下次启动队列管理器 CSQ1 时, 将从成员 CSQ1NEW 获取 CSQINP2 定义。为此, 您将使用以下命令启动队列管理器:

```
+CSQ1 START QMGR ENVPARM('INP2=NEW')
```

并且将使用 CSQ1NEW 代替 CSQ1NORM。注: z/OS 将符号参数的 KEYWORD=value 规范 (如 INP2=NEW 中所示) 限制为 255 个字符。

在异常终止后启动

IBM MQ 会自动检测重新启动是遵循正常关闭还是异常终止。

异常结束后启动队列管理器与发出 STOP QMGR 命令后启动队列管理器不同。STOP QMGR 之后, 系统会有序地完成其工作, 并在停止之前采用终止检查点。重新启动队列管理器时, 它将使用系统检查点和恢复日志中的信息来确定关闭时的系统状态。

但是, 如果队列管理器异常结束, 那么它将在无法完成其工作或采用终止检查点的情况下终止。在异常结束后重新启动队列管理器时, 它会使用日志中的信息刷新其在终止时的状态知识, 并通知您各种任务的状态。通常, 重新启动过程会解决所有不一致的状态。但是, 在某些情况下, 您必须采取具体步骤来解决不一致问题。

启动时的用户消息

成功启动队列管理器时, 队列管理器将生成一组启动消息。

停止队列管理器

在停止队列管理器之前, 所有与 IBM MQ 相关的 "写至操作员的应答" (WTOR) 消息都必须接收应答, 例如, 获取日志请求。第 291 页的图 44 中的每个命令都会终止一个正在运行的队列管理器。

```
+CSQ1 STOP QMGR
+CSQ1 STOP QMGR MODE(QUIESCE)
+CSQ1 STOP QMGR MODE(FORCE)

+CSQ1 STOP QMGR MODE(RESTART)
```

图 44: 停止队列管理器

命令 STOP QMGR 缺省为 STOP QMGR MODE (QUIESCE)。

在 QUIESCE 方式下，IBM MQ 不允许创建任何新的连接线程，但允许现有线程继续；仅当所有线程都已结束时，它才会终止。应用程序可以请求在队列管理器停顿时接收通知。因此，请尽可能使用 QUIESCE 方式，以便已请求通知的应用程序有机会断开连接。请参阅 [终止期间发生的情况](#) 以获取详细信息。

如果队列管理器未在响应 STOP QMGR MODE (QUIESCE) 命令的合理时间内终止，请使用 DISPLAY CONN 命令来确定是否存在任何连接线程，并执行必要的步骤来终止关联的应用程序。如果没有线程，请发出 STOP QMGR MODE (FORCE) 命令。

STOP QMGR MODE (QUIESCE) 和 STOP QMGR MODE (FORCE) 命令从 MVS 自动重新启动管理器 (ARM) 注销 IBM MQ，从而阻止 ARM 自动重新启动队列管理器。STOP QMGR MODE (RESTART) 命令的工作方式与 STOP QMGR MODE (FORCE) 命令的工作方式相同，但它不会从 ARM 注销 IBM MQ。这意味着队列管理器有资格立即自动重新启动。

如果 IBM MQ 子系统未向 ARM 注册，那么将拒绝 STOP QMGR MODE (RESTART) 命令，并将以下消息发送到 z/OS 控制台：

```
CSQY205I ARM element arm-element is not registered
```

如果未发出此消息，那么将自动重新启动队列管理器。有关 ARM 的更多信息，请参阅 [第 354 页的『使用 z/OS Automatic Restart Manager \(ARM\)』](#)。

仅当 STOP QMGR MODE (FORCE) 未终止队列管理器时，才取消队列管理器地址空间。

如果通过取消地址空间或使用命令 STOP QMGR MODE (FORCE) 来停止队列管理器，那么将与已连接的 CICS 或 IMS 系统保持一致性。资源的再同步在队列管理器重新启动时启动，并在与 CICS 或 IMS 系统建立连接时完成。

注：停止队列管理器时，可能会发现发出消息 IEF352I。如果 z/OS 检测到未能将地址空间标记为不可用将导致完整性暴露，那么会发出此消息。您可以忽略此消息。

停止消息

发出 STOP QMGR 命令后，将获取消息 CSQY009I 和 CSQY002I，例如：

```
CSQY009I +CSQ1 ' STOP QMGR' COMMAND ACCEPTED FROM
USER(userid), STOP MODE(FORCE)
CSQY002I +CSQ1 QUEUE MANAGER STOPPING
```

其中 `userid` 是发出 STOP QMGR 命令的用户标识，而 MODE 参数取决于该命令中指定的用户标识。

当 STOP 命令成功完成时，将在 z/OS 控制台上显示以下消息：

```
CSQ9022I +CSQ1 CSQYASCP ' STOP QMGR' NORMAL COMPLETION
CSQ3104I +CSQ1 CSQ3EC0X - TERMINATION COMPLETE
```

如果您正在使用 ARM，但未指定 MODE (RESTART)，那么还会显示以下消息：

```
CSQY204I +CSQ1 ARM DEREGISTER for element arm-element type  
arm-element-type successful
```

在显示以下消息之前，无法重新启动队列管理器：

```
CSQ3100I +CSQ1 CSQ3EC0X - SUBSYSTEM ssnm READY FOR START COMMAND
```

IBM MQ for z/OS 的操作和控制面板

您可以使用 IBM MQ 操作和控制面板对 IBM MQ 对象执行管理任务。使用本主题作为命令和控制面板的简介。

您可以使用这些面板来定义，显示，变更或删除 IBM MQ 对象。使用这些面板进行日常管理，并对对象进行少量更改。如果要设置或更改许多对象，请使用 CSQUTIL 实用程序的 COMMAND 函数。

操作和控制面板支持对通道启动程序 (例如，启动通道或 TCP/IP 侦听器)，集群和安全性的控制。它们还使您能够显示有关线程和页集使用情况的信息。

这些面板通过通过系统命令输入队列将 MQSC 类型 IBM MQ 命令发送到队列管理器来工作。

注：

1. z/OS IBM MQ 操作和控制面板 (CSQOREXX) 可能不支持从 V 7 开始添加的所有新功能和参数。例如，没有用于直接处理主题对象或预订的面板。

通过使用下列其中一种受支持的机制，您可以管理发布/预订定义以及其他面板中未直接提供的其他系统控件：

- a. IBM MQ 浏览者
- b. z/OS 控制台
- c. 可编程命令格式 (PCF) 消息
- d. CSQUTIL 的 COMMAND 函数

请注意，CSQOREXX 面板中的通用 **Command** 操作允许您发出任何有效的 MQSC 命令，包括与 SMDS 相关的命令。您可以使用 CSQUTIL 的 COMMAND 函数发出的所有命令。

2. 不能直接从面板中的命令行发出 IBM MQ 命令。
3. 要使用操作和控制面板，您必须具有正确的安全权限；这在 [命令安全性和命令资源安全性的用户标识](#) 中进行了描述。
4. 不能使用 CSQUTIL 或 CSQOREXX 面板提供用户标识和密码。相反，如果您的用户标识对 MQCONN 中的 BATCH 概要文件具有 UPDATE 权限，那么可以绕过 **CHKLOCL** (必需设置。请参阅 [在本地绑定的应用程序上使用 CHCKLOCL](#) 以获取更多信息。

操作和控制面板的调用和规则

您可以通过 ISPF 面板来控制 IBM MQ 和发出控制命令。

如何访问 IBM MQ 操作和控制面板

如果已针对 IBM MQ 更新 ISPF/PDF 主选项菜单，那么可以从该菜单访问 IBM MQ 操作和控制面板。有关更新菜单的详细信息，请参阅 [任务 20: 设置操作和控制面板](#)。

您可以从 TSO 命令处理器面板 (通常是 ISPF/PDF 主选项菜单上的选项 6) 访问 IBM MQ 操作和控制面板。为执行此操作而运行的 exec 的名称为 CSQOREXX。它有两个参数: *thlqual* 是要使用的 IBM MQ 库的高级限定符，*langletter* 是标识要使用的本地语言库 (例如，E for U.S) 的字母。英文)。如果 IBM MQ 库永久安装在 ISPF 设置中，那么可以省略这些参数。或者，可以从 TSO 命令行发出 CSQOREXX。

这些面板旨在供操作人员和管理员进行最低限度的正式培训使用。在运行面板的情况下阅读这些指示信息，并尝试建议的不同任务。

注: 使用面板时，具有 SYSTEM.CSQOREXX.* 创建。

操作和控制面板的规则

有关 IBM MQ 字符串和名称的一般规则，请参阅 [用于命名 IBM MQ 对象的规则](#)。但是，有一些规则仅适用于操作和控制面板：

- 请勿将字符串 (例如描述) 括在单引号或双引号中。
- 如果在文本字段中包含撇号或引号，那么不必重复或添加转义字符。这些字符与您输入的字符完全相同；例如：

```
This is Maria's queue
```

面板处理器会将它们加倍，以便您将它们传递到 IBM MQ。但是，如果必须截断数据才能执行此操作，那么将执行此操作。

- 您可以在大多数字段中使用大写或小写字符，当您按 Enter 键时，这些字符将转换为大写字符。例外情况如下所示：
 - 存储类名和耦合设施结构名，必须以大写 A 到 Z 开头，后跟大写 A 到 Z 或数字字符。
 - 某些未翻译的字段。其中包括：
 - 应用程序标识
 - 描述
 - 环境数据
 - 对象名 (但如果使用小写对象名，那么可能无法在 z/OS 控制台中输入该对象名)
 - 远程系统名称
 - 触发器数据
 - 用户数据
- 在名称中，将忽略前导空格和前导下划线。因此，不能有以空格或下划线开头的对象名。
- 下划线用于显示空白字段的范围。按 Enter 键时，尾部下划线将替换为空格。
- 许多描述和文本字段呈现在多个部分中，每个部分由 IBM MQ 独立处理。这意味着将保留尾部空格，并且文本不连续。

空白字段

为 IBM MQ 对象指定 **定义** 操作时，定义面板上的每个字段都包含一个值。请参阅显示面板的常规帮助 (扩展帮助)，以获取有关 IBM MQ 获取值的位置的信息。如果您在带有空格的字段上输入，并且不允许使用空格，那么 IBM MQ 会将安装缺省值放在该字段中，或者提示您输入必需值。

为 IBM MQ 对象指定 **变更** 操作时，变更面板上的每个字段都包含该字段的当前值。如果在带有空格的字段上输入，并且不允许使用空格，那么该字段的值将保持不变。

z/OS 上的对象和操作

操作和控制面板为您提供许多不同类型的对象以及可以对其执行的许多操作。

这些操作在初始面板上列出，使您能够处理对象并显示有关对象的信息。这些对象包括所有 IBM MQ 对象以及一些额外的对象。这些对象属于以下类别。

- [队列，进程，认证信息对象，名称列表，存储类和 CF 结构](#)

- [通道](#)
- [集群对象](#)
- [队列管理器和安全性](#)
- [连接](#)
- [系统](#)

请参阅 [操作](#)，以获取可对 IBM MQ 对象执行的操作的交叉引用表。

队列，进程，认证信息对象，名称列表，存储类和 CF 结构

这些是基本 IBM MQ 对象。每种类型可能有许多个。可以使用 LIST 或 DISPLAY，LIST with FILTER，DEFINE LIKE，MANAGE 和 ALTER 操作列出这些属性，并将其与过滤器一起列出，定义和删除，以及具有可显示和更改的属性。（使用 MANAGE 操作删除对象。）

此类别由以下对象组成：

| | |
|----------|--------------------------------------|
| QLOCAL | 本地队列 |
| QREMOTE | 远程队列 |
| QALIAS | 用于间接引用队列的别名队列 |
| QMODEL | 用于动态地定义队列的模型队列 |
| 队列 | 任何类型的队列 |
| 队列状态 | 本地队列的状态 |
| PROCESS | 有关触发器事件发生时将启动的应用程序的信息 |
| AUTHINFO | 认证信息：使用 LDAP 服务器执行证书撤销列表(CRL)检查所需的定义 |
| NAMELIST | 名称列表，如队列或集群 |
| STGCLASS | 存储类 |
| CFSTRUCT | 耦合设施(CF)结构 |
| CFSTATUS | CF 结构的状态 |

通道

通道用于分布式排队。每种类型可以有許多，并且可以列出这些类型，并随过滤器一起列出，定义，删除，显示和变更。它们还具有使用 START，STOP 和 PERFORM 操作的其他可用功能。PERFORM 提供复位，ping 和解析通道功能。

此类别由以下对象组成：

| | |
|-----------|---------|
| 通道 | 任何类型的通道 |
| SENDER | 发送通道 |
| 服务器 | 服务器通道 |
| RECEIVER | 接收方通道 |
| REQUESTER | 请求者通道 |
| CLUSRCVR | 集群接收方通道 |
| CLUSDR | 集群发送方通道 |
| SVRCONN | 服务器连接通道 |
| CLNTCONN | 客户机连接通道 |
| CHSTATUS | 通道连接的状态 |

集群对象

将自动为属于集群的队列和通道创建集群对象。基本队列和通道定义可以在另一个队列管理器上。每种类型可以有多个，而且名称可以重复。它们可以列出，与过滤器一起列出并显示。PERFORM，START 和 STOP 也可通过 LIST 操作获得。

此类别由以下对象组成:

| | |
|----------|------------------------------|
| CLUSQ | 集群队列，为属于集群的队列创建 |
| CLUSCHL | 集群通道，为属于集群的通道创建 |
| CLUSQMGR | 集群队列管理器，与集群通道相同，但由其队列管理器名称标识 |

集群通道和集群队列管理器具有 PERFORM，START 和 STOP 操作，但仅通过 DISPLAY 操作间接执行。

队列管理器和安全性

队列管理器和安全对象具有单个实例。它们可以列出，并且具有可显示和改变的属性 (使用 LIST 或 DISPLAY 以及 ALTER 操作)，并且具有其他可使用 PERFORM 操作的功能。

此类别由以下对象组成:

| | |
|-----|------------------------------|
| 经理 | 队列管理器 :PERFORM 操作提供暂挂和恢复集群功能 |
| 安全性 | 安全性功能 :PERFORM 操作提供刷新和重新验证功能 |

Connection

可以列出连接，并随过滤器一起列出和显示连接。

此类别仅由连接对象 CONNECT 组成。

系统

其它功能的集合。此类别由以下对象组成:

| | |
|--------|-------------|
| SYSTEM | 系统功能 |
| 控制 | SYSTEM 的同义词 |

可用的功能是:

| | |
|----------------|-----------------------------|
| LIST 或 DISPLAY | 显示队列共享组，分布式排队，页集或数据集使用情况信息。 |
| 执行 | 刷新或重置集群 |
| START | 启动通道启动程序或侦听器 |
| 结束 | 停止通道启动程序或侦听器 |

操作

下表显示了您可以对每种类型的对象执行的操作:

| Object | 更改 | 定义类似 | 管理 (1) | 列表或显示 | 具有过滤器的列表 | 执行 | 启动 | 停止 |
|----------|----|------|--------|-------|----------|----|----|----|
| AUTHINFO | X | X | X | X | X | | | |
| CFSTATUS | | | | X | | | | |
| CFSTRUCT | X | X | X | X | X | | | |
| 通道 | X | X | X | X | X | X | X | X |

表 25: IBM MQ 对象的有效操作和控制面板操作 (继续)

| Object | 更改 | 定义类似 | 管理 (1) | 列表或显示 | 具有过滤器的列表 | 执行 | 启动 | 停止 |
|-----------|----|------|--------|-------|----------|-------|-------|-------|
| CHSTATUS | | | | X | X | | | |
| CLNTCONN | X | X | X | X | X | | | |
| CLUSCHL | | | | X | X | X (2) | X (2) | X (2) |
| CLUSQ | | | | X | X | | | |
| CLUSQMGR | | | | X | X | X (2) | X (2) | X (2) |
| CLUSRCVR | X | X | X | X | X | X | X | X |
| CLUSDR | X | X | X | X | X | X | X | X |
| 连接 | | | | X | X | | | |
| 控制 | | | | X | | X | X | X |
| 经理 | X | | | X | | X | | |
| NAMELIST | X | X | X | X | X | | | |
| PROCESS | X | X | X | X | X | | | |
| QALIAS | X | X | X | X | X | | | |
| QLOCAL | X | X | X | X | X | | | |
| QMODEL | X | X | X | X | X | | | |
| QREMOTE | X | X | X | X | X | | | |
| 队列状态 | | | | X | X | | | |
| 队列 | X | X | X | X | X | | | |
| RECEIVER | X | X | X | X | X | X | X | X |
| REQUESTER | X | X | X | X | X | X | X | X |
| 安全性 | X | | | X | | X | | |
| SENDER | X | X | X | X | X | X | X | X |
| 服务器 | X | X | X | X | X | X | X | X |
| SVRCONN | X | X | X | X | X | | X | X |
| STGCLASS | X | X | X | X | X | | | |
| SYSTEM | | | | X | | X | X | X |

注:

1. 提供 Delete 和其他功能
2. 使用 列表或显示 操作

z/OS 上的对象处置

您可以指定需要使用的对象的 处置。此处置表示保留对象 **definition** 的位置以及该对象的行为方式。

仅当您使用以下任何对象类型时，处置才有意义:

- 队列
- 通道

- 进程
- 名称列表
- 存储类
- 认证信息对象

如果您正在使用其他对象类型，那么将忽略此处置。

允许的值包括：

Q

QMGR。对象定义位于队列管理器的页集上，并且只能由队列管理器访问。

C

COPY。对象定义位于队列管理器的页集上，并且只能由队列管理器访问。它们是定义为具有 GROUP 处置的对象的本地副本。

P

PRIVATE。对象定义位于队列管理器的页集上，并且只能由队列管理器访问。对象已定义为具有 QMGR 或 COPY 处置。

G

组。对象定义位于共享存储库中，并且可供队列共享组中的所有队列管理器访问。

S

共享。此处置仅适用于本地队列。队列定义位于共享存储库中，并且可供队列共享组中的所有队列管理器访问。

A

全部。如果操作队列管理器是目标队列管理器或 *，那么将包括 **所有** 处置的对象；否则，仅包括 QMGR 和 COPY 处置的对象。这是缺省值。

在 z/OS 上使用 ISPF 控制面板选择队列管理器，缺省值和级别

您可以在 ISPF 中使用 CSQOREXX exec 来控制队列管理器。

在查看初始面板时，您未连接到任何队列管理器。但是，只要按 Enter 键，您就会连接到队列管理器或在 **连接名称** 字段中指定的队列共享组中的队列管理器。可以将此字段留空；这意味着您正在将缺省队列管理器用于批处理应用程序。这是在 CSQBDEFV 中定义的（请参阅 [任务 19: 设置批处理, TSO 和 RRS 适配器](#) 以获取有关此问题的信息）。

使用 **目标队列管理器** 字段来指定要在其中执行您请求的操作的队列管理器。如果将此字段留空，那么缺省为 **连接名称** 字段中指定的队列管理器。您可以指定不是您所连接的目标队列管理器。在这种情况下，通常会指定提供队列管理器别名定义的远程队列管理器对象的名称（打开命令输入队列时，该名称将用作 *ObjectQMgrName*）。要执行此操作，必须设置适当的队列和通道以访问远程队列管理器。

操作队列管理器 字段允许您将与 **目标队列管理器** 字段中指定的队列管理器位于同一队列共享组中的队列管理器指定为要在其中执行您请求的操作的队列管理器。如果在此字段中指定 *，那么您请求的操作将在队列共享组中的所有队列管理器上执行。如果将此字段留空，那么它缺省为 **目标队列管理器** 字段中指定的值。

操作队列管理器 字段对应于使用 [MQSC 命令](#) 中描述的 CMDSCOPE 命令修饰符。

队列管理器缺省值

如果将任何队列管理器字段留空，或者选择连接到队列共享组，那么当您按 **Enter** 键时，将打开辅助窗口。此窗口确认您将使用的队列管理器的名称。按 **Enter** 键继续。当您在发出一些请求后返回到初始面板时，您会发现使用实际名称填写的字段。

队列管理器级别

仅当队列管理器在 z/OS 上的 IBM WebSphere MQ 710 或更高版本上运行时，“操作”和“控制”面板才能令人满意地工作。

如果不满足这些条件，那么操作可能仅部分起作用，不正确或根本不起作用，并且无法识别来自队列管理器的应答。

如果操作队列管理器未处于或高于 IBM MQ 8.0.0，那么不会显示某些字段，并且无法输入某些值。不允许使用一些对象和操作。在这种情况下，将打开一个辅助窗口，要求您确认是否要继续。

z/OS 将功能键和命令行与 z/OS 上的 ISPF 控制面板配合使用

要使用面板，必须使用功能键或在 ISPF 控制面板命令区域中输入等效命令。

- [功能键](#)
 - [处理操作](#)
 - [第 298 页的『显示 IBM MQ 用户消息』](#)
 - [取消操作](#)
 - [获取帮助](#)
- [使用命令行](#)

功能键

功能键具有 IBM MQ 的特殊设置。(这意味着不能将 ISPF 缺省值用于功能键;如果先前在任何位置使用了 KEYLIST OFF ISPF 命令,那么必须在任何操作和控制面板的命令区域中输入 KEYLIST ON,然后按 Enter 键以启用 IBM MQ 设置。)

这些功能键设置可以显示在面板上,如第 299 页的图 45 中所示。如果未显示设置,请在任何操作和控制面板的命令区域中输入 PFSHOW,然后按 **Enter** 键。要除去设置的显示,请使用命令 PFSHOW OFF。

操作和控制面板中的功能键设置符合 CUA 标准。虽然您可以通过常规 ISPF 过程(例如 **KEYLIST** 实用程序)来更改密钥设置,但建议您不要这样做。

注: 使用 **PFSHOW** 和 **KEYLIST** 命令会影响您拥有的任何其他逻辑 ISPF 屏幕,当您离开操作和控制面板时,它们的设置将保持不变。

处理操作

按 **Enter** 键以执行面板上请求的操作。面板中的信息将发送到队列管理器以进行处理。

每次在面板中按 **Enter** 键时,IBM MQ 都会生成一条或多条操作员消息。如果操作成功,那么会收到确认消息 CSQ9022I,否则会收到一些错误消息。

显示 IBM MQ 用户消息

在任何面板中按功能键 F10 以查看 IBM MQ 用户消息。

取消操作

在初始面板上,F3 和 F12 都退出操作和控制面板,并返回到 ISPF。未向队列管理器发送任何信息。

在任何其他面板上,按功能键 F3 或 F12 以使当前面板 **忽略自上次按 Enter 键以来输入的任何数据**。同样,不会向队列管理器发送任何信息。

- F3 将直接返回到初始面板。
- F12 将您带回上一个面板。

获得帮助

每个面板都有与其关联的帮助面板。帮助面板使用 ISPF 协议:

- 在任何面板上按功能键 F1 以查看有关该任务的常规帮助(扩展帮助)。
- 使用光标在任何字段上按功能键 F1 以查看有关该字段的特定帮助。
- 从任何字段帮助面板按功能键 F5 以获取常规帮助。
- 按功能键 F3 以返回到基本面板,即,从中按下功能键 F1 的面板。
- 从任何帮助面板按功能键 F6 以获取有关功能键的帮助。

如果帮助信息包含在第二个或后续页面中,那么将在面板的右上方显示 **更多** 指示符。使用以下功能键浏览帮助页面:

- F11 以进入下一个帮助页面(如果有)。

- F10 以返回到先前的帮助页面 (如果有)。

使用命令行

您永远不需要使用命令行来发出操作和控制面板所使用的命令，因为它们可从功能键获取。提供了命令行以允许您输入正常的 ISPF 命令 (例如 **PFSHOW**)。

ISPF 命令 **PANELID ON** 显示当前 CSQOREXX 面板的名称。

无论您具有哪些 ISPF 设置，命令行最初都显示在面板底部的缺省位置中。可以从任何操作和控制面板使用 **SETTINGS ISPF** 命令来更改命令行的位置。对于操作和控制面板的后续会话，将记住这些设置。

在 z/OS 上使用操作和控制面板

使用本主题来调查从 CSQOREXX 显示的初始控制面板

第 299 页的图 45 显示了启动面板会话时显示的面板。

```

IBM MQ for z/OS - Main Menu
Complete fields. Then press Enter.
Action . . . . . 1      0. List with filter  4. Manage
                        1. List or Display  5. Perform
                        2. Define like     6. Start
                        3. Alter           7. Stop
                        8. Command
Object type . . . . . CHANNEL
Name . . . . . *
Disposition . . . . . A  Q=Qmgr, C=Copy, P=Private, G=Group,
                        S=Shared, A=All
Connect name . . . . . MQ1C - local queue manager or group
Target queue manager . . . . . MQ1C
                        - connected or remote queue manager for command input
Action queue manager . . . . . MQ1C - command scope in group
Response wait time . . . . . 30 5 - 999 seconds
(C) Copyright IBM Corporation 1993, 2023. All rights reserved.
Command ==>
F1=Help      F2=Split      F3=Exit      F4=Prompt    F9=SwapNext  F10=Messages
F12=Cancel

```

图 45: IBM MQ 操作和控制初始面板

在此面板中，可以执行如下操作：

- 选择所需的本地队列管理器，以及是要在该队列管理器上，远程队列管理器上还是在与本地队列管理器相同的队列共享组中的另一个队列管理器上发出命令。如果需要更改队列管理器名称，请过度输入队列管理器名称。
- 通过在 **操作** 字段中输入相应的数字，选择要执行的操作。
- 指定要使用的对象类型。如果您不确定对象类型是什么，请按功能键 **F1** 以获取有关这些对象类型的帮助。
- 指定要使用的对象类型的处置。
- 显示指定类型的对象列表。在 **名称** 字段中输入星号 (*)，然后按 **Enter** 键以显示已在操作队列管理器上定义的对象 (指定类型) 的列表。然后，可以按顺序选择一个或多个要使用的对象。所有操作都可从列表获取。

注：建议您选择导致显示对象列表的选项，然后从该列表中工作。使用 **显示** 操作，因为所有对象类型都允许该操作。

在 z/OS 上使用命令工具

使用编辑器输入或修改要传递到队列管理器的 MQSC 命令。

从主面板 CSQOPRIA 中，选择选项 **8 命令** 以启动命令工具。

将显示顺序文件 *prefix.CSQUTIL.COMMANDS*，用作 CSQUTIL COMMAND 函数的输入；请参阅 [向 IBM MQ 发出命令](#)。

您不需要使用命令前缀字符串 (CPF) 作为命令的前缀。

通过使用连续字符 + 或 -终止当前行，可以在后续行上继续 MQSC 命令。或者，使用行编辑方式来提供长 MQSC 命令或命令中长属性值的值。

行编辑

要使用行编辑，请将光标移至编辑面板中的相应行，并使用 **F4** 在可滚动面板中显示单行。单行最多可包含 32 760 个字节的数据。

要保留行编辑：

- **F3 出口** 保存对行和出口所作的更改
- **F12 cancel** 返回到编辑面板，废弃对行所作的更改。

要废弃在编辑会话中所作的更改，请使用 **F12 cancel** 来终止编辑会话，使文件的内容保持不变。不执行命令。

执行命令

完成输入 MQSC 命令后，使用 **F3 exit** 终止编辑会话以保存文件内容，并调用 CSQUTIL 以将命令传递到队列管理器。命令处理的输出保存在文件 *prefix.CSQUTIL.OUTPUT*。编辑会话将在此文件上自动打开，以便您可以查看响应。按 **F3 退出** 以退出此会话并返回到主菜单。

在 z/OS 上使用 IBM MQ 对象

本文中描述的许多任务都涉及处理 IBM MQ 对象。对象类型为队列管理器，队列，进程定义，名称列表，通道，客户机连接通道，侦听器，服务和认证信息对象。

- [定义简单队列对象](#)
- [定义其他类型的对象](#)
- [使用对象定义](#)
- [使用名称列表](#)

定义简单队列对象

要定义新对象，请使用现有定义作为其基础。您可以通过以下三种方法之一来执行此操作：

- 通过选择作为在初始面板上选择的选项的结果而显示的列表的对象的对象。然后输入操作类型 2 (**定义类似**) 在所选对象旁边的操作字段中。新对象具有所选对象的属性 (处置除外)。然后，您可以根据需要更改新对象中的任何属性。
- 在初始面板上，选择 **定义相似** 操作类型，在 **对象类型** 字段中输入要定义的对象类型，然后在 **名称** 字段中输入特定现有对象的名称。新对象的属性与您在 **名称** 字段中指定的对象相同 (处置除外)。然后，您可以根据需要更改新对象定义中的任何属性。
- 通过选择 **定义相似** 操作类型，指定对象类型，然后将 **名称** 字段留空。然后，您可以定义新对象，并且该对象具有为安装定义的缺省属性。然后，您可以根据需要更改新对象定义中的任何属性。

注：您不会在初始面板上输入要定义的对象名称，而是在 **定义** 面板上输入要显示的对象名称。

以下示例演示如何使用现有队列作为模板来定义本地队列。

定义本地队列

要从操作和控制面板定义本地队列对象，请使用现有队列定义作为新定义的基础。有几个面板需要完成。当您已完成所有面板并且对属性正确感到满意时，按 **Enter** 键将定义发送到队列管理器，然后该队列管理器将创建实际队列。

在初始面板上使用 **定义类似项** 操作，或者对作为在初始面板上选择的选项的结果而显示的列表中的对象条目使用此操作。

例如，从初始面板开始，填写以下字段：

| | |
|------|-----------|
| 操作 | 2 (定义相似项) |
| 对象类型 | QLOCAL |

名称 QUEUE.YOU.LIKE. 这是为新队列提供属性的队列的名称。

按 Enter 键以显示 " **定义本地队列** " 面板。队列名称字段为空白, 以便您可以提供新队列的名称。描述是基于此新定义的队列的描述。请输入此字段以及您自己对新队列的描述。

其他字段中的值是作为此新队列基础的队列的值 (处置除外)。您可以根据需要输入这些字段。例如, 如果适当授权的应用程序可以将消息放入此队列中, 请在 **已启用放置** 字段中输入 Y (如果它还不是 Y)。

您可以通过将光标移动到字段中并按功能键 F1 来获取字段帮助。字段帮助提供有关可用于每个属性的值的信息。

完成第一个面板后, 按功能键 F8 以显示第二个面板。

提示:

1. 请勿在此阶段按 Enter 键, 否则将在您有机会完成其余字段之前创建队列。(如果您过早地按 Enter 键, 请不要担心; 以后始终可以更改定义。)
2. 请勿按功能键 F3 或 F12, 否则您输入的数据将丢失。

重复按功能键 F8 以查看并完成其余面板, 包括触发器定义, 事件控制和回退报告面板。

本地队列定义完成时

定义完成后, 按 Enter 键以将信息发送到队列管理器进行处理。队列管理器根据您提供的定义创建队列。如果您不希望创建队列, 请按功能键 F3 以退出并取消定义。

定义其他类型的对象

要定义其他类型的对象, 请使用现有定义作为新定义的基础, 如 [定义本地队列](#) 中所述。

在初始面板上使用 **定义类似项** 操作, 或者对作为在初始面板上选择的选项的结果而显示的列表中的对象条目使用此操作。

例如, 从初始面板开始, 填写以下字段:

| | |
|-------------|---------------------------------------------|
| 操作 | 2 (定义相似项) |
| 对象类型 | QALIAS, NAMELIST, PROCESS, CHANNEL 和其他资源对象。 |
| 名称 | 留空或输入相同类型的现有对象的名称。 |

按 Enter 键以显示相应的 DEFINE 面板。根据需要填写字段, 然后再次按 Enter 键以将信息发送到队列管理器。

与定义本地队列一样, 定义另一种类型的对象通常需要完成多个面板。定义名称列表需要一些其他工作, 如第 302 页的『[使用名称列表](#)』中所述。

使用对象定义

定义对象后, 可以在 **操作** 字段中指定操作, 以变更, 显示或管理该对象。

在每种情况下, 您都可以:

- 从显示为在初始面板上选择的选项的结果的列表中选择要处理的对象。例如, 在 **操作** 字段中输入 1 以显示对象, 在 **对象类型** 字段中输入 Queue, 在 **名称** 字段中输入 *, 将显示系统中定义的所有队列的列表。然后, 从此列表中选择需要处理的队列。
- 从初始面板开始, 在该面板中, 您可以通过填写 **对象类型** 和 **名称** 字段来指定要使用的对象。

改变对象定义

要更改对象定义, 请指定操作 3, 然后按 Enter 键以查看 ALTER 面板。这些面板与 DEFINE 面板非常相似。您可以更改所需的值。完成更改后, 按 Enter 键以将信息发送到队列管理器。

显示对象定义

如果要查看对象的详细信息而无法对其进行更改，请指定操作 1 并按 Enter 键以查看 DISPLAY 面板。同样，这些面板与 DEFINE 面板相似，只是您无法更改任何字段。更改对象名以显示另一个对象的详细信息。

删除对象

要删除对象，请指定操作 4 (管理)，删除操作是生成的菜单中显示的操作之一。选择删除操作。

将要求您确认您的请求。如果按功能键 F3 或 F12，那么将取消请求。如果按 Enter 键，那么将确认该请求并将其传递到队列管理器。然后，将删除您指定的对象。

注：除非启动通道启动程序，否则无法删除大多数类型的通道对象。

使用名称列表

使用名称列表时，请像处理其他对象一样继续操作。

对于操作 DEFINE LIKE 或 ALTER，按功能键 F11 以向列表添加名称或更改列表中的名称。这涉及使用 ISPF 编辑器，并且所有常规 ISPF 编辑命令都可用。在单独的行上输入名称列表中的每个名称。

以此方式使用 ISPF 编辑器时，功能键设置是常规 ISPF 设置，而不是其他操作和控制面板所使用的设置。

如果需要在列表中指定小写名称，请在编辑器面板命令行上指定 CAPS (OFF)。执行此操作时，在指定 CAPS (ON) 之前，您将来编辑的所有名称列表都是小写的。

完成编辑名称列表后，按功能键 F3 以结束 ISPF 编辑会话。然后按 Enter 键将更改发送到队列管理器。

注意：如果在此阶段未按 Enter 键，而是按功能键 F3，那么将丢失输入的任何更新。

使用多个集群传输队列实现系统

如果在单个集群中使用通道，或者在重叠的集群中使用通道，那么不会产生任何影响。选择并启动通道时，通道将根据定义选择传输队列。

过程

- 如果您正在使用 DEFCLXQ 选项，请参阅第 302 页的『使用队列的自动定义和切换』。
- 如果使用的是分阶段方法，请参阅第 303 页的『使用分阶段方法更改集群发送方通道』。

使用队列的自动定义和切换

如果计划使用 DEFCLXQ 选项，请使用此选项。将为每个通道和每个新通道创建一个队列。

过程

1. 查看 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 并更改属性 (如果需要)。此队列在成员 SCSQPROC (csq4insx) 中定义。
2. 创建 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 模型队列。
3. 对此模型队列和 SYSTEM.CLUSTER.TRANSMIT.** 队列。

对于 z/OS，通道启动程序启动式任务用户标识需要：

- 控制对 CLASS (MQADMIN) 的访问权

```
ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channelName
```

- 更新对 CLASS (MQQUEUE) 的访问权

```
ssid.SYSTEM.CLUSTER.TRANSMIT.channelName
```

使用分阶段方法更改集群发送方通道

如果计划使用登台方法，请使用此选项。此过程允许您在不同时间移至新的集群发送方通道，以满足企业的需求。

开始之前

- 确定业务应用程序以及使用的通道。
- 对于您使用的队列，显示它们所位于的集群。
- 显示通道以显示连接名称，远程队列管理器的名称以及通道支持的集群。

关于此任务

- 创建传输队列。在 z/OS 上，您可能要考虑用于队列的页集。
- 为队列设置安全策略。
- 更改任何队列监视以包含此队列名称。
- 确定要使用此传输队列的通道。通道应该具有类似的名称，因此通用字符 '*' 在 CLCHNAME 中标识通道。
- 当您准备好使用新功能时，请改变传输队列以指定要使用此传输队列的通道的名称。例如，CLUSTER1.TOPARIS 或 CLUSTER1.* 或 *.TOPARIS
- 启动通道

过程

1. 使用 DIS CLUSQMGR(xxxx) XMITQ 命令可显示集群中定义的集群发送方通道，其中 xxxx 是远程队列管理器的名称。
2. 设置传输队列的安全概要文件，并将队列访问权授予通道启动程序。
3. 定义要使用的传输队列，并指定 USAGE (XMITQ) INDXTYPE (CORRELID) SHARE 和 CLCHNAME (*value*) 通道启动程序启动式任务用户标识需要以下访问权：

```
alter class(MQADMIN) ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channel  
update class(MQQUEUE ssid.SYSTEM.CLUSTER.TRANSMIT.channel
```

并且使用 SWITCH 命令的用户标识需要以下访问权：

```
alter cl(MQADMIN) ssid.QUEUE.queueuname
```

4. 停止并重新启动通道。

当通道开始使用 MQSC 命令或使用 CSQUTIL 时，将发生通道更改。您可以使用 CSQUTIL 的 SWITCH CHANNEL (*) STATUS 来确定需要重新启动哪些通道

如果在启动通道时迁到问题，请停止该通道，解决问题，然后重新启动该通道。

请注意，您可以根据需要随时更改 CLCHNAME 属性。

使用的 CLCHNAME 的值是启动通道时的值，因此您可以在通道从启动时开始继续使用定义时更改 CLCHNAME 定义。通道在重新启动时使用新定义。

在 z/OS 上撤销更改

如果结果与您预期的不一样，您需要有一个流程来回退更改。

什么会出问题？

如果新传输队列不是您期望的：

1. 检查 CLCHNAME 是否与您期望的一样

2. 查看作业记录以检查切换过程是否已完成。如果没有，请稍后等待并检查通道的新传输队列。

如果要使用多个集群传输队列，请务必显式设计传输队列定义，并避免复杂的重叠配置。通过这种方式，您可以确保如果存在问题，可以返回到原始队列和配置。

如果在使用其他传输队列的过程中迁到问题，那么必须先解决任何问题，然后才能继续进行更改。

必须先完成现有变更请求，然后才能发出新的变更请求。例如，您：

1. 定义最大深度为 1 的新传输队列，并且有 10 条消息等待发送。
2. 更改传输队列以在 CLCHNAME 参数中指定通道名称。
3. 停止并重新启动通道。尝试移动消息失败并报告问题。
4. 将传输队列上的 CLCHNAME 参数更改为空白。
5. 停止并重新启动通道。通道继续尝试并完成原始请求，因此通道继续使用新的传输队列。
6. 需要解决这些问题并重新启动通道，以便成功完成消息移动。

下次重新启动通道时，它会选取任何更改，因此如果将 CLCHNAME 设置为空白，那么通道将不使用指定的传输队列。

在此示例中，将传输队列上的 CLCHNAME 更改为空白并不一定表示通道使用 SYSTEM.CLUSTER.TRANSMIT 队列，因为可能有其他传输队列的 CLCHNAME 参数与通道名称匹配。例如，通用名称或队列管理器属性 DEFCLXQ 可能设置为通道，因此通道使用动态队列而不是 SYSTEM.CLUSTER.TRANSMIT 队列。

编写程序以管理 IBM MQ for z/OS

您可以编写自己的应用程序以管理队列管理器。使用本主题来了解编写您自己的管理程序的需求。

"常规-使用" 编程接口信息的开始

这组主题包含使您能够从 IBM MQ 应用程序发出 IBM MQ 命令的提示和指导。

注：在本主题中，将使用 C 语言表示法来描述 MQI 调用。有关 COBOL，PL/I 和汇编程序语言中调用的典型调用，请参阅 [函数调用手册](#)。

了解所有工作方式

在大纲中，从应用程序发出命令的过程如下所示：

1. 将 IBM MQ 命令构建到称为 请求消息的 IBM MQ 消息类型中。该命令可以是 MQSC 或 PCF 格式。
2. 发送 (使用 MQPUT) 此消息到称为 system-command 输入队列的特殊队列。IBM MQ 命令处理器运行该命令。
3. 检索 (使用 MQGET) 作为应答队列上的 *reply messages* 命令的结果。这些消息包含确定命令是否成功以及结果是否成功所需的用户消息。

然后由您的应用程序来处理结果。

这组主题包含：

为管理程序准备队列

管理程序需要大量预定义队列以用于系统命令输入和接收响应。

此部分适用于 MQSC 格式的命令。有关 PCF 中的等效项，请参阅 [第 19 页的『使用 IBM MQ 可编程命令格式』](#)。

必须先定义并打开要使用的队列，然后才能发出任何 MQPUT 或 MQGET 调用。

定义系统命令输入队列

系统命令输入队列是名为 SYSTEM.COMMAND.INPUT。提供的 CSQINP2 初始化数据集 thlqual.SCSQPROC(CSQ4INSG) 包含系统命令输入队列的缺省定义。为了与其他平台上的 IBM MQ 兼

容，此队列的别名称为 SYSTEM.ADMIN.COMMAND.QUEUE。有关更多信息，请参阅 [IBM MQ](#) 随附的样本定义。

定义应答队列

您必须定义应答队列以从 IBM MQ 命令处理器接收应答消息。它可以是具有允许将应答消息放入其中的属性的任何队列。但是，对于正常操作，请指定以下属性：

- 用法 (正常)
- NOTRIGGER (除非应用程序使用触发)

避免将持久消息用于命令，但如果选择这样做，那么应答队列不得是临时动态队列。

提供的 CSQINP2 初始化数据集 thlqual.SCSQPROC(CSQ4INSG) 包含名为 SYSTEM.COMMAND.REPLY.MODEL。您可以使用此模型来创建动态应答队列。

注：命令处理器生成的应答的长度可达 15000 字节。

如果使用永久动态队列作为应答队列，那么应用程序应允许在尝试删除队列之前完成所有 PUT 和 GET 操作的时间，否则可以返回 MQRC2055 (MQRC_Q_NOT_EMPTY)。如果发生此情况，请在几秒钟后重试队列删除。

打开系统命令输入队列

必须先将应用程序连接到队列管理器，然后才能打开系统命令输入队列。使用 MQI 调用 MQCONN 或 MQCONNX 来执行此操作。

然后使用 MQI 调用 MQOPEN 来打开系统命令输入队列。要使用此调用：

1. 将 **Options** 参数设置为 MQOO_OUTPUT
2. 按如下所示设置 MQOD 对象描述符字段：

ObjectType

MQOT_Q (对象是队列)

ObjectName

SYSTEM.COMMAND.INPUT

ObjectQMgrName

如果要将请求消息发送到本地队列管理器，请将此字段留空。这意味着将在本地处理您的命令。

如果要在远程队列管理器上处理 IBM MQ 命令，请将其名称放在此处。您还必须设置正确的队列和链接，如 [分布式排队和集群](#) 中所述。

打开应答队列

要从 IBM MQ 命令检索应答，必须打开应答队列。执行此操作的一种方法是在 MQOPEN 调用中指定模型队列 SYSTEM.COMMAND.REPLY.MODEL，以创建永久动态队列作为应答队列。要使用此调用：

1. 将 **Options** 参数设置为 MQOO_INPUT_SHARED
2. 按如下所示设置 MQOD 对象描述符字段：

ObjectType

MQOT_Q (对象是队列)

ObjectName

应答队列的名称。如果您指定的队列名称是模型队列对象的名称，那么队列管理器将创建动态队列。

ObjectQMgrName

要在本地队列管理器上接收应答，请将此字段留空。

DynamicQName

指定要创建的动态队列的名称。

命令服务器是使用命令处理器组件的 IBM MQ 组件。您可以将格式化消息发送到命令服务器，该命令服务器将解释消息，运行管理请求以及将响应发送回管理应用程序。

命令服务器从系统命令输入队列读取请求消息，对它们进行验证，并将有效的消息作为命令传递到命令处理器。命令处理器处理命令，并将任何应答作为应答消息放在您指定的应答队列上。第一条应答消息包含用户消息 CSQN205I。请参阅第 309 页的『解释来自命令服务器的应答消息』以获取更多信息。命令服务器还处理通道启动程序和队列共享组命令 (无论从何处发出这些命令)。

标识用于处理命令的队列管理器

处理从管理程序发出的命令的队列管理器是拥有将消息放入的系统命令输入队列的队列管理器。

启动命令服务器

通常，启动队列管理器时会自动启动命令服务器。一旦从 START QMGR 命令返回消息 CSQ9022I 'START QMGR' NORMAL 完成，它就会变为可用。在系统终止阶段，当所有已连接的任务都已断开连接时，命令服务器将停止。

您可以使用 START CMDSERV 和 STOP CMDSERV 命令自行控制命令服务器。为了防止命令服务器在 IBM MQ 重新启动时自动启动，您可以将 STOP CMDSERV 命令添加到 CSQINP1 或 CSQINP2 初始化数据集。但是，建议不要这样做，因为这会阻止处理任何通道启动程序或队列共享组命令。

STOP CMDSERV 命令在处理完当前消息后立即停止命令服务器，如果未处理任何消息，则立即停止命令服务器。

如果命令服务器已被程序中的 STOP CMDSERV 命令停止，那么不能处理程序中的其他命令。要重新启动命令服务器，必须从 z/OS 控制台发出 START CMDSERV 命令。

如果在队列管理器运行时停止并重新启动命令服务器，那么在命令服务器重新启动时，将处理系统命令输入队列上在命令服务器停止时的所有消息。但是，如果在命令服务器停止后停止并重新启动队列管理器，那么在命令服务器重新启动时将仅处理系统命令输入队列上的持久消息。系统命令输入队列上的所有非持久消息都将丢失。

向命令服务器发送命令

对于每个命令，构建包含该命令的消息，然后将其放入系统命令输入队列。

构建包含 IBM MQ 命令的消息

您可以通过构建包含所需命令的请求消息，将 IBM MQ 命令合并到应用程序中。对于每个这样的命令：

1. 创建包含表示命令的字符串的缓冲区。
2. 发出 MQPUT 调用，在调用的 **buffer** 参数中指定缓冲区名称。

在 C 中执行此操作的最简单方法是使用 "char" 定义缓冲区。例如：

```
char message_buffer[ ] = "ALTER QLOCAL(SALES) PUT(ENABLED)";
```

构建命令时，请使用以 null 结束的字符串。请勿在以此方式定义的命令开头指定命令前缀字符串 (CPF)。这意味着如果要在另一个队列管理器上运行命令脚本，那么不必更改这些脚本。但是，您必须考虑将 CPF 包含在放入应答队列的任何响应消息中。

命令服务器将所有小写字符折叠为大写，除非它们包含在引号内。

命令可以是最大长度为 32 762 个字符的任何长度。

将消息放在系统命令输入队列上

使用 MQPUT 调用将包含命令的请求消息放入系统命令输入队列。在此调用中，指定已打开的应答队列的名称。

要使用 MQPUT 调用：

1. 设置以下 MQPUT 参数：

Hconn

MQCONN 或 MQCONNX 调用返回的连接句柄。

Hobj

由系统命令输入队列的 MQOPEN 调用返回的对象句柄。

BufferLength

格式化命令的长度。

Buffer

包含命令的缓冲区的名称。

2. 设置以下 MQMD 字段：

MsgType

MQMT_REQUEST

Format

MQFMT_STRING 或 MQFMT_NONE

如果您未使用与队列管理器相同的代码页，请根据需要设置 *CodedCharSetId* 并设置 MQFMT_STRING，以便命令服务器可以转换消息。请勿设置 MQFMT_ADMIN，因为这会导致将命令解释为 PCF。

ReplyToQ

应答队列的名称。

ReplyToQMGr

如果要将回复发送到本地队列管理器，请将此字段留空。如果要将 IBM MQ 命令发送到远程队列管理器，请将其名称放在此处。您还必须设置正确的队列和链接，如 [分布式排队和集群中](#) 所述。

3. 根据需要设置任何其他 MQMD 字段。通常应该对命令使用非持久消息。
4. 根据需要设置任何 *PutMsgOpts* 选项。

如果指定 MQPMO_SYNCPOINT (缺省值)，那么必须遵循带有同步点调用的 MQPUT 调用。

使用 MQPUT1 和 system-command 输入队列

如果只想将一条消息放在系统命令输入队列上，那么可以使用 MQPUT1 调用。此调用在一个调用中组合了 MQOPEN 的函数，后跟一个消息的 MQPUT，后跟一个 MQCLOSE。如果使用此调用，请相应地修改参数。有关详细信息，请参阅 [使用 MQPUT1 调用将一条消息放入队列中](#)。

检索对命令的回复

命令服务器针对它接收的每条请求消息向应答队列发送响应。任何管理应用程序都必须接收并处理应答消息。

当命令处理器处理命令时，任何应答消息都将放入 MQPUT 调用中指定的应答队列中。命令服务器发送与它接收的命令消息具有相同持久性的应答消息。

正在等待应答

使用 MQGET 调用从请求消息中检索应答。一条请求消息可以生成多条应答消息。有关详细信息，请参阅第 309 页的『[解释来自命令服务器的应答消息](#)』。

您可以指定 MQGET 调用等待生成应答消息的时间间隔。如果未获得回复，请使用以主题 [第 309 页的『如果您未收到回复』](#) 开头的核对表。

要使用 MQGET 调用：

1. 设置以下参数：

Hconn

MQCONN 或 MQCONNX 调用返回的连接句柄。

Hobj

MQOPEN 调用为应答队列返回的对象句柄。

Buffer

接收应答的区域的名称。

BufferLength

用于接收应答的缓冲区的长度。这必须至少为 80 个字节。

2. 要确保仅从发出的命令获取响应，必须指定相应的 *MsgId* 和 *CorrelId* 字段。这些取决于您在 MQPUT 调用中指定的报告选项 MQMD_REPORT:

MQRO_NONE

二进制零，'00 ... 00' (24 个空值)。

MQRO_NEW_MSG_ID

二进制零，'00 ... 00' (24 个空值)。

如果未指定任何这些选项，那么这是缺省值。

MQRO_PASS_MSG_ID

MQPUT 中的 *MsgId*。

MQRO_NONE

来自 MQPUT 调用的 *MsgId*。

MQRO_COPY_MSG_ID_TO_CORREL_ID

来自 MQPUT 调用的 *MsgId*。

如果未指定任何这些选项，那么这是缺省值。

MQRO_PASS_CORREL_ID

来自 MQPUT 调用的 *CorrelId*。

有关报告选项的更多详细信息，请参阅 [报告选项和消息标志](#)。

3. 设置以下 *GetMsgOpts* 字段:

Options

MQGMO_WAIT

如果使用的代码页与队列管理器不同，请设置 MQGMO_CONVERT，并在 MQMD 中根据需要设置 *CodedCharSetId*。

WaitInterval

对于来自本地队列管理器的应答，请尝试 5 秒。以毫秒为单位进行编码，这将变为 5 000。对于来自远程队列管理器的应答以及通道控制和状态命令，请尝试 30 秒。以毫秒为单位进行编码，这将变为 30 000。

已丢弃的消息数

如果命令服务器发现请求消息无效，那么它将废弃此消息并将消息 CSQN205I 写入指定的应答队列。如果没有应答队列，那么会将 CSQN205I 消息放入死信队列中。此消息中的返回码显示原始请求消息无效的原因:

00D5020F 它不是 MQMT_REQUEST 类型。

00D50210 它的长度为零。

00D50212 长度超过 32 762 个字节。

00D50211 它包含所有空白。

00D5483E 它需要转换，但 *Format* 不是 MQFMT_STRING。

其他 请参阅 [命令服务器代码](#)

命令服务器应答消息描述符

对于任何应答消息，将设置以下 MQMD 消息描述符字段：

| | |
|--------------------|------------------|
| <i>MsgType</i> | MQMT_REPLY |
| <i>Feedback</i> | MQFB_NONE |
| <i>Encoding</i> | MQENC_NATIVE |
| <i>Priority</i> | 对于您发出的消息中的 MQMD。 |
| <i>Persistence</i> | 对于您发出的消息中的 MQMD。 |
| <i>CorrelId</i> | 取决于 MQPUT 报告选项。 |
| <i>ReplyToQ</i> | 无。 |

命令服务器将 MQPMO 结构的 *Options* 字段设置为 MQPMO_NO_SYNCPOINT。这意味着您可以在创建回复时进行检索，而不是在下一个同步点作为组进行检索。

z/OS 解释来自命令服务器的应答消息

IBM MQ 正确处理的每条请求消息至少生成两条应答消息。每条回复消息都包含一条 IBM MQ 用户消息。

应答的长度取决于发出的命令。您可以获得的最长应答来自 DISPLAY NAMELIST，长度可达 15000 字节。

第一条用户消息 CSQN205I 始终包含：

- 回复的计数（十进制），您可以将其用作循环中的计数器以获取其余回复。计数包括此第一条消息。
- 来自命令预处理器的返回码。
- 原因码，这是来自命令处理器的原因码。

此消息不包含 CPF。

例如：

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
```

COUNT 字段的长度为 8 字节，并且右对齐。它始终从位置 18 开始，即紧跟在 COUNT= 之后。RETURN 字段的长度为 8 字节，以十六进制字符表示，紧跟在位置 35 的 RETURN= 之后。REASON 字段以十六进制字符表示的长度为 8 字节，紧跟在位置 52 的 REASON= 之后。

如果 RETURN= 值为 00000000，而 REASON= 值为 00000004，那么应答消息集不完整。检索 CSQN205I 消息指示的应答后，再发出 MQGET 调用以等待另一组应答。下一组回复中的第一条消息又是 CSQN205I，指示有多少个回复，以及是否还有更多的回复。

请参阅 [IBM MQ for z/OS 消息、完成和原因码](#) 文档以获取有关各个消息的更多详细信息。

如果您正在使用非英语语言功能，那么回复的文本和布局与此处显示的内容不同。但是，计数和返回码在消息 CSQN205I 中的大小和位置相同。

z/OS 如果您未收到回复

如果未收到对命令服务器的请求的响应，那么可以执行一系列步骤。

如果未收到对请求消息的回复，请完成以下核对表：

- 命令服务器在运行吗？

- *WaitInterval* 够长了吗?
- 是否正确定义了系统命令输入和应答队列?
- 对这些队列的 MQOPEN 调用是否成功?
- 是否同时为 MQPUT 和 MQGET 调用启用了系统命令输入和应答队列?
- 是否考虑增加队列的 MAXDEPTH 和 MAXMSGL 属性?
- 是否正确使用 *CorrelId* 和 *MsgId* 字段?
- 队列管理器是否仍在运行?
- 是否正确构建了命令?
- 是否定义了所有远程链接并正常运行?
- 是否正确定义了 MQPUT 调用?
- 应答队列是否已定义为临时动态队列而不是永久动态队列? (如果请求消息是持久的, 那么必须使用永久动态队列进行应答。)

当命令服务器生成应答但无法将它们写入您指定的应答队列时, 它会将它们写入死信队列。

z/OS 使用 MGCRE 传递命令

通过适当的授权, 应用程序可以使用 z/OS 服务例程向多个队列管理器发出请求。

如果您具有正确的权限, 那么可以通过 MGCRE (SVC 34) z/OS 服务将 IBM MQ 命令从程序传递到多个队列管理器。CPF 的值标识命令所定向到的特定队列管理器。有关 CPF 的信息, 请参阅 [用于命令安全性和命令资源安全性的用户标识](#) 和 第 287 页的『在 z/OS 上发出队列管理器命令』。

如果使用 MGCRE, 那么可以使用命令和响应令牌 (CART) 来获取对该命令的直接响应。

z/OS 命令及其应答的示例

使用本主题作为命令服务器的一系列命令示例以及来自命令服务器的响应。

以下是可内置到 IBM MQ 消息中的命令的一些示例以及作为回复的用户消息。除非另有说明, 否则应答的每一行都是单独的消息。

- [来自 DEFINE 命令的消息](#)
- [来自 DELETE 命令的消息](#)
- [来自 DISPLAY 命令的消息](#)
- [来自带有 CMDSCOPE 的命令的消息](#)
- [来自使用 CMDSCOPE 生成命令的命令的消息](#)

来自 DEFINE 命令的消息

以下命令:

```
DEFINE QLOCAL(Q1)
```

生成以下消息:

```
CSQN205I    COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
```

这些应答消息在正常完成时生成。

来自 DELETE 命令的消息

以下命令:

```
DELETE QLOCAL(Q2)
```

生成以下消息:

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008  
CSQM125I +CSQ1 CSQMUQLC QLOCAL (Q2) QSGDISP(QMGR) WAS NOT FOUND  
CSQM090E +CSQ1 CSQMUQLC FAILURE REASON CODE X'00D44002'  
CSQ9023E +CSQ1 CSQMUQLC ' DELETE QLOCAL' ABNORMAL COMPLETION
```

这些消息指示名为 Q2 的本地队列不存在。

来自 DISPLAY 命令的消息

以下示例显示了来自某些 DISPLAY 命令的回复。

查找死信队列的名称

如果要查找队列管理器的死信队列的名称, 请从应用程序发出以下命令:

```
DISPLAY QMGR DEADQ
```

将返回以下三条用户消息, 您可以从中抽取所需名称:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000  
CSQM409I +CSQ1 QMNAME(CSQ1) DEADQ(SYSTEM.DEAD.QUEUE      )  
CSQ9022I +CSQ1 CSQMDRTS ' DISPLAY QMGR' NORMAL COMPLETION
```

DISPLAY QUEUE 命令中的消息

以下示例显示命令的结果如何依赖于该命令中指定的属性。

示例 1

使用以下命令定义本地队列:

```
DEFINE QLOCAL(Q1) DESCR('A sample queue') GET(ENABLED) SHARE
```

如果从应用程序发出以下命令:

```
DISPLAY QUEUE(Q1) SHARE GET DESCR
```

将返回以下三条用户消息:

```

CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(Q1                                ) TYPE(
QLOCAL ) QSGDISP(QMGR  )
DESCR(A sample queue
) SHARE GET(ENABLED )
CSQ9022I +CSQ1 CSQMMSG ' DISPLAY QUEUE' NORMAL COMPLETION

```

注: 此处显示的第二条消息 CSQM401I 占用四行。

示例 2

两个队列具有以字母 A 开头的名称:

- A1 是其 PUT 属性设置为 DISABLED 的本地队列。
- A2 是其 PUT 属性设置为 ENABLED 的远程队列。

如果从应用程序发出以下命令:

```

DISPLAY QUEUE(A*) PUT

```

将返回以下四条用户消息:

```

CSQN205I    COUNT=    4, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(A1                                ) TYPE(
QLOCAL ) QSGDISP(QMGR  )
PUT(DISABLED )
CSQM406I +CSQ1 QUEUE(A2                                ) TYPE(
QREMOTE ) PUT(ENABLED )
CSQ9022I +CSQ1 CSQMMSG ' DISPLAY QUEUE' NORMAL COMPLETION

```

注: 此处显示的第二条和第三条消息 CSQM401I 和 CSQM406I 占用三行和两行。

来自 DISPLAY NAMELIST 命令的消息

使用以下命令定义名称列表:

```

DEFINE NAMELIST(N1) NAMES(Q1,SAMPLE_QUEUE)

```

如果从应用程序发出以下命令:

```

DISPLAY NAMELIST(N1) NAMES NAMCOUNT

```

将返回以下三条用户消息:

```

CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM407I +CSQ1 NAMELIST(N1                              ) QS
GDISP(QMGR  ) NAMCOUNT(    2) NAMES(Q1
,SAMPLE_QUEUE
)
CSQ9022I +CSQ1 CSQMMSG ' DISPLAY NAMELIST' NORMAL COMPLETION

```

注: 此处显示的第二条消息 CSQM407I 占用了三行。

来自带有 **CMDSCOPE** 的命令的消息

以下示例显示了使用 **CMDSCOPE** 属性输入的命令的回复。

来自 **ALTER PROCESS** 命令的消息

以下命令:

```
ALT PRO(V4) CMDSCOPE(*)
```

生成以下消息:

```
CSQN205I COUNT= 2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'ALT PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I COUNT= 5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ26
CSQM125I !MQ26 CSQMMSGP PROCESS(V4) QSGDISP(QMGR) WAS NOT FOUND
CSQM090E !MQ26 CSQMMSGP FAILURE REASON CODE X'00D44002'
CSQ9023E !MQ26 CSQMMSGP 'ALT PRO' ABNORMAL COMPLETION
CSQN205I COUNT= 3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'ALT PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP 'ALT PRO' NORMAL COMPLETION
CSQN205I COUNT= 2, RETURN=0000000C, REASON=00000008
CSQN123E !MQ25 'ALT PRO' command for CMDSCOPE(*) abnormal completion
```

这些消息告诉您该命令是在队列管理器 MQ25 上输入的, 并发送到两个队列管理器 (MQ25 和 MQ26)。该命令在 MQ25 上成功, 但进程定义在 MQ26 上不存在, 因此该命令在该队列管理器上失败。

DISPLAY PROCESS 命令中的消息

以下命令:

```
DIS PRO(V*) CMDSCOPE(*)
```

生成以下消息:

```
CSQN205I COUNT= 2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS PRO' command accepted for CMDSCOPE(*), sent to 2
CSQN205I COUNT= 5, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ26
CSQM408I !MQ26 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ26 PROCESS(V3) QSGDISP(QMGR)
CSQ9022I !MQ26 CSQMDRTS 'DIS PROCESS' NORMAL COMPLETION
CSQN205I COUNT= 7, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS PRO' command responses from MQ25
CSQM408I !MQ25 PROCESS(V2) QSGDISP(COPY)
CSQM408I !MQ25 PROCESS(V2) QSGDISP(GROUP)
CSQM408I !MQ25 PROCESS(V3) QSGDISP(QMGR)
CSQM408I !MQ25 PROCESS(V4) QSGDISP(QMGR)
CSQ9022I !MQ25 CSQMDRTS 'DIS PROCESS' NORMAL COMPLETION
CSQN205I COUNT= 2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS PRO' command for CMDSCOPE(*) normal completion
```

这些消息告诉您该命令是在队列管理器 MQ25 上输入的, 并发送到两个队列管理器 (MQ25 和 MQ26)。将显示有关每个队列管理器上所有进程的信息, 名称以字母 V 开头。

来自 **DISPLAY CHSTATUS** 命令的消息

以下命令:

```
DIS CHS(VT) CMDSCOPE(*)
```

生成以下消息:

```
CSQN205I COUNT= 2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'DIS CHS' command accepted for CMDSCOPE(*), sent to 2
CSQN205I COUNT= 4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ25
CSQM422I !MQ25 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ25 CSQXDRTS 'DIS CHS' NORMAL COMPLETION
CSQN205I COUNT= 4, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DIS CHS' command responses from MQ26
CSQM422I !MQ26 CHSTATUS(VT) CHLDISP(PRIVATE) CONNAME( ) CURRENT STATUS(STOPPED)
CSQ9022I !MQ26 CSQXDRTS 'DIS CHS' NORMAL COMPLETION
CSQN205I COUNT= 2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DIS CHS' command for CMDSCOPE(*) normal completion
```

这些消息告诉您该命令是在队列管理器 MQ25 上输入的, 并发送到两个队列管理器 (MQ25 和 MQ26)。将显示有关每个队列管理器上的通道状态的信息。

STOP CHANNEL 命令中的消息

以下命令:

```
STOP CHL(VT) CMDSCOPE(*)
```

生成以下消息:

```
CSQN205I COUNT= 2, RETURN=00000000, REASON=00000004
CSQN137I !MQ25 'STOP CHL' command accepted for CMDSCOPE(*), sent to 2
CSQN205I COUNT= 3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQM134I !MQ25 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
SQN205I COUNT= 3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQM134I !MQ26 CSQMTCHL STOP CHL(VT) COMMAND ACCEPTED
CSQN205I COUNT= 3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ26
CSQ9022I !MQ26 CSQXCRPS 'STOP CHL' NORMAL COMPLETION
CSQN205I COUNT= 3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'STOP CHL' command responses from MQ25
CSQ9022I !MQ25 CSQXCRPS 'STOP CHL' NORMAL COMPLETION
CSQN205I COUNT= 2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'STOP CHL' command for CMDSCOPE(*) normal completion
```

这些消息告诉您该命令是在队列管理器 MQ25 上输入的, 并发送到两个队列管理器 (MQ25 和 MQ26)。通道 VT 已在每个队列管理器上停止。

来自使用 CMDSCOPE 生成命令的命令的消息

以下命令:

```
DEF PRO(V2) QSGDISP(GROUP)
```

生成以下消息:

```
CSQN205I COUNT= 3, RETURN=00000000, REASON=00000004
CSQM122I !MQ25 CSQMMSGP 'DEF PRO' COMPLETED FOR QSGDISP(GROUP)
CSQN138I !MQ25 'DEFINE PRO' command generated for CMDSCOPE(*), sent to 2
CSQN205I COUNT= 3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ25
CSQ9022I !MQ25 CSQMMSGP 'DEFINE PROCESS' NORMAL COMPLETION
CSQN205I COUNT= 3, RETURN=00000000, REASON=00000004
CSQN121I !MQ25 'DEFINE PRO' command responses from MQ26
CSQ9022I !MQ26 CSQMMSGP 'DEFINE PROCESS' NORMAL COMPLETION
CSQN205I COUNT= 2, RETURN=00000000, REASON=00000000
CSQN122I !MQ25 'DEFINE PRO' command for CMDSCOPE(*) normal completion
```

这些消息告诉您该命令是在队列管理器 MQ25 上输入的。在共享存储库上创建对象时，将生成另一个命令并将其发送到队列共享组 (MQ25 和 MQ26) 中的所有活动队列管理器。

管理 z/OS 上的 IBM MQ 资源

使用本主题中的链接可了解如何管理 IBM MQ for z/OS 所使用的资源，例如，管理日志文件，数据集，页集，缓冲池和耦合设施结构。

使用以下链接以获取在使用 IBM MQ for z/OS 时可能必须完成的不同管理任务的详细信息：

- [第 315 页的『管理日志』](#)
- [第 322 页的『管理引导数据集 \(BSDS\)』](#)
- [第 329 页的『管理页面集』](#)
- [第 334 页的『如何备份和恢复页集』](#)
- [第 337 页的『如何使用 CSQUTIL 备份和复原队列』](#)
- [第 337 页的『管理缓冲池』](#)
- [第 338 页的『在 z/OS 上管理队列共享组和共享队列』](#)

相关概念

[第 277 页的『管理 IBM MQ for z/OS』](#)

管理队列管理器和关联资源包含您为了激活和管理这些资源而经常执行的任务。选择您首选的方法来管理队列管理器和关联的资源。

[第 277 页的『向 IBM MQ for z/OS 发出命令』](#)

您可以在批处理或交互方式下使用 IBM MQ 脚本命令 (MQSC) 来控制队列管理器。

[第 347 页的『在 z/OS 上恢复并重新启动』](#)

使用本主题来了解 IBM MQ 使用的恢复和重新启动机制。

相关参考

[第 284 页的『IBM MQ for z/OS 实用程序』](#)

IBM MQ for z/OS 提供了一组实用程序，您可以使用这些程序来帮助进行系统管理。

相关信息

[IBM MQ for z/OS 概念](#)

[在 z/OS 上规划 IBM MQ 环境](#)

[在 z/OS 上配置队列管理器](#)

[可编程命令格式参考](#)

[MQSC 引用](#)

[使用 IBM MQ for z/OS 实用程序](#)

管理日志

使用本主题来了解如何管理 IBM MQ 日志文件，包括日志归档过程，使用日志记录压缩，日志记录恢复和打印日志记录。

本主题描述了管理 IBM MQ 日志所涉及的任务。它包含以下部分：

使用 ARCHIVE LOG 命令归档日志

只要需要使用 ARCHIVE LOG 命令，授权操作员就可以归档当前 IBM MQ 活动日志数据集。

发出 ARCHIVE LOG 命令时，IBM MQ 会截断当前活动日志数据集，然后运行异步卸载进程，并使用卸载进程的记录更新 BSDS。

ARCHIVE LOG 命令具有 MODE (QUIESCE) 选项。使用此选项，IBM MQ 作业和用户将在落实点之后停顿，并在卸载之前在当前活动日志中捕获生成的一致性点。

在规划非现场恢复的备份策略时，请考虑使用 MODE (QUIESCE) 选项。它会创建一个系统范围的一致性点，在恢复期间将归档日志与最新的备份页集副本配合使用时，可将数据不一致的次数降至最低。例如：

```
ARCHIVE LOG MODE(QUIESCE)
```

如果在未指定 TIME 参数的情况下发出 ARCHIVE LOG 命令，那么停顿时间段缺省为 CSQ6ARVP 宏的 QUIESCE 参数的值。如果 ARCHIVE LOG MODE (QUIESCE) 完成所需的时间小于指定的时间，那么该命令将成功完成；否则，当该时间段到期时，该命令将失败。可以使用 TIME 选项显式指定时间段，例如：

```
ARCHIVE LOG MODE(QUIESCE) TIME(60)
```

此命令指定在发生 ARCHIVE LOG 处理之前最多 60 秒的停顿周期。

注意：在时间关键时使用 TIME 选项可能会严重中断 IBM MQ 所有使用 IBM MQ 资源的作业和用户的可用性。

缺省情况下，将从提交该命令时开始异步处理该命令。（要与其他 IBM MQ 命令同步处理该命令，请将 WAIT (YES) 选项与 QUIESCE 配合使用，但请注意，在整个 QUIESCE 时间段内，z/OS 控制台已从 IBM MQ 命令输入锁定。）

在停顿期间：

- 允许队列管理器上的作业和用户执行落实处理，但如果他们在落实后尝试更新任何 IBM MQ 资源，那么将暂挂这些作业和用户。
- 仅读取数据的作业和用户可能会受到影响，因为他们可能正在等待被暂挂的作业或用户挂起的锁定。
- 新任务可以启动，但无法更新数据。

DISPLAY LOG 命令的输出使用消息 CSQV400I 来指示停顿已生效。例如：

```

CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value   SET value
-----
INBUFF      60
OUTBUFF     400
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
MAXCNOFF    0
WRTHRSH    20
DEALLCT     0
COMPLOG     NONE
ZHYWRITE    NO
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full PPRC DSName
  1      68 NO VICY.CSQ1.LOGCOPY1.DS01
  2      68 NO VICY.CSQ1.LOGCOPY2.DS01
Restarted at 2014-04-15 09:49:30 using RBA=000000000891B000
Latest RBA=000000000891CCF8
Offload task is AVAILABLE
Full logs to offload - 0 of 4
CSQV400I +CSQ1 ARCHIVE LOG QUIESCE CURRENTLY ACTIVE
CSQ9022I +CSQ1 CSQJC001 ' DISPLAY LOG' NORMAL COMPLETION

```



注意: zHyper“写入”在 IBM MQ 9.0 中未启用，因此 *NO* 是唯一允许的值。

当所有更新都已停顿时，BSDS 中的停顿历史记录将使用截断活动日志数据集的日期和时间以及当前活动日志数据集中最后写入的 RBA 进行更新。IBM MQ 截断当前活动日志数据集，切换到下一个可用的活动日志数据集，并发出消息 CSQJ311I，指示卸载进程已启动。

如果在停顿期到期之前无法停顿更新，那么 IBM MQ 将发出消息 CSQJ317I，并且 ARCHIVE LOG 处理将终止。不会截断当前活动日志数据集，也不会切换到下一个可用日志数据集，并且不会启动卸载过程。

无论停顿是否成功，都将恢复所有暂挂的用户和作业，并 IBM MQ 发出消息 CSQJ312I，声明停顿已结束，更新活动已恢复。

如果在当前活动日志是最后一个可用活动日志数据集时发出 ARCHIVE LOG，那么不会处理该命令，并且 IBM MQ 会发出以下消息：

```

CSQJ319I - csect-name CURRENT ACTIVE LOG DATA SET IS THE LAST
AVAILABLE ACTIVE LOG DATA SET. ARCHIVE LOG PROCESSING
WILL BE TERMINATED

```

如果在另一个 ARCHIVE LOG 命令已在进行时发出 ARCHIVE LOG，那么不会处理新命令，并且 IBM MQ 会发出以下消息：

```

CSQJ318I - ARCHIVE LOG COMMAND ALREADY IN PROGRESS

```

有关归档期间发出的消息的信息，请参阅 [IBM MQ for z/OS](#) 的消息。

失败后重新启动日志归档进程

如果在日志归档过程中存在问题 (例如，分配问题或磁带安装问题)，那么可能会暂挂活动日志的归档。您可以取消归档进程并使用 ARCHIVE LOG CANCEL OFFLOAD 命令将其重新启动。此命令将取消当前正在进行的任何卸载处理，并重新启动归档进程。它从尚未归档的最早日志数据集开始，并继续执行需要卸载的所有活动日志数据集。将重新启动已暂挂的任何日志归档操作。

仅当您确定当前日志归档任务不再起作用，或者要重新启动先前失败的尝试时，才使用此命令。这是因为该命令可能会导致卸载任务异常终止，这可能会导致转储。

z/OS 控制归档和日志记录

您可以使用 CSQ6LOGP, CSQ6ARVP 和 CSQ6SYSP 宏来控制压缩, 打印, 归档, 恢复和日志记录。请注意, 仅在 IBM MQ 日志中记录对专用对象的更改。还会记录对 GROUP 对象 (例如共享入站通道) 的更改, 因为定义会在组中传播并在本地保存。

在定制队列管理器时, 归档和日志记录的许多方面由使用系统参数模块的 CSQ6LOGP, CSQ6ARVP 和 CSQ6SYSP 宏设置的参数控制。请参阅 [任务 17: 定制系统参数模块](#) 以获取这些宏的详细信息。

使用 IBM MQ MQSC SET LOG, SET SYSTEM 和 SET ARCHIVE 命令运行队列管理器时, 可以更改其中一些参数。它们显示在 [第 318 页的表 26](#) 中:

| SET 命令 | 参数 |
|--------|---------------------------------------------|
| LOG | WRTHRSH, MAXARCH, DEALLCT, MAXRTU 和 COMPLOG |
| 归档 | 全部 |
| SYSTEM | LOGLOAD |

您可以使用 MQSC DISPLAY LOG, DISPLAY ARCHIVE 和 DISPLAY SYSTEM 命令来显示所有参数的设置。这些命令还显示有关归档和日志记录的状态信息。

控制日志压缩

您可以使用下列其中一项来启用和禁用日志记录压缩:

- MQSC 中的 SET 和 DISPLAY LOG 命令; 请参阅 [MQSC 命令](#)
- 正在调用 PCF 接口。请参阅 [第 19 页的『IBM MQ 可编程命令格式简介』](#)
- 在系统参数模块中使用 CSQ6LOGP 宏; 请参阅 [使用 CSQ6LOGP](#)

打印日志记录

您可以使用 CSQ1LOGP 实用程序来抽取和打印日志记录。有关指示信息, 请参阅 [日志打印实用程序](#)。

恢复日志

通常, 您不需要备份和复原 IBM MQ 日志, 尤其是在使用双重日志记录的情况下。但是, 在极少数情况下 (例如日志上的 I/O 错误), 您可能需要恢复日志。使用 Access Method Services 删除并重新定义数据集, 然后将相应的双日志复制到其中。

z/OS 正在废弃归档日志数据集

您可以废弃归档日志数据集, 并选择自动或手动废弃日志。

必须保留足够的日志数据才能执行工作单元恢复, 页集介质恢复 (如果页集丢失) 或 CF 结构介质恢复 (如果 CF 结构丢失)。请勿废弃恢复可能需要的归档日志数据集; 如果废弃这些归档日志数据集, 那么可能无法执行必需的恢复操作。

如果您已确认可以废弃归档日志数据集, 那么可以通过以下任一方式执行此操作:

- [自动归档日志数据集删除](#)
- [手动删除归档日志数据集](#)

自动归档日志数据集删除

您可以使用 DASD 或磁带管理系统来自动删除归档日志数据集。IBM MQ 归档日志数据集的保留期由 CSQ6ARVP 安装宏中的保留期字段 ARCRETN 指定 (请参阅 [使用 CSQ6ARVP](#) 以获取更多信息)。

保留期的缺省值指定归档日志将保留 9999 天 (最大值)。

要点: 您可以更改保留期, 但必须确保可以满足计划的备份周期数。

创建归档日志数据集时, IBM MQ 使用保留期值作为 JCL 参数 RETPD 的值。

此 IBM MQ 参数可覆盖 MVS/DFP 存储管理子系统 (SMS) 设置的保留期。通常, 保留期设置为 IBM MQ 或 SMS 指定的较小值。存储管理员和 IBM MQ 管理员必须同意适用于 IBM MQ 的保留期值。

注: IBM MQ 没有自动从 BSDS 中删除有关归档日志数据集的信息的方法, 因为某些磁带管理系统提供保留期的外部手动覆盖。因此, 在数据集保留期已到期且磁带管理系统已擦除数据集之后, 有关归档日志数据集的信息仍可以在 BSDS 中。相反, 可能已超过最大归档日志数据集数, 并且 BSDS 中的数据可能已在数据集到达其到期日期之前删除。

如果自动删除归档日志数据集, 请记住该操作不会更新 BSDS 中的归档日志列表。您可以使用更改日志清单实用程序来更新 BSDS, 如第 323 页的『更改 BSDS』中所述。更新不是必需的。记录旧的归档日志会浪费 BSDS 中的空间, 但不会造成其他损害。

手动删除归档日志数据集

必须将所有日志记录保留回消息 CSQI024I 和 CSQI025I 中标识的最低 RBA。此 RBA 是使用您在使用 [方法 1: 完全备份创建恢复点](#) 时发出的 DISPLAY USAGE 命令获取的。

阅读 [在废弃任何日志之前为非共享资源创建恢复点](#)。

查找并废弃归档日志数据集

在建立了恢复所需的最低日志 RBA 之后, 您可以通过执行以下过程来查找仅包含早期日志记录的归档日志数据集:

1. 使用打印日志映射实用程序来打印 BSDS 的内容。有关输出的示例, 请参阅 [打印日志映射实用程序](#)。
2. 查找标题为 ARCHIVE LOG COPY n DATA SETS 的输出部分。如果使用双重日志记录, 那么有两个部分。标记为 STARTRBA 和 ENDRBA 的列显示每个卷中包含的 RBA 的范围。查找范围包括通过消息 CSQI024I 和 CSQI025I 找到的最小 RBA 的卷。这些是您需要保留的最早卷。如果您正在使用双日志记录, 那么有两个这样的卷。

如果没有卷具有适当的范围, 那么下列其中一种情况适用:

- 最低 RBA 尚未归档, 您可以废弃所有归档日志卷。
- 当卷数超过 CSQ6LOGP 宏的 MAXARCH 参数所允许的数目时, BSDS 中环绕的归档日志卷的列表。如果 BSDS 未注册归档日志卷, 那么该卷无法用于恢复。因此, 请考虑向 BSDS 添加有关现有卷的信息。有关指示信息, 请参阅第 325 页的『归档日志的更改』。

还要考虑增大 MAXARCH 的值。有关信息, 请参阅 [使用 CSQ6LOGP](#)。

3. 删除具有小于要保留的最早卷的 STARTRBA 值的 ENDRBA 值的任何归档日志数据集或卷。如果您正在使用双重日志记录, 请删除这两个副本。

由于 BSDS 条目环绕, 因此 BSDS 归档日志部分中的前几个条目可能比末尾的条目更新换代。看日期和时间的结合, 比较他们的年龄。请勿假定您可以废弃包含最小 LOGRBA 的归档日志条目之前的所有条目。

删除数据集。如果归档在磁带上, 请擦除磁带。如果它们位于 DASD 上, 请运行 z/OS 实用程序以删除每个数据集。然后, 如果希望 BSDS 仅列出现有归档卷, 请使用更改日志库存实用程序 (CSQJU003) 来删除废弃卷的条目。请参阅第 325 页的『归档日志的更改』, 以获取示例。

长时间运行的事务可能会导致跨越日志数据集的工作单元日志记录。IBM MQ 通过使用日志分流 (一种移动日志记录以优化保留的日志数据量的方法) 和队列管理器重新启动时间来处理此场景。

当工作单元被视为长时, 每个日志记录的表示将进一步写到日志下。这称为日志分流。在 [日志文件](#) 中对其进行了更完整的描述。

队列管理器使用这些已中断的日志记录, 而不是故障后的原始日志记录, 以确保工作单元完整性。这有两个好处:

- 减少了为工作单元协调而必须保留的日志数据量
- 必须在队列管理器重新启动时遍历较少的日志数据, 因此将更快地重新启动队列管理器

已阻止的日志记录不包含用于介质恢复操作的足够信息。

日志中保存的数据用于两个不同的用途: 介质恢复和工作单元协调。如果发生影响 CF 结构或页集的介质故障, 那么队列管理器可以通过恢复先前副本并使用日志中包含的数据更新此副本, 将介质恢复到故障点。在工作单元中执行的持久活动将记录在日志中, 以便在发生故障时, 可以将其回退或在已更改的资源上恢复锁定。要启用队列管理器恢复, 需要保留的日志数据量受这两个元素影响。

对于介质恢复, 必须保留足够的日志数据, 以便能够至少从最近的介质副本执行介质恢复并能够回退。(您的站点可能规定了从较旧的备份恢复的能力。) 对于工作单元完整性, 必须保留正在运行或不确定的工作单元中最旧的日志数据。

为了帮助您管理系统, 队列管理器会检测每个日志归档中的旧工作单元, 并在消息 CSQJ160 和 CSQJ161 中报告这些工作单元。内部任务读取这些旧工作单元的工作单元日志信息, 并以更简洁的格式将其重写到日志中的当前位置。消息 CSQR026 指示发生此情况的时间。MQSC 命令 DISPLAY USAGE TYPE (DATASET) 还可以帮助您管理日志数据的保留时间。该命令报告 3 部分恢复信息:

1. 必须保留多少日志才能进行工作单元恢复
2. 为了对页集进行介质恢复, 必须保留多少日志
3. 对于队列共享组中的队列管理器, 必须保留多少日志以用于 CF 结构的介质恢复

对于其中每个项, 将尝试将所需的最旧的日志数据映射到数据集。随着新的工作单元开始和停止, 我们期望 (1) 移至日志中的最新位置。如果未移动, 那么长时间运行的 UOW 消息会警告您存在问题。(2) 与页集介质恢复相关 (如果要立即关闭并重新启动队列管理器)。它不知道您上次备份页集的时间, 也不知道发生页集故障时可能必须使用的备份。它通常在检查点处理期间移至日志中的较新位置, 因为缓冲池中保留的更改将写入页集。在 (3) 中, 队列管理器知道在此队列管理器或队列共享组中的其他队列管理器上执行的 CF 结构备份。但是, CF 结构恢复需要合并队列共享组中自上次备份以来与 CF 结构交互的所有队列管理器中的日志数据。这意味着日志数据由基于时间戳记的日志记录序号 (或 LRSN) 标识, 因此适用于整个队列共享组, 而不是在队列共享组中的不同队列管理器上不同的 RBA。由于在队列共享组中的此队列管理器或其他队列管理器上执行 BACKUP CFSTRUCT 命令, 因此它通常会移至日志中的较新位置。

使用本主题来了解如何重置队列管理器的日志。

不能允许队列管理器日志 RBA 从日志 RBA 范围的末尾回绕到 0, 因为这将导致队列管理器中断, 并且所有持久数据都将变为不可恢复。日志 RBA 的末尾是值 FFFFFFFFFFFFFFFF (如果使用 6 字节的 RBAs) 或 FFFFFFFFFFFFFFFF (如果使用 8 字节的 RBAs)。

队列管理器发出消息 CSQI045I, CSQI046E, CSQI047E, CSQJ031D 和 CSQJ032E 以指示所使用的日志范围很重要, 并且您应该计划执行操作以避免意外中断。

当 RBA 值达到 FFF800000000 (如果正在使用 6 字节的日志 RBA) 或 FFFFFFFC00000000 (如果正在使用 8 字节的日志 RBA) 时, 队列管理器将终止, 原因码为 00D10257。

如果正在使用 6 字节的日志 RBA, 请考虑将队列管理器转换为使用 8 字节的日志 RBA, 而不是按照 [实现较大的日志相对字节地址中描述的过程](#)来重置队列管理器的日志。转换队列管理器以使用 8 字节的日志 RBAs 需要比重置日志更短的中断, 并增加必须重置日志之前的时间段。

在队列管理器初始化期间发出的消息 CSQJ034I 指示队列管理器的日志 RBA 范围结束 (已配置), 并且可用于确定是否正在使用 6 字节或 8 字节的日志 RBA。

重置队列管理器日志的过程如下所示:

1. 解决任何未解决的工作单元。在队列管理器启动时, 消息 CSQR005I 中显示的未解析工作单元数为 INDOUBT 计数。在每个检查点以及在队列管理器关闭时, 队列管理器会自动发出命令

DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED) 以提供有关未解决的工作单元的信息。

请参阅 [如何解析不确定的恢复单元](#), 以获取有关解析恢复单元的信息。最终的方法是使用 **RESOLVE INDOUBT MQSC** 命令手动解析不确定的恢复单元。

2. 完全关闭队列管理器。

您可以使用 **STOP QMGR** 或 **STOP QMGR MODE(FORCE)**, 因为这两个命令都会将任何已更改的页面从缓冲池清空到页集。

3. 如果队列管理器是队列共享组的一部分, 请在其他队列管理器上对队列共享组中的所有结构执行 CFSTRUCT 备份。这将确保最近的备份不在此队列管理器的日志中, 并且此队列管理器的日志对于 CFSTRUCT 恢复不是必需的。
4. 使用 CSQJU003 定义新日志和 BSDS (请参阅 [更改日志库存实用程序](#) 以获取有关使用更改日志库存实用程序的更多信息)。
5. 针对此队列管理器的所有页集运行 **CSQUTIL RESETPAGE** (请参阅 [复制页面并重置日志](#) 以获取有关使用此功能的更多信息)。请注意, 可以独立重置页集 RBA, 因此可以提交多个并发作业 (例如, 每个页集一个) 以减少此步骤的耗用时间。
6. 重新启动队列管理器

相关概念

第 321 页的『实现较大的日志相对字节地址』

在 IBM MQ 8.0 之前, IBM MQ for z/OS 使用 6 字节的日志 RBA 来标识日志中数据的位置。从 IBM MQ 8.0 开始, 日志 RBA 可以是 8 字节长的, 这将增加必须重置日志之前的时间段。

z/OS 实现较大的日志相对字节地址

在 IBM MQ 8.0 之前, IBM MQ for z/OS 使用 6 字节的日志 RBA 来标识日志中数据的位置。从 IBM MQ 8.0 开始, 日志 RBA 可以是 8 字节长的, 这将增加必须重置日志之前的时间段。

需要显式启用此新功能。请参阅 [规划以增加最大可寻址日志范围](#), 以了解规划以启用 8 字节日志 RBA 时的注意事项。

按所示顺序执行以下指示信息, 以在单个 IBM MQ for z/OS 队列管理器上启用 8 字节日志 RBA:

1. 使用 **OPMODE** 启用 IBM MQ 8.0 新功能。

CD 对于队列共享组中的队列管理器, 您不需要执行总队列共享组中断。您可以依次停止每个队列管理器, 为 **OPMODE = (NEWFUNC,800)** 或 **OPMODE = (NEWFUNC,900)** 启用该队列管理器并将其重新启动。

CD 在使用 **OPMODE = (NEWFUNC,800)** 或 **OPMODE = (NEWFUNC,900)** 运行队列共享组中的所有队列管理器后, 请对队列共享组中的每个队列管理器执行以下步骤, 直到所有队列管理器都使用新的 BSDS 运行为止。

2. 将具有类似属性的新 BSDS 数据集分配给当前 BSDS。您可以定制样本 CSQ4BSDS 并删除任何不相关的语句, 也可以使用现有 JCL, 但将 BSDS 名称更改为类似于 ++HLQ++.NEW.BSDS01 的名称。

注意:

- a. 检查新 BSDS 的属性。唯一可能更改的属性是 BSDS 的大小。
 - b. 新 BSDS 包含当前 BSDS 的更多数据, 因此, 您必须确保为新数据集分配足够的可用空间。请参阅 [规划日志记录环境](#) 以及关联的主题, 以获取定义新 BSDS 时的建议值。
3. 完全关闭队列管理器。
 4. 运行 **BSDS 转换实用程序 (CSQJUCNV)** 以将现有 BSDS 转换为新的 BSDS 数据集。这通常需要几秒钟才能运行。

在此过程中不会更改现有 BSDS, 您可以将其用于队列管理器的初始化 (如果转换失败)。

5. 将当前 BSDS 重命名为旧 BSDS，将新 BSDS 重命名为当前 BSDS，以便在下次重新启动队列管理器时使用新数据集。可以使用 DFSMS Access Method Services ALTER 命令，例如：

```
ALTER '++HLQ++.BSDS01' NEWNAME('++HLQ++.OLD.BSDS01')
ALTER '++HLQ++.NEW.BSDS01' NEWNAME('++HLQ++.BSDS01')
```

确保您还发出命令来重命名 VSAM 集群的数据部分和索引部分。

6. 重新启动队列管理器。它的启动时间应该与使用 6 字节日志 RBA 时的启动时间相同。


如果队列管理器由于未能访问已转换的 BSDS 而未成功重新启动，请尝试确定失败原因，解决问题并重试操作。如果需要，请联系 IBM 支持中心以获取帮助。

如有必要，可通过以下方式在此时回退更改：

- a. 将当前 BSDS 重命名为新的 BSDS。
- b. 将旧 BSDS 重命名为当前 BSDS。
- c. 正在重新启动队列管理器。

使用转换后的 BSDS 成功重新启动队列管理器后，请勿尝试使用旧的 BSDS 来启动队列管理器。

7. 消息 CSQJ034I 在队列管理器初始化期间发出，以指示队列管理器的日志 RBA 结束 (已配置)。确认显示的日志 RBA 范围的末尾是 FFFFFFFFFFFFFFFFFF。这指示 8 字节日志 RBA 正在使用中。

注：  为了在新的 IBM MQ 9.0 队列管理器上启用 8 字节日志 RBA，在首次启动之前，必须首先创建空版本 1 格式的 BSDS，并将其用作 BSDS 转换实用程序的输入，以生成版本 2 格式的 BSDS。有关如何执行此过程的信息，请参阅 [创建引导程序和日志数据集](#)。

相关信息

[计划增大最大可寻址日志范围](#)

[较大的日志相对字节地址](#)

[BSDS 转换实用程序 \(CSQJUCNV\)](#)

管理引导数据集 (BSDS)

引导数据集 (BSDS) 用于引用日志数据集和日志记录。使用本主题来了解如何检查，更改和恢复 BSDS。

有关更多信息，请参阅 [引导程序数据集](#)。

本主题描述了管理引导数据集所涉及的任务。它包含以下部分：

- [第 322 页的『了解 BSDS 包含的内容』](#)
- [第 323 页的『更改 BSDS』](#)
- [第 327 页的『恢复 BSDS』](#)

了解 BSDS 包含的内容

可以使用打印日志映射实用程序 (CSQJU004) 来检查 BSDS 的内容。

打印日志映射实用程序 (CSQJU004) 是一个批处理实用程序，用于列出 BSDS 中存储的信息。有关运行它的指示信息，请参阅 [打印日志映射实用程序](#)。

BSDS 包含：

- [时间戳记](#)
- [活动日志数据集状态](#)

BSDS 中的时间戳记

打印日志映射实用程序的输出显示用于记录存储在 BSDS 中的各种系统事件的日期和时间的的时间戳记。

以下时间戳记包含在报告的标题部分中:

系统 Timestamp

反映上次更新 BSDS 的日期和时间。可以在以下情况下更新 BSDS 时间戳记:

- 队列管理器将启动。
- 在日志写入活动期间达到写入阈值。根据您指定的输出缓冲区数和系统活动速率, BSDS 可能每秒更新几次,也可能在几秒,几分钟甚至几小时内未更新。有关写阈值的详细信息,请参阅 [使用 CSQ6LOGP 中 CSQ6LOGP 宏的 WRTHRS 参数](#)。
- 由于发生错误, IBM MQ 从其正常的双 BSDS 方式降至单个 BSDS 方式。当获取,插入,指向,更新或删除 BSDS 记录的请求失败时,可能会发生此情况。发生此错误时, IBM MQ 会更新剩余 BSDS 中的时间戳记,以强制时间戳记与禁用的 BSDS 不匹配。

实用程序时间戳记

更改日志库存实用程序 (CSQJU003) 变更 BSDS 内容的日期和时间。

以下时间戳记包含在报告的活动和归档日志数据集部分中:

活动日志日期

在 BSDS 中创建活动日志条目的日期,即完成 CSQJU003 NEWLOG 的日期。

活动日志时间

在 BSDS 中创建活动日志条目的时间,即完成 CSQJU003 NEWLOG 的时间。

归档日志日期

在 BSDS 中创建归档日志条目的日期,即完成 CSQJU003 NEWLOG 或完成归档本身的日期。

归档日志时间

在 BSDS 中创建归档日志条目的时间,即完成 CSQJU003 NEWLOG 或完成归档本身的时间。

活动日志数据集状态

BSDS 将活动日志数据集的状态记录为下列其中一项:

新建

已定义数据集,但 IBM MQ 从未使用过该数据集,或者在首次使用该数据集之前已将日志截断为某个点。在任一情况下,数据集开始和结束 RBA 值都将重置为零。

可重用

数据集已定义但从未由 IBM MQ 使用,或者数据集已卸载。在打印日志映射输出中,最后一个 RE 可用数据集的起始 RBA 值等于最后一个归档日志数据集的起始 RBA 值。

不可重用

数据集包含尚未卸载的记录。

已停止

卸载处理器在读取记录时迁到错误,无法从活动日志的其他副本获取该记录。

已截断

请完成下面任意一项任务:

- 发生 I/O 错误, IBM MQ 已停止写入此数据集。将卸载活动日志数据集,从起始 RBA 开始,直到截断的活动日志数据集中的最后一个有效记录段。最后一个有效记录段的 RBA 低于活动日志数据集的结束 RBA。日志记录切换到下一个可用的活动日志数据集,并且不间断地继续。

或

- 已调用 ARCHIVE LOG 函数,该函数已截断活动日志。

状态显示在打印日志映射实用程序的输出中。

更改 BSDS

您不必执行特殊步骤以使用日志记录事件记录来更新 BSDS,因为 IBM MQ 会自动执行此操作。

但是,如果执行以下任何操作,那么您可能想要更改 BSDS:

- 添加更多活动日志数据集。
- 将活动日志数据集复制到新分配的数据集,例如,在提供更大的活动日志分配时。

- 将日志数据集移至其他设备。
- 恢复损坏的 BSDS。
- 废弃过时的归档日志数据集。

您可以通过运行更改日志库存实用程序 (CSQJU003) 来更改 BSDS。仅当队列管理器处于不活动状态时运行此实用程序，否则您可能会获得不一致的结果。实用程序的操作由 SYSIN 数据集的语句控制。此部分显示了多个示例。有关完整指示信息，请参阅 [更改日志库存实用程序](#)。

仅当队列管理器处于不活动状态时，才能复制活动日志数据集，因为 IBM MQ 在队列管理器启动时将活动日志数据集分配为互斥 (DISP = OLD)。

z/OS 活动日志的更改

使用本主题可了解如何使用 BSDS 更改活动日志。

您可以使用更改日志实用程序在活动日志的 BSDS 中添加，删除和记录条目。此处仅显示了示例；请将显示的数据集名称替换为要使用的数据集名称。有关实用程序的更多详细信息，请参阅 [更改日志库存实用程序](#)。

请参阅以下部分以获取更多信息：

- [向 BSDS 添加记录条目](#)
- [从 BSDS 中删除有关活动日志数据集的信息](#)
- [记录有关 BSDS 中日志数据集的信息](#)
- [增大活动日志的大小](#)
- [使用 CSQJUFMT](#)

向 BSDS 添加记录条目

如果活动日志已标记为“已停止”，那么不会将其用于日志记录；但是，它将继续用于读取。使用访问方法服务来定义新的活动日志数据集，然后使用更改日志库存实用程序在 BSDS 中注册新的数据集。例如，使用：

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY1
NEWLOG DSNAME=MQM111.LOGCOPY2.DS10,COPY2
```

如果要将旧活动日志数据集的内容复制到新活动日志数据集，那么还可以在 NEWLOG 功能上提供 RBA 范围和开始和结束时间戳记。

从 BSDS 中删除有关活动日志数据集的信息

要从 BSDS 中删除有关活动日志数据集的信息，可以使用：

```
DELETE DSNAME=MQM111.LOGCOPY1.DS99
DELETE DSNAME=MQM111.LOGCOPY2.DS99
```

在 BSDS 中记录有关日志数据集的信息

要记录有关 BSDS 中现有活动日志数据集的信息，请使用：

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY2,STARTIME=19930212205198,
ENDTIME=19930412205200,STARTRBA=6400,ENDRBA=94FF
```

您可能需要在 BSDS 中插入包含此类信息的记录，因为：

- 已删除数据集的条目，但再次需要该条目。
- 您正在将一个活动日志数据集的内容复制到另一个数据集。

- 您正在从备份副本中恢复 BSDS。

增大活动日志的大小

实现这一过程有两种方法。

1. 当队列管理器处于活动状态时:

- a. 使用 JCL 定义新的较大日志数据集。
- b. 使用 MQSC DEFINE LOG 命令将新的日志数据集添加到活动队列管理器。
- c. 使用 MQSC ARCHIVE LOG 命令将当前活动日志移动到新的更大日志。
- d. 等待较小的活动日志数据集的归档完成。
- e. 使用 CSQJU003 实用程序来除去旧的小型活动日志，从而关闭队列管理器。
- f. 重新启动队列管理器。

2. 当队列管理器处于不活动状态时:

- a. 停止队列管理器。此步骤是必需的，因为 IBM MQ 会分配所有活动日志数据集，以便在其处于活动状态时独占使用。
- b. 使用带有 NEWNAME 选项的 Access Method Services ALTER 来重命名活动日志数据集。
- c. 使用 Access Method Services DEFINE 定义更大的活动日志数据集。
通过复用旧数据集名称，您不必运行更改日志清单实用程序来在 BSD 中建立新名称。旧数据集名称和正确的 RBA 范围已存在于 BSD 中。
- d. 使用 Access Method Services REPRO 将旧 (已重命名) 数据集复制到你相应的新数据集。
注: 此步骤可能需要很长时间，因此您的企业可能在此期间无法采取行动。
- e. 启动队列管理器。

如果所有日志数据集的大小都相同，那么您的系统将在操作上更一致且更高效。如果日志数据集大小不同，那么更难以跟踪系统的日志，因此会浪费空间。

CSQJUFMT 的使用

增大活动日志的大小时，请勿运行 CSQJUFMT 格式。

如果您运行 CSQJUFMT (为了在队列管理器首次写入新的活动日志时提供性能优势)，那么您将接收消息:

```
IEC070I 203-204,XS95GTLX,REPRO02,OUTPUT,B857,SPMG02, 358
IEC070I MG.W.MG4E.LOGCOPY1.DS02,MG.W.MG4E.LOGCOPY1.DS02.DATA,
IDC3302I ACTION ERROR ON MG.W.MG4E.LOGCOPY1.DS02
IDC3351I ** VSAM I/O RETURN CODE IS 28 - RPLFDBWD = X'2908001C'
IDC31467I MAXIMUM ERROR LIMIT REACHED.

IDC0005I NUMBER OF RECORDS PROCESSED WAS 0
```

此外，如果使用 Access Method Services REPRO，请确保定义新的空日志。

如果使用 REPRO 将旧 (已重命名) 数据集复制到你各自的新数据集，那么缺省值为 NOREPLACE。

这意味着 REPRO 不会替换已在指定数据集上的记录。对数据集执行格式化时，将重置 RBA 值。净结果是格式化后不为空的数据集。

归档日志的更改

使用本主题来了解如何更改归档日志。

您可以在 BSDS 中为归档日志添加条目，从条目中删除条目以及更改条目的密码。此处仅显示了示例; 请将显示的数据集名称替换为要使用的数据集名称。有关实用程序的更多详细信息，请参阅 [更改日志库存实用程序](#)。

- [添加归档日志](#)
- [删除归档日志](#)
- [更改归档日志的密码](#)

添加归档日志

当对象的恢复依赖于读取现有归档日志数据集时， BSDS 必须包含有关该数据集的信息，以便 IBM MQ 可以找到该数据集。要在 BSDS 中注册有关现有归档日志数据集的信息，请使用：

```
NEWLOG DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04,  
UNIT=TAPE,STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
```

删除归档日志

要删除一个或多个卷上的整个归档日志数据集，请使用：

```
DELETE DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04
```

更改归档日志的密码

如果更改现有归档日志数据集的密码，那么还必须更改 BSDS 中的信息。

1. 使用打印日志映射实用程序列出 BSDS。
2. 使用 CSQJU003 实用程序的 DELETE 函数删除具有已更改密码的归档日志数据集的条目 (请参阅主题 [更改日志清单实用程序](#))。
3. 将数据集命名为新的归档日志数据集。使用 CSQJU003 实用程序的 NEWLOG 函数 (请参阅主题 [更改日志清单实用程序](#))，并提供新密码，起始和结束 RBA 以及卷序列号 (可在打印日志映射实用程序输出中找到)，请参阅 [打印日志映射实用程序](#)。

要更改新归档日志数据集的密码，请使用：

```
ARCHIVE PASSWORD= password
```

要停止在新的归档日志数据集上放置密码，请使用：

```
ARCHIVE NOPASSWD
```

注：仅当您没有外部安全管理器时，才使用 ARCHIVE 实用程序功能。

更改日志和 BSDS 的高级限定符 (HLQ)

使用本主题来了解更改高级限定符 (HLQ) 所需的过程。

开始之前

在将任何日志或数据集复制到新数据集之前，必须正常结束队列管理器。这是为了确保数据一致，并且在重新启动期间不需要恢复。

关于此任务

此任务提供有关如何更改日志和 BSDS 的 HLQ 的信息。为此，请完成下列步骤：

过程

1. 运行日志打印实用程序 CSQJU004 以记录日志数据集信息。稍后需要此信息。

2. 您可以执行以下任一操作：
 - a) 在要重命名的日志和 BSDS 数据集上运行 DSS 备份和复原，或者
 - b) 使用 AMS DEFINE 和 REPRO 来创建 HLQ 数据集并从旧数据集复制数据。
3. 修改 MSTR 和 CHIN 过程以指向新的数据集。
4. 使用 CSQJU003 删除 BSDS 新副本中的旧日志信息。
5. 使用 CSQJU003 的 NEWLOG 函数将新的日志数据集定义到新的 BSDS。除 HLQ 外，保持每个日志的所有信息都相同。
6. 新的 BSDS 应反映为旧 BSDS 中的旧日志记录的所有信息。HLQ 应该是唯一改变的东西。

下一步做什么

比较新旧 BSDS 的 CSQJU004 输出，以确保它们在启动队列管理器之前看起来完全相同 (HLQ 除外)。

注：执行这些操作时必须小心。不正确的操作可能会导致不可恢复的情况。请检查 PRINT LOG MAP UTILITY 输出，并确保已包含恢复或重新启动所需的所有信息。

恢复 BSDS

如果 IBM MQ 正在双 BSDS 方式下运行，并且一个 BSDS 已损坏，那么强制 IBM MQ 进入单 BSDS 方式，那么 IBM MQ 将继续运行而不会出现问题 (直到下次重新启动)。

要将环境恢复为双 BSDS 方式，请执行以下操作：

1. 使用 "访问方法服务" 来重命名或删除损坏的 BSDS，并定义与损坏的 BSDS 同名的新 BSDS。您可以在 thlqual.SCSQPROC 中的作业 CSQ4BREC 中找到控制语句示例。
2. 发出 IBM MQ 命令 RECOVER BSDS 以在新分配的数据集中生成有效 BSDS 的副本，并恢复双 BSDS 方式。

如果 IBM MQ 在单 BSDS 方式下运行，并且 BSDS 已损坏，或者如果 IBM MQ 在双 BSDS 方式下运行，并且这两个 BSD 都已损坏，那么队列管理器将停止并且不会重新启动，直到修复 BSDS 数据集为止。在这种情况下：

1. 找到与最新归档日志数据集关联的 BSDS。最近一次出现的消息 CSQJ003I 中的作业记录上显示了最新归档日志的数据集名称，这指示卸载处理已成功完成。在准备此过程的其余部分时，最好保留该消息所记录的所有成功归档的日志：
 - 如果归档日志位于 DASD 上，那么将在任何可用的 DASD 上分配 BSDS。BSDS 名称类似于相应的归档日志数据集名称；仅将最后一个限定符的第一个字母从 A 更改为 B，如下列示例中所示：

| |
|--------------------------------|
| 归档日志名称 |
| CSQ.ARCHLOG1. A 0000001 |
| BSDS 副本名称 |
| CSQ.ARCHLOG1. B 0000001 |
 - 如果归档日志位于磁带上，那么 BSDS 是第一个归档日志卷的第一个数据集。BSDS 不会在以后的卷上重复。
2. 如果最新的归档日志数据集没有 BSDS 副本 (例如，因为卸载时发生错误)，请从较早的卸载处理中找到 BSDS 的较早副本。
3. 使用带有 NEWNAME 选项的 Access Method Services ALTER 命令来重命名 *damaged* BSD。如果要删除损坏的 BSDS，请使用 "访问方法服务 DELETE" 命令。对于每个损坏的 BSDS，请使用 "访问方法服务" 将新的 BSDS 定义为替换数据集。thlqual.SCSQPROC 中的作业 CSQ4BREC 包含用于定义新 BSDS 的 "访问方法服务" 控制语句。
4. 使用 "访问方法服务" REPRO 命令将 BSDS 从归档日志复制到您在步骤 第 327 页的『3』中定义的其中一个替换 BSD。请勿将任何数据复制到第二个替换 BSDS，请在步骤 第 328 页的『5』中执行此操作。
 - a. 打印替换 BSDS 的内容。

使用打印日志映射实用程序 (CSQJU004) 来打印替换 BSDS 的内容。这使您能够在继续恢复工作之前查看 BSDS 替换件的内容。

b. 更新替换 BSDS 中的归档日志数据集库存。

检查打印日志映射实用程序的输出，并检查替换 BSDS 是否不包含从中复制 BSDS 的归档日志的记录。如果替换的 BSDS 是旧副本，那么其库存可能不包含最近创建的所有归档日志数据集。必须更新归档日志数据集的 BSDS 库存以反映当前子系统库存。

使用更改日志库存实用程序 (CSQJU003) NEWLOG 语句来更新替换 BSDS，并添加从中复制 BSDS 的归档日志的记录。如果归档日志数据集受密码保护，请使用 NEWLOG 函数的 PASSWORD 选项。此外，如果归档日志数据集已编目，请确保 NEWLOG 函数的 CATALOG 选项正确设置为 CATALOG=YES。使用 NEWLOG 语句来添加在 BSDS 副本之后创建的任何其他归档日志数据集。

c. 更新替换 BSDS 中的密码。

BSDS 包含归档日志数据集和活动日志数据集的密码。要确保替换 BSDS 中的密码反映安装所使用的当前密码，请使用带有 PASSWORD 选项的更改日志库存 ARCHIVE 实用程序函数。

d. 更新替换 BSDS 中的活动日志数据集库存。

在异常情况下，自复制 BSDS 以来，您的安装可能已添加，删除或重命名活动日志数据集。在这种情况下，替换 BSDS 不会反映您的安装当前正在使用的活动日志数据集的实际数目或名称。

如果需要从替换 BSDS 日志库存中删除活动日志数据集，请使用更改日志库存实用程序 DELETE 功能。

如果需要将活动日志数据集添加到替换 BSDS 日志库存，请使用更改日志库存实用程序 NEWLOG 功能。确保在 NEWLOG 函数上正确指定了 RBA 范围。如果活动日志数据集受密码保护，请使用 PASSWORD 选项。

如果需要重命名替换 BSDS 日志库存中的活动日志数据集，请使用更改日志库存实用程序 DELETE 函数，后跟 NEWLOG 函数。确保在 NEWLOG 函数上正确指定了 RBA 范围。如果活动日志数据集受密码保护，请使用 PASSWORD 选项。

e. 更新替换 BSDS 中的活动日志 RBA 范围。

稍后，当队列管理器重新启动时，它会将 BSDS 中列出的活动日志数据集的 RBA 与实际活动日志数据集中的 RBA 进行比较。如果 RBA 不同意，那么队列管理器不会重新启动。当使用 BSDS 的旧副本时，问题会放大。要解决此问题，请使用更改日志库存实用程序 (CSQJU003) 通过实际活动日志数据集中的 RBA 来调整 BSDS 中找到的 RBA。执行此操作的方法是：

- 使用打印日志记录实用程序 (CSQ1LOGP) 来打印活动日志数据集的摘要报告。这将显示起始和结束 RBA。
- 当已知所有活动日志数据集的 RBA 时，将实际 RBA 范围与刚刚打印的 RBA 范围进行比较。

如果所有活动日志数据集的 RBA 范围都相等，那么您可以继续执行下一个恢复步骤，而无需执行任何其他工作。

如果 RBA 范围不相等，请调整 BSDS 中的值以反映实际值。对于需要调整 RBA 范围的每个活动日志数据集，请使用更改日志库存实用程序 DELETE 函数从替换 BSDS 中的库存中删除活动日志数据集。然后使用 NEWLOG 函数将活动日志数据集重新定义到 BSDS。如果活动日志数据集受密码保护，请使用 NEWLOG 函数的 PASSWORD 选项。

f. 如果仅为活动日志的每个副本指定了两个活动日志数据集，那么在队列管理器重新启动期间，IBM MQ 可能会迂到困难。当其中一个活动日志数据集已满并且尚未卸载，而第二个活动日志数据集接近填充时，可能会出现此问题。在这种情况下，请为活动日志的每个副本添加新的活动日志数据集，并在替换 BSDS 日志库存中定义每个新的活动日志数据集。

使用 Access Method Services DEFINE 命令为活动日志的每个副本定义新的活动日志数据集，并使用更改日志库存实用程序 NEWLOG 函数在替换 BSDS 中定义新的活动日志数据集。您无需在 NEWLOG 语句上指定 RBA 范围。但是，如果活动日志数据集受密码保护，请使用 NEWLOG 函数的 PASSWORD 选项。可在 thlqual.SCSQPROC 中的作业 CSQ4LREC 中找到用于完成此任务的示例控制语句。

5. 将更新后的 BSDS 复制到第二个新的 BSDS 数据集。BSD 现在完全相同。

使用打印日志映射实用程序 (CSQJU004) 在此时打印第二个替换 BSDS 的内容。

6. 请参阅 [活动日志问题](#)，以获取有关在丢失当前活动日志数据集时要执行的操作的信息。

7. 使用新构造的 BSDS 重新启动队列管理器。 IBM MQ 确定当前 RBA 以及需要归档的活动日志。

z/OS 管理页面集

使用本主题来了解如何管理与队列管理器关联的页集。

本主题描述如何添加，复制和通常管理与队列管理器关联的页集。它包含以下部分：

- [第 329 页的『如何更改页集的高级限定符 \(HLQ\)』](#)
- [第 329 页的『如何向队列管理器添加页集』](#)
- [第 330 页的『当某个页面集变满时要执行的操作』](#)
- [第 330 页的『如何平衡页集上的负载』](#)
- [如何增加页集的大小](#)
- [第 333 页的『如何减少页集』](#)
- [第 334 页的『如何重新引入页集』](#)
- [第 334 页的『如何备份和恢复页集』](#)
- [第 337 页的『如何删除页集』](#)
- [第 337 页的『如何使用 CSQUTIL 备份和复原队列』](#)

请参阅 [页集](#)，以获取页集，存储类，缓冲区和缓冲池的描述以及适用的一些性能注意事项。

如何更改页集的高级限定符 (HLQ)

此任务提供有关如何更改页集的 HLQ 的信息。要执行此任务，请执行以下操作：

1. 定义新的 HLQ 页集。
2. 如果大小分配与旧页集相同，请使用 REPRO 将现有页集复制到空的新 HLQ 页集。如果要增大页集的大小，请使用 CSQUTIL 的 FORMAT 函数来格式化目标页集。有关更多信息，请参阅 [格式化页集 \(FORMAT\)](#)。
3. 使用 CSQUTIL 的 COPYPAGE 函数将所有消息从源页集复制到目标页集。有关更多信息，请参阅 [展开页集 \(COPYPAGE\)](#)。
4. 更改队列管理器过程中的 CSQP00xx DD 语句以指向新的 HLQ 页集。

重新启动队列管理器并验证对页集所作的更改。

如何向队列管理器添加页集

此描述假定您有一个已在运行的队列管理器。例如，如果您的队列管理器必须处理使用新队列的新应用程序，那么您可能需要添加页集。

要添加新的页集，请使用以下过程：

1. 定义并格式化新页集。您可以使用 thlqual.SCSQPROC(CSQ4PAGE) 中的样本 JCL 作为基础。有关更多信息，请参阅 [格式化页集 \(FORMAT\)](#)。
请注意不要格式化任何正在使用的页集，除非您打算这样做。如果是这样，请使用 FORMAT 实用程序函数的 FORCE 选项。
2. 使用带有 DSN 选项的 DEFINE PSID 命令将页集与缓冲池相关联。
3. 通过发出 DEFINE STGCLASS 命令为页集添加相应的存储类定义。
4. (可选) 记录队列管理器的配置方式：
 - a. 将新页集添加到队列管理器的启动式任务过程。
 - b. 将新页集的定义添加到 CSQINP1 初始化数据集。

c. 将新存储类的定义添加到 CSQ4INYR 初始化数据集成员。

有关 DEFINE PSID 和 DEFINE STGCLASS 命令的详细信息，请参阅 [DEFINE PSID](#) 和 [DEFINE STGCLASS](#)。

当某个页面集变满时要执行的操作

您可以使用 IBM MQ 命令 DISPLAY USAGE 来了解有关页集利用率的信息。例如，命令：

```
DISPLAY USAGE PSID(03)
```

显示页集 03 的当前状态。这将告诉您此页集有多少可用页面。

如果您为页集定义了辅助扩展数据块，那么每次填充这些扩展数据块时都会动态展开这些扩展数据块。最终，将使用所有辅助扩展数据块，或者没有更多可用磁盘空间。如果发生这种情况，那么应用程序将接收到返回码 MQRC_STORAGE_MEDIUM_FULL。

如果应用程序从 MQI 调用接收到 MQRC_STORAGE_MEDIUM_FULL 返回码，那么这明确指示页集上没有足够的剩余空间。如果问题仍然存在或可能重现，那么必须执行一些操作来解决此问题。

您可以通过多种方式来处理此问题：

- 通过将队列从一个页集移动到另一个页集来平衡页集之间的负载。
- 展开页集。有关指示信息，请参阅第 332 页的『如何增加页集的大小』。
- 重新定义页集，以便它可以扩展到超过 4 GB 的最大大小 64 GB。请参阅 [定义大于 4 GB 的页集](#) 以获取指示信息。

如何平衡页集上的负载

页集上的负载均衡意味着将与一个或多个队列相关联的消息从一个页集移动到另一个较少使用的页集。如果展开页集不实际，请使用此技术。

要确定哪些队列正在使用页集，请使用相应的 IBM MQ 命令。例如，要找出哪些队列映射到页集 02，首先，通过使用以下命令找出哪些存储类映射到页集 02：

```
DISPLAY STGCLASS(*) PSID(02)
```

然后使用以下命令来查找哪些队列使用哪个存储类：

```
DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS
```

移动非共享队列

要将队列及其消息从一个页集移动到另一个页集，请使用 MQSC MOVE QLOCAL 命令（如 MOVE QLOCAL 中所述）。当您标识了要移动到新页集的一个或多个队列时，请对这些队列中的每个队列执行以下过程：

1. 确保要移动的队列未被任何应用程序使用（即，DISPLAY QSTATUS 命令中的 IPPROCS 和 OPPROCS 值为零），并且该队列没有未落实的消息（DISPLAY QSTATUS 命令中的 UNCOM 值为 NO）。

注：确保此状态继续的唯一方法是临时更改队列的安全授权。请参阅 [队列安全性概要文件](#) 以获取更多信息。

如果无法执行此操作，那么当应用程序开始使用队列时（尽管执行了设置 PUT (DISABLED) 之类的预防性步骤），此过程中的后续阶段可能会失败。但是，此过程永远不会丢失消息。

2. 通过更改队列定义以禁用 MQPUT，防止应用程序将消息放入要移动的队列中。将队列定义更改为 PUT (DISABLED)。

3. 使用以下命令定义与要移动的队列具有相同属性的临时队列:

```
DEFINE QL(TEMP_QUEUE) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED)
```

注: 如果此临时队列在先前运行中已存在, 请在执行定义之前将其删除。

4. 使用以下命令将消息移至临时队列:

```
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_QUEUE)
```

5. 使用以下命令删除正在移动的队列:

```
DELETE QLOCAL(Queue_To_Move)
```

6. 定义映射到所需页集的新存储类, 例如:

```
DEFINE STGCLASS(NEW) PSID(nn)
```

将新的存储类定义添加到准备好下次重新启动队列管理器的 CSQINP2 数据集。

7. 通过更改存储类属性来重新定义要移动的队列:

```
DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) STGCLASS(NEW)
```

重新定义队列时, 它基于步骤 [第 331 页的『3』](#) 中创建的临时队列。

8. 使用以下命令将消息移回新队列:

```
MOVE QLOCAL(TEMP) TOQLOCAL(Queue_To_Move)
```

9. 不再需要步骤 [第 331 页的『3』](#) 中创建的队列。使用以下命令将其删除:

```
DELETE QL(TEMP_QUEUE)
```

10. 如果要移动的队列是在 CSQINP2 数据集中定义的, 请在 CSQINP2 数据集中更改相应 DEFINE QLOCAL 命令的 STGCLASS 属性。添加 REPLACE 关键字, 以便替换现有队列定义。

[第 332 页的图 46](#) 显示了从负载均衡作业中抽取的数据。

```

//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=th1qua1.SCSQANLE,DISP=SHR
//          DD DSN=th1qua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
ALTER QL(Queue_TO_MOVE) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(Queue_TO_MOVE) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(Queue_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
DELETE QL(Queue_TO_MOVE)
DEFINE STGCLASS(NEW) PSID(2)
DEFINE QL(Queue_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(Queue_TO_MOVE)
DELETE QL(TEMP_QUEUE)
/*

```

图 46: 从页集的负载均衡作业中抽取

如何增加页集的大小

最初可以分配大于 4 GB 的页集，请参阅 [定义大于 4 GB 的页集](#)

可以通过指定 EXPAND (SYSTEM) 或 EXPAND (USER) 将页集定义为在页集变满时自动展开。如果您的页集是使用 EXPAND (NONE) 定义的，那么可以通过以下两种方法之一将其展开：

- 更改其定义以允许自动扩展。请参阅 [更改页集以允许自动扩展](#)
- 创建新的更大页集，并将消息从旧页集复制到新页集。请参阅 [将消息移至新的更大页集](#)

将页集定义为大于 4 GB

IBM MQ 可以使用最大大小为 64 GB 的页集，前提是数据集是使用 VSAM 的 "扩展可寻址性" 定义的。扩展可寻址性是 SMS 数据类授予的属性。在以下样本 JCL 中显示的示例中，管理类 "EXTENDED" 定义为具有 "扩展可寻址性" 的 SMS。如果当前未将现有页集定义为具有扩展可寻址性，请使用以下方法来迁移到扩展可寻址性格式数据集。

1. 停止队列管理器。
2. 使用 "访问方法服务" 来重命名现有页集。
3. 定义目标页集，大小与现有页集相同，但带有 DATACLAS (EXTENDED)。

注：扩展格式数据集必须由 SMS 管理。以下是请求 VSAM 数据集的扩展格式的机制：

- 使用具有 DSNTYPE 值 EXT 和子参数 R 或 P 的数据类来指示必需或首选。
- 在 DD 语句上编码 DSNTYPE=EXTREQ (需要扩展格式) 或 DSNTYPE=EXTPREF (首选扩展格式)。
- 对 DD 语句上的 LIKE= 参数进行编码以引用现有扩展格式数据集。

有关更多信息，请参阅 [定义扩展格式数据集的限制](#)。

4. 使用 CSQUTIL 的 COPYPAGE 函数将所有消息从源页集复制到目标页集。请参阅 [展开页集 \(COPYPAGE\)](#) 以获取更多详细信息。
5. 重新启动队列管理器。
6. 更改页集以使用系统扩展，以允许其在当前分配后继续增长。

以下 JCL 显示了 "访问方法服务" 命令示例：

```

//S1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ALTER 'VICY.CSQ1.PAGE01' -
NEWNAME('VICY.CSQ1.PAGE01.OLD')
ALTER 'VICY.CSQ1.PAGE01.DATA' -
NEWNAME('VICY.CSQ1.PAGE01.DATA.OLD')
DEFINE CLUSTER (NAME('VICY.CSQ1.PAGE01') -
MODEL('VICY.CSQ1.PAGE01.OLD') -
DATACLAS(EXTENDED))
/*

```

改变页集以允许自动扩展

将 ALTER PSID 命令与 EXPAND (USER) 或 EXPAND (SYSTEM) 选项配合使用。请参阅 [ALTER PSID](#) 和 [展开页集 \(COPYPAGE\)](#)，以获取有关展开页集的常规信息。

将消息移动到新的更大的页集

此技术涉及停止和重新启动队列管理器。这将删除在重新启动时不在共享队列上的任何非持久消息。如果您有不希望删除的非持久消息，请改为使用负载均衡。要获取更多详细信息，请参阅第 330 页的『[如何平衡页集上的负载](#)』。在此描述中，要展开的页集称为源页集；新的较大页集称为目标页集。

请按照以下步骤操作：

1. 停止队列管理器。
2. 定义目标页集，确保其大于源页集，并具有较大的辅助扩展数据块值。
3. 使用 CSQUTIL 的 FORMAT 函数来格式化目标页集。有关更多详细信息，请参阅 [格式化页集 \(FORMAT\)](#)。
4. 使用 CSQUTIL 的 COPYPAGE 函数将所有消息从源页集复制到目标页集。请参阅 [展开页集 \(COPYPAGE\)](#) 以获取更多详细信息。
5. 通过执行下列其中一项操作，使用目标页集重新启动队列管理器：
 - 更改队列管理器启动式任务过程以引用目标页集。
 - 使用 "访问方法服务" 删除源页集，然后重命名目标页集，使其具有与源页集相同的名称。

注意：

在删除任何 IBM MQ 页集之前，请确保已生成所需的备份副本。

如何减少页集

阻止除 IBM MQ 管理员以外的所有用户使用队列管理器。例如，通过更改访问安全设置。

如果您有一个大部分为空的大页集（如 DISPLAY USAGE 命令所示），那么您可能希望减小其大小。执行此操作的过程涉及使用 CSQUTIL 的 COPY，FORMAT 和 LOAD 函数（请参阅 IBM MQ 实用程序）。此过程不适用于页集零 (0)，因为减少此页集的大小不切实际；唯一的方法是重新初始化队列管理器（请参阅第 353 页的『[重新初始化队列管理器](#)』）。此过程的先决条件是尝试从系统中除去所有用户，以便所有 UOW 都已完成并且页集一致。

1. 使用带有 QUIESCE 或 FORCE 属性的 STOP QMGR 命令来停止队列管理器。
2. 使用 PSID 选项运行 CSQUTIL 的 SCOPY 函数，以从大页集中复制所有消息数据并将其保存在连续数据集中。
3. 定义新的较小页集数据集以替换大页集。
4. 对您在步骤第 333 页的『[3](#)』中创建的页集运行 CSQUTIL 的 FORMAT TYPE (NEW) 函数。
5. 使用步骤第 333 页的『[3](#)』中创建的页集重新启动队列管理器。
6. 运行 CSQUTIL 的 LOAD 函数以重新装入步骤第 333 页的『[2](#)』期间保存的所有消息。
7. 允许所有用户访问队列管理器。
8. 删除旧的大页集。

如何重新引入页集

在某些场景中，能够使旧页集再次联机到队列管理器是很有用的。除非执行特定操作，否则当旧页集联机时，队列管理器将识别存储在页集本身和检查点记录中的页集恢复 RBA 是旧的，因此将自动启动该页集的介质恢复以更新该页集。

此类介质恢复只能在队列管理器重新启动时执行，并且可能需要相当长的时间，尤其是在必须读取磁带上保存的归档日志时。但是，通常在此情况下，页集在中间时间段内处于脱机状态，因此日志中不包含与页集恢复相关的信息。

提供了以下三个选项：

允许执行完全介质恢复。

1. 停止队列管理器。
2. 确保定义在队列管理器的启动式任务过程和 CSQINP1 初始化数据集中都可用于页集。
3. 重新启动队列管理器。

允许销毁页集上的任何消息。

如果页集已脱机很长时间（例如，某些月份），并且现在已决定将其用于其他用途，那么此选项很有用。

1. 使用带有 TYPE (NEW) 选项的 CSQUTIL 的 FORMAT 函数来格式化页集。
2. 将页集的定义添加到队列管理器的启动式任务过程和 CSQINP1 初始化数据集。
3. 重新启动队列管理器。

使用 TYPE (NEW) 选项进行格式化将清除页集的当前内容，并告知队列管理器忽略检查点中有关页集的任何历史信息。

使页集联机以避免介质恢复过程。

仅当您确定自队列管理器的干净关闭以来页集已脱机时，才使用此方法。当页集在短时间内处于脱机状态时，此选项最合适，这通常是由于操作问题（例如，在队列管理器启动时正在运行的备份）。

1. 使用带有 TYPE (REPLACE) 选项的 CSQUTIL 的 FORMAT 函数来格式化页集。
2. 使用带有 DSN 选项的 DEFINE PSID 命令将页集动态添加回队列管理器，或者允许在队列管理器重新启动时添加该页集。

使用 TYPE (REPLACE) 选项进行格式化将检查该页集是否已被队列管理器完全关闭，并将其标记为不会执行介质恢复。不会对页集的内容进行其他更改。

如何备份和恢复页集

有不同的机制可用于备份和恢复。使用本主题来了解这些机制。

本部分描述了以下主题：

- [第 334 页的『为非共享资源创建恢复点』](#)
- [第 335 页的『备份页集』](#)
- [第 336 页的『恢复页集』](#)
- [如何删除页集](#)

有关如何为共享资源创建恢复点的信息，请参阅 [第 341 页的『恢复共享队列』](#)。

为非共享资源创建恢复点

IBM MQ 可以将对象和非共享持久消息恢复到其当前状态，前提是：

1. 存在来自较早点的页集副本。
2. 所有 IBM MQ 日志都可用于从该点执行恢复。

这些资源表示非共享资源的恢复点。

对象和消息都保存在页集上。同一页集中可能存在来自不同队列的多个对象和消息。出于恢复目的，无法单独备份对象和消息，因此必须整体备份页集以确保正确恢复数据。

IBM MQ 恢复日志包含对对象进行的所有持久消息和更改的记录。如果 IBM MQ 失败 (例如，由于页集上的 I/O 错误)，那么可以通过复原备份副本并重新启动队列管理器来恢复页集。IBM MQ 将日志更改应用于从备份副本点开始的页集。

创建恢复点有两种方法：

完全备份

停止队列管理器，这将强制对页集进行所有更新。

这允许您从恢复点重新启动，仅使用备份的页集数据集以及从该点开始的日志。

模糊备份

获取页集的模糊备份副本，而不停止队列管理器。

如果使用此方法，并且关联的日志稍后会损坏或丢失，那么无法使用模糊页集备份副本进行恢复。这是因为模糊页集备份副本包含队列管理器状态的不一致视图，并且依赖于可用的日志。如果日志不可用，那么需要返回到子系统处于不活动状态时生成的最后一组备份页集副本 ([方法 1](#)) 并接受来自该时间的数据丢失。

方法 1: 完全备份

此方法涉及关闭队列管理器。这将强制对页集进行所有更新，以使页集处于一致状态。

1. 停止正在使用队列管理器的所有 IBM MQ 应用程序 (允许它们首先完成)。例如，可以通过更改访问安全性或队列设置来完成此操作。
2. 当所有活动都已完成时，显示并解决任何不确定的恢复单元。(使用命令 `DISPLAY CONN` 和 `RESOLVE INDOUBT`，如 [DISPLAY CONN](#) 和 [RESOLVE INDOUBT](#) 中所述。)

这将使页集处于一致状态；如果不执行此操作，那么您的页集可能不一致，并且您实际上正在执行模糊备份。

3. 发出 `ARCHIVE LOG` 命令以确保将最新的日志数据写出到日志数据集。
4. 发出 `STOP QMGR MODE (QUIESCE)` 命令。记录 `CSQI024I` 或 `CSQI025I` 消息中的最低 RBA 值 (请参阅 [CSQI024I](#) 和 [CSQI025I](#) 以获取更多信息)。您应该使日志数据集从 RBA 值指示的日志数据集开始，直到当前日志数据集为止。
5. 获取所有队列管理器页集的备份副本 (请参阅 [第 335 页的『备份页集』](#))。

方法 2: 模糊备份

此方法不涉及关闭队列管理器。因此，在备份过程中，更新可能位于虚拟存储器缓冲区中。这意味着页集未处于一致状态，只能用于使用日志进行恢复。

1. 发出 `DISPLAY USAGE TYPE (ALL)` 命令，并在 `CSQI024I` 或 `CSQI025I` 消息中记录 RBA 值 (请参阅 [CSQI024I](#) 和 [CSQI025I](#) 以获取更多信息)。
2. 生成页集的备份副本 (请参阅 [第 335 页的『备份页集』](#))。
3. 发出 `ARCHIVE LOG` 命令，以确保将最新的日志数据写出到日志数据集。要从恢复点重新启动，必须保持日志数据集从 RBA 值指示的日志数据集开始，直到当前日志数据集为止。

备份页集

要恢复页集，IBM MQ 需要知道日志中要返回的距离。IBM MQ 在每个页集的页零中维护一个日志 RBA 编号，称为恢复日志序号 (LSN)。此数字是日志中的起始 RBA，IBM MQ 可以从中恢复页集。当您备份页集时，也会复制此数字。

如果稍后使用副本来恢复页集，那么 IBM MQ 必须有权访问从此 RBA 值到当前 RBA 的所有日志记录。这意味着您必须保留足够的日志记录，以使 IBM MQ 能够从您打算保留的页集的最旧备份副本中恢复。

使用 `ADRDSSU COPY` 函数来复制页集。

有关更多信息，请参阅 [COPY DATASET Command 语法 for Logical Data Set](#) 文档。

例如：

```

//STEP2 EXEC PGM=ADRDSU,REGION=6M
//SYSPRINT DD SYSOUT=H
//SYSIN DD *
COPY -
DATASET(INCLUDE(SCENDATA.MQPA.PAGESET.*)) -
RENAMEU(SCENDATA.MQPA.PAGESET.** ,SCENDATA.MQPA.BACKUP1.** ) -
SPHERE -
REPUNC -
FASTREPLICATION(PREF )-
CANCELERROR -
TOL(ENQF)
/*
//

```

如果在队列管理器运行时复制页集，那么必须使用首先复制页集的页零的复制实用程序。如果不执行此操作，那么可能会损坏页集中的数据。

如果动态扩展页集的过程中断(例如，通过对正在丢失的系统的电源)，那么您仍可以使用 ADRDSU 来备份页集。

如果执行访问方法服务 IDCAMS LISTCAT ENT('page set data set name') ALLOC，那么您将看到 HI-ALLOC-RBA 高于 HI-USED-RBA。

下次此页集填满时，将再次扩展此页集(如果可能)，并且将使用高已用 RBA 和最高已分配 RBA 之间的页面以及另一个新的扩展数据块。

备份对象定义

您还应该备份对象定义的副本。为此，请使用 CSQUTIL COMMAND 函数的 MAKEDEF 功能(如 [向 IBM MQ \(COMMAND\) 发出命令](#) 中所述)。

每当您生成队列管理器的备份副本并保留最新版本时，请备份对象定义。

恢复页集

如果队列管理器由于故障而终止，那么通常可以在重新启动期间执行所有恢复的情况下重新启动队列管理器。但是，如果任何页集或日志数据集不可用，那么无法进行此类恢复。现在可以恢复的程度取决于页集和日志数据集的备份副本的可用性。

要从恢复点重新启动，您必须具有：

- 要恢复的页集的备份副本。
- 如果使用了第 335 页的『方法 2: 模糊备份』中描述的“模糊”备份过程，包括记录的 RBA 值的日志数据集，ARCHIVE LOG 命令生成的日志数据集以及这些日志数据集之间的所有日志数据集。
- 如果使用了完全备份，但没有遵循 ARCHIVE LOG 命令的日志数据集，那么不需要对所有页集运行 CSQUTIL 实用程序的 FORMAT TYPE (REPLACE) 函数。

要将页集恢复到其当前状态，还必须具有自 ARCHIVE LOG 命令以来的所有日志数据集和记录。

有两种方法可用于恢复页集。要使用任一方法，必须停止队列管理器。

简单恢复

这是比较简单的方法，适用于大多数恢复情况。

1. 删除要从备份复原的页集。
2. 使用 ADRDSU COPY 函数从备份副本中恢复页集。

或者，您可以将备份副本重命名为原始名称，或者将队列管理器过程中的 CSQP00xx DD 语句更改为指向备份页集。但是，如果丢失或损坏页集，那么将不再具有要从中复原的备份副本。

3. 重新启动队列管理器。
4. 队列管理器成功重新启动后，您可以重新启动应用程序

5. 恢复已复原页面的正常备份过程。

高级恢复

如果您具有要恢复的大页集，或者如果自上次备份副本以来页集上有许多活动，那么此方法将提供性能优势。但是，它需要比简单方法更多的手动干预，这可能会增加出错的风险和执行恢复所需的时间。

1. 删除并重新定义要从备份复原的页集。
2. 使用 ADRSSU 将页集的备份副本复制到新的页集中。使用辅助扩展数据块值定义新页集，以便可以动态扩展该页集。

或者，您可以将备份副本重命名为原始名称，或者将队列管理器过程中的 CSQP00xx DD 语句更改为指向备份页集。但是，如果丢失或损坏页集，那么将不再具有要从中复原的备份副本。
3. 更改队列管理器的 CSQINP1 定义，以使与正在恢复的页集相关联的缓冲池尽可能大。通过使缓冲池变大，您可能可以使所有已更改的页面都驻留在缓冲池中，并减少对页集的 I/O 量。
4. 重新启动队列管理器。
5. 当队列管理器成功重新启动时，停止它 (使用停顿)，然后使用该页集的正常缓冲池定义重新启动它。在此第二次重新启动成功完成后，您可以重新启动应用程序
6. 恢复已复原页面的正常备份过程。

重新启动队列管理器时发生的情况

重新启动队列管理器时，它将应用对日志中注册的页集所作的所有更改，从该页集的重启点开始。IBM MQ 可以通过此方式恢复多个页集。如果需要，将在介质恢复期间动态展开页集。

在重新启动期间，IBM MQ 通过获取以下值中的最小值来确定要从其开始的日志 RBA:

- 从每个页集的检查点日志记录恢复 LSN。
- 从每个页集中的页零恢复 LSN。
- 在执行备份时系统中最旧的不完整恢复单元的 RBA。

所有对象定义都存储在页集零上。消息可以存储在任何可用的页集上。

注: 如果页集 0 不可用，那么队列管理器无法重新启动。

如何删除页集

使用 DELETE PSID 命令删除页集; 请参阅 [DELETE PSID](#) 以获取此命令的详细信息。

无法删除仍由任何存储类引用的页集。使用 DISPLAY STGCLASS 来查找引用页集的存储类。

已从 IBM MQ 取消分配数据集，但未将其删除。它仍可供将来使用，或者可以使用 z/OS 工具将其删除。

从队列管理器的启动式任务过程中除去页集。

从 CSQINP1 初始化数据集中除去页集的定义。

如何使用 CSQUTIL 备份和复原队列

使用本主题作为参考，以获取有关使用 CSQUTIL 进行备份和复原的更多信息。

您可以使用 CSQUTIL 实用程序函数来备份和复原队列。要备份队列，请使用 COPY 或 SCOPY 函数将消息从队列复制到数据集。要恢复队列，请使用补充函数 LOAD 或 SLOAD。有关更多信息，请参阅 [IBM MQ 实用程序](#)。

管理缓冲池

如果要更改或删除缓冲池，请使用本主题。

本主题描述如何变更和删除缓冲池。它包含以下部分:

- [第 338 页的『如何更改缓冲池中的缓冲区数』](#)

- [第 338 页的『如何删除缓冲池』](#)

缓冲池是在队列管理器初始化期间使用从初始化输入数据集 CSQINP1 发出的 `DEFINE BUFFPOOL` 命令定义的。在队列管理器运行时，可以使用本主题中详细描述的流程来更改其属性以响应业务需求。队列管理器将当前缓冲池属性记录在检查点日志记录中。这些将在后续队列管理器重新启动时自动复原，除非 CSQINP1 中的缓冲池定义包含 `REPLACE` 属性。

使用 `DISPLAY USAGE` 命令可显示当前缓冲区属性。

您还可以使用带有 `DSN` 选项的 `DEFINE PSID` 命令动态定义缓冲池。

如果动态更改缓冲池，那么还应该在初始化数据集 CSQINP1 中更新其定义。

请参阅 [规划 z/OS](#)，以获取页集，存储类，缓冲区和缓冲池的描述以及适用的某些性能注意事项。

注：缓冲池使用大量存储器。当您增加缓冲池的大小或定义新的缓冲池时，请确保有足够的存储空间可用。有关更多信息，请参阅 [地址空间存储器](#)。

如何更改缓冲池中的缓冲区数

如果缓冲池太小，那么该条件可能会导致控制台上出现消息 `CSQP020E`，您可以使用 `ALTER BUFFPOOL` 命令为其分配更多缓冲区，如下所示：

1. 通过查看日志中的 `CSQY220I` 消息，确定可用于新缓冲区的空间量。可用空间以 MB 为单位进行报告。由于缓冲区大小为 4 KB，因此每 MB 可用空间允许您分配 256 个缓冲区。请勿将所有可用空间分配给缓冲区，因为其他任务需要一些可用空间。

如果缓冲池使用固定 4 KB 页面，即其 `PAGECLAS` 属性为 `FIXED4KB`，请确保 LPAR 上有足够的可用实存储器。

2. 如果报告的可用空间不足，请使用命令从另一个缓冲池释放一些缓冲区

```
ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
```

其中 `buf-pool-id` 是要从中回收空间的缓冲池，而 `integer` 是要分配给此缓冲池的新缓冲区数，必须小于分配给它的原始缓冲区数。

3. 使用命令将缓冲区添加到要扩展的缓冲池

```
ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
```

其中 `buf-pool-id` 是要扩展的缓冲池，而 `integer` 是要分配给此缓冲池的新缓冲区数，必须大于分配给它的原始缓冲区数。

如何删除缓冲池

当任何页集不再使用缓冲池时，请将其删除以释放分配给它的虚拟存储器。

使用 `DELETE BUFFPOOL` 命令删除缓冲池。如果任何页集正在使用此缓冲池，那么该命令将失败。

有关如何删除页集的信息，请参阅 [第 337 页的『如何删除页集』](#)。

在 z/OS 上管理队列共享组和共享队列

IBM MQ 可以使用不同类型的共享资源，例如队列共享组，共享队列和耦合设施。使用本主题来查看管理这些共享资源所需的过程。

本节包含有关以下主题的信息：

- [第 339 页的『管理队列共享组』](#)
- [第 341 页的『管理共享队列』](#)
- [第 345 页的『管理组对象』](#)
- [第 346 页的『管理耦合设施』](#)

您可以向队列共享组 (QSG) 添加或除去队列管理器，并管理关联的 Db2 表。

本主题包含有关以下任务的部分：

- [第 339 页的『设置队列共享组』](#)
- [第 339 页的『将队列管理器添加到队列共享组』](#)
- [第 340 页的『从队列共享组中除去队列管理器』](#)
- [第 341 页的『从 Db2 表中除去队列共享组』](#)
- [第 341 页的『验证 Db2 定义的一致性』](#)

设置队列共享组

每个队列共享组具有最多四个字符的名称。该名称在网络中必须是唯一的，并且必须与所有队列管理器名称不同。

请遵循以下步骤来设置队列共享组：

1. 如果这是第一个使用 Db2 数据共享组的队列共享组，请 [设置 Db2 环境](#)。
2. [设置耦合设施](#)。
3. 将队列共享组添加到 Db2 表。使用队列共享组实用程序 (CSQ5PQSG) 的 ADD QSG 函数。此程序在 [队列共享组实用程序](#) 中进行了描述。thlqual.SCSQPROC(CSQ45AQS) 中提供了样本。
4. 通过执行 [第 339 页的『将队列管理器添加到队列共享组』](#) 中的步骤将队列管理器添加到队列共享组
5. 通过执行 [第 346 页的『添加耦合设施结构』](#) 中的步骤，将应用程序结构定义到 IBM MQ。
6. 如果需要，请 [将非共享队列迁移到共享队列](#)。
7. 为了实现可用性，请创建进出队列共享组的共享通道。
 - 对于与队列共享组的连接：
 - 设置 VIPA 套接字或硬件路由器以在 QSG 中的可用队列管理器之间分配工作负载。
 - 使用 QSGDISP (GROUP) 定义接收方通道，以确保通道定义在 QSG 中的所有队列管理器上都可用。
 - 在每个队列管理器上使用 INDISP (GROUP) 启动侦听器，以用于与 QSG 的 MCA 通道连接。到 QSG 的客户机连接仍应连接到使用 INDISP (QMGR) 启动的侦听器。
 - 更改应用程序以使用 QSG 名称 (而不是特定队列管理器名称) 进行连接。
 - 确保 QSG 中所有队列管理器上的通道认证规则相同，以允许应用程序连接到 QSG 中的任何队列管理器。
 - 对于队列共享组外的连接：
 - 定义共享传输队列。
 - 使用 QSGDISP (GROUP) 和 DEFCDISP (SHARED) 定义出站通道。

如果将现有通道转换为共享通道，那么在启动通道之前可能需要发出 [RESET CHANNEL](#) 命令，因为通道所使用的同步队列已更改。

将队列管理器添加到队列共享组

可以将队列管理器添加到现有队列共享组。

注意：

- 队列共享组必须存在，然后才能向其添加队列管理器。
- 队列管理器只能是一个队列共享组的成员。

遵循以下步骤将队列管理器添加到队列共享组：

1. 执行 [实现队列共享组的 ESM 安全控制](#) 中的任务，以授予对队列管理器和通道启动程序用户标识的相应访问权。
2. 如果队列共享组具有配置为将数据卸载到 SMDS 的 CF 结构，请执行 [设置 SMDS 环境中的任务](#)。
3. 停止队列管理器。
4. 使用队列共享组实用程序 (CSQ5PQSG) 的 ADD QMGR 函数。此程序在 [队列共享组实用程序中描述](#)。thlqual.SCSQPROC(CSQ45AQM) 中提供了样本。
5. [更改系统参数模块](#) 以添加队列共享组数据：
 - a. 修改 CSQ6SYSP 以指定 QSGDATA 参数。请参阅 [使用 CSQ6SYSP](#) 以获取更多信息。
 - b. 组装并链接系统参数模块。您可能希望对装入模块使用其他名称。
 - c. 更改启动过程以使用新模块。
6. 复制并定制样本成员 thlqual.SCSQPROC(CSQ4INSS)，这将定义必需的 CF 结构和 SYSTEM 队列。将定制成员添加到队列管理器启动 JCL 中的 CSQINP2 DD。
7. 使用队列共享组系统参数模块重新启动队列管理器。
8. (可选) 迁移到以队列共享组名 (而不是队列管理器名称) 作为前缀的安全概要文件。
9. 如果共享通道用于与 QSG 的连接，请创建通道认证规则，以镜像 QSG 中其他队列管理器上的通道认证规则，从而允许应用程序连接到 QSG 中的任何队列管理器。
10. (可选) 执行以下任一操作以允许连接到 QSG 中队列管理器的应用程序将消息放入 QSG 中其他队列管理器托管的队列：
 - 通过发出命令 ALTER QMGR IGQ (ENABLED) 来开启 [组内排队](#)。
 - 定义到 QSG 中其他队列管理器的传输队列和通道。定义与目标队列管理器同名的传输队列可避免定义远程队列和队列管理器别名。

注：要将队列管理器添加到包含运行较早版本的 IBM MQ 的队列管理器的现有队列共享组，必须首先将该组中最高版本的 IBM MQ 的共存 PTF 应用于该组中的每个较早版本的队列管理器。

从队列共享组中除去队列管理器

仅当另一进程不需要队列管理器的日志，并且队列管理器拥有的所有 SMDS 都为空时，才能从队列共享组中除去队列管理器。

如果日志包含以下内容，那么需要这些日志：

- 队列共享组使用的其中一个耦合设施 (CF) 应用程序结构的最新备份
- 将来的复原过程所需的数据，即，自上次备份排除时间间隔值描述的时间以来，队列管理器已使用可恢复结构。

如果其中一个或两个点适用，或者队列管理器拥有的 SMDS 包含消息，那么无法除去队列管理器。要确定未来复原过程需要哪些队列管理器日志，请使用带有 TYPE (BACKUP) 选项的 MQSC DISPLAY CFSTATUS 命令 (有关此命令的详细信息，请参阅 [DISPLAY CFSTATUS](#))。

使用以下步骤从队列共享组中除去队列管理器：

1. 停止连接到将消息放入共享队列的队列管理器的任何应用程序。
2. 解析涉及此队列管理器的任何不确定工作单元。
3. 通过发出命令 DISPLAY USAGE TYPE (SMDS) 来确定队列管理器拥有的任何 SMDS 中是否有任何消息。
4. 如果有任何应用程序结构的卸载消息，请等待直到从队列中检索到这些消息。在继续之前，DISPLAY USAGE TYPE (SMDS) 报告的卸载消息数应该为零。
5. 使用 STOP QMGR MODE (QUIESCE) 彻底关闭队列管理器。
6. 等待时间间隔至少相当于您将在下一步的 BACKUP CFSTRUCT 命令中指定的 EXCLINT 参数的值。
7. 在另一个队列管理器上，通过使用 MQSC BACKUP CFSTRUCT 命令并根据上一步中的要求指定 EXCLINT 值，对每个可恢复的 CF 结构运行 CF 结构备份。

8. 通过检查命令 DISPLAY CFSTATUS (*) TYPE (BACKUP) 的输出，确认不需要队列管理器的日志来复原任何 CF 结构。
9. 使用 CSQ5PQSG 实用程序的 REMOVE QMGR 函数从队列共享组中除去队列管理器。此程序在 [队列共享组实用程序](#) 中进行了描述。thlqual.SCSQPROC(CSQ45RQM) 中提供了样本。
10. 在重新启动队列管理器之前，请将 QSGDATA 系统参数重置为其缺省值，然后重新创建系统参数模块。有关如何定制系统参数的信息，请参阅 [使用 CSQ6SYSP](#)。

请注意，在除去队列共享组中的最后一个队列管理器时，必须使用 FORCE 选项，而不是 REMOVE。这将从队列共享组中除去队列管理器，而不执行恢复所需的队列管理器日志的一致性检查。仅当要删除队列共享组时，才应执行此操作。

从 Db2 表中除去队列共享组

要从 Db2 表中除去队列共享组，请使用队列共享组实用程序的 REMOVE QSG 函数 (CSQ5PQSG)。此程序在 [队列共享组实用程序](#) 中进行了描述。thlqual.SCSQPROC(CSQ45RQS) 中提供了样本。

仅当从队列共享组中除去了所有队列管理器 (如第 340 页的『[从队列共享组中除去队列管理器](#)』中所述) 之后，才能从公共 Db2 数据共享组表中除去队列共享组。

从队列共享组管理表中删除队列共享组记录时，将从其他 IBM MQ Db2 表中删除与该队列共享组相关的所有对象和管理信息。这包括共享队列和组对象信息。

验证 Db2 定义的一致性

如果 Db2 对象定义由于任何原因而变得不一致，那么可能会发生队列共享组中共享队列的问题。

要验证队列管理器，CF 结构和共享队列的 Db2 对象定义的一致性，请使用队列共享组实用程序 (CSQ5PQSG) 的 VERIFY QSG 函数。此程序在 [队列共享组实用程序](#) 中进行了描述。

管理共享队列

使用本主题来了解如何恢复，移动和迁移共享队列。

本部分描述了下列任务：

- [第 341 页的『恢复共享队列』](#)
- [第 342 页的『移动共享队列』](#)
- [第 344 页的『将非共享队列迁移到共享队列』](#)
- [暂挂 Db2 连接](#)

恢复共享队列

IBM MQ 可以恢复共享队列上的持久消息，前提是：

- 已执行包含消息的 CF 结构的备份。
- 队列共享组中所有队列管理器的所有日志都可用，以便从执行备份的位置执行恢复。
- Db2 可用，并且结构备份表是最新的 CF 结构备份表。

共享队列上的消息存储在耦合设施 (CF) 结构中。持久消息可以放在共享队列上，并且与非共享队列上的持久消息一样，会将它们复制到队列管理器日志中。提供了 MQSC BACKUP CFSTRUCT 和 RECOVER CFSTRUCT 命令，以允许在不太可能发生耦合设施故障的情况下恢复 CF 结构。在这种情况下，存储在受影响结构中的任何非持久消息都将丢失，但持久消息可以恢复。在恢复结构之前，将阻止使用该结构的任何其他应用程序活动。

要启用恢复，必须使用 MQSC BACKUP CFSTRUCT 命令频繁备份耦合设施列表结构。CF 结构中的消息将写入进行备份的队列管理器的活动日志数据集。它将备份记录写入 Db2: 正在备份的 CF 结构的名称，执行备份的队列管理器的名称，该队列管理器日志上此备份的 RBA 范围以及备份时间。备份 CF 列表结构，即使您未主动使用共享队列也是如此，例如，如果您已设置队列共享组，打算在将来使用该队列共享组。

可以通过向可执行恢复的队列管理器发出 MQSC RECOVER CFSTRUCT 命令来恢复 CF 结构; 可以使用队列共享组中的任何队列管理器。您可以指定只恢复单一 CF 结构, 也可以同时恢复多个 CF 结构。

如前所述, 频繁备份 CF 列表结构很重要, 否则恢复 CF 结构可能需要很长时间。此外, 无法取消恢复过程。

共享队列的定义保留在 Db2 数据库中, 因此可以根据需要使用标准 Db2 数据库过程进行恢复。请参阅[共享队列和队列共享组](#)以获取更多信息。

移动共享队列

本节描述如何通过将共享队列从一个耦合设施结构移动到另一个耦合设施结构来执行负载均衡。它还描述了如何将非共享队列移至共享队列, 以及如何将共享队列移至非共享队列。

移动队列时, 需要在过程中定义临时队列。这是因为每个队列都必须具有唯一的名称, 因此不能有两个同名的队列, 即使这些队列具有不同的队列处置。IBM MQ 允许有两个同名的队列 (如步骤 [第 342 页的『2』](#) 中所示), 但不能使用这些队列。

- 将队列从一个耦合设施结构移动到另一个耦合设施结构
- 将非共享队列移至共享队列
- 将共享队列移至非共享队列

将队列从一个耦合设施结构移动到另一个耦合设施结构

要将队列及其消息从一个 CF 结构移至另一个 CF 结构, 请使用 MQSC [MOVE QLOCAL](#) 命令。确定要移动到新 CF 结构的一个或多个队列后, 请使用以下过程来移动每个队列:

1. 确保要移动的队列未被任何应用程序使用, 即, 队列共享组中的所有队列管理器上的队列属性 IPROCS 和 OPROCS 为零。
2. 通过更改队列定义以禁用 MQPUT, 防止应用程序将消息放入要移动的队列中。将队列定义更改为 PUT (DISABLED)。
3. 使用以下命令定义与要移动的队列具有相同属性的临时队列:

```
DEFINE QL(TEMP_QUEUE) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
```

注: 如果此临时队列存在于先前运行中, 请在执行定义之前将其删除。

4. 使用以下命令将消息移至临时队列:

```
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_QUEUE)
```

5. 使用以下命令删除正在移动的队列:

```
DELETE QLOCAL(Queue_To_Move)
```

6. 使用以下命令重新定义要移动的队列, 更改 CFSTRUCT 属性:

```
DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
```

重新定义队列时, 它基于步骤 [第 342 页的『3』](#) 中创建的临时队列。

7. 使用以下命令将消息移回新队列:

```
MOVE QLOCAL(TEMP) TOQLOCAL(Queue_To_Move)
```

8. 不再需要步骤 第 342 页的『3』中创建的队列。使用以下命令将其删除:

```
DELETE QL(TEMP_Queue)
```

9. 如果要移动的队列是在 CSQINP2 数据集中定义的, 请在 CSQINP2 数据集中更改相应 DEFINE QLOCAL 命令的 CFSTRUCT 属性。添加 REPLACE 关键字, 以便替换现有队列定义。

第 343 页的图 47 显示了用于将队列从一个 CF 结构移动到另一个 CF 结构的样本作业。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=thlqua1.SCSQANLE,DISP=SHR
//      DD DSN=thlqua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
ALTER QL(Queue_To_Move) PUT(DISABLED)
DELETE QL(TEMP_Queue) PURGE
DEFINE QL(TEMP_Queue) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_Queue)
DELETE QL(Queue_To_Move)
DEFINE QL(Queue_To_Move) LIKE(TEMP_Queue) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_Queue) TOQLOCAL(Queue_To_Move)
DELETE QL(TEMP_Queue)
/*
```

图 47: 用于将队列从一个 CF 结构移动到另一个 CF 结构的样本作业

将非共享队列移至共享队列

将非共享队列移动到共享队列的过程类似于将队列从一个 CF 结构移动到另一个 CF 结构的过程 (请参阅第 342 页的『将队列从一个耦合设施结构移动到另一个耦合设施结构』)。第 343 页的图 48 提供了用于执行此操作的样本作业。

注: 请记住, 共享队列上的消息在最大消息大小, 消息持久性和队列索引类型方面受到某些限制, 因此您可能无法将某些非共享队列移至共享队列。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=thlqua1.SCSQANLE,DISP=SHR
//      DD DSN=thlqua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
ALTER QL(Queue_To_Move) PUT(DISABLED)
DELETE QL(TEMP_Queue) PURGE
DEFINE QL(TEMP_Queue) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED)
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_Queue)
DELETE QL(Queue_To_Move)
DEFINE QL(Queue_To_Move) LIKE(TEMP_Queue) CFSTRUCT(NEW) QSGDISP(SHARED)
MOVE QLOCAL(TEMP_Queue) TOQLOCAL(Queue_To_Move)
DELETE QL(TEMP_Queue)
/*
```

图 48: 用于将非共享队列移动到共享队列的样本作业

将共享队列移至非共享队列

将共享队列移动到非共享队列的过程类似于将队列从一个 CF 结构移动到另一个 CF 结构的过程 (请参阅第 342 页的『将队列从一个耦合设施结构移动到另一个耦合设施结构』)。

第 344 页的图 49 提供了用于执行此操作的样本作业。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=thlqua1.SCSQANLE,DISP=SHR
// DD DSN=thlqua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
ALTER QL(Queue_To_Move) PUT(DISABLED)
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(Queue_To_Move) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
MOVE QLOCAL(Queue_To_Move) TOQLOCAL(TEMP_QUEUE)
DELETE QL(Queue_To_Move)
DEFINE QL(Queue_To_Move) LIKE(TEMP_QUEUE) STGCLASS(NEW) QSGDISP(QMGR)
MOVE QLOCAL(TEMP_QUEUE) TOQLOCAL(Queue_To_Move)
DELETE QL(TEMP_QUEUE)
/*
```

图 49: 用于将共享队列移至非共享队列的样本作业

将非共享队列迁移到共享队列

将非共享队列迁移到共享队列有两个阶段:

- 迁移队列共享组中的第一个 (或唯一) 队列管理器
- 迁移队列共享组中的任何其他队列管理器

迁移队列共享组中的第一个 (或唯一) 队列管理器

第 343 页的图 48 显示了用于将非共享队列移至共享队列的示例作业。对需要迁移的每个队列执行此操作。

注:

1. 共享队列上的消息在最大消息大小, 消息持久性和队列索引类型方面受到某些限制, 因此您可能无法将某些非共享队列移至共享队列。
2. 必须对共享队列使用正确的索引类型。如果将传输队列迁移为共享队列, 那么索引类型必须为 MSGID。

如果队列为空, 或者您不需要保留其上的消息, 那么迁移队列会更简单。第 345 页的图 50 显示了要在这些情况下使用的示例作业。


```

//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD DSN=th1qua1.SCSQANLE,DISP=SHR
// DD DSN=th1qua1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
COMMAND DDNAME(MOVEQ)
/*
//MOVEQ DD *
DELETE QL(TEMP_QUEUE) PURGE
DEFINE QL(TEMP_QUEUE) LIKE(Queue_TO_MOVE) PUT(ENABLED) GET(ENABLED)
DELETE QL(Queue_TO_MOVE)
DEFINE QL(Queue_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
DELETE QL(TEMP_QUEUE)
/*

```

图 50: 用于将无消息的非共享队列移动到共享队列的样本作业

迁移队列共享组中的任何其他队列管理器

1. 对于与现有共享队列不具有相同名称的每个队列，请按 [第 343 页的图 48](#) 或 [第 345 页的图 50](#) 中所述移动该队列。
2. 对于与现有共享队列同名的队列，请使用 [第 345 页的图 51](#) 中显示的命令将消息移至共享队列。

```

MOVE QLOCAL(Queue_TO_MOVE) QSGDISP(QMGR) TOQLOCAL(Queue_TO_MOVE)
DELETE QLOCAL(Queue_TO_MOVE) QSGDISP(QMGR)

```

图 51: 将消息从非共享队列移至现有共享队列

暂挂与 Db2 的连接

如果要将维护或服务应用于与共享队列相关的 Db2 表或包，而不停止队列管理器，那么必须临时断开数据共享组 (DSG) 中的队列管理器与 Db2 的连接。

要执行此操作：

1. 使用 MQSC 命令 [SUSPEND QMGR FACILITY \(Db2\)](#)。
2. 执行绑定。
3. 使用 MQSC 命令 [RESUME QMGR FACILITY \(Db2\)](#) 重新连接到 Db2

请注意，这些命令的使用存在限制。



注意：暂挂 Db2 连接时，以下操作将不可用。因此，您需要在企业最忙碌的时候执行此工作。

- 访问共享队列对象以进行管理 (定义，删除和变更)
- 启动共享通道
- 在 Db2 中存储消息
- 备份或恢复 CFSTRUCT

管理组对象

使用本主题来了解如何使用组对象。

IBM MQ 会自动将组对象的定义复制到使用该组对象的每个队列管理器的页集零。您可以临时变更定义的副本，并且 IBM MQ 允许您从存储库副本刷新页集副本。IBM MQ 始终尝试在启动时从存储库副本刷新页集副本 (对于通道对象，这是在通道启动程序重新启动时完成的)。这将确保页集副本反映存储库上的版本，包括在队列管理器处于不活动状态时进行的任何更改。

存在未执行刷新的情况，例如：

- 如果打开了队列的副本，那么将更改队列使用情况的刷新将失败。
- 如果队列的副本包含消息，那么删除该队列的刷新将失败。

在这些情况下，不会对该副本执行刷新，而是对所有其他队列管理器上的副本执行刷新。在添加，更改或删除组对象之后，以及在队列管理器或通道启动程序重新启动时，检查并更正复制对象的任何问题。

z/OS 管理耦合设施

使用本主题来了解如何添加或移除耦合设施 (CF) 结构。

本部分描述了下列任务：

- [第 346 页的『添加耦合设施结构』](#)
- [第 346 页的『除去耦合设施结构』](#)

添加耦合设施结构

要添加耦合设施结构，请使用以下过程：

1. 在 CFRM 策略数据集中定义 CF 结构。有关在 [设置耦合设施](#) 中设置耦合设施的信息描述了用于命名耦合设施结构的规则，以及如何在 CFRM 策略数据集中定义结构。
2. 如果要配置用于将消息数据卸载到 SMDS 的结构，请分配并预先格式化数据集。请参阅 [创建共享消息数据集](#) 以获取详细信息。
3. 使用 [DEFINE CFSTRUCT](#) 命令定义 IBM MQ 的结构。

除去耦合设施结构

要除去耦合设施结构，请使用以下过程：

1. 使用以下命令以获取使用要删除的耦合设施结构的所有队列的列表：

```
DISPLAY QUEUE(*) QSGDISP(SHARED) CFSTRUCT(structure-name)
```

2. 删除所有使用该结构的队列。
3. 使用 [DELETE CFSTRUCT](#) 命令从 IBM MQ 中删除 CF 结构。
4. 如果结构配置为将消息数据卸载到 SMDS，请删除 SMDS。
5. 从 CFRM 策略数据集中除去结构定义，然后运行 [IXCMIAPU](#) 实用程序。（这是设置耦合设施的定制任务的反面，如 [设置耦合设施](#) 中所述。）

z/OS 调整耦合设施列表监视

使用本主题来了解耦合设施列表监视

耦合设施 (CF) 列表监视用于监视包含 IBM MQ 共享队列的列表结构的状态。将消息添加到共享队列时，如果队列深度从零过渡到非零，那么 CF 将通知队列共享组中的所有队列管理器。当收到通知时，队列管理器可能会执行许多操作，包括通知正在使用 TRIGGER (FIRST) 的触发器监视器或正在执行 GET-WAIT 的应用程序。

缺省情况下，CF 会同时通知队列共享组中的所有队列管理器。在某些配置中，这可能会导致问题，例如：

- 偏差的工作负载分布，其中很大百分比的消息会转至队列共享组中的特定队列管理器，通常是在最快的 LPAR 上运行的队列管理器，或者最接近 CF 的队列管理器，或者
- 大量失败获取，导致 CPU 时间浪费。

z/OS V2R3 引入了名为 **KEYRNOTIFYDELAY** 的新耦合设施资源管理器 (CFRM) 属性，该属性可用于包含共享队列 (即，应用程序结构，而不是管理结构) 的列表结构，并且可以针对某些工作负载将工作负载偏差和空 MQGET 调用或空 MQGET 调用的影响降至最低。

只能对在 CFLEVEL 22 或更高版本上运行的 CF 中的结构设置 **KEYRNOTIFYDELAY**。

其值必须是 1 到 7 位十进制数字，范围从 0 到 1,000,000 微秒。如果设置为非零值，并且队列深度从零过渡到非零，那么 CF 将从队列共享组中选择单个队列管理器，并在该组中的所有其他队列管理器之前通知该队列管理器。

以循环方式选择队列管理器。如果所选队列管理器未在 **KEYRNOTIFYDELAY** 描述的时间间隔内处理消息，那么还将通知队列共享组中的所有其他队列管理器。

此处提供了有关 **KEYRNOTIFYDELAY** 的更多信息：[了解 Keyrange Monitoring Notification Delay](#)。

请注意，有两个类似的 CFRM 属性称为 **LISTNOTIFYDELAY** 和 **SUBNOTIFYDELAY**。这两者都不会对 IBM MQ 工作负载产生任何可测量的影响。

z/OS 在 z/OS 上恢复并重新启动

使用本主题来了解 IBM MQ 使用的恢复和重新启动机制。

z/OS 重新启动 IBM MQ

在队列管理器终止后，根据队列管理器终止的方式，需要不同的重新启动过程。使用本主题来了解可以使用的不同重新启动过程。

本主题包含有关在以下情况下如何重新启动队列管理器的信息：

- [第 347 页的『正常关闭后重新启动』](#)
- [第 347 页的『异常终止后重新启动』](#)
- [第 347 页的『如果您丢失了页集，请重新启动』](#)
- [第 348 页的『在丢失日志数据集时重新启动』](#)
- [如果您丢失了 CF 结构，请重新启动](#)

正常关闭后重新启动

如果使用 STOP QMGR 命令停止了队列管理器，那么系统将以有序方式完成其工作，并在停止之前采用终止检查点。重新启动队列管理器时，它将使用系统检查点和恢复日志中的信息来确定关闭时的系统状态。

要重新启动队列管理器，请发出 START QMGR 命令，如 [第 288 页的『在 z/OS 上启动和停止队列管理器』](#) 中所述。

异常终止后重新启动

IBM MQ 会自动检测重新启动是遵循正常关闭还是异常终止。

异常终止后启动队列管理器与发出 STOP QMGR 命令后启动队列管理器不同。如果队列管理器异常终止，那么它将在无法完成其工作或采用终止检查点的情况下终止。

要重新启动队列管理器，请发出 START QMGR 命令，如 [第 288 页的『在 z/OS 上启动和停止队列管理器』](#) 中所述。在异常终止后重新启动队列管理器时，它会使用日志中的信息刷新其在终止时的状态知识，并通知您各种任务的状态。

通常，重新启动过程会解决所有不一致的状态。但是，在某些情况下，您必须采取具体步骤来解决不一致问题。在 [第 358 页的『手动恢复工作单元』](#) 中对此进行了描述。

如果您丢失了页集，请重新启动

如果丢失了页集，那么需要先从备份副本复原这些页集，然后才能重新启动队列管理器。在 [第 334 页的『如何备份和恢复页集』](#) 中对此进行了描述。

在这些情况下，由于介质恢复所需的时间长度，队列管理器可能需要很长时间才能重新启动。

在丢失日志数据集时重新启动

如果在停止队列管理器 (使用 STOP QMGR 命令) 之后日志的两个副本都丢失或损坏，那么可以重新启动队列管理器，前提是您具有一组一致的页集 (使用 [方法 1: 完全备份](#) 生成)。

请遵循以下过程：

1. 定义新的页集以对应于队列管理器中的每个现有页集。有关页集定义的信息，请参阅 [任务 15: 定义页集](#)。
确保每个新页集都大于相应的源页集。
2. 使用 CSQUTIL 的 FORMAT 函数来格式化目标页集。请参阅 [格式化页面集](#) 以获取更多详细信息。
3. 使用 CSQUTIL 的 RESETPAGE 函数来复制现有页集或将其重置到位，并重置每个页中的日志 RBA。有关此功能的更多信息，请参阅 [复制页集并重置日志](#)。
4. 使用 CSQJU003 重新定义队列管理器日志数据集和 BSDS (请参阅 [更改日志库存实用程序](#))。
5. 使用新页集重新启动队列管理器。要执行此操作，请执行下列其中一项操作：
 - 更改队列管理器启动式任务过程以引用新的页集。请参阅 [任务 6: 为 IBM MQ 队列管理器 创建过程](#) 以获取更多信息。
 - 使用 "访问方法服务" 来删除旧页集，然后重命名新页集，为它们提供与旧页集相同的名称。

注意：在删除任何 IBM MQ 页集之前，请确保已生成所需的备份副本。

如果队列管理器是队列共享组的成员，那么 GROUP 和 SHARED 对象定义通常不受丢失或损坏的日志影响。但是，如果丢失或损坏的日志覆盖的工作单元中涉及任何共享队列消息，那么对此类未落实消息的影响不可预测。

注：如果日志损坏并且队列管理器是队列共享组的成员，那么恢复共享持久消息的能力可能会丢失。对具有 RECOVER (YES) 属性的所有 CF 结构的队列共享组中的另一个活动队列管理器立即发出 BACKUP CFSTRUCT 命令。

如果您丢失了 CF 结构，请重新启动

如果丢失 CF 结构，那么无需重新启动，因为队列管理器不会终止。

z/OS 上的备用站点恢复

您可以恢复单个队列管理器或队列共享组，或者考虑磁盘镜像。

请参阅以下部分以获取更多详细信息：

- [在备用站点上恢复单个队列管理器](#)
- [恢复队列共享组](#)。
 - [CF 结构介质恢复](#)
 - [备份主站点上的队列共享组](#)
 - [在备用站点上恢复队列共享组](#)
- [使用磁盘镜像](#)

在备用站点上恢复单个队列管理器

如果 IBM MQ 计算中心发生完全丢失，那么可以在恢复站点上的另一个队列管理器或队列共享组上进行恢复。(请参阅 [第 351 页的『在备用站点恢复队列共享组』](#) 以了解队列共享组的备用站点恢复过程。)

要在恢复站点的另一个队列管理器上恢复，必须定期备份页集和日志。与所有数据恢复操作一样，灾难恢复的目标是尽可能少地丢失数据，工作负载处理(更新)和时间。

在恢复站点:

- 恢复队列管理器 **必须** 具有与丢失的队列管理器相同的名称。
- 在每个恢复队列管理器上使用的系统参数模块(例如 CSQZPARM) 必须包含与相应的丢失队列管理器相同的参数。

完成此操作后，请重新建立所有队列管理器，如以下过程中所述。这可用于在单个队列管理器的恢复站点上执行灾难恢复。它假定可用的所有内容都是:

- 在主站点正常运行所创建的归档日志和 BSD 的副本(活动日志将与主站点上的队列管理器一起丢失)。
- 来自主站点的队列管理器的页集副本，这些页集的存在时间与可用的最新归档日志副本相同或更长。

您可以对活动日志和归档日志使用双重日志记录，在这种情况下，需要将 BSDS 更新应用于两个副本:

1. 定义新的页集数据集，并将其与来自主站点的页集副本中的数据一起装入。
2. 定义新的活动日志数据集。
3. 定义新的 BSDS 数据集，并使用 Access Method Services REPRO 将最新归档的 BSDS 复制到其中。
4. 使用打印日志映射实用程序 CSQJU004 来打印来自此最新 BSDS 的信息。在归档此 BSDS 时，您最近归档的日志将被截断为活动日志，而不会显示为归档日志。记录此日志的 STARTRBA 和 ENDRBA。
5. 使用更改日志库存实用程序 CSQJU003，通过使用步骤第 349 页的『4』中记录的 STARTRBA 和 ENDRBA，在刚刚复原的 BSDS 中注册此最新归档日志数据集。
6. 使用 DELETE 选项 CSQJU003 从 BSDS 中除去所有活动日志信息。
7. 使用 CSQJU003 的 NEWLOG 选项将活动日志添加到 BSDS，请勿指定 STARTRBA 或 ENDRBA。
8. 使用 CSQJU003 将重新启动控制记录添加到 BSDS。指定 CRESTART CREATE, ENDRBA=highrba，其中 highrba 是最新的可用归档日志的高 RBA (在步骤第 349 页的『4』中找到) 以及 1。

BSDS 现在将所有活动日志描述为空，所有已归档日志都可用，并且在日志结束后没有检查点。

9. 使用 START QMGR 命令重新启动队列管理器。在初始化期间，将发出类似如下的操作员应答消息:

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.  
REPLY Y TO CONTINUE, N TO CANCEL
```

输入 Y 以启动队列管理器。队列管理器启动，并将数据恢复到 CRESTART 语句中指定的 ENDRBA。

有关使用 CSQJU003 和 CSQJU004 的信息，请参阅 [使用 IBM MQ 实用程序](#)。

以下示例显示步骤 6，7 和 8 的 CSQJU003 的样本输入语句:

```
* Step 6  
DELETE DSNAME=MQM2.LOGCOPY1.DS01  
DELETE DSNAME=MQM2.LOGCOPY1.DS02  
DELETE DSNAME=MQM2.LOGCOPY1.DS03  
DELETE DSNAME=MQM2.LOGCOPY1.DS04  
DELETE DSNAME=MQM2.LOGCOPY2.DS01  
DELETE DSNAME=MQM2.LOGCOPY2.DS02  
DELETE DSNAME=MQM2.LOGCOPY2.DS03  
DELETE DSNAME=MQM2.LOGCOPY2.DS04  
  
* Step 7  
NEWLOG DSNAME=MQM2.LOGCOPY1.DS01,COPY1  
NEWLOG DSNAME=MQM2.LOGCOPY1.DS02,COPY1  
NEWLOG DSNAME=MQM2.LOGCOPY1.DS03,COPY1  
NEWLOG DSNAME=MQM2.LOGCOPY1.DS04,COPY1  
NEWLOG DSNAME=MQM2.LOGCOPY2.DS01,COPY2  
NEWLOG DSNAME=MQM2.LOGCOPY2.DS02,COPY2  
NEWLOG DSNAME=MQM2.LOGCOPY2.DS03,COPY2  
NEWLOG DSNAME=MQM2.LOGCOPY2.DS04,COPY2  
  
* Step 8  
CRESTART CREATE, ENDRBA=063000
```

在恢复站点上重新启动通道启动程序时需要考虑的事项与使用 ARM 在其他 z/OS 映像上重新启动通道启动程序时所面临的事项相同。请参阅第 356 页的『在 IBM MQ 网络中使用 ARM』以获取更多信息。恢复策略还应涵盖 IBM MQ 产品库的恢复以及使用 IBM MQ 的应用程序编程环境 (例如 CICS)。

更改日志库存实用程序 (CSQJU003) 的其他功能也可用于灾难恢复方案。HIGHRBA 函数允许更新引导数据集中写入的最高 RBA 值和卸载的最高 RBA 值。CHECKPT 函数允许添加新的检查点队列记录或删除 BSDS 中的现有检查点队列记录。

注意: 这些函数可能会影响 IBM MQ 数据的完整性。 仅在 IBM 服务人员的指导下在灾难恢复方案中使用。

快速复制技术

如果在队列管理器冻结时生成所有页集和日志的副本, 那么这些副本将是可用于在备用站点上重新启动队列管理器的一致集。它们通常支持以更快的速度重新启动队列管理器, 因为要执行的介质恢复很少。

使用 SUSPEND QMGR LOG 命令来冻结队列管理器。此命令将缓冲池清空到页集, 采用检查点, 并停止任何进一步的日志写入活动。暂挂日志写入活动后, 将有效冻结队列管理器, 直到您发出 RESUME QMGR LOG 命令为止。在冻结队列管理器时, 可以复制页集和日志。

通过使用复制工具 (例如 FLASHCOPY 或 SNAPSHOT) 来快速复制页集和日志, 可以将冻结队列管理器的时间减少到最小。

但是, 在队列共享组中, SUSPEND QMGR LOG 命令可能不是一个很好的解决方案。要生效, 日志的副本必须全部包含相同的恢复时间点, 这意味着必须在队列共享组中的所有队列管理器上同时发出 SUSPEND QMGR LOG 命令, 因此整个队列共享组将冻结一段时间。

恢复队列共享组

发生主站点灾难时, 可以使用主站点中的备份数据集在远程站点上重新启动队列共享组。要恢复队列共享组, 您需要在队列共享组中的所有队列管理器之间协调恢复, 并与其他资源 (主要是 Db2) 进行协调。本部分详细描述了这些任务。

- [CF 结构介质恢复](#)
- [备份主站点上的队列共享组](#)
- [在备用站点上恢复队列共享组](#)

CF 结构介质恢复

用于在共享队列上保存持久消息的 CF 结构的介质恢复依赖于具有可由已记录更新的应用程序进行转发恢复的介质的备份。使用 MQSC BACKUP CFSTRUCT 命令定期备份 CF 结构。对共享队列 (MQGET 和 MQPUT) 的所有更新都将写入执行更新的队列管理器的日志中。要执行 CF 结构的介质恢复, 必须已从使用该 CF 结构的所有队列管理器的日志将记录的更新应用于该备份。使用 MQSC RECOVER CFSTRUCT 命令时, IBM MQ 会自动合并来自相关队列管理器的日志, 并将更新应用于最新备份。

CF 结构备份将写入处理 BACKUP CFSTRUCT 命令的队列管理器的日志, 因此没有要收集并传输到备用站点的其他数据集。

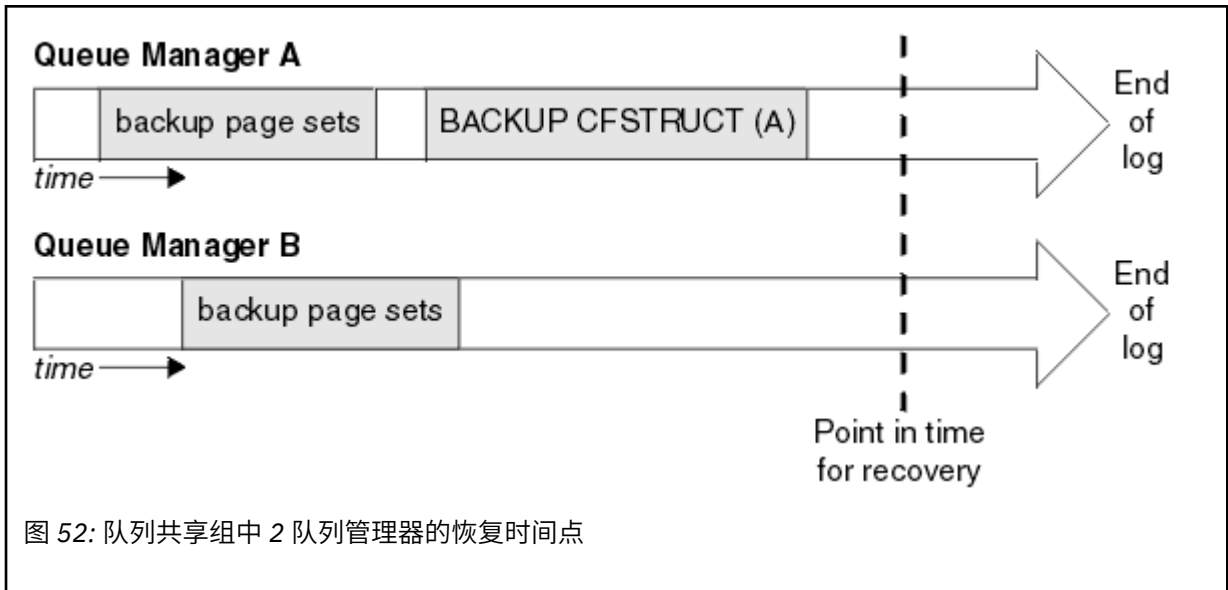
备份主站点上的队列共享组

在主站点, 您需要定期建立一组一致的备份, 这些备份可在发生灾难时用于在备用站点重建队列共享组。对于单个队列管理器, 恢复可以是任意时间点, 通常是远程站点上可用的日志末尾。但是, 如果持久消息已存储在共享队列上, 那么必须合并队列共享组中所有队列管理器的日志以恢复共享队列, 因为队列共享组中的任何队列管理器可能已在队列上执行更新 (MQPUT 或 MQGET)。

要恢复队列共享组, 您需要建立一个在所有队列管理器的日志数据的日志范围内的时间点。但是, 由于只能从日志中 **转发** 恢复介质, 因此此时间点必须在发出 BACKUP CFSTRUCT 命令之后以及在执行任何页集备份之后。(通常, 恢复的时间点可能对应于工作日或周的结束时间。)

下图显示了队列共享组中两个队列管理器的时间线。对于每个队列管理器, 将执行页集的模糊备份 (请参阅 [方法 2: 模糊备份](#))。在队列管理器 A 上, 发出 BACKUP CFSTRUCT 命令。随后, 将在每个队列管

理器上发出 ARCHIVE LOG 命令以截断活动日志，并将其从队列管理器脱机复制到介质，可将其传输到备用站点。日志结束标识发出 ARCHIVE LOG 命令的时间，因此标记在备用站点上通常可用的日志数据的范围。恢复的时间点必须介于任何页集或 CF 结构备份的结束与备用站点上可用的最早日志结束之间。



IBM MQ 将与 CF 结构备份相关联的信息记录在 Db2 中的表中。根据您的要求，您可能希望将 IBM MQ 的恢复时间点与 Db2 的恢复时间点进行协调，或者可以获取 IBM MQ CSQ.ADMIN_B_STRBACKUP 表。

要准备恢复，请执行以下操作：

1. 为队列共享组中的每个队列管理器创建页集备份。
2. 对具有 RECOVER (YES) 属性的每个 CF 结构发出 BACKUP CFSTRUCT 命令。您可以从单个队列管理器发出这些命令，也可以从队列共享组中的不同队列管理器发出这些命令以平衡工作负载。
3. 完成所有备份后，发出 ARCHIVE LOG 命令以切换活动日志并创建队列共享组中每个队列管理器的日志和 BSD 的副本。
4. 传输页集备份，已归档日志，队列共享组中所有队列管理器的已归档 BSDS 以及您选择的 Db2 备份信息 (非现场)。

在备用站点恢复队列共享组

在可以恢复队列共享组之前，需要准备环境：

1. 如果在安装队列共享组时，耦合设施中有来自实践启动的旧信息，那么需要先清除此信息：

注：如果耦合设施中没有旧信息，那么可以省略此步骤。

- a. 输入以下 z/OS 命令以显示此队列共享组的 CF 结构：

```
D XCF,STRUCTURE,STRNAME= qsgname
```

- b. 对于以队列共享组名开头的所有结构，请使用 z/OS 命令 SETXCF FORCE CONNECTION 来强制连接关闭这些结构：

```
SETXCF FORCE,CONNECTION,STRNAME= strname,CONNAME=ALL
```

- c. 对每个结构使用以下命令来删除所有 CF 结构：

```
SETXCF FORCE,STRUCTURE,STRNAME= strname
```

2. 复原 Db2 系统和数据共享组。

3. 恢复 CSQ.ADMIN_B_STRBACKUP 表，以便它包含有关在主站点上执行的最新结构备份的信息。

注：STRBACKUP 表包含最新的结构备份信息很重要。较旧的结构备份信息可能需要由于最近的 DISPLAY USAGE TYPE (DATASET) 命令提供的信息而废弃的数据集，这意味着恢复的 CF 结构将不包含准确的信息。

4. 对队列共享组中的每个队列管理器运行 CSQ5PQSG 实用程序的 ADD QMGR 命令。这将复原每个队列管理器的 XCF 组条目。

在此场景中运行实用程序时，以下消息正常：

```
CSQU566I Unable to get attributes for admin structure, CF not found
or not allocated
CSQU546E Unable to add QMGR queue_manager_name entry,
already exists in DB2 table CSQ.ADMIN_B_QMGR
CSQU148I CSQ5PQSG Utility completed, return code=4
```

要恢复队列共享组中的队列管理器：

1. 定义新的页集数据集，并将其与来自主站点的页集副本中的数据一起装入。
2. 定义新的活动日志数据集。
3. 定义新的 BSDS 数据集，并使用 Access Method Services REPRO 将最新归档的 BSDS 复制到其中。
4. 使用打印日志映射实用程序 CSQJU004 来打印来自此最新 BSDS 的信息。在归档此 BSDS 时，您最近归档的日志将被截断为活动日志，而不会显示为归档日志。记录此日志的 STARTRBA，STARTLRSN，ENDRBA 和 ENDLRSN 值。
5. 使用更改日志库存实用程序 CSQJU003，通过使用步骤 [第 352 页的『4』](#) 中记录的值，在刚刚复原的 BSDS 中注册此最新归档日志数据集。
6. 使用 DELETE 选项 CSQJU003 从 BSDS 中除去所有活动日志信息。
7. 使用 CSQJU003 的 NEWLOG 选项将活动日志添加到 BSDS，请勿指定 STARTRBA 或 ENDRBA。
8. 计算队列共享组的 *recoverylrsn*。*recoverylrsn* 是队列共享组中所有队列管理器中 ENDLRSN 的最低值（如步骤 [第 352 页的『4』](#) 中所记录），减去 1。例如，如果队列共享组中有两个队列管理器，其中一个队列管理器的 ENDLRSN 为 B713 3C72 22C5，另一个队列管理器为 B713 3D45 2123，那么 *recoverylrsn* 为 B713 3C72 22C4。
9. 使用 CSQJU003 将重新启动控制记录添加到 BSDS。指定：

```
CRESTART CREATE, ENDLRSN= recoverylrsn
```

其中 *recoverylrsn* 是您在步骤 [第 352 页的『8』](#) 中记录的值。

BSDS 现在将所有活动日志描述为空，所有已归档日志都可用，并且在日志结束后没有检查点。

对于队列共享组中的每个队列管理器，必须将 CRESTART 记录添加到 BSDS。

10. 使用 START QMGR 命令重新启动队列共享组中的每个队列管理器。在初始化期间，将发出类似如下的操作员应答消息：

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
REPLY Y TO CONTINUE, N TO CANCEL
```

应答 Y 以启动队列管理器。队列管理器启动，并将数据恢复到 CRESTART 语句中指定的 ENDRBA。

对于 IBM WebSphere MQ 7.0.1 和更高版本，启动的第一个队列管理器可以为队列共享组的其他成员以及它自己的成员重建管理结构分区，并且在此阶段不再需要重新启动队列共享组中的每个队列管理器。

11. 当重建所有队列管理器的管理结构数据时，对每个 CF 应用程序结构发出 RECOVER CFSTRUCT 命令。

如果对单个队列管理器上的所有结构发出 RECOVER CFSTRUCT 命令，那么日志合并过程仅执行一次，因此比对每个 CF 结构在不同的队列管理器上发出命令更快，其中每个队列管理器都必须执行日志合并步骤。

在队列共享组中使用条件重新启动处理时，IBM WebSphere MQ 7.0.1 和更高版本的队列管理器将执行同级管理重建，请检查同级 BSDS 是否包含与其自身相同的 CRESTART LRSN。这是为了确保重建的管理结构的完整性。因此，在下次无条件重新启动组中的任何成员之前，重新启动 QSG 中的其他同级很重要，因此他们可以处理自己的 CRESTART 信息。

使用磁盘镜像

许多安装现在使用磁盘镜像技术 (例如，IBM Metro Mirror (以前称为 PPRC)) 在备用站点上制作数据集的同步副本。在这种情况下，由于备用站点上的 IBM MQ 页面集和日志实际上与主站点上的页面集和日志完全相同，因此许多详细步骤变得没有必要。在使用此类技术的情况下，在备用站点上重新启动队列共享组的步骤可概括为：

- 清除备用站点上的 IBM MQ CF 结构。(这些通常包含任何先前灾难恢复练习中的剩余信息)。
- 复原 IBM MQ 队列共享组所使用的数据库中的 Db2 系统和所有表。
- 重新启动队列管理器。在 IBM WebSphere MQ 7.0.1 之前，需要重新启动队列共享组中定义的每个队列管理器，因为在队列管理器重新启动期间，每个队列管理会恢复其自己的管理结构分区。在重新启动每个队列管理器之后，可以再次关闭不在其主 LPAR 上的那些队列管理器。对于 IBM WebSphere MQ 7.0.1 和更高版本，第一个队列管理器已开始为队列共享组的其他成员以及它自己的成员重新构建管理结构分区，并且不再需要重新启动队列共享组中的每个队列管理器。
- 重建管理结构后，恢复应用程序结构。

重新初始化队列管理器

如果队列管理器已异常终止，那么您可能无法将其重新启动。这可能是由于您的页集或日志已丢失，截断或损坏。如果发生此情况，那么可能必须重新初始化队列管理器 (执行冷启动)。

注意

仅当无法以任何其他方式重新启动队列管理器时，才会执行冷启动。执行冷启动使您能够恢复队列管理器和对象定义；您将无法恢复消息数据。在执行此操作之前，请检查本主题中描述的其他重新启动方案是否都不适用于您。

重新启动后，将定义所有 IBM MQ 对象并可供使用，但没有消息数据。

注：当队列管理器是集群的一部分时，请勿重新初始化该队列管理器。必须首先从集群中除去队列管理器 (在集群中的其他队列管理器上使用 RESET CLUSTER 命令)，然后重新初始化该队列管理器，最后将其作为新的队列管理器重新引入集群。

这是因为在重新初始化期间，队列管理器标识 (QMID) 已更改，因此必须从集群中除去具有旧队列管理器标识的任何集群对象。

有关更多信息，请参阅以下部分：

- [重新初始化不在队列共享组中的队列管理器](#)
- [重新初始化队列共享组中的队列管理器](#)

重新初始化不在队列共享组中的队列管理器

要重新初始化队列管理器，请执行以下过程：

1. 准备要在重新启动队列管理器时使用的对象定义语句。要执行此操作，请执行以下任一操作：
 - 如果页集 0 可用，请使用 CSQUTIL SDEFS 函数 (请参阅 [生成 IBM MQ 定义命令列表](#))。您必须获取所有对象类型 (认证信息对象，CF 结构，通道，名称列表，进程，队列和存储类) 的定义。
 - 如果页集 0 不可用，请使用上次备份对象定义时的定义。
2. 重新定义队列管理器数据集 (在完成步骤 [第 353 页](#) 的『1』之前不执行此操作)。

请参阅 [创建引导程序和日志数据集](#) 以及 [定义页集](#) 以获取更多信息。

3. 使用新定义和初始化的日志数据集， BSDS 和页集重新启动队列管理器。使用您在步骤 [第 353 页的『1』](#) 中创建的对象定义输入语句作为 CSQINP2 初始化输入数据集中的输入。

重新初始化队列共享组中的队列管理器

在队列共享组中，重新初始化队列管理器更复杂。由于页集或日志问题，可能需要重新初始化一个或多个队列管理器，但 Db2 或要处理的耦合设施也可能存在问题。正因为如此，有多种替代方案：

冷启动 (cold start)

重新初始化整个队列共享组涉及强制所有耦合设施结构，从 Db2 清除队列共享组的所有对象定义，删除或重新定义日志和 BSDS 以及为队列共享组中的所有队列管理器格式化页集。

保留共享定义

删除或重新定义日志和 BSDS，为队列共享组中的所有队列管理器格式化页集，并强制所有耦合设施结构。重新启动时，将删除所有消息。队列管理器将重新创建对应于 Db2 数据库中仍然存在的 GROUP 对象的 COPY 对象。任何共享队列仍然存在并且可以使用。

已重新启动单个队列管理器

删除或重新定义单个队列管理器的日志和 BSDS 以及格式页集 (这将删除其所有专用对象和消息)。重新启动时，队列管理器将重新创建对应于 Db2 数据库中仍然存在的 GROUP 对象的 COPY 对象。任何共享队列仍然存在，就像它们上的消息一样，并且可以使用。

队列共享组的时间点恢复

这是备用站点灾难恢复方案。

共享对象将恢复到 Db2 恢复所实现的时间点 (如 [A Db2 系统故障](#) 中所述)。可以将每个队列管理器恢复到可从备用站点上可用的备份副本实现的时间点。

持久消息可以在队列共享组中使用，并且可以使用 MQSC RECOVER CFSTRUCT 命令进行恢复。请注意，此命令将恢复到失败的时间。但是，不会恢复非持久共享队列消息；除非您已使用 CSQUTIL 实用程序的 COPY 功能独立生成备份副本，否则这些消息将丢失。

不需要尝试将每个队列管理器复原到同一时间点，因为不同队列管理器上的本地对象 (实际正在恢复的对象) 之间没有相互依赖关系，并且队列管理器在重新启动时与 Db2 再同步会根据队列管理器的需要创建或删除队列管理器上的 COPY 对象。

使用 z/OS Automatic Restart Manager (ARM)

使用本主题来了解如何使用 ARM 自动重新启动队列管理器。

本节包含有关以下主题的信息：

- [第 354 页的『什么是 ARM?』](#)
- [第 355 页的『ARM 策略』](#)
- [第 356 页的『在 IBM MQ 网络中使用 ARM』](#)

什么是 ARM?

z/OS 自动重新启动管理器 (ARM) 是 z/OS 恢复功能，可提高队列管理器的可用性。当作业或任务失败或运行该作业或任务的系统失败时，ARM 可以在没有操作员干预的情况下重新启动该作业或任务。

如果队列管理器或通道启动程序失败，那么 ARM 将在同一 z/OS 映像上重新启动该队列管理器或通道启动程序。如果 z/OS，并因此导致整个相关子系统和应用程序组发生故障，那么 ARM 可以按预定义顺序在综合系统中的另一个 z/OS 映像上自动重新启动所有发生故障的系统。这称为跨系统重新启动。

仅在特殊情况下由 ARM 重新启动通道启动程序。如果队列管理器由 ARM 重新启动，请从 CSQINP2 初始化数据集重新启动通道启动程序 (请参阅 [第 356 页的『在 IBM MQ 网络中使用 ARM』](#))。

在发生 z/OS 故障时，可以使用 ARM 在综合系统中的其他 z/OS 映像上重新启动队列管理器。[第 356 页的『在 IBM MQ 网络中使用 ARM』](#) 中描述了 IBM MQ ARM 在其他 z/OS 映像上重新启动的网络含义。

要启用自动重新启动：

- 安装 ARM 耦合数据集。
- 定义您希望 z/OS 在 ARM 策略中执行的自动重新启动操作。
- 启动 ARM 策略。

此外，IBM MQ 必须在启动时向 ARM 注册 (这会自动发生)。

注: 如果要自动重新启动不同 z/OS 映像中的队列管理器，那么必须将每个队列管理器定义为可能在其上重新启动该队列管理器的每个 z/OS 映像中的子系统，并使用综合系统范围唯一的四字符子系统名称。

ARM 耦合数据集

在启动需要 ARM 支持的任何队列管理器之前，请确保定义 ARM 所需的耦合数据集，并确保这些数据集处于联机状态和活动状态。如果耦合数据集在队列管理器启动时不可用，那么 IBM MQ 自动 ARM 注册将失败。在此情况下，IBM MQ 假定缺少耦合数据集意味着您不需要 ARM 支持，并且初始化将继续。

请参阅 [z/OS MVS Setting up a Sysplex](#)，以获取有关 ARM 耦合数据集的信息。

z/OS ARM 策略

"自动重新启动管理器" 策略是用户定义的规则，用于控制可控制队列管理器的任何重新启动的 ARM 函数。

ARM 函数由用户定义的 ARM 策略控制。每个运行要由 ARM 重新启动的队列管理器实例的 z/OS 映像都必须连接到具有活动 ARM 策略的 ARM 耦合数据集。

IBM 提供了缺省 ARM 策略。您可以定义新策略，也可以使用 z/OS 随附的 [管理数据实用程序 \(IXCMIAPU\)](#) 来覆盖策略缺省值。z/OS MVS 设置综合系统 [描述此实用程序](#)，并包含有关如何定义 ARM 策略的完整详细信息。

第 355 页的图 53 显示了 ARM 策略的示例。如果队列管理器发生故障或整个系统发生故障，那么此样本策略将重新启动综合系统中的任何队列管理器。

```
//IXCMIAPU EXEC PGM=IXCMIAPU,REGION=2M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DATA TYPE(ARM)
DEFINE POLICY NAME(ARMPOL1) REPLACE(YES)
RESTART_GROUP(DEFAULT)
ELEMENT(*)
RESTART_ATTEMPTS(0) /* Jobs not to be restarted by ARM */
RESTART_GROUP(GROUP1)
ELEMENT(SYSQMGRM*) /* These jobs to be restarted by ARM */
/*
```

图 53: 样本 ARM 策略

有关更多信息，请参阅：

- [定义 ARM 策略](#)
- [激活 ARM 策略](#)
- [向 ARM 注册](#)

定义 ARM 策略

设置 ARM 策略，如下所示：

- 为每个还包含连接到该队列管理器实例的任何 CICS 或 IMS 子系统的队列管理器实例定义 RESTART_GROUPS。如果使用子系统命名约定，那么您可能可以使用 "?" 以及元素名称中的 "*" 通配符，以通过最小的定义工作量来定义 RESTART_GROUPS。
- 为通道启动程序指定 TERMTYPE (ELEMTERM)，以指示仅当通道启动程序失败且 z/OS 映像未失败时才会重新启动这些启动程序。

- 为队列管理器指定 TERMTYPE (ALLTERM)，以指示在队列管理器发生故障或 z/OS 映像发生故障时将重新启动这些队列管理器。
- 为队列管理器和通道启动程序指定 RESTART_METHOD (BOTH, PERSIST)。这指示 ARM 在上次启动期间使用它保存的 JCL (在解析系统符号之后) 重新启动。它指示 ARM 执行此操作，而不管单个元素是失败还是 z/OS 映像失败。
- 接受所有其他 ARM 策略选项的缺省值。

激活 ARM 策略

要启动自动重新启动管理策略，请发出以下 z/OS 命令：

```
SETXCF START,POLICY,TYPE=ARM,POLNAME= mypol
```

启动策略时，连接到 ARM 耦合数据集的所有系统都使用相同的活动策略。

使用 SETXCF STOP 命令来禁用自动重新启动。

向 ARM 注册

IBM MQ 在队列管理器启动期间自动注册为 ARM 元素 (取决于 ARM 可用性)。它在其关闭阶段注销，除非请求不注销。

在启动时，队列管理器确定 ARM 是否可用。如果是，IBM MQ 将使用名称 SYSMQMGR *ssid* 进行注册，其中 *ssid* 是 4 个字符的队列管理器名称，SYSMQMGR 是元素类型。

STOP QMGR MODE (QUIESCE) 和 STOP QMGR MODE (FORCE) 命令从 ARM 注销队列管理器 (如果在启动时已向 ARM 注册)。这将阻止 ARM 重新启动此队列管理器。STOP QMGR MODE (RESTART) 命令不会从 ARM 注销队列管理器，因此可以立即自动重新启动。

每个通道启动程序地址空间确定 ARM 是否可用，如果可用，那么使用元素名称 SYSMQCH *ssid* 进行注册，其中 *ssid* 是队列管理器名称，SYSMQCH 是元素类型。

通道启动程序在正常停止时始终会从 ARM 注销，并且仅当它异常结束时才会保持已注册状态。如果队列管理器发生故障，那么将始终注销通道启动程序。

在 IBM MQ 网络中使用 ARM

您可以设置队列管理器，以便在重新启动队列管理器时自动启动通道启动程序和关联的侦听器。

要确保针对 LU 6.2 和 TCP/IP 通信协议在同一 z/OS 映像上完全自动重新启动队列管理器，请执行以下操作：

- 通过将相应的 START LISTENER 命令添加到 CSQINPX 数据集来自动启动侦听器。
- 通过将相应的 START CHINIT 命令添加到 CSQINP2 数据集，自动启动通道启动程序。

要使用 TCP/IP 或 LU6.2 重新启动队列管理器，请参阅

- [第 356 页的『使用 TCP/IP 在其他 z/OS 映像上重新启动』](#)
- [第 358 页的『在具有 LU 6.2 的其他 z/OS 映像上重新启动』](#)

请参阅 [任务 13: 定制初始化输入数据集](#)，以获取有关 CSQINP2 和 CSQINPX 数据集的信息。

使用 TCP/IP 在其他 z/OS 映像上重新启动

如果使用 TCP/IP 作为通信协议，并且使用虚拟 IP 地址，那么可以将这些地址配置为在其他 z/OS 映像上恢复，从而允许连接到该队列管理器的通道重新连接而不进行任何更改。否则，仅当您正在使用集群或使用 WLM 动态域名系统 (DNS) 逻辑组名连接到队列共享组时，才能在将队列管理器移动到其他 z/OS 映像之后重新分配 TCP/IP 地址。

- [使用集群时](#)
- [连接到队列共享组时](#)

使用集群时

z/OS ARM 通过在同一综合系统中的另一个 z/OS 映像上重新启动队列管理器来响应系统故障; 此系统具有与原始 z/OS 映像不同的 TCP/IP 地址。以下说明了在将队列管理器的 TCP/IP 地址由 ARM 重新启动移动到另一个 z/OS 映像之后, 如何使用 IBM MQ 集群来重新分配该地址。

当客户机队列管理器检测到队列管理器故障 (作为通道故障) 时, 它会通过将其集群传输队列上的适当消息重新分配给另一个服务器队列管理器来响应, 该服务器队列管理器主管另一个目标集群队列实例。但是, 它无法重新分配通过亲缘关系约束绑定到原始服务器的消息, 也无法重新分配由于服务器队列管理器在批处理结束期间失败而存在疑问的消息。要处理这些消息, 请执行以下操作:

1. 向每个 z/OS 队列管理器分配不同的集群接收方通道名称和不同的 TCP/IP 端口。每个队列管理器都需要一个不同的端口, 以便两个系统可以在 z/OS 映像上共享单个 TCP/IP 堆栈。其中一个是最初在该 z/OS 映像上运行的队列管理器, 另一个是在系统发生故障后 ARM 将在该 z/OS 映像上重新启动的队列管理器。配置每个 z/OS 映像上的每个端口, 以便 ARM 可以重新启动任何 z/OS 映像上的任何队列管理器。
2. 为要在通道启动程序启动期间引用的每个队列管理器和 z/OS 映像组合创建不同的通道启动程序命令输入文件 (CSQINPX)。

Each CSQINPX file must include a START LISTENER PORT(port) command specific to that queue manager, and an ALTER CHANNEL command for a cluster-receiver channel specific to that queue manager and z/OS image combination. ALTER CHANNEL 命令需要将连接名称设置为重新启动该连接的 z/OS 映像的 TCP/IP 名称。它必须包含特定于重新启动的队列管理器的端口号作为连接名称的一部分。

每个队列管理器的启动 JCL 可以具有此 CSQINPX 文件的固定数据集名称, 并且每个 z/OS 映像必须具有非共享 DASD 卷上每个 CSQINPX 文件的不同版本。

如果发生 ARM 重新启动, 那么 IBM MQ 会将更改后的通道定义发布到集群存储库, 而集群存储库会将其发布到表示对服务器队列管理器感兴趣的所有客户机队列管理器。

客户机队列管理器将服务器队列管理器故障视为通道故障, 并尝试重新启动失败的通道。当客户机队列管理器学习新的服务器连接名称时, 通道重新启动会将客户机队列管理器重新连接到重新启动的服务器队列管理器。然后, 客户机队列管理器可以再同步其消息, 解析客户机队列管理器的传输队列上的任何不确定消息, 并且正常处理可以继续。

连接到队列共享组时

通过 TCP/IP 动态域名系统 (DNS) 逻辑组名连接到队列共享组时, 通道定义中的连接名称指定队列共享组的逻辑组名, 而不是物理机器的主机名或 IP 地址。当此通道启动时, 它将连接到动态 DNS, 然后连接到队列共享组中的其中一个队列管理器。使用队列共享组为 IBM MQ for z/OS 设置通信对此过程进行了说明。

在不太可能发生图像故障的情况下, 将发生下列其中一种情况:

- 失败映像上的队列管理器从综合系统上运行的动态 DNS 注销。通道通过进入 RE 试用状态来响应连接失败, 然后连接到在综合系统上运行的动态 DNS。动态 DNS 将入站请求分配给仍在其余映像上运行的队列共享组的其中一个剩余成员。
- 如果队列共享组中没有其他队列管理器处于活动状态, 并且 ARM 在另一个映像上重新启动队列管理器和通道启动程序, 那么组侦听器将从此新映像向动态 DNS 注册。这意味着逻辑组名 (来自通道的连接名称字段) 连接到动态 DNS, 然后连接到同一队列管理器, 现在正在另一个映像上运行。不需要对通道定义进行任何更改。

要进行此类型的恢复, 必须注意以下几点:

- 在 z/OS 上, 动态 DNS 在综合系统中的某个 z/OS 映像上运行。如果此映像失败, 那么需要配置动态 DNS, 以便在综合系统中有一个辅助名称服务器处于活动状态, 作为主名称服务器的替代方法。可在 [OS/390 SecureWay CS IP 配置手册](#) 中找到有关主动态 DNS 服务器和辅助动态 DNS 服务器的信息。

- TCP/IP 组侦听器可能已在此 z/OS 映像上不可用的特定 IP 地址上启动。如果是这样，那么可能需要在新的映像上的其他 IP 地址上启动侦听器。如果您正在使用虚拟 IP 地址，那么可以将这些地址配置为在其他 z/OS 映像上恢复，以便不需要更改 START LISTENER 命令。

在具有 LU 6.2 的其他 z/OS 映像上重新启动

如果仅使用 LU 6.2 通信协议，请执行以下过程以在综合系统中不同 z/OS 映像上的队列管理器自动重新启动后启用网络重新连接：

- 使用唯一的子系统名称定义综合系统中的每个队列管理器。
- 使用唯一的 LUNAME 定义综合系统中的每个通道启动程序。这在队列管理器属性和 START LISTENER 命令中都指定。

注：LUNAME 在 APPC 侧表中命名一个条目，然后将其映射到实际的 LUNAME。

- 设置共享 APPC 端表，此表由综合系统中的每个 z/OS 映像引用。这应该包含每个通道启动程序的 LUNAME 的条目。请参阅 [z/OS MVS Planning: APPC/MVS Management](#) 以获取有关此问题的信息。
- 为综合系统中的每个通道启动程序设置 SYS1.PARMLIB 的 APPCPM Xx 成员，以包含 LUADD 来激活该通道启动程序的 APPC 侧表条目。这些成员应该由每个 z/OS 映像共享。相应的 SYS1.PARMLIB 成员由 z/OS 命令 SET APPC= xx 激活，此命令是在不同 z/OS 映像上的队列管理器 (及其通道启动程序) 的 ARM 重新启动期间自动发出的，如以下文本中所述。
- 使用 LU62ARM 队列管理器属性为每个通道启动程序指定此 SYS1.PARMLIB 成员的 Xx 后缀。这将导致通道启动程序发出必需的 z/OS 命令 SET APPC= xx 以激活其 LUNAME。

定义 ARM 策略，以便它仅在其 z/OS 映像保持运行时失败时重新启动通道启动程序；必须授权与 XCFAS 地址空间关联的用户标识发出 IBM MQ 命令 START CHINIT。如果通道启动程序的 z/OS 映像也失败，请不要自动重新启动该通道启动程序，而是使用 CSQINP2 和 CSQINPX 数据集中的命令来启动通道启动程序和侦听器。

z/OS 手动恢复工作单元

您可以手动恢复队列共享组中的工作单元 CICS，IMS，RRS 或其他队列管理器。您可以使用队列管理器命令来显示与队列管理器的每个连接相关联的工作单元的状态。

本主题包含有关以下主题的信息：

- [第 358 页的『显示连接和线程』](#)
- [第 359 页的『手动恢复 CICS 个恢复单元』](#)
- [第 362 页的『手动恢复 IMS 个恢复单元』](#)
- [第 363 页的『手动恢复 RRS 恢复单元』](#)
- [第 364 页的『在队列共享组中的另一个队列管理器上恢复恢复单元』](#)

显示连接和线程

您可以使用 DISPLAY CONN 命令来获取有关与队列管理器及其关联工作单元的连接的信息。您可以显示活动工作单元以查看当前正在发生的情况，或者查看需要终止哪些工作单元以允许队列管理器关闭，并且可以显示未解决的工作单元以帮助恢复。

活动工作单元

要仅显示活动工作单元，请使用

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ ACTIVE)
```

未解决的工作单元

未解析的工作单元 (也称为 "不确定线程") 是处于两阶段落实操作的第二次传递中的工作单元。资源代表其保存在 IBM MQ 中。要显示未解析的工作单元, 请使用

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

需要外部干预来解决未解决的工作单元的状态。这可能只涉及启动恢复协调程序 (CICS, IMS 或 RRS), 也可能涉及更多内容, 如以下部分中所述。

手动恢复 CICS 个恢复单元

使用本主题来了解 CICS 适配器重新启动时发生的情况, 然后说明如何处理出现的任何未解决的恢复单元。

CICS 适配器重新启动时发生的情况

每当连接中断时, 适配器都必须在重新连接过程期间经历重新启动阶段。重新启动阶段将再同步资源。CICS 与 IBM MQ 之间的再同步使您能够识别和解析不确定的工作单元。

再同步可能由以下原因导致:

- 来自分布式排队组件的显式请求
- 与 IBM MQ 建立连接时的隐式请求

如果再同步是由连接到 IBM MQ 导致的, 那么事件序列为:

1. 连接过程从 IBM MQ 检索不确定工作单元 (UOW) 标识的列表。
2. UOW 标识显示在控制台上的 CSQC313I 消息中。
3. UOW 标识将传递到 CICS。
4. CICS 对每个不确定的 UOW 标识启动再同步任务 (CRSY)。
5. 每个不确定 UOW 的任务结果将显示在控制台上。

您需要检查在连接过程中显示的消息:

CSQC313I

显示 UOW 处于不确定状态。

CSQC400I

标识 UOW 并后跟下列其中一条消息:

- CSQC402I 或 CSQC403I 显示已成功解析 (落实或回退) UOW。
- CSQC404E, CSQC405E, CSQC406E 或 CSQC407E 显示未解析 UOW。

CSQC409I

显示已成功解析所有 UOW。

CSQC408I

显示并非所有 UOW 都已成功解析。

CSQC314I

警告使用 * 突出显示的 UOW 标识不会自动解析。这些 UOW 必须由分布式排队组件在重新启动时显式解析。

第 360 页的图 54 显示了 z/OS 控制台上显示的重新启动消息的示例集。

```

CSQ9022I +CSQ1 CSQYASCP ' START QMGR' NORMAL COMPLETION
+CSQC323I VICIC1 CSQCQCON CONNECT received from TERMID=PB62 TRANID=CKCN
+CSQC303I VICIC1 CSQCCON CSQCSERV loaded. Entry point is 850E8918
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F0E2178D25 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F055B2AC25 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6EFFD60D425 is in doubt
+CSQC313I VICIC1 CSQCCON UOWID=VICIC1.A6E5A6F07AB56D22 is in doubt
+CSQC307I VICIC1 CSQCCON Successful connection to subsystem VC2
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAD18) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BAA10) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008BA708) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAE88) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CAB80) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA878) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA570) connect
successful
+CSQC472I VICIC1 CSQCSERV Server subtask (TCB address=008CA268) connect
successful
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F0E2178D25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F055B2AC25
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6F07AB56D22
+CSQC403I VICIC1 CSQCTRUE Resolved BACKOUT for
+CSQC400I VICIC1 CSQCTRUE UOWID=VICIC1.A6E5A6EFFD60D425
+CSQC409I VICIC1 CSQCTRUE Resynchronization completed successfully

```

图 54: 重新启动消息示例

CSQC313I 消息总数应等于 CSQC402I 加上 CSQC403I 消息总数。如果总数不相等，那么存在连接进程无法解析的 UOW。无法解决的那些 UOW 是由 CICS (例如，冷启动) 或 IBM MQ 的问题或通过分发排队引起的。修复这些问题后，您可以通过断开连接然后重新连接来启动另一个再同步。

或者，您可以使用 RESOLVE INDOUBT 命令和消息 CSQC400I 中显示的 UOW 标识自行解析每个未完成的 UOW。然后，必须启动断开连接和连接以清除 CICS 中的恢复单元描述符。您需要知道 UOW 的正确结果，才能手动解析 UOW。

与未解决的 UOW 相关联的所有消息都由 IBM MQ 锁定，并且没有任何批处理，TSO 或 CICS 任务可以访问这些消息。

如果 CICS 失败并且需要紧急重新启动，那么请勿改变 CICS 系统的 GENERIC APPLID。如果执行此操作，然后重新连接到 IBM MQ，那么无法保证与 IBM MQ 的数据完整性。这是因为 IBM MQ 将 CICS 的新实例视为不同的 CICS (因为 APPLID 不同)。然后，基于错误的 CICS 日志进行不确定解析。

如何手动解析 CICS 恢复单元

如果适配器异常结束，那么 CICS 和 IBM MQ 构建不确定列表将动态列示或在重新启动期间列示，具体取决于导致异常结束的子系统。

注: 如果使用 DFH\$INDB 样本程序来显示工作单元，那么您可能会发现它不会始终正确显示 IBM MQ 个 UOW。

当 CICS 连接到 IBM MQ 时，可能有一个或多个恢复单元尚未解析。

将下列其中一条消息发送到控制台:

- CSQC404E
- CSQC405E
- CSQC406E

- CSQC407E
- CSQC408I

有关这些消息的含义的详细信息，请参阅 [CICS 适配器和网桥消息](#) 消息。

CICS 保留连接启动期间未解析的恢复单元的详细信息。当条目不再显示在 IBM MQ 提供的列表上时，将清除该条目。

必须使用 IBM MQ 命令手动解析 CICS 无法解析的任何恢复单元。此手动过程很少在安装中使用，因为只有当操作错误或软件问题导致无法自动解决时才需要此过程。必须调查在不确定解决期间发现的任何不一致情况。

要解析恢复单元：

1. 使用以下命令从 IBM MQ 获取恢复单元列表：

```
+CSQ1 DISPLAY CONN( * ) WHERE(UOWSTATE EQ UNRESOLVED)
```

您会收到以下消息：

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC85772CBE3E0001)
EXTCONN(C3E2D8C3C7D9F0F940404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2005-02-04)
UOWLOGTI(10.17.44)
UOWSTDA(2005-02-04)
UOWSTTI(10.17.44)
UOWSTATE(UNRESOLVED)
NID(IYRCSQ1 .BC8571519B60222D)
EXTURID(BC8571519B60222D)
QMURID(0000002BDA50)
URTYPE(CICS)
USERID(MQTEST)
APPLTAG(IYRCSQ1)
ASID(0000)
APPLTYPE(CICS)
TRANSID(GP02)
TASKNO(0000096)
END CONN DETAILS
```

对于 CICS 连接，NID 由 CICS applid 和 CICS 在写入同步点日志条目时提供的唯一编号组成。此唯一编号存储在同步点处理时写入 CICS 系统日志和 IBM MQ 日志的记录中。此值在 CICS 中称为恢复令牌。

2. 扫描 CICS 日志以查找与特定恢复单元相关的条目。

查找与任务相关的安装的 PREPARE 记录，其中恢复令牌字段 (JCSRMTKN) 等于从网络标识获取的值。网络标识由 DISPLAY CONN 命令输出中的 IBM MQ 提供。

恢复单元的 CICS 日志中的 PREPARE 记录提供了 CICS 任务号。可以使用此编号来查找此 CICS 任务的日志上的所有其他条目。

扫描日志时，可以使用 CICS 日志打印实用程序 DFHJUP。有关使用此程序的详细信息，请参阅 *CICS Operations and Utilities Guide*。

3. 扫描 IBM MQ 日志以查找 NID 与特定恢复单元相关的记录。然后使用此记录中的 URID 来获取此恢复单元的其余日志记录。

扫描 IBM MQ 日志时，请注意 IBM MQ 启动消息 CSQJ001I 提供此会话的启动 RBA。

打印日志记录程序 (CSQ1LOGP) 可用于此目的。

4. 如果需要，请在 IBM MQ 中执行不确定解析。

可以指示 IBM MQ 使用 `IBM MQ RESOLVE INDOUBT` 命令对恢复单元执行恢复操作。

要恢复与特定 `connection-name` 关联的所有线程，请使用 `NID (*)` 选项。

该命令会生成下列其中一条消息，显示线程是已落实还是已回退：

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id ABORT SCHEDULED
```

执行不确定解析时，CICS 和适配器不知道用于 IBM MQ 落实或回退恢复单元的命令，因为只有 IBM MQ 资源受影响。但是，CICS 会保留有关 IBM MQ 无法解析的不确定线程的详细信息。当显示的列表为空时，或者当列表不包含 CICS 具有详细信息的恢复单元时，将清除此信息。

手动恢复 IMS 个恢复单元

使用本主题来了解 IMS 适配器重新启动时发生的情况，然后说明如何处理出现的任何未解决的恢复单元。

IMS 适配器重新启动时发生的情况

每当重新启动与 IBM MQ 的连接时（在队列管理器重新启动或 `IMS /START SUBSYS` 命令之后），IMS 都会启动以下再同步过程：

1. IMS 提供了它认为对 IBM MQ IMS 适配器有疑问的工作单元 (UOW) 标识的列表，一次一个带有解析参数“落实”或“回退”。
2. IMS 适配器将解析请求传递到 IBM MQ，并将结果报告回 IMS。
3. 在处理了所有 IMS 解析请求之后，IMS 适配器从 IBM MQ 中获取 IBM MQ 仍存在疑问的所有 UOW 的列表，这些 UOW 由 IMS 系统启动。这些将在消息 `CSQQ008I` 中报告给 IMS 主终端。

注：当 UOW 处于不确定状态时，任何关联的 IBM MQ 消息都将被 IBM MQ 锁定，并且不可用于任何应用程序。

如何手动解析 IMS 恢复单元

当 IMS 连接到 IBM MQ 时，IBM MQ 可能有一个或多个未解析的不确定恢复单元。

如果 IBM MQ 具有 IMS 未解析的不确定恢复单元，那么将在 IMS 主终端发出以下消息：

```
CSQQ008I nn units of recovery are still in doubt in queue manager qmgr-name
```

如果发出此消息，那么 IMS 已冷启动或使用不完整的日志磁带启动。如果 IBM MQ 或 IMS 由于软件错误或其他子系统故障而异常终止，那么也可以发出此消息。

接收到 `CSQQ008I` 消息后：

- 连接保持活动状态。
- IMS 应用程序仍然可以访问 IBM MQ 资源。
- 某些 IBM MQ 资源仍处于锁定状态。

如果未解析不确定线程，那么 IMS 消息队列可以开始构建。如果 IMS 队列填充到容量，那么 IMS 将终止。您必须了解此潜在困难，并且必须监视 IMS，直到完全解决不确定的恢复单元。

恢复过程

使用以下过程来恢复 IMS 工作单元：

1. 使用 `/SWI OLDS` 强制关闭 IMS 日志，然后归档 IMS 日志。使用实用程序 `DFSERA10` 来打印先前 IMS 日志磁带中的记录。类型 `X'3730'` 日志记录指示 `phase-2` 落实请求，类型 `X'38'` 日志记录指示异常中止请求。记录每个从属区域中最后一个事务的请求操作。

- 运行 DL/I 批处理作业以回退未达到落实点的每个涉及的 PSB。此过程可能需要一些时间，因为仍在处理事务。它还可能锁定大量记录，这可能会影响处理的其余部分和消息队列的其余部分。
- 使用以下命令从 IBM MQ 生成不确定的恢复单元列表：

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

您会收到以下消息：

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3E2C5C3F240404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2005-02-15)
UOWLOGTI(16.39.43)
UOWSTDA(2005-02-15)
UOWSTTI(16.39.43)
UOWSTATE(UNRESOLVED)
NID(IM8F .BC45A794D3810344)
EXTURID(
0000052900000000
)
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0000)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)
```

对于 IMS，NID 由 IMS 连接名称和 IMS 提供的唯一编号组成。该值在 IMS 中称为恢复令牌。有关更多信息，请参阅 *IMS 定制指南*。

- 将 DISPLAY THREAD 消息中显示的 NID (IMS 标识加上十六进制 OASN) 与 DFSERA10 输出中显示的 OASN (4 字节十进制) 进行比较。决定是落实还是回退。
- 使用 [RESOLVE INDOUBT](#) 命令在 IBM MQ 中执行不确定解析，如下所示：

```
RESOLVE INDOUBT( connection-name )
ACTION(COMMIT|BACKOUT)
NID( network-id )
```

要恢复与 *connection-name* 关联的所有线程，请使用 NID (*) 选项。该命令会生成下列其中一条消息，以指示线程是已落实还是已回退：

```
CSQV414I THREAD network-id COMMIT SCHEDULED
CSQV415I THREAD network-id BACKOUT SCHEDULED
```

执行不确定解析时，IMS 和适配器不知道用于 IBM MQ 落实或回退不确定恢复单元的命令，因为只有 IBM MQ 个资源受影响。

手动恢复 RRS 恢复单元

使用本主题来了解如何确定是否存在不确定的 RRS 恢复单元，以及如何手动解析这些恢复单元。

当 RRS 连接到 IBM MQ 时，IBM MQ 可能有一个或多个未解析的不确定恢复单元。如果 IBM MQ 具有 RRS 未解析的不确定恢复单元，那么将在 z/OS 控制台上发出下列其中一条消息：

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

IBM MQ 和 RRS 都提供了用于显示有关不确定恢复单元的信息的工具，以及用于手动解析这些单元的方法。在 IBM MQ 中，使用 DISPLAY CONN 命令来显示有关不确定 IBM MQ 线程的信息。该命令的输出包含那些将 RRS 作为协调程序的 IBM MQ 线程的 RRS 恢复单元标识。这可用于确定恢复单元的结果。使用 RESOLVE INDOUBT 命令手动解析 IBM MQ 不确定线程。此命令可用于在确定正确的决策后落实或回退恢复单元。

z/OS 在队列共享组中的另一个队列管理器上恢复恢复单元

使用本主题来识别和手动恢复队列共享组中其他队列管理器上的恢复单元。

如果作为队列共享组的成员的队列管理器发生故障并且无法重新启动，那么该组中的其他队列管理器可以执行对等恢复并从中接管。但是，队列管理器可能具有无法由同级恢复解析的不确定恢复单元，因为该恢复单元的最终处置仅对失败的队列管理器是已知的。当最终重新启动队列管理器时，将解析这些恢复单元，但在此之前，它们仍处于不确定状态。

这意味着某些资源 (例如，消息) 可能被锁定，从而使组中的其他队列管理器无法使用这些资源。在这种情况下，可以使用 DISPLAY THREAD 命令在不活动的队列管理器上显示这些工作单元。如果要手动解析这些恢复单元以使消息可供组中的其他队列管理器使用，那么可以使用 RESOLVE INDOUBT 命令。

当发出 DISPLAY THREAD 命令以显示不确定的恢复单元时，可以使用 QMNAME 关键字来指定不活动队列管理器的名称。例如，如果发出以下命令：

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT) QMNAME(QM01)
```

您将接收到以下消息：

```
CSQV436I +CSQ1 INDOUBT THREADS FOR QM01 -  
NAME   THREAD-XREF   URID NID  
USER1  0000000000000000000000000000 CSQ:0001.0  
USER2  0000000000000000000000000000 CSQ:0002.0  
DISPLAY THREAD REPORT COMPLETE
```

如果指定的队列管理器处于活动状态，那么 IBM MQ 不会返回有关不确定线程的信息，但会发出以下消息：

```
CSQV435I CANNOT USE QMNAME KEYWORD, QM01 IS ACTIVE
```

使用 IBM MQ 命令 RESOLVE INDOUBT 来手动解析不确定线程。使用 QMNAME 关键字在命令中指定不活动队列管理器的名称。

此命令可用于落实或回退恢复单元。此命令仅解析恢复单元的共享部分；任何本地消息都不受影响并保持锁定状态，直到队列管理器重新启动或重新连接到 CICS，IMS 或 RRS 批处理。

z/OS IBM MQ 和 IMS

IBM MQ 提供了两个组件来与 IMS，IBM MQ - IMS 适配器和 IBM MQ - IMS 网桥进行交互。这些组件通常称为 IMS 适配器和 IMS 网桥。

z/OS 操作 IMS 适配器

使用本主题来了解如何操作 IMS 适配器，该适配器将 IBM MQ 连接到 IMS 系统。

注：IMS 适配器不包含任何操作和控制面板。

本主题包含以下几部分：

- [第 365 页的『控制 IMS 连接』](#)

- [第 365 页的『从 IMS 控制区域进行连接』](#)
- [第 367 页的『显示不确定的恢复单元』](#)
- [第 368 页的『控制 IMS 从属区域连接』](#)
- [第 371 页的『断开连接 IMS』](#)
- [第 371 页的『控制 IMS 触发器监视器』](#)

控制 IMS 连接

使用本主题来了解用于控制和监视与 IBM MQ 的连接的 IMS 操作员命令。

IMS 提供了以下操作员命令来控制 and 监视与 IBM MQ 的连接:

/CHANGE 子系统

从 IMS 中删除不确定的恢复单元。

/DISPLAY OASN SUBSYS

显示未完成的恢复元素。

/DISPLAY 子系统

显示连接状态和线程活动。

/START SUBSYS

将 IMS 控制区域连接到队列管理器。

/STOP SUBSYS

将 IMS 与队列管理器断开连接。

/TRACE

控制 IMS 跟踪。

有关这些命令的更多信息，请参阅 *IMS/ESA Operator 's Reference* 手册以了解您正在使用的 IMS 级别。

IMS 命令响应将发送到从中发出命令的终端。发出 IMS 命令的授权基于 IMS 安全性。

从 IMS 控制区域进行连接

使用本主题来了解可用于从 IMS 连接到 IBM MQ 的机制。

IMS 从其控制区域与使用 IMS 的每个队列管理器建立一个连接。必须启用 IMS 才能通过下列其中一种方式建立连接:

- 在以下任一期间自动执行:
 - 冷启动初始化。
 - 如果 IBM MQ 连接在 IMS 关闭时处于活动状态，那么会热启动 IMS。
- 作为对 IMS 命令的响应:

```
/START SUBSYS sysid
```

其中 *sysid* 是队列管理器名称。

无论队列管理器是否处于活动状态，都可以发出该命令。

直到对队列管理器进行第一次 MQ API 调用之后，才会建立连接。在此之前，IMS 命令 /DIS SUBSYS 将状态显示为 "NOT CONN"。

启动 IMS 和队列管理器的顺序并不重要。

如果使用 STOP QMGR 命令，IMS 命令 /STOP SUBSYS 或异常结束来停止队列管理器，那么 IMS 无法自动重新启用与队列管理器的连接。因此，必须使用 IMS 命令 /START SUBSYS 建立连接。

初始化适配器并连接到队列管理器

适配器是一组使用 IMS 外部子系统连接设施装入到 IMS 控制区域和从属区域中的模块。

此过程将初始化适配器并连接到队列管理器：

1. 从 IMS 读取子系统成员 (SSM)。PROCLIB。选择的 SSM 是 IMS EXEC 参数。对于 IMS 可以连接到的每个队列管理器，成员中都有一个条目。每个条目都包含有关 IBM MQ 适配器的控制信息。
2. 装入 IMS 适配器。
注：IMS 为 SSM 成员中定义的每个 IBM MQ 实例装入一个适配器模块副本。
3. 连接 IBM MQ 的外部子系统任务。
4. 使用 CTL EXEC 参数 (IMS 标识) 作为连接名称运行适配器。

无论连接是初始化的一部分还是 IMS 命令 /START SUBSYS 的结果，该过程都是相同的。

如果当 IMS 尝试建立连接时队列管理器处于活动状态，那么将发送以下消息：

- 到 z/OS 控制台：

```
DFS3613I ESS TCB INITIALIZATION COMPLETE
```

- 到 IMS 主终端：

```
CSQQ000I IMS/TM imsid connected to queue manager ssnm
```

当 IMS 尝试建立连接并且队列管理器处于不活动状态时，每次应用程序进行 MQI 调用时，都会将以下消息发送到 IMS 主终端：

```
CSQQ001I IMS/TM imsid not connected to queue manager ssnm.  
Notify message accepted  
DFS3607I MQM1 SUBSYSTEM ID EXIT FAILURE, FC = 0286, RC = 08,  
JOBNAME = IMSEMPR1
```

如果在启动与 IMS 的连接或在系统启动时收到 DFS3607I 消息，那么这指示队列管理器不可用。要防止生成大量消息，必须执行下列其中一项操作：

1. 启动相关队列管理器。
2. 发出 IMS 命令：

```
/STOP SUBSYS
```

以便 IMS 不会期望连接到队列管理器。

如果不执行任何操作，那么每次在区域中调度作业时，以及每次应用程序向队列管理器发出连接请求时，都会发出 DFS3607I 消息和关联的 CSQQ001I 消息。

线程连接

在 MPP 或 IFP 区域中，IMS 在将第一个应用程序调度到该区域时建立线程连接，即使该应用程序未进行 IBM MQ 调用也是如此。在 BMP 区域中，当应用程序进行其第一次 IBM MQ 调用 (MQCONN 或 MQCONNX) 时，将建立线程连接。此线程将在区域的持续时间内保留，或者保留到连接停止为止。

对于消息驱动区域和非消息驱动区域，与线程关联的恢复线程交叉引用标识 *Thread-xref* 为：

```
PSTid + PSBname
```

其中：

PSTid

分区规范表区域标识

PSBname

程序规范块名称

您可以在 IBM MQ 命令中使用连接标识作为唯一标识，在这种情况下，IBM MQ 会自动将这些标识插入到它生成的任何操作员消息中。

显示不确定的恢复单元

您可以显示不确定的恢复单元并尝试恢复这些单元。

本主题中用于列示和恢复不确定的恢复单元的操作步骤仅适用于相对简单的情况。如果队列管理器在连接到 IMS 时异常结束，那么 IMS 可能会在 IBM MQ 不知道的情况下落实或回退工作。当队列管理器重新启动时，该工作称为不确定。必须对工作状态作出决定。

要显示不确定恢复单元的列表，请发出以下命令：

```
+CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

IBM MQ 通过类似如下的消息进行响应：

```
CSQM201I +CSQ1 CSQMDRTC DIS CONN DETAILS
CONN(BC0F6125F5A30001)
EXTCONN(C3E2D8C3C3E2D8F140404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2004-11-02)
UOWLOGTI(12.27.58)
UOWSTDA(2004-11-02)
UOWSTTI(12.27.58)
UOWSTATE(UNRESOLVED)
NID(CSQ1CHIN.BC0F5F1C86FC0766)
EXTURID(00000000000001F000000007472616E5F6964547565204E6F762020...)
QMURID(0000000026232)
URTYPE(XA)
USERID( )
APPLTAG(CSQ1CHIN)
ASID(0000)
APPLTYPE(CHINIT)
CHANNEL( )
CONNNAME( )
END CONN DETAILS
```

有关此消息中属性的说明，请参阅 [DISPLAY CONN](#) 命令的描述。

恢复不确定的恢复单元

要恢复不确定的恢复单元，请发出以下命令：

```
+CSQ1 RESOLVE INDOUBT( connection-name ) ACTION(COMMIT|BACKOUT)
NID( net-node.number )
```

其中:

connection-name

IMS 系统标识。

ACTION

指示是落实 (COMMIT) 还是回退 (BACKOUT) 此恢复单元。

net-node.number

关联的 *net-node.number*。

发出 RESOLVE INDOUBT 命令后, 将显示下列其中一条消息:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id BACKOUT SCHEDULED
```

解析残差恢复条目

在给定时间, IMS 会构建残差恢复条目 (RREs) 列表。RREs 是 IBM MQ 可能存在疑问的恢复单元。它们在以下几种情况下出现:

- 如果队列管理器未处于活动状态, 那么 IMS 具有在队列管理器处于活动状态之前无法解析的 RER。这些 RER 不是问题。
- 如果队列管理器处于活动状态并已连接到 IMS, 并且 IMS 回退了 IBM MQ 已落实的工作, 那么 IMS 适配器将发出消息 CSQQ010E。如果两个系统中的数据必须一致, 那么存在问题。有关解决此问题的信息, 请参阅第 362 页的『手动恢复 IMS 个恢复单元』。
- 如果队列管理器处于活动状态并连接到 IMS, 那么即使没有消息通知您此问题, 仍可能存在 RER。建立到 IMS 的 IBM MQ 连接后, 可以发出以下 IMS 命令以确定是否存在问题:

```
/DISPLAY OASN SUBSYS sysid
```

要清除 RRE, 请发出下列其中一个 IMS 命令:

```
/CHANGE SUBSYS sysid RESET
/CHANGE SUBSYS sysid RESET OASN nnnn
```

其中, *nnnn* 是响应 +CSQ1 DISPLAY 命令时列出的原始应用程序序号。这是程序实例的调度号, 它在自上次 IMS 冷启动以来该程序的调用序列中占有一席之地。IMS 不能有两个具有相同调度号的不确定恢复单元。

这些命令会重置 IMS 的状态; 它们不会导致与 IBM MQ 进行任何通信。

控制 IMS 从属区域连接

您可以控制, 监视以及在必要时终止 IMS 与 IBM MQ 之间的连接。

控制 IMS 从属区域连接涉及以下活动:

- [从从属区域进行连接](#)
- [区域错误选项](#)
- [监视连接上的活动](#)
- [与从属区域断开连接](#)

从从属区域连接

在控制区域中使用的 IMS 适配器也会装入到从属区域中。将建立从每个从属区域到 IBM MQ 的连接。此连接用于协调 IBM MQ 和 IMS 工作的落实。要初始化并建立连接, IMS 执行以下操作:

1. 从 IMS 读取子系统成员 (SSM)。PROCLIB。

可以在从属区域 EXEC 参数上指定子系统成员。如果未指定,那么将使用控制区域 SSM。如果该区域永远不可能连接到 IBM MQ,那么为了避免装入适配器,请指定没有条目的成员。

2. 装入 IBM MQ 适配器。

对于批处理消息程序,直到应用程序发出其第一个消息传递命令之后,才会执行装入。此时,IMS 会尝试建立连接。

对于消息处理程序区域或 IMS 快速路径区域,在初始化该区域时进行尝试。

区域错误选项

如果队列管理器未处于活动状态,或者从应用程序发送第一个消息传递命令时资源不可用,那么所执行的操作取决于在 SSM 条目上指定的错误选项。选项为:

R

将相应的返回码发送到应用程序。

Q

应用程序异常结束,异常结束代码为 U3051。输入消息将重新排队。

A

应用程序异常结束,异常结束代码为 U3047。将废弃输入消息。

监视连接上的活动

当应用程序发出其第一个成功的 IBM MQ 请求时,将从从属区域建立线程。您可以通过从 IBM MQ 发出以下命令来显示有关连接以及当前使用这些连接的应用程序的信息:

```
+CSQ1 DISPLAY CONN(*) ALL
```

该命令会生成类似如下的消息:

```

CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
TYPE(CONN)
CONNOPTS(
MQCNO_STANDARD_BINDING
)
UOWLOGDA(2004-12-15)
UOWLOGTI(16.39.43)
UOWSTDA(2004-12-15)
UOWSTTI(16.39.43)
UOWSTATE(ACTIVE)
NID( )
EXTURID(
0000052900000000
)
QMURID(00000354B76E)
URTYPE(IMS)
USERID(STCPI)
APPLTAG(IM8F)
ASID(0049)
APPLTYPE(IMS)
PSTID(0004)
PSBNAME(GP01MPP)

```

对于控制区域，*thread-xref* 是特殊值 CONTROL。对于从属区域，它是与 PSBname 并置的 PSTid。*auth-id* 是作业卡中的用户字段，或者是 z/OS 启动过程表中的标识。

有关所显示列表的说明，请参阅 [IBM MQ for z/OS 消息](#)，完成和原因码 文档中对消息 CSQV402I 的描述。

IMS 提供了用于监视与 IBM MQ 的连接显示命令。它显示在每个从属区域连接，LTERM 用户名和控制区域连接状态上哪个程序处于活动状态。命令为：

```
/DISPLAY SUBSYS name
```

IMS 与 IBM MQ 之间的连接状态显示为下列其中一项：

```

CONNECTED
NOT CONNECTED
CONNECT IN PROGRESS
STOPPED
STOP IN PROGRESS
INVALID SUBSYSTEM NAME= name
SUBSYSTEM name NOT DEFINED BUT RECOVERY OUTSTANDING

```

来自每个从属区域的线程状态为下列其中一项：

```

CONN
CONN, ACTIVE (includes LTERM of user)

```

与从属区域断开连接

更改 IMS 的 SSM 成员中的值。PROCLIB，断开从属区域的连接。要执行此操作，您必须：

1. 发出 IMS 命令：

```
/STOP REGION
```

2. 更新 SSM 成员。
3. 发出 IMS 命令:

```
/START REGION
```

z/OS 断开连接 IMS

当 IMS 或队列管理器终止时，将结束连接。或者，IMS 主终端操作员可以显式断开连接。

要终止 IMS 与 IBM MQ 之间的连接，请使用以下 IMS 命令:

```
/STOP SUBSYS sysid
```

该命令将以下消息发送到发出该消息的终端，通常是主终端操作员 (MTO):

```
DFS058I STOP COMMAND IN PROGRESS
```

IMS 命令:

```
/START SUBSYS sysid
```

需要重新建立连接。

注: 如果 IMS 触发器监视器正在运行，那么 IMS 命令 /STOP SUBSYS 未完成。

z/OS 控制 IMS 触发器监视器

您可以使用 CSQQTRMN 事务来停止并启动 IMS 触发器监视器。

IMS 触发器监视器 (CSQQTRMN 事务) 在 [设置 IMS 触发器监视器](#) 中进行了描述。

要控制 IMS 触发器监视器，请参阅:

- [启动 CSQQTRMN](#)
- [停止 CSQQTRMN](#)

正在启动 CSQQTRMN

1. 启动面向批处理的 BMP，该 BMP 针对要监视的每个启动队列运行程序 CSQQTRMN。
2. 修改批处理 JCL 以添加 DDname CSQQUT1，该 DDname 指向包含以下信息的数据集:

```
QMGRNAME=q_manager_name      Comment: queue manager name
INITQUEUEUENAME=init_q_name   Comment: initiation queue name
LTERM=lterm                   Comment: LTERM to remove error messages
CONSOLEMESSAGES=YES          Comment: Send error messages to console
```

其中:

q_manager_name 队列管理器的名称 (如果此名称为空, 那么将采用 CSQQDEFV 中指定的缺省值)

init_q_name 要监视的启动队列的名称

lterm 错误消息目标的 IMS LTERM 名称 (如果为空, 那么缺省值为 MASTER)。

CONSOLEMESSAGES= YES 还会将发送到指定 IMS LTERM 的消息的请求发送到 z/OS 控制台。如果此参数被省略或拼写错误, 那么缺省值为 NOT 将消息发送到控制台。

3. 如果需要处理 CSQQUT1 输入的打印报告, 请添加 DD 名称 CSQQUT2 。

注:

1. 数据集 CSQQUT1 是使用 LRECL=80 定义的。其他 DCB 信息取自数据集。数据集 CSQQUT2 的 DCB 为 RECFM=VBA 和 LRECL=125。
2. 您只能在每条记录上放置一个关键字。关键字值由关键字后面的第一个空格定界; 这意味着您可以包含注释。列 1 中的星号表示整个输入记录都是注释。
3. 如果您对 QMGRNAME 或 LTERM 关键字进行了拼写错误, 那么 CSQQTRMN 将使用该关键字的缺省值。
4. 在提交触发器监视器 BMP 作业之前, 请确保子系统已在 IMS 中启动 (通过 /START SUBSYS 命令)。如果未启动, 那么触发器监视器作业将终止, 异常终止代码为 U3042。

正在停止 CSQQTRMN

一旦启动, CSQQTRMN 就会运行, 直到由于下列其中一个事件导致 IBM MQ 和 IMS 之间的连接中断为止:

- 队列管理器结束
- IMS 结束

或输入 z/OS STOP **jobname** 命令。

z/OS 控制 IMS 网桥

使用本主题来了解可用于控制 IMS 网桥的 IMS 命令。

没有用于控制 IBM MQ-IMS 网桥的 IBM MQ 命令。但是, 您可以通过以下方式停止将消息传递到 IMS:

- 对于非共享队列, 通过对所有网桥队列使用 ALTER QLOCAL (xxx) GET (DISABLED) 命令。
- 对于集群队列, 使用 SUSPEND QMGR CLUSTER (xxx) 命令。仅当另一个队列管理器也在托管集群网桥队列时, 此操作才有效。
- 对于集群队列, 使用 SUSPEND QMGR FACILITY (IMSBRIDGE) 命令。不会向 IMS 发送更多消息, 但会从 IMS 接收任何未完成事务的响应。

要再次开始向 IMS 发送消息, 请发出 RESUME QMGR FACILITY (IMSBRIDGE) 命令。

您还可以使用 MQSC 命令 DISPLAY SYSTEM 来显示网桥是否已暂挂。

请参阅 [MQSC 命令](#) 以获取这些命令的详细信息。

有关更多信息, 请参阅:

- [第 373 页的『启动和停止 IMS 网桥』](#)
- [第 373 页的『控制 IMS 连接』](#)
- [控制网桥队列](#)

- [第 374 页的『再同步 IMS 网桥』](#)
- [使用 tpipe 名称](#)
- [从 IMS 中删除消息](#)
- [删除 tpipe](#)
- [第 376 页的『IMS 事务到期』](#)

启动和停止 IMS 网桥

通过启动 OTMA 来启动 IBM MQ 网桥。使用 IMS 命令:

```
/START OTMA
```

或者通过在 IMS 系统参数中指定 OTMA=YES 来自动将其启动。如果 OTMA 已启动，那么当队列管理器启动完成时，网桥将自动启动。当 OTMA 启动时，将生成 IBM MQ 事件消息。

使用 IMS 命令:

```
/STOP OTMA
```

以停止 OTMA 通信。发出此命令时，将生成 IBM MQ 事件消息。

控制 IMS 连接

IMS 提供了以下操作程序命令来控制 and 监视与 IBM MQ 的连接:

/DEQUEUE TMEMBER *tmember* TPIPE *tpipe*

从 Tpipe 中除去消息。指定 PURGE 以除去所有消息，或指定 PURGE1 以仅除去第一条消息。

/DISPLAY OTMA

显示有关 OTMA 服务器和客户机以及客户机状态的摘要信息。

/DISPLAY TMEMBER 名称

显示有关 OTMA 客户机的信息。

/DISPLAY TRACE TMEMBER 名称

显示有关正在跟踪的内容的信息。

/SECURE OTMA

设置安全选项。

/START OTMA

支持通过 OTMA 进行通信。

/START TMEMBER *tmember* TPIPE *tpipe*

启动指定的 Tpipe。

/STOP OTMA

通过 OTMA 停止通信。

/STOP TMEMBER *tmember* TPIPE *tpipe*

停止指定的 Tpipe。

/TRACE

控制 IMS 跟踪。

有关这些命令的更多信息，请参阅 *IMS/ESA Operators Reference* 手册以了解您正在使用的 IMS 级别。

IMS 命令响应将发送到从中发出命令的终端。发出 IMS 命令的授权基于 IMS 安全性。

控制网桥队列

要停止通过网桥与具有 XCF 成员名 *tmember* 的队列管理器通信, 请发出以下 IMS 命令:

```
/STOP TMEMBER tmember TPIPE ALL
```

要恢复通信, 请发出以下 IMS 命令:

```
/START TMEMBER tmember TPIPE ALL
```

可以使用 MQ DISPLAY QUEUE 命令显示队列的 Tpipe。

要在单个 Tpipe 上停止与队列管理器的通信, 请发出以下 IMS 命令:

```
/STOP TMEMBER tmember TPIPE tpipe
```

将为每个活动网桥队列创建一个或两个 Tpipe, 因此发出此命令将停止与 IBM MQ 队列的通信。要恢复通信, 请使用以下 IMS 命令:

```
/START TMEMBER tmember TPIPE tpipe
```

或者, 您可以更改 IBM MQ 队列的属性以使其被禁止。

再同步 IMS 网桥

每当重新启动队列管理器 IMS 或 OTMA 时, 都会自动重新启动 IMS 网桥。

IMS 网桥执行的第一个任务是与 IMS 进行再同步。这涉及 IBM MQ 和 IMS 检查每个同步的 Tpipe 上的序号。当使用落实方式 0 (commit-then-send) 将持久消息从 IBM MQ - IMS 网桥队列发送到 IMS 时, 将使用同步的 Tpipe。

如果网桥无法与 IMS 再同步, 那么将在消息 CSQ2023E 中返回 IMS 检测代码, 并且将停止与 OTMA 的连接。如果网桥无法与单个 IMS Tpipe 再同步, 那么将在消息 CSQ2025E 中返回 IMS 检测代码, 并停止 Tpipe。如果 Tpipe 已冷启动, 那么可恢复的序号将自动重置为 1。

如果网桥在与 Tpipe 再同步时发现不匹配的序号, 那么将发出消息 CSQ2020E。使用 IBM MQ 命令 RESET TPIPE 来启动与 IMS Tpipe 的再同步。您需要提供 XCF 组和成员名以及 Tpipe 的名称; 此信息由消息提供。

您还可以指定:

- 要在 Tpipe 中为 IBM MQ 发送的消息设置新的可恢复序号, 并将其设置为合作伙伴的接收序号。如果未指定此值, 那么合作伙伴的接收序号将设置为当前 IBM MQ 发送序号。
- 要在 Tpipe 中为 IBM MQ 接收的消息设置新的可恢复序号, 并将其设置为合作伙伴的发送序号。如果未指定此值, 那么合作伙伴的发送序号将设置为当前 IBM MQ 接收序号。

如果存在与 Tpipe 关联的未解析恢复单元, 那么也会在消息中通知该恢复单元。使用 IBM MQ 命令 RESET TPIPE 来指定是落实恢复单元, 还是将其回退。如果落实恢复单元, 那么消息批处理已发送到 IMS, 并且将从网桥队列中删除。如果将恢复单元回退, 那么会将消息返回到网桥队列, 以便稍后将消息发送到 IMS。

未同步落实方式 1 (send-then-commit) Tpipe。

落实方式 1 事务的注意事项

在 IMS 中，落实方式 1 (CM1) 事务在同步点之前发送其输出应答。

CM1 事务可能无法发送其应答，例如：

- 要发送应答的 Tpipe 已停止
- OTMA 已停止
- OTMA 客户机 (即，队列管理器) 已消失
- 应答队列和死信队列不可用

由于这些原因，IMS 应用程序发送消息伪异常终止，代码为 U0119。在这种情况下，不会停止 IMS 事务和程序。

这些原因通常会阻止将消息发送到 IMS 以及从 IMS 传递应答。在下列情况下，可能会发生 U0119 异常终止：

- 在消息位于 IMS 中时，Tpipe，OTMA 或队列管理器已停止
- IMS 在另一 Tpipe 上回复入局消息，并且 Tpipe 已停止
- IMS 会回复到其他 OTMA 客户机，并且该客户机不可用。

每当发生 U0119 异常终止时，到 IMS 的入局消息和到 IBM MQ 的应答消息都将丢失。如果由于任何这些原因无法交付 CM0 事务的输出，那么它将在 IMS 内的 Tpipe 上排队。

使用 tpipe 名称

用于控制 IBM MQ - IMS 网桥的许多命令都需要 *tpipe* 名称。使用本主题来了解如何查找 *tpipe* 名称的更多详细信息。

对于控制 IBM MQ - IMS 网桥的许多命令，您需要 *tpipe* 名称。您可以从 DISPLAY QUEUE 命令获取 *tpipe* 名称并注意以下几点：

- 在定义本地队列时分配 *tpipe* 名称
- 为本地队列提供了两个 *tpipe* 名称，一个用于同步，另一个用于非同步
- 直到在特定于该特定本地队列的 IMS 和 IBM MQ 之间进行了某些通信之后，IMS 才会知道 *tpipe* 名称
- 要使 *tpipe* 可供 IBM MQ - IMS 网桥使用，必须将其关联队列分配给已完成正确 XCF 组和成员名字段的存储类

从其中删除消息 IMS

如果 Tmember/Tpipe 已停止，那么可以删除通过 IMS 网桥发往 IBM MQ 的消息。要删除具有 XCF 成员名 *tmember* 的队列管理器的一条消息，请发出以下 IMS 命令：

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE1
```

要删除 Tpipe 上的所有消息，请发出以下 IMS 命令：

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE
```

删除 tpipe

您无法自行删除 IMS *tpipe*。IMS 将在以下时间将其删除：

- 当 IMS 处于冷启动状态时，将删除同步的 *tpipe*。

- 重新启动 IMS 时，将删除未同步的 tpipe。

IMS 事务到期

到期时间与事务关联; 任何 IBM MQ 消息都可以具有与其关联的到期时间。使用 MQMD.Expiry 字段将到期时间间隔从应用程序传递到 IBM MQ。时间是消息到期前的持续时间，以十分之一秒为单位的值表示。尝试执行消息的 MQGET (晚于其已到期) 会导致从队列中除去消息并执行到期处理。到期时间随着 IBM MQ 网络上队列管理器之间的消息流而缩短。当 IMS 消息通过 IMS 网桥传递到 OTMA 时，剩余的消息到期时间将作为事务到期时间传递到 OTMA。

如果事务具有指定的到期时间，那么 OTMA 将在 IMS 中的三个不同位置使输入事务到期:

- 从 XCF 接收输入消息
- 输入消息排队时间
- 应用程序 GU 时间

在 GU 时间之后不会执行到期。

事务 EXPRTIME 可由以下人员提供:

- IMS 事务定义 (transaction definition)
- IMS OTMA 消息头
- IMS DFSINSXO 用户出口
- IMS CREATE 或 UPDATE TRAN 命令

IMS 指示它已通过异常终止具有 0243 的事务并发出消息来使事务到期。发出的消息是非共享队列环境中的 DFS555I 或共享队列环境中的 DFS2224I。

z/OS 在 z/OS 上操作 Advanced Message Security

Advanced Message Security 地址空间接受使用 z/OS MODIFY 命令的命令。

要输入 Advanced Message Security 地址空间的命令，请使用 z/OS MODIFY 命令。

例如，

```
F qmgr AMSM, cmd
```

接受以下 MODIFY 命令:

| 命令 | 选项 | 描述 |
|----------|----------------------|-------------------------------------------------------|
| DISPLAY | | 显示版本信息 |
| 刷新 | keyRing 策略 ALL | 刷新密钥环证书和/或安全策略。 |
| SMFAUDIT | SUCCESS 失败 ALL | 设置在 AMS 成功保护/取消保护消息和/或 AMS 未能保护/取消保护消息时是否需要 SMF 审计。 |
| SMFTYPE | 0 - 255 | 设置要在 AMS 保护/取消保护消息时生成的 SMF 记录类型。要禁用 SMF 审计，请指定记录类型 0。 |

注: 要指定选项，必须以逗号分隔。例如:


```
F qmgIAMS,REFRESH KEYRING
F qmgIAMS,SMFAUDIT ALL
F qmgIAMS,SMFTYPE 180
```

REFRESH 命令。

发出 MQOPEN 调用的应用程序将执行更改。现有应用程序将继续使用该应用程序打开队列时提供的选项。要选取更改，应用程序必须关闭并重新打开队列。

V 9.0.1 z/OS z/OS Connect 的 IBM MQ for z/OS 服务提供者

IBM MQ for z/OS Service Provider for z/OS Connect (MQ Service Provider) 处理通过 z/OS Connect 到达的请求。MQ Service Provider 允许 REST 感知应用程序与 z/OS 资产进行交互，这些资产是使用 IBM MQ for z/OS 队列和主题公开的。您可以实现这一点，而不必担心使用异步消息传递所需的编码。

要点: z/OS Connect EE V 3.0.21.0 和更高版本提供了支持服务归档文件的 MQ Service Provider 增强版本。您应该迁移到该版本的 z/OS Connect EE 并使用内置 MQ Service Provider，而不是使用 IBM MQ for z/OS 产品随附的服务提供者。

在 z/OS IBM Documentation 中的 Connect EE 信息中，z/OS Connect EE 中 MQ Service Provider 的快速启动方案在 [快速启动方案](#) 下提供，详细参考信息在 [使用 IBM MQ 服务提供者](#) 下提供。

本部分由以下信息组成：

相关信息

视频: [IBM MQ Service Provider for z/OS Connect \(YouTube\)](#)

V 9.0.1 z/OS IBM MQ for z/OS Service Provider for z/OS Connect-概述

IBM MQ for z/OS Service Provider for z/OS Connect 概述，说明服务提供程序使用的原则和动词。

要点: z/OS Connect EE V 3.0.21.0 和更高版本提供了支持服务归档文件的 MQ Service Provider 增强版本。您应该迁移到该版本的 z/OS Connect EE 并使用内置 MQ Service Provider，而不是使用 IBM MQ for z/OS 产品随附的服务提供者。

在 z/OS IBM Documentation 中的 Connect EE 信息中，z/OS Connect EE 中 MQ Service Provider 的快速启动方案在 [快速启动方案](#) 下提供，详细参考信息在 [使用 IBM MQ 服务提供者](#) 下提供。

MQ Service Provider 支持两个版本的 z/OS Connect:

IBM z/OS Connect V 1 (z/OS Connect V1)

这是 z/OS 上 WebSphere Liberty Profile (WLP) 的组件，免费提供，将 z/OS 资产作为 REST 接口公开，并允许远程应用程序 (例如在移动设备上运行的应用程序) 向其发送使用 JSON 格式化的数据。

有关更多信息，请参阅 [IBM z/OS Connect 概述](#)。

要点:

1. IBM z/OS Connect V1 的 MQ Service Provider 需要 IBM WebSphere MQ 7.5 资源适配器。服务结束日期为 2018 年 4 月 30th。

在此之后，IBM z/OS Connect V1 将不支持 MQ Service Provider，您应该改为使用 IBM z/OS Connect EE。

因此，IBM z/OS Connect V1 更适合用于开发和概念证明目的，而不是用于生产环境。

2. MQ Service Provider 仅支持 WLP 8.5.5.9 和更高版本中提供的 z/OS Connect V1 代码。

IBM z/OS Connect Enterprise Edition (z/OS Connect EE)

这是一个单独的收费产品，基于 WLP，它提供了 IBM z/OS Connect V1 中的所有功能，以及许多增强功能，包括用于生成 RESTful API 的工具 (API 编辑器) 以及与 IBM API Connect 的集成。

有关更多信息，请参阅 [IBM z/OS Connect EE](#)。

要点: MQ Service Provider 仅支持 z/OS Connect EE V2.0.3.0 (APAR [PI66869](#)) 或更高版本。

无论使用哪个版本的 z/OS Connect，MQ Service Provider 都提供相同的功能。MQ Service Provider 支持 z/OS Connect EE 的所有功能，例如 API 编辑器。

在整个文档中，术语 *z/OS Connect* 用于引用两个版本，除非需要引用特定版本，在这种情况下，将使用先前文本中定义的其中一个名称。

z/OS Connect 公开的 z/OS 资产不需要了解 REST 或 JSON，因为 z/OS Connect 会将 REST 调用映射到本地调用，并且还会提供变换以在 JSON 和本地数据结构 (例如 COBOL 副本) 之间进行转换。

MQ Service Provider 作为单独的 WLP 功能部件提供，可安装到相关的 z/OS Connect 安装中。请注意，设置 MQ Service Provider 的大部分工作都基于配置，因此 REST 应用程序不需要知道 IBM MQ。但是，针对更高级的应用程序提供了对诸如 MQMD 字段之类的项的访问权。

MQ Service Provider 将 IBM MQ 队列和主题显示为 [第 378 页的『服务』](#)。

IBM MQ 的受支持版本

IBM MQ for z/OS 8.0 和更高版本支持 MQ Service Provider。

低于 IBM MQ 9.0.1 的版本将不会随 MQ Service Provider 一起提供。请改为参阅 [第 381 页的『获取 MQ Service Provider』](#)，以获取有关如何获取 MQ Service Provider 的信息。

z/OS 连接 IBM Documentation 部分中描述的文档同样适用于 IBM MQ 的所有受支持版本。

其他注意事项

MQ Service Provider 使用：

- 用于非破坏性 MQGET 调用的 HTTP GET
- 用于破坏性 MQGET 调用的 HTTP DELETE
- MQPUT 调用的 HTTP POST

这些是 IBM MQ Bridge for HTTP 使用的相同动词。同样，MQ Service Provider 使用 HTTP 头作为指定各种内容 (例如 MQMD 值) 的方法。

有关 HTTP 调用的更多信息，请参阅 [第 380 页的『MQ Service Provider 支持的动词』](#)

请注意，表示为 JSON 数据的唯一数据是消息的有效内容。这意味着，如果应用程序需要的都是消息数据，那么它们不会不必要地向 IBM MQ 概念公开。

MQ Service Provider 使用 WLP 中的 IBM MQ 资源适配器支持，因此基于 JMS。

V 9.0.1 z/OS 服务

MQ Service Provider 将 IBM MQ 队列和主题以及后面的应用程序作为服务公开。本部分描述了两种类型的服务：单向和双向。

单向服务

单向服务可用于在单个 IBM MQ 队列或主题上提供 RESTful API。RESTful 客户机可以向单向服务发出带有 JSON 有效内容的 HTTP POST，并且服务会获取有效内容并将有效内容作为消息体的消息发送到目标队列或主题。

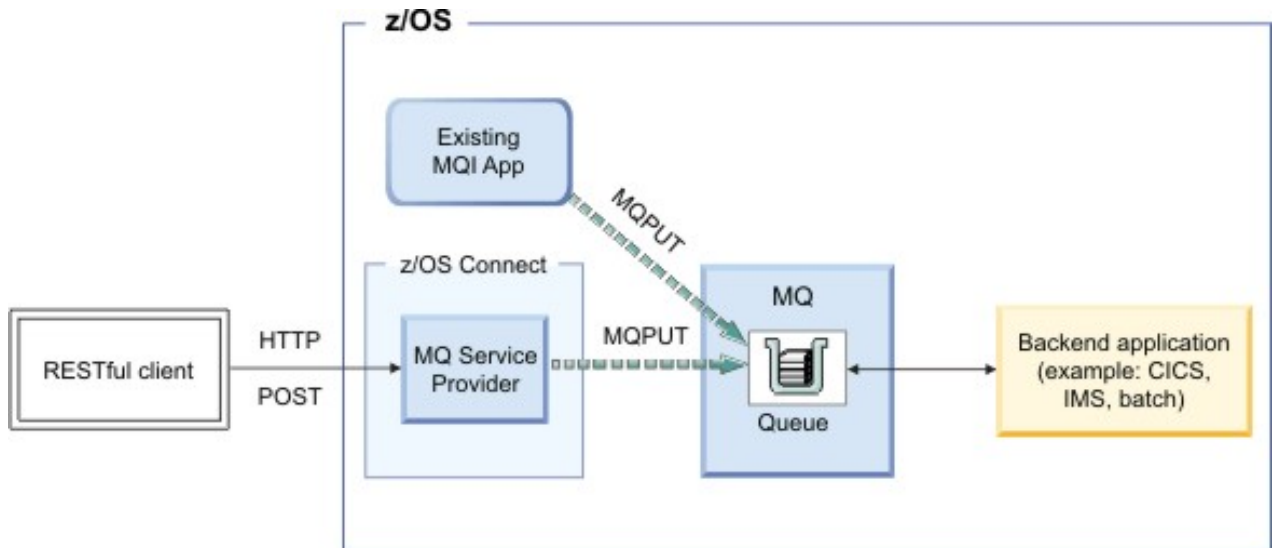


图 55: 单向服务

单向服务还允许针对 IBM MQ 队列发出 HTTP DELETE 和 HTTP GET 请求。

HTTP DELETE 会导致从队列中破坏性获取可用消息。HTTP GET 会从队列中浏览第一条可用消息。

注: 除非发生其他一些操作 (例如 HTTP DELETE 或消息到期) 以从队列中除去消息, 否则两个 HTTP GET 调用将返回同一消息。

消息主体将以 JSON 格式返回到 RESTful 客户机。如果消息主体尚未采用 JSON 格式 (例如, COBOL 副本), 那么可以使用数据变换将 z/OS Connect 配置为将其转换为 JSON。

双向服务

双向服务允许 RESTful 客户机对一对队列执行请求/应答消息传递。

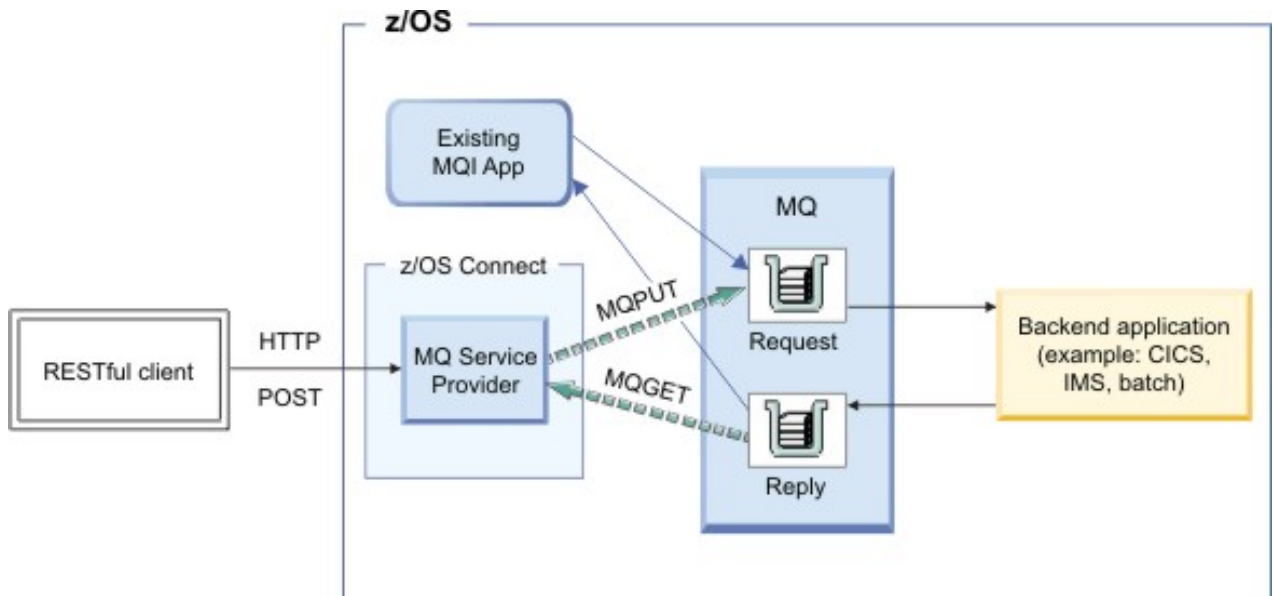


图 56: 双向服务

客户机发出 HTTP POST 请求, 指定 JSON 有效内容。服务会获取有效内容, (可选) 将其转换为其他格式 (例如, COBOL 副本), 并将其作为消息发送到请求队列。

后端应用程序使用消息, 对其进行处理, 并生成放在应答队列上的响应。服务查找此消息, 获取其有效内容, (可选) 将其转换为 JSON, 并将其作为 HTTP POST 的响应主体返回。

MQ Service Provider 支持 IBM MQ Bridge for HTTP 中使用的 HTTP GET，HTTP DELETE 和 HTTP POST 动词，其含义相同。使用这些动词允许与 IBM MQ for z/OS 进行基本交互，并且这些动词的组合允许显示更复杂的函数。

执行 HTTP GET 或 HTTP DELETE 时，MQ Service Provider 将忽略请求中提供的任何主体。

如果 HTTP GET 或 HTTP DELETE 成功：

- 在 HTTP 响应主体中返回消息有效内容，HTTP 状态码为 200。
- 返回没有有效内容的消息，但没有 HTTP 主体。如果没有可用的消息，那么不会返回 HTTP 主体，并且 HTTP 状态码为 204 (无内容)。



注意：z/OS Connect 期望 JSON 有效内容始终采用对象格式；即，最小有效有效内容为 {}。

执行 HTTP POST 时，唯一期望的参数是消息有效内容 (将在 HTTP 主体中) 以及 [mqzOSConnect 服务元素](#) 中描述的各种元素。

如果 POST 成功，请执行以下操作：

单向服务

返回 HTTP 204 (No Content) 代码以及空有效内容。

双向服务

返回 JSON 格式的应答消息内容，或者如果没有可用的应答消息，那么返回 HTTP 响应代码为 204 (无内容) 的空响应主体。

HTTP 头

向单向服务或双向服务发出 HTTP POST 时，HTTP 主体必须是有效的 JSON 对象。必须指定以下内容：

- 到 HTTP_POST 的 Content-Type=application/json HTTP 头
- UTF-8 的编码

根据所使用的 HTTP 动词，还可以指定许多可选 HTTP 头来调整 MQ Service Provider 的行为。有关更多详细信息，请参阅 [可与 MQ Service Provider 配合使用的 HTTP 头](#)。

HTTP 代码

如果 MQ Service Provider 检测到错误，那么服务会设置 400-599 范围内的 HTTP 状态码。否则，将始终返回先前记录的状态码。

客户端错误

如果调用应用程序传入了无效数据，那么会向调用者返回 400-499 范围内的 HTTP 状态码。确切的状态码取决于错误。

如果抛出的异常检测到无效数据，那么将提供 JSON 格式的堆栈跟踪，如服务器端示例中所示。

服务器端错误

如果 MQ Service Provider 迁到意外错误，那么堆栈跟踪将转换为 JSON 格式，并以 HTTP 状态码 500 (内部服务器错误) 返回给调用者。适当的诊断也将写入 z/OS Connect 日志。

简化的错误响应有效内容示例为：

```
{
  "errorText": "CSQZ0006E: An unexpected JMSEException occurred while processing a request
for the 'mq7' service. ",
  "exceptionData": {
    "exceptions": [
      {
        "stackTrace": [
          "com.ibm.msg.client.jms.DetailedInvalidDestinationException: JMSWMQ2008:
Failed to open MQ queue 'ThisQueueDoesNotExist'.",
          <further content removed for brevity>
        ]
      }
    ]
  }
},
```

```

    {
      "stackTrace": [
        "com.ibm.mq.MQException: JMSMQ0001: WebSphere MQ call failed with
        compcode '2' ('MQCC_FAILED') reason '2085' ('MQRC_UNKNOWN_OBJECT_NAME').",
        <further content removed for brevity>
      ]
    }
  ]
}
}
}
}

```

V 9.0.1 z/OS 获取 MQ Service Provider

支持对 IBM MQ for z/OS 8.0 和更高版本使用 MQ Service Provider。有两种方法可获取 MQ Service Provider 以用于任何受支持的 IBM MQ 版本。

1. 第 381 页的『从 IBM MQ 9.0.1 (或更高版本) 的 Unix 系统服务组件功能部件获取 MQ Service Provider』
2. 第 381 页的『从 Fix Central 获取 MQ Service Provider』

从 IBM MQ 9.0.1 (或更高版本) 的 Unix 系统服务组件功能部件获取 MQ Service Provider

按照程序目录中的指示信息安装功能部件。MQ Service Provider 将位于文件系统上的 *PATHPREFIX/zosconnect* 目录中，其中 *PATHPREFIX* 是安装时选择的值。缺省情况下，对于 IBM MQ 9.0.1，*PATHPREFIX* 为 */usr/lpp/mqm/V9R0M1*。

从 Fix Central 获取 MQ Service Provider

转至 [Fix Central](#)，并通过搜索 *IBM-MQ-zOSConnect-ServiceProvider* 来找到 MQ Service Provider。

这应该会找到可以下载到工作站的 tar 文件，例如 *9.0.1-IBM-MQ-zOSConnect-ServiceProvider.tar*。



注意: 同一 tar 文件也与 IBM MQ 8.0 配合使用。IBM MQ 8.0 没有单独的 tar 文件。

下载后，将 tar 文件传输到相应 z/OS LPAR 上的目录 *PATHPREFIX*。此目录应该是可供 z/OS Connect 访问的位置。

通过发出以下命令来解压缩文件：

```
tar -oxvf 9.0.1-IBM-MQ-zOSConnect-ServiceProvider.tar
```

根据您的企业，更改目录结构的许可权。

目录结构

两种获取 MQ Service Provider 的方法都将创建相同的目录结构。此结构的根目录在本文档中称为 *MQSP_ROOT*，其中 *MQSP_ROOT* 是 *PATHPREFIX/zosconnect*。

在 *MQSP_ROOT* 下有以下文件和目录：

| | |
|---------------------------------|------------------------------------------------------|
| mqzosconnect.properties | Properties file that can be copied into z/OS Connect |
| v1.0/ | Directory containing MQ Service Provider for z/OS |
| Connect V1 | |
| lib/ | |
| com.ibm.mq.zosconnect_1.0.0.jar | Feature jar file for MQ Service Provider for z/OS |
| Connect V1 | |
| features/ | |
| zosConnectMQ-1.0.mf | Feature manifest for MQ Service Provider for z/OS |
| Connect V1 | |
| v2.0/ | Directory containing MQ Service Provider for z/OS |
| Connect EE | |
| lib/ | |
| com.ibm.mq.zosconnect_2.0.0.jar | Feature jar file for MQ Service Provider for z/OS |
| Connect EE | |

V 9.0.1 z/OS 事务注意事项

HTTP 不是事务协议，因此无法对 MQ Service Provider 执行的消息传递操作进行事务协调。

这将产生以下影响：

- 如果对单向服务执行 HTTP POST，并且在客户机接收到 HTTP 响应之前连接失败，那么客户机无法立即告知消息是否已发送到配置的队列或主题。
- 如果对单向服务执行 HTTP DELETE，并且在客户机接收到 HTTP 响应之前连接失败，那么消息可能已从队列中破坏性地获取并丢失，因为无法回滚破坏性获取。
- 如果对双向服务执行 HTTP POST，并且在客户机接收到 HTTP 响应之前连接失败，那么客户机无法告知发生故障的位置。请求消息可能已发送到请求队列，或者响应消息可能已从响应队列获取并丢失。
- 无法将多个 HTTP 动词的结果协调到单向或双向服务。

V 9.0.1 z/OS IBM z/OS Connect EE - MQ Service Provider 设置过程

要在 IBM z/OS Connect EE 上设置 MQ Service Provider，需要安装的组件的概述。

关于此任务

按照显示的顺序执行以下任务，以安装 MQ Service Provider 及其所有必备软件。

V 9.0.1 z/OS 安装 IBM z/OS Connect EE

安装 IBM z/OS Connect EE 的需求和过程。

开始之前

确保您有权 [更新 z/OS Connect EE V2 嵌入式 WLP](#)。

关于此任务

此过程设置适合与 MQ Service Provider 配合使用的 z/OS Connect EE 服务器。如果您已具有以可代替使用的相同方式设置的服务器。

过程

1. 使用以下描述的指示信息创建服务器: [创建 z/OS Connect EE Server](#)
2. 通过遵循 [配置 Liberty Angel 进程和 z/OS 授权服务](#) 中详细描述的过程来启用 TXRRS 授权服务。
3. 通过以下方法验证 TXRRS 授权服务是否已正确设置:
 - a) 启动服务器。
遵循 [启动和停止 z/OS Connect EE](#) 中详细描述的过程
 - b) 查看位于以下位置的服务器日志:

```
/var/zosconnect/servers/server_name/logs/messages.log
```

这些是 ASCII 文件，应该包含类似于以下内容的输出：

```
A CWWKE0001I: The server test has been launched.
I CWWKB0103I: Authorized service group LOCALCOM is available.
I CWWKB0103I: Authorized service group PRODMGR is available.
I CWWKB0103I: Authorized service group SAFCRE is available.
I CWWKB0103I: Authorized service group TXRRS is available.
I CWWKB0103I: Authorized service group WOLA is available.
I CWWKB0103I: Authorized service group ZOSDUMP is available.
```

```
I CWWKB0103I: Authorized service group ZOSWLM is available.
I CWWKB0103I: Authorized service group CLIENT.WOLA is available.
I CWWKB0108I: IBM CORP product z/OS Connect version 02.00 successfully registered with z/OS
```

检查输出以查看 TXRRS 授权服务组是否可用。在前面的示例中，粗体文本中的行显示服务组可用。

MQ Service Provider 需要此服务组才能使用

c) 正在停止服务器。

遵循 [从命令行启动和停止服务器](#) 中详细描述的过程

结果

您已成功安装 z/OS Connect EE。

下一步做什么

安装 IBM MQ 资源适配器。

相关任务

第 382 页的『[IBM z/OS Connect EE - MQ Service Provider 设置过程](#)』

要在 IBM z/OS Connect EE 上设置 MQ Service Provider，需要安装的组件的概述。

第 383 页的『[安装 IBM MQ 资源适配器](#)』

IBM z/OS Connect EE 中的 IBM MQ 消息传递提供程序功能部件使用称为 IBM MQ 资源适配器的 IBM MQ 组件。资源适配器作为 IBM MQ for z/OS Unix 系统服务组件功能部件的一部分随 IBM MQ 提供。

V 9.0.1

z/OS

安装 IBM MQ 资源适配器

IBM z/OS Connect EE 中的 IBM MQ 消息传递提供程序功能部件使用称为 IBM MQ 资源适配器的 IBM MQ 组件。资源适配器作为 IBM MQ for z/OS Unix 系统服务组件功能部件的一部分随 IBM MQ 提供。

开始之前

确保已安装要连接到的队列管理器版本的 IBM MQ for z/OS Unix 系统服务组件功能部件。

如果要连接到多个版本的队列管理器，那么应使用最新版本中的功能部件。

关于此任务

此过程将 IBM MQ 资源适配器安装到 IBM z/OS Connect EE 中。

过程

1. 找到要连接到的 IBM MQ 版本的 Unix 系统服务组件目录。

例如，此目录可能称为 `/usr/lpp/mqm/V9R0M1/`。该目录应包含 `java/lib` 子目录，其中包含大量本机库（.so 文件）。



注意: 如果要连接到多个版本的队列管理器，请使用最新版本。

2. 编辑 z/OS Connect EE 服务器的 `server.xml` 文件。

添加下列行：

```
<variable name="wmqJmsClient.rar.location"
  value="MQJAVA_LIB_DIR/jca/wmq.jmsra.rar"/>
<wmqJmsClient nativeLibraryPath="MQJAVA_LIB_DIR"/>
```

其中 `MQJAVA_LIB_DIR` 基于步骤 [第 383 页的『1』](#) 中的目录，例如 `/usr/lpp/mqm/V9R0M1/java/lib`。

b. 保存更改。

第一行告诉 IBM z/OS Connect EE 在何处找到 IBM MQ 资源适配器。

第二行告诉 IBM MQ 资源适配器在何处查找用于与 IBM MQ 的绑定连接的本机库。

有关更多信息，请参阅 [wmqJmsClient](#) 和 [将 JMS 应用程序部署到 Liberty 以使用 IBM MQ 消息传递提供程序](#)。

3. 设置服务器的 STEPLIB 以包含 IBM MQ 库。

必须执行此操作，以便位于步骤 [第 383 页](#) 的『1』中的本机库可以连接到队列管理器。

通常，这涉及编辑用于启动服务器的 JCL 以包含以下内容：

```
//STEPLIB DD DSN=HLQ.SCSQAUTH,DISP=SHR  
// DD DSN=HLQ.SCSQANLE,DISP=SHR
```

其中 *HLQ* 是包含 IBM MQ 安装的数据集的高级限定符。

如果要从同一服务器连接到 IBM MQ 的多个版本，请使用最新版本中的数据集。

结果

您已部分安装 IBM MQ 资源适配器。

请注意，直到启用其父功能部件 (wmqJmsClient-2.0) 之后，才会完全安装 IBM MQ 资源适配器，如 [第 385 页](#) 的『启用 z/OS Connect 和 MQ Service Provider』中所述。

下一步做什么

将 MQ Service Provider 安装到 IBM z/OS Connect EE 中。

相关任务

[第 382 页](#) 的『IBM z/OS Connect EE - MQ Service Provider 设置过程』

要在 IBM z/OS Connect EE 上设置 MQ Service Provider，需要安装的组件的概述。

[第 382 页](#) 的『安装 IBM z/OS Connect EE』

安装 IBM z/OS Connect EE 的需求和过程。

[第 384 页](#) 的『将 MQ Service Provider 安装到 IBM z/OS Connect EE 中』

在使用 MQ Service Provider 之前，必须将其安装到 IBM z/OS Connect EE 中。

将 MQ Service Provider 安装到 IBM z/OS Connect EE 中

在使用 MQ Service Provider 之前，必须将其安装到 IBM z/OS Connect EE 中。

关于此任务

执行以下过程来执行此操作：

过程

1. 找到 IBM z/OS Connect EE 产品扩展目录。

对于典型安装，这是此处描述的 `/var/zosconnect/v2r0/extensions` 目录：[设置产品扩展目录](#)。

2. 将 `MQSP_ROOT/mqzosconnect.properties` 复制到步骤 [第 384 页](#) 的『1』中的目录。

3. 编辑复制的文件。

请注意，此文件是 ASCII 码文件。将 `PATH_TO_INSTALL` 更改为 `MQSP_ROOT/v2.0>`，并保存更改。



注意：步骤 [第 384 页](#) 的『3』假定运行服务器的用户标识对 `MQSP_ROOT` 目录结构具有读访问权。如果情况并非如此，请添加对用户标识的足够访问权，或者将 `MQSP_ROOT` 的内容移至具有足够访问权的位置。

结果

现在，您已安装 MQ Service Provider 的所有必备软件。

下一步做什么

现在必须启用 MQ Service Provider 和 z/OS Connect。

相关任务

[第 382 页的『IBM z/OS Connect EE - MQ Service Provider 设置过程』](#)

要在 IBM z/OS Connect EE 上设置 MQ Service Provider，需要安装的组件的概述。

[第 383 页的『安装 IBM MQ 资源适配器』](#)

IBM z/OS Connect EE 中的 IBM MQ 消息传递提供程序功能部件使用称为 IBM MQ 资源适配器的 IBM MQ 组件。资源适配器作为 IBM MQ for z/OS Unix 系统服务组件功能部件的一部分随 IBM MQ 提供。

[第 385 页的『启用 z/OS Connect 和 MQ Service Provider』](#)

要启用 z/OS Connect 和 MQ Service Provider 必须执行的操作。

启用 z/OS Connect 和 MQ Service Provider

要启用 z/OS Connect 和 MQ Service Provider 必须执行的操作。

开始之前

确保您已完成以下过程：

- [第 382 页的『安装 IBM z/OS Connect EE』](#)
- [第 383 页的『安装 IBM MQ 资源适配器』](#)
- [第 384 页的『将 MQ Service Provider 安装到 IBM z/OS Connect EE 中』](#)

关于此任务

此过程将同时启用 z/OS Connect 和 MQ Service Provider。

过程

1. 编辑已创建的 z/OS Connect EE 服务器的 `server.xml`，并将整个 **featureManager** 元素替换为以下行：

```
<featureManager>
  <feature>zosconnect:zosconnect-2.0</feature>
  <feature>appSecurity-2.0</feature>
  <feature>jms-2.0</feature>
  <feature>mqzosconnect:zosConnectMQ-2.0</feature>
  <feature>wmqJmsClient-2.0</feature>
  <feature>zosTransaction-1.0</feature>
</featureManager>
```



注意：仅当这些条目尚不存在时，才修改这些条目。

2. 配置 z/OS Connect 的安全性。
请参阅 [配置 z/OS Connect EE 的安全性](#)，以获取有关如何执行此过程的详细信息。
3. 启动服务器。

下一步做什么

检查是否正确设置了 z/OS Connect。

相关任务

[第 382 页的『IBM z/OS Connect EE - MQ Service Provider 设置过程』](#)

要在 IBM z/OS Connect EE 上设置 MQ Service Provider，需要安装的组件的概述。

[第 386 页的『正在检查是否正确设置了 z/OS Connect』](#)

如何正确设置 z/OS Connect。

如何正确设置 z/OS Connect。

开始之前

确保已执行 [第 385 页的『启用 z/OS Connect 和 MQ Service Provider』](#) 中详细描述的过程

关于此任务

z/OS Connect 提供了 RESTful API，可用于查询已安装的服务并执行管理操作，例如停止和启动这些服务。

过程

1. 针对 z/OS Connect 发出 HTTP GET 以查询当前安装的服务的列表。
通过输入表单的 URL 来使用 Web 浏览器执行此操作

```
https://HOST_NAME:HTTPS_PORT/zosConnect/services
```

其中 *HOST_NAME* 和 *HTTPS_PORT* 是您在 [第 382 页的『安装 IBM z/OS Connect EE』](#) 的步骤 [第 382 页的『1』](#) 中输入的值。

例如，

```
https://yourdomainname:12342/zosConnect/services
```

2. 在浏览器提示时输入用户标识和密码。
这些是您在 [第 385 页的『启用 z/OS Connect 和 MQ Service Provider』](#) 的步骤 [第 385 页的『2』](#) 中的用户元素中输入的值。

结果

这将生成以下 JSON 响应，指示 z/OS Connect 正在运行，但尚未安装任何服务。请注意，如果使用了安装了服务的现有 z/OS Connect 服务器，那么将显示这些服务。

```
-----  
{  
  "zosConnectServices":[  ]  
}
```

下一步做什么

在 z/OS Connect EE 上设置简单的单向 IBM MQ 服务。

相关任务

[第 382 页的『IBM z/OS Connect EE - MQ Service Provider 设置过程』](#)

要在 IBM z/OS Connect EE 上设置 MQ Service Provider，[需要安装的组件的概述](#)。

按顺序使用以下步骤来设置简单的单向 MQ Service Provider 服务。

开始之前

确保已正确设置所有组件，如 [启用 z/OS Connect](#) 以及 [MQ Service Provider](#) 和 [检查 z/OS Connect 是否正确设置](#) 中所述

过程

1. 使用 MQSC 或 IBM MQ Explorer 在目标 z/OS 队列管理器上创建名为 ONE_WAY_QUEUE 的队列。
2. 定义 IBM MQ 消息传递提供程序连接工厂和队列。
要执行此操作，请将以下内容添加到已创建但在服务器元素内的 z/OS Connect EE 服务器的 `server.xml` 底部。

```
-----  
<jmsConnectionFactory id="cf1" jndiName="jms/cf1" connectionManagerRef="ConMgr1">  
  <properties.wmqJms  
    transportType="BINDINGS"  
    queueManager="MQ21"/>  
</jmsConnectionFactory>  
  
<connectionManager id="ConMgr1" maxPoolSize="5"/>  
  
<jmsQueue id="q1" jndiName="jms/d1">  
  <properties.wmqJms  
    baseQueueName="ONE_WAY_QUEUE"/>  
</jmsQueue>  
-----
```

注意:

- a. 将 **queueManager** 属性的值更改为正确的目标队列管理器名称。
 - b. 使用绑定的 **transportType**。表示使用跨内存连接与队列管理器进行通信。这是使用 MQ Service Provider 时唯一受支持的 **transportType**。
3. 通过将以下内容添加到已创建但在服务器元素内的 z/OS Connect EE 服务器的 `server.xml`，定义简单的单向 IBM MQ 服务。

```
-----  
<zosconnect_zosConnectService id="zosconnMQ1"  
  invokeURI="/oneWay"  
  serviceName="oneWay"  
  serviceRef="oneWay " />  
  
<mqzosconnect_mqzOSConnectService id="oneWay "  
  connectionFactory="jms/cf1"  
  destination="jms/d1"/>  
-----
```

`zosConnectService` 元素定义新服务以 z/OS 使用 `oneWay` 的 **serviceName** 进行连接。:

- **invokeURI** 属性使调用服务更容易。
- **serviceRef** 属性必须与 z/OS Connect 服务提供者的标识属性匹配，在本例中，此属性由 `mqzOSConnectService` 元素提供。

`mqzOSConnectService` 元素定义由 MQ Service Provider 提供的单个服务实例。

connectionFactory 和 **destination** 属性分别告知实例如何找到 IBM MQ 消息传递提供程序连接工厂和队列。

请参阅 [mqzOSConnectService 元素](#) 以获取此结构中的属性的详细信息。

结果

您已设置简单的单向服务。

下一步做什么

您需要测试服务。

相关任务

第 388 页的『在 z/OS Connect EE 上测试单向服务』
确保单向服务正常工作的一系列步骤。

开始之前

确保已成功完成 第 386 页的『在 z/OS Connect EE 上设置简单的单向 MQ Service Provider 服务』。

过程

1. 检查 z/OS Connect 是否识别新服务。

要执行此操作，请重新发出 [检查是否正确设置了 z/OS Connect](#) 中详细描述的步骤。

由于现在定义了服务，因此您应该会看到类似于以下输出的内容：

```
-----
{
  "zosConnectServices": [
    {
      "ServiceName": "oneWay",
      "ServiceDescription": "DATA_UNAVAILABLE",
      "ServiceProvider": "IBM MQ for z/OS service provider for IBM z/OS Connect EE V2.0",
      "ServiceURL": "https://yourdomainname:12342/zosConnect/services/oneWay"
    }
  ]
}
-----
```

注: ServiceDescription 作为 DATA_UNAVAILABLE 返回，因为未在 zosConnectService 元素上指定 **serviceDescription** 属性。

2. 向服务发出 HTTP POST。这将导致向 ONE_WAY_QUEUE 发送新消息，并在 POST 上指定任何有效内容。

执行此类测试的最佳方法是使用类似于 Google Chrome 的 [Advanced REST client \(ARC\)](#) 插件。

- a) 使用 ARC 针对以下 URL 发出 HTTP POST:

```
https://HOST_NAME:HTTPS_PORT/oneWay
```

- b) 为主体指定一些有效的 JSON，例如:

```
{"name1" : "value1", "name2" : "value2"}
```

您需要指定两个头:

- i) 授权 = Basic ENCODED_USERID_PASSWORD

其中 ENCODED_USERID_PASSWORD 是基本 64 位编码的用户标识和密码，如 [基本认证方案](#) 中所述。

- ii) 内容类型 = application/json

第一个头用于登录到 z/OS Connect，并且是 [启用 z/OS Connect 和 MQ Service Provider](#) 中使用的用户标识和密码的编码格式。

第二个头通知服务器有效内容是 JSON，这是向 MQ Service Provider 发出 HTTP POST 调用时唯一受支持的值。

如果未指定此头，那么将发生 HTTP 415 错误，指示不支持的介质类型结果。

向单向服务发出 HTTP POST 会导致返回空的 HTTP 响应主体和 HTTP 响应代码 204 (无内容)。

ARC 输出一条消息，指示响应不包含任何数据。

要验证是否已放入消息，请使用 IBM MQ Explorer 之类的工具来浏览 ONE_WAY_QUEUE 的内容。队列上应该有一条格式为 MQSTR 的 IBM MQ 消息，其中包含已发送的 JSON 有效内容。

3. 使用 ARC 对以下 URL 发出 HTTP DELETE:

执行此操作将以破坏性方式获取刚刚发送到单向服务的消息。

这应提供包含最初发送的 JSON 有效内容的响应主体。您可以使用 IBM MQ Explorer 来确认已破坏性地获取消息。

相关任务

第 386 页的『在 z/OS Connect EE 上设置简单的单向 MQ Service Provider 服务』

按顺序使用以下步骤来设置简单的单向 MQ Service Provider 服务。

V 9.0.1

z/OS

在 z/OS Connect EE 上设置简单的双向 IBM MQ 服务

如何在 z/OS Connect EE 中针对名为 MQ_REQUEST_Q 和 MQ_REPLY_Q 的现有 IBM MQ 队列对设置双向 IBM MQ 服务。

开始之前

确保已正确设置所有组件，如第 395 页的『启用 z/OS Connect 和 MQ Service Provider』和 第 395 页的『正在检查是否正确设置了 z/OS Connect』中所述。

关于此任务

这比设置单向服务略多，因为您需要后端应用程序来使用使用 z/OS Connect 发送的请求消息，并生成应答消息。

此任务假定 MQ_REQUEST_Q 和 MQ_REPLY_Q 队列由现有后端应用程序 (例如， CICS 或 IMS 事务) 使用，该应用程序从 MQ_REQUEST_Q 获取请求消息并生成放置在 MQ_REPLY_Q 上的应答消息。

过程

1. 定义 IBM MQ 消息传递提供程序连接工厂，以及到 z/OS Connect EE 服务器的两个 IBM MQ 消息传递提供程序队列。

IBM MQ 消息传递提供程序队列映射到现有 MQ_REQUEST_Q 和 MQ_REPLY_Q 队列。要执行此操作，请将以下内容添加到已创建但在服务器元素内的 z/OS Connect EE 服务器的 `server.xml` 底部。

```
<jmsConnectionFactory id="cf2" jndiName="jms/cf2"
    connectionManagerRef="ConMgr2">
  <properties.wmqJms
    transportType="BINDINGS"
    queueManager="MQ21"/>
</jmsConnectionFactory>

<connectionManager id="ConMgr2" maxPoolSize="5"/>

<jmsQueue id="requestQueue" jndiName="jms/requestQueue">
  <properties.wmqJms baseQueueName="MQ_REQUEST_Q"/>
</jmsQueue>

<jmsQueue id="replyQueue" jndiName="jms/replyQueue">
  <properties.wmqJms baseQueueName="MQ_REPLY_Q"/>
</jmsQueue>
```

注意:

- a. 将 **queueManager** 属性的值更改为正确的目标队列管理器名称。
 - b. 根据后端应用程序的性质，您可能需要对这些元素执行进一步配置。请参阅第 400 页的『配置 MQ Service Provider』以获取更多信息。
 - c. 可能需要根据并发请求数来调整 **maxPoolSize** 属性。有关此属性的更多信息，请参阅 [为 JMS 连接配置连接池](#)。
2. 通过将以下内容添加到您已创建但在服务器元素内的 z/OS Connect EE 服务器的 `server.xml`，定义双向 IBM MQ 服务。

```
<zosconnect_zosConnectService id="zosconnMQ2"
    invokeURI="/twoWay"
    serviceName="twoWay"
    serviceRef="twoWay" />

<mqzosconnect_mqzOSConnectService id="twoWay"
    connectionFactory="jms/cf2"
    destination="jms/requestQueue"
    replyDestination="jms/replyQueue"
    waitInterval="10000" />
```

注意:

- 根据后端应用程序的性质，您可能需要对这些元素执行进一步配置。例如，您可能需要数据转换。请参阅第 400 页的『配置 MQ Service Provider』以获取更多信息。
- 您可能需要调整 **waitInterval** 的值，请参阅 [waitInterval](#) 以获取更多信息。
- 在此示例中，假定后端应用程序放置在应答目标上的消息具有与请求目标上的消息标识相同的相关标识。

如果情况并非如此，那么需要将 **replySelection** 属性添加到 `mqzosconnect_mqzOSConnectService` 元素，并相应地设置值。

请参阅 [replySelection](#) 以获取更多信息。

结果

您已在 z/OS Connect EE 中设置双向服务，该服务可与使用一对 IBM MQ 队列的现有后端应用程序配合使用。

下一步做什么

您可以调用双向服务，向 URL `https://host_name:port_no/twoWay` 发出 HTTP POST 命令。

执行此操作的步骤与第 388 页的『在 z/OS Connect EE 上测试单向服务』中描述的步骤类似，只是仅支持 HTTP POST 动词，您应该发送适用于现有后端应用程序的 JSON 有效内容数据。

相关任务

第 386 页的『在 z/OS Connect EE 上设置简单的单向 MQ Service Provider 服务』
按顺序使用以下步骤来设置简单的单向 MQ Service Provider 服务。

V 9.0.1 z/OS 生成服务归档 (SAR) 文件

在使用 z/OS Connect EE 创建 API 之前，需要生成 SAR 文件。

关于此任务

您可以使用 IBM MQ 通过使用 IBM Documentation 中的 *z/OS Connect EE* 信息中描述的 CICS-WOLA 进程来实现此目的。请参阅 [根据 CICS COBOL 副本生成服务归档](#) 以获取更多信息。

要点: 以下文本中描述的过程是可选的。如果您不想使用 API 编辑器提供的工具，那么不需要 SAR 文件。但是，如果要使用 z/OS Connect EE 提供的数据变换支持，那么您可能希望使用上述链接中描述的 JSON 模式和绑定文件。

上述链接中描述的过程将使用 z/OS Connect EE 随附的 **BAQLS2JS** 实用程序。

此实用程序:

- 生成可用于 z/OS Connect EE 中提供的数据变换功能的 JSON 模式和绑定文件。

如果要在 z/OS Connect EE 中配置这些文件，请通过 `zosconnect_zosConnectDataXform` 元素在配置中设置这些文件。

请参阅第 403 页的『将数据转换与 MQ Service Provider 配合使用』以获取更多信息。

2. 生成一个 SAR 文件，其中包含先前步骤中的 JSON 模式以及对特定 z/OS Connect EE 服务实例的 **serviceName** 属性的引用。

将使用 SERVICE-NAME 参数提供对 **BAQLS2JS** 的引用。

例如，如果将 SERVICE-NAME=MQ1WayBackend 设置为 **BAQLS2JS** 的属性，那么 **BAQLS2JS** 将生成可用于配置如下的 IBM MQ 服务实例的 SAR:

```
<zosconnect_zosConnectService id="MQ1WayService"
  serviceName="MQ1WayBackend"
  serviceRef="mq1way"
  invokeURI="/mq1way"
  dataXformRef="xformJSON2Byte"/>

<mqzosconnect_mqzOSConnectService id="mq1way"
  connectionFactory="jms/cf1"
  destination="jms/oneWayRequestQ"/>
```

结果

生成后，可以将 SAR 文件导入到 API 编辑器中以生成 API。有关更多信息，请参阅 [使用 z/OS Connect EE API 编辑器创建 API](#)。

在 API 编辑器中构造 API 后，将生成 API 归档 (AAR)，然后可以将其部署到 z/OS Connect EE，在此将针对 **BAQLS2JS** 中指定的 SERVICE-NAME 所引用的服务运行该 API 归档。有关更多信息，请参阅 [将 API 部署到 z/OS Connect EE 服务器](#)。

相关任务

第 386 页的『[在 z/OS Connect EE 上设置简单的单向 MQ Service Provider 服务](#)』

按顺序使用以下步骤来设置简单的单向 MQ Service Provider 服务。

第 389 页的『[在 z/OS Connect EE 上设置简单的双向 IBM MQ 服务](#)』

如何在 z/OS Connect EE 中针对名为 MQ_REQUEST_Q 和 MQ_REPLY_Q 的现有 IBM MQ 队列对设置双向 IBM MQ 服务。

V 9.0.1 z/OS IBM z/OS Connect V1 - MQ Service Provider 设置过程

要使 MQ Service Provider 能够在 IBM z/OS Connect V1 上设置，需要安装的组件的概述。

关于此任务

按照显示的顺序执行以下任务，以安装 MQ Service Provider 及其所有必备软件。

V 9.0.1 z/OS 安装 WebSphere Application Server Liberty

安装 WebSphere Application Server Liberty (WLP) 的要求和过程。

开始之前

确保您有权访问包含 z/OS Connect 的 z/OS 的 WLP 版本，例如 16.0.0.2。

关于此任务

此过程将安装 WLP 并设置服务器。

过程

1. 通过遵循 [使用 Installation Manager](#) 中详细描述的过程来安装 WLP。
请注意，对于这些指示信息的其余部分，此目录结构称为 **WLP_ROOT**。
2. 遵循 [手动创建 Liberty 服务器](#) 中详细描述的过程来创建新服务器
请参阅 [设置 z/OS Connect](#)，其中描述了如何在创建的服务器上启用 z/OS Connect。



注意: 您不需要执行可选的 *WebSphere* 优化本地适配器 (WOLA) 步骤。

3. 通过遵循在 [Liberty for z/OS 上启用 z/OS 授权服务](#) 中详细描述的过程来启用 TXRRS 授权服务
4. 通过以下方法验证 TXRRS 授权服务是否已正确设置:
 - a) 启动服务器。
遵循 [从命令行启动和停止服务器](#) 中详细描述的过程
 - b) 查看位于以下位置的服务器日志:

```
WLP_ROOT/usr/servers/server_name/logs/message.log
```

这些是 ASCII 文件，应该包含类似于以下内容的输出:

```
A CWWKE0001I: The server server1 has been launched.
I CWWKB0103I: Authorized service group LOCALCOM is available.
I CWWKB0103I: Authorized service group PRODMGR is available.
I CWWKB0103I: Authorized service group SAFCRE is available.
I CWWKB0103I: Authorized service group TXRRS is available.
I CWWKB0103I: Authorized service group WOLA is available.
I CWWKB0103I: Authorized service group ZOSDUMP is available.
I CWWKB0103I: Authorized service group ZOSWLM is available.
I CWWKB0103I: Authorized service group CLIENT.WOLA is available.
I CWWKB0108I: IBM CORP product WAS FOR Z/OS version 16.0 successfully registered with z/OS.
```

检查输出以查看 TXRRS 授权服务组是否可用。在前面的示例中，粗体文本中的行显示服务组可用。

MQ Service Provider 需要此服务组可用。

- c) 正在停止服务器。
遵循 [从命令行启动和停止服务器](#) 中详细描述的过程

结果

您已成功安装 WLP。

下一步做什么

安装 IBM MQ 资源适配器。

相关任务

第 391 页的『[IBM z/OS Connect V1 - MQ Service Provider 设置过程](#)』

要使 MQ Service Provider 能够在 IBM z/OS Connect V1 上设置，需要安装的组件的概述。

第 392 页的『[安装 IBM MQ 资源适配器](#)』

WLP 中的 IBM MQ 消息传递提供程序功能部件使用称为 IBM MQ 资源适配器的 IBM MQ 组件。出于兼容性原因，z/OS Connect V1 的用户必须使用 IBM WebSphere MQ 7.5 资源适配器，而不考虑它所连接到的队列管理器的版本。

V 9.0.1

z/OS

安装 IBM MQ 资源适配器

WLP 中的 IBM MQ 消息传递提供程序功能部件使用称为 IBM MQ 资源适配器的 IBM MQ 组件。出于兼容性原因，z/OS Connect V1 的用户必须使用 IBM WebSphere MQ 7.5 资源适配器，而不考虑它所连接到的队列管理器的版本。

开始之前

转至 [获取 IBM MQ Resource Adapter for WLP](#)(描述如何链接到 Fix Central)，然后下载最新的资源适配器。

资源适配器将采用以下格式: `7.5.0.Fix_level-WS-MQ-Java-InstallRA.jar`。请注意，该过程引用的是此 jar 文件。

关于此任务

此过程将在 WLP 中安装 IBM MQ 资源适配器。

过程

1. 解压缩 jar 文件的内容，转至 wmq 目录，然后以二进制方式将 wmq.jmsra.rar 文件通过 FTP 传输到目标文件系统上的相应目录中。
对于这些指示信息的其余部分，此目录称为 MQRA_ROOT。
2. 找到目标队列管理器的 Unix 系统服务组件目录。
例如，此目录可能称为 /mqm/V9R0M1/。该目录应包含 java/lib 子目录，其中包含大量本机库 (.so 文件)。
3. 编辑服务器的 server.xml 文件。
添加下列行：

```
<variable name="wmqJmsClient.rar.location"
  value="MQRA_ROOT/wmq.jmsra.rar"/>
<wmqJmsClient nativeLibraryPath="MQJAVA_LIB_DIR" />
```

其中 MQJAVA_LIB_DIR 是位于步骤 [第 393 页的『2』](#) 中的目录，例如 /usr/lpp/mqm/V9R0M1/java/lib。

- b. 保存更改。

第一行告诉 WLP 在何处找到 IBM MQ 资源适配器。

第二行告诉 IBM MQ 资源适配器在何处查找用于与 IBM MQ 的绑定连接的本机库。

有关更多信息，请参阅 [wmqJmsClient](#) 和 [将 JMS 应用程序部署到 Liberty 以使用 IBM MQ 消息传递提供程序](#)。

4. 设置 STEPLIB。
必须执行此操作，以便位于步骤 [第 393 页的『2』](#) 中的本机库可以连接到队列管理器。
在用于启动 WLP 服务器的 shell 中运行以下命令：

```
export STEPLIB=HLQ.SCSQAUTH:HLQ.SCSQANLE
```

其中 HLQ 是包含 IBM MQ 安装的数据集的高级限定符。

结果

您已部分安装 IBM MQ 资源适配器。请注意，在启用其父功能部件 (wmqJmsClient-1.1) 之前，将不会完全安装 IBM MQ 资源适配器。

下一步做什么

将 MQ Service Provider 安装到 WLP 中。

相关任务

[第 391 页的『IBM z/OS Connect V1 - MQ Service Provider 设置过程』](#)

要使 MQ Service Provider 能够在 IBM z/OS Connect V1 上设置，需要安装的组件的概述。

[第 391 页的『安装 WebSphere Application Server Liberty』](#)

安装 WebSphere Application Server Liberty (WLP) 的要求和过程。

[第 393 页的『将 MQ Service Provider 安装到 WLP 中』](#)

在使用 MQ Service Provider 之前，必须将其安装到 WLP 中。

V 9.0.1 **z/OS** **将 MQ Service Provider 安装到 WLP 中**

在使用 MQ Service Provider 之前，必须将其安装到 WLP 中。

开始之前

使用 [第 381 页的『获取 MQ Service Provider』](#) 中包含的信息获取并安装 MQ Service Provider。

关于此任务

MQ Service Provider 是一个标准 WLP 功能部件，您可以使用 [打包和安装 Liberty 功能部件](#) 中描述的其中一种方法来安装该功能部件。

请参阅 [目录位置和属性](#) 以说明 `${wlp.install.dir}` 和 `${wlp.user.dir}` 的含义。

以下文本提供了方法的概述。

过程

1. 将该功能部件安装到 WLP 内核中。



注意: 此方法需要对 WLP 安装目录的写访问权。如果您对 WLP 安装目录没有写访问权，那么选项 [第 394 页的『2』](#) 可能更合适。

为此，请复制：

- a) `MQSP_ROOT/v1.0/lib/com.ibm.mq.zosconnect_1.0.0.jar` 到 `${wlp.install.dir}/lib`。
 - b) `MQSP_ROOT/v1.0/lib/features/zosConnectMQ-1.0.mf` 到 `${wlp.install.dir}/lib/features`。
2. 将该功能部件安装到用户配置中。
要执行此复制：
 - a) `MQSP_ROOT/v1.0/lib/com.ibm.mq.zosconnect_1.0.0.jar` 到 `{wlp.user.dir}/extension/lib`。
 - b) `MQSP_ROOT/v1.0/lib/features/zosConnectMQ-1.0.mf` 到 `{wlp.user.dir}/extension/lib/features`。
 3. 将该功能部件作为产品扩展进行安装。
要执行此操作：
 - a) 将 `MQSP_ROOT/mqzosconnect.properties` 复制到 `${wlp.install.dir}/etc/extensions`。
 - b) 编辑复制的文件。
请注意，此文件是 ASCII 码文件。将 `PATH_TO_INSTALL` 更改为 `MQSP_ROOT/v1.0`，并保存更改。

结果

现在，您已安装 MQ Service Provider 的所有必备软件。

注：用于安装 MQ Service Provider 的方法会影响在 `server.xml` 中的引用方式。示例如 [第 395 页的『启用 z/OS Connect 和 MQ Service Provider』](#) 中所示。

下一步做什么

现在必须启用 MQ Service Provider 和 z/OS Connect。

相关任务

[第 391 页的『IBM z/OS Connect V1 - MQ Service Provider 设置过程』](#)

要使 MQ Service Provider 能够在 IBM z/OS Connect V1 上设置，需要安装的组件的概述。

[第 392 页的『安装 IBM MQ 资源适配器』](#)

WLP 中的 IBM MQ 消息传递提供程序功能部件使用称为 IBM MQ 资源适配器的 IBM MQ 组件。出于兼容性原因，z/OS Connect V1 的用户必须使用 IBM WebSphere MQ 7.5 资源适配器，而不考虑它所连接到的队列管理器的版本。

第 395 页的『[启用 z/OS Connect 和 MQ Service Provider](#)』
要启用 z/OS Connect 和 MQ Service Provider 必须执行的操作。

V 9.0.1 **z/OS** **启用 z/OS Connect 和 MQ Service Provider**

要启用 z/OS Connect 和 MQ Service Provider 必须执行的操作。

开始之前

确保您已完成以下过程:

- [第 391 页的『安装 WebSphere Application Server Liberty』](#)
- [第 392 页的『安装 IBM MQ 资源适配器』](#)
- [第 393 页的『将 MQ Service Provider 安装到 WLP 中』](#)

关于此任务

此过程将同时启用 z/OS Connect 和 MQ Service Provider。

过程

1. 编辑已创建的 `server.xml`，并将整个 **featureManager** 元素替换为这些行。



注意: `<feature>zosConnectMQ-1.0</feature>` 行的准确格式取决于您用于安装 MQ Service Provider 的方法，如 [第 393 页的『将 MQ Service Provider 安装到 WLP 中』](#) 中所述。

如果使用了:

- 方法 [第 394 页的『1』](#)，行显示为 `<feature>zosConnectMQ-1.0</feature>`。
- 方法 [第 394 页的『2』](#)，行显示为 `<feature>usr:zosConnectMQ-1.0</feature>`。
- 方法 [第 394 页的『3』](#)，行显示为 `<feature>mqzosconnect:zosConnectMQ-1.0</feature>`。

此外，仅当以下条目尚不存在时，才修改这些条目。

```
<featureManager>
<feature>zosConnect-1.0</feature>
<feature>appSecurity-2.0</feature>
<feature>zosConnectMQ-1.0</feature>
<feature>wmqJmsClient-1.1</feature>
<feature>zosTransaction-1.0</feature>
</featureManager>
```

2. 配置 z/OS Connect 的安全性。
请参阅 [为 z/OS Connect 配置安全性](#)，以获取有关如何执行此过程的详细信息。
3. 启动服务器。
遵循 [从命令行启动和停止服务器](#)中详细描述的过程。

下一步做什么

检查是否正确设置了 z/OS Connect。

相关任务

[第 391 页的『IBM z/OS Connect V1 - MQ Service Provider 设置过程』](#)
要使 MQ Service Provider 能够在 IBM z/OS Connect V1 上设置，需要安装的组件的概述。

[第 395 页的『正在检查是否正确设置了 z/OS Connect』](#)
如何正确设置 z/OS Connect。

V 9.0.1 **z/OS** **正在检查是否正确设置了 z/OS Connect**

如何正确设置 z/OS Connect。

开始之前

确保已执行 [第 395 页的『启用 z/OS Connect 和 MQ Service Provider』](#) 中详细描述的过程

关于此任务

z/OS Connect 提供了 RESTful API，可用于查询已安装的服务并执行管理操作，例如停止和启动这些服务。

过程

1. 针对 z/OS Connect 发出 HTTP GET 以查询当前安装的服务的列表。
通过输入表单的 URL 来使用 Web 浏览器执行此操作

```
https://HOST_NAME:HTTPS_PORT/zosConnect/services
```

其中 *HOST_NAME* 和 *HTTPS_PORT* 是您在 [第 391 页的『安装 WebSphere Application Server Liberty』](#) 的步骤 [第 391 页的『2』](#) 中输入的值。

例如，

```
https://yourdomainname:12342/zosConnect/services
```

2. 在浏览器提示时输入用户标识和密码。
这些是您在 [第 395 页的『启用 z/OS Connect 和 MQ Service Provider』](#) 的步骤 [第 395 页的『2』](#) 中的用户元素中输入的值。

结果

这将生成以下 JSON 响应，指示 z/OS Connect 正在运行，但尚未安装任何服务：

```
-----  
{  
  "zosConnectServices":[  ]  
}
```



下一步做什么

[第 396 页的『在 z/OS Connect V1 上设置简单的单向 MQ Service Provider 服务』](#)

相关任务

[第 391 页的『IBM z/OS Connect V1 - MQ Service Provider 设置过程』](#)

要使 MQ Service Provider 能够在 IBM z/OS Connect V1 上设置，需要安装的组件的概述。

  **在 z/OS Connect V1 上设置简单的单向 MQ Service Provider 服务**

按顺序使用以下步骤来设置简单的单向 MQ Service Provider 服务。

开始之前

确保已正确设置所有组件，如 [启用 z/OS Connect 以及 MQ Service Provider](#) 和 [检查 z/OS Connect 是否正确设置](#) 中所述

过程

1. 使用 MQSC 或 IBM MQ Explorer 在目标 z/OS 队列管理器上创建名为 ONE_WAY_QUEUE 的队列。
2. 定义 IBM MQ 消息传递提供程序连接工厂和队列。

要执行此操作，请将以下内容添加到 `server.xml` 的底部，但在服务器元素中。

```
<jmsConnectionFactory id="cf1" jndiName="jms/cf1" connectionManagerRef="ConMgr1">
  <properties.wmqJms
    transportType="BINDINGS"
    queueManager="MQ21" />
</jmsConnectionFactory>

<connectionManager id="ConMgr1" maxPoolSize="5"/>

<jmsQueue id="q1" jndiName="jms/d1">
  <properties.wmqJms
    baseQueueName="ONE_WAY_QUEUE" />
</jmsQueue>
```

注意:

- a. 将 **queueManager** 属性的值更改为正确的目标队列管理器名称。
 - b. 使用绑定的 **transportType**。表示使用跨内存连接与队列管理器进行通信。这是使用 MQ Service Provider 时唯一受支持的 **transportType**。
3. 通过将以下内容添加到 `server.xml`，但在 `server` 元素中定义简单的单向 MQ Service Provider 服务。

```
<zosConnectService id="zosconnMQ1"
  invokeURI="/oneWay"
  serviceName="oneWay"
  serviceRef="oneWay" />

<mqzOSConnectService id="oneWay"
  connectionFactory="jms/cf1"
  destination="jms/d1" />
```

`zosConnectService` 元素定义新服务以 z/OS 使用 `oneWay` 的 **serviceName** 进行连接。:

- **invokeURI** 属性使调用服务更容易。
- **serviceRef** 属性必须与 z/OS Connect 服务提供者的标识属性匹配，在本例中，此属性由 `mqzOSConnectService` 元素提供。

`mqzOSConnectService` 元素定义由 MQ Service Provider 提供的单个服务实例。

connectionFactory 和 **destination** 属性分别告知实例如何找到 IBM MQ 消息传递提供程序连接工厂和队列。

请参阅 `mqzOSConnectService` 元素 以获取此结构中的属性的详细信息。

结果



您已设置简单的单向服务。

下一步做什么

您需要测试服务。

相关任务

第 397 页的『在 z/OS Connect V1 上测试单向服务』
确保单向服务正常工作的一系列步骤。

  在 z/OS Connect V1 上测试单向服务
确保单向服务正常工作的一系列步骤。

开始之前

确保已成功完成 [第 396 页](#) 的『在 z/OS Connect V1 上设置简单的单向 MQ Service Provider 服务』。

过程

1. 检查 z/OS Connect 是否识别新服务。

要执行此操作，请重新发出 [检查是否正确设置了 z/OS Connect](#) 中详细描述的步骤。

由于现在定义了服务，因此您应该会看到类似于以下输出的内容：

```
-----  
{  
  "zosConnectServices": [  
    {  
      "ServiceName": "oneWay",  
      "ServiceDescription": "DATA_UNAVAILABLE",  
      "ServiceProvider": "IBM MQ for z/OS service provider for IBM z/OS Connect" V1.0,  
      "ServiceURL": "https://yourdomainname:12342/zosConnect/services/oneWay"  
    }  
  ]  
}
```

注: ServiceDescription 作为 DATA_UNAVAILABLE 返回，因为未在 zosConnectService 元素上指定 **serviceDescription** 属性。

2. 向服务发出 HTTP POST。这将导致向 ONE_WAY_QUEUE 发送新消息，并在 POST 上指定任何有效内容。

执行此类测试的最佳方法是使用类似于 Google Chrome 的 [Advanced REST client \(ARC\)](#) 插件。

- a) 使用 ARC 针对以下 URL 发出 HTTP POST:

```
https://HOST_NAME:HTTPS_PORT/oneWay
```

- b) 为主体指定一些有效的 JSON，例如:

```
{"name1" : "value1", "name2" : "value2"}
```

您需要指定两个头:

- i) 授权 = Basic ENCODED_USERID_PASSWORD

其中 ENCODED_USERID_PASSWORD 是基本 64 位编码的用户标识和密码，如 [基本认证方案](#) 中所述。

- ii) 内容类型 = application/json

第一个头用于登录到 z/OS Connect，并且是 [第 395 页](#) 的『启用 z/OS Connect 和 MQ Service Provider』中使用的用户标识和密码的编码格式。

第二个头通知服务器有效内容是 JSON，这是向 MQ Service Provider 发出 HTTP POST 调用时唯一受支持的值。

如果未指定此头，那么将发生 HTTP 415 错误，指示不支持的介质类型结果。

向单向服务发出 HTTP POST 会导致返回空的 HTTP 响应主体和 HTTP 响应代码 204 (无内容)。

ARC 输出一条消息，指示响应不包含任何数据。

要验证是否已放入消息，请使用 IBM MQ Explorer 之类的工具来浏览 ONE_WAY_QUEUE 的内容。队列上应该有一条格式为 MQSTR 的 IBM MQ 消息，其中包含已发送的 JSON 有效内容。

3. 使用 ARC 对以下 URL 发出 HTTP DELETE:

```
https://HOST_NAME:HTTPS_PORT/oneWay
```

执行此操作将以破坏性方式获取刚刚发送到单向服务的消息。

这应提供包含最初发送的 JSON 有效内容的响应主体。您可以使用 IBM MQ Explorer 来确认已破坏性地获取消息。

相关任务

第 396 页的『在 z/OS Connect V1 上设置简单的单向 MQ Service Provider 服务』
按顺序使用以下步骤来设置简单的单向 MQ Service Provider 服务。

V 9.0.1 z/OS 在 z/OS Connect Version 1 上设置简单的双向 IBM MQ 服务

如何在 z/OS Connect V1 中针对名为 MQ_REQUEST_Q 和 MQ_REPLY_Q 的现有 IBM MQ 队列对设置双向 IBM MQ 服务。

开始之前

确保已正确设置所有组件，如第 395 页的『启用 z/OS Connect 和 MQ Service Provider』和 第 395 页的『正在检查是否正确设置了 z/OS Connect』中所述。

关于此任务

这比设置单向服务稍有涉及，因为您需要后端应用程序使用使用 z/OS Connect 发送的请求消息并生成应答消息。

此任务假定 MQ_REQUEST_Q 和 MQ_REPLY_Q 队列由现有后端应用程序 (例如，CICS 或 IMS 事务) 使用，该应用程序从 MQ_REQUEST_Q 获取请求消息并生成放置在 MQ_REPLY_Q 上的应答消息。

过程

1. 定义 IBM MQ 消息传递提供程序连接工厂，以及到 z/OS Connect V1 服务器的两个 IBM MQ 消息传递提供程序队列。

IBM MQ 消息传递提供程序队列映射到现有 MQ_REQUEST_Q 和 MQ_REPLY_Q 队列。要执行此操作，请将以下内容添加到 `server.xml` 的底部，但在服务器元素中。

```
<jmsConnectionFactory id="cf2" jndiName="jms/cf2"
    connectionManagerRef="ConMgr2">
  <properties.wmqJms
    transportType="BINDINGS"
    queueManager="MQ21"/>
</jmsConnectionFactory>

<connectionManager id="ConMgr2" maxPoolSize="5"/>

<jmsQueue id="requestQueue" jndiName="jms/requestQueue">
  <properties.wmqJms baseQueueName="MQ_REQUEST_Q"/>
</jmsQueue>

<jmsQueue id="replyQueue" jndiName="jms/replyQueue">
  <properties.wmqJms baseQueueName="MQ_REPLY_Q"/>
</jmsQueue>
```

注意:

- a. 将 `queueManager` 属性的值更改为正确的目标队列管理器名称。
 - b. 根据后端应用程序的性质，您可能需要对这些元素执行进一步配置。请参阅第 400 页的『配置 MQ Service Provider』以获取更多信息。
 - c. 可能需要根据并发请求数来调整 `maxPoolSize` 属性。有关此属性的更多信息，请参阅 [为 JMS 连接配置连接池](#)。
2. 通过将以下内容添加到 `server.xml`，但在服务器元素中定义双向 IBM MQ 服务。

```
-----
<zosConnectService id="zosconnMQ2"
    invokeURI="/twoWay"
    serviceName="twoWay"
    serviceRef="twoWay"/>

<mqzOSConnectService id="twoWay"
```

```
connectionFactory="jms/cf2"
destination="jms/requestQueue"
replyDestination="jms/replyQueue"
waitInterval="10000"/>
```

注意:

- 根据后端应用程序的性质，您可能需要对这些元素执行进一步配置。例如，您可能需要数据转换。请参阅第 400 页的『[配置 MQ Service Provider](#)』以获取更多信息。
- 您可能需要调整 `waitInterval` 的值，请参阅 [waitInterval](#) 以获取更多信息。
- 在此示例中，假定后端应用程序放置在应答目标上的消息具有与请求目标上的消息标识相同的相关标识。

如果情况并非如此，那么需要将 `replySelection` 属性添加到 `mqzOSConnectService` 元素，并相应地设置值。

请参阅 [replySelection](#) 以获取更多信息。

结果

您已在 z/OS Connect V1 中设置双向服务，该服务可与使用一对 IBM MQ 队列的现有后端应用程序配合使用。

下一步做什么

您可以调用双向服务，向 URL `https://host_name:port_no/twoWay` 发出 HTTP POST 命令。

执行此操作的步骤与第 397 页的『[在 z/OS Connect V1 上测试单向服务](#)』中描述的步骤类似，只是仅支持 HTTP POST 动词，您应该发送适用于现有后端应用程序的 JSON 有效内容数据。

相关任务

第 396 页的『[在 z/OS Connect V1 上设置简单的单向 MQ Service Provider 服务](#)』按顺序使用以下步骤来设置简单的单向 MQ Service Provider 服务。

V 9.0.1 z/OS 配置 MQ Service Provider

在使用 MQ Service Provider 之前，需要考虑各种方面。

本节包括：

- [第 400 页的『使用 MQ Service Provider 时的安全注意事项』](#)
- [第 403 页的『将数据转换与 MQ Service Provider 配合使用』](#)
- [第 408 页的『使用 MQI 应用程序』](#)

V 9.0.1 z/OS 使用 MQ Service Provider 时的安全注意事项

MQ Service Provider 有两组安全注意事项。

- 给定用户是否可以向 z/OS Connect 公开的 MQ Service Provider 服务提交 RESTful 请求。
- 是否授权给定 MQ Service Provider 服务访问队列管理器及其资源。

给定用户是否可以向从 z/OS Connect 公开的 MQ Service Provider 服务提交 RESTful 请求

这由 z/OS Connect 的 `server.xml` 文件中的配置控制。

对于 z/OS Connect EE，请参阅 [配置 z/OS Connect EE 的安全性](#)。

对于 z/OS Connect V1，请参阅 [为 z/OS Connect 配置安全性](#)。

是否授权给定 MQ Service Provider 服务访问队列管理器及其资源

MQ Service Provider 基本上是一个 JMS 应用程序，基于 WLP 随附的 IBM MQ 消息传递提供程序，该应用程序使用绑定方式连接连接到一个或多个队列管理器。

因此，可以采用与共享这些特征的应用程序相同的方式来保护 MQ Service Provider。本主题描述了这些相似之处，并指出了不同之处。

由于 MQ Service Provider 是绑定应用程序，因此它将连接到队列管理器并提供用户标识和可选密码。可以选择使用 [连接认证](#) 进行验证。

如果有效，那么允许用户根据队列管理器的安全配置进行连接。有关更多信息，请参阅 [在 z/OS 上设置安全性](#)。

mqzOSConnect 服务元素和 server.xml 中 **jmsConnectionFactory** 元素的 `properties.wmqJMS` 子元素上的配置属性会影响向队列管理器提供的用户标识和可选密码。

下表详细介绍了各种可能性。

 **注意:** 安全性配置属性之间存在优先顺序。**mqzOSConnectService** 属性覆盖 **properties.wmqJms** 属性，而 **useCallerPrincipal** 属性覆盖所有其他属性。

这两个密码属性都可以采用纯文本或编码格式提供。您应该使用编码格式，因为有权访问 server.xml 的任何人都可以纯文本查看密码。

这两个版本的 z/OS Connect 随附了一个名为 **securityUtility** 的工具，可用于对密码进行编码。有关更多信息，请参阅 [Liberty: securityUtility 命令](#)。

| mqzOSConnect 服务元素 | | properties.wmqJms 元素 | | 结果 |
|-------------------|---------|----------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| useCaller 主体 | 用户名和密码 | userName | password | |
| 不是 set/false | 不是集/空白 | 不是集/空白 | 不是集/空白 | 将向队列管理器提供与 z/OS Connect 地址空间相关联的用户名以进行授权和认证。未显示密码。 |
| 不是 set/false | 不是集/空白 | set | 不是集/空白 | properties.wmqJms 元素上的用户名将提供给队列管理器以进行授权和认证。未显示密码。 |
| 不是 set/false | 不是集/空白 | set | set | properties.wmqJms 元素上的用户名和密码将提供给队列管理器以用于授权和认证目的。 |
| 不是 set/false | 两个值都集 | 如果设置则忽略 | 如果设置则忽略 | mqzOSConnectService 元素上的用户名和密码将提供给队列管理器以用于授权和认证目的。 |
| true | 如果设置则忽略 | 如果设置则忽略 | 如果设置则忽略 | 将采用已向 z/OS Connect 认证的用户主体，并将其用户名提供给队列管理器以进行授权和认证。未显示密码。 此处提供了有关如何为 z/OS Connect EE 配置安全性的信息 ¹ 。 z/OS Connect V 1 的信息位于此处 ² 。 |

注意:

1. [配置 z/OS Connect EE 的安全性](#)
2. [配置 z/OS Connect 的安全性](#)

示例

这将显示单向服务，其中连接工厂和 mqzOSConnectService 都指定 **userID** 和 **password**。在这两种情况下，都会对密码进行编码。将使用在 mqzOSConnectService 定义上定义的 **userID** 和 **password**。

z/OS 连接 V1

```
<zosConnectService
id="samplezOSConnectService1"
invokeURI="/samplezOSConnectService1"
serviceName="samplezOSConnectService1_name"
serviceRef="samplezOSConnectService1_MQ"/>

<mqzOSConnectService
id="samplezOSConnectService1_MQ"
connectionFactory="jms/sampleCF1"
userName="bill"
password="{aes}AJ+DdZ+1u0KEG5KIwUz4LvHBAQ8nTd3y8K8HAI+48Tt"
destination="jms/sampleQ1"/>

<jmsConnectionFactory
id="sampleCF1"
jndiName="jms/sampleCF1"
connectionManagerRef="sampleCF2ConnectionManager1">

<properties.wmqJms
transportType="BINDINGS"
queueManager="MQ21"
userName="matt"
password="{xor}GBMeEg9uERg="/>
</jmsConnectionFactory>

<jmsQueue
id="sampleQ1"
jndiName="jms/sampleQ1">

<properties.wmqJms
baseQueueName="SampleQ1"/>
</jmsQueue>
```

z/OS 连接 EE

```
<zosconnect_zosConnectService
id="samplezOSConnectService1"
invokeURI="/samplezOSConnectService1"
serviceName="samplezOSConnectService1_name"
serviceRef="samplezOSConnectService1_MQ"/>

<mqzosconnect_mqzOSConnectService
id="samplezOSConnectService1_MQ"
connectionFactory="jms/sampleCF1"
userName="bill"
password="{aes}AJ+DdZ+1u0KEG5KIwUz4LvHBAQ8nTd3y8K8HAI+48Tt"
destination="jms/sampleQ1"/>

<jmsConnectionFactory
id="sampleCF1"
jndiName="jms/sampleCF1"
connectionManagerRef="sampleCF2ConnectionManager1">

<properties.wmqJms
transportType="BINDINGS"
queueManager="MQ21"
userName="matt"
password="{xor}GBMeEg9uERg="/>
</jmsConnectionFactory>

<jmsQueue
id="sampleQ1"
jndiName="jms/sampleQ1">

<properties.wmqJms
baseQueueName="SampleQ1"/>
</jmsQueue>
```

此示例显示了使用与第一个示例相同的 JMS 对象 (队列 `sampleQ1` 和连接工厂 `sampleCF1`) 的单向服务定义。向 z/OS Connect 认证的主体将提供给队列管理器，因为指定了 `useCallerPrincipal="true"`。

z/OS 连接 V1

```
<zosConnectService
id="samplezOSConnectService2"
invokeURI="/samplezOSConnectService2"
serviceName="samplezOSConnectService2_name"
serviceRef="samplezOSConnectService2_MQ"/>

<mqzOSConnectService
id="samplezOSConnectService2_MQ"
connectionFactory="jms/sampleCF1"
destination="jms/sampleQ1"
useCallerPrincipal="true"/>
```

z/OS 连接 EE

```
<zosconnect_zosConnectService
id="samplezOSConnectService2"
invokeURI="/samplezOSConnectService2"
serviceName="samplezOSConnectService2_name"
serviceRef="samplezOSConnectService2_MQ"/>

<mqzosconnect_mqzOSConnectService
id="samplezOSConnectService2_MQ"
connectionFactory="jms/sampleCF1"
destination="jms/sampleQ1"
useCallerPrincipal="true"/>
```

V 9.0.1

z/OS

将数据转换与 MQ Service Provider 配合使用

z/OS Connect 能够在调用后端 z/OS 资产 (例如 CICS 事务) 之前将 JSON 数据变换为任意格式，并将响应从 z/OS 资产变换回 JSON。

此功能可由可插拔数据转换提供程序提供。内置提供程序允许将 JSON 变换为 COBOL，PLI 或 C 结构，或者从 COBOL，PLI 或 C 结构变换为 JSON。

MQ Service Provider 用于 z/OS Connect 数据变换，但您应该考虑一些注意事项。

有关 z/OS Connect 数据转换的更多信息

对于 z/OS Connect Enterprise Edition，请参阅 [定义数据变换器](#)。

对于 z/OS Connect 版本 1，请参阅 [定义 z/OS Connect 消息有效内容变换](#)。

将消息发送至 IBM MQ

如果 MQ Service Provider 服务配置了数据变换，并且要将消息发送到队列管理器 (例如，如果它是已接收到包含 JSON 的 HTTP POST 的单向或双向服务)，那么 MQ Service Provider 将执行以下步骤：

1. 从 HTTP 请求获取 JSON 有效内容。
2. 将有效内容传递到 z/OS Connect 以执行从 JSON 到字节数组的数据转换。
3. 采用数据转换的结果，并以 JMS BytesMessage 的形式将其发送到队列管理器。

缺省情况下，发送的消息具有空白 MQMD **Format** 字段。在许多情况下，这不合适，因此您可以将 `mqzOSConnect` 服务元素中的 `mqmdFormat` 属性设置为适当的值。

内置数据转换支持始终生成 CCSID 为 37 的输出。需要在 MQMD CodedCharSetId 字段中指定此信息，否则从队列获取的应用程序可能无法对消息进行解码。通过在 `mqzOSConnectService` 服务元素所引用的队列上设置 CCSID 属性来执行此操作。

以下示例配置说明了将用于将消息发送到名为 `SampleQ1` 的队列的单向服务的相应配置。

将使用 MQMD Format 字段 `AFORMAT` 和 CodedCharSetId 字段 37 发送消息。

zosConnectDataXform 元素告诉 z/OS Connect 在何处查找数据变换的配置，并使用 **zosConnectService** 元素的 **dataXformRef** 属性进行引用。

有关双向服务的配置，请参阅第 404 页的『从 IBM MQ 接收消息』。

```
<jmsConnectionFactory
  id="sampleCF1"
  jndiName="jms/sampleCF1"
  connectionManagerRef="sampleCF1ConnectionManager">

  <properties.wmqJms
    transportType="BINDINGS"
    queueManager="MQ21" />
</jmsConnectionFactory>

<connectionManager
  id="sampleCF1ConnectionManager"
  maxPoolSize="5" />

<jmsQueue id="sampleQ1"
  jndiName="jms/sampleQ1">

  <properties.wmqJms
    baseQueueName="SampleQ1"
    CCSID="37" />
</jmsQueue>

<zosConnectService
  id="samplezOSConnectService1"
  invokeURI="/samplezOSConnectService1"
  serviceName="samplezOSConnectService1_name"
  serviceRef="samplezOSConnectService1_MQ"
  dataXformRef="xformJSON2Byte" />

<mqzOSConnectService
  id="samplezOSConnectService1_MQ"
  connectionFactory="jms/sampleCF1"
  mqmdFormat="AFORMAT"
  destination="jms/sampleQ1" />

<zosConnectDataXform id="xformJSON2Byte"
  bindFileLoc="/XF0RM_R00T/bindfiles" bindFileSuffix=".bnd"
  requestSchemaLoc="/XF0RM_R00T/json" requestSchemaSuffix=".json"
  responseSchemaLoc="/XF0RM_R00T/json"
  responseSchemaSuffix=".json" />
```

从 IBM MQ 接收消息

如果 MQ Service Provider 实例配置了数据变换，并且将从队列管理器接收消息 (例如，如果它是对 HTTP GET 或 DELETE 进行操作的单向服务，或者是接收了 HTTP POST 的双向服务)，那么 MQ Service Provider 将执行以下步骤。



注意: 如果您正在使用双向服务，那么该服务将已执行第 403 页的『将消息发送至 IBM MQ』中描述的步骤。

1. 从队列获取消息。
2. 检查消息是否为 JMS **BytesMessage** 或 JMS **TextMessage**。如果消息都不存在，那么将生成错误并返回给调用者。"
3. 将消息有效内容传递到 z/OS Connect，以执行从字节数组到 JSON 的数据转换。
4. 获取数据变换的结果，并将其作为 HTTP 方法的响应返回。

根据收到的消息类型，您可能需要执行一些额外的配置。这是因为 MQ Service Provider 需要将接收到的消息有效内容转换为正确的格式以传递到配置的数据变换。

缺省数据变换期望有效内容使用 CCSID 37，但是 z/OS 资产可能未生成此 CCSID 中的消息。

所需的配置取决于是否接收到 **BytesMessage** 还是 **TextMessage**，以及您是使用 z/OS Connect V1 还是 z/OS Connect EE。

在 z/OS Connect V1 上接收到 BytesMessage

如果要接收 **BytesMessage**，那么可以在用于接收消息的队列定义上指定 **receiveConversion="QMGR"** 和 **receiveCCSID="37"** 属性。

这在以下示例中进行了说明。在这种情况下，**sampleQ2Receive** 定义同时设置了 **receiveConversion** 和 **receiveCCSID** 属性。

```
<jmsConnectionFactory
  id="sampleCF2"
  jndiName="jms/sampleCF2"
  connectionManagerRef="sampleCF2ConnectionManager">

  <properties.wmqJms
    transportType="BINDINGS"
    queueManager="MQ21"/>
</jmsConnectionFactory>

<connectionManager
  id="sampleCF2ConnectionManager"
  maxPoolSize="5"/>

<jmsQueue id="sampleQ2Send"
  jndiName="jms/sampleQ2Send">

  <properties.wmqJms
    baseQueueName="SampleQ2Send"
    CCSID="37"/>
</jmsQueue>

<jmsQueue id="sampleQ2Receive"
  jndiName="jms/sampleQ2Receive">

  <properties.wmqJms
    baseQueueName="SampleQ2Receive"
    receiveCCSID="37"
    receiveConversion="QMGR"/>
</jmsQueue>

<zosConnectService
  id="samplezOSConnectService2"
  invokeURI="/samplezOSConnectService2"
  serviceName="samplezOSConnectService2_name"
  serviceRef="samplezOSConnectService2_MQ"
  dataXformRef="xformJSON2Byte"/>

<mqzOSConnectService
  id="samplezOSConnectService2_MQ"
  connectionFactory="jms/sampleCF2"
  mqmdFormat="AFORMAT"
  destination="jms/sampleQ2Send"
  replyDestination="jms/sampleQ3Receive"/>

<zosConnectDataXform id="xformJSON2Byte"
  bindFileLoc="/XFORM_ROOT/bindfiles" bindFileSuffix=".bnd"
  requestSchemaLoc="/XFORM_ROOT/json" requestSchemaSuffix=".json"
  responseSchemaLoc="/XFORM_ROOT/json"
  responseSchemaSuffix=".json"/>
```

在 z/OS Connect EE 上接收到 BytesMessage

如果要接收 **BytesMessage**，那么可以在用于接收消息的队列定义上指定 **receiveConversion="QMGR"** 和 **receiveCCSID="37"** 属性。

这在以下示例中进行了说明。在这种情况下，**sampleQ2Receive** 定义同时设置了 **receiveConversion** 和 **receiveCCSID** 属性。

```
<jmsConnectionFactory
  id="sampleCF2"
  jndiName="jms/sampleCF2"
  connectionManagerRef="sampleCF2ConnectionManager">

  <properties.wmqJms
```

```

        transportType="BINDINGS"
        queueManager="MQ21" />
</jmsConnectionFactory>

<connectionManager
  id="sampleCF2ConnectionManager"
  maxPoolSize="5" />

<jmsQueue id="sampleQ2Send"
  jndiName="jms/sampleQ2Send">

  <properties.wmqJms
    baseQueueName="SampleQ2Send"
    CCSID="37" />
</jmsQueue>

<jmsQueue id="sampleQ2Receive"
  jndiName="jms/sampleQ2Receive">

  <properties.wmqJms
    baseQueueName="SampleQ2Receive"
    receiveCCSID="37"
    receiveConversion="QMGR" />
</jmsQueue>

<zosconnect_zosConnectService
  id="samplezOSConnectService2"
  invokeURI="/samplezOSConnectService2"
  serviceName="samplezOSConnectService2_name"
  serviceRef="samplezOSConnectService2_MQ"
  dataXformRef="xformJSON2Byte" />

<mqzosconnect_mqzOSConnectService
  id="samplezOSConnectService2_MQ"
  connectionFactory="jms/sampleCF2"
  mqmdFormat="AFORMAT"
  destination="jms/sampleQ2Send"
  replyDestination="jms/sampleQ3Receive" />

<zosconnect_zosConnectDataXform id="xformJSON2Byte"
  bindFileLoc="/XFORM_ROOT/bindfiles" bindFileSuffix=".bnd"
  requestSchemaLoc="/XFORM_ROOT/json" requestSchemaSuffix=".json"
  responseSchemaLoc="/XFORM_ROOT/json"
  responseSchemaSuffix=".json" />

```

在 z/OS Connect V1 上接收到 TextMessage

如果要接收 **TextMessage**，那么您需要将消息转换为数据转换所期望的 CCSID (缺省情况下为 37)。

如果使用定制数据变换，并且变换期望的 CCSID 与 37 不同，那么可以在 [mqzOSConnect 服务元素](#) 中指定 [receiveTextCCSID](#) 属性以设置相应的 CCSID。

这在以下示例中进行了说明。在这种情况下，样本 `zOSConnectService3_MQ` 定义将 [mqzOSConnect 服务元素](#) 属性中的 [receiveTextCCSID](#) 属性设置为 1208 (UTF-8)。

```

<jmsConnectionFactory
  id="sampleCF3"
  jndiName="jms/sampleCF3"
  connectionManagerRef="sampleCF3ConnectionManager">

  <properties.wmqJms
    transportType="BINDINGS"
    queueManager="MQ21" />
</jmsConnectionFactory>

<connectionManager
  id="sampleCF3ConnectionManager"
  maxPoolSize="5" />

<jmsQueue id="sampleQ3Send"
  jndiName="jms/sampleQ3Send">

  <properties.wmqJms
    baseQueueName="SampleQ3Send"
    CCSID="37" />
</jmsQueue>

```

```

<jmsQueue id="sampleQ3Receive"
  jndiName="jms/sampleQ3Receive">

  <properties.wmqJms
    baseQueueName="SampleQ3Receive"/>
</jmsQueue>

<zosConnectService
  id="samplezOSConnectService3"
  invokeURI="/samplezOSConnectService3"
  serviceName="samplezOSConnectService3_name"
  serviceRef="samplezOSConnectService3_MQ"
  dataXformRef="customDataXForm"/>

<mqzOSConnectService
  id="samplezOSConnectService3_MQ"
  connectionFactory="jms/sampleCF3"
  mqmdFormat="AFORMAT"
  destination="jms/sampleQ3Send"
  replyDestination="jms/sampleQ3Receive"
  receiveTextCCSID="1208"/>

```

在 z/OS Connect EE 上接收到 TextMessage

如果要接收 **TextMessage**，那么您需要将消息转换为数据转换所期望的 CCSID (缺省情况下为 37)。

如果使用定制数据变换，并且变换期望的 CCSID 与 37 不同，那么可以在 [mqzOSConnect 服务元素](#) 中指定 `receiveTextCCSID` 属性以设置相应的 CCSID。

这在以下示例中进行了说明。在这种情况下，样本 `zOSConnectService3_MQ` 定义将 [mqzOSConnect 服务元素](#) 属性中的 `receiveTextCCSID` 属性设置为 1208 (UTF-8)。

```

<jmsConnectionFactory
  id="sampleCF3"
  jndiName="jms/sampleCF3"
  connectionManagerRef="sampleCF3ConnectionManager">

  <properties.wmqJms
    transportType="BINDINGS"
    queueManager="MQ21"/>
</jmsConnectionFactory>

<connectionManager
  id="sampleCF3ConnectionManager"
  maxPoolSize="5"/>

<jmsQueue id="sampleQ3Send"
  jndiName="jms/sampleQ3Send">

  <properties.wmqJms
    baseQueueName="SampleQ3Send"
    CCSID="37"/>
</jmsQueue>

<jmsQueue id="sampleQ3Receive"
  jndiName="jms/sampleQ3Receive">

  <properties.wmqJms
    baseQueueName="SampleQ3Receive"/>
</jmsQueue>

<zosconnect_zosConnectService
  id="samplezOSConnectService3"
  invokeURI="/samplezOSConnectService3"
  serviceName="samplezOSConnectService3_name"
  serviceRef="samplezOSConnectService3_MQ"
  dataXformRef="customDataXForm"/>

<mqzosconnect_mqzOSConnectService
  id="samplezOSConnectService3_MQ"
  connectionFactory="jms/sampleCF3"
  mqmdFormat="AFORMAT"
  destination="jms/sampleQ3Send"
  replyDestination="jms/sampleQ3Receive"
  receiveTextCCSID="1208"/>

```

MQ Service Provider 为使用 IBM MQ 的现有应用程序提供 RESTful 接口。这些应用程序可以使用 IBM MQ classes for JMS 或消息队列接口 (MQI) 与 IBM MQ 进行交互。

缺省情况下，IBM MQ classes for JMS 发送包含 MQRFH2 头的消息。但是，大多数 MQI 应用程序不使用 MQRFH2 头。

如果 IBM MQ classes for JMS 应用程序与尚未设计为使用 MQRFH2 头的 MQI 应用程序进行交互，那么需要进行配置以阻止 IBM MQ classes for JMS 发送包含 MQRFH2 头的消息。请参阅 [将 JMS 消息映射到 IBM MQ 消息](#) 以获取更多信息。

如果 MQ Service Provider 要将消息发送到队列 (将由 MQI 应用程序使用)，而这些队列不需要 MQRFH2 头，那么您需要配置 z/OS Connect 以防止发送 MQRFH2 头。

通过将 **targetClient="MQ"** 属性添加到 `server.xml` 中的相关 IBM MQ 消息传递提供程序队列来实现此配置。

以下样本配置显示了 **id** 为 `mqiService` 的单向 MQ Service Provider 服务。反过来，这已配置为将 MQ 消息传递提供程序队列与 **id** `mqiQueue` 配合使用。

已使用 **targetClient="MQ"** 属性配置 `mqiQueue`。这意味着，当应用程序向该队列发送 JMS 消息 (例如，如果应用程序向 `mqiService` 发出 HTTP POST) 时，该消息将不会添加 MQRFH2 头。

z/OS 连接 V1

```
<mqzOSConnectService
  id="mqiService"
  connectionFactory="jms/mqiCF"
  destination="jms/mqiQueue"/>
<jmsQueue
  id="mqiQueue"
  jndiName="jms/mqiQueue">
  <properties.wmqJms
    baseQueueName="MQIQueue"
    targetClient = "MQ"/>
</jmsQueue>
```

z/OS 连接 EE

```
<mqzosconnect_mqzOSConnectService
  id="mqiService"
  connectionFactory="jms/mqiCF"
  destination="jms/mqiQueue"/>
<jmsQueue
  id="mqiQueue"
  jndiName="jms/mqiQueue">
  <properties.wmqJms
    baseQueueName="MQIQueue"
    targetClient = "MQ"/>
</jmsQueue>
```


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示的）保证，包括但不限于暗示的有关非侵权，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示的保证。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 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 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: