8.0

管理 *IBM MQ*

附註

使用本資訊及其支援的產品之前，請先閱讀第 321 頁的『注意事項』中的資訊。

目錄

管理 IBM MQ

管理佇列管理程式及相關聯資源包括您經常執行以啟動及管理那些資源的作業。 選擇您偏好用來管理佇列管理程式及相關聯資源的方法。

您可以在本端或遠端管理 IBM MQ 物件，請參閱 第 7 頁的『本端和遠端管理』。

您可以使用許多不同的方法，在「IBM MQ」中建立及管理佇列管理程式及其相關資源。 這些方法包括指令行介面、圖形使用者介面及管理 API。 如需每一個介面的相關資訊，請參閱本主題中的小節及鏈結。

視您的平台而定，您可以使用不同的指令集來管理 IBM MQ：

- 第 5 頁的『IBM MQ 控制指令』
- 第 5 頁的『IBM MQ Script (MQSC) 指令』
- 第 6 頁的『可程式指令格式 (PCF)』
- IBM i 第 6 頁的『IBM i 控制語言 (CL)』

還有下列其他選項可用來建立及管理 IBM MQ 物件:

- 第 6 頁的『MQ Explorer』
- 第 7 頁的『Windows 預設配置應用程式』
- 第 7 頁的『Microsoft 叢集服務 (MSCS)』

z/OS 如需 IBM MQ for z/OS 上管理介面及選項的相關資訊，請參閱 第 221 頁的『管理 IBM MQ for z/OS』。

您可以使用 PCF 指令來自動化本端及遠端佇列管理程式的部分管理及監視作業。 在某些平台上使用「IBM MQ 管理介面 (MQAI)」也可以簡化這些指令。 如需自動化管理作業的相關資訊，請參閱 第 8 頁的『自動化管理作業』。

IBM MQ 控制指令

控制指令可讓您對佇列管理程式本身執行管理作業。

IBM MQ for Windows， UNIX 和 Linux® 系統提供您在系統指令行發出的 控制指令 。

控制指令在 在分散式平台上建立及管理佇列管理程式中有說明。 如需控制指令的指令參考手冊，請參閱 IBM MQ 控制指令。

IBM MQ Script (MQSC) 指令

使用 MQSC 指令來管理佇列管理程式物件，包括佇列管理程式本身、佇列、程序定義、名稱清單、通道、用戶端連線通道、接聽器、服務及鑑別資訊物件。

您可以使用 `runmqsc` 指令，向佇列管理程式發出 MQSC 指令。 您可以透過互動方式執行此動作，從鍵盤發出指令，或者您可以重新導向標準輸入裝置 (stdin)，以從 ASCII 文字檔執行一系列指令。 在這兩種情況下，指令的格式都相同。

視指令上設定的旗標而定，您可以在三種模式下執行 `runmqsc` 指令:

- 驗證模式，其中 MQSC 指令會在本端佇列管理程式上進行驗證，但不會執行
- 直接模式，其中 MQSC 指令在本端佇列管理程式上執行
- 間接模式，其中 MQSC 指令在遠端佇列管理程式上執行

MQSC 指令中指定的物件屬性會以大寫形式 (例如 RQMNAME) 顯示在此區段中，雖然它們不區分大小寫。 MQSC 指令屬性名稱限制為 8 個字元。

MQSC 指令可在 IBM i z/OS 所有平台上使用，包括 IBM i 及 z/OS。 比較指令集中彙總了 MQSC 指令。

在 Windows、 UNIX 或 Linux 上，您可以使用 MQSC 作為在系統指令行發出的單一指令。 若要發出更複雜或多個指令，可以將 MQSC 建置在您從 Windows、 UNIX 或 Linux 系統指令行執行的檔案中。 MQSC 可以傳送至遠端佇列管理程式。 如需完整資料，請參閱 建置指令 Script。

IBM i 在 IBM i 上，若要在 IBM i 伺服器上發出指令，請在 Script 檔中建立指令清單，然後使用 STRMQMQSC 指令執行該檔案。

附註: IBM i

1. 請勿使用 QTEMP 程式庫作為 STRMQMQSC 的輸入程式庫，因為 QTEMP 程式庫的使用受到限制。 您必須使用另一個檔案庫作為指令的輸入檔。
2. 在 IBM i 上，從 Script 檔發出之指令的 MQSC 回應會在排存檔中傳回。

第 67 頁的『Script (MQSC) 指令』 包含每一個 MQSC 指令及其語法的說明。

如需在本端管理中使用 MQSC 指令的相關資訊，請參閱 第 66 頁的『使用 MQSC 指令執行本端管理作業』。

可程式指令格式 (PCF)

「可程式指令格式 (PCF)」定義可在程式與網路中任何佇列管理程式 (支援 PCF) 之間交換的指令及回覆訊息。 您可以在系統管理應用程式中使用 PCF 指令來管理 IBM MQ 物件: 鑑別資訊物件、通道、通道接聽器、名稱清單、程序定義、佇列管理程式、佇列、服務及儲存類別。 應用程式可以從網路中的單一點運作，以使用本端佇列管理程式與任何佇列管理程式 (本端或遠端) 通訊指令及回覆資訊。

如需 PCF 的相關資訊，請參閱 第 9 頁的『可程式指令格式簡介』。

如需指令及回應的 PCF 及結構定義，請參閱 可程式化指令格式參照。

IBM i 控制語言 (CL)

IBM i

此語言可用來向 IBM MQ for IBM i 發出管理指令。 可以在指令行或寫入 CL 程式來發出指令。 這些指令執行與 PCF 指令類似的功能，但格式不同。 CL 指令是專為伺服器而設計，而 CL 回應是設計為人類可讀的，而 PCF 指令與平台無關，且指令及回應格式都是供程式使用。

如需「IBM i 控制語言 (CL)」的完整資料，請參閱 IBM MQ for IBM i CL 指令。

MQ Explorer

使用 MQ Explorer，您可以執行下列動作:

- 定義及控制各種資源，包括佇列管理程式、佇列、程序定義、名稱清單、通道、用戶端連線通道、接聽器、服務及叢集。
- 啟動或停止本端佇列管理程式及其相關聯的處理程序。
- 檢視工作站上或其他工作站中的佇列管理程式及其相關聯物件。
- 檢查佇列管理程式、叢集及通道的狀態。
- 從佇列狀態查看哪些應用程式、使用者或通道已開啟特定佇列。

在 Windows 和 Linux 系統上，您可以使用系統功能表、 `MQExplorer` 執行檔或 **strmqcfg** 指令來啟動 MQ Explorer 。

在 Linux 上，若要順利啟動 MQ Explorer ，您必須能夠將檔案寫入起始目錄，且起始目錄必須存在。

如需相關資訊，請參閱第 54 頁的『使用 MQ Explorer 進行管理』。

您可以使用 MQ Explorer 來管理其他平台 (包括 z/OS) 上的遠端佇列管理程式，以取得詳細資料並下載 SupportPac MS0T，請參閱 https://www.ibm.com/support/docview.wss?uid=swg24021041。

Windows 預設配置應用程式

您可以使用 Windows 預設配置程式來建立一組 入門範本 (或預設) IBM MQ 物件。 表 1: Windows 預設配置應用程式所建立的物件中列出所建立預設物件的摘要。

Microsoft 叢集服務 (MSCS)

Microsoft 叢集服務 (MSCS) 可讓您將伺服器連接至 叢集，以提供更高的資料及應用程式可用性，並讓您更容易管理系統。 MSCS 可以自動偵測及回復伺服器或應用程式的故障情形。

請務必不要混淆 MSCS 意義上的叢集與 IBM MQ 叢集。 區別是:

IBM MQ 叢集
是一部以上電腦上兩個以上佇列管理程式的群組，提供自動互連，並容許在它們之間共用佇列以進行負載平衡及備援。

MSCS 叢集
連接在一起並配置的電腦群組，如果其中一部電腦失敗， MSCS 會執行 失效接手，將應用程式的狀態資料從失敗的電腦傳送至叢集中的另一部電腦，並在該處重新起始其作業。

支援 Microsoft 叢集服務 (MSCS) 提供如何將 IBM MQ for Windows 系統配置成使用 MSCS 的詳細資訊。

相關概念
第 62 頁的『管理本端 IBM MQ 物件』
本節告訴您如何管理本端 IBM MQ 物件，以支援使用「訊息佇列介面 (MQI)」的應用程式。 在此環境定義中，本端管理是指建立、顯示、變更、複製及刪除 IBM MQ 物件。

第 114 頁的『管理遠端 IBM MQ 物件』

第 157 頁的『管理 IBM i』
介紹可讓您在 IBM i 上管理 IBM MQ 的方法。

第 221 頁的『管理 IBM MQ for z/OS』
管理佇列管理程式及相關聯資源包括您經常執行以啟動及管理那些資源的作業。 選擇您偏好用來管理佇列管理程式及相關聯資源的方法。

相關資訊
IBM MQ 技術概觀
規劃
z/OS 在 z/OS 上規劃 IBM MQ 環境
配置
z/OS 配置 z/OS
交易式支援實務
失去與 XA 資源管理程式的聯絡時的考量

本端和遠端管理

您可以在本端或遠端管理 IBM MQ 物件。

本端管理 表示對您在本端系統上定義的任何佇列管理程式執行管理作業。 您可以存取其他系統，例如透過 TCP/IP 終端機模擬程式 **telnet**，並在那裡執行管理。 在 IBM MQ 中，您可以將此視為本端管理，因為不涉及任何通道，即通訊由作業系統管理。

IBM MQ 支援透過稱為 遠端管理的單一聯絡點進行管理。 這可讓您從本端系統發出在另一個系統上處理的指令，並同時套用至「 IBM MQ 檔案總管」。 例如，您可以發出遠端指令來變更遠端佇列管理程式上的佇列定義。 您不需要登入該系統，雖然您需要定義適當的通道。 目標系統上的佇列管理程式及指令伺服器必須在執行中。

有些指令無法以這種方式發出，尤其是建立或啟動佇列管理程式，以及啟動指令伺服器。 若要執行這種類型的作業，您必須登入遠端系統並從該處發出指令，或建立可以為您發出指令的處理程序。 此限制也適用於 IBM MQ Explorer。

第 114 頁的『管理遠端 IBM MQ 物件』 更詳細地說明遠端管理的主題。

如何使用 IBM MQ 控制指令

本節說明如何使用 IBM MQ 控制指令。

如果您想要發出控制指令，您的使用者 ID 必須是大部分控制指令的 mqm 群組成員。 如需此作業的相關資訊，請參閱 在 UNIX、 Linux 及 Windows 系統上管理 IBM MQ 的權限。 此外，請注意下列環境特定資訊:

IBM MQ for Windows
所有控制指令都可以從指令行發出。 指令名稱及其旗標不區分大小寫: 您可以用大寫、小寫或大小寫的組合來輸入它們。 不過，控制指令 (例如佇列名稱) 的引數會區分大小寫。

在語法說明中，以連字號 (-) 作為旗標指示器。 您可以使用正斜線 (/) 而非連字號。

IBM MQ for UNIX 和 Linux 系統
所有 IBM MQ 控制指令都可以從 Shell 發出。 所有指令都須區分大小寫。

可以使用「IBM MQ 檔案總管」來發出控制指令的子集。

如需相關資訊，請參閱 IBM MQ 控制指令

自動化管理作業

您可能會決定將部分管理及監視作業自動化，對您的安裝是有益的。 您可以使用可程式化指令格式 (PCF) 指令，將本端及遠端佇列管理程式的管理作業自動化。 本節假設您有管理 IBM MQ 物件的經驗。

PCF 指令

IBM MQ 可程式指令格式 (PCF) 指令可用來將管理作業程式設計到管理程式中。 如此一來，從程式中，您可以操作佇列管理程式物件 (佇列、程序定義、名稱清單、通道、用戶端連線通道、接聽器、服務及鑑別資訊物件)，甚至自行操作佇列管理程式。

PCF 指令涵蓋 MQSC 指令所提供的相同函數範圍。 您可以撰寫程式，從單一節點向網路中的任何佇列管理程式發出 PCF 指令。 透過此方式，您可以將管理作業集中化及自動化。

每一個 PCF 指令都是內嵌在 IBM MQ 訊息的應用程式資料部分中的資料結構。 每一個指令都會以與任何其他訊息相同的方式，使用 MQI 功能 MQPUT 來傳送至目標佇列管理程式。 如果指令伺服器正在接收訊息的佇列管理程式上執行，則指令伺服器會將它解譯為指令訊息並執行指令。 為了取得回覆，應用程式會發出 MQGET 呼叫，並以另一個資料結構傳回回覆資料。 然後，應用程式可以處理回覆並相應地採取行動。

註: 與 MQSC 指令不同， PCF 指令及其回覆不是您可以讀取的文字格式。

簡言之，以下是建立 PCF 指令訊息所需的部分內容:

訊息描述子
這是標準 IBM MQ 訊息描述子，其中:

- 訊息類型 (*MsqType*) 為 MQMT_REQUEST。
- 訊息格式 (*Format*) 是 MQFMT_ADMIN。

應用程式資料
包含 PCF 訊息 (包括 PCF 標頭)，其中:

- PCF 訊息類型 (*Type*) 指定 MQCFT_COMMAND。
- 指令 ID 指定指令，例如 *Change Queue* (MQCMD_CHANGE_Q)。

如需 PCF 資料結構及其實作方式的完整說明，請參閱 第 9 頁的『可程式指令格式簡介』。

PCF 物件屬性

PCF 中的物件屬性不限於 8 個字元，因為它們適用於 MQSC 指令。 它們以斜體顯示在本手冊中。 例如，RQMNAME 的 PCF 對等項目為 *RemoteQMgrName*。

跳出 PCF

Escape PCF 是在訊息文字內包含 MQSC 指令的 PCF 指令。 您可以使用 PCF 將指令傳送至遠端佇列管理程式。 如需跳出 PCF 的相關資訊，請參閱 跳出。

可程式指令格式簡介

「可程式指令格式 (PCF)」定義可在程式與網路中任何佇列管理程式 (支援 PCF) 之間交換的指令及回覆訊息。 PCF 可簡化佇列管理程式管理及其他網路管理。 它們可以用來解決分散式網路的複雜管理問題，尤其是當網路的大小和複雜性增加時。

下列項目支援本產品說明文件中說明的「可程式指令格式」：

- 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
- IBM WebSphere MQ for HP Integrity NonStop Server

問題 PCF 指令解決

分散式網路的管理可能會變得複雜。 隨著網路的規模和複雜性增加，管理問題持續增加。

傳訊和佇列作業特有的管理範例包括:

- 資源管理。

  例如，佇列建立和刪除。
- 效能監視。

  例如，佇列深度上限或訊息速率。
- 控制。

  例如，調整佇列參數，例如佇列深度上限、訊息長度上限，以及啟用和停用佇列。
- 訊息遞送。

  透過網路的替代路徑定義。

IBM MQ PCF 指令可用來簡化佇列管理程式管理及其他網路管理。 PCF 指令可讓您使用單一應用程式，從網路內的單一佇列管理程式執行網路管理。

何謂 PCF?

PCF 定義可在程式與網路中任何佇列管理程式 (支援 PCF) 之間交換的指令及回覆訊息。 您可以在系統管理應用程式中使用 PCF 指令來管理 IBM MQ 物件: 鑑別資訊物件、通道、通道接聽器、名稱清單、程序定義、佇列管理程式、佇列、服務及儲存類別。 應用程式可以從網路中的單一點運作，以使用本端佇列管理程式與任何佇列管理程式 (本端或遠端) 通訊指令及回覆資訊。

每一個佇列管理程式都有一個具有標準佇列名稱的管理佇列，且您的應用程式可以將 PCF 指令訊息傳送至該佇列。 每一個佇列管理程式也有一個指令伺服器，可處理來自管理佇列的指令訊息。 因此，網路中的任何佇列管理程式都可以處理 PCF 指令訊息，而且可以使用您指定的回覆佇列，將回覆資料傳回您的應用程式。 PCF 指令及回覆訊息是使用一般「訊息佇列介面 (MQI)」來傳送及接收。

如需可用 PCF 指令 (包括其參數) 的清單，請參閱 可程式指令格式的定義。

使用可程式指令格式

您可以在系統管理程式中使用 PCF 進行 IBM MQ 遠端管理。

本節包括:

- 第 10 頁的『PCF 指令訊息』
- 第 12 頁的『回應』
- z/OS 第 14 頁的『延伸回應』
- IBM MQ 物件的命名規則
- 第 15 頁的『PCF 指令的權限檢查』

PCF 指令訊息

PCF 指令訊息包含 PCF 標頭、該標頭中所識別的參數，以及使用者定義的訊息資料。 訊息是使用「訊息佇列」介面呼叫來發出。

每一個指令及其參數都會以個別指令訊息形式傳送，其中包含 PCF 標頭，後面接著許多參數結構; 如需 PCF 標頭的詳細資料，請參閱 MQCFH-PCF 標頭，如需參數結構的範例，請參閱 MQCFST-PCF 字串參數。 PCF 標頭會識別相同訊息中的指令及後面的參數結構數目。 每一個參數結構都會提供一個參數給指令。

指令伺服器所產生之指令的回覆具有類似的結構。 有一個 PCF 標頭，後面接著一些參數結構。 回覆可以由多個訊息組成，但指令一律只由一個訊息組成。

在 z/OS 以外的平台上， PCF 指令傳送至的佇列一律稱為 SYSTEM.ADMIN.COMMAND.QUEUE。

z/OS 在 z/OS 上，指令會傳送至 SYSTEM.COMMAND.INPUT(雖然 SYSTEM.ADMIN.COMMAND.QUEUE 可以是它的別名。 處理此佇列的指令伺服器會將回覆傳送至指令訊息的訊息描述子中 *ReplyToQ* 及 *ReplyToQMgr* 欄位所定義的佇列。

如何發出 PCF 指令訊息

使用一般「訊息佇列介面 (MQI)」呼叫、MQPUT、MQGET 等，在其佇列中放置及擷取 PCF 指令及回應訊息。

註:

請確定指令伺服器正在目標佇列管理程式上執行，以供 PCF 指令在該佇列管理程式上處理。

如需所提供標頭檔的清單，請參閱 IBM MQ COPY、標頭、併入和模組檔案。

PCF 指令的訊息描述子

IBM MQ 訊息描述子完整記錄在 MQMD-訊息描述子中。

PCF 指令訊息在訊息描述子中包含下列欄位:

Report
任何有效值 (視需要)。

MsgType
此欄位必須是 MQMT_REQUEST ， 以指出需要回應的訊息。

Expiry
任何有效值 (視需要)。

Feedback
設為 MQFB_NONE

Encoding
如果您要傳送至 IBM i IBM i、 Windows、 UNIX 或 Linux 系統，請將此欄位設為用於訊息資料的編碼; 必要的話， 會執行轉換。

CodedCharSetId
如果您要傳送至

IBM i,

Windows、 UNIX 或 Linux 系統，將此欄位設為用於訊息資料的編碼字集 ID; 必要的話，會執行轉換。

Format
設為 MQFMT_ADMIN。

Priority
任何有效值 (視需要)。

Persistence
任何有效值 (視需要)。

MsgId
傳送端應用程式可以指定任何值，也可以指定 MQMI_NONE 來要求佇列管理程式產生唯一訊息 ID。

CorrelId
傳送端應用程式可以指定任何值，也可以指定 MQCI_NONE 以指出沒有相關性 ID。

ReplyToQ
接收回應的佇列名稱。

ReplyToQMgr
回應的佇列管理程式名稱 (或空白)。

訊息環境定義欄位
視需要，這些欄位可以設為任何有效值。 一般而言，放置訊息選項 MQPMO_DEFAULT_CONTEXT 是用來將訊息環境定義欄位設為預設值。

如果您使用 version-2 MQMD 結構，則必須設定下列其他欄位:

GroupId
設為 MQGI_NONE

MsgSeqNumber
設為 1

Offset
設為 0

MsgFlags
設為 MQMF_NONE

OriginalLength
設為 MQOL_UNDEFINED

傳送使用者資料

PCF 結構也可以用來傳送使用者定義的訊息資料。 在此情況下，訊息描述子 *Format* 欄位必須設為 MQFMT_PCF。

在指定佇列中傳送及接收 *PCF* 訊息

將 PCF 訊息傳送至指定的佇列

若要將訊息傳送至指定佇列， mqPutBag 呼叫會將指定工具袋的內容轉換為 PCF 訊息，並將訊息傳送至指定佇列。 在呼叫之後，袋子的內容保持不變。

作為此呼叫的輸入，您必須提供:

- MQI 連線控點。
- 要放置訊息之佇列的物件控點。
- 訊息描述子。 如需訊息描述子的相關資訊，請參閱 MQMD-訊息描述子。

- 使用 MQPMO 結構放置訊息選項。 如需 MQPMO 結構的相關資訊，請參閱 MQPMO-Put-message 選項。
- 要轉換為訊息之工具袋的控點。

  註: 如果工具袋包含管理訊息，且已使用 mqAddInquiry 呼叫將值插入工具袋，則 MQIASY_COMMAND 資料項目的值必須是 MQAI 可辨識的 INQUIRE 指令。

如需 mqPutBag 呼叫的完整說明，請參閱 mqPutBag。

從指定佇列接收 PCF 訊息

若要從指定佇列接收訊息， mqGetBag 呼叫會從指定佇列取得 PCF 訊息，並將訊息資料轉換為資料工具袋。

作為此呼叫的輸入，您必須提供:

- MQI 連線控點。
- 要從中讀取訊息之佇列的物件控點。
- 訊息描述子。 在 MQMD 結構內， `Format` 參數必須是 MQFMT_ADMIN、MQFMT_EVENT 或 MQFMT_PCF。

  註: 如果在工作單元內收到訊息 (亦即，使用 MQGMO_SYNCPOINT 選項) ，且訊息具有不受支援的格式，則可以取消工作單元。 然後會在佇列上恢復該訊息，並且可以使用 MQGET 呼叫而非 mqGetBag 呼叫來擷取該訊息。 如需訊息描述子的相關資訊，請參閱 MQGMO-Get-message 選項。
- 使用 MQGMO 結構取得訊息選項。 如需 MQGMO 結構的相關資訊，請參閱 MQMD-訊息描述子。
- 要包含已轉換訊息之工具袋的控點。

如需 mqGetBag 呼叫的完整說明，請參閱 mqGetBag。

回應

為了回應每一個指令，指令伺服器會產生一或多個回應訊息。 回應訊息的格式與指令訊息類似。

PCF 標頭與它作為回應的指令具有相同的指令 ID 值 (如需詳細資料，請參閱 MQCFH-PCF 標頭)。 根據要求的報告選項來設定訊息 ID 和相關性 ID。

如果指令訊息的 PCF 標頭類型是 MQCFT_COMMAND ，則只會產生標準回應。 z/OS 以外的所有平台都支援這類指令。 較舊的應用程式在 z/OS 上不支援 PCF; 「IBM MQ Windows 探險家」是這類應用程式 (不過，6.0 版 或更新版本 IBM MQ Explorer 在 z/OS 上確實支援 PCF)。

如果指令訊息的 PCF 標頭類型是 MQCFT_COMMAND_XR ，則會產生延伸或標準回應。 這類指令在 z/OS 及部分其他平台上受支援。 在 z/OS 上發出的指令只會產生延伸回應。 在其他平台上，可能會產生任一類型的回應。

如果單一指令指定同屬物件名稱，則會針對每一個相符物件在其自己的訊息中傳回個別回應。 對於回應產生，具有通用名稱的單一指令會被視為多個個別指令 (除了控制欄位 MQCFC_LAST 或 MQCFC_NOT_LAST 之外)。 否則，一個指令訊息會產生一個回應訊息。

某些 PCF 回應可能會傳回結構，即使未要求也一樣。 此結構顯示在回應的定義中 (可程式指令格式的定義) 如 一律傳回。 對於這些回應，需要為回應中的物件命名的原因，以識別套用資料的物件。

回應的訊息描述子

回應訊息在訊息描述子中具有下列欄位:

MsgType
此欄位是 MQMT_REPLY。

MsgId
此欄位由佇列管理程式產生。

CorrelId
此欄位是根據指令訊息的報告選項所產生。

Format
此欄位是 MQFMT_ADMIN。

Encoding
設為 MQENC_NATIVE。

CodedCharSetId
設為 MQCCSI_Q_MGR。

Persistence
與指令訊息中的相同。

Priority
與指令訊息中的相同。

使用 MQPMO_PASS_IDENTITY_CONTEXT 產生回應。

標準回應

標頭類型為 MQCFT_COMMAND 的指令訊息，會產生標準回應。 z/OS 以外的所有平台都支援這類指令。

標準回應有三種類型:

- 確定回應
- 錯誤回應
- 資料回應

確定回應

此回應包含以指令格式標頭開頭且 *CompCode* 欄位為 MQCC_OK 或 MQCC_WARNING 的訊息。

若為 MQCC_OK， *Reason* 是 MQRC_NONE。

若為 MQCC_WARNING， *Reason* 會識別警告的本質。 在此情況下，指令格式標頭後面可能會接著一或多個適用於此原因碼的警告參數結構。

在任一情況下，對於 inquire 指令，可能會遵循下列各節中說明的進一步參數結構。

錯誤回應

如果指令有錯誤，則會傳送一或多個錯誤回應訊息 (即使指令通常只有單一回應訊息，也可能會傳送多個錯誤回應訊息)。 這些錯誤回應訊息會適當地設定 MQCFC_LAST 或 MQCFC_NOT_LAST。

每一個這類訊息都以回應格式標頭開頭， *CompCode* 值為 MQCC_FAILED，且 *Reason* 欄位可識別特定錯誤。 一般而言，每一則訊息會說明不同的錯誤。 此外，每則訊息在標頭後面都有零或一個 (永不超過一個) 錯誤參數結構。 此參數結構 (如果有的話) 是 MQCFIN 結構，且 *Parameter* 欄位包含下列其中一項:

- MQIACF_PARAMETER_ID

  結構中的 *Value* 欄位是錯誤參數 (例如 MQCA_Q_NAME) 的參數 ID。

- MQIACF_ERROR_ID

  此值與 MQRC_UNEXPECTED_ERROR 的 *Reason* 值 (在指令格式標頭中) 搭配使用。 MQCFIN 結構中的 *Value* 欄位是指令伺服器收到的非預期原因碼。

- MQIACF_SELECTOR

  如果隨指令傳送的清單結構 (MQCFIL) 包含重複的選取器或無效的選取器，則會發生此值。 指令格式標頭中的 *Reason* 欄位識別錯誤， MQCFIN 結構中的 *Value* 欄位是錯誤指令的 MQCFIL 結構中的參數值。

- MQIACF_ERROR_OFFSET

  當「連線測試通道」指令上發生資料比較錯誤時，即會發生此值。 結構中的 *Value* 欄位是「連線測試通道」比較錯誤的偏移。

- MQIA_CODED_CHAR_SET_ID

  當送入 PCF 指令訊息的訊息描述子中的編碼字集 ID 不符合目標佇列管理程式的編碼字集 ID 時，會發生此值。 結構中的 *Value* 欄位是佇列管理程式的編碼字集 ID。

最後一則 (或僅) 錯誤回應訊息是摘要回應，其 *CompCode* 欄位為 MQCC_FAILED， *Reason* 欄位為 MQRCCF_COMMAND_FAILED。 此訊息在標頭後面沒有參數結構。

資料回應

此回應包含對 inquire 指令的 OK 回應 (如先前所述)。 「正常」回應後面接著包含所要求資料的其他結構，如 可程式指令格式的定義中所述。

應用程式不得相依於以任何特定順序傳回的這些其他參數結構。

延伸回應

在 z/OS 上發出的指令會產生延伸回應。

延伸回應有三種類型:

- 訊息回應，類型為 MQCFT_XR_MSG
- 項目回應，類型為 MQCFT_XR_ITEM
- 摘要回應，類型為 MQCFT_XR_SUMMARY

每一個指令可以產生一個以上回應集。 每一組回應都包含一或多個訊息，從 PCF 標頭的 *MsgSeqNumber* 欄位中的 1 開始循序編號。 每一個集合中最後一個 (或僅) 回應的 *Control* 欄位具有值 MQCFC_LAST。 對於集中的所有其他回應，此值為 MQCFC_NOT_LAST。

任何回應都可以包括一個以上選用 MQCFBS 結構，其中 *Parameter* 欄位設為 MQBACF_RESPONSE_SET，該值是回應集 ID。 ID 是唯一的，可識別包含回應的回應集。 對於每一組回應，都會有一個 MQCFBS 結構來識別它。

延伸回應至少有兩個參數結構:

- *Parameter* 欄位設為 MQBACF_RESPONSE_ID 的 MQCFBS 結構。 此欄位中的值是回應所屬回應集的 ID。 第一個集合中的 ID 是任意的。 在後續集合中，此 ID 是先前在 MQBACF_RESPONSE_SET 結構中通知的 ID。
- *Parameter* 欄位設為 MQCACF_RESPONSE_Q_MGR_NAME 的 MQCFST 結構，值是回應集來自其中的佇列管理程式名稱。

許多回應都有其他參數結構，下列各節會說明這些結構。

除非找到具有 MQCFC_LAST 的回應，否則您無法事先取得回應來判斷集合中的回應數目。 您也無法事先判斷有多少回應集，因為任何集可能包括 MQBACF_RESPONSE_SET 結構，以指出會產生其他集。

Inquire 指令的延伸回應

查詢指令通常會針對找到符合指定搜尋準則的每一個項目產生項目回應 (類型 MQCFT_XR_ITEM)。 項目回應在標頭中具有值為 MQCC_OK 的 *CompCode* 欄位，以及值為 MQRC_NONE 的 *Reason* 欄位。 它還包括說明項目及其所要求屬性的其他參數結構，如 可程式指令格式的定義中所述。

如果項目發生錯誤，則標頭中 *CompCode* 欄位的值為 MQCC_FAILED，且 *Reason* 欄位會識別特定錯誤。 包含其他參數結構以識別項目。

除了項目回應之外，某些 Inquire 指令可能會傳回一般 (非名稱專用) 訊息回應。 這些回應是 MQCFT_XR_MSG 類型的參考或錯誤回應。

如果 Inquire 指令成功，則可能會選擇性地有摘要回應 (類型 MQCFT_XR_SUMMARY)， *CompCode* 值為 MQCC_OK， *Reason* 欄位值為 MQRC_NONE。

如果「查詢」指令失敗，可能會傳回項目回應，且可能選擇性地有摘要回應 (類型 MQCFT_XR_SUMMARY)， *CompCode* 值為 MQCC_FAILED， *Reason* 欄位值為 MQRCCF_COMMAND_FAILED。

對 INQUIRE 以外指令的延伸回應

成功指令會產生訊息回應，其中標頭中 *CompCode* 欄位的值為 MQCC_OK，而 *Reason* 欄位的值為 MQRC_NONE。 一律至少有一則訊息; 它可能是參考資訊 (MQCFT_XR_MSG) 或摘要

(MQCFT_XR_SUMMARY)。 可能有其他參考資訊 (MQCFT_XR_MSG 類型) 訊息。 每一個參考訊息可能包含一些其他參數結構，以及指令的相關資訊; 請參閱個別指令說明，以瞭解可能發生的結構。

失敗的指令會產生錯誤訊息回應 (類型 MQCFT_XR_MSG)，其中標頭中 *CompCode* 欄位的值為 MQCC_FAILED，而 *Reason* 欄位會識別特定錯誤。 每則訊息可能包含許多其他參數結構，以及錯誤的相關資訊: 請參閱可能發生之結構的個別錯誤說明。 可能會產生參考訊息回應。 可能會選擇性地有摘要回應 (MQCFT_XR_SUMMARY)，其 *CompCode* 值為 MQCC_FAILED，而 *Reason* 欄位值為 MQRCC_COMMAND_FAILED。

使用 CommandScope 對指令的延伸回應

如果指令使用 *CommandScope* 參數，或導致產生使用 *CommandScope* 參數的指令，則會從接收指令的佇列管理程式中設定起始回應集。 然後會針對指令所導向的每一個佇列管理程式產生個別的回應集 (如同發出多個個別指令一樣)。 最後，有來自接收端佇列管理程式的回應集，其中包括整體摘要回應 (類型 MQCFT_XR_SUMMARY)。 MQCACF_RESPONSE_Q_MGR_NAME 參數結構可識別產生每一個集合的佇列管理程式。

起始回應集具有下列其他參數結構:

- MQIACF_COMMAND_INFO (MQCFIN)。 此結構中可能的值為 MQCMDI_CMDSCOPE_ACCEPTED 或 MQCMDI_CMDSCOPE_GENERATED。
- MQIACF_CMDSCOPE_Q_MGR_COUNT (MQCFIN)。 此結構指出指令傳送至其中的佇列管理程式數目。

PCF 指令的權限檢查

處理 PCF 指令時，指令訊息中訊息描述子的 *UserIdentifier* 會用於必要的 IBM MQ 物件權限檢查。 在每一個平台上以不同方式實作權限檢查，如本主題所述。

在處理指令的系統上執行檢查; 因此此使用者 ID 必須存在於目標系統上，且具有處理指令的必要權限。 如果訊息來自遠端系統，則達成目標系統上現有 ID 的方法之一是在本端及遠端系統上都具有相符的使用者 ID。

IBM MQ for IBM i

IBM i

為了處理任何 PCF 指令，使用者 ID 必須具有目標系統上 IBM MQ 物件的 *dsp* 權限。

此外，還會針對特定 PCF 指令執行 IBM MQ 物件權限檢查，如 第 17 頁的表 1 中所示。

在大部分情況下，這些檢查與在本端系統上發出的同等 IBM MQ CL 指令所執行的檢查相同。 如需從 IBM MQ 權限至 IBM i 系統權限之對映的相關資訊，以及 IBM MQ CL 指令的權限需求，請參閱 在 IBM i 上設定安全。 有關結束程式的安全詳細資料在 使用安全結束程式的鏈結層次安全 文件中提供。

若要處理下列任何指令，使用者 ID 必須是群組設定檔 QMQMADM 的成員:

- Ping 通道
- 變更通道
- 複製通道
- 建立通道
- 刪除通道
- 重設通道
- 解析通道
- 啟動通道
- 停止通道
- 啟動通道起始程式
- 啟動通道接聽器

IBM MQ for Windows、 UNIX 和 Linux 系統

Windows Linux UNIX

為了處理任何 PCF 指令，使用者 ID 必須對目標系統上的佇列管理程式物件具有 *dsp* 權限。 此外，還會針對特定 PCF 指令執行 IBM MQ 物件權限檢查，如 第 17 頁的表 1 中所示。

若要處理下列任何指令，使用者 ID 必須屬於群組 *mqm*。

註: 僅限 Windows，使用者 ID 可以屬於群組 *Administrators* 或群組 *mqm*。

- 變更通道
- 複製通道
- 建立通道
- 刪除通道
- Ping 通道
- 重設通道
- 啟動通道
- 停止通道
- 啟動通道起始程式
- 啟動通道接聽器
- 解析通道
- 重設叢集
- 重新整理叢集
- 暫停佇列管理程式
- 回復佇列管理程式

IBM WebSphere MQ for HP Integrity NonStop Server

為了處理任何 PCF 指令，使用者 ID 必須對目標系統上的佇列管理程式物件具有 *dsp* 權限。 此外，還會針對特定 PCF 指令執行 IBM MQ 物件權限檢查，如 第 17 頁的表 1 中所示。

若要處理下列任何指令，使用者 ID 必須屬於群組 *mqm*:

- 變更通道
- 複製通道
- 建立通道
- 刪除通道
- Ping 通道
- 重設通道
- 啟動通道
- 停止通道
- 啟動通道起始程式
- 啟動通道接聽器
- 解析通道
- 重設叢集
- 重新整理叢集
- 暫停佇列管理程式
- 回復佇列管理程式

IBM MQ 物件權限

表 *1: Windows*、 *HP Integrity NonStop Server*、 IBM i *IBM i*、 *UNIX* 和 *Linux* 系統-物件權限

指令	IBM MQ 物件權限	類別權限 (適用於物件類型)
變更鑑別資訊	dsp 和 chg	不適用
變更通道	dsp 和 chg	不適用
變更通道接聽器	dsp 和 chg	不適用
變更用戶端連線通道	dsp 和 chg	不適用
變更名單	dsp 和 chg	不適用
變更處理程序	dsp 和 chg	不適用
變更佇列	dsp 和 chg	不適用
變更佇列管理程式	chg 請參閱附註 *3* 和附註 *5*	不適用
變更服務	dsp 和 chg	不適用
清除佇列	clr	不適用
複製鑑別資訊	dsp	crt
複製鑑別資訊 (取代) 請參閱附註 *1*	*from:* dsp *to:* chg	crt
複製通道	dsp	crt
複製通道 (取代) 請參閱附註 *1*	*from:* dsp *to:* chg	crt
複製通道接聽器	dsp	crt
複製通道接聽器 (取代) 請參閱附註 *1*	*from:* dsp *to:* chg	crt
複製用戶端連線通道	dsp	crt
複製用戶端連線通道 (取代) 請參閱附註 *1*	*from:* dsp *to:* chg	crt
複製名單	dsp	crt
複製名單 (取代) 請參閱附註 *1*	*from:* dsp *to:* dsp and chg	crt
複製處理程序	dsp	crt
複製處理程序 (取代) 請參閱附註 *1*	*from:* dsp *to:* chg	crt
複製佇列	dsp	crt
複製佇列 (取代) 請參閱附註 *1*	*from:* dsp *to:* dsp and chg	crt
建立鑑別資訊	*(*系統預設鑑別資訊*)* dsp	crt
建立鑑別資訊 (取代) 請參閱附註 *1*	*(*系統預設鑑別資訊*)* dsp 至*:* chg	crt
建立通道	*(*系統預設通道*)* dsp	crt
建立通道 (取代) 請參閱附註 *1*	*(*系統預設通道*)* dsp 至*:* chg	crt
建立通道接聽器	*(*系統預設接聽器*)* dsp	crt
建立通道接聽器 (取代) 請參閱附註 *1*	*(*系統預設接聽器*)* dsp 至*:* chg	crt

表 *1: Windows*、 *HP Integrity NonStop Server*、 IBM i *IBM i*、 *UNIX* 和 *Linux* 系統-物件權限 *(*繼續*)*		
指令	**IBM MQ 物件權限**	**類別權限 (適用於物件類型)**
建立用戶端連線通道	*(*系統預設通道*)* dsp	crt
建立用戶端連線通道 (取代) 請參閱附註 *1*	*(*系統預設通道*)* dsp 至*:* chg	crt
建立名單	*(*系統預設名單*)* dsp	crt
建立名單 (取代) 請參閱附註 *1*	*(*系統預設名稱清單*)* dsp 至*:* dsp 及 chg	crt
建立處理程序	*(*系統預設處理程序*)* dsp	crt
建立程序 (取代) 請參閱附註 *1*	*(*系統預設處理程序*)* dsp *:* chg	crt
建立佇列	*(*系統預設佇列*)* dsp	crt
建立佇列 (取代) 請參閱附註 *1*	*(*系統預設佇列*)* dsp *:* dsp 及 chg	crt
建立服務	*(*系統預設佇列*)* dsp	crt
建立服務 (取代) 請參閱附註 *1*	*(*系統預設佇列*)* dsp *:* chg	crt
刪除鑑別資訊	dsp 和 dlt	不適用
刪除權限記錄	*(*佇列管理程式物件*)* chg 請參閱附註 *4*	請參閱附註 *4*
刪除通道	dsp 和 dlt	不適用
刪除通道接聽器	dsp 和 dlt	不適用
刪除用戶端連線通道	dsp 和 dlt	不適用
刪除名單	dsp 和 dlt	不適用
刪除處理程序	dsp 和 dlt	不適用
刪除佇列	dsp 和 dlt	不適用
刪除服務	dsp 和 dlt	不適用
查詢鑑別資訊	dsp	不適用
查詢權限記錄	請參閱附註 *4*	請參閱附註 *4*
查詢通道	dsp	不適用
查詢通道接聽器	dsp	不適用
查詢通道狀態 (適用於 **ChannelType** MQCHT_CLSSDR)	inq	不適用
查詢用戶端連線通道	dsp	不適用
查詢名單	dsp	不適用
查詢處理程序	dsp	不適用
查詢佇列	dsp	不適用
查詢佇列管理程式	請參閱附註 *3*	不適用
查詢佇列狀態	dsp	不適用
查詢服務	dsp	不適用
Ping 通道	ctrl	不適用

表 1: Windows、 HP Integrity NonStop Server、 IBM i IBM i、 UNIX 和 Linux 系統-物件權限 (繼續)		
指令	**IBM MQ 物件權限**	**類別權限 (適用於物件類型)**
Ping 佇列管理程式	請參閱附註 *3*	不適用
重新整理佇列管理程式	(佇列管理程式物件) 變更	不適用
重新整理安全 (適用於 **SecurityType** MQSECTYPE_SSL)	(佇列管理程式物件) 變更	不適用
重設通道	ctrlx	不適用
重設佇列管理程式	(佇列管理程式物件) 變更	不適用
重設佇列統計資料	dsp 和 chg	不適用
解析通道	ctrlx	不適用
設定權限記錄	*(佇列管理程式物件)* chg 請參閱附註 *4*	請參閱附註 *4*
啟動通道	ctrl	不適用
停止通道	ctrl	不適用
停止連線	(佇列管理程式物件) 變更	不適用
啟動接聽器	ctrl	不適用
停止接聽器	ctrl	不適用
啟動服務	ctrl	不適用
停止服務	ctrl	不適用
Esc 鍵	請參閱附註 *2*	請參閱附註 *2*

附註:

1. 如果要取代的物件確實存在，則此指令適用，否則權限檢查是針對「建立」或「複製而不取代」。
2. 必要權限由跳出文字所定義的 MQSC 指令決定，它相當於先前的其中一個指令。
3. 若要處理任何 PCF 指令，使用者 ID 必須對目標系統上的佇列管理程式物件具有 dsp 權限。
4. 除非已使用 -a 參數啟動指令伺服器，否則會授權此 PCF 指令。 依預設，指令伺服器會在啟動「佇列管理程式」時啟動，且沒有 -a 參數。 See the System Administration Guide for further information.
5. 授與佇列管理程式的使用者 ID *chg* 權限，可讓您設定所有群組及使用者的權限記錄。 請勿將此權限授與一般使用者或應用程式。

IBM MQ 也提供一些通道安全結束點，讓您可以提供自己的使用者結束程式來進行安全檢查。 如需詳細資料，請參閱 顯示頻道 手冊。

IBM MQ for z/OS

z/OS

如需 z/OS 上權限檢查的相關資訊，請參閱 作業 1: 識別 z/OS 系統參數 。

使用 MQAI 來簡化 PCF 的使用

MQAI 是 IBM MQ 的管理介面，可在 AIX、 HP-UX、 IBM i、 Linux、Solaris 及 Windows 上使用。

MQAI 透過使用 資料工具袋在佇列管理程式上執行管理作業。 資料工具袋可讓您以比使用 PCF 更容易的方式處理物件的內容 (或參數)。

使用 MQAI 的優點如下:

簡化 PCF 訊息的使用
MQAI 是更容易管理 IBM MQ 的方法。 如果您使用 MQAI，則不需要撰寫自己的 PCF 訊息。 這可避免與複雜資料結構相關聯的問題。

若要在使用 MQI 呼叫撰寫的程式中傳遞參數， PCF 訊息必須包含指令以及字串或整數資料的詳細資料。 若要手動建立此配置，您必須在程式中為每個結構新增數個陳述式，並且必須配置記憶體空間。 這項任務可能長而艱鉅。

使用 MQAI 撰寫的程式會將參數傳遞至適當的資料工具袋，且每一個結構只需要一個陳述式。 使用 MQAI 資料工具袋不需要您處理陣列及配置儲存體，並提供與 PCF 詳細資料的某種程度隔離。

更容易處理錯誤狀況
很難從 PCF 指令取得回覆碼。 MQAI 可讓程式更容易處理錯誤狀況。

建立並移入資料工具袋之後，您可以使用 `mqExecute` 呼叫，將管理指令訊息傳送至佇列管理程式的指令伺服器。 此呼叫會等待任何回應訊息。 `mqExecute` 呼叫會處理與指令伺服器的交換，並在 回應工具袋中傳回回應。

如需 MQAI 的相關資訊，請參閱 第 20 頁的『IBM MQ 管理介面 (MQAI) 簡介』。

相關資訊
IBM MQ 管理介面參照

IBM MQ 管理介面 (MQAI) 簡介

IBM MQ 管理介面 (MQAI) 是 IBM MQ 的程式設計介面。 它在 IBM MQ 佇列管理程式上使用資料工具袋執行管理作業，以比使用「可程式指令格式 (PCF)」更容易的方式處理物件的內容 (或參數)。

MQAI 概念和術語

MQAI 是 IBM MQ 的程式設計介面，使用 C 語言以及 Visual Basic for Windows。 它可以在 z/OS 以外的平台上使用。

它會使用資料工具袋在 IBM MQ 佇列管理程式上執行管理作業。 資料工具袋可讓您以比使用其他管理介面「可程式指令格式 (PCF)」更容易的方式來處理物件的內容 (或參數)。 相較於使用 MQGET 和 MQPUT 呼叫， MQAI 提供更容易操作的 PCF。

如需資料工具袋的相關資訊，請參閱 第 46 頁的『資料工具袋』。 如需 PCF 的相關資訊，請參閱 第 9 頁的『可程式指令格式簡介』

使用 MQAI

您可以使用 MQAI 來執行下列動作:

- 簡化 PCF 訊息的使用。 MQAI 是管理 IBM MQ 的簡單方法; 您不需要撰寫自己的 PCF 訊息，因此可避免與複雜資料結構相關聯的問題。
- 更容易處理錯誤狀況。 很難從 IBM MQ Script (MQSC) 指令取得回覆碼，但 MQAI 可讓程式更容易處理錯誤狀況。
- 在應用程式之間交換資料。 應用程式資料會以 PCF 格式傳送，並由 MQAI 壓縮及解壓縮。 如果您的訊息資料是由整數和字串所組成，您可以使用 MQAI 來利用 PCF 資料的 IBM MQ 內建資料轉換。 這可避免寫入資料轉換結束程式的需要。 如需使用 MQAI 來管理 IBM MQ 以及在應用程式之間交換資料的相關資訊，請參閱 第 19 頁的『使用 MQAI 來簡化 PCF 的使用』。

使用 MQAI 的範例

顯示的清單提供一些示範 MQAI 用法的範例程式。 範例會執行下列作業:

1. 建立本端佇列。 第 21 頁的『用於建立本端佇列的範例 C 程式 (amqsaicq.c)』

2. 使用簡式事件監視器在畫面上顯示事件。 第 25 頁的『使用事件監視器顯示事件的範例 C 程式 (amqsaiem.c)』
3. 列印所有本端佇列及其現行深度的清單。 第 37 頁的『用於查詢佇列及列印資訊的範例 C 程式 (amqsailq.c)』
4. 列印所有通道及其類型的清單。 第 32 頁的『用於查詢通道物件的範例 C 程式 (amqsaicl.c)』

建置 MQAI 應用程式

若要使用 MQAI 建置應用程式，您可以鏈結至 IBM MQ 的相同程式庫。 如需如何建置 IBM MQ 應用程式的相關資訊，請參閱 建置程序化應用程式。

使用 MQAI 來配置 IBM MQ 的提示

MQAI 使用 PCF 訊息將管理指令傳送至指令伺服器，而不是直接處理指令伺服器本身。 在 第 41 頁的『配置 IBM MQ 的提示和要訣』 中可以找到使用 MQAI 來配置 IBM MQ 的提示

IBM MQ 管理介面 (MQAI)

IBM MQ for Windows、 AIX、 IBM i IBM i、 Linux、 HP-UX 及 Solaris 支援 IBM MQ 管理介面 (MQAI)。 MQAI 是 IBM MQ 的程式設計介面，可為您提供 MQI 的替代方案，用於傳送及接收 PCF。

MQAI 使用 資料袋 可讓您比透過 MQAI 直接使用 PCF 更容易處理物件的內容 (或參數)。

MQAI 透過將參數傳遞至資料工具袋，以提供對 PCF 訊息的更簡單的程式設計存取權，因此每一個結構只需要一個陳述式。 此存取權不需要程式設計師處理陣列及配置儲存體，並提供與 PCF 詳細資料的部分隔離。

MQAI 透過將 PCF 訊息傳送至指令伺服器並等待回應來管理 IBM MQ 。

本手冊的第二節說明 MQAI。 如需 MQAI 元件物件模型介面的說明，請參閱 使用 Java™ 文件。

用於建立本端佇列的範例 C 程式 (amqsaicq.c)

範例 C 程式 `amqsaicq.c` 會使用 MQAI 建立本端佇列。

```
/********************************************************************************/
/*                                                                              */
/* 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);

}
```

使用事件監視器顯示事件的範例 C 程式 (amqsaiem.c)

範例 C 程式 amqsaiem.c 使用 MQAI 示範基本事件監視器。

```
/********************************************************************/
/*                                                                  */
/* 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)
     strncpy(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, &itemCount, &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 hdle*/
                         &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;
}
```

用於查詢通道物件的範例 C 程式 (amqsaicl.c)

範例 C 程式 `amqsaicl.c` 使用 MQAI 查詢通道物件。

```
/********************************************************************************/
/*                                                                              */
/* 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 <cmqcfc.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 &gt; 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 &gt; 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, 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);
}
```

用於查詢佇列及列印資訊的範例 C 程式 (amqsailq.c)

範例 C 程式 `amqsailq.c` 會使用 MQAI 查詢本端佇列的現行深度。

```
/******************************************************************************/
/*                                                                            */
/* 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);
}
```

配置 IBM MQ 的提示和要訣

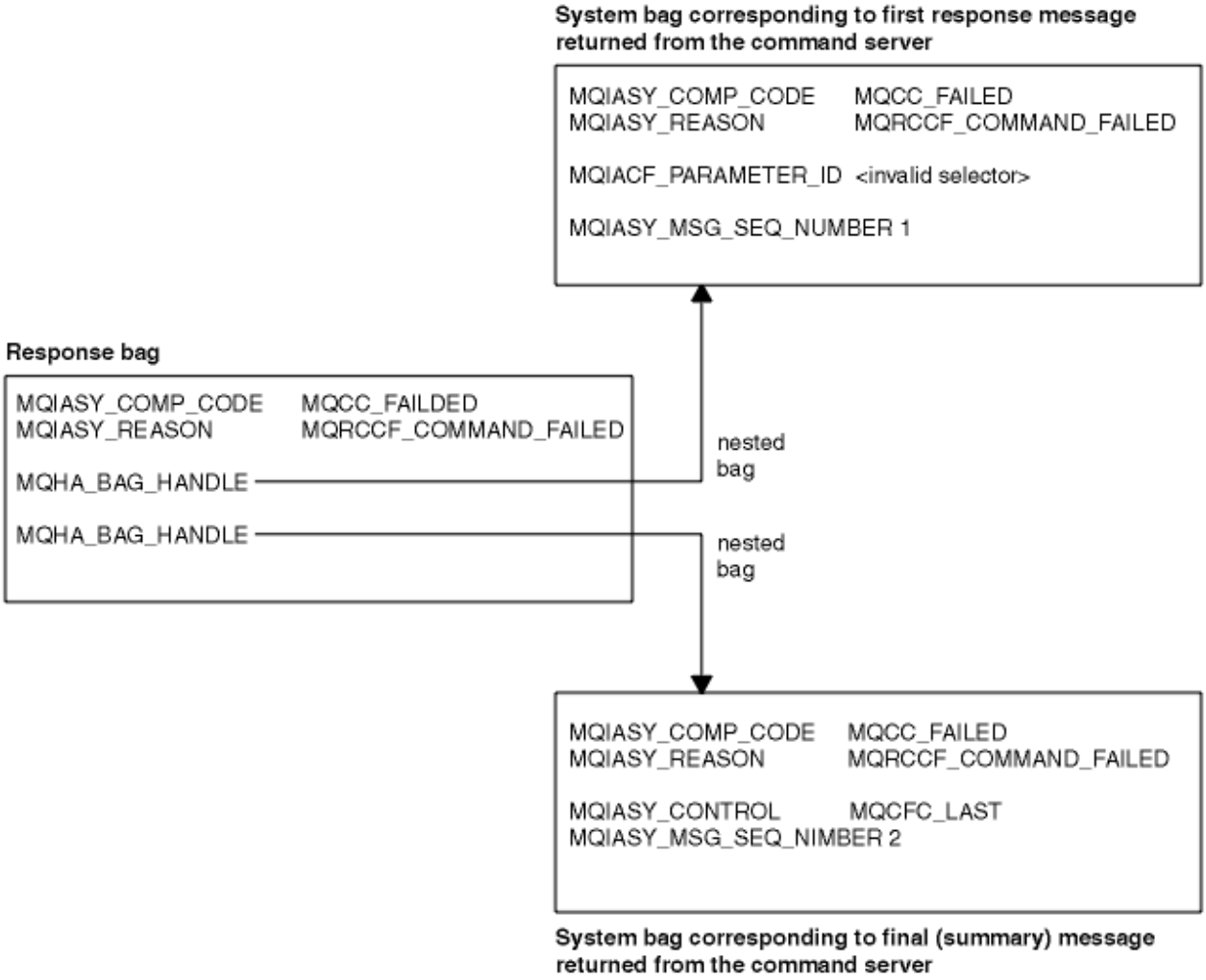
使用 MQAI 時的程式設計提示和要訣。

MQAI 使用 PCF 訊息將管理指令傳送至指令伺服器，而不是直接處理指令伺服器本身。 以下是使用 MQAI 來配置 IBM MQ 的一些提示:

- IBM MQ 中的字串會以空白填補至固定長度。 使用 C，通常可以提供以空值結尾的字串作為 IBM MQ 程式設計介面的輸入參數。
- 如果要清除字串屬性的值，請將它設為單一空白，而不是空字串。
- 請事先考量您要變更的屬性，並只查詢那些屬性。
- 某些屬性無法變更，例如佇列名稱或通道類型。 請確定您嘗試只變更那些可修改的屬性。 請參閱特定 PCF 變更物件的必要及選用參數清單。 請參閱 可程式指令格式的定義。
- 如果 MQAI 呼叫失敗，則會將失敗的部分詳細資料傳回回應工具袋。 然後可以在可由選取元 MQHA_BAG_HANDLE 存取的巢狀工具袋中找到進一步詳細資料。 例如，如果 mqExecute 呼叫失敗，原因碼為 MQRCCF_COMMAND_FAILED，則會在回應工具袋中傳回此資訊。 此原因碼的可能原因是指定的選取元對指令訊息類型無效，且在可由工具袋控點存取的巢狀工具袋中找到此資訊詳細資料。

  如需 MQExecute 的相關資訊，請參閱 第 53 頁的『使用 mqExecute 呼叫將管理指令傳送至指令伺服器』

  下圖顯示此實務範例:



進階主題

檢索、資料轉換及使用訊息描述子的相關資訊

- 編製索引

  在工具袋中取代或移除現有資料項目時使用索引，以保留插入順序。 您可以在 第 42 頁的『在 MQAI 中編製索引』中找到檢索的完整詳細資料。

- 資料轉換

  MQAI 資料工具袋中包含的字串可以是各種編碼字集，並且可以使用 mqSet 整數呼叫來轉換這些字集。 如需資料轉換的完整資料，請參閱 第 43 頁的『MQAI 中的資料轉換』。

- 使用訊息描述子

  MQAI 會產生訊息描述子，當建立資料工具袋時，會將訊息描述子設為起始值。 如需使用訊息描述子的完整詳細資料，請參閱 第 44 頁的『在 MQAI 中使用訊息描述子』。

在 *MQAI* 中編製索引

在工具袋中取代或移除現有資料項目時，會使用索引。 有三種類型的檢索，可讓您輕鬆擷取資料項目。

工具袋中資料項目內的每一個選取器及值都有三個相關聯的索引號碼:

- 相對於具有相同選取元的其他項目的索引。
- 相對於項目所屬選取器種類 (使用者或系統) 的索引。
- 相對於工具袋中所有資料項目的索引 (使用者和系統)。

這容許依使用者選取元及/或系統選取元來編製索引，如 第 43 頁的圖 1 所示。

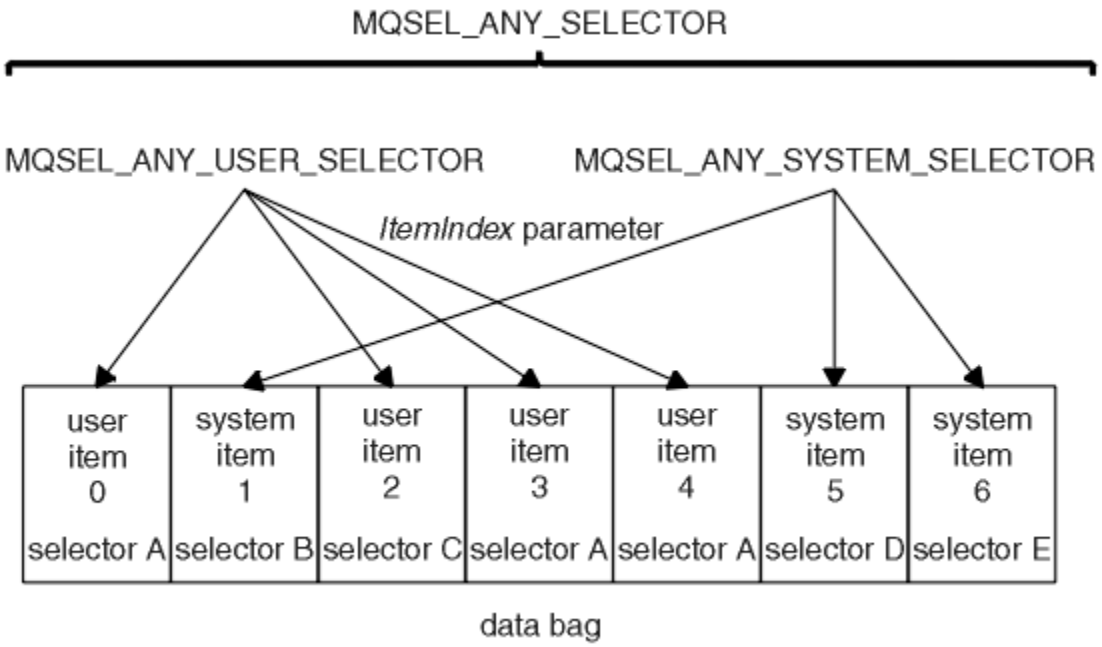


圖 *1:* 編製索引

在圖 第 43 頁的圖 1 中，下列索引配對可以參照使用者項目 3 (選取元 A):

Selector	***ItemIndex***
選取元 A	1
MQSEL_ANY_USER_SELECTOR	2
MQSEL_ANY_SELECTOR	3

索引從零開始，如 C 中的陣列; 如果有 'n' 個出現項目，則索引範圍從零到'n-1'，且沒有間隙。

在工具袋中取代或移除現有資料項目時，會使用索引。 以此方式使用時，會保留插入順序，但可能會影響其他資料項目的索引。 如需此範例，請參閱 變更工具袋內的資訊 及 刪除資料項目。

三種類型的檢索可讓您輕鬆擷取資料項目。 例如，如果工具袋中特定選取器有三個實例，則 mqCountItems 呼叫可以計算該選取器的實例數，而 mqInquire* 呼叫可以同時指定選取器和索引以僅查詢那些值。 這對於具有值清單 (例如通道上的部分結束程式) 的屬性非常有用。

MQAI 中的資料轉換

MQAI 資料工具袋中包含的字串可以是各種編碼字集。 可以使用 mqSet 整數呼叫來轉換這些字串。

如同 PCF 訊息， MQAI 資料工具袋中包含的字串可以是各種編碼字集。 通常， PCF 訊息中的所有字串都使用相同的編碼字集; 亦即，與佇列管理程式相同的字集。

資料工具袋中的每一個字串項目都包含兩個值: 字串本身及 CCSID。 新增至工具袋的字串是從 mqAddString 或 mqSetString 呼叫的 *Buffer* 參數取得。 CCSID 是從包含 MQIASY_CODED_CHAR_SET_ID 選取元的系統項目取得。 這稱為 *bag CCSID* ，可以使用 mqSetInteger 呼叫進行變更。

當您查詢資料工具袋中包含的字串值時， CCSID 是來自呼叫的輸出參數。

第 43 頁的表 2 顯示將資料工具袋轉換成訊息時所套用的規則，反之亦然:

表 *2: CCSID* 處理			
MQAI 呼叫	**CCSID**	**要呼叫的輸入**	**要呼叫的輸出**
mqBagToBuffer	工具袋 CCSID (1)	已忽略	未變更
mqBagToBuffer	袋中的字串 CCSID	已使用	未變更

表 2: *CCSID* 處理 *(*繼續*)*			
MQAI 呼叫	**CCSID**	**要呼叫的輸入**	**要呼叫的輸出**
mqBagToBuffer	緩衝區中的字串 CCSID	不適用	從工具袋中的字串 CCSID 複製
mqBufferToBag	工具袋 CCSID (1)	已忽略	未變更
mqBufferToBag	緩衝區中的字串 CCSID	已使用	未變更
mqBufferToBag	袋中的字串 CCSID	不適用	從緩衝區中的字串 CCSID 複製
mqPut 工具袋	MQMD CCSID	已使用	未變更 (2)
mqPut 工具袋	工具袋 CCSID (1)	已忽略	未變更
mqPut 工具袋	袋中的字串 CCSID	已使用	未變更
mqPut 工具袋	已傳送訊息中的字串 CCSID	不適用	從工具袋中的字串 CCSID 複製
mqGet 工具袋	MQMD CCSID	用於訊息的資料轉換	設為所傳回資料的 CCSID (3)
mqGet 工具袋	工具袋 CCSID (1)	已忽略	未變更
mqGet 工具袋	訊息中的字串 CCSID	已使用	未變更
mqGet 工具袋	袋中的字串 CCSID	不適用	從訊息中的字串 CCSID 複製
mqExecute	要求工具袋 CCSID	用於要求訊息的 MQMD (4)	未變更
mqExecute	回覆工具袋 CCSID	用於回覆訊息的資料轉換 (4)	設為所傳回資料的 CCSID (3)
mqExecute	要求工具袋中的字串 CCSID	用於要求訊息	未變更
mqExecute	回覆工具袋中的字串 CCSID	不適用	從回覆訊息中的字串 CCSID 複製

附註:

1. 工具袋 CCSID 是具有選取元 MQIASY_CODED_CHAR_SET_ID 的系統項目。
2. MQCCSI_Q_MGR 已變更為實際佇列管理程式 CCSID。
3. 如果要求資料轉換，則傳回的資料 CCSID 與輸出值相同。 如果未要求資料轉換，則傳回的資料 CCSID 與訊息值相同。 請注意，如果要求資料轉換但失敗，則不會傳回任何訊息。
4. 如果 CCSID 是 MQCCSI_DEFAULT ，則會使用佇列管理程式的 CCSID。

在 *MQAI* 中使用訊息描述子

建立資料工具袋時， MQAI 產生的訊息描述子會設為起始值。

PCF 指令類型是從具有選取元 MQIASY_TYPE 的系統項目取得。 當您建立資料工具袋時，會根據您建立的工具袋類型來設定此項目的起始值:

表 3: *PCF* 指令類型	
袋的型別	**MQIASY_TYPE 項目的起始值**
MQCBO_ADMIN_BAG	MQCFT_COMMAND

表 3: PCF 指令類型 (繼續)	
袋的型別	**MQIASY_TYPE 項目的起始值**
MQCBO_COMMAND_BAG	MQCFT_COMMAND
MQCBO_ *	MQCFT_USER

當 MQAI 產生訊息描述子時， *Format* 及 *MsgType* 參數中使用的值取決於具有選取元 MQIASY_TYPE 的系統項目值，如 第 44 頁的表 3 所示。

表 4: MQMD 的格式及 MsgType 參數		
PCF 指令類型	**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 45 頁的表 4 顯示如果您建立管理工具袋或指令工具袋，則訊息描述子的 *Format* 是 MQFMT_ADMIN，而 *MsgType* 是 MQMT_REQUEST。 這適用於在預期回應時傳送至指令伺服器的 PCF 要求訊息。

訊息描述子中的其他參數會採用 第 45 頁的表 5 中所示的值。

表 5: 訊息描述子值	
參數	**值**
StrucId	MQMD_STRUC_ID
Version	MQMD_VERSION_1
Report	MQRO_NONE
MsgType	請參閱第 45 頁的表 4。
Expiry	30 秒 (附註 第 46 頁的『1』)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	取決於工具袋 CCSID (附註 第 46 頁的『2』)
Format	請參閱第 45 頁的表 4。
Priority	MQPRI_PRIORITY_AS_Q_DEF
Persistence	MQPER_NOT_PERSISTENT
MsgId	MQMI_NONE
CorrelId	MQCI_NONE
BackoutCount	0
ReplyToQ	請參閱附註 第 46 頁的『3』
ReplyToQMgr	空白

附註:

1. 可以在 mqExecute 呼叫上使用 *OptionsBag* 參數來置換此值。 如需此作業的相關資訊，請參閱 mqExecute。
2. 請參閱 第 43 頁的『MQAI 中的資料轉換』。
3. 使用者指定的回覆佇列或 MQAI 產生的暫時動態佇列的名稱，用於 MQMT_REQUEST 類型的訊息。 否則為空白。

資料工具袋

資料工具袋是使用 MQAI 處理物件內容或參數的方法。

資料袋

- 資料工具袋包含零個以上 資料項目。 這些資料項目會在放入工具袋時在工具袋內訂購。 這稱為 插入順序。 每一個資料項目都包含一個 選取器，用於識別資料項目及該資料項目的 值，該資料項目可以是整數、64 位元整數、整數過濾器、字串、字串過濾器、位元組字串、位元組字串過濾器或另一個工具袋的控點。 在 第 48 頁的『資料項目』 中詳細說明資料項目

  選取元有兩種類型: 使用者選取元 和 系統選取元。 這些在 MQAI 選取器中說明。 選取元通常是唯一的，但相同的選取元可以有多個值。 在此情況下， 索引 會識別所需選取元的特定出現項目。 索引在 第 42 頁的『在 MQAI 中編製索引』 中說明。

  圖 1 顯示這些概念的階層。

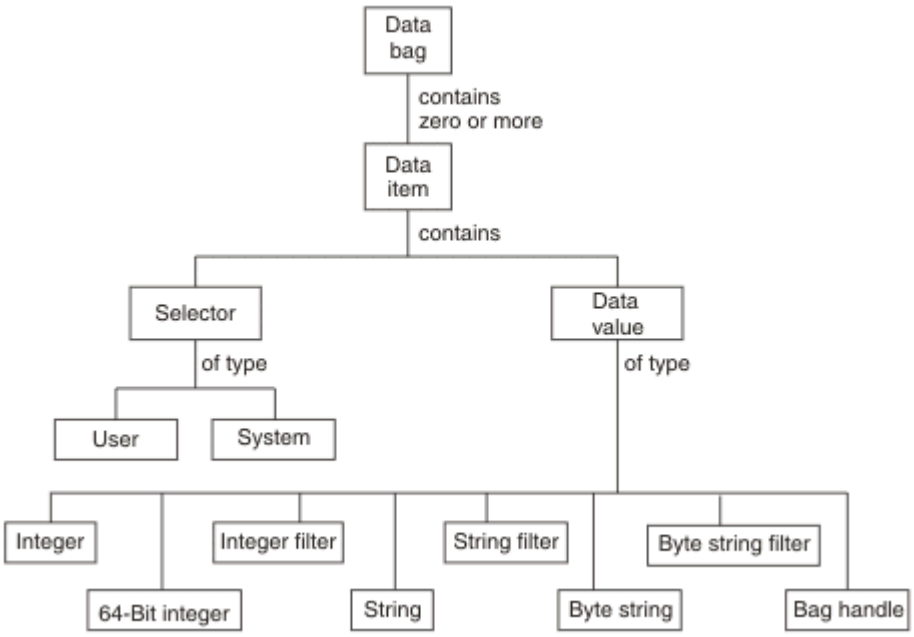


圖 *2: MQAI* 概念的階層

前一段已說明階層。

資料工具袋類型

根據您要執行的作業，您可以選擇要建立的資料工具袋類型:

使用者工具袋 (user bag)
用於使用者資料的簡式工具袋。

管理工具袋 (administration bag)
透過將管理訊息傳送至指令伺服器，為用來管理 IBM MQ 物件的資料所建立的工具袋。 管理工具袋會自動隱含某些選項，如 第 47 頁的『建立及刪除資料工具袋』中所述。

指令工具袋 (command bag)
也會為管理 IBM MQ 物件的指令建立一個工具袋。 不過，與管理工具袋不同，指令工具袋不會自動暗示某些選項，雖然這些選項可用。 如需選項的相關資訊，請參閱 第 47 頁的『建立及刪除資料工具袋』。

群包
用來保留一組分組資料項目的工具袋。 群組工具袋無法用於管理 IBM MQ 物件。

此外，當從指令伺服器傳回回覆訊息並放入使用者的輸出工具袋時， MQAI 會建立 **系統工具袋** 。 使用者無法修改系統工具袋。

使用資料工具袋本主題列出使用資料工具袋的不同方式:

使用資料工具袋

下列清單顯示使用資料袋的不同方式:

- 您可以建立及刪除資料工具袋 第 47 頁的『建立及刪除資料工具袋』。
- 您可以使用資料工具袋 第 48 頁的『放置及接收資料袋』在應用程式之間傳送資料。
- 您可以將資料項目新增至資料工具袋 第 49 頁的『將資料項目新增至工具袋』。
- 您可以在資料工具袋 第 49 頁的『將查詢指令新增至工具袋』內新增查詢指令。
- 您可以在資料工具袋 第 50 頁的『在資料袋內查詢』內查詢。
- 您可以對資料工具袋 第 52 頁的『計數資料項目』內的資料項目進行計數。
- 您可以變更資料工具袋 第 50 頁的『變更工具袋內的資訊』內的資訊。
- 您可以清除資料工具袋 第 51 頁的『使用 mqClear 工具袋呼叫來清除工具袋』。
- 您可以截斷資料工具袋 第 51 頁的『使用 mqTruncateBag 呼叫截斷工具袋』。
- 您可以轉換工具袋及緩衝區 第 52 頁的『轉換袋子和緩衝器』。

建立及刪除資料工具袋

建立資料工具袋

若要使用 MQAI ，請先使用 mqCreateBag 呼叫來建立資料工具袋。 作為此呼叫的輸入，您可以提供一個以上選項來控制工具袋的建立。

MQCreateBag 呼叫的 *Options* 參數可讓您選擇是否建立使用者工具袋、指令工具袋、群組工具袋或管理工具袋。

若要建立使用者工具袋、指令工具袋或群組工具袋，您可以選擇一個以上進一步的選項，以執行下列動作:

- 當工具袋中有兩個以上相同選取器的相鄰出現項目時，請使用清單表單。
- 在將資料項目新增至 PCF 訊息時重新排序資料項目，以確保參數的順序正確。 如需資料項目的相關資訊，請參閱 第 48 頁的『資料項目』。
- 檢查您新增至工具袋之項目的使用者選取元值。

管理工具袋會自動暗示這些選項。

資料工具袋由其控點識別。 工具袋控點從 mqCreate 工具袋傳回，且必須在使用資料工具袋的所有其他呼叫上提供。

如需 mqCreateBag 呼叫的完整說明，請參閱 mqCreateBag。

刪除資料工具袋

使用者建立的任何資料工具袋也必須使用 mqDelete 工具袋呼叫來刪除。 例如，如果在使用者程式碼中建立工具袋，則也必須在使用者程式碼中刪除該工具袋。

MQAI 會自動建立並刪除系統工具袋。 如需此作業的相關資訊，請參閱 第 53 頁的『使用 mqExecute 呼叫將管理指令傳送至指令伺服器』。 使用者代碼無法刪除系統工具袋。

如需 mqDeleteBag 呼叫的完整說明，請參閱 mqDeleteBag。

放置及接收資料袋

透過使用 mqPutBag 和 mqGetBag 呼叫來放置和取得資料工具袋，也可以在應用程式之間傳送資料。 這可讓 MQAI 處理緩衝區，而非應用程式。 mqPutBag 呼叫會將指定工具袋的內容轉換為 PCF 訊息，並將訊息傳送至指定佇列，而 mqGetBag 呼叫會從指定佇列中移除訊息，並將它轉換回資料工具袋。 因此，mqPutBag 呼叫等同於後面接著 MQPUT 的 mqBagToBuffer 呼叫，而 mqGetBag 等同於後面接著 mqBufferToBag 的 MQGET 呼叫。

如需在特定佇列中傳送及接收 PCF 訊息的相關資訊，請參閱 第 11 頁的『在指定佇列中傳送及接收 PCF 訊息』

註: 如果您選擇使用 mqGetBag 呼叫，則訊息內的 PCF 詳細資料必須正確; 如果不正確，則會產生適當的錯誤結果，且不會傳回 PCF 訊息。

資料項目

資料項目用於在建立資料工具袋時移入資料工具袋。 這些資料項目可以是使用者或系統項目。

這些使用者項目包含使用者資料，例如所管理物件的屬性。 應該使用系統項目來進一步控制產生的訊息: 例如，產生訊息標頭。 如需系統項目的相關資訊，請參閱 第 48 頁的『系統項目』。

資料項目類型

建立資料工具袋之後，您可以將整數或字串項目移入其中。 您可以查詢這三種類型的項目。

資料項目可以是整數或字串項目。 以下是 MQAI 內可用的資料項目類型:

- 整數
- 64 位元整數
- 整數過濾器
- 字串
- 字串過濾器
- 位元組字串
- 位元組字串過濾器
- 袋柄

使用資料項目

以下是使用資料項目的方式:

- 第 52 頁的『計數資料項目』.
- 第 52 頁的『刪除資料項目』.
- 第 49 頁的『將資料項目新增至工具袋』.
- 第 49 頁的『過濾及查詢資料項目』.

系統項目

系統項目可用於:

- PCF 標頭的產生。 系統項目可以控制 PCF 指令 ID、控制選項、訊息序號及指令類型。
- 資料轉換。 系統項目會處理工具袋中字串項目的字集 ID。

與所有資料項目一樣，系統項目由選取器和值組成。 如需這些選取器及其適用項目的相關資訊，請參閱 MQAI 選取器。

系統項目是唯一的。 系統選取器可以識別一或多個系統項目。 每一個系統選取器只會出現一次。

大部分系統項目都可以修改 (請參閱 第 50 頁的『變更工具袋內的資訊』)，但使用者無法變更 bag-creation 選項。 您無法刪除系統項目。 (請參閱 第 52 頁的『刪除資料項目』。)

將資料項目新增至工具袋

建立資料工具袋時，您可以將資料項目移入其中。 這些資料項目可以是使用者或系統項目。 如需資料項目的相關資訊，請參閱 第 48 頁的『資料項目』。

MQAI 可讓您將整數項目、64 位元整數項目、整數過濾器項目、字串項目、字串過濾器、位元組字串項目及位元組字串過濾器項目新增至工具袋，這會顯示在 第 49 頁的圖 3 中。 項目由選取器識別。 通常一個選取器只會識別一個項目，但並非一律如此。 如果工具袋中已存在具有指定選取元的資料項目，則該選取元的其他實例會新增至工具袋結尾。

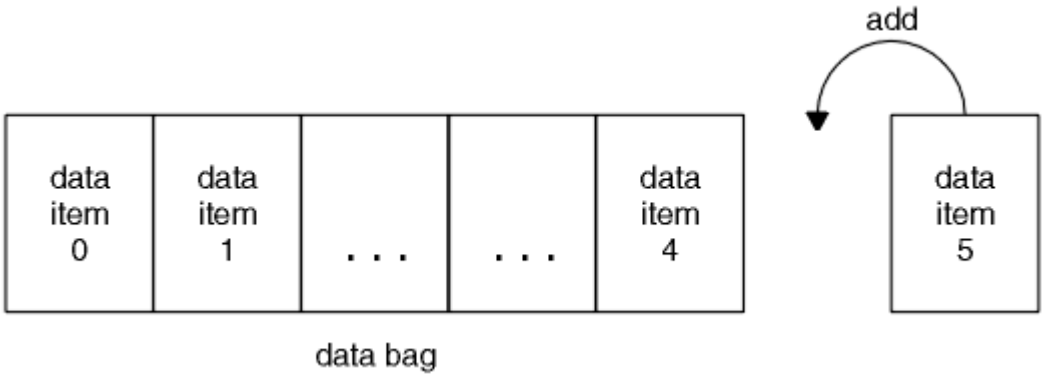


圖 *3:* 新增資料項目

使用 mqAdd* 呼叫將資料項目新增至工具袋:

- 若要新增整數項目，請使用 mqAddInteger 呼叫，如 mqAddInteger 中所述
- 若要新增 64 位元整數項目，請使用 mqAddInteger64 呼叫，如 mqAddInteger64 中所述
- 若要新增整數過濾器項目，請使用 mqAddIntegerFilter 呼叫，如 mqAddIntegerFilter 中所述。
- 若要新增字串項目，請使用 mqAdd 字串呼叫，如 mqAdd 字串 中所述
- 若要新增字串過濾器項目，請使用 mqAddStringFilter 呼叫，如 mqAddStringFilter 中所述。
- 若要新增位元組字串項目，請使用 mqAddByteString 呼叫，如 mqAddByteString 中所述。
- 若要新增位元組字串過濾器項目，請使用 mqAddByteString 過濾器呼叫，如 mqAddByteString 過濾器 中所述。

如需將資料項目新增至工具袋的相關資訊，請參閱 第 48 頁的『系統項目』。

將查詢指令新增至工具袋

mqAdd 查詢呼叫用來將查詢指令新增至工具袋。 通話特別用於管理目的，因此只能與管理袋搭配使用。 它可讓您指定要從 IBM MQ 查詢之屬性的選取元。

如需 mqAdd 查詢呼叫的完整說明，請參閱 mqAdd 查詢。

過濾及查詢資料項目

使用 MQAI 來查詢 IBM MQ 物件的屬性時，您可以使用兩種方式來控制傳回給程式的資料。

- 您可以 **過濾** 使用 mqAddInteger 和 mqAddString 呼叫傳回的資料。 此方法可讓您指定 *Selector* 與 *ItemValue* 配對，例如:

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

此範例指定佇列類型 (*Selector*) 必須是本端 (*ItemValue*)，且此規格必須符合您要查詢之物件 (在此情況下，是佇列) 的屬性。

其他可過濾的屬性對應於可在 第 9 頁的『可程式指令格式簡介』中找到的 PCF Inquire * 指令。 例如，若要查詢通道的屬性，請參閱本產品說明文件中的 Inquire Channel 指令。 Inquire Channel 指令的 "Required parameters" 和 "Optional parameters" 會識別可用於過濾的選取元。

- 您可以使用 mqAdd 查詢呼叫來 **查詢** 物件的特定屬性。 這會指定您感興趣的選取元。 如果您未指定選取元，則會傳回物件的所有屬性。

以下是過濾及查詢佇列屬性的範例:

```
/* 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, ...)
```

如需過濾及查詢資料項目的其他範例，請參閱 第 20 頁的『使用 MQAI 的範例』。

在資料袋內查詢

您可以查詢下列項目:

- 使用 mqInquire 整數呼叫的整數項目值。 請參閱 mqInquire 整數。
- 使用 mqInquireInteger64 呼叫的 64 位元整數項目值。 請參閱 mqInquireInteger64。
- 使用 mqInquireIntegerFilter 呼叫的整數過濾器項目值。 請參閱 mqInquireIntegerFilter。
- 使用 mqInquire 字串呼叫的字串項目值。 請參閱 mqInquire 字串。
- 使用 mqInquireStringFilter 呼叫的字串過濾器項目值。 請參閱 mqInquireStringFilter。
- 使用 mqInquireByteString 呼叫的位元組字串項目值。 請參閱 mqInquireByteString。
- 使用 mqInquireByteString 過濾呼叫的位元組字串過濾器項目值。 請參閱 mqInquireByteString 過濾器。
- 使用 mqInquireBag 呼叫的工具袋控點值。 請參閱 mqInquire 工具袋。

您也可以使用 mqInquireItemInfo 呼叫來查詢特定項目的類型 (整數、64 位元整數、整數過濾器、字串、字串過濾器、位元組字串、位元組字串過濾器或工具袋控點)。 請參閱 mqInquireItemInfo。

變更工具袋內的資訊

MQAI 可讓您使用 mqSet* 呼叫來變更工具袋內的資訊。 您可以:

1. 修改工具袋內的資料項目。 索引容許透過識別要修改之項目的出現項目來取代參數的個別實例 (請參閱 第 50 頁的圖 4)。

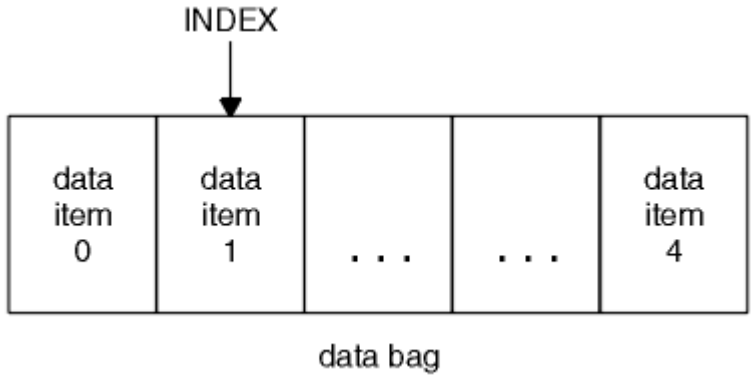


圖 *4:* 修改單一資料項目

2. 刪除所指定選取器的所有現有出現項目，並將新的出現項目新增至工具袋結尾。 (請參閱 第 51 頁的圖 5。) 特殊索引值容許取代參數的 **所有** 實例。

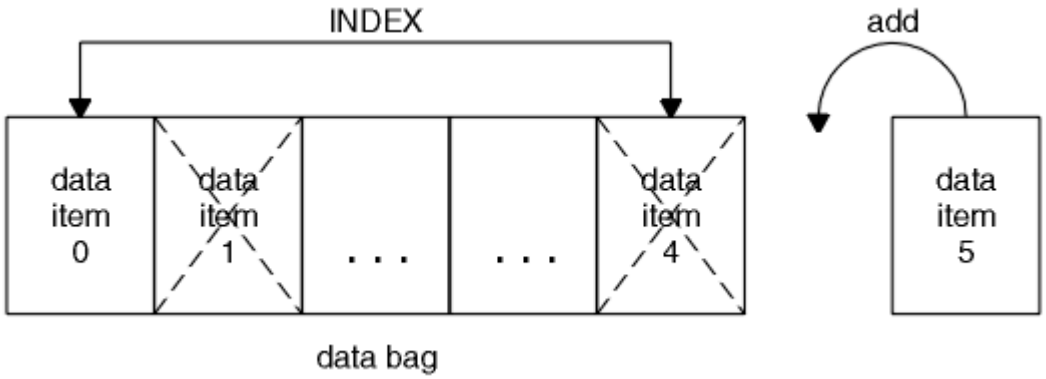


圖 *5:* 修改所有資料項目

註: 索引會保留工具袋內的插入順序，但可能會影響其他資料項目的索引。

mqSet 整數呼叫可讓您修改工具袋內的整數項目。 mqSetInteger64 呼叫可讓您修改 64 位元整數項目。 mqSetIntegerFilter 呼叫可讓您修改整數過濾器項目。 mqSet 字串呼叫可讓您修改字串項目。 mqSetStringFilter 呼叫可讓您修改字串過濾器項目。 mqSetByteString 呼叫可讓您修改位元組字串項目。 mqSetByteString 過濾呼叫可讓您修改位元組字串過濾項目。 或者，您可以使用這些呼叫來刪除所指定選取器的所有現有出現項目，並在工具袋尾端新增出現項目。 資料項目可以是使用者項目或系統項目。

如需這些呼叫的完整說明，請參閱:

- mqSet 整數
- mqSetInteger64
- mqSetIntegerFilter
- mqSet 字串
- mqSetStringFilter
- mqSetByteString
- mqSetByteString 過濾器

使用 *mqClear* 工具袋呼叫來清除工具袋

mqClearBag 呼叫會從使用者工具袋中移除所有使用者項目，並將系統項目重設為其起始值。 也會刪除內含在該袋中的系統袋。

如需 mqClear 工具袋呼叫的完整說明，請參閱 mqClear 工具袋。

使用 *mqTruncateBag* 呼叫截斷工具袋

mqTruncateBag 呼叫會從工具袋尾端刪除項目 (從最近新增的項目開始) ，以減少使用者工具袋中的使用者項目數目。 例如，當使用相同的標頭資訊來產生多個訊息時，可以使用它。

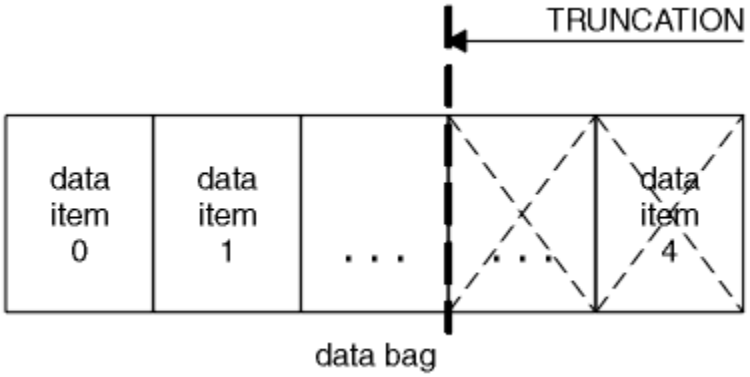


圖 *6:* 截斷工具袋

如需 mqTruncateBag 呼叫的完整說明，請參閱 mqTruncateBag。

轉換袋子和緩衝器

若要在應用程式之間傳送資料，首先會將訊息資料放置在工具袋中。 然後，使用 mqBagToBuffer 呼叫將工具袋中的資料轉換為 PCF 訊息。 使用 MQPUT 呼叫將 PCF 訊息傳送至所需的佇列。 此圖 第 52 頁的圖 7 中顯示。 如需 mqBagToBuffer 呼叫的完整說明，請參閱 mqBagToBuffer。

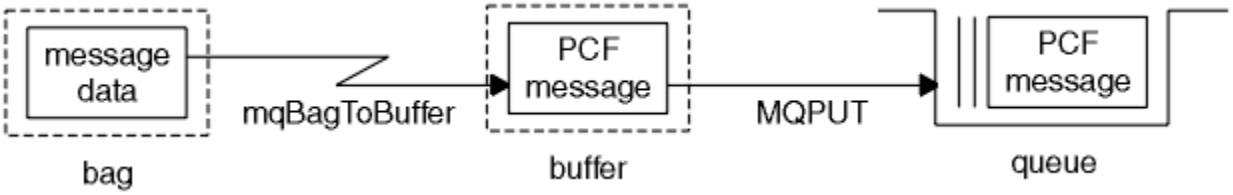


圖 *7:* 將工具袋轉換成 *PCF* 訊息

若要接收資料，會使用 MQGET 呼叫將訊息接收至緩衝區。 然後，如果緩衝區包含有效的 PCF 訊息，則會使用 mqBufferToBag 呼叫將緩衝區中的資料轉換為工具袋。 此圖 第 52 頁的圖 8 中顯示。 如需 mqBufferToBag 呼叫的完整說明，請參閱 mqBufferToBag。

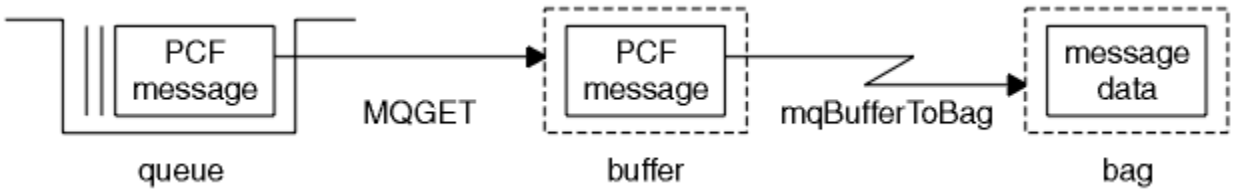


圖 *8:* 將 *PCF* 訊息轉換成工具袋表單

計數資料項目

mqCount 個項目呼叫會計算資料工具袋中儲存的使用者項目及/或系統項目數目，並傳回此數目。 例如，`mqCountItems( Bag, 7, ...)` 會傳回工具袋中具有選取器 7 的項目數。 它可以依個別選取元、依使用者選取元、依系統選取元或依所有選取元來計算項目。

註: 此呼叫會計算資料項目的數目，而不是工具袋中唯一選取器的數目。 選取器可以多次出現，因此工具袋中的唯一選取器可能少於資料項目。

如需 mqCount 個項目呼叫的完整說明，請參閱 mqCount 個項目。

刪除資料項目

您可以使用多種方式從工具袋中刪除項目。 您可以:

- 從工具袋中移除一或多個使用者項目。 如需相關詳細資訊，請參閱第 52 頁的『使用 mqDelete 項目呼叫從工具袋中刪除資料項目』。
- 從工具袋中刪除 **所有** 使用者項目，即 清除 工具袋。 如需詳細資訊，請參閱 第 51 頁的『使用 mqClear 工具袋呼叫來清除工具袋』。
- 從工具袋結尾刪除使用者項目，亦即 截斷 工具袋。 如需相關詳細資訊，請參閱第 51 頁的『使用 mqTruncateBag 呼叫截斷工具袋』。

使用 *mqDelete* 項目呼叫從工具袋中刪除資料項目

mqDelete 項目呼叫會從工具袋中移除一個以上使用者項目。 索引用來刪除下列任一項:

1. 單一出現的指定選取元。 (請參閱 第 53 頁的圖 9。)

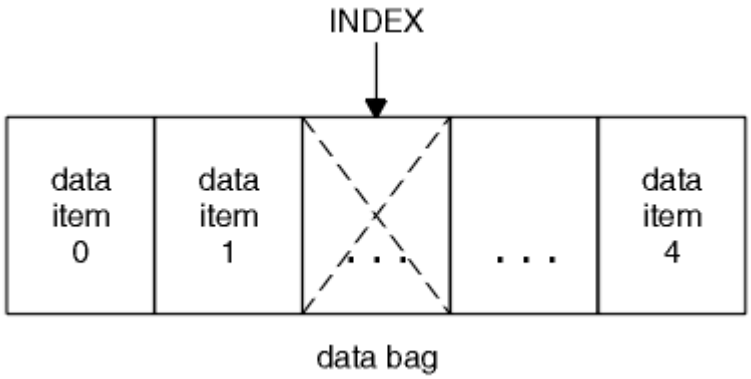


圖 *9:* 刪除單一資料項目

or

2. 指定選取元的所有出現項目。 (請參閱 第 53 頁的圖 10。)

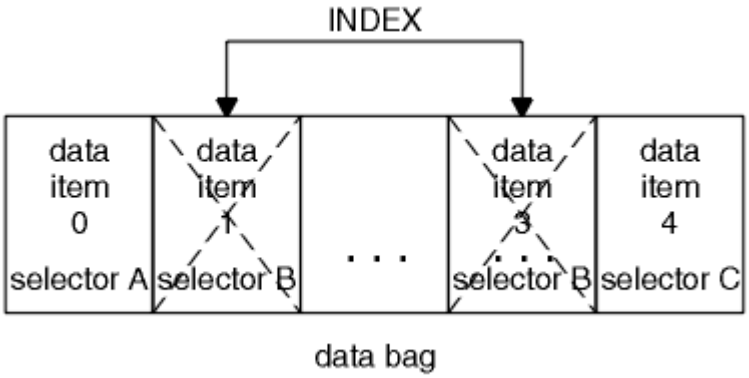


圖 *10:* 刪除所有資料項目

註: 索引會保留工具袋內的插入順序，但可能會影響其他資料項目的索引。 例如， mqDelete 項目呼叫不會保留已刪除項目之後的資料項目的索引值，因為索引會重組以填補已刪除項目中剩餘的間隙。

如需 mqDelete 項目呼叫的完整說明，請參閱 mqDelete 項目。

使用 mqExecute 呼叫將管理指令傳送至指令伺服器

建立並移入資料工具袋之後，可以使用 mqExecute 呼叫將管理指令訊息傳送至佇列管理程式的指令伺服器。 這會處理與指令伺服器的交換，並在工具袋中傳回回應。

建立並移入資料工具袋之後，您可以將管理指令訊息傳送至佇列管理程式的指令伺服器。 最簡單的方法是使用 mqExecute 呼叫來執行此動作。 mqExecute 呼叫會以非持續訊息形式傳送管理指令訊息，並等待任何回應。 回應會在回應工具袋中傳回。 例如，這些可能包含數個 IBM MQ 物件或一系列 PCF 錯誤回應訊息相關屬性的相關資訊。 因此，回應工具袋只能包含回覆碼，或可以包含 巢狀工具袋。

回應訊息會放在系統所建立的系統工具袋中。 例如，若要查詢物件名稱，會建立系統工具袋來保留那些物件名稱，並將工具袋插入使用者工具袋中。 然後這些工具袋的控點會插入回應工具袋中，且選取元 MQHA_BAG_HANDLE 可以存取巢狀工具袋。 系統工具袋會保留在儲存體中 (如果未刪除的話)，直到刪除回應工具袋為止。

巢狀 的概念顯示在 第 54 頁的圖 11 中。

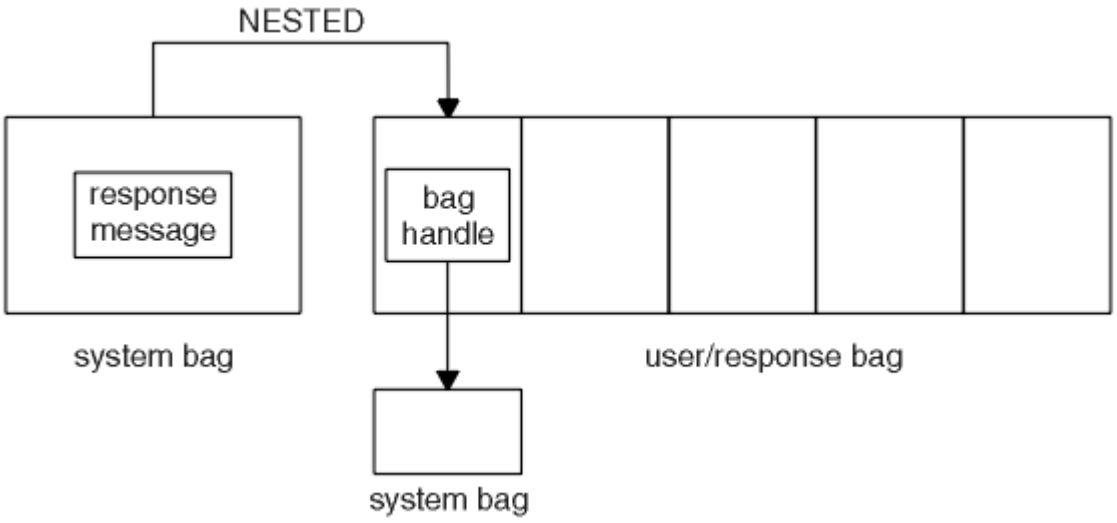


圖 *11:* 巢狀*(N)*

作為 mqExecute 呼叫的輸入，您必須提供:

- MQI 連線控點。
- 要執行的指令。 這應該是其中一個 MQCMD_ * 值。

  註: 如果 MQAI 無法辨識此值，則仍會接受此值。 不過，如果使用 mqAddInquiry 呼叫將值插入工具袋，則此參數必須是 MQAI 可辨識的 INQUIRE 指令。 也就是說，參數的格式應該是 MQCMD_INQUIRE_ *。
- 選擇性地，包含控制呼叫處理的選項之工具袋的控點。 您也可以在這裡指定 MQAI 應該等待每一個回覆訊息的時間上限 (毫秒)。
- 管理工具袋的控點，包含要發出之管理指令的詳細資料。
- 接收回覆訊息的回應工具袋控點。

下列是選用項目:

- 要放置管理指令之佇列的物件控點。

  如果未指定任何物件控點，則會將管理指令放置在 SYSTEM.ADMIN.COMMAND.QUEUE 。 這是預設值。
- 要放置回覆訊息之佇列的物件控點。

  您可以選擇將回覆訊息放置在 MQAI 自動建立的動態佇列上。 所建立的佇列僅在呼叫期間存在，並由 MQAI 從 mqExecute 呼叫結束時刪除。

如需 mqExecute 呼叫的用法範例，請參閱 程式碼範例

使用 MQ Explorer 進行管理

MQ Explorer 可讓您從執行 Windows 或 Linux x86-64 的電腦執行網路的本端或遠端管理。

IBM MQ for Windows 及 IBM MQ for Linux x86-64 提供稱為 MQ Explorer 的管理介面來執行管理作業，作為使用控制或 MQSC 指令的替代方案。 比較指令集 顯示您可以使用 MQ Explorer 執行的動作。

MQ Explorer 可讓您透過將 MQ Explorer 指向您感興趣的佇列管理程式及叢集，從執行 Windows 或 Linux x86-64 的電腦執行網路的本端或遠端管理。 第 55 頁的『遠端佇列管理程式』中說明可以使用 MQ Explorer 來管理的 IBM MQ 平台及層次。

若要配置遠端 IBM MQ 佇列管理程式，以便 MQ Explorer 可以管理它們，請參閱 第 56 頁的『必備軟體及定義』。

它可讓您執行作業，通常與在 Windows 或 Linux x86-64 系統網域內本端或遠端設定及細部調整 IBM MQ 的工作環境相關聯。

在 Linux 上，如果您有多個 Eclipse 安裝架構， MQ Explorer 可能無法啟動。 如果發生這種情況，請使用不同於您用於其他 Eclipse 安裝的使用者 ID 來啟動 MQ Explorer 。

在 Linux 上，若要順利啟動 MQ Explorer，您必須能夠將檔案寫入起始目錄，且起始目錄必須存在。

您可以使用「IBM MQ 檔案總管」執行的動作

這是您可以使用「IBM MQ 探險家」執行的作業清單。

使用「IBM MQ 檔案總管」，您可以執行下列動作:

- 建立及刪除佇列管理程式 (僅在本端機器上)。
- 啟動和停止佇列管理程式 (僅在本端機器上)。
- 定義、顯示及變更 IBM MQ 物件 (例如佇列及通道) 的定義。
- 瀏覽佇列上的訊息。
- 啟動和停止通道。
- 檢視通道、接聽器、佇列或服務物件的狀態資訊。
- 檢視叢集中的佇列管理程式。
- 請檢查以查看哪些應用程式、使用者或通道已開啟特定佇列。
- 使用「建立新的叢集」精靈來建立新的佇列管理程式叢集。
- 使用「將佇列管理程式新增至叢集」精靈，將佇列管理程式新增至叢集。
- 管理搭配 Secure Sockets Layer (SSL) 通道安全使用的鑑別資訊物件。
- 建立及刪除通道起始程式、觸發監視器及接聽器。
- 啟動或停止指令伺服器、通道起始程式、觸發監視器及接聽器。
- 將特定服務設為在佇列管理程式啟動時自動啟動。
- 修改佇列管理程式的內容。
- 變更本端預設佇列管理程式。
- 呼叫 **strmqikm** (ikeyMan) GUI 來管理 Secure Sockets Layer (SSL) 憑證，將憑證與佇列管理程式相關聯，以及配置和設定憑證儲存庫 (僅在本端機器上)。
- 從 IBM MQ 物件建立 JMS 物件，並從 JMS 物件建立 IBM MQ 物件。
- 為任何目前支援的類型建立 JMS Connection Factory。
- 修改任何服務的參數，例如接聽器的 TCP 埠號或通道起始程式佇列名稱。
- 啟動或停止服務追蹤。

您可以使用一系列 內容視圖 及 內容對話框來執行管理作業。

「內容」視圖
「內容視圖」是可以顯示下列內容的畫面:

- 與 IBM MQ 本身相關的屬性及管理選項。
- 與一或多個相關物件相關的屬性及管理選項。
- 叢集的屬性及管理選項。

內容對話框
內容對話框是一個畫面，在一系列欄位中顯示與物件相關的屬性，其中有些可以編輯。

您可以使用 *Navigator* 視圖來導覽「IBM MQ 檔案總管」。「Navigator」可讓您選取所需的「內容視圖」。

遠端佇列管理程式

您可以連接的受支援佇列管理程式有兩個異常狀況。

從 Windows 或 Linux (x86 及 x86-64 平台) 系統中，「IBM MQ 探險家」可以連接至所有支援的佇列管理程式，但有下列例外:

- 早於 6.0 版的 IBM MQ for z/OS 佇列管理程式。
- 目前支援的 MQSeries V2 佇列管理程式。

IBM MQ Explorer 會處理不同指令層次與平台之間的功能差異。 不過，如果它遇到無法辨識的屬性，則該屬性將不可見。

如果您想要使用 IBM MQ V5.3 電腦上的「IBM MQ 探險家」，在 Windows 上遠端管理 V6.0 或更新版本佇列管理程式，則必須在 IBM MQ for Windows V5.3 電腦上安裝 Fix Pack 9 (CSD9) 或更新版本。

如果您想要在 iSeries 上使用「IBM MQ 探險家」，在 IBM MQ V6.0 或更新版本電腦上遠端管理 V5.3 佇列管理程式，則必須在 IBM MQ for iSeries V5.3 電腦上安裝 Fix Pack 11 (CSD11) 或更新版本。 此修正套件可更正「IBM MQ 探險家」與 iSeries 佇列管理程式之間的連線問題。

決定是否使用 IBM MQ Explorer

決定是否在安裝時使用 IBM MQ Explorer 時，請考量本主題中列出的資訊。

您需要注意下列要點:

物件名稱
如果您在「IBM MQ 探險家」中使用佇列管理程式及其他物件的小寫名稱，當您使用 MQSC 指令來處理物件時，必須以單引號括住物件名稱，否則 IBM MQ 無法辨識它們。

大型佇列管理程式
「IBM MQ 探險家」最適用於小型佇列管理程式。 如果單一佇列管理程式上有大量物件，當「IBM MQ 探險家」擷取要呈現在視圖中的必要資訊時，您可能會遇到延遲。

叢集
IBM MQ 叢集可能包含數百或數千個佇列管理程式。 「IBM MQ 探險家」會使用樹狀結構來呈現叢集中的佇列管理程式。 叢集的實體大小不會大幅影響「IBM MQ 探險家」的速度，因為在您選取之前，「IBM MQ 探險家」不會連接至叢集中的佇列管理程式。

設定 IBM MQ Explorer

本節概述您在設定 IBM MQ Explorer 時需要採取的步驟。

- 第 56 頁的『必備軟體及定義』
- 第 57 頁的『安全』
- 第 60 頁的『顯示及隱藏佇列管理程式和叢集』
- 第 60 頁的『叢集成員關係』
- 第 61 頁的『資料轉換』

必備軟體及定義

在嘗試使用 MQ Explorer 之前，請確定您滿足下列需求。

「MQ Explorer」只能使用 TCP/IP 通訊協定來連接遠端佇列管理程式。

請檢查:

1. 指令伺服器正在每個遠端管理佇列管理程式上執行。
2. 必須在每個遠端佇列管理程式上執行適當的 TCP/IP 接聽器物件。 此物件可以是 IBM MQ 接聽器，也可以是在 UNIX 和 Linux 系統上的 inetd 常駐程式。
3. 伺服器連線通道，依預設名稱為 SYSTEM.ADMIN.SVRCONN，存在於所有遠端佇列管理程式中。

   您可以使用下列 MQSC 指令來建立通道:

   ```
   DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)
   ```

   此指令會建立基本通道定義。 如果您想要更準確的定義 (例如，設定安全)，則需要其他參數。 如需相關資訊，請參閱 DEFINE CHANNEL。
4. 系統佇列 SYSTEM.MQEXPLORER.REPLY.MODEL 必須存在。

安全

如果您在您必須控制使用者對特定物件的存取權的環境中使用 IBM MQ，則可能需要考量使用「IBM MQ 探險家」的安全層面。

使用 *IBM MQ Explorer* 的授權

任何使用者都可以使用「IBM MQ 探險家」，但需要某些權限才能連接、存取及管理佇列管理程式。

如果要使用「IBM MQ 探險家」來執行本端管理作業，使用者必須具備執行管理作業的必要權限。 如果使用者是 mqm 群組的成員，則使用者有權執行所有本端管理作業。

若要使用「IBM MQ 探險家」連接至遠端佇列管理程式並執行遠端管理作業，執行「IBM MQ 探險家」的使用者必須具有下列權限:

- 對目標佇列管理程式物件的 CONNECT 權限
- 目標佇列管理程式物件的 INQUIRE 權限
- 目標佇列管理程式物件的 DISPLAY 權限
- 佇列 SYSTEM.MQEXPLORER.REPLY.MODEL
- 對佇列 SYSTEM.MQEXPLORER.REPLY.MODEL
- 對佇列 SYSTEM.MQEXPLORER.REPLY.MODEL
- 對佇列 SYSTEM.ADMIN.COMMAND.QUEUE
- 佇列 SYSTEM.ADMIN.COMMAND.QUEUE
- 執行所選取動作的權限

註: INPUT 權限與佇列中使用者的輸入相關 (取得作業)。 OUTPUT 權限與從使用者到佇列 (放置作業) 的輸出相關。

若要連接至「IBM MQ for z/OS」上的遠端佇列管理程式，並使用「IBM MQ 探險家」執行遠端管理作業，必須提供下列項目:

- 系統佇列 SYSTEM.MQEXPLORER.REPLY.MODEL 的 RACF 設定檔
- 佇列的 RACF 設定檔 AMQ.MQEXPLORER.*

此外，執行「IBM MQ 探險家」的使用者必須具備下列權限:

- RACF 對系統佇列的 UPDATE 權限 SYSTEM.MQEXPLORER.REPLY.MODEL
- RACF 對佇列的 UPDATE 權限 AMQ.MQEXPLORER.*
- 對目標佇列管理程式物件的 CONNECT 權限
- 執行所選取動作的權限
- MQCMDS 類別中所有 hlq.DISPLAY.object 設定檔的 READ 權限

如需如何授與 IBM MQ 物件權限的相關資訊，請參閱 在 UNIX 或 Linux 系統及 Windows 上提供 IBM MQ 物件的存取權。

如果使用者嘗試執行未獲授權執行的作業，則目標佇列管理程式會呼叫授權失敗程序，且作業會失敗。

「IBM MQ 檔案總管」中的預設過濾器會顯示所有 IBM MQ 物件。 如果有任何使用者沒有 DISPLAY 權限的 IBM MQ 物件，則會產生授權失敗。 如果正在記錄權限事件，請將顯示的物件範圍限制為使用者具有 DISPLAY 權限的那些物件。

連接遠端佇列管理程式的安全

您必須保護「IBM MQ 探險家」與每一個遠端佇列管理程式之間的通道安全。

「IBM MQ 探險家」會連接至遠端佇列管理程式，作為 MQI 用戶端應用程式。 這表示每一個遠端佇列管理程式都必須具有伺服器連線通道的定義，以及適當的 TCP/IP 接聽器。 如果您未保護伺服器連線通道的安全，惡意應用程式可能會連接至相同的伺服器連線通道，並以無限制權限來取得佇列管理程式物件的存取權。 為了保護伺服器連線通道的安全，請為通道的 MCAUSER 屬性指定非空白值，使用通道鑑別記錄，或使用安全結束程式。

MCAUSER 屬性的預設值是本端使用者 ID。 如果您指定非空白使用者名稱作為伺服器連線通道的 MCAUSER 屬性，則所有使用此通道連接至佇列管理程式的程式都會以指定使用者身分執行，且具有相同層次的權限。 如果您使用通道鑑別記錄，則不會發生此情況。

搭配使用安全結束程式與 *IBM MQ Explorer*

您可以使用「IBM MQ 探險家」來指定預設安全結束程式及佇列管理程式特定安全結束程式。

您可以從「IBM MQ 檔案總管」定義預設安全結束程式，可用於所有新的用戶端連線。 在建立連線時，可以置換此預設結束程式。 您也可以定義單一佇列管理程式或一組佇列管理程式的安全結束程式，這會在建立連線時生效。 您可以使用 IBM MQ Explorer 來指定結束程式。 如需相關資訊，請參閱 IBM MQ 說明中心。

使用 *MQ Explorer* 來使用啟用 *SSL* 的 *MQI* 通道連接至遠端佇列管理程式

「MQ Explorer」會使用 MQI 通道來連接遠端佇列管理程式。 如果您想要使用 SSL 安全保護 MQI 通道，則必須使用用戶端通道定義表來建立通道。

如需如何使用用戶端通道定義表來建立 MQI 通道的相關資訊，請參閱 IBM MQ MQI clients 概觀。

當您使用用戶端通道定義表建立通道時，可以使用「MQ Explorer」，使用啟用 SSL 的 MQI 通道來連接遠端佇列管理程式，如 第 58 頁的『管理遠端佇列管理程式之系統上的作業』 及 第 58 頁的『管理 MQ Explorer 的系統上的作業』 中所述。

管理遠端佇列管理程式之系統上的作業

在管理遠端佇列管理程式的系統上，執行下列作業:

1. 定義通道的伺服器連線及用戶端連線配對，並為兩個通道上伺服器連線的 *SSLCIPH* 屬性指定適當的值。 如需 *SSLCIPH* 屬性的相關資訊，請參閱 使用 SSL 保護通道
2. 將通道定義表 `AMQCLCHL.TAB`(位於佇列管理程式的 `@ipcc` 目錄中) 傳送至管理 MQ Explorer 的系統。
3. 在指定埠上啟動 TCP/IP 接聽器。
4. 將 CA 和個人 SSL 憑證都放在佇列管理程式的 SSL 目錄中:
   - `/var/mqm/qmgrs/+QMNAME+/SSL` 適用於 UNIX 和 Linux 系統
   - `C:\Program Files\IBM\WebSphere MQ\qmgrs\+QMNAME+\SSL` 適用於 Windows 系統

   其中 `+QMNAME+` 是代表佇列管理程式名稱的記號。
5. 建立 CMS 類型為 `key.kdb` 的金鑰資料庫檔。 透過勾選 **strmqikm** 中的選項 (iKeyman)，將密碼隱藏在檔案中 GUI，或搭配使用 `-stash` 選項與 **runmqckm** 指令。
6. 將 CA 憑證新增至前一個步驟中建立的金鑰資料庫。
7. 將佇列管理程式的個人憑證匯入金鑰資料庫。

如需在 Windows 系統上使用 Secure Sockets Layer 的詳細資訊，請參閱 在 UNIX、 Linux 及 Windows 系統上使用 SSL 或 TLS。

管理 MQ Explorer 的系統上的作業

在管理 MQ Explorer 的系統上，執行下列作業:

1. 建立名稱為 `key.jks` 之 JKS 類型的金鑰資料庫檔。 設定此金鑰資料庫檔的密碼。

   MQ Explorer 使用 Java 金鑰儲存庫檔 (JKS) 來確保 SSL 安全，因此為了配置 MQ Explorer 的 SSL 而建立的金鑰儲存庫檔必須符合此項。
2. 將 CA 憑證新增至前一個步驟中建立的金鑰資料庫。
3. 將佇列管理程式的個人憑證匯入金鑰資料庫。
4. 在 Windows 和 Linux 系統上，使用系統功能表、 `MQExplorer` 執行檔或 **strmqcfg** 指令來啟動 MQ Explorer 。
5. 從 MQ Explorer 工具列中，按一下 **視窗-> 喜好設定**，然後展開 **IBM MQ 瀏覽器**，並按一下 **SSL 用戶端憑證儲存庫**。 同時在「授信憑證儲存庫」和「個人憑證儲存庫」中，輸入 第 58 頁的『管理 MQ Explorer 的系統上的作業』 步驟 1 中所建立之 JKS 檔的名稱和密碼，然後按一下 **確定**。

6. 關閉「**喜好設定**」視窗，然後用滑鼠右鍵按一下 **佇列管理程式**。 按一下 **顯示/隱藏佇列管理程式**，然後在「**顯示/隱藏佇列管理程式**」畫面上按一下 **新增**。
7. 鍵入佇列管理程式的名稱，然後選取 **直接連接** 選項。 按「下一步」。
8. 選取 **使用用戶端通道定義表 (CCDT)**，並在管理遠端佇列管理程式的系統上，指定您在 第 58 頁的『管理遠端佇列管理程式之系統上的作業』 的步驟 2 中從遠端佇列管理程式傳送之通道表檔案的位置。
9. 按一下**完成**。 您現在可以從「MQ Explorer」存取遠端佇列管理程式。

透過另一個佇列管理程式連接

「IBM MQ 探險家」可讓您透過「IBM MQ 探險家」已連接的中繼佇列管理程式，來連接至佇列管理程式。

在此情況下，「IBM MQ 探險家」會指定下列指令，將 PCF 指令訊息放入中繼佇列管理程式:

- 物件描述子 (MQOD) 中的 *ObjectQMgr* 名稱 參數，作為目標佇列管理程式的名稱。 如需佇列名稱解析的相關資訊，請參閱 名稱解析。
- 訊息描述子 (MQMD) 中作為本端 userId 的 *UserIdentifier* 參數。

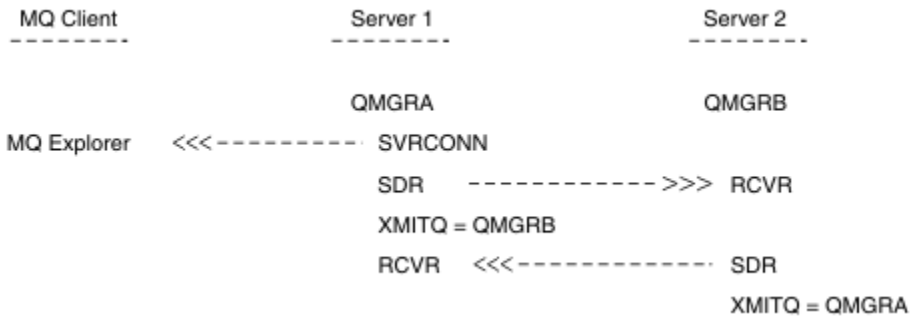
如果接著使用連線透過中繼佇列管理程式連接至目標佇列管理程式，則會再次在訊息描述子 (MQMD) 的 *UserIdentifier* 參數中傳送 userId。 為了讓目標佇列管理程式上的 MCA 接聽器接受此訊息，必須設定 MCAUSER 屬性，或 userId 必須已存在且具有放置權限。

目標佇列管理程式上的指令伺服器會將訊息放入傳輸佇列，並在訊息描述子 (MQMD) 的 *UserIdentifier* 參數中指定 userId。 若要讓此放置成功， userId 必須已存在於具有放置權限的目標佇列管理程式上。

下列範例顯示如何透過中繼佇列管理程式，將佇列管理程式連接至「IBM MQ 探險家」。

建立與佇列管理程式的遠端管理連線。 驗證:

- 伺服器上的佇列管理程式作用中，且已定義伺服器連線通道 (SVRCONN)。
- 接聽器作用中。
- 指令伺服器作用中。
- SYSTEM.MQ EXPLORER.REPLY.MODEL 佇列，且您具有足夠的權限。
- 已啟動佇列管理程式接聽器、指令伺服器及傳送端通道。



在此範例中:

- 「IBM MQ 探險家」會使用用戶端連線連接至佇列管理程式 QMGRA (在 Server1 上執行)。
- Server2 上的佇列管理程式 QMGRB 現在可以透過中繼佇列管理程式 (QMGRA) 連接至「IBM MQ 探險家」
- 使用「IBM MQ 探險家」連接至 QMGRB 時，請選取 QMGRA 作為中繼佇列管理程式

在此狀況下，沒有從「IBM MQ 檔案總管」到 QMGRB 的直接連線; QMGRB 的連線是透過 QMGRA。

Server2 上的佇列管理程式 QMGRB 會使用傳送端-接收端通道連接至 Server1 上的 QMGRA。 QMGRA 與 QMGRB 之間的通道必須以能夠進行遠端管理的方式來設定; 請參閱 第 116 頁的『準備通道及傳輸佇列以進行遠端管理』。

顯示及隱藏佇列管理程式和叢集

「MQ Explorer」一次可以顯示多個佇列管理程式。 從「顯示/隱藏佇列管理程式」畫面 (可從「佇列管理程式」樹狀結構節點的功能表中選取)，您可以選擇是否顯示另一部 (遠端) 機器的相關資訊。 會自動偵測本端佇列管理程式。

如果要顯示遠端佇列管理程式，請執行下列動作:

1. 用滑鼠右鍵按一下 **佇列管理程式** 樹狀結構節點，然後選取 **顯示/隱藏佇列管理程式**。
2. 按一下**新增** 。 這時會顯示「顯示/隱藏佇列管理程式」畫面。
3. 在提供的欄位中輸入遠端佇列管理程式的名稱及主機名稱或 IP 位址。

   主機名稱或 IP 位址是用來使用其預設伺服器連線通道 SYSTEM.ADMIN.SVRCONN 或使用者定義的伺服器連線通道。
4. 按一下**完成**。

「顯示/隱藏佇列管理程式」畫面也會顯示所有可見佇列管理程式的清單。 您可以使用此畫面，從導覽視圖中隱藏佇列管理程式。

如果「MQ Explorer」顯示屬於叢集成員的佇列管理程式，則會偵測並自動顯示叢集。

如果要從這個畫面匯出遠端佇列管理程式清單，請執行下列動作:

1. 關閉「顯示/隱藏佇列管理程式」畫面。
2. 在 MQ Explorer 的「導覽」窗格中，用滑鼠右鍵按一下最高 **IBM MQ** 樹狀結構節點，然後選取 **匯出 MQ Explorer 設定**
3. 按一下 **MQ Explorer > MQ Explorer 設定**
4. 選取 **連線資訊> 遠端佇列管理程式**。
5. 選取檔案以儲存匯出的設定。
6. 最後，按一下 **完成**，將遠端佇列管理程式連線資訊匯出至指定的檔案。

如果要匯入遠端佇列管理程式的清單，請執行下列動作:

1. 在 MQ Explorer 的「導覽」窗格中，用滑鼠右鍵按一下最高 **IBM MQ** 樹狀結構節點，然後選取 **匯入 MQ Explorer 設定** 。
2. 按一下 **MQ Explorer > MQ Explorer 設定**
3. 按一下 **瀏覽**，並導覽至包含遠端佇列管理程式連線資訊的檔案路徑。
4. 按一下**開啟**。 如果檔案包含遠端佇列管理程式清單，則會選取 **連線資訊> 遠端佇列管理程式** 方框。
5. 最後，按一下 **完成**，將遠端佇列管理程式連線資訊匯入「IBM MQ 探險家」。

叢集成員關係

「IBM MQ 探險家」需要屬於叢集成員之佇列管理程式的相關資訊。

如果佇列管理程式是叢集的成員，則會自動移入叢集樹狀結構節點。

如果在「IBM MQ 探險家」執行時，佇列管理程式會成為叢集的成員，則您必須使用叢集的最新管理資料來維護「IBM MQ 探險家」，以便它可以有效地與叢集通訊，並在要求時顯示正確的叢集資訊。 為了執行此動作，「IBM MQ 探險家」需要下列資訊:

- 儲存庫佇列管理程式的名稱
- 儲存庫佇列管理程式的連線名稱 (如果它位於遠端佇列管理程式上)

利用這項資訊，「IBM MQ 檔案總管」可以:

- 使用儲存庫佇列管理程式來取得叢集中的佇列管理程式清單。
- 管理屬於叢集成員且位於受支援平台及指令層次上的佇列管理程式。

在下列情況下，無法進行管理:

- 選擇的儲存庫變成無法使用。 IBM MQ Explorer 不會自動切換至替代儲存庫。

- 無法透過 TCP/IP 聯絡選擇的儲存庫。
- 選擇的儲存庫是在「IBM MQ 探險家」不支援的平台及指令層次上執行的佇列管理程式上執行。

可以管理的叢集成員可以是本端成員，或者如果可以使用 TCP/IP 來聯絡它們，則可以是遠端成員。 「IBM MQ 探險家」會直接連接至本身是叢集成員的本端佇列管理程式，而不使用用戶端連線。

資料轉換

IBM MQ Explorer 以 CCSID 1208 (UTF-8) 運作。 這可讓「IBM MQ 探險家」正確地顯示來自遠端佇列管理程式的資料。 不論是直接連接佇列管理程式，還是使用中間佇列管理程式，「IBM MQ 探險家」都需要將所有送入訊息轉換成 CCSID 1208 (UTF-8)。

如果您嘗試在「IBM MQ 探險家」與具有「IBM MQ 探險家」無法辨識之 CCSID 的佇列管理程式之間建立連線，則會發出錯誤訊息。

字碼頁轉換中說明支援的轉換。

延伸 MQ Explorer

IBM MQ for Windows 及 IBM MQ for Linux x86-64 提供稱為 MQ Explorer 的管理介面來執行管理作業，作為使用控制或 MQSC 指令的替代方案。

本資訊僅適用於 IBM MQ for Windows 及 IBM MQ for Linux x86-64 平台。

MQ Explorer 以與 Eclipse 架構及 Eclipse 支援的其他外掛程式應用程式一致的樣式來呈現資訊。

透過延伸 MQ Explorer，系統管理者能夠自訂 MQ Explorer ，以改善其管理 IBM MQ 的方式。

如需相關資訊，請參閱 MQ Explorer 產品說明文件中的 延伸 *MQ Explorer* 。

使用 IBM MQ 工作列應用程式 (僅限 Windows)

「IBM MQ 工作列」應用程式會在伺服器上的 Windows 系統匣中顯示圖示。 此圖示提供 IBM MQ 的現行狀態，以及您可以從中執行一些簡式動作的功能表。

在 Windows 上， IBM MQ 圖示位於伺服器的系統匣中，並以顏色編碼的狀態符號重疊，其可能具有下列其中一種意義:

綠色
正常運作; 目前沒有警示

藍
不確定; IBM MQ 正在啟動或關閉

黃色
警示; 一個以上服務失敗或已失敗

若要顯示功能表，請用滑鼠右鍵按一下 IBM MQ 圖示。 從功能表中，您可以執行下列動作:

- 按一下 **開啟** ，以開啟「IBM MQ 警示監視器」
- 按一下 **結束** 以結束「IBM MQ 工作列」應用程式
- 按一下 **IBM MQ 檔案總管** 以啟動 IBM MQ 檔案總管
- 按一下 **停止 IBM MQ** 以停止 IBM MQ
- 按一下 **關於 IBM MQ** ，以顯示「IBM MQ 警示監視器」的相關資訊

IBM MQ 警示監視器應用程式 (僅限 Windows)

IBM MQ 警示監視器是一個錯誤偵測工具，可識別並記錄本端機器上 IBM MQ 的問題。

警示監視器會顯示 IBM MQ 伺服器本端安裝架構的現行狀態相關資訊。 它也會監視「Windows 進階配置及電源介面 (ACPI)」，並確保施行 ACPI 設定。

從 IBM MQ 警示監視器中，您可以:

- 直接存取 IBM MQ Explorer

- 檢視所有未解決警示的相關資訊
- 關閉本端機器上的 IBM MQ 服務
- 透過網路將警示訊息遞送至可配置的使用者帳戶，或遞送至 Windows 工作站或伺服器

管理本端 IBM MQ 物件

本節告訴您如何管理本端 IBM MQ 物件，以支援使用「訊息佇列介面 (MQI)」的應用程式。 在此環境定義中，本端管理是指建立、顯示、變更、複製及刪除 IBM MQ 物件。

除了本節中詳述的方法之外，您還可以使用 IBM MQ Explorer 來管理本端 IBM MQ 物件; 請參閱 第 54 頁的『使用 MQ Explorer 進行管理』。

本節包含下列資訊:

- 使用 MQI 的應用程式
- 第 66 頁的『使用 MQSC 指令執行本端管理作業』
- 第 74 頁的『使用佇列管理程式』
- 第 76 頁的『使用本端佇列』
- 第 80 頁的『使用別名佇列』
- 第 97 頁的『使用模型佇列』
- 第 104 頁的『使用服務』
- 第 109 頁的『管理用於觸發的物件』

啟動及停止佇列管理程式

請使用本主題作為停止及啟動佇列管理程式的簡介。

啟動佇列管理程式

若要啟動佇列管理程式，請使用 `strmqm` 指令，如下所示:

```
strmqm saturn.queue.manager
```

在 IBM MQ for Windows 及 IBM MQ for Linux (x86 及 x86-64 平台) 系統上，您可以啟動佇列管理程式，如下所示:

1. 開啟 IBM MQ Explorer。
2. 從「 Navigator 」視圖中選取佇列管理程式。
3. 按一下 `Start` 。 即會啟動佇列管理程式。

如果佇列管理程式啟動花費數秒以上的時間， IBM MQ 會間歇性地發出參考訊息，詳細說明啟動進度。

在佇列管理程式已啟動並準備好接受連線要求之前， `strmqm` 指令不會傳回控制權。

自動啟動佇列管理程式

在「 IBM MQ for Windows 」中，當系統使用「 IBM MQ 探險家」啟動時，您可以自動啟動佇列管理程式。如需相關資訊，請參閱第 54 頁的『使用 MQ Explorer 進行管理』。

停止佇列管理程式

使用 **endmqm** 指令來停止佇列管理程式。

註: 您必須從與您使用之佇列管理程式相關聯的安裝中使用 **endmqm** 指令。 您可以使用 `dspmq -o installation` 指令找出與佇列管理程式相關聯的安裝。

例如，若要停止稱為 QMB 的佇列管理程式，請輸入下列指令:

```
endmqm QMB
```

在 IBM MQ for Windows 及 IBM MQ for Linux (x86 及 x86-64 平台) 系統上，您可以停止佇列管理程式，如下所示:

1. 開啟 IBM MQ Explorer。
2. 從「 Navigator 」視圖中選取佇列管理程式。
3. 按一下 Stop... 。 即會顯示「結束佇列管理程式」畫面。
4. 選取受控制或立即。
5. 按一下 OK 。 佇列管理程式停止。

靜止關機 (quiesced shutdown)

依預設， **endmqm** 指令會對指定的佇列管理程式執行靜止關閉。 這可能需要一些時間才能完成。 靜止關閉會等到所有已連接的應用程式都已斷線為止。

使用這種類型的關機來通知應用程式停止。 如果您發出:

```
endmqm -c QMB
```

當所有應用程式都停止時，系統不會告訴您。（ endmqm -c QMB 指令相當於 endmqm QMB 指令。）

不過，如果您發出:

```
endmqm -w QMB
```

指令會等到所有應用程式都已停止且佇列管理程式已結束為止。

立即關閉 (immediate shutdown)

若為立即關閉，任何現行 MQI 呼叫都可以完成，但任何新呼叫都會失敗。 這種類型的關閉不會等待應用程式與佇列管理程式中斷連線。

若為立即關閉，請鍵入:

```
endmqm -i QMB
```

強制關機 (preemptive shutdown)

註: 除非使用 **endmqm** 指令停止佇列管理程式的所有其他嘗試都失敗，否則請勿使用此方法。 此方法可能會對連接的應用程式產生無法預期的結果。

如果立即關閉無法運作，您必須透過指定 -p 旗標來強制 強制 關閉。 例如:

```
endmqm -p QMB
```

這會立即停止佇列管理程式。 如果此方法仍然無法運作，請參閱 第 64 頁的『手動停止佇列管理程式』 以取得替代解決方案。

如需 **endmqm** 指令及其選項的詳細說明，請參閱 endmqm。

如果您在關閉佇列管理程式時發生問題

關閉佇列管理程式的問題通常是由應用程式所造成。 例如，當應用程式:

- 不要適當地檢查 MQI 回覆碼

- 不要求靜止通知
- 終止而不中斷與佇列管理程式的連線 (透過發出 MQDISC 呼叫)

如果在停止佇列管理程式時發生問題，您可以使用 Ctrl-C 來中斷 **endmqm** 指令。 然後，您可以發出另一個 **endmqm** 指令，但這次帶有指定所需關機類型的旗標。

手動停止佇列管理程式

如果停止佇列管理程式的標準方法失敗，請嘗試這裡說明的方法。

停止佇列管理程式的標準方式是使用 endmqm 指令。 若要手動停止佇列管理程式，請使用本節中說明的其中一個程序。 如需如何使用控制指令對佇列管理程式執行作業的詳細資料，請參閱 在分散式平台上建立及管理佇列管理程式。

在 *IBM MQ for Windows* 中停止佇列管理程式

如何結束處理程序及 IBM MQ 服務，以停止 IBM MQ for Windows 中的佇列管理程式。

如果要停止在 IBM MQ for Windows 下執行的佇列管理程式，請執行下列動作:

1. 使用「Windows 作業管理程式」來列出執行中處理程序的名稱 (ID)。
2. 使用「Windows 作業管理程式」或 **taskkill** 指令，以下列順序 (如果正在執行中) 結束處理程序:

AMQZMUC0	重要程序管理程式
AMQZXMA0	執行控制器
AMQZFUMA	OAM 處理程序
AMQZLAA0	LQM 代理程式
AMQZLSA0	LQM 代理程式
AMQZMUF0	公用程式管理程式
AMQZMGR0	程序控制器
AMQZMUR0	可重新啟動的程序管理程式
AMQFQPUB	發佈訂閱程序
AMQFCXBA	分配管理系統工作者處理程序
AMQRMPPA	程序儲存區處理程序
AMQCRSTA	非執行緒回應者工作處理程序
AMQCRS6B	LU62 接收端通道及用戶端連線
AMQRRMFA	儲存庫處理程序 (適用於叢集)
AMQPCSEA	指令伺服器
RUNMQTRM	呼叫伺服器的觸發監視器
RUNMQDLQ	呼叫無法傳送郵件的佇列處理程式
RUNMQCHI	通道起始程式處理程序
RUNMQLSR	通道接聽器處理程序
AMQXSSVN	共用記憶體伺服器

3. 從 Windows 控制台上的 **管理工具** > **服務** 停止 IBM MQ 服務。
4. 如果您已嘗試所有方法，且佇列管理程式未停止，請將系統重新開機。

「Windows 作業管理程式」及 **tasklist** 指令提供有限的作業相關資訊。 如需協助判斷哪些處理程序與特定佇列管理程式相關的相關資訊，請考慮使用 *Process Explorer* (procexp.exe) 之類的工具，可從 Microsoft 網站 (https://www.microsoft.com) 下載。

在 *IBM MQ for UNIX* 及 *Linux* 系統中停止佇列管理程式

如何結束處理程序及 IBM MQ 服務，以停止 IBM MQ for UNIX 及 Linux 中的佇列管理程式。 如果停止及移除佇列管理程式的標準方法失敗，您可以嘗試這裡說明的方法。

若要停止在 IBM MQ for UNIX 及 Linux 系統下執行的佇列管理程式，請執行下列動作:

1. 使用 ps 指令，尋找仍在執行中之佇列管理程式的處理程序 ID。 例如，如果佇列管理程式稱為 QMNAME，請使用下列指令:

   ```
   ps -ef | grep QMNAME
   ```

2. 結束任何仍在執行中的佇列管理程式處理程序。 使用 kill 指令，並指定使用 ps 指令探索到的處理程序 ID。

   若要結束處理程序，請使用 **kill -KILL <pid>** 或對等的 **kill -9 <pid>** 指令。

   您必須逐一完成您要結束的 PID ， 每次都會發出該指令。

   重要: 如果您使用 **9(SIGKILL)** 以外的任何信號，則處理程序可能不會停止，且您會得到無法預期的結果。

   按下列順序結束處理程序:

amqzmuc0	重要程序管理程式
amqzxma0	執行控制器
阿姆克茲富馬	OAM 處理程序
amqzlaa0	LQM 代理程式
amqzlsa0	LQM 代理程式
amqzmuf0	公用程式管理程式
amqzmur0	可重新啟動的程序管理程式
amqzmgr0	程序控制器
amqfqpub	發佈訂閱程序
amqfcxba	分配管理系統工作者處理程序
amqrmppa	程序儲存區處理程序
amqcrsta	非執行緒回應者工作處理程序
amqcrs6b	LU62 接收端通道及用戶端連線
amqrrmfa	儲存庫處理程序 (適用於叢集)
amqpcsea	指令伺服器
runmqtrm	呼叫伺服器的觸發監視器
runmqdlq	呼叫無法傳送郵件的佇列處理程式
runmqchi	通道起始程式處理程序
runmqlsr	通道接聽器處理程序

註: 您可以使用 **kill -9** 指令來結束無法停止的處理程序。

如果您手動停止佇列管理程式，則可能會採用 FFST ， 並將 FDC 檔案放置在 /var/mqm/errors. 中，請勿將此視為佇列管理程式中的問題報告。

即使在您使用此方法停止佇列管理程式之後，佇列管理程式仍會正常重新啟動。

停止 MQI 通道

當您對伺服器連線通道發出 STOP CHANNEL 指令時，您可以選擇要使用什麼方法來停止用戶端連線通道。這表示可以控制發出 MQGET 等待呼叫的用戶端通道，而且您可以決定停止通道的方式及時間。

可以用三種模式來發出 STOP CHANNEL 指令，指出如何停止通道:

靜止
在處理任何現行訊息之後停止通道。

如果啟用共用交談，則 IBM MQ MQI client 會及時察覺停止要求; 此時間取決於網路速度。 由於對 IBM MQ 發出後續呼叫，用戶端應用程式會察覺停止要求。

強制
立即停止通道。

終止
立即停止通道。 如果通道作為處理程序執行，則可以終止通道的處理程序，或者如果通道作為執行緒執行，則為其執行緒。

這是一個多階段的程序。 如果使用模式終止，則會嘗試停止伺服器連線通道，首先使用模式靜止，然後使用模式強制，必要的話，使用模式終止。 在不同的終止階段期間，用戶端可以收到不同的回覆碼。 如果處理程序或執行緒已終止，用戶端會收到通訊錯誤。

傳回應用程式的回覆碼會根據所發出的 MQI 呼叫及所發出的 STOP CHANNEL 指令而有所不同。 用戶端將接收 MQRC_CONNECTION_QUIESCING 或 MQRC_CONNECTION_BROKEN 回覆碼。 如果用戶端偵測到 MQRC_CONNECTION_QUIESCING，則應該嘗試完成現行交易並終止。 這無法與 MQRC_CONNECTION_BROKEN 搭配使用。 如果用戶端未完成交易且終止的速度足夠快，則會在幾秒後取得 CONNECTION_BROKEN。 具有 MODE (FORCE) 或 MODE (TERMINATE) 的 STOP CHANNEL 指令比具有 MODE (QUIESCE) 更有可能導致 CONNECTION_BROKEN。

相關資訊
MQI 通道

使用 MQSC 指令執行本端管理作業

本節介紹 MQSC 指令，並告訴您如何將它們用於某些一般作業。

如果您使用 IBM MQ for Windows 或 IBM MQ for Linux (x86 及 x86-64 平台)，您也可以使用 IBM MQ Explorer 來執行本節中說明的作業。 如需相關資訊，請參閱第 54 頁的『使用 MQ Explorer 進行管理』。

您可以使用 MQSC 指令來管理佇列管理程式物件，包括佇列管理程式本身、佇列、程序定義、通道、用戶端連線通道、接聽器、服務、名稱清單、叢集及鑑別資訊物件。 本節處理佇列管理程式、佇列及程序定義; 如需通道、用戶端連線通道及接聽器物件的概觀，請參閱 物件。 如需用於管理佇列管理程式物件之所有 MQSC 指令的相關資訊，請參閱 第 67 頁的『Script (MQSC) 指令』。

您可以使用 `runmqsc` 指令，向佇列管理程式發出 MQSC 指令。 (如需此指令的詳細資料，請參閱 runmqsc。) 您可以透過互動方式來執行此動作，從鍵盤發出指令，或者您可以重新導向標準輸入裝置 (`stdin`)，以從 ASCII 文字檔執行一系列指令。 在這兩種情況下，指令的格式都相同。 (如需從文字檔執行指令的相關資訊，請參閱 第 70 頁的『從文字檔執行 MQSC 指令』。)

您可以使用三種方式來執行 `runmqsc` 指令，視指令上設定的旗標而定:

- 請驗證指令而不執行它，其中 MQSC 指令已在本端佇列管理程式上驗證，但未執行。
- 在本端佇列管理程式上執行指令，其中 MQSC 指令在本端佇列管理程式上執行。
- 在遠端佇列管理程式上執行指令，其中 MQSC 指令在遠端佇列管理程式上執行。

您也可以執行指令，後面接著問號以顯示語法。

MQSC 指令中指定的物件屬性會以大寫形式 (例如 RQMNAME) 顯示在此區段中，雖然它們不區分大小寫。 MQSC 指令屬性名稱限制為 8 個字元。 MQSC 指令可在其他平台上使用，包括 IBM i 及 z/OS。

從 IBM MQ 8.0，您可以使用 MQPROMPT 環境變數來設定您選擇的提示。 除了純文字之外， MQPROMPT 變數也容許使用 +VARNAME+ notation，以與 IBM MQ 服務物件定義相同的方式來插入環境變數 (請參閱 第 104 頁的『定義服務物件』)。 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
```

MQSC 指令詳述於 MQSC 指令 區段中。

相關資訊

runmqsc (執行 MQSC 指令)

Script (MQSC) 指令

MQSC 指令提供在 IBM MQ 平台上發出人類可讀指令的統一方法。 如需 可程式化指令格式 (PCF) 指令的相關資訊，請參閱 第 9 頁的『可程式指令格式簡介』。

指令的一般格式顯示在 MQSC 指令中。

使用 MQSC 指令時，您應該遵守下列規則：

- 每一個指令都以主要參數 (動詞) 開頭，後面接著次要參數 (名詞)。 然後，如果有物件的名稱或同屬名稱 (在括弧中)，則後面接著該物件的名稱或同屬名稱 (在大部分指令上都有)。 在此之後，參數通常可以任何順序出現; 如果參數具有對應值，則該值必須直接出現在與它相關的參數之後。

  註: z/OS 在 z/OS 上，次要參數不必是秒。
- 關鍵字、括弧及值可以用任意數目的空白及逗點區隔。 語法圖中顯示的逗點一律可以取代為一或多個空白。 每一個參數前面必須至少有一個空白 (在主要參數後面) z/OS 在 z/OS 上除外 。
- 在指令的開頭或結尾，以及參數、標點符號和值之間，可以出現任意數目的空白。 例如，下列指令有效：

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

  一對引號內的空白是有效的。
- 其他逗點可以出現在容許空白的任何位置，且會被視為空白 (當然，除非它們是在以引號括住的字串內)。
- 不容許重複的參數。 也不容許重複具有其 "NO" 版本 (如 REPLACE NOREPLACE) 的參數。
- 包含下列字元以外的空白、小寫字元或特殊字元的字串：
  - 句點 (.)
  - 正斜線 (/)
  - 底線 (_)
  - 百分比符號 (%)

  必須以單引號括住，除非它們是：
  - z/OS 從 IBM MQ for z/OS 作業及控制面板發出
  - 以星號 IBM i 結尾的同屬值 (在 IBM i 上，這些值必須以單引號括住)
  - 單一星號 (例如 TRACE (*)) IBM i (在 IBM i 上，這些必須以單引號括住)
  - 包含冒號的範圍規格 (例如， CLASS (01:03))

  如果字串本身包含單引號，則單引號會以兩個單引號來代表。不包含在引號內的小寫字元會轉換成大寫。
- 在 z/OS 以外的平台上，不包含任何字元 (亦即，兩個單引號之間沒有空格) 的字串會解譯為以單引號括住的空格，亦即，以與 ('') 相同的方式解譯。 如果所使用的屬性是下列其中一項，則例外：
  - TOPICSTR

- SUB
- USERDATA
- SELECTOR

則兩個不含空格的單引號會解譯為零長度字串。

z/OS 在 z/OS 上，如果您想要以單引號括住空格，則必須將它輸入為 ('')。 不含字元 ('') 的字串與輸入 () 相同。

- 在 v7.0 中，那些字串屬性中以 MQCHARV 類型為基礎的任何尾端空白 (例如， SELECTOR、子使用者資料) 都會被視為顯著，表示 'abc' 不等於 'abc'。
- 左括弧後面接著右括弧，中間沒有重要資訊，例如

```
NAME ( )
```

除非特別註明，否則無效。

- 關鍵字不區分大小寫: AltER、alter 及 ALTER 皆可接受。任何未包含在引號內的內容都會變成大寫。
- 部分參數已定義同義字。例如， DEF 一律是 DEFINE 的同義字，因此 DEF QLOCAL 有效。 不過，同義字並不只是最小字串; DEFI 不是 DEFINE 的有效同義字。

註: DELETE 參數沒有同義字。 這是為了避免在使用 DEF (DEFINE 的同義字) 時意外刪除物件。

如需使用 MQSC 指令來管理 IBM MQ 的概觀，請參閱 第 66 頁的『使用 MQSC 指令執行本端管理作業』。

MQSC 指令使用特定特殊字元來具有特定意義。 如需這些特殊字元及其用法的相關資訊，請參閱 具有特殊意義的字元。

若要瞭解如何使用 MQSC 指令來建置 Script ，請參閱 建置指令 Script。

如需 z/OS 直欄中各符號的說明，請參閱在 z/OS 上使用指令。

如需 MQSC 指令的完整清單，請參閱 MQSC 指令。

相關資訊
建置指令 Script

IBM MQ 物件名稱

如何在 MQSC 指令中使用物件名稱。

在範例中，我們對物件使用一些完整名稱。 這是為了協助您識別您正在處理的物件類型。

當您發出 MQSC 指令時，只需要指定佇列的本端名稱。 在我們的範例中，我們使用佇列名稱，例如:

```
ORANGE.LOCAL.QUEUE
```

名稱的 LOCAL.QUEUE 部分說明此佇列是本端佇列。 一般而言，本端佇列的名稱需要 不 。

我們也使用名稱 saturn.queue.manager 作為佇列管理程式名稱。 名稱的 queue.manager 部分說明此物件是佇列管理程式。 一般而言，佇列管理程式名稱 不 需要。

MQSC 指令中區分大小寫

MQSC 指令 (包括其屬性) 可以大寫或小寫撰寫。 除非名稱以單引號括住，否則 MQSC 指令中的物件名稱會變成大寫 (亦即， QUEUE 和佇列沒有區別)。 如果未使用引號，則會以大寫名稱來處理物件。 如需相關資訊，請參閱 具有特殊意義的字元 。

在某些 IBM MQ 環境中， runmqsc 指令呼叫 (與所有 IBM MQ 控制指令相同) 會區分大小寫。 如需相關資訊，請參閱 使用控制指令 。

標準輸入及輸出

標準輸入裝置(也稱為 stdin) 是從中取得系統輸入的裝置。 通常這是鍵盤，但您可以指定輸入來自序列埠或磁碟檔 (舉例來說)。 標準輸出裝置(也稱為 stdout) 是將系統輸出傳送至其中的裝置。 通常這是一個顯示畫面，但您可以將輸出重新導向至序列埠或檔案。

On operating-system commands and IBM MQ control commands, the < operator redirects input. 如果此運算子後接檔名，則會從檔案取得輸入。 同樣地，> 運算子會重新導向輸出; 如果此運算子後接檔名，則輸出會導向至該檔案。

以互動方式使用 MQSC 指令

您可以使用指令視窗或 Shell，以互動方式使用 MQSC 指令。

若要以互動方式使用 MQSC 指令，請開啟指令視窗或 Shell，並輸入:

```
runmqsc
```

在此指令中，尚未指定佇列管理程式名稱，因此預設佇列管理程式會處理 MQSC 指令。 如果您要使用不同的佇列管理程式，請在 **runmqsc** 指令上指定佇列管理程式名稱。 例如，若要在佇列管理程式 jupiter.queue.manager 上執行 MQSC 指令，請使用下列指令:

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

在此之後，此佇列管理程式會處理您鍵入的所有 MQSC 指令，並假設它位於相同節點上且已在執行中。

現在您可以視需要鍵入任何 MQSC 指令。 例如，嘗試此動作:

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

對於具有太多參數而無法在一行中容納的指令，請使用接續字元來指出指令在下列行上繼續執行:

- 減號 (-) 表示指令將從下列字行的開頭繼續執行。
- 加號 (+) 指出指令將從下一行上的第一個非空白字元繼續執行。

指令輸入以非連續字元的非空白行的最終字元終止。 您也可以輸入分號 (;) 來明確終止指令輸入。 (如果您不小心在指令輸入的最後一行結尾輸入接續字元，這特別有用。)

來自 MQSC 指令的意見

當您發出 MQSC 指令時，佇列管理程式會傳回操作員訊息，以確認您的動作或告訴您您所做的錯誤。 例如:

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

此訊息確認已建立佇列。

```
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
```

此訊息指出您已產生語法錯誤。

這些訊息會傳送至標準輸出裝置。 如果您未正確輸入指令，請參閱 MQSC 指令 以取得正確語法。

結束 MQSC 指令的互動式輸入

若要停止使用 MQSC 指令，請輸入 END 指令。

或者，您可以使用作業系統的 EOF 字元。

從文字檔執行 MQSC 指令

以互動方式執行 MQSC 指令適用於快速測試，但如果您有很長的指令，或反覆地使用特定指令序列，請考慮從文字檔重新導向 `stdin` 。

第 69 頁的『標準輸入及輸出』 包含 `stdin` 和 `stdout` 的相關資訊。 若要從文字檔重新導向 `stdin` ，請先使用一般文字編輯器建立包含 MQSC 指令的文字檔。 當您使用 `runmqsc` 指令時，請使用重新導向運算子。 例如，下列指令會執行文字檔 `myprog.in` 中包含的一連串指令:

```
runmqsc < myprog.in
```

同樣地，您也可以將輸出重新導向至檔案。 包含輸入之 MQSC 指令的檔案稱為 MQSC 指令檔。 包含來自佇列管理程式的回覆的輸出檔稱為輸出檔。

如果要在 `runmqsc` 指令上重新導向 `stdin` 和 `stdout` ，請使用下列指令形式:

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

此指令會呼叫 MQSC 指令檔 `myprog.in.` 中包含的 MQSC 指令，因為我們尚未指定佇列管理程式名稱，所以 MQSC 指令會針對預設佇列管理程式執行。 輸出會傳送至文字檔 `myprog.out`。 第 71 頁的圖 12 顯示來自 MQSC 指令檔的擷取 `myprog.in` ， 而 第 72 頁的圖 13 顯示 `myprog.out` 中對應輸出的擷取。

若要在 `runmqsc` 指令上重新導向 `stdin` 及 `stdout` ， 對於非預設值的佇列管理程式 (`saturn.queue.manager`) ， 請使用下列指令形式:

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

MQSC 指令檔

MQSC 指令以人類可讀的格式 (即 ASCII 文字) 撰寫。 第 71 頁的圖 12 是 MQSC 指令檔的摘錄，顯示 MQSC 指令 (DEFINE QLOCAL) 及其屬性。 MQSC 指令 包含每一個 MQSC 指令及其語法的說明。

```
.
.
.
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;
.
.
.
```

圖 *12:* 從 *MQSC* 指令檔擷取

為了在 IBM MQ 環境之間實現可攜性，請將 MQSC 指令檔中的行長度限制為 72 個字元。 加號表示指令在下一行繼續執行。

MQSC 指令報告

runmqsc 指令會傳回一個報告，該報告將傳送至 stdout。 報告包含:

- 將 MQSC 指令識別為報告來源的標頭:

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

  其中 jupiter.queue.manager 是佇列管理程式的名稱。
- 發出的 MQSC 指令選用編號清單。 依預設，輸入的文字會回應至輸出。 在此輸出內，每個指令都以序號作為字首，如 第 72 頁的圖 13 所示。 不過，您可以在 runmqsc 指令上使用 -e 旗標來抑制輸出。
- 發現錯誤的任何指令的語法錯誤訊息。
- 操作員訊息 ，指出執行每一個指令的結果。 例如，成功完成 DEFINE QLOCAL 指令的操作員訊息為:

  ```
  AMQ8006: IBM MQ queue created.
  ```
- 執行 Script 檔時因一般錯誤而產生的其他訊息。
- 報告的簡短統計摘要，指出讀取的指令數、具有語法錯誤的指令數，以及無法處理的指令數。

  註: 佇列管理程式只會嘗試處理那些沒有語法錯誤的指令。

```
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.
:
.
.
```

圖 *13:* 從 *MQSC* 指令報告檔擷取

執行提供的 MQSC 指令檔

IBM MQ 隨附下列 MQSC 指令檔:

amqscos0.tst
範例程式所使用的物件定義。

amqscic0.tst
CICS 交易的佇列定義。

在 IBM MQ for Windows 中，這些檔案位於 *MQ_INSTALLATION_PATH*\tools\mqsc\samples 目錄中。 *MQ_INSTALLATION_PATH* 代表 IBM MQ 安裝所在的高階目錄。

在 UNIX 和 Linux 系統上，這些檔案位於 *MQ_INSTALLATION_PATH*/samp 目錄中。 *MQ_INSTALLATION_PATH* 代表 IBM MQ 安裝所在的高階目錄。

執行它們的指令如下:

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

使用 runmqsc 驗證指令

您可以使用 runmqsc 指令來驗證本端佇列管理程式上的 MQSC 指令，而不實際執行它們。 若要這麼做，請在 runmqsc 指令中設定 -v 旗標，例如:

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

當您針對 MQSC 指令檔呼叫 runmqsc 時，佇列管理程式會驗證每一個指令並傳回報告，而不會實際執行 MQSC 指令。 這可讓您檢查指令檔中指令的語法。 在下列情況下，這尤其重要:

- 從指令檔執行大量指令。
- 多次使用 MQSC 指令檔。

傳回的報告類似於 第 72 頁的圖 13 中顯示的報告。

您無法使用此方法來遠端驗證 MQSC 指令。 例如，如果您嘗試此指令:

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

會忽略 -w 旗標 (用來指出佇列管理程式在遠端)，並在本端驗證模式下執行指令。 30 是 IBM MQ 等待遠端佇列管理程式回覆的秒數。

從批次檔執行 MQSC 指令

如果您有非常長的指令，或重複使用特定的指令序列，請考量從批次檔重新導向 `stdin` 。

若要從批次檔重新導向 `stdin`，請先使用一般文字編輯器建立包含 MQSC 指令的批次檔。 當您使用 `runmqsc` 指令時，請使用重新導向運算子。 下列範例:

1. 建立測試佇列管理程式 TESTQM
2. 建立相符的 CLNTCONN 及接聽器集，以使用 TCP/IP 埠 1600
3. 建立測試佇列 TESTQ
4. 使用 amqsputc 範例程式將訊息放入佇列

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

crtmqm $MYTEMPQM
strmqm $MYTEMPQM
runmqlsr -m $MYTEMPQM -t TCP -p $MYPORT &

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

amqsputc TESTQ $MYTEMPQM << EOF
hello world
EOF

endmqm -i $MYTEMPQM
```

圖 *14:* 從批次檔執行 *MQSC* 指令的範例 *Script*

解決 MQSC 指令的問題

如果您無法執行 MQSC 指令，請使用本主題中的資訊來查看這些一般問題是否適用於您。 當您讀取指令產生的錯誤時，問題並不總是顯而易見。

當您使用 `runmqsc` 指令時，請記住下列各項:

- 使用 **<** 運算子來重新導向來自檔案的輸入。 如果您省略此運算子，佇列管理程式會將檔名解譯為佇列管理程式名稱，並發出下列錯誤訊息:

  ```
  AMQ8118: IBM MQ queue manager does not exist.
  ```

- 如果您將輸出重新導向至檔案，請使用 **>** 重新導向運算子。 依預設，在呼叫 `runmqsc` 時，會將檔案放置在現行工作目錄中。 指定完整檔名，以將輸出傳送至特定檔案及目錄。
- 使用下列指令來顯示所有佇列管理程式，以確認您已建立要執行指令的佇列管理程式:

  ```
  dspmq
  ```

- 佇列管理程式必須在執行中。 如果不是，請啟動它; (請參閱 啟動佇列管理程式)。 如果您嘗試啟動已在執行中的佇列管理程式，則會收到錯誤訊息。

- 如果您尚未定義預設佇列管理程式，請在 runmqsc 指令上指定佇列管理程式名稱，否則會收到下列錯誤:

```
AMQ8146: IBM MQ queue manager not available.
```

- 您無法將 MQSC 指令指定為 runmqsc 指令的參數。 例如，這無效:

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

- 在發出 runmqsc 指令之前，您無法輸入 MQSC 指令。
- 您無法從 runmqsc 執行控制指令。 例如，當您以互動方式執行 MQSC 指令時，無法發出 strmqm 指令來啟動佇列管理程式。 如果這樣做，您會收到類似下列的錯誤訊息:

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

1 : strmqm 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
```

使用佇列管理程式

可用來顯示或變更佇列管理程式屬性的 MQSC 指令範例。

顯示佇列管理程式屬性

若要顯示 **runmqsc** 指令上所指定佇列管理程式的屬性，請使用下列 MQSC 指令:

```
DISPLAY QMGR
```

此指令的一般輸出顯示在 第 75 頁的圖 15 中

```
DISPLAY QMGR
     1 : DISPLAY QMGR
AMQ8408: Display Queue Manager details.
   QMNAME(QM1)                             ACCTCONO(DISABLED)
   ACCTINT(1800)                           ACCTMQI(OFF)
   ACCTQ(OFF)                              ACTIVREC(MSG)
   ACTVCONO (DISABLED)                     ACTVTRC (OFF)
   ALTDATE(2012-05-27)                     ALTTIME(16.14.01)
   AUTHOREV(DISABLED)                      CCSID(850)
   CHAD(DISABLED)                          CHADEV(DISABLED)
   CHADEXIT( )                             CHLEV(DISABLED)
   CLWLDATA( )                             CLWLEXIT( )
   CLWLLEN(100)                            CLWLMRUC(999999999)
   CLWLUSEQ(LOCAL)                         CMDEV(DISABLED)
   CMDLEVEL(800)                           COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)
   CONFIGEV(DISABLED)                      CRDATE(2011-05-27)
   CRTIME(16.14.01)                        DEADQ()
   DEFXMITQ( )                             DESCR( )
   DISTL(YES)                              INHIBTEV(DISABLED)
   IPADDRV(IPV4)                           LOCALEV(DISABLED)
   LOGGEREV(DISABLED)                      MARKINT(5000)
   MAXHANDS(256)                           MAXMSGL(4194304)
   MAXPROPL(NOLIMIT)                       MAXPRTY(9)
   MAXUMSGS(10000)                         MONACLS(QMGR)
   MONCHL(OFF)                             MONQ(OFF)
   PARENT( )                               PERFMEV(DISABLED)
   PLATFORM(WINDOWSNT)                        PSRTYCNT(5)
   PSNPMSG(DISCARD)                        PSNPRES(NORMAL)
   PSSYNCPT(IFPER)                         QMID(QM1_2011-05-27_16.14.01)
   PSMODE(ENABLED)                         REMOTEEV(DISABLED)
   REPOS( )                                REPOSNL( )
   ROUTEREC(MSG)                           SCHINIT(QMGR)
   SCMDSERV(QMGR)                          SSLCRLNL( )
   SSLCRYP( )                              SSLEV(DISABLED)
   SSLFIPS(NO)                             SSLKEYR(C:\Program Files\IBM\WebSphere
MQ\Data\qmgrs\QM1\ssl\key)
   SSLRKEYC(0)                             STATACLS(QMGR)
   STATCHL(OFF)                            STATINT(1800)
   STATMQI(OFF)                            STATQ(OFF)
   STRSTPEV(ENABLED)                       SYNCPT
   TREELIFE(1800)                          TRIGINT(999999999)
```

圖 *15: DISPLAY QMGR* 指令的一般輸出

註: SYNCPT 是唯讀佇列管理程式屬性。

ALL 參數是 DISPLAY QMGR 指令上的預設值。 它會顯示所有佇列管理程式屬性。 特別是，輸出會告訴您預設佇列管理程式名稱、無法傳送郵件的佇列名稱，以及指令佇列名稱。

您可以輸入下列指令來確認這些佇列存在:

```
DISPLAY QUEUE (SYSTEM.*)
```

這會顯示符合詞幹 SYSTEM.*的佇列清單。 括弧是必要項目。

變更佇列管理程式屬性

若要變更 **runmqsc** 指令上所指定佇列管理程式的屬性，請使用 MQSC 指令 ALTER QMGR，並指定您要變更的屬性及值。 例如，使用下列指令來變更 jupiter.queue.manager 的屬性:

```
runmqsc jupiter.queue.manager
ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

ALTER QMGR 指令會變更使用的無法傳送郵件的佇列，並啟用禁止事件。

相關資訊

佇列管理程式的屬性

使用本端佇列

本節包含可用來管理本端、模型及別名佇列的部分 MQSC 指令範例。

如需這些指令的詳細資訊，請參閱 MQSC 指令 。

定義本端佇列

對於應用程式，本端佇列管理程式是應用程式所連接的佇列管理程式。 本端佇列管理程式所管理的佇列據說是該佇列管理程式的本端佇列。

請使用 MQSC 指令 DEFINE QLOCAL 來建立本端佇列。 您也可以使用預設本端佇列定義中定義的預設，或者您可以修改預設本端佇列的佇列性質。

註: 預設本端佇列名稱為 SYSTEM.DEFAULT.LOCAL.QUEUE ，它是在系統安裝上建立的。

例如，後面的 DEFINE QLOCAL 指令定義稱為 ORANGE.LOCAL.QUEUE :

- 它會啟用取得、啟用放置，並以優先順序為基礎來運作。
- 它是 一般 佇列; 它不是起始佇列或傳輸佇列，且不會產生觸發訊息。
- 佇列深度上限為 5000 則訊息; 訊息長度上限為 4194304 個位元組。

```
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);
```

註:

1. 除了說明的值之外，所有顯示的屬性值都是預設值。 我們在這裡展示它們是為了說明。 如果您確定預設值是您想要的或尚未變更，則可以省略它們。 另請參閱 第 76 頁的『顯示預設物件屬性』。
2. USAGE (NORMAL) 指出此佇列不是傳輸佇列。
3. 如果您已在相同佇列管理程式上具有名稱為 ORANGE.LOCAL.QUEUE，此指令失敗。 如果您要改寫佇列的現有定義，請使用 REPLACE 屬性，但另請參閱 第 77 頁的『變更本端佇列屬性』。

顯示預設物件屬性

您可以使用 DISPLAY QUEUE 指令來顯示定義 IBM MQ 物件時從預設物件取得的屬性。

當您定義 IBM MQ 物件時，它會採用您未從預設物件指定的任何屬性。 例如，當您定義本端佇列時，佇列會從預設本端佇列 (稱為 SYSTEM.DEFAULT.LOCAL.QUEUE。 若要確切查看這些屬性，請使用下列指令:

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE)
```

此指令的語法不同於對應 DEFINE 指令的語法。 在 DISPLAY 指令上，您可以只提供佇列名稱，而在 DEFINE 指令上，您必須指定佇列類型，即 QLOCAL、QALIAS、QMODEL 或 QREMOTE。

您可以透過個別指定屬性來選擇性地顯示屬性。 例如:

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
MAXDEPTH +
MAXMSGL +
CURDEPTH;
```

此指令會顯示三個指定的屬性，如下所示:

```
AMQ8409: Display Queue details.
QUEUE(ORANGE.LOCAL.QUEUE)        TYPE(QLOCAL)
```

```
CURDEPTH(0)                MAXDEPTH(5000)
MAXMSGL(4194304)
```

CURDEPTH 是現行佇列深度，即佇列上的訊息數。 這是要顯示的有用屬性，因為透過監視佇列深度，您可以確保佇列不會變滿。

複製本端佇列定義

您可以在 DEFINE 指令上使用 LIKE 屬性來複製佇列定義。

例如：

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
LIKE (ORANGE.LOCAL.QUEUE)
```

此指令會使用與原始佇列 ORANGE.LOCAL.QUEUE，而不是系統預設本端佇列的 QUEUE。 輸入要複製的佇列名稱 **與** 建立佇列時所輸入的名稱完全相同。 如果名稱包含小寫字元，請以單引號括住名稱。

您也可以使用此形式的 DEFINE 指令來複製佇列定義，但會替代對原始屬性所做的一或多項變更。 例如：

```
DEFINE QLOCAL (THIRD.QUEUE) +
LIKE (ORANGE.LOCAL.QUEUE) +
MAXMSGL(1024);
```

此指令會複製佇列 ORANGE.LOCAL.QUEUE 至佇列 THIRD.QUEUE，但指定新佇列上的訊息長度上限為 1024 個位元組，而不是 4194304 個位元組。

註：

1. 當您在 DEFINE 指令上使用 LIKE 屬性時，您只會複製佇列屬性。 您未複製佇列上的訊息。
2. 如果您定義本端佇列，但未指定 LIKE ，則它與 DEFINE LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE)。

變更本端佇列屬性

您可以使用 ALTER QLOCAL 指令或 DEFINE QLOCAL 指令與 REPLACE 屬性，以兩種方式來變更佇列屬性。

在 第 76 頁的『定義本端佇列』中，稱為 ORANGE.LOCAL.QUEUE 。 例如，假設您要將此佇列上的訊息長度上限減少為 10,000 個位元組。

- 使用 ALTER 指令：

  ```
  ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
  ```

  此指令會變更單一屬性，即訊息長度上限的屬性; 所有其他屬性則維持相同。
- 搭配使用 DEFINE 指令與 REPLACE 選項，例如：

  ```
  DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
  ```

  這個指令不僅會變更訊息長度上限，也會變更所有其他屬性，這些屬性會提供其預設值。 現在佇列已啟用放置，而先前已禁止放置。 「啟用放置」是由佇列 SYSTEM.DEFAULT.LOCAL.QUEUE。

  如果您 **減少** 現有佇列上的訊息長度上限，則現有訊息不會受到影響。 不過，任何新訊息都必須符合新準則。

清除本端佇列

您可以使用 CLEAR 指令來清除本端佇列。

從稱為 MAGENTA.QUEUE，請使用下列指令：

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

註: 沒有可讓您改變主意的提示; 一旦您按 Enter 鍵，訊息便會遺失。

在下列情況下，您無法清除佇列:

- 有未確定的訊息已放置在同步點下的佇列上。
- 應用程式目前已開啟佇列。

刪除本端佇列

您可以使用 MQSC 指令 DELETE QLOCAL 來刪除本端佇列。

如果佇列上有未確定的訊息，則無法刪除。 不過，如果佇列有一或多個已確定訊息，且沒有未確定的訊息，則只有在您指定 PURGE 選項時，才能刪除它。 例如:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

指定 NOPURGE 而非 PURGE ，可確保佇列包含任何已確定訊息時不會被刪除。

瀏覽佇列

IBM MQ 提供一個範例佇列瀏覽器，您可以用來查看佇列上訊息的內容。 瀏覽器以來源及執行檔格式提供。

MQ_INSTALLATION_PATH 代表 IBM MQ 安裝所在的高階目錄。

在 IBM MQ for Windows 中，範例佇列瀏覽器的檔名及路徑如下:

來源
MQ_INSTALLATION_PATH\tools\c\samples\

執行檔
MQ_INSTALLATION_PATH\tools\c\samples\bin\amqsbcg.exe

在 IBM MQ for UNIX 和 Linux 中，檔名和路徑如下:

來源
MQ_INSTALLATION_PATH/samp/amqsbcg0.c

執行檔
MQ_INSTALLATION_PATH/samp/bin/amqsbcg

此範例需要兩個輸入參數: 佇列名稱及佇列管理程式名稱。 例如:

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

此指令的一般結果顯示在 第 79 頁的圖 16 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'000000000000000000000000000000000000000000000000'
  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'000000000000000000000000000000000000000000000000'
  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
```

圖 *16:* 來自佇列瀏覽器的一般結果

相關資訊

瀏覽器範例程式

啟用大型佇列

IBM MQ 支援大於 2 GB 的佇列。

在 Windows 系統上，不需要任何其他啟用，即可支援大型檔案。 在 AIX、 HP-UX、 Linux 和 Solaris 系統上，您需要明確啟用大型檔案支援，才能建立大於 2 GB 的佇列檔。 如需如何執行此動作的相關資訊，請參閱作業系統文件。

部分公用程式 (例如 tar) 無法處理大於 2 GB 的檔案。 在啟用大型檔案支援之前，請先檢查作業系統文件，以取得您使用的公用程式限制的相關資訊。

如需規劃佇列所需儲存體數量的相關資訊，請造訪 IBM MQ 網站以取得平台專用效能報告: https://www.ibm.com/support/docview.wss?rs=171&uid=swg27007150&loc=en_US&cs=utf-8&lang=en#1

使用別名佇列

您可以定義別名佇列來間接參照另一個佇列或主題。

V 8.0.0.6

小心: 發佈清單不支援使用指向主題物件的別名佇列。 從 8.0.0 版 Fix Pack 6 開始，如果別名佇列指向發佈清單中的主題物件，則 IBM MQ 會傳回 MQRC_ALIAS_BASE_Q_TYPE_ERROR。

別名佇列所參照的佇列可以是下列任何一項:

- 本端佇列 (請參閱 第 76 頁的『定義本端佇列』)。
- 遠端佇列的本端定義 (請參閱 第 121 頁的『建立遠端佇列的本端定義』)。
- 主題。

別名佇列不是實際佇列，而是在執行時期解析為實際 (或目標) 佇列的定義。 別名佇列定義指定目標佇列。當應用程式對別名佇列發出 MQOPEN 呼叫時，佇列管理程式會將別名解析為目標佇列名稱。

別名佇列無法解析為另一個本端定義的別名佇列。 不過，別名佇列可以解析為本端佇列管理程式所屬之叢集中的其他位置所定義的別名佇列。 如需進一步資訊，請參閱 名稱解析 。

別名佇列適用於:

- 提供不同應用程式對目標佇列的不同存取權層次。
- 容許不同的應用程式以不同的方式使用相同的佇列。 (您可能想要指派不同的預設優先順序或不同的預設持續性值。)
- 簡化維護、移轉及工作量平衡。 (您可能要變更目標佇列名稱，而不需要變更您繼續使用別名的應用程式。)

例如，假設已開發應用程式，將訊息放置在稱為 MY.ALIAS.QUEUE。 它指定此佇列在提出 MQOPEN 要求時的名稱，如果它將訊息放置在此佇列上，則會間接指定此佇列的名稱。 應用程式不知道佇列是別名佇列。對於每一個使用此別名的 MQI 呼叫，佇列管理程式會解析實際佇列名稱，該名稱可以是本端佇列，也可以是在此佇列管理程式中定義的遠端佇列。

透過變更 TARGQ 屬性的值，您可以將 MQI 呼叫重新導向至另一個佇列，可能是在另一個佇列管理程式上。這對於維護、移轉及負載平衡非常有用。

定義別名佇列

下列指令會建立別名佇列:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGET (YELLOW.QUEUE)
```

此指令會重新導向指定 MY.ALIAS.QUEUE。 指令不會建立目標佇列; 如果佇列 YELLOW.QUEUE 在執行時期不存在。

如果您變更別名定義，則可以將 MQI 呼叫重新導向至另一個佇列。 例如:

```
ALTER QALIAS (MY.ALIAS.QUEUE) TARGET (MAGENTA.QUEUE)
```

此指令會將 MQI 呼叫重新導向至另一個佇列 MAGENTA.QUEUE。

您也可以使用別名佇列，讓單一佇列 (目標佇列) 看起來具有不同應用程式的不同屬性。 作法是定義兩個別名，每一個應用程式一個別名。 假設有兩個應用程式:

- 應用程式 ALPHA 可以在 YELLOW.QUEUE，但不容許從中取得訊息。
- 應用程式測試版可以從 YELLOW.QUEUE，但不容許在其上放置訊息。

下列指令定義針對應用程式 ALPHA 啟用及停用的別名:

```
DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
TARGET (YELLOW.QUEUE) +
PUT (ENABLED) +
GET (DISABLED)
```

下列指令定義已停用放置並啟用應用程式測試版的別名:

```
DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
TARGET (YELLOW.QUEUE) +
PUT (DISABLED) +
GET (ENABLED)
```

A 會使用佇列名稱 ALPHAS.ALIAS.QUEUE 在其 MQI 呼叫中 ;BETA 會使用佇列名稱 BETAS.ALIAS.QUEUE。它們都存取相同的佇列，但使用不同的方式。

當您定義佇列別名時，您可以使用 LIKE 和 REPLACE 屬性，就像您將這些屬性與本端佇列搭配使用一樣。

搭配使用其他指令與別名佇列

您可以使用適當的 MQSC 指令來顯示或變更別名佇列屬性，或刪除別名佇列物件。 例如:

使用下列指令來顯示別名佇列的屬性:

```
DISPLAY QUEUE (ALPHAS.ALIAS.QUEUE)
```

使用下列指令來變更別名所解析的基本佇列名稱，其中 `force` 選項會強制變更，即使佇列已開啟也一樣:

```
ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGQ(ORANGE.LOCAL.QUEUE) FORCE
```

請使用下列指令來刪除此佇列別名:

```
DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

如果應用程式目前已開啟別名佇列，則無法刪除該佇列。 如需此指令及其他別名佇列指令的相關資訊，請參閱 MQSC 指令 。

使用無法傳送郵件的佇列

每一個佇列管理程式通常都有一個本端佇列，用來作為無法傳送郵件的佇列，以便儲存無法遞送至正確目的地的訊息，以供稍後擷取。 您可以告知佇列管理程式有關無法傳送郵件的佇列，並指定如何處理在無法傳送郵件的佇列上找到的訊息。 使用無法傳送郵件的佇列可能會影響遞送訊息的順序，因此您可能選擇不使用它們。

若要告知佇列管理程式無法傳送郵件的佇列，請在 `crtmqm` 指令 (例如 `crtmqm -u DEAD.LETTER.QUEUE`) 上指定無法傳送郵件的佇列名稱，或稍後在 ALTER QMGR 指令上使用 DEADQ 屬性來指定一個。 您必須先定義無法傳送郵件的佇列，然後才能使用它。

稱為 SYSTEM.DEAD.LETTER.QUEUE 可與產品一起使用。 當您建立佇列管理程式時，會自動建立此佇列。必要的話，您可以修改此定義，並將它重新命名。

無法傳送郵件的佇列沒有特殊需求，除了:

- 它必須是本端佇列
- 其 MAXMSGL (訊息長度上限) 屬性必須讓佇列能夠容納佇列管理程式必須處理的最大訊息， **加上** 無法傳送郵件的標頭 (MQDLH) 大小。

使用無法傳送郵件的佇列可能會影響遞送訊息的順序，因此您可能選擇不使用它們。 您可以設定 USEDLQ 通道屬性，以判定無法遞送訊息時是否使用無法傳送郵件的佇列。 此屬性可以配置為佇列管理程式的部分功能

使用無法傳送郵件的佇列，而其他功能則不使用。 如需在不同 MQSC 指令上使用 USEDLQ 通道屬性的相關資訊，請參閱 DEFINE CHANNEL、 DISPLAY CHANNEL、 ALTER CHANNEL 及 DISPLAY CLUSQMGR。

IBM MQ 提供無法傳送郵件的佇列處理程式，可讓您指定如何處理或移除在無法傳送郵件的佇列上找到的訊息。 請參閱 第 82 頁的『處理無法傳送郵件的佇列上的訊息』。

相關資訊

無法傳送郵件的佇列

未遞送訊息疑難排解

處理無法傳送郵件的佇列上的訊息

為了處理無法傳送郵件的佇列 (DLQ) 上的訊息， MQ 提供預設 DLQ 處理程式。 處理程式會根據您定義的規則表格中的項目來比對 DLQ 上的訊息。

訊息可以由佇列管理程式、訊息通道代理程式 (MCA) 及應用程式放置在 DLQ 上。 DLQ 上的所有訊息都必須以 無法傳送郵件的標頭 結構 MQDLH 作為字首。 佇列管理程式或訊息通道代理程式放置在 DLQ 上的訊息一律具有此標頭; 將訊息放置在 DLQ 上的應用程式必須提供此標頭。 MQDLH 結構的 原因 欄位包含原因碼，可識別訊息在 DLQ 上的原因。

所有 IBM MQ 環境都需要一個常式，以定期處理 DLQ 上的訊息。 IBM MQ 提供一個預設常式，稱為 無法傳送郵件的佇列處理程式 (DLQ 處理程式) ， 您可以使用 `runmqdlq` 指令來呼叫它。

透過使用者撰寫的 規則表格，將 DLQ 上處理訊息的指示提供給 DLQ 處理程式。 也就是說， DLQ 處理程式會根據規則表格中的項目來比對 DLQ 上的訊息; 當 DLQ 訊息符合規則表格中的項目時， DLQ 處理程式會執行與該項目相關聯的動作。

相關資訊

無法傳送郵件的佇列

未遞送訊息疑難排解

IBM MQ for IBM i 無法傳送郵件的佇列處理程式

使用此資訊來瞭解無法傳送郵件的佇列處理程式，以及如何呼叫。

無法傳送郵件的佇列 (DLQ) (有時稱為 無法遞送的訊息佇列) 是無法遞送至其目的地佇列之訊息的保留佇列。 網路中的每個佇列管理程式都應該有相關聯的 DLQ。

註: 通常最好避免在 DLQ 上放置訊息。 如需使用及避免 DLQ 的相關資訊，請參閱 第 81 頁的『使用無法傳送郵件的佇列』。

佇列管理程式、訊息通道代理程式及應用程式可以將訊息放置在 DLQ 上。 DLQ 上的所有訊息都必須以 無法傳送郵件的標頭 結構 MQDLH 作為字首。 由佇列管理程式或訊息通道代理程式放置在 DLQ 上的訊息一律具有 MQDLH。 一律提供 MQDLH 給在 DLQ 上放置訊息的應用程式。 MQDLH 結構的 原因 欄位包含原因碼，可識別訊息在 DLQ 上的原因。

在所有 IBM MQ 環境中，必須有定期執行的常式，才能處理 DLQ 上的訊息。 IBM MQ 提供一個預設常式，稱為 無法傳送郵件的佇列處理程式 (DLQ 處理程式) ， 您可以使用 STRMQMDLQ 指令來呼叫該常式。 使用者撰寫的 規則表格 提供指示給 DLQ 處理程式，以處理 DLQ 上的訊息。 也就是說， DLQ 處理程式會根據規則表格中的項目來比對 DLQ 上的訊息。 當 DLQ 訊息符合規則表格中的項目時， DLQ 處理程式會執行與該項目相關聯的動作。

呼叫 DLQ 處理程式

請使用 STRMQMDLQ 指令來呼叫 DLQ 處理程式。 您可以用兩種方式來命名您要處理的 DLQ 及您要使用的佇列管理程式:

- 作為指令提示中 STRMQMDLQ 的參數。 例如:

```
STRMQMDLQ UDLMSGQ(ABC1.DEAD.LETTER.QUEUE) SRCMBR(QRULE) SRCFILE(library/QTXTSRC)
MQMNAME(MY.QUEUE.MANAGER)
```

- 在規則表格中。 例如:

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE)
```

註: 規則表是可以採用任何名稱的來源實體檔內的成員。

這些範例適用於稱為 ABC1.DEAD.LETTER.QUEUE 的 DLQ，由預設佇列管理程式所擁有。

如果您未如所示指定 DLQ 或佇列管理程式，則安裝的預設佇列管理程式會與屬於該佇列管理程式的 DLQ 一起使用。

STRMQMDLQ 指令會從規則表格取得其輸入。

您必須獲得授權，才能同時存取 DLQ 本身，以及 DLQ 上的訊息所轉遞至的任何訊息佇列，以便執行 DLQ 處理程式。 您也必須獲得授權來採用其他使用者的身分，才能讓 DLQ 在訊息環境定義中具有使用者 ID 權限的佇列上放置訊息。

相關資訊

無法傳送郵件的佇列

未遞送訊息疑難排解

DLQ 處理程式規則表格

DLQ 處理程式規則表格定義 DLQ 處理程式如何處理到達 DLQ 的訊息

DLQ 處理程式規則表格定義 DLQ 處理程式如何處理到達 DLQ 的訊息。 規則表格中有兩種類型的項目:

- 表格中的第一個項目 (選用) 包含 控制資料。
- 表格中的所有其他項目都是要遵循 DLQ 處理程式的 規則 。 每一個規則都包含符合訊息的 型樣 (一組訊息性質)，以及當 DLQ 上的訊息符合指定型樣時要採取的 動作 。 規則表格中必須至少有一個規則。

規則表格中的每一個項目都包含一個以上關鍵字。

控制資料

本節說明您可以併入 DLQ 處理程式規則表格中控制資料項目的關鍵字。 請注意下列項目:

- 關鍵字的預設值 (如果有的話) 會畫底線。
- 垂直線 (|) 會區隔替代方案。 您只能指定其中一個。
- 所有關鍵字都是選用的。

INPUTQ (*QueueName* | ' ')
您要處理的 DLQ 名稱:

1. 您指定為 **STRMQMDLQ** 指令參數的任何 UDLMSGQ 值 (或 *DFT) 都會置換規則表格中的任何 INPUTQ 值。
2. 如果您指定空白 UDLMSGQ 值作為 **STRMQMDLQ** 指令的參數，則會使用規則表格中的 INPUTQ 值。
3. 如果您指定空白 UDLMSGQ 值作為 **STRMQMDLQ** 指令的參數，並在規則表格中指定空白 INPUTQ 值，則會使用系統預設無法傳送郵件的佇列。

INPUTQM (*QueueManagerName* | ' ')
擁有 INPUTQ 關鍵字上所指名之 DLQ 的佇列管理程式名稱。

如果您未指定佇列管理程式，或在規則表中指定 INPUTQM ('')，則系統會使用預設佇列管理程式進行安裝。

RETRYINT (間隔 | 60)
DLQ 處理程式應嘗試重新處理 DLQ 上第一次嘗試無法處理且已要求重複嘗試的訊息的間隔 (以秒為單位)。 依預設，重試間隔為 60 秒。

WAIT (YES | NO | *nnn*)
當 DLQ 處理程式偵測到沒有可處理的進一步訊息時， DLQ 處理程式是否應該等待進一步訊息到達 DLQ。

YES
導致 DLQ 處理程式無限期等待。

NO
當 DLQ 處理程式偵測到 DLQ 是空的或未包含可處理的訊息時，會導致 DLQ 處理程式終止。

nnn
導致 DLQ 處理程式在偵測到佇列是空的或沒有可處理的訊息之後，等待 *nnn* 秒，讓新工作在終止之前到達。

對忙碌 DLQ 指定 WAIT (YES)，並指定 WAIT (NO) 或 WAIT (*nnn*) 適用於活動層次較低的 DLQ。 如果容許 DLQ 處理程式終止，請使用觸發來重新呼叫它。

您可以提供 DLQ 的名稱作為 **STRMQMDLQ** 指令的輸入參數，作為在規則表格中併入控制資料的替代方案。 如果在規則表格及 **STRMQMDLQ** 指令的輸入中同時指定任何值，則 **STRMQMDLQ** 指令上指定的值優先。

註: 如果控制資料項目包含在規則表格中，則它必須是表格中的第一個項目。

規則 *(*型樣和動作*)*
使用此資訊來瞭解 DLQ 規則。

以下是 DLQ 處理程式規則表格中的規則範例:

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
ACTION (RETRY) RETRY (3)
```

此規則會指示 DLQ 處理程式嘗試 3 次將任何放置在 DLQ 上的持續訊息遞送至其目的地佇列，因為禁止 MQPUT 及 MQPUT1 。

本節說明您可以包含在規則中的關鍵字。 請注意下列項目:

- 關鍵字的預設值 (如果有的話) 會畫底線。 對於大部分關鍵字，預設值為 * (星號)，其符合任何值。
- 垂直線 (|) 會區隔替代方案。 您只能指定其中一個。
- ACTION 以外的所有關鍵字都是選用的。

本節以型樣相符關鍵字 (DLQ 上的訊息符合的那些關鍵字) 的說明開始。 然後，它會說明動作關鍵字 (那些決定 DLQ 處理程式如何處理相符訊息的關鍵字)。

型樣相符關鍵字
下列範例說明型樣相符關鍵字。 使用它們來指定 DLQ 上符合哪些訊息的值。 所有型樣相符關鍵字都是選用的。

APPLIDAT (*ApplIdentity* 資料 | *)
DLQ 上訊息的 *ApplIdentity* 資料 值，在訊息描述子 MQMD 中指定。

APPLNAME (*PutApplName* | *)
發出 MQPUT 或 MQPUT1 呼叫的應用程式名稱，如 DLQ 上訊息的訊息描述子 MQMD 的 *PutAppl* 名稱 欄位中所指定。

APPLTYPE (*PutAppl* 類型 | *)
在 DLQ 上訊息的訊息描述子 MQMD 中指定的 *PutAppl* 類型 值。

DESTQ (*QueueName* | *)
指定訊息的訊息佇列名稱。

DESTQM (*QueueManagerName* | *)
訊息目的地訊息佇列的佇列管理程式名稱。

回饋 (回饋 | *)
當 *MsgType* 值為 MQMT_REPORT 時， 意見 會說明報告的本質。

您可以使用符號名稱。 例如，您可以使用符號名稱 MQFB_COA 來識別 DLQ 上需要確認到達其目的地佇列的那些訊息。

FORMAT (格式 | *)
訊息傳送者用來說明訊息資料格式的名稱。

MSGTYPE (*MsgType* | *)
DLQ 上訊息的訊息類型。

您可以使用符號名稱。 例如，您可以使用符號名稱 MQMT_REQUEST 來識別 DLQ 上需要回覆的那些訊息。

持續保存 (持續性 | *)
訊息的持續性值。 (訊息的持續性會決定它是否在重新啟動佇列管理程式之後仍然存在。)

您可以使用符號名稱。 例如，您可以使用符號名稱 MQPER_PERSISTENT 來識別 DLQ 上持續存在的那些訊息。

REASON (*ReasonCode* | *)
說明訊息放置到 DLQ 的原因碼。

您可以使用符號名稱。 例如，您可以使用符號名稱 MQRC_Q_FULL 來識別放置在 DLQ 上的那些訊息，因為其目的地佇列已滿。

REPLYQ (*QueueName* | *)
在 DLQ 上訊息的訊息描述子 MQMD 中指定的回覆目的地佇列名稱。

REPLYQM (*QueueManagerName* | *)
REPLYQ 關鍵字中指定之回覆目的地佇列的佇列管理程式名稱。

USERID (*UserIdentifier* | *)
在 DLQ 上產生訊息之使用者的使用者 ID，如訊息描述子 MQMD 中所指定。

動作關鍵字
說明動作關鍵字。 請使用它們來決定如何處理相符訊息。

ACTION (DISCARD | IGNORE | RETRY | FWD)
針對 DLQ 上符合此規則中定義之型樣的任何訊息所採取的動作。

捨棄
導致從 DLQ 中刪除訊息。

IGNORE
導致訊息保留在 DLQ 上。

重試
導致 DLQ 處理程式重試將訊息放入其目的地佇列。

轉遞
導致 DLQ 處理程式將訊息轉遞至 FWDQ 關鍵字上指定的佇列。

您必須指定 ACTION 關鍵字。 嘗試實作動作的次數由 RETRY 關鍵字控管。 控制資料的 RETRYINT 關鍵字控制兩次嘗試之間的間隔。

FWDQ (*QueueName* |&DESTQ|&REPLYQ)
當您選取 ACTION 關鍵字時，將訊息轉遞至其中的訊息佇列名稱。

QueueName
訊息佇列的名稱。 FWDQ ('') 無效。

&DESTQ
從 MQDLH 結構中的 *DestQName* 欄位取得佇列名稱。

&REPLYQ
從訊息描述子 MQMD 的 *ReplyToQ* 欄位中取得佇列名稱。

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* 名稱 |&DESTQM|&REPLYQM|' ')
訊息轉遞至其中之佇列的佇列管理程式。

***QueueManager* 名稱**
當您選取 ACTION (FWD) 關鍵字時，訊息轉遞至其中之佇列的佇列管理程式名稱。

&DESTQM
從 MQDLH 結構中的 *DestQMgr* 名稱 欄位取得佇列管理程式名稱。

&REPLYQM
從訊息描述子 MQMD 的 *ReplyToQMgr* 欄位中取得佇列管理程式名稱。

' '
FWDQM ('') 是預設值，用於識別本端佇列管理程式。

標頭 (YES | NO)
MQDLH 是否應該保留在要求 ACTION (FWD) 的訊息上。 依預設， MQDLH 會保留在訊息上。 HEADER 關鍵字對 FWD 以外的動作無效。

PUTAUT (DEF | CTX)
DLQ 處理程式應用來放置訊息的權限:

DEF
放置具有 DLQ 處理程式本身權限的訊息。

CTX
導致在訊息環境定義中放置具有使用者 ID 權限的訊息。 如果您指定 PUTAUT (CTX) ，則必須獲得授權來假設其他使用者的身分。

RETRY (*RetryCount* | 1)
嘗試動作的次數，範圍為 1-999,999,999 (在控制資料的 RETRYINT 關鍵字上指定的間隔)。

註: DLQ 處理程式嘗試實作任何特定規則的次數特定於 DLQ 處理程式的現行實例; 此計數不會在重新啟動之間持續保存。 如果您重新啟動 DLQ 處理程式，則嘗試套用規則的次數會重設為零。

規則表格使用慣例
規則表必須遵守下列關於其語法、結構和內容的慣例。

- 規則表格必須至少包含一個規則。
- 關鍵字可以任意順序出現。
- 關鍵字只能在任何規則中併入一次。
- 關鍵字不區分大小寫。
- 關鍵字及其參數值必須與其他關鍵字至少以一個空白或逗點區隔。
- 規則的開頭或結尾，以及關鍵字、標點符號和值之間可以出現任意數目的空白。
- 每一個規則必須從新行開始。
- 為了可攜性，線路的有效長度不得大於 72 個字元。
- 使用加號 (+) 作為一行的最後一個非空白字元，以指出規則從下一行的第一個非空白字元繼續。 使用減號 (-) 作為行上的最後一個非空白字元，以指出規則從下一行開始繼續。 在關鍵字和參數內可以出現接續字元。

  例如:

  ```
  APPLNAME('ABC+
  D')
  ```

  產生 'ABCD'。

  ```
  APPLNAME('ABC-
  D')
  ```

  結果為 'ABC D'。
- 以星號 (*) 開頭的註解行可以在規則表格中的任何位置出現。
- 空白行予以忽略。
- DLQ 處理程式規則表格中的每一個項目都包含一個以上關鍵字及其相關聯參數。 參數必須遵循下列語法規則:

– 每一個參數值必須至少包含一個有效字元。 以引號括住的值中的定界引號不被視為重要。 例如，這些參數有效:

```
FORMAT('ABC')       3 個有效字元
FORMAT(ABC)         3 個有效字元
FORMAT('A')         1 個重要字元
FORMAT(A)           1 個重要字元
FORMAT(' ')         1 個重要字元
```

這些參數無效，因為它們不包含任何有效字元:

```
FORMAT('')
FORMAT( )
FORMAT()
FORMAT
```

– 萬用字元。 您可以使用問號 (?) 來取代任何單一字元，但尾端空白除外。 您可以使用星號 (*) 來取代零或多個相鄰字元。 星號 (*) 及問號 (?) 在參數值中 一**律** 解譯為萬用字元。
– 您無法在下列關鍵字的參數中包含萬用字元 :ACTION、HEADER、RETRY、FWDQ、FWDQM 及 PUTAUT。
– 當執行萬用字元相符時，參數值中的尾端空白，以及 DLQ 上訊息中的對應欄位中的尾端空白並不重要。 不過，引號中字串內的前導和內嵌空白對萬用字元相符很重要。
– 數值參數不能包含問號 (?) 萬用字元。 您可以併入星號 (*) 來取代整個數值參數，但不能併入星號作為數值參數的一部分。 例如，這些是有效的數值參數:

```
MSGTYPE(2)          只有回覆訊息合格
MSGTYPE(*)          任何訊息類型都適用
MSGTYPE('*')        任何訊息類型都適用
```

不過， `MSGTYPE('2*')` 無效，因為它包含星號 (*) 作為數值參數的一部分。

– 數值參數必須在 0-999 999 999 的範圍內。 如果參數值在此範圍內，則即使它目前在關鍵字相關的欄位中無效，也會被接受。 您可以對數值參數使用符號名稱。
– 如果字串值短於關鍵字相關的 MQDLH 或 MQMD 中的欄位，則該值會以空白填補欄位長度。 如果值 (排除星號) 比欄位長，則會診斷錯誤。 例如，這些都是 8 個字元欄位的有效字串值:

```
'ABCDEFGH'              8 個字元
'A*C*E*G*I'             5 個字元 (不含星號)
'*A*C*E*G*I*K*M*O*'     8 個字元 (不含星號)
```

– 包含空白、小寫字元或句點 (.)、正斜線 (?)、底線 (_) 及百分比符號 (%) 以外的特殊字元的字串必須以單引號括住。 未以引號括住的小寫字元會轉換成大寫。 如果字串包含引號，則必須使用兩個單引號來表示引號的開頭和結尾。 計算字串長度時，每一個出現的雙引號都會計算為單一字元。

處理規則表格

DLQ 處理程式會搜尋規則表格，以找出其型樣符合 DLQ 上訊息的規則。

搜尋會從表格中的第一個規則開始，並循序在表格中繼續執行。 找到具有相符型樣的規則時，規則表格會嘗試來自該規則的動作。 每當嘗試套用規則時， DLQ 處理程式會將該規則的重試次數增加 1。 如果第一次嘗試失敗，則會重複嘗試，直到嘗試次數符合 RETRY 關鍵字上指定的數目為止。 如果所有嘗試都失敗， DLQ 處理程式會搜尋表格中的下一個相符規則。

後續比對規則會重複此程序，直到動作成功為止。 當每一個符合規則已嘗試其 RETRY 關鍵字上指定的次數，且所有嘗試都失敗時，會假設 ACTION (IGNORE)。 如果找不到相符規則，也會假設 ACTION (IGNORE)。

註:

1. 只會針對 DLQ 上以 MQDLH 開頭的訊息來探查相符規則型樣。 不是以 MQDLH 開頭的訊息會定期報告為發生錯誤，並無限期保留在 DLQ 上。
2. 所有型樣關鍵字都可以預設，以便規則只能由動作組成。 不過請注意，僅動作規則會套用至佇列上具有 MQDLHs 且尚未根據表格中其他規則進行處理的所有訊息。
3. 當 DLQ 處理程式啟動時，會驗證規則表格，且此時會標示錯誤旗標。 (訊息及原因碼中說明 DLQ 處理程式發出的錯誤訊息。) 您可以隨時對規則表格進行變更，但這些變更在 DLQ 處理程式重新啟動之前不會生效。
4. DLQ 處理程式不會變更訊息、MQDLH 或訊息描述子的內容。 DLQ 處理程式一律使用訊息選項 MQPMO_PASS_ALL_CONTEXT 將訊息放入其他佇列。
5. 規則表格中的連續語法錯誤可能無法辨識，因為規則表格的驗證可避免產生重複錯誤。
6. DLQ 處理程式會使用 MQOO_INPUT_AS_Q_DEF 選項開啟 DLQ。
7. 可以使用相同的規則表格，針對相同的佇列同時執行 DLQ 處理程式的多個實例。 不過， DLQ 與 DLQ 處理程式之間的一對一關係更為常見。

確定已處理所有 *DLQ* 訊息

DLQ 處理程式會保留 DLQ 上已看到但未移除之所有訊息的記錄。

如果您使用 DLQ 處理程式作為過濾器，從 DLQ 擷取一小部分訊息，則 DLQ 處理程式仍會在 DLQ 上保留未處理的那些訊息記錄。 此外，即使 DLQ 定義為先進先出 (FIFO) ， DLQ 處理程式也無法保證會看到到達 DLQ 的新訊息。 如果佇列不是空的，則會定期重新掃描 DLQ 以檢查所有訊息。

基於這些原因，請嘗試確定 DLQ 包含儘可能少的訊息。 如果容許無法捨棄或轉遞至其他佇列的訊息 (不論任何原因) 累積在佇列上，則 DLQ 處理程式的工作量會增加，且 DLQ 本身有填滿的危險。

您可以採取特定措施，讓 DLQ 處理程式清空 DLQ。 例如，嘗試不使用 ACTION (IGNORE) ，這會在 DLQ 上留下訊息。 (請記住，對於表格中其他規則未明確指出的訊息，會假設 ACTION (IGNORE)。) 相反地，對於您將忽略的那些訊息，請使用可將訊息移至另一個佇列的動作。 例如:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

同樣地，將表格中的最終規則設為「擷取並更新」，以處理表格中先前規則未處理的訊息。 例如，表格中的最終規則可能如下所示:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

這會導致進入表格中最終規則的訊息轉遞至佇列 REALLY.DEAD.QUEUE，在其中可以手動處理這些訊息。 如果您沒有這類規則，則訊息可能會無限期保留在 DLQ 上。

範例 *DLQ* 處理程式規則表格

以下是包含單一控制資料輸入及數個規則的範例規則表格:

```
*****************************************************************
*    An example rules table for the STRMQMDLQ 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)
```

呼叫 *DLQ* 處理程式

使用 `runmqdlq` 指令來呼叫 DLQ 處理程式。 您可以用兩種方式來命名您要處理的 DLQ 及您要使用的佇列管理程式。

這兩種方式如下:

- 作為命令提示字元中 runmqdlq 的參數。 例如:

```
runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER <qrule.rul
```

- 在規則表格中。 例如:

```
INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
```

這些範例適用於稱為 ABC1.DEAD.LETTER.QUEUE，由佇列管理程式 ABC1.QUEUE.MANAGER。

如果您未如所示指定 DLQ 或佇列管理程式，則安裝的預設佇列管理程式會與屬於該佇列管理程式的 DLQ 一起使用。

runmqdlq 指令會從 stdin 取得其輸入; 您可以從規則表格重新導向 stdin ，使規則表格與 runmqdlq 產生關聯。

若要執行 DLQ 處理程式，您必須獲授權存取 DLQ 本身及 DLQ 上的訊息所轉遞至的任何訊息佇列。 若要讓 DLQ 處理程式在訊息環境定義中具有使用者 ID 權限的佇列上放置訊息，您也必須獲得授權來假設其他使用者的身分。

如需 runmqdlq 指令的相關資訊，請參閱 runmqdlq。

相關資訊

無法傳送郵件的佇列
未遞送訊息疑難排解

範例 *DLQ* 處理程式 *amqsdlq*

除了使用 runmqdlq 指令所呼叫的 DLQ 處理程式之外， IBM MQ 還提供範例 DLQ 處理程式 amqsdlq 的來源，其函數類似於 runmqdlq 所提供的函數。

您可以自訂 amqsdlq ，以提供符合您需求的 DLQ 處理程式。 例如，您可以決定想要一個 DLQ 處理程式，它可以處理沒有無法傳送的郵件標頭的訊息。 (預設 DLQ 處理程式和範例 amqsdlq 都只會處理 DLQ 上以無法傳送郵件的標頭 MQDLH 開頭的訊息。 不是以 MQDLH 開頭的訊息會被識別為錯誤，並無限期保留在 DLQ 上。)

MQ_INSTALLATION_PATH 代表 IBM MQ 安裝所在的高階目錄。

在 IBM MQ for Windows 中， amqsdlq 的來源是在下列目錄中提供:

```
MQ_INSTALLATION_PATH\tools\c\samples\dlq
```

並在下列目錄中提供已編譯版本:

```
MQ_INSTALLATION_PATH\tools\c\samples\bin
```

在 IBM MQ for UNIX 和 Linux 系統中， amqsdlq 的來源提供在下列目錄中:

MQ_INSTALLATION_PATH/samp/dlq

並在下列目錄中提供已編譯版本:

MQ_INSTALLATION_PATH/samp/bin

您也可以在用戶端模式下編譯 **amqsdlq** 。 如需相關資訊，請參閱 撰寫用戶端程序化應用程式、 建置 IBM MQ MQI clients 的應用程式及 在 IBM MQ MQI client 環境中執行應用程式。

DLQ 處理程式規則表格

DLQ 處理程式規則表格定義 DLQ 處理程式如何處理到達 DLQ 的訊息。

規則表格中有兩種類型的項目:

- 表格中的第一個項目 (選用) 包含 控制資料。

- 表格中的所有其他項目都是要遵循 DLQ 處理程式的 規則 。 每一個規則都包含符合訊息的 型樣 (一組訊息性質) ，以及當 DLQ 上的訊息符合指定型樣時要採取的 動作 。 規則表格中必須至少有一個規則。

規則表格中的每一個項目都包含一個以上關鍵字。

相關資訊

無法傳送郵件的佇列

未遞送訊息疑難排解

控制資料

本節說明您可以併入 DLQ 處理程式規則表格中控制資料項目的關鍵字。

註:

- 垂直線 (|) 會區隔替代方案，只能指定其中一個替代方案。
- 所有關鍵字都是選用的。

INPUTQ (*QueueName* | ' ')

您要處理的 DLQ 名稱:

1. 您提供作為 `runmqdlq` 指令參數的任何 INPUTQ 值都會置換規則表格中的任何 INPUTQ 值。
2. 如果您未指定 INPUTQ 值作為 `runmqdlq` 指令的參數，但在規則表格中 **指定** 值，則會使用規則表格中的 INPUTQ 值。
3. 如果未指定 DLQ ，或您在規則表格中指定 INPUTQ ('') ，則會使用屬於佇列管理程式的 DLQ 名稱，以及提供作為 `runmqdlq` 指令參數的名稱。
4. 如果您未將 INPUTQ 值指定為 `runmqdlq` 指令的參數或指定為規則表格中的值，則會使用屬於規則表格中 INPUTQM 關鍵字上所指名之佇列管理程式的 DLQ。

INPUTQM (*QueueManagerName* | ' ')

擁有 INPUTQ 關鍵字上所指名之 DLQ 的佇列管理程式名稱:

1. 您提供作為 `runmqdlq` 指令參數的任何 INPUTQM 值都會置換規則表格中的任何 INPUTQM 值。
2. 如果您未指定 INPUTQM 值作為 `runmqdlq` 指令的參數，則會使用規則表格中的 INPUTQM 值。
3. 如果未指定佇列管理程式，或您在規則表中指定 INPUTQM ('') ，則會使用安裝的預設佇列管理程式。

RETRYINT (間隔 | 60)

DLQ 處理程式應該重新處理 DLQ 上第一次嘗試無法處理且已要求重複嘗試的訊息的間隔 (以秒為單位)。依預設，重試間隔為 60 秒。

WAIT (YES | NO | *nnn*)

當 DLQ 處理程式偵測到沒有可處理的進一步訊息時， DLQ 處理程式是否應該等待進一步訊息到達 DLQ。

YES
DLQ 處理程式會無限期等待。

NO
當 DLQ 處理程式偵測到 DLQ 是空的或未包含可處理的訊息時，即會結束 DLQ 處理程式。

nnn
在 DLQ 處理程式偵測到佇列是空的或沒有可處理的訊息之後，它會等待 *nnn* 秒，讓新工作在結束之前到達。

對忙碌 DLQ 指定 WAIT (YES) ，並指定 WAIT (NO) 或 WAIT (*nnn*) 適用於活動層次較低的 DLQ。 如果容許 DLQ 處理程式終止，請使用觸發來重新呼叫它。 如需觸發的相關資訊，請參閱 使用觸發程式啟動 IBM MQ 應用程式。

將控制資料併入規則表格的替代方案是提供 DLQ 及其佇列管理程式的名稱作為 `runmqdlq` 指令的輸入參數。 如果您同時在規則表格中及作為 `runmqdlq` 指令的輸入指定值，則 `runmqdlq` 指令上指定的值優先。

如果您在規則表格中包括控制資料項目，則它必須是表格中的 **第一個** 項目。

規則 (型樣和動作)
型樣相符關鍵字 (與 DLQ 上的訊息相符的關鍵字) 及動作關鍵字 (決定 DLQ 處理程式如何處理相符訊息的關鍵字) 的說明。 也會提供範例規則。

型樣相符關鍵字

您用來指定 DLQ 上的訊息相符值的型樣相符關鍵字如下。 (所有型樣相符關鍵字都是選用的):

APPLIDAT (*ApplIdentity* 資料 | *)
在 DLQ 上訊息的訊息描述子 MQMD 中指定的 *ApplIdentity* 資料 值。

APPLNAME (*PutApplName* | *)
發出 MQPUT 或 MQPUT1 呼叫的應用程式名稱，如 DLQ 上訊息的訊息描述子 MQMD 的 *PutAppl* 名稱 欄位中所指定。

APPLTYPE (*PutAppl* 類型 | *)
PutAppl 類型 值，在 DLQ 上訊息的訊息描述子 MQMD 中指定。

DESTQ (*QueueName* | *)
指定訊息的訊息佇列名稱。

DESTQM (*QueueManagerName* | *)
訊息目的地之訊息佇列的佇列管理程式名稱。

回饋 (回饋 | *)
當 *MsgType* 值為 MQFB_REPORT 時， 意見 會說明報告的本質。

您可以使用符號名稱。 例如，您可以使用符號名稱 MQFB_COA 來識別 DLQ 上需要確認其到達目的地佇列的訊息。

FORMAT (格式 | *)
訊息傳送者用來說明訊息資料格式的名稱。

MSGTYPE (*MsgType* | *)
DLQ 上訊息的訊息類型。

您可以使用符號名稱。 例如，您可以使用符號名稱 MQMT_REQUEST 來識別 DLQ 上需要回覆的那些訊息。

持續保存 (持續性 | *)
訊息的持續性值。 (訊息的持續性會決定它是否在重新啟動佇列管理程式之後仍然存在。)

您可以使用符號名稱。 例如，您可以使用符號名稱 MQPER_PERSISTENT 來識別 DLQ 上持續存在的訊息。

REASON (*ReasonCode* | *)
說明訊息放置到 DLQ 的原因碼。

您可以使用符號名稱。 例如，您可以使用符號名稱 MQRC_Q_FULL 來識別放置在 DLQ 上的那些訊息，因為其目的地佇列已滿。

REPLYQ (*QueueName* | *)
在 DLQ 上訊息的訊息描述子 MQMD 中指定的回覆目的地佇列名稱。

REPLYQM (*QueueManagerName* | *)
回覆目的地佇列的佇列管理程式名稱，如 DLQ 上訊息的訊息描述子 MQMD 中所指定。

USERID (*UserIdentifier* | *)
在 DLQ 上產生訊息之使用者的使用者 ID ，如 DLQ 上訊息的訊息描述子 MQMD 中所指定。

動作關鍵字

用來說明如何處理相符訊息的動作關鍵字如下:

ACTION (DISCARD | IGNORE | RETRY | FWD)
DLQ 上任何符合此規則中定義之型樣的訊息所要採取的動作。

捨棄
從 DLQ 刪除訊息。

IGNORE
在 DLQ 上保留訊息。

重試
如果第一次嘗試將訊息放入其目的地佇列失敗，請重試。 RETRY 關鍵字會設定嘗試實作動作的次數。 控制資料的 RETRYINT 關鍵字控制兩次嘗試之間的間隔。

轉遞
將訊息轉遞至 FWDQ 關鍵字上指定的佇列。

您必須指定 ACTION 關鍵字。

FWDQ (*QueueName* |&DESTQ|&REPLYQ)
要求 ACTION (FWD) 時要將訊息轉遞至其中的訊息佇列名稱。

QueueName
訊息佇列的名稱。 FWDQ ('') 無效。

&DESTQ
從 MQDLH 結構中的 *DestQName* 欄位取得佇列名稱。

&REPLYQ
從訊息描述子 MQMD 的 *ReplyToQ* 欄位中取得佇列名稱。

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* 名稱 |&DESTQM|&REPLYQM|' ')
要將訊息轉遞至其中之佇列的佇列管理程式。

***QueueManager* 名稱**
要求 ACTION (FWD) 時要轉遞訊息之佇列的佇列管理程式名稱。

&DESTQM
從 MQDLH 結構中的 *DestQMgr* 名稱 欄位取得佇列管理程式名稱。

&REPLYQM
從訊息描述子 MQMD 的 *ReplyToQMgr* 欄位中取得佇列管理程式名稱。

' '
FWDQM ('') 是預設值，用於識別本端佇列管理程式。

標頭 (YES | NO)
MQDLH 是否應該保留在要求 ACTION (FWD) 的訊息上。 依預設， MQDLH 會保留在訊息上。 HEADER 關鍵字對 FWD 以外的動作無效。

PUTAUT (DEF | CTX)
DLQ 處理程式應用來放置訊息的權限:

DEF
放置具有 DLQ 處理程式本身權限的訊息。

CTX
將具有使用者 ID 權限的訊息放置在訊息環境定義中。 如果您指定 PUTAUT (CTX) ，則必須授權您採用其他使用者的身分。

RETRY (*RetryCount* | 1)
嘗試動作的次數，範圍為 1-999,999,999 (在控制資料的 RETRYINT 關鍵字上指定的間隔)。 DLQ 處理程式嘗試實作任何特定規則的次數特定於 DLQ 處理程式的現行實例; 此計數不會在重新啟動之間持續保存。 如果 DLQ 處理程式重新啟動，則嘗試套用規則的次數會重設為零。

範例規則

以下是 DLQ 處理程式規則表格中的規則範例:

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
ACTION (RETRY) RETRY (3)
```

此規則指示 DLQ 處理程式三次嘗試將任何放置在 DLQ 上的持續訊息遞送至其目的地佇列，因為禁止 MQPUT 及 MQPUT1 。

本節其餘部分會說明您可以在規則上使用的所有關鍵字。 請注意下列項目：

- 關鍵字的預設值 (如果有的話) 會畫底線。 對於大部分關鍵字，預設值為 * (星號) ，其符合任何值。
- 垂直線 (|) 會區隔替代方案，只能指定其中一個替代方案。
- ACTION 以外的所有關鍵字都是選用的。

規則表格使用慣例

DLQ 處理程式規則表格的語法、結構及內容必須遵守這些慣例。

規則表格必須遵循下列慣例：

- 規則表格必須至少包含一個規則。
- 關鍵字可以任意順序出現。
- 關鍵字只能在任何規則中併入一次。
- 關鍵字不區分大小寫。
- 關鍵字及其參數值必須與其他關鍵字至少以一個空白或逗點區隔。
- 規則的開頭或結尾，以及關鍵字、標點符號和值之間可以有任意數目的空白。
- 每一個規則必須從新行開始。
- 在 Windows 系統上，表格中的最後一個規則必須以換行字元結尾。 若要達到此目的，您可以確保在規則結尾按 Enter 鍵，以便表格的最後一行是空白行。
- 基於可攜性原因，行的有效長度不得大於 72 個字元。
- 使用加號 (+) 作為行上的最後一個非空白字元，以指出規則從下一行中的第一個非空白字元繼續。 使用減號 (-) 作為一行上的最後一個非空白字元，以指出規則從下一行開始繼續。 在關鍵字和參數內可以出現接續字元。

  例如：

  ```
  APPLNAME('ABC+
  D')
  ```

  產生 'ABCD' ，以及

  ```
  APPLNAME('ABC-
  D')
  ```

  結果為 'ABC D'。
- 以星號 (*) 開頭的註解行可以在規則表格中的任何位置出現。
- 空白行予以忽略。
- DLQ 處理程式規則表格中的每一個項目都包含一個以上關鍵字及其相關聯參數。 參數必須遵循下列語法規則：
  - 每一個參數值必須至少包含一個有效字元。 以引號括住的值中的定界單引號不被視為有效。 例如，這些參數有效：

FORMAT('ABC')	3 個有效字元
FORMAT(ABC)	3 個有效字元
FORMAT('A')	1 個重要字元
FORMAT(A)	1 個重要字元
FORMAT(' ')	1 個重要字元

    這些參數無效，因為它們不包含任何有效字元：

```
FORMAT('')
FORMAT( )
FORMAT()
FORMAT
```

- 萬用字元。 您可以使用問號 (?) 代替任何單一字元，但尾端空白除外; 您可以使用星號 (*) 代替零或多個相鄰字元。 星號 (*) 及問號 (?) 在參數值中 **一律** 解譯為萬用字元。
- 在下列關鍵字的參數中不能包含萬用字元 :ACTION、HEADER、RETRY、FWDQ、FWDQM 及 PUTAUT。
- 當執行萬用字元相符時，參數值中的尾端空白，以及 DLQ 上訊息中的對應欄位中的尾端空白並不重要。 不過，字串內以單引號括住的前導和內嵌空白對萬用字元相符很重要。
- 數值參數不能包含問號 (?) 萬用字元。 您可以使用星號 (*) 來取代整個數值參數，但不能作為數值參數的一部分。 例如，這些是有效的數值參數:

```
MSGTYPE(2)          只有回覆訊息合格
MSGTYPE(*)          任何訊息類型都適用
MSGTYPE('*')        任何訊息類型都適用
```

  不過， `MSGTYPE('2*')` 無效，因為它包含星號 (*) 作為數值參數的一部分。
- 數值參數必須在 0-999 999 999 的範圍內。 如果參數值在此範圍內，則即使它目前在關鍵字相關的欄位中無效，也會被接受。 您可以對數值參數使用符號名稱。
- 如果字串值短於關鍵字相關的 MQDLH 或 MQMD 中的欄位，則該值會以空白填補欄位長度。 如果值 (排除星號) 比欄位長，則會診斷錯誤。 例如，這些都是 8 個字元欄位的有效字串值:

```
'ABCDEFGH'          8 個字元
'A*C*E*G*I'         5 個字元 (不含星號)
'*A*C*E*G*I*K*M*O   8 個字元 (不含星號)
*'
```

- 以單引號括住包含空格、小寫字元或句點 (.)、正斜線 (?)、底線 (_) 及百分比符號 (%) 以外的特殊字元的字串。 未以單引號括住的小寫字元會轉換成大寫。 如果字串包括引號，請使用兩個單引號來表示引號的開頭和結尾。 計算字串長度時，每一個出現的雙引號都會計算為單一字元。

如何處理規則表格

DLQ 處理程式會在規則表格中搜尋型樣符合 DLQ 上訊息的規則。

搜尋會從表格中的第一個規則開始，並循序在表格中繼續執行。 當 DLQ 處理程式找到具有相符型樣的規則時，它會從該規則採取動作。 每當 DLQ 處理程式套用規則時，它會將該規則的重試次數增加 1。 如果第一次嘗試失敗， DLQ 處理程式會重試，直到嘗試次數符合 RETRY 關鍵字上指定的數目為止。 如果所有嘗試都失敗， DLQ 處理程式會搜尋表格中的下一個相符規則。

後續比對規則會重複此程序，直到動作成功為止。 當每一個符合規則已嘗試其 RETRY 關鍵字上指定的次數，且所有嘗試都失敗時，會假設 ACTION (IGNORE)。 如果找不到相符規則，也會假設 ACTION (IGNORE)。

註:

1. 只會針對 DLQ 上以 MQDLH 開頭的訊息來探查相符規則型樣。 不是以 MQDLH 開頭的訊息會定期報告為發生錯誤，並無限期保留在 DLQ 上。
2. 所有型樣關鍵字都可以預設，以便規則只能由動作組成。 不過請注意，僅動作規則會套用至佇列上具有 MQDLHs 且尚未根據表格中其他規則進行處理的所有訊息。
3. 當 DLQ 處理程式啟動時，會驗證規則表格，並在該時間標示錯誤。 您可以隨時對規則表格進行變更，但這些變更在 DLQ 處理程式重新啟動之前不會生效。

4. DLQ 處理程式不會變更訊息內容、MQDLH 或訊息描述子。 DLQ 處理程式一律使用訊息選項 MQPMO_PASS_ALL_CONTEXT 將訊息放入其他佇列。
5. 可能無法辨識規則表格中的連續語法錯誤，因為規則表格的設計旨在避免在驗證期間產生重複錯誤。
6. DLQ 處理程式會使用 MQOO_INPUT_AS_Q_DEF 選項開啟 DLQ。
7. 可以使用相同的規則表格，針對相同的佇列同時執行 DLQ 處理程式的多個實例。 不過， DLQ 與 DLQ 處理程式之間的一對一關係更為常見。

相關資訊

無法傳送郵件的佇列

未遞送訊息疑難排解

確定已處理所有 *DLQ* 訊息

DLQ 處理程式會保留 DLQ 上已看到但未移除之所有訊息的記錄。

如果您使用 DLQ 處理程式作為過濾器，從 DLQ 擷取一小部分訊息，則 DLQ 處理程式仍必須保留 DLQ 上未處理之訊息的記錄。 此外，即使 DLQ 定義為先進先出 (FIFO) ， DLQ 處理程式也無法保證看到到達 DLQ 的新訊息。 如果佇列不是空的，則會定期重新掃描 DLQ 以檢查所有訊息。

基於這些原因，請嘗試確保 DLQ 包含儘可能少的訊息; 如果容許無法捨棄或轉遞至其他佇列的訊息 (不論原因為何) 累積在佇列上，則 DLQ 處理程式的工作量會增加，且 DLQ 本身可以填滿。

您可以採取特定措施，讓 DLQ 處理程式清空 DLQ。 例如，嘗試不使用 ACTION (IGNORE) ，這會在 DLQ 上留下訊息。 (請記住，對於表格中其他規則未明確指出的訊息， 會假設 ACTION (IGNORE)。) 相反地，對於您將忽略的那些訊息，請使用可將訊息移至另一個佇列的動作。 例如:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

同樣地，將表格中的最終規則設為「擷取並更新」，以處理表格中先前規則未處理的訊息。 例如，表格中的最終規則可能如下所示:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

這會將符合表格中最終規則的訊息轉遞至佇列 REALLY.DEAD.QUEUE，在其中可以手動處理這些訊息。 如果您沒有這類規則，則訊息可能會無限期保留在 DLQ 上。

範例 *DLQ* 處理程式規則表格

runmqdlq 指令的規則表格範例，包含單一控制資料項目及數個規則。

```
*********************************************************************
*    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)
```

相關資訊

無法傳送郵件的佇列

未遞送訊息疑難排解

runmqdlq (執行無法傳送郵件的佇列處理程式)

使用模型佇列

如果佇列管理程式收到來自應用程式的 MQI 呼叫，並指定已定義為模型佇列的佇列名稱，則會建立 動態佇列 。 建立佇列時，佇列管理程式會產生新動態佇列的名稱。 模型佇列 是一個範本，指定從它建立的任何動態佇列的屬性。 模型佇列為應用程式根據需要建立佇列提供方便的方法。

定義模型佇列

您可以使用定義本端佇列的相同方式，來定義具有一組屬性的模型佇列。 模型佇列和本端佇列具有相同的屬性集，但在模型佇列上，您可以指定所建立的動態佇列是暫時還是永久。 (永久佇列是在佇列管理程式重新啟動之間維護，暫時佇列則不是。) 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)
```

此指令會建立模型佇列定義。 從 DEFTYPE 屬性中，您可以看到從這個範本建立的實際佇列是永久動態佇列。 任何未指定的屬性都會自動從 SYSYTEM.DEFAULT.MODEL.QUEUE 預設佇列。

當您定義模型佇列時，可以使用 LIKE 及 REPLACE 屬性，其方式與您將它們與本端佇列搭配使用的方式相同。

搭配使用其他指令與模型佇列

您可以使用適當的 MQSC 指令來顯示或變更模型佇列的屬性，或刪除模型佇列物件。 例如：

使用下列指令來顯示模型佇列的屬性:

```
DISPLAY QUEUE (GREEN.MODEL.QUEUE)
```

使用下列指令來變更模型，以在從此模型建立的任何動態佇列上啟用放置:

```
ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)
```

使用下列指令來刪除此模型佇列:

```
DELETE QMODEL (RED.MODEL.QUEUE)
```

使用管理主題

使用 MQSC 指令來管理管理主題。

如需這些指令的詳細資訊，請參閱 MQSC 指令 。

相關概念

第 99 頁的『定義管理主題』
使用 MQSC 指令 **DEFINE TOPIC** 來建立管理主題。 在定義管理主題時，您可以選擇性地設定每一個主題屬性。

第 99 頁的『顯示管理主題物件屬性』
請使用 MQSC 指令 **DISPLAY TOPIC** 來顯示管理主題物件。

第 100 頁的『變更管理主題屬性』
您可以使用 **ALTER TOPIC** 指令或 **DEFINE TOPIC** 指令搭配 **REPLACE** 屬性，以兩種方式來變更主題屬性。

第 100 頁的『複製管理主題定義』
您可以在 **DEFINE** 指令上使用 LIKE 屬性來複製主題定義。

第 100 頁的『刪除管理主題定義』
您可以使用 MQSC 指令 **DELETE TOPIC** 來刪除管理主題。

相關資訊

管理主題物件

定義管理主題

使用 MQSC 指令 **DEFINE TOPIC** 來建立管理主題。 在定義管理主題時，您可以選擇性地設定每一個主題屬性。

主題的任何未明確設定的屬性都會繼承自預設管理主題 SYSTEM.DEFAULT.TOPIC，安裝系統安裝時建立的。

例如，下面的 **DEFINE TOPIC** 指令定義稱為 **ORANGE.TOPIC** 的主題，具有下列性質:

- 解析為主題字串 ORANGE。 如需如何使用主題字串的相關資訊，請參閱 結合主題字串。
- 任何設為 ASPARENT 的屬性都會使用這個主題的上層主題所定義的屬性。 此動作會在主題樹狀結構上重複，直到根主題 SYSTEM.BASE.TOPIC 。 如需相關資訊，請參閱 主題樹狀結構。

```
DEFINE TOPIC (ORANGE.TOPIC) +
TOPICSTR (ORANGE) +
DEFPRTY(ASPARENT) +
NPMSGDLV(ASPARENT)
```

註:

- 除了主題字串的值之外，所有顯示的屬性值都是預設值。 它們僅在這裡顯示作為圖解。 如果您確定預設值是您想要的或尚未變更，則可以省略它們。 另請參閱 第 99 頁的『顯示管理主題物件屬性』。
- 如果您在相同佇列管理程式上已有名稱為 ORANGE.TOPIC，此指令失敗。 如果您要改寫主題的現有定義，請使用 REPLACE 屬性，但另請參閱 第 100 頁的『變更管理主題屬性』

顯示管理主題物件屬性

請使用 MQSC 指令 **DISPLAY TOPIC** 來顯示管理主題物件。

若要顯示所有主題，請使用:

```
DISPLAY TOPIC(ORANGE.TOPIC)
```

您可以透過個別指定屬性來選擇性地顯示屬性。 例如:

```
DISPLAY TOPIC(ORANGE.TOPIC) +
TOPICSTR +
DEFPRTY +
NPMSGDLV
```

此指令會顯示三個指定的屬性，如下所示:

```
AMQ8633: Display topic details.
     TOPIC(ORANGE.TOPIC)                                     TYPE(LOCAL)
     TOPICSTR(ORANGE)                                           DEFPRTY(ASPARENT)
     NPMSGDLV(ASPARENT)
```

若要顯示在執行時期使用的 ASPARENT 值主題，請使用 DISPLAY TPSTATUS。 例如，使用:

```
DISPLAY TPSTATUS(ORANGE) DEFPRTY NPMSGDLV
```

指令會顯示下列詳細資料:

```
AMQ8754: Display topic status details.
TOPICSTR(ORANGE)            DEFPRTY(0)
NPMSGDLV(ALLAVAIL)
```

當您定義管理主題時，它會採用預設管理主題 (稱為 SYSTEM.DEFAULT.TOPIC。 若要查看這些預設屬性為何，請使用下列指令:

```
DISPLAY TOPIC (SYSTEM.DEFAULT.TOPIC)
```

變更管理主題屬性

您可以使用 **ALTER TOPIC** 指令或 **DEFINE TOPIC** 指令搭配 **REPLACE** 屬性，以兩種方式來變更主題屬性。

例如，如果您想要變更遞送至名為 ORANGE.TOPIC，若要設為 5，請使用下列其中一個指令。

- 使用 **ALTER** 指令:

```
ALTER TOPIC(ORANGE.TOPIC) DEFPRTY(5)
```

  這個指令會將單一屬性 (即遞送給這個主題之訊息的預設優先順序) 變更為 5; 所有其他屬性維持相同。

- 使用 **DEFINE** 指令:

```
DEFINE TOPIC(ORANGE.TOPIC) DEFPRTY(5) REPLACE
```

  這個指令會變更遞送給這個主題之訊息的預設優先順序。 所有其他屬性都會獲得其預設值。

如果您變更傳送至這個主題之訊息的優先順序，現有的訊息不會受到影響。 不過，如果發佈應用程式未提供，則任何新訊息都會使用指定的優先順序。

複製管理主題定義

您可以在 **DEFINE** 指令上使用 LIKE 屬性來複製主題定義。

例如:

```
DEFINE TOPIC (MAGENTA.TOPIC) +
LIKE (ORANGE.TOPIC)
```

此指令會建立主題 MAGENTA.TOPIC，具有與原始主題 ORANGE.TOPIC，而不是系統預設管理主題。 輸入要複製的主題名稱，與您建立主題時輸入的名稱完全相同。 如果名稱包含小寫字元，請以單引號括住名稱。

您也可以使用 **DEFINE** 指令的這種形式來複製主題定義，但對原始的屬性進行變更。 例如:

```
DEFINE TOPIC(BLUE.TOPIC) +
TOPICSTR(BLUE) +
LIKE(ORANGE.TOPIC)
```

您也可以複製 BLUE.TOPIC 至主題 GREEN.TOPIC，並指定當發佈無法遞送至其正確的訂閱者佇列時，不會將它們放置在無法傳送郵件的佇列上。 例如:

```
DEFINE TOPIC(GREEN.TOPIC) +
TOPICSTR(GREEN) +
LIKE(BLUE.TOPIC) +
USEDLQ(NO)
```

刪除管理主題定義

您可以使用 MQSC 指令 **DELETE TOPIC** 來刪除管理主題。

```
DELETE TOPIC(ORANGE.TOPIC)
```

應用程式將無法再開啟主題以進行發佈，或使用物件名稱 ORANGE.TOPIC。 發佈已開啟主題的應用程式可以繼續發佈已解析的主題字串。 在刪除主題之後，已對此主題進行的任何訂閱都會繼續接收發佈。

未參照此主題物件，但使用此主題物件所代表的已解析主題字串 (在此範例中為 'ORANGE') 的應用程式會繼續運作。 在此情況下，它們會從主題樹狀結構中較高位置的主題物件繼承內容。 如需相關資訊，請參閱 主題樹狀結構。

使用訂閱

使用 MQSC 指令來管理訂閱。

訂閱可以是 **SUBTYPE** 屬性中定義的三種類型之一:

ADMIN
由使用者以管理方式定義。

Proxy
內部建立的訂閱，用於在佇列管理程式之間遞送發佈。

API
以程式化方式建立，例如，使用 MQI MQSUB 呼叫。

如需這些指令的詳細資訊，請參閱 MQSC 指令 。

相關概念

第 101 頁的『定義管理訂閱』
使用 MQSC 指令 **DEFINE SUB** 來建立管理訂閱。 您也可以使用預設本端訂閱定義中定義的預設值。 或者，您可以從預設本端訂閱 SYSTEM.DEFAULT.SUB (在安裝系統時建立)。

第 102 頁的『顯示訂閱的屬性』
您可以使用 **DISPLAY SUB** 指令來顯示佇列管理程式已知的任何訂閱的已配置屬性。

第 102 頁的『變更本端訂閱屬性』
您可以使用 **ALTER SUB** 指令或 **DEFINE SUB** 指令搭配 **REPLACE** 屬性，以兩種方式來變更訂閱屬性。

第 103 頁的『複製本端訂閱定義』
您可以在 **DEFINE** 指令上使用 **LIKE** 屬性來複製訂閱定義。

第 103 頁的『刪除訂閱』
您可以使用 MQSC 指令 **DELETE SUB** 來刪除本端訂閱。

定義管理訂閱

使用 MQSC 指令 **DEFINE SUB** 來建立管理訂閱。 您也可以使用預設本端訂閱定義中定義的預設值。 或者，您可以從預設本端訂閱 SYSTEM.DEFAULT.SUB (在安裝系統時建立)。

例如，後面的 **DEFINE SUB** 指令定義稱為 ORANGE 且具有下列性質的訂閱:

- 可延續訂閱，表示它在佇列管理程式重新啟動時持續保存，期限無限制。
- 接收對 ORANGE 主題字串所做的發佈，以及發佈應用程式所設定的訊息優先順序。
- 此訂閱的遞送發佈會傳送至本端佇列 SUBQ ，此佇列必須在定義訂閱之前定義。

```
DEFINE SUB (ORANGE) +
TOPICSTR (ORANGE) +
DESTCLAS (PROVIDED) +
DEST (SUBQ) +
EXPIRY (UNLIMITED) +
PUBPRTY (ASPUB)
```

註:

- 訂閱與主題字串名稱不必相符。
- 除了目的地和主題字串的值之外，所有顯示的屬性值都是預設值。 它們僅在這裡顯示作為圖解。 如果您確定預設值是您想要的或尚未變更，則可以省略它們。 另請參閱「第 102 頁的『顯示訂閱的屬性』」。
- 如果您已在名稱為 ORANGE 的相同佇列管理程式上具有本端訂閱，則此指令會失敗。 如果您要改寫佇列的現有定義，請使用 **REPLACE** 屬性，但另請參閱 第 102 頁的『變更本端訂閱屬性』。
- 如果佇列 SUBQ 不存在，則此指令會失敗。

顯示訂閱的屬性

您可以使用 **DISPLAY SUB** 指令來顯示佇列管理程式已知的任何訂閱的已配置屬性。

例如，使用:

```
DISPLAY SUB (ORANGE)
```

您可以透過個別指定屬性來選擇性地顯示屬性。 例如:

```
DISPLAY SUB (ORANGE) +
SUBID +
TOPICSTR +
DURABLE
```

此指令會顯示三個指定的屬性，如下所示:

```
AMQ8096: IBM MQ subscription inquired.
     SUBID(414D5120414141202020202020202020EE921E4E20002A03)
     SUB(ORANGE)                                                    TOPICSTR(ORANGE)
     DURABLE(YES)
```

TOPICSTR 是此訂閱者正在其上運作的已解析主題字串。 當定義訂閱來使用主題物件時，會使用該物件中的主題字串作為建立訂閱時所提供之主題字串的字首。 SUBID 是建立訂閱時由佇列管理程式指派的唯一 ID。 這是要顯示的有用屬性，因為部分訂閱名稱可能很長，或在不同的字集中，可能變成不切實際。

顯示訂閱的替代方法是使用 SUBID:

```
DISPLAY SUB +
SUBID(414D5120414141202020202020202020EE921E4E20002A03) +
TOPICSTR +
DURABLE
```

此指令會提供與之前相同的輸出:

```
AMQ8096: IBM MQ subscription inquired.
SUBID(414D5120414141202020202020202020EE921E4E20002A03)
     SUB(ORANGE)                                                    TOPICSTR(ORANGE)
     DURABLE(YES)
```

依預設不會顯示佇列管理程式上的 Proxy 訂閱。 若要顯示它們，請指定 **SUBTYPE** 為 PROXY 或 ALL。

您可以使用 DISPLAY SBSTATUS 指令來顯示「執行時期」屬性。 例如，使用下列指令:

```
DISPLAY SBSTATUS(ORANGE) NUMMSGS
```

會顯示下列輸出:

```
AMQ8099: IBM MQ subscription status inquired.
SUB(ORANGE)
SUBID(414D5120414141202020202020202020EE921E4E20002A03)
NUMMSGS(0)
```

當您定義管理訂閱時，它會採用預設訂閱 (稱為 SYSTEM.DEFAULT.SUB。 若要查看這些預設屬性為何，請使用下列指令:

```
DISPLAY SUB (SYSTEM.DEFAULT.SUB)
```

變更本端訂閱屬性

您可以使用 **ALTER SUB** 指令或 **DEFINE SUB** 指令搭配 **REPLACE** 屬性，以兩種方式來變更訂閱屬性。

例如，如果您想要將遞送至稱為 ORANGE 之訂閱的訊息優先順序變更為 5，請使用下列其中一個指令:

- 使用 ALTER 指令:

```
ALTER SUB(ORANGE) PUBPRTY(5)
```

此指令會將遞送至此訂閱之訊息優先順序的單一屬性變更為 5; 所有其他屬性則維持相同。

- 使用 DEFINE 指令:

```
DEFINE SUB (ORANGE) PUBPRTY(5) REPLACE
```

此指令不僅會變更遞送至此訂閱之訊息的優先順序，還會變更所有其他屬性的預設值。

如果您變更傳送至此訂閱之訊息的優先順序，則現有訊息不受影響。 不過，任何新訊息都是指定的優先順序。

複製本端訂閱定義

您可以在 **DEFINE** 指令上使用 **LIKE** 屬性來複製訂閱定義。

例如:

```
DEFINE SUB (BLUE) +
LIKE (ORANGE)
```

您也可以將 sub REAL 的屬性複製到 sub THIRD.SUB，並指定已遞送發佈資訊的 correlID 是 THIRD，而不是發佈者 correlID。 例如:

```
DEFINE SUB(THIRD.SUB) +
LIKE(BLUE) +
DESTCORL(ORANGE)
```

刪除訂閱

您可以使用 MQSC 指令 **DELETE SUB** 來刪除本端訂閱。

```
DELETE SUB(ORANGE)
```

您也可以使用 SUBID 來刪除訂閱:

```
DELETE SUB SUBID(414D5120414141202020202020202020EE921E4E20002A03)
```

檢查訂閱的訊息

關於這項作業

當定義訂閱時，它會與佇列相關聯。 符合此訂閱的已發佈訊息會放入此佇列。

請注意，下列 **runmqsc** 指令僅顯示已接收訊息的訂閱。

若要檢查目前已排入訂閱佇列的訊息，請執行下列步驟:

程序

1. 若要檢查佇列中是否有訂閱類型 **DISPLAY SBSTATUS(<sub_name>) NUMMSGS** 的訊息，請參閱 第 102 頁的『顯示訂閱的屬性』。
2. 如果 **NUMMSGS** 值大於零，請鍵入 **DISPLAY SUB(<sub_name>)DEST** 來識別與訂閱相關聯的佇列。
3. 使用傳回的佇列名稱，您可以遵循 第 78 頁的『瀏覽佇列』中說明的技術來檢視訊息。

使用服務

服務物件是一種方法，可用來在佇列管理程式中管理其他處理程序。 使用服務，您可以定義在佇列管理程式啟動及結束時啟動及停止的程式。 一律會以啟動佇列管理程式之使用者的使用者 ID 來啟動 IBM MQ 服務。

若要定義新的 IBM MQ 服務定義，請使用 MQSC 指令 DEFINE SERVICE。

服務物件可以是下列其中一種類型:

伺服器
伺服器是將參數 SERVTYPE 指定為 SERVER 的服務物件。 伺服器服務物件是在啟動指定的佇列管理程式時所執行的程式定義。 伺服器服務物件定義通常長時間執行的程式。 例如，伺服器服務物件可用來執行觸發監視器處理程序，例如 `runmqtrm` 。

一個伺服器服務物件只能同時執行一個實例。 可以使用 MQSC 指令 DISPLAY SVSTATUS 來監視執行中伺服器服務物件的狀態。

指令
指令是將參數 SERVTYPE 指定為 COMMAND 的服務物件。 指令服務物件類似於伺服器服務物件，不過指令服務物件的多個實例可以同時執行，且無法使用 MQSC 指令 DISPLAY SVSTATUS 來監視其狀態。

如果執行 MQSC 指令 STOP SERVICE ，則不會執行任何檢查，以判斷 MQSC 指令 START SERVICE 所啟動的程式在執行停止程式之前是否仍在作用中。

定義服務物件

您可以定義具有各種屬性的服務物件。

屬性如下:

SERVTYPE
定義服務物件的類型。 可能的值如下:

SERVER
伺服器服務物件。

一次只能執行一個伺服器服務物件實例。 可以使用 MQSC 指令 DISPLAY SVSTATUS 來監視伺服器服務物件的狀態。

指令
指令服務物件。

一個指令服務物件的多個實例可以同時執行。 無法監視指令服務物件的狀態。

STARTCMD
執行以啟動服務的程式。 必須指定程式的完整路徑。

STARTARG
傳遞至啟動程式的引數。

STDERR
指定服務程式的標準錯誤 (stderr) 應重新導向至其中的檔案路徑。

STDOUT
指定服務程式標準輸出 (stdout) 應重新導向至其中的檔案路徑。

STOPCMD
執行以停止服務的程式。 必須指定程式的完整路徑。

STOPARG
傳遞至停止程式的引數。

CONTROL
指定如何啟動和停止服務:

手動
服務不會自動啟動或自動停止。 它是透過使用 START SERVICE 及 STOP SERVICE 指令來控制。 這是預設值。

QMGR
在啟動和停止佇列管理程式的同時，要啟動和停止所定義的服務。

STARTONLY
服務會在佇列管理程式啟動的同時啟動，但在佇列管理程式停止時不會要求停止。

相關概念
第 105 頁的『管理服務』
透過使用 CONTROL 參數，服務物件的實例可以由佇列管理程式自動啟動及停止，或使用 MQSC 指令 START SERVICE 及 STOP SERVICE 啟動及停止。

管理服務

透過使用 CONTROL 參數，服務物件的實例可以由佇列管理程式自動啟動及停止，或使用 MQSC 指令 START SERVICE 及 STOP SERVICE 啟動及停止。

當啟動服務物件的實例時，會將訊息寫入佇列管理程式錯誤日誌，其中包含服務物件的名稱及已啟動處理程序的處理程序 ID。 伺服器服務物件啟動的日誌項目範例如下:

```
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.
```

啟動指令服務物件的日誌項目範例如下:

```
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.
```

當實例伺服器服務停止時，會將訊息寫入佇列管理程式錯誤日誌，其中包含服務名稱及結束處理程序的處理程序 ID。 停止伺服器服務物件的日誌項目範例如下:

```
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.
```

相關參考
第 105 頁的『其他環境變數』
啟動服務時，啟動服務程序所處的環境繼承自佇列管理程式環境。 可以透過將您要定義的變數新增至其中一個 service.env 環境置換檔案，來定義要在服務處理程序的環境中設定的其他環境變數。

其他環境變數

啟動服務時，啟動服務程序所處的環境繼承自佇列管理程式環境。 可以透過將您要定義的變數新增至其中一個 service.env 環境置換檔案，來定義要在服務處理程序的環境中設定的其他環境變數。

註:

您可以將環境變數新增至兩個可能的檔案:

- 機器範圍 service.env 檔案，位於 UNIX 和 Linux 系統上的 /var/mqm 中，或在 Windows 系統上安裝期間所選取的資料目錄中。
- 佇列管理程式範圍 service.env 檔，位於佇列管理程式資料目錄中。 例如，名為 QMNAME 之佇列管理程式的環境置換檔案位置為:
  - 在 UNIX 和 Linux 系統上: /var/mqm/qmgrs/QMNAME/service.env
  - 在 Windows 系統上: C:\ProgramData\IBM\MQ\qmgrs\QMNAME\service.env

這兩個檔案都會處理 (如果有的話)，且佇列管理程式範圍檔案中的定義優先於機器範圍檔案中的那些定義。

任何環境變數都可以在 service.env 中指定。 例如，如果 IBM MQ 服務執行一些指令，則在 service.env 檔案中設定 PATH 使用者變數可能很有用。 您設定變數的值不能是環境變數; 例如 CLASSPATH= *%CLASSPATH%* 不正確。 同樣地，在 Linux PATH= *$PATH* :/opt/mqm/bin 上，會產生非預期的結果。

CLASSPATH 必須大寫，且類別路徑陳述式只能包含文字。 部分服務 (例如遙測) 會設定自己的類別路徑。 會將定義在 service.env 中的 CLASSPATH 新增至該類別路徑。

檔案中定義的變數格式 service.env 是名稱/值變數配對的清單。 每一個變數都必須在新行上定義，且會在明確定義時採用每一個變數，包括空格。 service.env 檔案的範例如下:

```
#********************************************************************#
#*                                                                  *#
#* <N_OCO_COPYRIGHT>                                                *#
#* Licensed Materials - Property of IBM                             *#
#*                                                                  *#
#* 63H9336                                                          *#
#* (C) Copyright IBM Corporation 2005, 2022.                        *#
#*                                                                  *#
#* <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
```

相關參考

第 106 頁的『服務定義上的可取代插入項目』
在服務物件的定義中，可以替換記號。 執行服務程式時，會自動將替代的記號取代為其擴充文字。 替代記號可以從下列一般記號清單中取得，或從檔案 service.env 中定義的任何變數取得。

服務定義上的可取代插入項目

在服務物件的定義中，可以替換記號。 執行服務程式時，會自動將替代的記號取代為其擴充文字。 替代記號可以從下列一般記號清單中取得，或從檔案 service.env 中定義的任何變數取得。

以下是可在服務物件定義中用來替代記號的一般記號:

MQ_INSTALL_PATH
IBM MQ 的安裝位置。

MQ_DATA_PATH
IBM MQ 資料目錄的位置:

- 在 UNIX 和 Linux 系統上， IBM MQ 資料目錄位置是 /var/mqm/
- 在 Windows 系統上， IBM MQ 資料目錄的位置是在安裝 IBM MQ 期間選取的資料目錄

QMNAME
現行佇列管理程式名稱。

MQ 服務 _ 名稱
服務的名稱。

MQ_SERVER_PID
此記號只能由 STOPARG 和 STOPCMD 引數使用。

對於伺服器服務物件，此記號會取代為 STARTCMD 及 STARTARG 引數所啟動之處理程序的處理程序 ID。 否則，此記號將取代為 0。

MQ 佇列管理程式資料路徑
佇列管理程式資料目錄的位置。

MQ 佇列管理程式資料名稱
佇列管理程式的轉換名稱。 如需名稱轉換的相關資訊，請參閱 瞭解 IBM MQ 檔名。

若要使用可更換的插入項目，請將 + 字元內的記號插入任何 STARTCMD、STARTARG、STOPCMD、STOPARG、STDOUT 或 STDERR 字串中。 如需此範例，請參閱 第 107 頁的『使用服務物件的範例』。

使用服務物件的範例

本節中的服務是以 UNIX 樣式路徑分隔字元撰寫，除非另有說明。

使用伺服器服務物件

此範例顯示如何定義、使用及變更伺服器服務物件，以啟動觸發監視器。

1. 使用下列 MQSC 指令定義伺服器服務物件:

```
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+ 是代表安裝目錄的記號。
   +QMNAME+ 是代表佇列管理程式名稱的記號。
   ACCOUNTS.INITIATION.QUEUE 是起始佇列。
   amqsstop 是隨附於 IBM MQ 的範例程式，它會要求佇列管理程式岔斷處理程序 ID 的所有連線。
   amqsstop 會產生 PCF 指令，因此指令伺服器必須在執行中。
   +MQ_SERVER_PID+ 是代表傳遞至停止程式之處理程序 ID 的記號。

   如需一般記號的清單，請參閱 第 106 頁的『服務定義上的可取代插入項目』。
2. 下次啟動佇列管理程式時，將執行伺服器服務物件的實例。 不過，我們將使用下列 MQSC 指令立即啟動伺服器服務物件的實例:

```
START SERVICE(S1)
```

3. 使用下列 MQSC 指令顯示伺服器服務程序的狀態:

```
DISPLAY SVSTATUS(S1)
```

4. 此範例現在顯示如何變更伺服器服務物件，並透過手動重新啟動伺服器服務程序來挑選更新項目。 伺服器服務物件已變更，因此起始佇列指定為 JUPITER.INITIATION.QUEUE。 使用下列 MQSC 指令:

```
ALTER SERVICE(S1) +
STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

   註: 在重新啟動服務之前，執行中服務不會取得其服務定義的任何更新項目。
5. 伺服器服務程序會重新啟動，以便使用下列 MQSC 指令來挑選變更:

```
STOP SERVICE(S1)
```

後面接著:

```
START SERVICE(S1)
```

伺服器維修程序會重新啟動，並挑選在 第 107 頁的『4』中所做的變更。

註: 只有在服務定義中指定了 STOPCMD 引數時，才能使用 MQSC 指令 STOP SERVICE。

使用指令服務物件

此範例顯示如何定義指令服務物件，以在啟動或停止佇列管理程式時，啟動將項目寫入作業系統系統日誌的程式。

1. 使用下列 MQSC 指令定義指令服務物件:

```
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 和 Linux 系統提供的指令，用來寫入系統日誌。
+QMNAME+ 是代表佇列管理程式名稱的記號。

在佇列管理程式僅結束時使用指令服務物件

此範例顯示如何定義指令服務物件，以在僅停止佇列管理程式時，啟動將項目寫入作業系統系統日誌的程式。

1. 使用下列 MQSC 指令定義指令服務物件:

```
DEFINE SERVICE(S3) +
CONTROL(QMGR) +
SERVTYPE(COMMAND) +
STOPCMD('/usr/bin/logger') +
STOPARG('Queue manager +QMNAME+ stopping')
```

其中:

logger 是隨 IBM MQ 提供的範例程式，可將項目寫入作業系統的系統日誌。
+QMNAME+ 是代表佇列管理程式名稱的記號。

傳遞引數的相關資訊

此範例顯示如何定義伺服器服務物件，以在啟動佇列管理程式時啟動稱為 runserv 的程式。

此範例以 Windows 樣式路徑分隔字元撰寫。

要傳遞至起始程式的其中一個引數是包含空格的字串。 此引數需要以單一字串傳遞。 為了達到此目的，會使用雙引號來定義指令服務物件，如下列指令所示:

1. 使用下列 MQSC 指令定義伺服器服務物件:

```
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+ 是代表佇列管理程式名稱的記號。
"C:\Program Files\Tools\" 是包含空格的字串，將以單一字串傳遞。

自動啟動服務

此範例顯示如何定義伺服器服務物件，當佇列管理程式啟動時可用來自動啟動「觸發監視器」。

1. 使用下列 MQSC 指令定義伺服器服務物件:

```
DEFINE SERVICE(TRIG_MON_START) +
CONTROL(QMGR) +
SERVTYPE(SERVER) +
STARTCMD('runmqtrm') +
STARTARG('-m +QMNAME+ -q +IQNAME+')
```

其中:

+QMNAME+ 是代表佇列管理程式名稱的記號。
+IQNAME+ 是使用者在其中一個 service.env 檔案中定義的環境變數，代表起始佇列的名稱。

管理用於觸發的物件

IBM MQ 可讓您在符合佇列上的特定條件時自動啟動應用程式。 例如，當佇列上的訊息數達到指定數目時，您可能想要啟動應用程式。 此機能稱為 觸發。 您必須定義支援觸發的物件。

在 使用觸發程式來啟動 IBM MQ 應用程式中詳細說明的觸發作業。

定義用於觸發的應用程式佇列

應用程式佇列是應用程式透過 MQI 進行傳訊所使用的本端佇列。 觸發需要在應用程式佇列上定義一些佇列屬性。

觸發本身由 *Trigger* 屬性啟用 (MQSC 指令中的 TRIGGER)。 在此範例中，當本端佇列 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)
是所定義應用程式佇列的名稱。

PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
是程序定義的名稱，定義要由觸發監視器程式啟動的應用程式。

MAXMSGL (2000)
是佇列上訊息的長度上限。

DEFPSIST (YES)
指定依預設持續保存此佇列上的訊息。

INITQ (MOTOR.INS.INIT.QUEUE)
是佇列管理程式要放置觸發訊息的起始佇列名稱。

TRIGGER
是觸發程式屬性值。

TRIGTYPE (DEPTH)
指定當必要優先順序 (TRIGMPRI) 的訊息數達到 TRIGDPTH 中指定的數目時，產生觸發事件。

TRIGDPTH (100)
是產生觸發事件所需的訊息數目。

TRIGMPRI (5)
是佇列管理程式在決定是否產生觸發事件時要計算的訊息優先順序。 只會計算優先順序為 5 或以上的訊息。

定義起始佇列

當觸發事件發生時，佇列管理程式會將觸發訊息放置在應用程式佇列定義中指定的起始佇列上。 起始佇列沒有特殊設定，但您可以使用下列本端佇列 MOTOR.INS.INIT.QUEUE 用於指引:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
GET (ENABLED) +
NOSHARE +
NOTRIGGER +
MAXMSGL (2000) +
MAXDEPTH (1000)
```

定義處理程序

使用 DEFINE PROCESS 指令來建立程序定義。 程序定義會定義應用程式，以用來處理來自應用程式佇列的訊息。 應用程式佇列定義會命名要使用的處理程序，因此會將應用程式佇列與要用來處理其訊息的應用程式相關聯。 這是透過應用程式佇列 MOTOR.INSURANCE.QUEUE。 下列 MQSC 指令會定義必要的處理程序 MOTOR.INSURANCE.QUOTE.PROCESS，在下列範例中識別:

```
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
是程序定義的名稱。

DESCR ('Insurance request message processing')
說明與此定義相關的應用程式。 當您使用 DISPLAY PROCESS 指令時，會顯示此文字。 這可協助您識別處理程序執行的動作。 如果您在字串中使用空格，則必須以單引號括住字串。

APPLTYPE (UNIX)
是要啟動的應用程式類型。

APPLICID ('/u/admin/test/IRMP01')
是應用程式執行檔的名稱，指定為完整檔名。 在 Windows 系統中，一般 APPLICID 值會是 `c:\appl\test\irmp01.exe`。

USERDATA ('open, close, 235')
是使用者定義資料，可供應用程式使用。

顯示程序定義的屬性

請使用 DISPLAY PROCESS 指令來檢查定義的結果。 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)
```

您也可以使用 MQSC 指令 ALTER PROCESS 來變更現有的程序定義，並使用 DELETE PROCESS 指令來刪除程序定義。

在兩個系統之間使用 dmpmqmsg 公用程式

dmpmqmsg 公用程式 (舊稱為 **qload**) 會併入 8.0 版中的產品。 以前， **qload** 公用程式已提供為 SupportPac MO03。

概觀

dmpmqmsg 公用程式可讓您將佇列或其訊息的內容複製或移動至檔案。 此檔案可以視需要儲存，並在稍後的某個點使用，將訊息重新載入回佇列。

重要: 此檔案具有公用程式可理解的特定格式。 不過，檔案是人類可讀的，因此您可以在編輯器中更新它，然後再重新載入它。 如果您編輯檔案，則不得變更其格式。

可能的用途如下:

- 將佇列上的訊息儲存至檔案。 可能用於保存，稍後重新載入回佇列。
- 重新載入含有您先前儲存至檔案之訊息的佇列。
- 從佇列中移除舊訊息。
- 從儲存位置「重播」測試訊息，甚至在必要時維護訊息之間的正確時間。

小心: SupportPac MO03 使用 **-l** 參數來指定本端或用戶端連結。 **-l** 已取代為 **-c** 參數。

-P 現在用於字碼頁資訊，而非 **-c**。

如需指令及可用參數的進一步資訊，請參閱 dmpmqmsg 。

在 Linux 上使用 dmpmqmsg 公用程式的範例，使用 Windows 機器

您具有 Linux 機器上的佇列管理程式，該機器在佇列 (*Q1*) 中具有您要移至相同佇列管理程式中另一個佇列 (*Q2*) 的訊息。 您想要從 Windows 機器起始 **dmpmqmsg** 公用程式。

佇列 (*Q1*) 有四則訊息已使用範例 **amqsput** (本端佇列管理程式) 或 **amqsputc** (遠端佇列管理程式) 應用程式來新增。

在 Linux 機器上，您會看到:

```
display ql(Q1) CURDEPTH
        2 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q1)
TYPE(QLOCAL)
     CURDEPTH(4)
```

設定 MQSERVER 環境變數以指向 Linux 中的佇列管理程式。 例如:

```
set MQSERVER=SYSTEM.DEF.SVRCONN/TCP/veracruz.x.com(1414)
```

其中 *veracruz* 是機器的名稱。

執行 **dmpmqmsg** 公用程式以讀取佇列 *Q1*，並將輸出儲存在 `c:\temp\mqqload.txt` 中。

以遠端用戶端身分連接至在 Linux 主機及 MQSERVER 所建立埠中執行的佇列管理程式 *QM_VER*。 您可以使用下列屬性，作為遠端用戶端來達成連線: `-c`。

```
dmpmqmsg -m QM_VER -i Q1 -f c:\temp\mqqload.txt -c
Read      - Files:    0   Messages:       4   Bytes:           22
Written - Files:    1   Messages:       4   Bytes:           22
```

輸出檔 `c:\temp\mqqload.txt` 包含文字，使用 **dmpmqmsg** 公用程式瞭解的格式。

在 Windows 機器上，發出 **dmpmqmsg** 指令 (使用 `-o` 選項而非 `-i` 選項)，以從 Windows 機器上的檔案載入 Linux 機器上的佇列 (*Q2*):

```
dmpmqmsg -m QM_VER -o Q2 -f c:\temp\mqqload.txt -c
Read      - Files:    1   Messages:       4   Bytes:           22
Written - Files:    0   Messages:       4   Bytes:           22
```

在 Linux 機器上，請注意現在佇列中有四則已從檔案還原的訊息。

```
display ql(Q2) CURDEPTH
        6 : display ql(Q2) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q2)
TYPE(QLOCAL)
     CURDEPTH(4)
```

在 Linux 機器上，

從原始佇列中刪除訊息。

```
clear qlocal(Q1)
        4 : clear qlocal(Q1)
AMQ8022: IBM MQ queue cleared.
```

確認原始佇列上沒有其他訊息:

```
display ql(Q1) CURDEPTH
        5 : display ql(Q1) CURDEPTH
AMQ8409: Display Queue details.
     QUEUE(Q1)
TYPE(QLOCAL)
     CURDEPTH(0)
```

如需指令及其參數的說明，請參閱 dmpmqmsg。

相關概念

第 112 頁的『使用 dmpmqmsg 公用程式的範例』

您可以使用 **dmpmqmsg** 公用程式 (舊稱為 **qload**) 的簡單方式。 此公用程式已併入 8.0 版中的產品。

使用 dmpmqmsg 公用程式的範例

您可以使用 **dmpmqmsg** 公用程式 (舊稱為 **qload**) 的簡單方式。 此公用程式已併入 8.0 版中的產品。

先前 **qload** 公用程式是以 SupportPac MO03 提供。

將佇列卸載至檔案

在指令行上使用下列選項，將佇列上的訊息儲存至檔案:

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile
```

此指令會從佇列取得訊息副本，並將它們儲存在指定的檔案中。

將佇列卸載至一系列檔案

您可以在檔名中使用 insert 字元，將佇列卸載至一系列檔案。 在此模式中，每一則訊息都會寫入新檔案:

```
dmpmqmsg -m QM1 -i Q1 -f c:\myfile%n
```

此指令會將佇列卸載至檔案、 myfile1、 myfile2、 myfile3 等。

從檔案載入佇列

若要使用您儲存在 第 112 頁的『將佇列卸載至檔案』中的訊息重新載入佇列，請在指令行上使用下列選項:

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

此指令會將佇列卸載至檔案、 myfile1、 myfile2、 myfile3 等。

從一系列檔案載入佇列

您可以在檔名中使用 insert 字元，從一系列檔案載入佇列。 在此模式中，每一則訊息都會寫入新檔案:

```
dmpmqmsg -m QM1 -o Q1 -f c:\myfile%n
```

此指令會將佇列載入至檔案、 myfile1、 myfile2、 myfile3 等。

將訊息從一個佇列複製到另一個佇列

將 第 112 頁的『將佇列卸載至檔案』中的 file 參數取代為另一個佇列名稱，並使用下列選項:

```
dmpmqmsg -m QM1 -i Q1 -o Q2
```

此指令容許將來自一個佇列的訊息複製到另一個佇列。

將前 100 則訊息從一個佇列複製到另一個佇列

使用前一個範例中的指令，並新增 -r#100 選項:

```
dmpmqmsg -m QM1 -i Q1 -o Q2 -r#100
```

將訊息從一個佇列移至另一個佇列

第 113 頁的『從檔案載入佇列』上的變異。 請注意使用僅瀏覽佇列的 **-i** (小寫) 與破壞性從佇列取得的 **-I** (大寫) 之間的區別:

```
dmpmqmsg -m QM1 -I Q1 -o Q2
```

將超過一天的訊息從一個佇列移至另一個佇列

此範例顯示使用年齡選擇。 可以選取早於、早於或在經歷時間範圍內的訊息。

```
dmpmqmsg -m QM1 -I Q1 -o Q2 -T1440
```

顯示目前在佇列上的訊息經歷時間

在指令行上使用下列選項:

```
dmpmqmsg -m QM1 -i Q1 -f stdout -dT
```

使用訊息檔案

從佇列卸載訊息之後 (如 第 112 頁的『將佇列卸載至檔案』)，您可能想要編輯檔案。

您也可能想要變更檔案的格式，以使用您在卸載佇列時未指定的其中一個顯示選項。

即使在卸載佇列之後，您也可以使用 **dmpmqmsg** 公用程式，將檔案重新處理成所需的格式。 在指令行上使用下列選項。

```
dmpmqmsg -f c:\oldfile -f c:\newfile -dA
```

如需指令及其參數的說明，請參閱 dmpmqmsg。

管理遠端 IBM MQ 物件

本節告訴您如何使用 MQSC 指令管理遠端佇列管理程式上的 IBM MQ 物件，以及如何使用遠端佇列物件來控制訊息及回覆訊息的目的地。

本節說明:

- 第 114 頁的『通道、叢集及遠端佇列作業』
- 第 115 頁的『從本端佇列管理程式進行遠端管理』
- 第 121 頁的『建立遠端佇列的本端定義』
- 第 124 頁的『使用遠端佇列定義作為別名』
- 第 125 頁的『資料轉換』

通道、叢集及遠端佇列作業

佇列管理程式會傳送訊息，並在必要時接收回應，以與另一個佇列管理程式進行通訊。 接收端佇列管理程式可以是:

- 在同一部機器上
- 在相同位置的另一部機器上 (甚至在世界另一端)
- 在與本端佇列管理程式相同的平台上執行
- 在 IBM MQ 支援的另一個平台上執行

這些訊息可能源自:

- 使用者撰寫的應用程式，將資料從一個節點傳送至另一個節點
- 使用 PCF 指令或 MQAI 的使用者撰寫管理應用程式
- IBM MQ 檔案總管。
- 正在傳送佇列管理程式:
  - 將檢測事件訊息傳送至另一個佇列管理程式
  - 以間接模式 (其中指令在另一個佇列管理程式上執行) 從 `runmqsc` 指令發出的 MQSC 指令

在訊息可以傳送至遠端佇列管理程式之前，本端佇列管理程式需要有偵測訊息到達並傳輸訊息的機制，其中包含:

- 至少一個通道
- 傳輸佇列
- 通道起始程式

若要讓遠端佇列管理程式接收訊息，則需要接聽器。

通道是兩個佇列管理程式之間的單向通訊鏈結，可以在遠端佇列管理程式上傳送任何數目佇列的訊息。

通道的每一端都有個別的定義。 例如，如果一端是傳送端或伺服器，則另一端必須是接收端或要求端。 簡式通道由本端佇列管理程式端的 傳送端通道定義 和遠端佇列管理程式端的 接收端通道定義 組成。 這兩個定義必須具有相同的名稱，且一起構成單一訊息通道。

如果您想要遠端佇列管理程式回應本端佇列管理程式所傳送的訊息，請設定第二個通道，將回應傳回本端佇列管理程式。

請使用 MQSC 指令 DEFINE CHANNEL 來定義通道。 在本節中，除非另有指定，否則與通道相關的範例會使用預設通道屬性。

通道的每一端都有一個訊息通道代理程式 (MCA) ，可控制訊息的傳送及接收。 MCA 會從傳輸佇列中取得訊息，並將它們放置在佇列管理程式之間的通訊鏈結上。

傳輸佇列是特殊化本端佇列，在 MCA 挑選訊息並將它們傳送至遠端佇列管理程式之前，暫時保留訊息。 您可以在 遠端佇列定義上指定傳輸佇列的名稱。

您可以容許 MCA 使用多個執行緒來傳送訊息。 此處理程序稱為 管線化。 管線化可讓 MCA 更有效率地傳送訊息，改善通道效能。 如需如何配置通道以使用管線化的詳細資料，請參閱 通道屬性 。

第 116 頁的『準備通道及傳輸佇列以進行遠端管理』 告訴您如何使用這些定義來設定遠端管理。

如需一般設定分散式佇列的相關資訊，請參閱 分散式佇列元件。

使用叢集進行遠端管理

在使用分散式佇列的 IBM MQ 網路中，每個佇列管理程式都是獨立的。 如果一個佇列管理程式需要將訊息傳送至另一個佇列管理程式，它必須定義傳輸佇列、遠端佇列管理程式的通道，以及要傳送訊息的每個佇列的遠端佇列定義。

叢集 是佇列管理程式的群組，其設定方式可讓佇列管理程式透過單一網路直接彼此通訊，而不需要複雜的傳輸佇列、通道及佇列定義。 叢集可以輕鬆設定，且通常包含以某種方式邏輯相關且需要共用資料或應用程式的佇列管理程式。 即使最小叢集也可降低系統管理成本。

與建立傳統分散式佇列環境相比，在叢集中建立佇列管理程式網路所涉及的定義較少。 使用較少的定義，您可以更快速且輕鬆地設定或變更網路，並減少在定義中發生錯誤的風險。

若要設定叢集，每一個佇列管理程式需要一個叢集傳送端 (CLUSSDR) 及一個叢集接收端 (CLUSRCVR) 定義。 您不需要任何傳輸佇列定義或遠端佇列定義。 在叢集內使用遠端管理的原則相同，但定義本身已大幅簡化。

從本端佇列管理程式進行遠端管理

本節告訴您如何使用 MQSC 及 PCF 指令從本端佇列管理程式管理遠端佇列管理程式。

對於 MQSC 和 PCF 指令來說，準備佇列和通道基本上是相同的。 在本節中，範例顯示 MQSC 指令，因為它們較容易瞭解。 如需使用 PCF 指令撰寫管理程式的相關資訊，請參閱 第 10 頁的『使用可程式指令格式』。

您以互動方式或從包含指令的文字檔，將 MQSC 指令傳送至遠端佇列管理程式。 遠端佇列管理程式可能位於相同機器上，或更通常位於不同機器上。 您可以在其他 IBM MQ 環境 (包括 UNIX 和 Linux 系統、Windows 系統、 IBM i 及 z/OS) 中遠端管理佇列管理程式。

若要實作遠端管理，您必須建立特定的物件。 除非您有特殊化需求，否則預設值 (例如，訊息長度上限) 已足夠。

準備佇列管理程式以進行遠端管理

如何使用 MQSC 指令來準備佇列管理程式進行遠端管理。

第 116 頁的圖 17 顯示使用 **runmqsc** 指令進行遠端管理所需的佇列管理程式及通道的配置。 物件 `source.queue.manager` 是來源佇列管理程式，您可以從中發出 MQSC 指令，並將這些指令的結果 (操作員訊息) 傳回至其中。 物件 `target.queue.manager` 是目標佇列管理程式的名稱，它會處理指令並產生任何操作員訊息。

註: 如果您搭配使用 **runmqsc** 與 -w 選項，則 `source.queue.manager` **必須** 是預設佇列管理程式。 如需建立佇列管理程式的進一步資訊，請參閱 crtmqm。

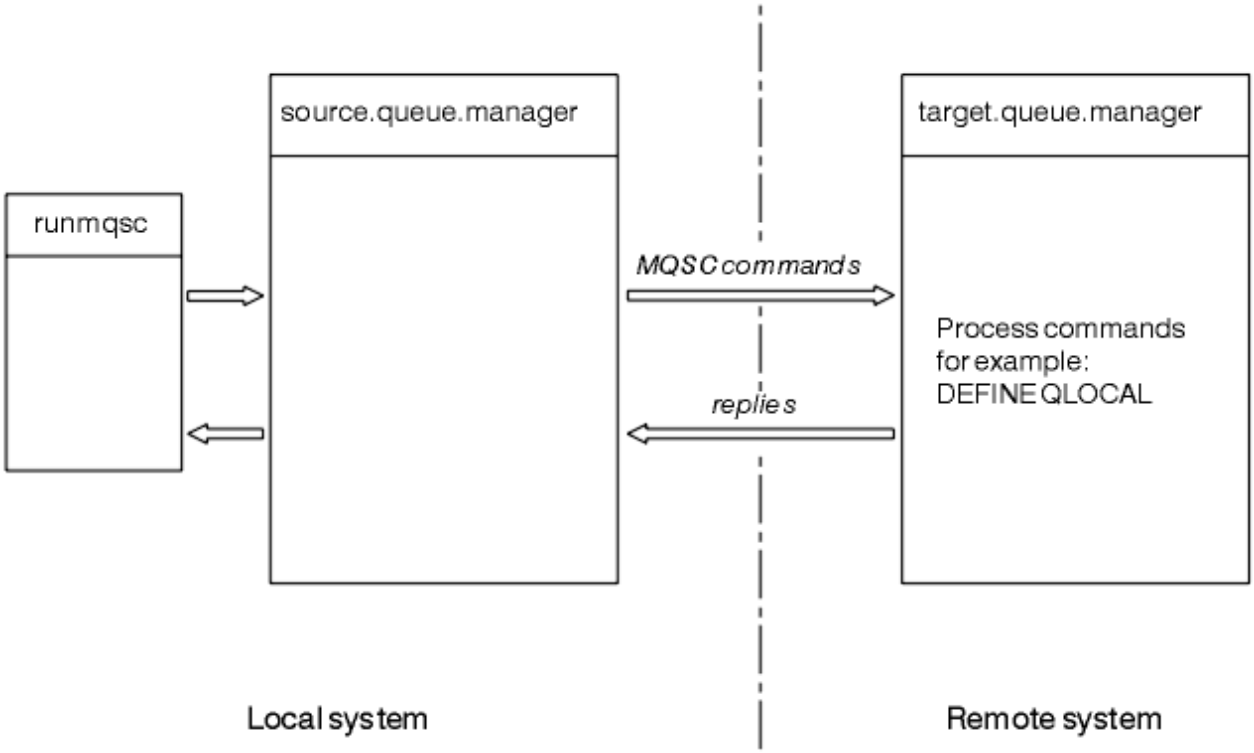


圖 *17:* 使用 *MQSC* 指令進行遠端管理

在兩個系統上，如果您尚未這樣做:

- 使用 `crtmqm` 指令，建立佇列管理程式及預設物件。
- 使用 `strmqm` 指令啟動佇列管理程式。

在目標佇列管理程式上:

- 指令佇列 SYSTEM.ADMIN.COMMAND.QUEUE，必須存在。 依預設，當建立佇列管理程式時，會建立此佇列。

您必須在本端或透過 Telnet 之類的網路機能來執行這些指令。

準備通道及傳輸佇列以進行遠端管理

如何使用 MQSC 指令來準備通道及傳輸佇列，以進行遠端管理。

若要從遠端執行 MQSC 指令，請設定兩個通道 (每個方向一個通道) 及其相關聯的傳輸佇列。 此範例假設您使用 TCP/IP 作為傳輸類型，且您知道涉及的 TCP/IP 位址。

通道 `source.to.target` 用於將 MQSC 指令從來源佇列管理程式傳送至目標佇列管理程式。 其傳送端位於 `source.queue.manager`，接收端位於 `target.queue.manager`。 通道 `target.to.source` 用於傳回來自指令的輸出，以及產生至來源佇列管理程式的任何操作員訊息。 您也必須為每一個通道定義傳輸佇列。 此佇列是提供接收端佇列管理程式名稱的本端佇列。 除非您使用佇列管理程式別名，否則 XMITQ 名稱必須符合遠端佇列管理程式名稱，遠端管理才能運作。 第 117 頁的圖 18 彙總此配置。

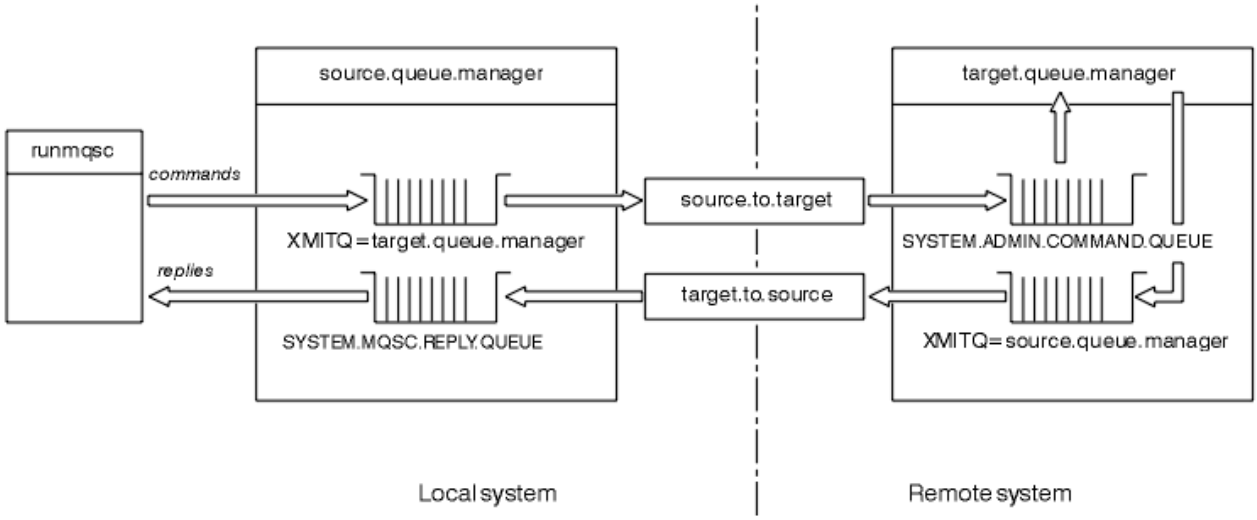


圖 *18:* 設定遠端管理的通道及佇列

如需設定通道的相關資訊，請參閱 配置分散式佇列作業 。

定義通道、接聽器及傳輸佇列

在來源佇列管理程式 (`source.queue.manager`) 上，發出下列 MQSC 指令以定義通道、接聽器及傳輸佇列:

1. 在來源佇列管理程式中定義傳送端通道:

```
DEFINE CHANNEL ('source.to.target') +
CHLTYPE(SDR) +
CONNAME (RHX5498) +
XMITQ ('target.queue.manager') +
TRPTYPE(TCP)
```

2. 在來源佇列管理程式中定義接收端通道:

```
DEFINE CHANNEL ('target.to.source') +
CHLTYPE(RCVR) +
TRPTYPE(TCP)
```

3. 在來源佇列管理程式上定義接聽器:

```
DEFINE LISTENER ('source.queue.manager') +
TRPTYPE (TCP)
```

4. 在來源佇列管理程式上定義傳輸佇列:

```
DEFINE QLOCAL ('target.queue.manager') +
USAGE (XMITQ)
```

在目標佇列管理程式 (`target.queue.manager`) 上發出下列指令，以建立通道、接聽器及傳輸佇列:

1. 在目標佇列管理程式上定義傳送端通道:

```
DEFINE CHANNEL ('target.to.source') +
CHLTYPE(SDR) +
CONNAME (RHX7721) +
XMITQ ('source.queue.manager') +
TRPTYPE(TCP)
```

2. 在目標佇列管理程式上定義接收端通道:

```
DEFINE CHANNEL ('source.to.target') +
CHLTYPE(RCVR) +
TRPTYPE(TCP)
```

3. 在目標佇列管理程式上定義接聽器:

```
DEFINE LISTENER ('target.queue.manager') +
TRPTYPE (TCP)
```

4. 在目標佇列管理程式上定義傳輸佇列:

```
DEFINE QLOCAL ('source.queue.manager') +
USAGE (XMITQ)
```

註: 在傳送端通道定義中指定給 CONNAME 屬性的 TCP/IP 連線名稱僅供說明。 這是位於連線 另一端 的機器網路名稱。 請使用適合您網路的值。

啟動接聽器及通道

如何使用 MQSC 指令來啟動接聽器及通道。

使用下列 MQSC 指令來啟動這兩個接聽器:

1. 透過發出下列 MQSC 指令，在來源佇列管理程式 `source.queue.manager` 上啟動接聽器:

```
START LISTENER ('source.queue.manager')
```

2. 透過發出下列 MQSC 指令，在目標佇列管理程式 `target.queue.manager` 上啟動接聽器:

```
START LISTENER ('target.queue.manager')
```

使用下列 MQSC 指令來啟動兩個傳送端通道:

1. 發出下列 MQSC 指令，在來源佇列管理程式 `source.queue.manager` 上啟動傳送端通道:

```
START CHANNEL ('source.to.target')
```

2. 發出下列 MQSC 指令，在目標佇列管理程式 `target.queue.manager` 上啟動傳送端通道:

```
START CHANNEL ('target.to.source')
```

通道自動定義

您可以使用 MQSC 指令 ALTER QMGR (或 PCF 指令「變更佇列管理程式」) 來更新佇列管理程式物件，以啟用接收端及伺服器連線定義的自動定義。

如果 IBM MQ 收到入埠連接要求，且找不到適當的接收端或伺服器連線通道，它會自動建立通道。 自動定義是根據 IBM MQ 提供的兩個預設定義: SYSTEM.AUTO.RECEIVER 和 SYSTEM.AUTO.SVRCONN。

如需自動建立通道定義的相關資訊，請參閱 準備通道。 如需自動定義叢集的通道的相關資訊，請參閱 使用自動定義的通道。

管理用於遠端管理的指令伺服器

如何啟動、停止及顯示指令伺服器的狀態。 對於涉及 PCF 指令、MQAI 以及遠端管理的所有管理而言，指令伺服器是必要的。

每一個佇列管理程式都可以有相關聯的指令伺服器。 指令伺服器會處理來自遠端佇列管理程式的任何送入指令，或來自應用程式的 PCF 指令。 它會將指令呈現給佇列管理程式進行處理，並根據指令的原點傳回完成碼或操作員訊息。

註: 若為遠端管理，請確定目標佇列管理程式正在執行中。 否則，包含指令的訊息無法離開從中發出指令的佇列管理程式。 相反地，這些訊息會在提供遠端佇列管理程式的本端傳輸佇列中排入佇列。 請避免此狀況。

有個別控制指令可用來啟動及停止指令伺服器。 如果指令伺服器正在執行中，則 IBM MQ for Windows 或 IBM MQ for Linux (x86 及 x86-64 平台) 的使用者可以使用 IBM MQ Explorer 來執行下列各節中說明的作業。 如需相關資訊，請參閱第 54 頁的『使用 MQ Explorer 進行管理』。

啟動指令伺服器

視佇列管理程式屬性 *SCMDSERV* 的值而定，指令伺服器會在佇列管理程式啟動時自動啟動，或必須手動啟動。 可以使用 MQSC 指令 ALTER QMGR 指定參數 SCMDSERV 來變更佇列管理程式屬性的值。 依預設，指令伺服器會自動啟動。

如果 *SCMDSERV* 設為 MANUAL，請使用下列指令啟動指令伺服器:

```
strmqcsv saturn.queue.manager
```

其中 `saturn.queue.manager` 是正在啟動指令伺服器的佇列管理程式。

顯示指令伺服器的狀態

對於遠端管理，請確定目標佇列管理程式上的指令伺服器正在執行中。 如果它不在執行中，則無法處理遠端指令。 任何包含指令的訊息都會排入目標佇列管理程式的指令佇列。

若要顯示佇列管理程式的指令伺服器狀態，請發出下列 MQSC 指令:

```
DISPLAY QMSTATUS CMDSERV
```

停止指令伺服器

若要結束前一個範例所啟動的指令伺服器，請使用下列指令:

```
endmqcsv saturn.queue.manager
```

您可以使用兩種方式來停止指令伺服器:

- 若為受管制的停止，請搭配使用 `endmqcsv` 指令與 -c 旗標，這是預設值。
- 如需立即停止，請搭配使用 `endmqcsv` 指令與 -i 旗標。

註: 停止佇列管理程式也會結束與其相關聯的指令伺服器。

在遠端佇列管理程式上發出 MQSC 指令

您可以使用特定形式的 **runmqsc** 指令，在遠端佇列管理程式上執行 MQSC 指令。

如果要從遠端處理 MQSC 指令，指令伺服器 **必須** 在目標佇列管理程式上執行。 (這在來源佇列管理程式上不是必要的)。 如需如何在佇列管理程式上啟動指令伺服器的相關資訊，請參閱 第 118 頁的『管理用於遠端管理的指令伺服器』。

然後，您可以在來源佇列管理程式上鍵入下列指令，以間接模式以互動方式執行 MQSC 指令:

```
runmqsc -w 30 target.queue.manager
```

此形式的 `runmqsc` 指令 (含 -w 旗標) 會以間接模式執行 MQSC 指令，其中指令會 (以修改過的格式) 放置在指令伺服器輸入佇列上，並依序執行。

當您鍵入 MQSC 指令時，會將它重新導向至遠端佇列管理程式，在此情況下為 `target.queue.manager`。 逾時設為 30 秒; 如果在 30 秒內未收到回覆，則會在本端 (來源) 佇列管理程式上產生下列訊息:

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

當您停止發出 MQSC 指令時，本端佇列管理程式會顯示任何已到達的逾時回應，並捨棄任何進一步的回應。

來源佇列管理程式預設為預設本端佇列管理程式。 如果您在 **runmqsc** 指令中指定 -m *LocalQmgr* 名稱 選項，則可以透過任何本端佇列管理程式來引導要發出的指令。

在間接模式中，您也可以在遠端佇列管理程式上執行 MQSC 指令檔。 例如:

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

其中 `mycomds.in` 是包含 MQSC 指令的檔案， `report.out` 是報告檔。

建議遠端發出指令的方法

當您在遠端佇列管理程式上發出指令時，請考慮使用下列方法:

1. 將要在遠端系統上執行的 MQSC 指令放置在指令檔中。
2. 在 `runmqsc` 指令上指定 -v 旗標，以在本端驗證 MQSC 指令。

   您無法使用 `runmqsc` 來驗證另一個佇列管理程式上的 MQSC 指令。
3. 請檢查指令檔在本端執行，且沒有錯誤。
4. 在遠端系統上執行指令檔。

如果您在遠端使用 MQSC 指令時發生問題

如果您在遠端執行 MQSC 指令時遇到困難，請確定您有:

- 已在目標佇列管理程式上啟動指令伺服器。
- 已定義有效的傳輸佇列。
- 定義這兩個訊息通道的兩端:
  - 傳送指令的通道。
  - 要傳回回覆的通道。
- 在通道定義中指定了正確的連線名稱 (CONNAME)。
- 在您啟動訊息通道之前，已啟動接聽器。
- 已檢查斷線間隔是否尚未過期，例如，如果通道已啟動，但在一段時間之後關閉。 如果您手動啟動通道，這尤其重要。
- 從來源佇列管理程式傳送對目標佇列管理程式沒有意義的要求 (例如，包含遠端佇列管理程式不支援之參數的要求)。

另請參閱 第 73 頁的『解決 MQSC 指令的問題』。

在 *z/OS* 上使用佇列管理程式

您可以在本手冊說明的平台上，從佇列管理程式向 z/OS 佇列管理程式發出 MQSC 指令。 不過，如果要這麼做，您必須修改 `runmqsc` 指令和傳送端的通道定義。

特別是您在來源節點上的 `runmqsc` 指令中新增 -x 旗標，以指定目標佇列管理程式在 z/OS: 下執行

```
runmqsc -w 30 -x target.queue.manager
```

建立遠端佇列的本端定義

遠端佇列的本端定義是本端佇列管理程式上參照遠端佇列管理程式上佇列的定義。

您不需要從本端位置定義遠端佇列，但這樣做的好處是應用程式可以透過其本端定義的名稱來參照遠端佇列，而不必指定由遠端佇列所在的佇列管理程式 ID 所限定的名稱。

瞭解遠端佇列的本端定義如何運作

應用程式連接至本端佇列管理程式，然後發出 MQOPEN 呼叫。 在開放式呼叫中，指定的佇列名稱是本端佇列管理程式上遠端佇列定義的名稱。 遠端佇列定義提供目標佇列的名稱、目標佇列管理程式，以及選擇性地提供傳輸佇列。 若要將訊息放置在遠端佇列上，應用程式會發出 MQPUT 呼叫，並指定從 MQOPEN 呼叫傳回的控點。 佇列管理程式會在訊息開頭的傳輸標頭中使用遠端佇列名稱和遠端佇列管理程式名稱。 此資訊是用來將訊息遞送至網路中正確的目的地。

作為管理者，您可以透過變更遠端佇列定義來控制訊息的目的地。

下列範例顯示應用程式如何將訊息放置在遠端佇列管理程式所擁有的佇列上。 應用程式會連接至佇列管理程式，例如 saturn.queue.manager。 目標佇列由另一個佇列管理程式所擁有。

在 MQOPEN 呼叫上，應用程式會指定下列欄位:

欄位值	說明
ObjectName CYAN.REMOTE.QUEUE	指定遠端佇列物件的本端名稱。 這會定義目標佇列及目標佇列管理程式。
ObjectType (佇列)	將此物件識別為佇列。
ObjectQmgrName 空白或 saturn.queue.manager	這是選用欄位。 如果空白，則會採用本端佇列管理程式的名稱。 (這是遠端佇列定義所在的佇列管理程式。)

之後，應用程式會發出 MQPUT 呼叫，將訊息放到這個佇列中。

在本端佇列管理程式上，您可以使用下列 MQSC 指令來建立遠端佇列的本端定義:

```
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)
指定遠端佇列物件的本端名稱。 這是連接至此佇列管理程式的應用程式必須在 MQOPEN 呼叫中指定的名稱，以在遠端佇列管理程式 jupiter.queue.manager 上開啟佇列 AUTOMOBILE.INSURANCE.QUOTE.QUEUE 。

DESCR ('Queue for auto insurance requests from the branches')
提供說明使用佇列的其他文字。

RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)
指定遠端佇列管理程式上目標佇列的名稱。 這是指定佇列名稱 CYAN.REMOTE.QUEUE。 佇列 AUTOMOBILE.INSURANCE.QUOTE.QUEUE 必須定義為遠端佇列管理程式上的本端佇列。

RQMNAME (jupiter.queue.manager)
指定擁有目標佇列 AUTOMOBILE.INSURANCE.QUOTE.QUEUE。

XMITQ (INQUOTE.XMIT.QUEUE)
指定傳輸佇列的名稱。 這是選用項目; 如果未指定傳輸佇列的名稱，則會使用與遠端佇列管理程式同名的佇列。

在任一情況下，必須使用 *Usage* 屬性將適當的傳輸佇列定義為本端佇列，並指定它是 MQSC 指令中的傳輸佇列 (USAGE (XMITQ))。

將訊息放置在遠端佇列上的替代方式

使用遠端佇列的本端定義不是將訊息放置在遠端佇列上的唯一方法。 應用程式可以在 MQOPEN 呼叫中指定完整佇列名稱 (包括遠端佇列管理程式名稱)。 在此情況下，您不需要遠端佇列的本端定義。 不過，這表示在執行時期，應用程式必須知道或有權存取遠端佇列管理程式的名稱。

搭配使用其他指令與遠端佇列

您可以使用 MQSC 指令來顯示或變更遠端佇列物件的屬性，也可以刪除遠端佇列物件。 例如：

- 顯示遠端佇列的屬性:

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- 變更遠端佇列以啟用放置。 這不會影響目標佇列，只會影響指定此遠端佇列的應用程式:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- 刪除此遠端佇列。 這不會影響目標佇列，只會影響其本端定義:

```
DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

註: 當您刪除遠端佇列時，只會刪除遠端佇列的本端表示法。 您不刪除遠端佇列本身或其中的任何訊息。

定義傳輸佇列

傳輸佇列是當佇列管理程式透過訊息通道將訊息轉遞至遠端佇列管理程式時所使用的本端佇列。

通道提供指向遠端佇列管理程式的單向鏈結。 訊息會在傳輸佇列中排入佇列，直到通道可以接受它們為止。 當您定義通道時，必須在訊息通道傳送端指定傳輸佇列名稱。

MQSC 指令屬性 USAGE 定義佇列是傳輸佇列還是一般佇列。

預設傳輸佇列

當佇列管理程式將訊息傳送至遠端佇列管理程式時，它會使用下列順序來識別傳輸佇列:

1. 在遠端佇列本端定義的 XMITQ 屬性上命名的傳輸佇列。
2. 與目標佇列管理程式同名的傳輸佇列。 (此值是遠端佇列之本端定義的 XMITQ 預設值。)
3. 在本端佇列管理程式的 DEFXMITQ 屬性上指定的傳輸佇列。

例如，下列 MQSC 指令會針對傳送至 `target.queue.manager` 的訊息，在 `source.queue.manager` 上建立預設傳輸佇列:

```
DEFINE QLOCAL ('target.queue.manager') +
DESCR ('Default transmission queue for target qm') +
USAGE (XMITQ)
```

應用程式可以將訊息直接放置在傳輸佇列上，或透過遠端佇列定義間接放置。 另請參閱 第 121 頁的『建立遠端佇列的本端定義』。

檢查分散式網路的非同步指令是否已完成

在分散式網路中使用時，許多指令都是非同步的。 視指令及發出時的網路狀態而定，可能需要大量時間才能完成。 佇列管理程式在完成時不會發出訊息，因此您需要其他方法來檢查指令是否已完成。

關於這項作業

幾乎您對叢集所做的任何配置變更都可能非同步完成。 這是因為在叢集內運作的內部管理和更新週期。 對於發佈/訂閱階層，任何會影響訂閱的配置變更可能會非同步完成。 從指令名稱來看，這並不總是顯而易見的。

下列 MQSC 指令可能全部非同步完成。 其中每一個指令都有 PCF 對等項目，且大部分也可以從 MQ Explorer 內取得。 在沒有工作量的小型網路上執行時，這些指令通常會在幾秒內完成。 然而，大型且忙碌的網路卻不是如此。 此外， **REFRESH CLUSTER** 指令可能需要更長的時間，尤其是同時在多個佇列管理程式上發出時。

若要確信這些指令已完成，請檢查預期的物件是否存在於遠端佇列管理程式上。

程序

- ALTER QMGR

  對於 ALTER QMGR PARENT 指令，請使用 DISPLAY PUBSUB TYPE(PARENT) ALL 來追蹤所要求母項關係的狀態。

  若為 ALTER QMGR REPOS 及 ALTER QMGR REPOSNL 指令，請使用 DISPLAY CLUSQMGR QMTYPE 來確認完成。

- DEFINE CHANNEL、 ALTER CHANNEL 及 DELETE CHANNEL

  對於表格 ALTER CHANNEL 參數中列出的所有參數，請使用 DISPLAY CLUSQMGR 指令來監視變更何時延伸到叢集。

- DEFINE NAMELIST、 ALTER NAMELIST 及 DELETE NAMELIST。

  如果您在 **QMgr** 物件的 **CLUSNL** 屬性上使用 **NAMELIST** ，則佇列或叢集通道可能會影響該物件。 適當監視受影響的物件。

  SYSTEM.QPUBSUB.QUEUE.NAMELIST 的變更可能會影響在發佈/訂閱階層中建立或取消 Proxy 訂閱。 請使用 DISPLAY SUB SUBTYPE(PROXY) 指令來監視此情況。

- DEFINE 佇列、 ALTER 佇列及 DELETE 佇列。

  對於表格 DISPLAY QUEUE 指令可傳回的參數中列出的所有參數，請使用 DISPLAY QCLUSTER 指令來監視變更何時延伸到叢集。

- DEFINE SUB 和 DELETE SUB

  當您在主題字串上定義第一個訂閱時，您可以在發佈/訂閱階層或發佈/訂閱叢集中建立 Proxy 訂閱。 同樣地，當您刪除主題字串的前次訂閱時，您可以取消發佈/訂閱階層或發佈/訂閱叢集中的 Proxy 訂閱。

  若要檢查定義或刪除訂閱的指令是否已完成，請檢查預期的 Proxy 訂閱是否存在於分散式網路中的其他佇列管理程式上。 如果您在叢集中使用 直接遞送 ，請檢查叢集中其他局部儲存庫上是否存在預期的 Proxy 訂閱。 如果您在叢集中使用 主題主機遞送 ，請檢查相符主題主機上是否存在預期的 Proxy 訂閱。 請使用下列 MQSC 指令:

  ```
  DISPLAY SUB(*) SUBTYPE(PROXY)
  ```

  當在叢集或階層中發出 MQI 呼叫時，對下列對等訂閱及取消訂閱 MQI 呼叫使用相同的檢查:

  – 使用 MQSUB 進行訂閱。

  – 搭配使用 MQCLOSE 與 MQCO_REMOVE_SUB 來取消訂閱。

- DEFINE TOPIC、 ALTER TOPIC 及 DELETE TOPIC

  如果要檢查定義、變更或刪除叢集主題的指令已完成，請顯示叢集中其他局部儲存庫中的主題 (如果您使用 直接遞送) 或在其他主題主機上 (如果您使用 主題主機遞送)。

  對於表格 DISPLAY TOPIC 指令可傳回的參數中列出的所有參數，請使用 DISPLAY TCLUSTER 指令來監視變更何時延伸到叢集。

  註:

- **CLUSTER** 參數可能會影響發佈/訂閱叢集中 Proxy 訂閱的建立或取消。
- **PROXYSUB** 和 **SUBSCOPE** 參數可能會影響在發佈/訂閱階層或發佈/訂閱叢集中建立或取消 Proxy 訂閱。
- 請使用 DISPLAY SUB SUBTYPE(PROXYSUB) 指令來監視此情況。

- 重新整理叢集

  如果您正在執行 **REFRESH CLUSTER** 指令，請輪詢叢集指令佇列深度。 在尋找物件之前，請等待它達到零，並保持為零。

  1. 請使用下列 MQSC 指令來檢查叢集指令佇列深度是否為零。

     ```
     DISPLAY QL(SYSTEM.CLUSTER.COMMAND.QUEUE) CURDEPTH
     ```

  2. 重複檢查，直到佇列深度達到零，並在後續檢查中保持零。

  REFRESH CLUSTER 指令會移除並重建物件，在大型配置中可能需要很長時間才能完成。 請參閱發佈/訂閱叢集的 REFRESH CLUSTER 注意事項。

- REFRESH QMGR 類型 (PROXYSUB)

  若要檢查 **REFRESH QMGR TYPE(PROXYSUB)** 指令是否已完成，請檢查 Proxy 訂閱是否已在分散式網路中的其他佇列管理程式上更正。 如果您在叢集中使用 直接遞送 ，請檢查叢集中其他局部儲存庫上的 Proxy 訂閱是否已更正。 如果您在叢集中使用 主題主機遞送 ，請檢查是否已在相符主題主機上更正預期的 Proxy 訂閱。 請使用下列 MQSC 指令:

  ```
  DISPLAY SUB(*) SUBTYPE(PROXYSUB)
  ```

- 重設叢集

  若要檢查 **RESET CLUSTER** 指令是否已完成，請使用 DISPLAY CLUSQMGR。

- RESET QMGR 類型 (PUBSUB)

  若要檢查 **RESET QMGR** 指令是否已完成，請使用 DISPLAY PUBSUB TYPE(PARENT|CHILD)。

  註: RESET QMGR 指令可能會導致取消發佈/訂閱階層或發佈/訂閱叢集中的 Proxy 訂閱。 請使用 DISPLAY SUB SUBTYPE(PROXYSUB) 指令來監視此情況。

- 您也可能想要監視在指令完成時趨向零佇列深度的其他系統佇列。

  例如，您可能想要監視 SYSTEM.INTER.QMGR.CONTROL 佇列及 SYSTEM.INTER.QMGR.FANREQ 佇列。 請參閱 監視叢集裡的 Proxy 訂閱資料流量 ，以及 在發佈/訂閱網路中平衡生產者和消費者。

下一步

如果這些檢查未確認非同步指令已完成，則可能發生錯誤。 若要調查，請先檢查發出指令之佇列管理程式的日誌，然後 (針對叢集) 檢查叢集完整儲存庫日誌。

相關資訊

z/OS z/OS 上 CLUSTER 指令的非同步行為

使用遠端佇列定義作為別名

除了在另一個佇列管理程式上尋找佇列之外，您還可以針對佇列管理程式別名及回覆目的地佇列別名使用遠端佇列的本端定義。 這兩種類型的別名都是透過遠端佇列的本端定義來解析。 您必須設定適當的通道，讓訊息到達其目的地。

佇列管理程式別名

別名是由訊息路徑上的佇列管理程式修改目標佇列管理程式 (如訊息中所指定) 的處理程序。佇列管理程式別名很重要，因為您可以使用它們來控制佇列管理程式網路內的訊息目的地。

您可以在控制點上變更佇列管理程式上的遠端佇列定義來執行此動作。 傳送端應用程式不知道指定的佇列管理程式名稱是別名。

如需佇列管理程式別名的相關資訊，請參閱 何謂別名?。

回覆目的地佇列別名

當應用程式在佇列上放置 要求訊息 時，可以選擇性地指定回覆目的地佇列的名稱。

如果處理訊息的應用程式擷取回覆目的地佇列的名稱，必要的話，它會知道傳送 回覆訊息的位置。

回覆目的地佇列別名是由訊息路徑上的佇列管理程式變更回覆目的地佇列 (如要求訊息中所指定) 的處理程序。 傳送端應用程式不知道指定的回覆目的地佇列名稱是別名。

回覆目的地佇列別名可讓您變更回覆目的地佇列的名稱，以及選擇性地變更其佇列管理程式。 這可讓您依序控制用於回覆訊息的路徑。

如需要求訊息、回覆訊息及回覆目的地佇列的相關資訊，請參閱 訊息類型 及 回覆目的地佇列及佇列管理程式。

如需回覆目的地佇列別名的相關資訊，請參閱 回覆目的地佇列別名及叢集。

資料轉換

IBM MQ 定義格式 (也稱為 內建格式) 的訊息資料 可以由佇列管理程式從一個編碼字集轉換為另一個編碼字集，前提是這兩個字集都與單一語言或一組類似語言相關。

例如，支援 ID 為 (CCSID) 850 和 500 的編碼字集之間的轉換，因為兩者都適用於西歐語言。

如需 EBCDIC 換行 (NL) 字元轉換為 ASCII 的相關資訊，請參閱 所有佇列管理程式。

支援的轉換定義在 資料轉換中。

當佇列管理程式無法轉換內建格式的訊息時

如果佇列管理程式的 CCSID 代表不同的國家語言群組，則無法自動轉換內建格式的訊息。 例如，不支援 CCSID 850 與 CCSID 1025 (這是使用斯拉夫語 Script 之語言的 EBCDIC 編碼字集) 之間的轉換，因為其中一個編碼字集中的許多字元無法以另一個編碼字集表示。 如果您具有以不同國家語言運作的佇列管理程式網路，且不支援部分編碼字集之間的資料轉換，則可以啟用預設轉換。 第 125 頁的『預設資料轉換』中說明預設資料轉換。

檔案 ccsid.tbl

檔案 ccsid.tbl 用於下列目的:

- 在 IBM MQ for Windows 中，它會記錄所有支援的字碼集。
- 在 AIX 和 HP-UX 平台上，作業系統會在內部保留支援的程式碼集。
- 對於所有其他 UNIX 和 Linux 平台，受支援的字碼集保留在 IBM MQ 提供的轉換表中。
- 它指定任何其他字碼集。 若要指定其他字碼集，您需要編輯 ccsid.tbl (檔案中提供如何執行此動作的指引)。
- 它指定任何預設資料轉換。

您可以更新記錄在 ccsid.tbl 中的資訊; 例如，如果未來版本的作業系統支援其他編碼字集，您可能想要這樣做。

在 IBM MQ for Windows 中，依預設 ccsid.tbl 位於 `C:\Program Files\IBM\WebSphere MQ\conv\table` 目錄中。

在 IBM MQ for UNIX 和 Linux 系統中， ccsid.tbl 位於 `/var/mqm/conv/table` 目錄中。

預設資料轉換

如果您在通常不支援資料轉換的兩部機器之間設定通道，則必須啟用預設資料轉換，通道才能運作。

若要啟用預設資料轉換，請編輯 ccsid.tbl 檔案，以指定預設 EBCDIC CCSID 及預設 ASCII CCSID。 檔案中包括如何執行此動作的指示。 您必須在將使用通道連接的所有機器上執行此動作。 重新啟動佇列管理程式，讓變更生效。

預設資料轉換處理程序如下:

- 如果不支援來源和目標 CCSID 之間的轉換，但來源和目標環境的 CCSID 都是 EBCDIC 或都是 ASCII，則會將字元資料傳遞至目標應用程式而不進行轉換。
- 如果一個 CCSID 代表 ASCII 編碼字集，而另一個代表 EBCDIC 編碼字集，則 IBM MQ 會使用 ccsid.tbl 中定義的預設資料轉換 CCSID 來轉換資料。

註: 嘗試限制將字元轉換為在指定給訊息的編碼字集及預設編碼字集中具有相同編碼值的字元。 如果您只使用對 IBM MQ 物件名稱有效的字元集 (如 命名 IBM MQ 物件 中所定義) 你一般會滿足這個要求。 在日本使用的 EBCDIC CCSID 290、930、1279 及 5026 發生異常狀況，其中小寫字元具有與其他 EBCDIC CCSID 不同的代碼。

以使用者定義格式轉換訊息

佇列管理程式無法將使用者定義格式的訊息從一個編碼字集轉換成另一個編碼字集。 如果您需要以使用者定義的格式轉換資料，則必須為每一種此類格式提供資料轉換結束程式。 請勿使用預設 CCSID 來轉換使用者定義格式的字元資料。 如需以使用者定義格式轉換資料及寫入資料轉換結束程式的相關資訊，請參閱 寫入資料轉換結束程式。

變更佇列管理程式 CCSID

當您已使用 ALTER QMGR 指令的 CCSID 屬性來變更佇列管理程式的 CCSID 時，請停止並重新啟動佇列管理程式，以確保所有執行中的應用程式 (包括指令伺服器及通道程式) 都已停止並重新啟動。

這是必要的，因為任何在變更佇列管理程式 CCSID 時執行的應用程式都會繼續使用現有的 CCSID。

管理 IBM MQ Telemetry

使用 MQ Explorer 或在指令行管理 IBM MQ Telemetry。 使用瀏覽器來配置遙測通道、控制遙測服務，以及監視連接至 IBM MQ 的 MQTT 用戶端。 使用 JAAS、SSL 和 IBM MQ 物件權限管理程式來配置 IBM MQ Telemetry 的安全。

使用 MQ Explorer 進行管理

使用瀏覽器來配置遙測通道、控制遙測服務，以及監視連接至 IBM MQ 的 MQTT 用戶端。 使用 JAAS、SSL 和 IBM MQ 物件權限管理程式來配置 IBM MQ Telemetry 的安全。

使用指令行管理

可以在指令行使用 IBM MQ MQSC 指令完全管理 IBM MQ Telemetry。

IBM MQ Telemetry 文件也有範例 Script，示範 IBM MQ Telemetry Transport v3 用戶端應用程式的基本用法。

在使用之前，請先閱讀並瞭解 Developing applications for IBM MQ Telemetry 小節中 IBM MQ Telemetry Transport 範例程式 中的範例。

相關資訊

IBM MQ Telemetry

MQXR 內容

在 Linux 及 AIX 上配置遙測的佇列管理程式

請遵循下列手動步驟，將佇列管理程式配置成執行 IBM MQ Telemetry。 您可以執行自動化程序，以使用 MQ Explorer 的 IBM MQ Telemetry 支援來設定更簡單的配置。

開始之前

1. 如需如何安裝 IBM MQ 及 IBM MQ Telemetry 特性的相關資訊，請參閱 安裝 IBM MQ Telemetry 。
2. 建立並啟動佇列管理程式。 在這項作業中，佇列管理程式稱為 *qMgr* 。
3. 在這項作業中，您可以配置遙測 (MQXR) 服務。 MQXR 內容設定儲存在平台專用內容檔中: `mqxr_unix.properties`。 您通常不需要直接編輯 MQXR 內容檔，因為幾乎所有設定都可以透過 MQSC 管理指令或「MQ 探險家」來配置。 如果您決定直接編輯檔案，請先停止佇列管理程式，再進行變更。 請參閱 MQXR 內容。

關於這項作業

MQ Explorer 的 IBM MQ Telemetry 支援包括精靈及範例指令程序 `sampleMQM`。 他們使用來賓使用者 ID 來設定起始配置; 請參閱 使用 MQ Explorer 驗證 IBM MQ Telemetry 的安裝 和 IBM MQ Telemetry Transport 範例程式。

請遵循此作業中的步驟，使用不同的授權架構來手動配置 IBM MQ Telemetry 。

程序

1. 在遙測範例目錄中開啟指令視窗。

   遙測範例目錄為 `/opt/mqm/mqxr/samples`。
2. 建立遙測傳輸佇列。

   ```
   echo "DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000)" | runmqsc qMgr
   ```

   第一次啟動遙測 (MQXR) 服務時，它會建立 `SYSTEM.MQTT.TRANSMIT.QUEUE`。

   它在此作業中手動建立，因為 `SYSTEM.MQTT.TRANSMIT.QUEUE` 必須在遙測 (MQXR) 服務啟動之前存在，才能授權存取它。
3. 設定預設傳輸佇列

   第一次啟動遙測 (MQXR) 服務時，它不會變更佇列管理程式使 `SYSTEM.MQTT.TRANSMIT.QUEUE` 成為預設傳輸佇列。

   若要使 `SYSTEM.MQTT.TRANSMIT.QUEUE` 成為預設傳輸佇列，請變更預設傳輸佇列內容。 使用 MQ Explorer 或下列範例中的指令來變更內容:

   ```
   echo "ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE')" | runmqsc qMgr
   ```

   變更預設傳輸佇列可能會干擾您現有的配置。 將預設傳輸佇列變更為 `SYSTEM.MQTT.TRANSMIT.QUEUE` 的原因是讓直接傳送訊息至 MQTT 用戶端更容易。 在不變更預設傳輸佇列的情況下，您必須為每個接收 MQ Explorer 訊息的用戶端新增遠端佇列定義; 請參閱 第 131 頁的『將訊息直接傳送至用戶端』。
4. 遵循 第 132 頁的『授權 MQTT 用戶端存取 IBM MQ 物件』 中的程序來建立一或多個使用者 ID。 使用者 ID 具有發佈、訂閱及傳送發佈至 MQTT 用戶端的權限。
5. 安裝遙測 (MQXR) 服務

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   另請參閱 第 128 頁的圖 19 中的程式碼範例。
6. 啟動服務

   ```
   echo "START SERVICE(SYSTEM.MQXR.SERVICE)" | runmqsc qMgr
   ```

   啟動佇列管理程式時，會自動啟動遙測 (MQXR) 服務。

它在此作業中手動啟動，因為佇列管理程式已在執行中。

7. 使用 MQ Explorer，配置遙測通道以接受來自 MQTT 用戶端的連線。

   遙測通道必須配置為其身分是步驟 4 中定義的其中一個使用者 ID。

   另請參閱 DEFINE CHANNEL (MQTT)。

8. 執行範例用戶端來驗證配置。

   若要讓範例用戶端使用遙測通道，該通道必須授權用戶端發佈、訂閱及接收發佈。 依預設，範例用戶端會連接至埠 1883 上的遙測通道。 另請參閱 IBM MQ Telemetry Transport 程式範例。

範例

第 128 頁的圖 19 顯示在 Linux 上手動建立 SYSTEM.MQXR.SERVICE 的 **runmqsc** 指令。

```
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')
```

圖 *19: installMQXRService_unix.mqsc*

在 Windows 上配置遙測的佇列管理程式

請遵循下列手動步驟，將佇列管理程式配置成執行 IBM MQ Telemetry。 您可以執行自動化程序，以使用 MQ Explorer 的 IBM MQ Telemetry 支援來設定更簡單的配置。

開始之前

1. 如需如何安裝 IBM MQ 及 IBM MQ Telemetry 特性的相關資訊，請參閱 安裝 IBM MQ Telemetry 。
2. 建立並啟動佇列管理程式。 在這項作業中，佇列管理程式稱為 *qMgr* 。
3. 在這項作業中，您可以配置遙測 (MQXR) 服務。 MQXR 內容設定儲存在平台專用內容檔中: mqxr_win.properties。 您通常不需要直接編輯 MQXR 內容檔，因為幾乎所有設定都可以透過 MQSC 管理指令或 MQ Explorer 來配置。 如果您決定直接編輯檔案，請先停止佇列管理程式，再進行變更。 請參閱 MQXR 內容。

關於這項作業

MQ Explorer 的 IBM MQ Telemetry 支援包括精靈及範例指令程序 sampleMQM。 他們使用來賓使用者 ID 來設定起始配置; 請參閱 使用 MQ Explorer 驗證 IBM MQ Telemetry 的安裝 和 IBM MQ Telemetry Transport 範例程式。

請遵循此作業中的步驟，使用不同的授權架構來手動配置 IBM MQ Telemetry 。

程序

1. 在遙測範例目錄中開啟指令視窗。

   遙測範例目錄為 *WMQ program installation directory*\mqxr\samples。

2. 建立遙測傳輸佇列。

   ```
   echo DEFINE QLOCAL('SYSTEM.MQTT.TRANSMIT.QUEUE') USAGE(XMITQ) MAXDEPTH(100000) | runmqsc qMgr
   ```

   第一次啟動遙測 (MQXR) 服務時，它會建立 SYSTEM.MQTT.TRANSMIT.QUEUE。

它在此作業中手動建立，因為 SYSTEM.MQTT.TRANSMIT.QUEUE 必須在遙測 (MQXR) 服務啟動之前存在，才能授權存取它。

3. 設定 *qMgr* 的預設傳輸佇列

```
echo ALTER QMGR DEFXMITQ('SYSTEM.MQTT.TRANSMIT.QUEUE') | runmqsc qMgr
```

圖 *20:* 設定預設傳輸佇列

第一次啟動遙測 (MQXR) 服務時，它不會變更佇列管理程式使 SYSTEM.MQTT.TRANSMIT.QUEUE 成為預設傳輸佇列。

若要使 SYSTEM.MQTT.TRANSMIT.QUEUE 成為預設傳輸佇列，請變更預設傳輸佇列內容。 使用 MQ Explorer 或 第 129 頁的圖 20 中的指令來變更內容。

變更預設傳輸佇列可能會干擾您現有的配置。 將預設傳輸佇列變更為 SYSTEM.MQTT.TRANSMIT.QUEUE 的原因是讓直接傳送訊息至 MQTT 用戶端更容易。 在不變更預設傳輸佇列的情況下，您必須為每個接收 IBM MQ 訊息的用戶端新增遠端佇列定義; 請參閱 第 131 頁的『將訊息直接傳送至用戶端』。

4. 遵循 第 132 頁的『授權 MQTT 用戶端存取 IBM MQ 物件』 中的程序來建立一或多個使用者 ID。 使用者 ID 具有發佈、訂閱及傳送發佈至 MQTT 用戶端的權限。
5. 安裝遙測 (MQXR) 服務

```
type
installMQXRService_win.mqsc | runmqsc qMgr
```

6. 啟動服務

```
echo START SERVICE(SYSTEM.MQXR.SERVICE) | runmqsc qMgr
```

啟動佇列管理程式時，會自動啟動遙測 (MQXR) 服務。

它在此作業中手動啟動，因為佇列管理程式已在執行中。

7. 使用 MQ Explorer，配置遙測通道以接受來自 MQTT 用戶端的連線。

遙測通道必須配置為其身分是步驟 4 中定義的其中一個使用者 ID。

另請參閱 DEFINE CHANNEL (MQTT)。

8. 執行範例用戶端來驗證配置。

若要讓範例用戶端使用遙測通道，該通道必須授權用戶端發佈、訂閱及接收發佈。 依預設，範例用戶端會連接至埠 1883 上的遙測通道。 另請參閱 IBM MQ Telemetry Transport 程式範例。

手動建立 SYSTEM.MQXR.SERVICE

第 129 頁的圖 21 顯示在 Windows 上手動建立 SYSTEM.MQXR.SERVICE 的 **runmqsc** 指令。

```
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')
```

圖 *21: installMQXRService_win.mqsc*

配置分散式佇列作業，以將訊息傳送至 MQTT 用戶端

IBM MQ 應用程式可以透過發佈至用戶端所建立的訂閱，或直接傳送訊息，來傳送 MQTT v3 用戶端訊息。 無論使用哪種方法，都會將訊息放置在 SYSTEM.MQTT.TRANSMIT.QUEUE 上，並由遙測 (MQXR) 服務傳送至用戶端。 有數種方法可以在 SYSTEM.MQTT.TRANSMIT.QUEUE 上放置訊息。

發佈訊息以回應 MQTT 用戶端訂閱

遙測 (MQXR) 服務會代表 MQTT 用戶端建立訂閱。 用戶端是任何符合用戶端所傳送訂閱之發佈的目的地。 遙測服務會將相符發佈轉遞回用戶端。

MQTT 用戶端以佇列管理程式形式連接至 IBM MQ ，並將其佇列管理程式名稱設為 ClientIdentifier。 要傳送至用戶端的發佈目的地是傳輸佇列 SYSTEM.MQTT.TRANSMIT.QUEUE。 遙測服務會使用目標佇列管理程式名稱作為特定用戶端的金鑰，將 SYSTEM.MQTT.TRANSMIT.QUEUE 上的訊息轉遞至 MQTT 用戶端。

遙測 (MQXR) 服務會使用 ClientIdentifier 作為佇列管理程式名稱來開啟傳輸佇列。 遙測 (MQXR) 服務會將佇列的物件控點傳遞至 MQSUB 呼叫，以轉遞符合用戶端訂閱的發佈。 在物件名稱解析中，ClientIdentifier 建立為遠端佇列管理程式名稱，且傳輸佇列必須解析為 SYSTEM.MQTT.TRANSMIT.QUEUE。 使用標準 IBM MQ 物件名稱解析， *ClientIdentifier* 會如下解析; 請參閱 第 130 頁的表 6。

1. *ClientIdentifier* 不符合任何項目。

   ClientIdentifier 是遠端佇列管理程式名稱。 它不符合本端佇列管理程式名稱、佇列管理程式別名或傳輸佇列名稱。

   未定義佇列名稱。 目前，遙測 (MQXR) 服務會將 SYSTEM.MQTT.PUBLICATION.QUEUE 設為佇列名稱。 MQTT v3 用戶端不支援佇列，因此用戶端會忽略已解析的佇列名稱。

   本端佇列管理程式內容 **預設傳輸佇列**名稱必須設為 SYSTEM.MQTT.TRANSMIT.QUEUE，以便將發佈資訊放置在 SYSTEM.MQTT.TRANSMIT.QUEUE 上傳送至用戶端。

2. *ClientIdentifier* 符合名為 *ClientIdentifier* 的佇列管理程式別名。

   ClientIdentifier 是遠端佇列管理程式名稱。 它符合佇列管理程式別名的名稱。

   佇列管理程式別名必須以 *ClientIdentifier* 作為遠端佇列管理程式名稱來定義。

   透過在佇列管理程式別名定義中設定傳輸佇列名稱，預設傳輸不需要設為 SYSTEM.MQTT.TRANSMIT.QUEUE。

表 *6: MQTT* 佇列管理程式別名的名稱解析

	輸入		**輸出**		
ClientIdentifier	**佇列管理程式名稱**	**佇列名稱**	**佇列管理程式名稱**	**佇列名稱**	**傳輸佇列**
不符合任何項目	*ClientIdentifier*	未定義	*ClientIdentifier*	未定義	預設傳輸佇列。 SYSTEM.MQTT.TRANSMIT.QUEUE
符合名為 *ClientIdentifier* 的佇列管理程式別名	*ClientIdentifier*	未定義	*ClientIdentifier*	未定義	SYSTEM.MQTT.TRANSMIT.QUEUE

如需名稱解析的進一步相關資訊，請參閱 名稱解析。

任何 IBM MQ 程式都可以發佈至相同的主題。 發佈會傳送至其訂閱者，包括訂閱主題的 MQTT v3 用戶端。

如果在叢集中建立具有屬性 CLUSTER(*clusterName*)的管理主題，則叢集中的任何應用程式都可以發佈至用戶端; 例如:

```
echo DEFINE TOPIC('MQTTExamples') TOPICSTR('MQTT Examples') CLUSTER(MQTT) REPLACE | runmqsc qMgr
```

圖 *22:* 在 *Windows* 上定義叢集主題

註: 請勿提供叢集屬性給 SYSTEM.MQTT.TRANSMIT.QUEUE 。

MQTT 用戶端訂閱者和發佈者可以連接至不同的佇列管理程式。 訂閱者和發佈者可以是相同叢集的一部分，也可以透過發佈/訂閱階層來連接。 發佈會使用 IBM MQ 從發佈者遞送至訂閱者。

將訊息直接傳送至用戶端

除了用戶端建立訂閱並接收符合訂閱主題的發佈之外，另一種方法是將訊息直接傳送至 MQTT v3 用戶端。 MQTT V3 用戶端應用程式無法直接傳送訊息，但其他應用程式 (例如 IBM MQ 應用程式) 可以。

IBM MQ 應用程式必須知道 MQTT v3 用戶端的 ClientIdentifier 。 因為 MQTT v3 用戶端沒有佇列，所以目標佇列名稱會作為主題名稱傳遞至 MQTT v3 應用程式用戶端 messageArrived 方法。 例如，在 MQI 程式中，建立用戶端作為 ObjectQmgr **名稱**的物件描述子:

```
MQOD.ObjectQmgrName = ClientIdentifier ;
MQOD.ObjectName = name ;
```

圖 *23:* 將訊息傳送至 *MQTT v3* 用戶端目的地的 *MQI* 物件描述子

如果使用 JMS 撰寫應用程式，請建立點對點目的地; 例如:

```
javax.jms.Destination jmsDestination =
(javax.jms.Destination)jmsFactory.createQueue
("queue://ClientIdentifier/name");
```

圖 *24: JMS* 目的地，用於將訊息傳送至 *MQTT v3* 用戶端

若要將自發訊息傳送至 MQTT 用戶端，請使用遠端佇列定義。 遠端佇列管理程式名稱必須解析為用戶端的 ClientIdentifier 。 傳輸佇列必須解析為 SYSTEM.MQTT.TRANSMIT.QUEUE ; 請參閱 第 131 頁的表 7。 遠端佇列名稱可以是任何名稱。 用戶端會將它當作主題字串來接收。

表 *7: MQTT* 用戶端遠端佇列定義的名稱解析

輸入		**輸出**		
佇列名稱	**佇列管理程式名稱**	**佇列名稱**	**佇列管理程式名稱**	**傳輸佇列**
遠端佇列定義的名稱	空白或本端佇列管理程式名稱	用作主題字串的遠端佇列名稱	*ClientIdentifier*	SYSTEM.MQTT.TRANSMIT.QUEUE

如果已連接用戶端，則會將訊息直接傳送至 MQTT 用戶端，該用戶端會呼叫 messageArrived 方法; 請參閱 messageArrived 方法。

如果用戶端已中斷與持續性階段作業的連線，則訊息會儲存在 SYSTEM.MQTT.TRANSMIT.QUEUE 中; 請參閱 MQTT 無狀態及有狀態階段作業。 當用戶端重新連接至階段作業時，會將它轉遞至用戶端。

如果您傳送非持續訊息，則會以「最多一次」服務品質 QoS=0 將訊息傳送至用戶端。 如果您直接將持續訊息傳送至用戶端，依預設會以「正好一次」服務品質 QoS=2 來傳送持續訊息。 由於用戶端可能沒有持續性機制，因此用戶端可以降低直接傳送訊息所接受的服務品質。 若要降低直接傳送至用戶端之訊息的服務品質，請訂閱主題 DEFAULT.QoS。 指定用戶端可支援的服務品質上限。

MQTT 用戶端識別、授權及鑑別

遙測 (MQXR) 服務會使用 MQTT 通道，代表 MQTT 用戶端發佈或訂閱 IBM MQ 主題。 IBM MQ 管理者會配置用於 IBM MQ 授權的 MQTT 通道身分。 管理者可以定義通道的一般身分，或者使用已連接至通道的用戶端的 Username 或 ClientIdentifier。

遙測 (MQXR) 服務可以使用用戶端提供的 Username 或者使用用戶端憑證來鑑別用戶端。 則使用用戶端提供的密碼來鑑別 Username。

彙總： 用戶端識別是用戶端身分的選項。 根據環境定義，用戶端由 ClientIdentifier、Username、管理者建立的一般用戶端身分或用戶端憑證識別。 用於確實性檢查的用戶端 ID 不一定是用於授權的 ID。

MQTT 用戶端程式會設定使用 MQTT 通道傳送至伺服器的 **使用者名稱** 及 **密碼** 。 它們還可以設定加密和鑑別連線所需的 SSL 內容。 管理者會決定是否鑑別 MQTT 通道，以及如何鑑別通道。

若要授權 MQTT 用戶端存取 IBM MQ 物件，請授權用戶端的 ClientIdentifier 或 Username ，或授權一般用戶端身分。 若要允許用戶端連接至 IBM MQ，請鑑別 Username 或使用用戶端憑證。 配置 JAAS 以鑑別 Username，或者配置 SSL 以鑑別用戶端憑證。

如果您在用戶端設定 **密碼** ，請使用 VPN 加密連線，或將 MQTT 通道配置為使用 SSL ，以保持密碼專用。

管理用戶端憑證非常困難。 因此，如果與密碼鑑別相關聯的風險可以接受，則一般會使用密碼鑑別來鑑別用戶端。

如果可以安全地管理和儲存用戶端憑證，則可以依賴於憑證鑑別。 不過，在使用遙測的環境類型中，極少能夠安全地管理憑證。 而是使用用戶端憑鑑別裝置，補充在伺服器鑑別用戶端密碼。 因為其他複雜性，用戶端憑證的使用限制為高度機密的應用。 使用兩種形式的鑑別稱為兩因素鑑別。 您必須知道其中一個因素（例如密碼），並具有另一個因素（例如憑證）。

在高度機密的應用（例如 chip-and-pin 裝置）中，在製造期間會鎖定裝置，以防止竄改內部軟硬體。 將受時間限制的授信用戶端憑證複製到裝置。 將裝置部署至要使用它的位置。 每次使用裝置時，都會使用密碼或智慧卡的另一個憑證，來執行進一步鑑別。

MQTT 用戶端身分及授權

使用用戶端 ID、 Username 或一般用戶端身分來授權存取 IBM MQ 物件。

IBM MQ 管理者有三個選項，可用來選取 MQTT 通道的身分。 管理者在定義或修改用戶端所使用的 MQTT 通道時進行選擇。 身分用於授與 IBM MQ 主題的存取權。 選擇順序如下:

1. 用戶端 ID (請參閱 USECLNTID)。
2. 管理者為通道提供的身分 (通道的 MCAUSER)。 請參閱 MCAUSER)。
3. 如果上述任一選項都不適用，則從 MQTT 用戶端傳遞的 **使用者名稱** (Username 是 MqttConnectOptions 類別的屬性。 必須在用戶端連接至服務之前將其設定。 其預設值為空值)。

避免麻煩: 之後會參照此程序所選擇的身分，例如 DISPLAY CHSTATUS (MQTT) 指令作為用戶端的 MCAUSER。 請注意，這不一定與選項 (2) 中所參照通道的 MCAUSER 相同。

使用 IBM MQ **setmqaut** 指令來選取與 MQTT 通道相關聯的身分已授權使用哪些物件及哪些動作。 例如，下列程式碼授權通道身分 MQTTClient，由佇列管理程式 QM1 的管理者提供:

```
 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
```

授權 *MQTT* 用戶端存取 *IBM MQ* 物件

遵循下列步驟，以授權 MQTT 用戶端發佈及訂閱 IBM MQ 物件。 下列步驟遵循四個替代存取控制型樣。

開始之前

MQTT 用戶端在連接至遙測通道時，會獲指派身分，以存取 IBM MQ 中的物件。 「IBM MQ 管理者」會使用 IBM MQ Explorer 來配置遙測通道，以將三種身分類型之一提供給用戶端:

1. ClientIdentifier

2. **使用者名稱**
3. 管理者指派給通道的名稱。

不論使用哪一種類型，已安裝的授權服務必須將身分定義給 IBM MQ 作為主體。 Windows 或 Linux 上的預設授權服務稱為「物件權限管理程式 (OAM)」。 如果您是使用 OAM，則身分必須定義為使用者 ID。

使用身分為用戶端或用戶端集合提供發佈或訂閱 IBM MQ 中所定義主題的許可權。 如果 MQTT 用戶端已訂閱主題，請使用身分來授與它接收產生的發佈資訊的許可權。

很難管理具有數萬個 MQTT 用戶端的系統，每個用戶端都需要個別存取權。 其中一個解決方案是定義一般身分，並將個別 MQTT 用戶端與其中一個一般身分相關聯。 定義所需數量的一般身分，以定義不同的許可權組合。 另一個解決方案是撰寫您自己的授權服務，它比作業系統更容易處理數以千計的使用者。

您可以使用 OAM，以兩種方式將 MQTT 用戶端結合成一般身分:

1. 定義多個遙測通道，每一個遙測通道都具有管理者使用「IBM MQ 探險家」配置的不同使用者 ID。 使用不同 TCP/IP 埠號連接的用戶端會與不同的遙測通道相關聯，並獲指派不同的身分。
2. 定義單一遙測通道，但讓每一個用戶端從一小組使用者 ID 中選取 **使用者名稱** 。 管理者會配置遙測通道，以選取用戶端 **使用者名稱** 作為其身分。

在此作業中，遙測通道的身分稱為 *mqttUser*，不論其設定方式為何。 如果用戶端集合使用不同的身分，請使用多個 *mqttUsers*，每一個用戶端集合一個。 由於作業使用 OAM，因此每一個 *mqttUser* 都必須是使用者 ID。

關於這項作業

在這項作業中，您可以選擇四個存取控制型樣，以符合特定需求。 型樣在存取控制的精度方面有所不同。

- 第 133 頁的『無存取控制』
- 第 133 頁的『粗略存取控制』
- 第 133 頁的『中階存取控制』
- 第 134 頁的『精細存取控制』

模型的結果是將 *mqttUsers* 許可權集指派給發佈和訂閱 IBM MQ，以及從 IBM MQ 接收發佈。

無存取控制
MQTT 用戶端會獲授與 IBM MQ 管理權限，且可以對任何物件執行任何動作。

程序

1. 建立使用者 ID *mqttUser*，以作為所有 MQTT 用戶端的身分。
2. 將 *mqttUser* 新增至 mqm 群組; 請參閱 在 Windows 上新增使用者至群組，或 在 Linux 上新增使用者至群組

粗略存取控制
MQTT 用戶端有權發佈及訂閱，以及將訊息傳送至 MQTT 用戶端。 他們沒有執行其他動作或存取其他物件的權限。

程序

1. 建立使用者 ID *mqttUser*，以作為所有 MQTT 用戶端的身分。
2. 授權 *mqttUser* 發佈及訂閱所有主題，以及將發佈傳送至 MQTT 用戶端。

```
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
```

中階存取控制
MQTT 用戶端分成不同的群組，以發佈及訂閱不同的主題集，以及將訊息傳送至 MQTT 用戶端。

程序

1. 在發佈/訂閱主題樹狀結構中建立多個使用者 ID、 *mqttUsers* 及多個管理主題。
2. 將不同的 *mqttUsers* 授權給不同的主題。

```
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. 建立群組 *mqtt*，並將所有 *mqttUsers* 新增至群組。
4. 授權 *mqtt* 將主題傳送至 MQTT 用戶端。

```
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

精細存取控制
MQTT 用戶端會納入現有的存取控制系統中，授權群組對物件執行動作。

關於這項作業

使用者 ID 會指派給一或多個作業系統群組，視它需要的授權而定。 如果 IBM MQ 應用程式發佈及訂閱與 MQTT 用戶端相同的主題空間，請使用此模型。 群組稱為 Publish *X*、 Subscribe *Y* 及 mqtt

Publish *X*
Publish *X* 群組的成員可以發佈至 *topicX*。

Subscribe *Y*
Subscribe *Y* 群組的成員可以訂閱 *topicY*。

mqtt
mqtt 群組的成員可以將發佈傳送至 MQTT 用戶端。

程序

1. 在發佈/訂閱主題樹狀結構中，建立多個配置給多個管理主題的群組 Publish *X* 和 Subscribe *Y* 。
2. 建立群組 mqtt。
3. 建立多個使用者 ID *mqttUsers*，並視他們獲授權執行的動作而定，將使用者新增至任何群組。
4. 將不同的 Publish *X* 及 Subscribe *X* 群組授權給不同的主題，並授權 *mqtt* 群組將訊息傳送至 MQTT 用戶端。

```
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
```

MQTT 使用密碼進行用戶端鑑別

使用用戶端密碼鑑別 Username。 用於鑑別用戶端的身分，可以不同於用於授權用戶端發佈至和訂閱主題的身分。

遙測 (MQXR) 服務會使用 JAAS 來鑑別用戶端 Username。 JAAS 使用 MQTT 用戶端提供的 **密碼** 。

IBM MQ 管理者透過配置用戶端所連接的 MQTT 通道，來決定是否鑑別 **使用者名稱**，或完全不鑑別。 可以將用戶端指派給不同通道，而且可以配置每個通道以使用不同的方法鑑別其用戶端。 如果使用 JAAS，則您可以配置哪些方法必須鑑別用戶端，哪些方法可以選擇性地鑑別用戶端。

選擇用於鑑別的身分不會影響選擇用於授權的身分。 為了方便管理，您可能會設定用於授權的一般身分，但是鑑別要使用該身分的每個使用者。 下列程序概述的步驟，用於鑑別要使用一般身分的個別使用者：

1. IBM MQ 管理者使用 IBM MQ Explorer 將 MQTT 通道身分設為任何名稱，例如 MQTTClientUser。

2. IBM MQ 管理者授權 MQTTClient 發佈和訂閱任何主題:

```
 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. 在連接至伺服器之前， MQTT 用戶端應用程式開發者會建立 MqttConnectOptions 物件，並設定 Username 和 Password 。
4. 安全開發者建立 JAAS LoginModule，以利用 Password 鑑別 Username，並將它併入 JAAS 配置檔。
5. IBM MQ 管理者會配置 MQTT 通道，以使用 JAAS 來鑑別用戶端的 UserName 。

MQTT 使用 SSL 進行用戶端鑑別

MQTT 用戶端與佇列管理程式之間的連線一律由 MQTT 用戶端起始。 MQTT 用戶端一律是 SSL 用戶端。 伺服器的用戶端鑑別和 MQTT 用戶端的伺服器鑑別皆為選用項目。

透過向用戶端提供專用簽章數位憑證，您可以向 WebSphere MQ 鑑別 MQTT 用戶端。 WebSphere MQ 管理者可以強制 MQTT 用戶端使用 SSL 向佇列管理程式鑑別本身。 您只能透過交互鑑別來要求用戶端鑑別。

作為使用 SSL 的替代方案，一些類型的「虛擬私密網路 (VPN)」（例如 IPsec），會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 MQTT 用戶端連接至遙測通道。

使用 SSL 進行的用戶端鑑別依賴於具有密碼的用戶端。 對於自簽憑證，密碼是用戶端的私密金鑰，否則是憑證管理中心提供的金鑰。 金鑰用於簽章用戶端的數位憑證。 擁有對應公開金鑰的任何人都可以驗證該數位憑證。 可以信任憑證，如果這些憑證已鏈結，則可以透過憑證鏈回溯追蹤至授信主要憑證。 用戶端驗證會將用戶端所提供憑證鏈中的所有憑證傳送至伺服器。 伺服器會檢查憑證鏈，直到它找到信任的憑證為止。 授信憑證是從自簽憑證產生的公用憑證，或者是一般由憑證管理中心發出的主要憑證。 在最後一個選用步驟中，可以將信任的憑證與「現用」的憑證撤銷清冊相互比較。

授信憑證可能由憑證管理中心發出，並且已經包含在 JRE 憑證儲存庫中。 它可以是自簽憑證，也可以是已作為授信憑證新增至遙測通道金鑰儲存庫的任何憑證。

註: 遙測通道具有結合的金鑰儲存庫/信任儲存庫，其中保留一個以上遙測通道的私密金鑰，以及鑑別用戶端所需的任何公用憑證。 因為 SSL 通道必須具有金鑰儲存庫，所以它與通道信任儲存庫是同一個檔案，永不參照 JRE 憑證儲存庫。 言下之意，如果用戶端鑑別需要 CA 主要憑證，您必須將主要憑證放置於通道的金鑰儲存庫中，即使 JRE 憑證儲存庫中已經存在 CA 主要憑證也是如此。 永不參照 JRE 憑證儲存庫。

考慮用戶端鑑別打算應對的威脅，以及用戶端和伺服器在應對威脅時所扮演的角色。 單獨鑑別用戶端憑證不足以防止未獲授權存取系統。 如果其他使用者已在使用該用戶端裝置，則該用戶端裝置不需使用憑證持有者的權限就可以運作。 切勿依賴單一防禦措施來防範意外攻擊。 至少使用雙重因數的鑑別方法，以及補充關於擁有憑證的私密資訊知識。 例如，使用 JAAS，並且使用伺服器發出的密碼鑑別用戶端。

用戶端憑證的主要威脅是落入不適合的人手中。 憑證保存在用戶端上受密碼保護的金鑰儲存庫中。 系統是如何將憑證放入金鑰儲存庫中？ MQTT 用戶端如何取得金鑰儲存庫的密碼？ 密碼保護的安全程度如何？ 遙測裝置通常易於移除，然後可被私下入侵。 裝置硬體是否必須具有防竄改功能？ 配送和保護用戶端憑證非常困難，這稱為金鑰管理問題。

次要威脅是無意中誤用裝置來存取伺服器。 比方說，如果 MQTT 應用程式被竄改，則它可能會使用已鑑別的用戶端身分，利用伺服器配置中的缺點。

若要使用 SSL 來鑑別 MQTT 用戶端，請配置遙測通道及用戶端。

相關概念

第 136 頁的『使用 SSL 進行 MQTT 用戶端鑑別的遙測通道配置』

IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 會利用金鑰檔的受通行詞組保護的存取權，來配置 SSL 通道。 如果未利用通行詞組或金鑰檔定義 SSL 通道，則該通道不會接受 SSL 連線。

相關資訊

使用 SSL 進行用戶端鑑別的 MQTT 用戶端配置

使用 *SSL* 進行 *MQTT* 用戶端鑑別的遙測通道配置

IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 會利用金鑰檔的受通行詞組保護的存取權，來配置 SSL 通道。 如果未利用通行詞組或金鑰檔定義 SSL 通道，則該通道不會接受 SSL 連線。

將 SSL 遙測通道的 `com.ibm.mq.MQTT.ClientAuth` 內容設為 `REQUIRED`，以強制在該通道上連接的所有用戶端提供其已驗證數位憑證的證明。 用戶端憑證是使用來自憑證管理中心的憑證來鑑別，並導向授信主要憑證。 如果用戶端憑證是自簽的，或由來自憑證管理中心的憑證所簽署，則用戶端或憑證管理中心的公開簽署憑證必須安全地儲存在伺服器上。

將公開簽署的用戶端憑證或來自憑證管理中心的憑證放置在遙測通道金鑰儲存庫中。 在伺服器上，公開簽署的憑證儲存在與私密簽署憑證相同的金鑰檔中，而不是儲存在個別信任儲存庫中。

伺服器會使用它所擁有的所有公用憑證及密碼組合，來驗證它所傳送之任何用戶端憑證的簽章。 伺服器會驗證金鑰鏈。 佇列管理程式可以配置成根據憑證撤銷清冊來測試憑證。 佇列管理程式撤銷名稱清單內容是 `SSLCRLNL`。

如果伺服器金鑰儲存庫中的憑證已驗證用戶端傳送的任何憑證，則會鑑別用戶端。

IBM MQ 管理者可以配置相同的遙測通道，以使用 JAAS 來檢查用戶端的 `UserName` 或 `ClientIdentifier` (具有用戶端 **密碼**)。

您可以對多個遙測通道使用相同的金鑰儲存庫。

驗證裝置上受密碼保護的用戶端金鑰儲存庫中至少一個數位憑證會向伺服器鑑別用戶端。 數位憑證僅用於由 IBM MQ 進行鑑別。 它不是用來驗證用戶端的 TCP/IP 位址，或設定授權或帳戶的用戶端身分。 伺服器採用的用戶端身分是用戶端的 `Username` 或 `ClientIdentifier`，或 IBM MQ 管理者所建立的身分。

您也可以使用 SSL 密碼組合來進行用戶端鑑別。 如果您計劃使用 SHA-2 密碼組合，請參閱第 137 頁的『搭配使用 SHA-2 密碼組合與 MQTT 通道的系統需求』。

相關概念

第 137 頁的『使用 SSL 進行通道鑑別的遙測通道配置』
IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 會利用金鑰檔的受通行詞組保護的存取權，來配置 SSL 通道。 如果未利用通行詞組或金鑰檔定義 SSL 通道，則該通道不會接受 SSL 連線。

相關資訊

DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)
CipherSpecs 和 CipherSuites

使用 SSL 進行遙測通道鑑別

MQTT 用戶端與佇列管理程式之間的連線一律由 MQTT 用戶端起始。 MQTT 用戶端一律是 SSL 用戶端。 伺服器的用戶端鑑別和 MQTT 用戶端的伺服器鑑別皆為選用項目。

除非將用戶端配置為使用支援匿名連線的 CipherSpec，否則它一律會嘗試鑑別伺服器。 如果鑑別失敗，則不會建立連線。

作為使用 SSL 的替代方案，一些類型的「虛擬私密網路 (VPN)」（例如 IPsec），會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 MQTT 用戶端連接至遙測通道。

使用 SSL 的伺服器鑑別會鑑別作為要傳送機密資訊目標的伺服器。 用戶端會針對放置在其信任儲存庫或 JRE `cacerts` 儲存庫中的憑證，執行符合從伺服器傳送之憑證的檢查。

JRE 憑證儲存庫是 JKS 檔案 `cacerts`。 它位於 `JRE InstallPath\lib\security\`。 安裝後，它的預設密碼為 `changeit`。 您可以將信任的憑證儲存在 JRE 憑證儲存庫或用戶端信任儲存庫中。 不能同時使用這兩個儲存庫。 如果您想要將用戶端信任的公用憑證與其他 Java 應用程式使用的憑證分開，請使用用戶端信任儲存庫。 如果您想要對用戶端上執行的所有 Java 應用程式使用一般憑證儲存庫，請使用 JRE 憑證儲存庫。 如果決定使用 JRE 憑證儲存庫，請檢閱它所包含的憑證，以確保您信任這些憑證。

透過提供不同的信任提供者，您可以修改 JSSE 配置。 您可以自訂信任提供者，以對憑證執行不同的檢查。 在部分已使用 MQTT 用戶端的 OGSi 環境中，環境會提供不同的信任提供者。

若要使用 SSL 來鑑別遙測通道，請配置伺服器及用戶端。

使用 SSL 進行通道鑑別的遙測通道配置

IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 會利用金鑰檔的受通行詞組保護的存取權，來配置 SSL 通道。 如果未利用通行詞組或金鑰檔定義 SSL 通道，則該通道不會接受 SSL 連線。

將伺服器的數位憑證 (使用其私密金鑰簽署) 儲存在遙測通道將在伺服器上使用的金鑰儲存庫中。 如果您要將金鑰鏈傳輸至用戶端，請將任何憑證儲存在金鑰儲存庫的金鑰鏈中。 使用 IBM MQ 瀏覽器來配置遙測通道以使用 SSL。 請提供金鑰儲存庫的路徑，以及用來存取金鑰儲存庫的通行詞組。 如果您未設定通道的 TCP/IP 埠號，則 SSL 遙測通道埠號預設為 8883。

您也可以使用 SSL 密碼組合來進行通道鑑別。 如果您計劃使用 SHA-2 密碼組合，請參閱第 137 頁的『搭配使用 SHA-2 密碼組合與 MQTT 通道的系統需求』。

相關概念

第 136 頁的『使用 SSL 進行 MQTT 用戶端鑑別的遙測通道配置』
IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 會利用金鑰檔的受通行詞組保護的存取權，來配置 SSL 通道。 如果未利用通行詞組或金鑰檔定義 SSL 通道，則該通道不會接受 SSL 連線。

相關資訊

DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)
CipherSpecs 和 CipherSuites

搭配使用 SHA-2 密碼組合與 MQTT 通道的系統需求

如果您使用支援 SHA-2 密碼組合的 Java 版本，您可以使用這些組合來保護 MQTT (遙測) 通道及用戶端應用程式的安全。

對於包括遙測 (MQXR) 服務的 IBM MQ 8.0 ，最低 Java 版本為 Java 7 from IBM ， SR6。 依預設，從 IBM(SR4) 開始， Java 7 中支援 SHA-2 密碼組合。 因此，您可以搭配使用 SHA-2 密碼組合與遙測 (MQXR) 服務，以保護 MQTT (遙測) 通道的安全。

如果您使用不同的 JRE 來執行 MQTT 用戶端，則需要確定它也支援 SHA-2 密碼組合。

相關概念

第 137 頁的『使用 SSL 進行通道鑑別的遙測通道配置』
IBM MQ 管理者在伺服器上配置遙測通道。 會將每個通道配置為接受不同埠號上的 TCP/IP 連線。 會利用金鑰檔的受通行詞組保護的存取權，來配置 SSL 通道。 如果未利用通行詞組或金鑰檔定義 SSL 通道，則該通道不會接受 SSL 連線。

相關資訊

遙測 (MQXR) 服務
DEFINE CHANNEL (MQTT)
ALTER CHANNEL (MQTT)

遙測通道上的發佈保密

使用 SSL 來加密透過連線的傳輸，可保護透過遙測通道以任一方向傳送之 MQTT 發佈資訊的隱私權。

連接至遙測通道的 MQTT 用戶端會使用 SSL ，以使用對稱金鑰加密法來保護通道上所傳輸發佈的隱私權。 因為不會鑑別端點，所以您不能信任單獨使用的通道加密。 將保密安全與伺服器或交互鑑別結合使用。

作為使用 SSL 的替代方案，一些類型的「虛擬私密網路 (VPN)」（例如 IPsec） ，會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 MQTT 用戶端連接至遙測通道。

對於會加密通道和鑑別伺服器的一般配置，請參閱第 136 頁的『使用 SSL 進行遙測通道鑑別』。

如果加密 SSL 連線而不鑑別伺服器，則會使連線受到中間人攻擊。 雖然您交換的資訊可以防竊聽，但是您不知道與您交換資訊的對象是誰。 除非您控制網路，否則您就會面臨別人截取您的 IP 傳輸，以及假冒端點的問題。

您可以建立加密的 SSL 連線，而不鑑別伺服器，方法是使用支援匿名 SSL 的 Diffie-Hellman 金鑰交換 CipherSpec。 在用戶端和伺服器之間共用且用於加密 SSL 傳輸的主要機密，在建立時不必交換私密簽章的伺服器憑證。

因為匿名連線不安全，所以大部分 SSL 實作不會預設為使用匿名 CipherSpec。 如果遙測通道接受 SSL 連線的用戶端要求，則該通道必須具有受通行詞組保護的金鑰儲存庫。 依預設，因為 SSL 實作不會使用匿名 CipherSpec，所以金鑰儲存庫必須包含用戶端可以鑑別的私密簽章憑證。

如果您使用匿名 CipherSpec，則伺服器金鑰儲存庫必須存在，但是它不必包含任何私密簽章的憑證。

建立加密連線的另一種方法，是利用您自己的實作，取代用戶端上的信任提供者。 您的信任提供者不會鑑別伺服器憑證，但是會加密連線。

小心: 將 TLS 與 MQTT 搭配使用時，您可以使用大型訊息，不過，這樣做可能會影響效能。 MQTT 已針對處理小型訊息 (大小通常介於 1KB 與 1MB 之間) 進行最佳化。

MQTT Java 用戶端及遙測通道的 SSL 配置

配置 SSL 以鑑別遙測通道及 MQTT Java 用戶端，並加密它們之間的訊息傳送。 MQTT Java 用戶端使用 Java Secure Socket Extension (JSSE) 來使用 SSL 連接遙測通道。 作為使用 SSL 的替代方案，一些類型的「虛擬私密網路 (VPN)」（例如 IPsec），會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 MQTT 用戶端連接至遙測通道。

您可以配置 Java MQTT 用戶端與遙測通道之間的連線，以使用透過 TCP/IP 的 SSL 通訊協定。 維護安全的內容視配置 SSL 以使用 JSSE 的方式而定。 您可以配置三個不同的安全等級，從最安全的配置開始:

1. 只允許信任的 MQTT 用戶端連接。 僅將 MQTT 用戶端連接至授信遙測通道。 加密用戶端與佇列管理程式之間的訊息; 請參閱 第 135 頁的『MQTT 使用 SSL 進行用戶端鑑別』
2. 僅將 MQTT 用戶端連接至授信遙測通道。 加密用戶端和佇列管理程式之間的訊息；請參閱第 136 頁的『使用 SSL 進行遙測通道鑑別』。
3. 加密用戶端和佇列管理程式之間的訊息；請參閱第 137 頁的『遙測通道上的發佈保密』。

JSSE 配置參數

修改 JSSE 參數，以變更配置 SSL 連線的方式。 JSSE 配置參數組織成三個集:

1. IBM MQ Telemetry 通道
2. MQTT Java 用戶端
3. JRE

使用「IBM MQ 探險家」來配置遙測通道參數。 在 `MqttConnectionOptions.SSLProperties` 屬性中設定 MQTT Java 用戶端參數。 在用戶端和伺服器上，透過編輯 JRE 安全目錄中的檔案，修改 JRE 安全參數。

IBM MQ Telemetry channel

使用「IBM MQ 探險家」來設定所有遙測通道 SSL 參數。

`ChannelName`

在所有通道上，`ChannelName` 都是必要的參數。

通道名稱識別與特定埠號相關聯的通道。 命名通道以協助您管理 MQTT 用戶端集。

`PortNumber`

在所有通道上，`PortNumber` 都是選用參數。 對於 TCP 通道，預設值為 1883，對於 SSL 通道，預設值為 8883。

與此通道相關聯的 TCP/IP 埠號。 透過指定為通道定義的埠，將 MQTT 用戶端連接至通道。 如果通道具有 SSL 內容，則用戶端必須使用 SSL 通訊協定進行連接；例如：

```
MQTTClient mqttClient = new MqttClient( "ssl://www.example.org:8884", "clientId1");
mqttClient.connect();
```

KeyFileName

KeyFileName 是 SSL 通道的必要參數。 對於 TCP 通道，必須省略。

KeyFile 名稱 是 Java 金鑰儲存庫的路徑，其中包含您提供的數位憑證。 在伺服器上，使用 JKS、JCEKS 或 PKCS12 作為金鑰儲存庫類型。

使用下列其中一個副檔名來識別金鑰儲存庫類型：

```
.jks
.jceks
.p12
.pkcs12
```

使用任何其他副檔名的金鑰儲存庫將被視為 JKS 金鑰儲存庫。

您可以將伺服器上的其中一種金鑰儲存庫類型和用戶端上的其他金鑰儲存庫類型結合使用。

將伺服器的專用憑證放置在金鑰儲存庫中。 該憑證稱為伺服器憑證。 憑證可以是自簽憑證，也可以是簽章管理中心所簽章的憑證鏈之一部分。

如果您使用憑證鏈，請將相關憑證置於伺服器的金鑰儲存庫中。

會將伺服器憑證及其憑證鏈中的所有憑證，傳送至用戶端以鑑別伺服器身分。

如果您已將 ClientAuth 設定為 Required，則金鑰儲存庫必須包含鑑別用戶端所需的所有憑證。 用戶端會傳送自簽憑證或憑證鏈，同時會對照金鑰儲存庫內的憑證，以此資料的第一個驗證來鑑別用戶端。 透過使用憑證鏈，即使多個用戶端由不同的用戶端憑證發出，都可利用單一憑證加以驗證。

PassPhrase

PassPhrase 是 SSL 通道的必要參數。 對於 TCP 通道，必須省略。

通行詞組用於保護金鑰儲存庫。

ClientAuth

ClientAuth 是一個選用 SSL 參數。 它預設為不執行用戶端鑑別。 對於 TCP 通道，必須省略。

如果要遙測 (MQXR) 服務先鑑別用戶端，再允許用戶端連接至遙測通道，請設定 ClientAuth。

如果設定 ClientAuth，則用戶端必須使用 SSL 連接至伺服器，並鑑別伺服器。 作為設定 ClientAuth 的結果，用戶端會將其數位憑證及其金鑰儲存庫中的所有憑證傳送至伺服器。 其數位憑證稱為用戶端憑證。 會針對通道金鑰儲存庫和 JRE cacerts 儲存庫中保有的憑證，鑑別這些憑證。

CipherSuite

CipherSuite 是一個選用 SSL 參數。 它預設為嘗試所有已啟用的 CipherSpecs。 對於 TCP 通道，必須省略。

如果要使用特定 CipherSpec，請將 CipherSuite 設定為必須用於建立 SSL 連線的 CipherSpec 名稱。

遙測服務及 MQTT 用戶端會從每一端啟用的所有 CipherSpecs 協議一般 CipherSpec 。 如果在連線的任一端或全部兩端指定特定 CipherSpec，則它必須與另一端的 CipherSpec 相符。

透過將其他提供者新增至 JSSE，安裝其他密碼。

Federal Information Processing Standards (FIPS)

FIPS 是一項選用設定。 依預設不會對其設定。

在佇列管理程式的「內容」畫面中，或者使用 **runmqsc**，可以設定 SSLFIPS。 SSLFIPS 指定是否僅使用通過 FIPS 認證的演算法。

名單

撤銷名單是一項選用設定。 依預設不會對其設定。

在佇列管理程式的「內容」畫面中，或者使用 **runmqsc**，可以設定 SSLCRLNL。 SSLCRLNL 指定用於提供憑證撤銷位置的鑑別資訊物件名單。

不會使用其他設定 SSL 內容的佇列管理程式參數。

MQTT Java 用戶端

在 MqttConnectionOptions.SSLProperties 中設定 Java 用戶端的 SSL 內容; 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);
```

特定內容的名稱和值在 MqttConnectOptions 類別中說明。 如需 MQTT 用戶端程式庫的用戶端 API 文件鏈結，請參閱 MQTT 用戶端程式設計參考手冊。

Protocol

Protocol 是選用項目。

通訊協定是在與遙測伺服器協議後選取。 如果您需要特定通訊協定，則您可以選取它。 如果遙測伺服器不支援該通訊協定，則連線失敗。

ContextProvider

ContextProvider 是選用項目。

KeyStore

KeyStore 是選用項目。 如果在伺服器端設定 ClientAuth 以強制鑑別用戶端，請予以配置。

將使用用戶端的私密金鑰簽章的用戶端數位憑證，放入金鑰儲存庫。 指定金鑰儲存庫路徑和密碼。 類型和提供者是選用項目。 JKS 是預設類型，IBMJCE 是預設提供者。

指定不同的金鑰儲存庫提供者，以參照新增金鑰儲存庫提供者的類別。 透過設定金鑰管理者名稱，傳遞金鑰儲存庫提供者所用演算法的名稱，以實例化 KeyManagerFactory。

TrustStore

TrustStore 是選用項目。 您可以將您信任的所有憑證都放在 cacerts 儲存庫中。

如果想要擁有不同的用戶端信任儲存庫，請配置信任儲存庫。 如果伺服器是使用常用 CA（已將其主要憑證儲存在 cacerts 中）簽章的憑證，則無法配置信任儲存庫。

將伺服器的公共簽章的憑證或主要憑證新增至信任儲存庫，並指定信任儲存庫路徑和密碼。 JKS 是預設類型，IBMJCE 是預設提供者。

指定不同的信任儲存庫提供者，以參照新增信任儲存庫提供者的類別。 透過設定信任管理者名稱，傳遞信任儲存庫提供者所用演算法的名稱，以實例化 TrustManagerFactory。

JRE

可影響用戶端及伺服器上 SSL 行為的 Java 安全的其他方面，在 JRE 中配置。 Windows 上的配置檔位於 *Java Installation Directory*\jre\lib\security 中。 如果您使用的是 IBM MQ 隨附的 JRE，則路徑如下表中所示:

表 *8: JRE SSL* 配置檔的檔案路徑（依平台）	
平台	**菲萊帕特**
Windows	*WMQ Installation Directory*\java\jre\lib\security

表 *8: JRE SSL* 配置檔的檔案路徑（依平台）*(*繼續*)*	
平台	**菲萊帕特**
其他 UNIX 和 Linux 平台	*WMQ Installation Directory*/java/jre64/jre/lib/security

常用憑證管理中心

cacerts 檔案包含常用憑證管理中心的主要憑證。 除非您指定信任儲存庫，否則依預設會使用 cacerts。 如果您使用 cacerts 信任儲存庫或未提供信任儲存庫，則必須檢閱並編輯 cacerts 中的簽章者清單，以滿足您的安全需求。

您可以使用執行 IBM Key Management 公用程式的 IBM MQ 指令 strmqkm 來開啟 cacerts 。 使用密碼 changeit，將 cacerts 作為 JKS 檔案開啟。 修改密碼以維護檔案的安全。

配置安全類別

使用 java.security 檔案以登錄其他安全提供者和其他預設安全內容。

許可權

使用 java.policy 檔案來修改授與資源的權限。 javaws.policy 會授與 javaws.jar 的權限

加密強度

一些 JRE 提供強度減弱的加密。 如果您無法將金鑰匯入至金鑰儲存庫，則原因可能是加密的強度減弱。 請嘗試使用 **strmqikm** 指令來啟動 **ikeyman** ，或從 IBM 開發人員套件、安全資訊下載強大但適用範圍有限的檔案。

重要: 您所在的國家/地區可能會對加密軟體的進口、佔有、使用或轉口至其他國家/地區施加限制。 在下載或使用未限定政策檔案之前，您必須檢查您所在國家/地區的法律。 請檢查其進口、佔有、使用和轉口加密軟體的相關法規和政策，判斷是否允許該軟體。

修改信任提供者以允許用戶端連接至所有伺服器

此範例說明如何新增信任提供者，並從 MQTT 用戶端程式碼參照它。 範例不會執行用戶端或伺服器鑑別。 產生的 SSL 連線已加密但未經鑑別。

第 141 頁的圖 25 中的程式碼 Snippet 會設定 MQTT 用戶端的 AcceptAllProviders 信任提供者和信任管理程式。

```
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);
```

圖 *25: MQTT* 用戶端程式碼 *Snippet*

```
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());
}
```

圖 *26: 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() };
}
```

圖 *27: 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);
}
}
```

圖 *28: AcceptAllX509TrustManager.java*

遙測通道 JAAS 配置

配置 JAAS 以鑑別用戶端傳送的 Username。

IBM MQ 管理者會使用 JAAS 來配置哪些 MQTT 通道需要用戶端鑑別。 指定要執行 JAAS 鑑別的每個通道的 JAAS 配置名稱。 通道可以全部使用相同的 JAAS 配置，也可以使用不同的 JAAS 配置。 配置定義在 *WMQData directory*\qmgrs*qMgrName*\mqxr\jaas.config 中。

jaas.config 檔案依 JAAS 配置名稱組織。 在每個配置名稱下面，是「登入」配置清單；請參閱 第 143 頁的圖 29。

JAAS 提供四個標準「登入」模組。 標準 NT 和 UNIX 「登入」模組值有限。

JndiLoginModule
針對在 JNDI (Java 命名和目錄介面) 下配置的目錄服務進行鑑別。

Krb5LoginModule
使用 Kerberos 通訊協定進行鑑別。

NTLoginModule
使用現行使用者的 NT 安全資訊進行鑑別。

UnixLoginModule
使用現行使用者的 UNIX 安全資訊進行鑑別。

使用 NTLoginModule 或 UnixLoginModule 的問題是遙測 (MQXR) 服務以 mqm 身分執行，而不是以 MQTT 通道的身分執行。 mqm 是傳遞至 NTLoginModule 或 UnixLoginModule 以進行鑑別的身分，而不是用戶端的身分。

若要解決此問題，請撰寫您自己的「登入」模組，或者使用其他標準「登入」模組。 IBM MQ Telemetry 隨附範例 JAASLoginModule.java 。 它是 javax.security.auth.spi.LoginModule 介面的實作。 可以使用它來開發您自己的鑑別方法。

您提供的所有新 LoginModule 類別都必須在遙測 (MQXR) 服務的類別路徑上。 請勿將類別放在類別路徑中的 IBM MQ 目錄中。 建立您自己的目錄，並定義遙測 (MQXR) 服務的完整類別路徑。

透過在 service.env 檔案中設定類別路徑，可以擴增遙測 (MQXR) 服務使用的類別路徑。 CLASSPATH 必須大寫，並且類別路徑陳述式只能包含文字。 不能在 CLASSPATH 中使用變數；例如 CLASSPATH=%CLASSPATH% 就是不正確的。 遙測 (MQXR) 服務會設定其專屬類別路徑。 會將定義在 service.env 中的 CLASSPATH 新增至該類別路徑。

遙測 (MQXR) 服務提供兩個回呼，針對連接至 MQTT 通道的用戶端傳回 **使用者名稱** 及 **密碼** 。 **使用者名稱** 和 **密碼** 設定在 MqttConnectOptions 物件中。 請參閱第 144 頁的圖 30，以取得如何存取 Username 和 Password 的範例。

範例

具有一個具名配置 MQXRConfig 的 JAAS 配置檔範例。

```
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";
};
```

圖 *29:* 範例 *jaas.config* 檔案

JAAS 登入模組的範例編碼為接收 MQTT 用戶端提供的 **使用者名稱** 和 **密碼** 。

```
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;
}
```

圖 *30:* 範例 *JAASLoginModule.Login()* 方法

相關資訊

AuthCallback MQXR 類別

解決問題：Telemetry 服務未呼叫 JAAS 登入模組

V 8.0.0.4 管理 IBM MQ Light

您可以使用 MQ Explorer 或在指令行管理 MQ Light 。 使用「檔案總管」來配置通道，並監視連接至 IBM MQ 的 MQ Light 用戶端。 使用 TLS 和 JAAS 來配置 MQ Light 的安全。

開始之前

如需在平台上安裝 AMQP 的相關資訊，請參閱 選擇要安裝的項目。 使用 IBM MQ V8.0.0.4 原廠更新來安裝 AMQP 服務元件，而不是 V8.0.0.4 修正套件。 您無法在早於 V8.0.0.4 版的佇列管理程式版本上安裝 AMQP 元件。

使用 MQ Explorer 進行管理

使用「探險家」來配置 AMQP 通道，並監視連接至 IBM MQ 的 MQ Light 用戶端。 您可以使用 TLS 和 JAAS 來配置 MQ Light 的安全。

使用指令行管理

您可以在指令行使用 IBM MQ MQSC 指令來管理 MQ Light 。

V 8.0.0.4 檢視 MQ Light 用戶端使用中的 IBM MQ 物件

您可以檢視 MQ Light 用戶端正在使用的不同 IBM MQ 資源，例如連線及訂閱。

連線

啟動 AMQP 服務時，會建立新的 Hconns 並連接至佇列管理程式。 當 MQ Light 用戶端發佈訊息時，會使用此 Hconns 儲存區。 您可以使用 **DISPLAY CONN** 指令來檢視 Hconns。 例如：

```
DISPLAY CONN(*) TYPE(CONN) WHERE (APPLDESC LK 'WebSphere MQ Advanced Message Queuing Protocol*')
```

此指令也會顯示任何用戶端特定的 Hconns。 具有空白用戶端 ID 屬性的 Hconns 是儲存區中使用的 Hconns

當 MQ Light 用戶端連接至 AMQP 通道時，新的 Hconn 會連接至佇列管理程式。 此 Hconn 用於針對 MQ Light 用戶端已建立的訂閱非同步使用訊息。 您可以使用 **DISPLAY CONN** 指令來檢視特定 MQ Light 用戶端所使用的 Hconn。 例如：

```
DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_abcd1234')
```

用戶端建立的訂閱

當 MQ Light 用戶端訂閱主題時，會建立新的 IBM MQ 訂閱。 訂閱名稱包括下列資訊：

- 用戶端的名稱。 如果用戶端加入共用訂閱，則會使用共用名稱
- 用戶端訂閱的主題型樣
- 字首。 如果用戶端建立非共用訂閱，則字首為 `private`；如果用戶端加入共用訂閱，則字首為 `share`

若要檢視特定 MQ Light 用戶端正在使用的訂閱，請執行 **DISPLAY SUB** 指令，並對 `private` 字首進行過濾：

```
DISPLAY SUB(':private:*')
```

若要檢視多個用戶端正在使用的共用訂閱，請執行 **DISPLAY SUB** 指令並過濾 `share` 字首：

```
DISPLAY SUB(':share:*')
```

因為共用訂閱可以由多個 MQ Light 用戶端使用，所以您可能想要檢視目前使用來自共用訂閱之訊息的用戶端。 您可以透過列出目前在訂閱佇列上開啟控點的 Hconns 來執行此動作。 若要檢視目前使用共用的用戶端，請完成下列步驟：

1. 尋找共用訂閱用作目的地的佇列名稱。 例如：

   ```
   DISPLAY SUB(':private:recv_e298452:public') DEST
      5 : DISPLAY SUB(':private:recv_e298452:public') DEST
   AMQ8096: WebSphere MQ subscription inquired.
      SUBID(414D5120514D312020202020202020707E0A565C2D0020)
      SUB(:private:recv_e298452:public)
      DEST(SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)
   ```

2. 執行 **DISPLAY CONN** 指令，以尋找在該佇列上開啟的控點：

   ```
   DISPLAY CONN(*) TYPE(HANDLE) WHERE  (OBJNAME
   EQ SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)
      21 : DISPLAY CONN(*) TYPE(HANDLE) WHERE(OBJNAME EQ
   SYSTEM.MANAGED.DURABLE.560A7E7020002D5B)

   AMQ8276: Display Connection details.
      CONN(707E0A56642B0020)
      EXTCONN(414D5143514D31202020202020202020)
      TYPE(HANDLE)

      OBJNAME(SYSTEM.BASE.TOPIC)      OBJTYPE(TOPIC)

      OBJNAME(SYSTEM.MANAGED.DURABLE.560A7E7020002961)
      OBJTYPE(QUEUE)
   ```

3. 針對每一個控點，檢視已開啟控點的 MQ Light 用戶端 ID:

```
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 8.0.0.4 MQ Light 用戶端識別、授權及鑑別

與其他 IBM MQ 用戶端應用程式一樣，您可以透過多種方式來保護 AMQP 連線的安全。

您可以使用下列安全特性來保護 IBM MQ 的 AMQP 連線安全:

- 通道鑑別記錄
- 連線鑑別
- 通道 MCA 使用者配置
- IBM MQ 權限定義
- TLS 連線功能

從安全角度來看，建立連線包含下列兩個步驟:

- 決定連線是否應繼續
- 決定應用程式假設稍後進行權限檢查的 IBM MQ 身分

下列資訊概述不同的 IBM MQ 配置，以及 AMQP 用戶端嘗試建立連線時所完成的步驟。 並非所有 IBM MQ 配置都使用所有說明的步驟。 例如，部分配置不會將 TLS 用於公司防火牆內的連線，部分配置會使用 TLS，但不會使用用戶端憑證進行鑑別。 許多環境不使用自訂或自訂 JAAS 模組。

建立連線

下列步驟說明 AMQP 用戶端正在建立連線時所發生的情況。 這些步驟決定連線是否繼續，以及應用程式假設進行權限檢查的 IBM MQ 身分:

1. 如果用戶端開啟與 IBM MQ 的 TLS 連線並提供憑證，則佇列管理程式會嘗試驗證用戶端憑證。
2. 如果用戶端提供使用者名稱及密碼認證，則佇列管理程式會接收 AMQP SASL 訊框，並檢查 MQ CONNAUTH 配置。
3. 會檢查 MQ 通道鑑別規則 (例如， IP 位址及 TLS 憑證 DN 是否有效)
4. 除非通道鑑別規則另有決定，否則會主張通道 MCAUSER。
5. 如果已配置 JAAS 模組，則會呼叫它
6. MQ CONNECT 權限檢查套用至產生的 MQ 使用者 ID。
7. 以假設的 IBM MQ 身分建立連線。

發佈訊息

下列步驟說明 AMQP 用戶端發佈訊息時所發生的情況。 這些步驟決定連線是否繼續，以及應用程式假設進行權限檢查的 IBM MQ 身分:

1. AMQP 鏈結連接訊框到達佇列管理程式。 會針對連線期間所建立的 MQ 使用者身分，檢查指定主題字串的 IBM MQ 發佈權限。

2. 訊息已發佈至指定的主題字串。

訂閱主題型樣

下列步驟說明 AMQP 用戶端訂閱主題型樣時發生的情況。 這些步驟決定連線是否繼續，以及應用程式假設進行權限檢查的 IBM MQ 身分:

1. AMQP 鏈結連接訊框到達佇列管理程式。 會針對連線期間所建立的 MQ 使用者身分，檢查指定主題型樣的 IBM MQ 訂閱權限。
2. 已建立訂閱。

V 8.0.0.4 MQ Light 用戶端身分及授權

使用通道上或通道鑑別規則中定義的 MQ Light 用戶端 ID、 MQ Light 使用者名稱或一般用戶端身分，以取得存取 IBM MQ 物件的授權。

管理者在定義或修改 AMQP 通道、配置佇列管理程式 CONNAUTH 設定或定義通道鑑別規則時做出選擇。 身分用於授與 IBM MQ 主題的存取權。 根據下列各項來進行選擇:

1. 通道 USECLNTID 屬性。
2. 佇列管理程式 CONNAUTH 規則的 ADOPTCTX 屬性。
3. 通道上定義的 MCAUSER 屬性。
4. 相符通道鑑別規則的 USERSRC 屬性。

避免麻煩: 之後會參照此處理程序所選擇的身分，例如 DISPLAY CHSTATUS (AMQP) 指令，作為用戶端的 MCAUSER。 請注意，這不一定與選項 (2) 中所參照通道的 MCAUSER 相同。

使用 IBM MQ **setmqaut** 指令來選取與 AMQP 通道相關聯的身分所授權使用的物件及動作。 例如，下列指令會授權佇列管理程式 QM1 的管理者所提供的通道身分 AMQPClient:

```
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 8.0.0.4 MQ Light 使用密碼進行用戶端鑑別

使用用戶端密碼鑑別 MQ Light 使用者名稱。 您可以使用不同於用來授權用戶端發佈及訂閱主題之身分的身分來鑑別用戶端。

AMQP 服務可以使用 MQ CONNAUTH 或 JAAS 來鑑別用戶端使用者名稱。 如果已配置其中一個，則 MQ CONNAUTH 配置或 JAAS 模組會驗證用戶端提供的密碼。

下列程序概述針對本端 OS 使用者及密碼鑑別個別使用者的範例步驟，如果成功，則採用一般身分 AMQPUser:

1. IBM MQ 管理者會使用「IBM MQ 探險家」將 AMQP 通道 MCAUSER 身分設為任何名稱，例如 AMQPUser。
2. IBM MQ 管理者授權 AMQPUser 發佈及訂閱任何主題:

   ```
   setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p AMQPUser -all +pub +sub +connect
   ```

3. IBM MQ 管理者會配置 IDPWOS CONNAUTH 規則，以檢查用戶端提供的使用者名稱及密碼。 CONNAUTH 規則應該設定 CHCKCLNT (REQUIRED) 及 ADOPTCTX (NO)。

註: 建議您使用通道鑑別規則，並將 MCAUSER 通道屬性設為沒有專用權的使用者，以容許對佇列管理程式連線有更多控制權。

V 8.0.0.4 通道上的發佈隱私權

透過使用 TLS 來加密透過連線的傳輸，可保護透過任一方向在 AMQP 通道中傳送的 AMQP 發佈資訊的隱私權。

連接至 AMQP 通道的 AMQP 用戶端會使用 TLS 來保護通道上使用對稱金鑰加密法所傳輸發佈的隱私權。 因為不會鑑別端點，所以您不能信任單獨使用的通道加密。 將保密安全與伺服器或交互鑑別結合使用。

作為使用 TLS 的替代方案，部分類型的「虛擬專用網路 (VPN)」(例如 IPsec) 會鑑別 TCP/IP 連線的端點。 VPN 會加密流經網路的每個 IP 封包。 一旦建立這類 VPN 連線，即建立可信的網路。 您可以透過 VPN 網路使用 TCP/IP 將 AMQP 用戶端連接至 AMQP 通道。

在不鑑別伺服器的情況下加密 TLS 連線會對中間人攻擊公開連線。 雖然您交換的資訊可以防竊聽，但是您不知道與您交換資訊的對象是誰。 除非您控制網路，否則您就會面臨別人截取您的 IP 傳輸，以及假冒端點的問題。

您可以使用支援匿名 TLS 的 Diffie-Hellman 金鑰交換 CipherSpec 來建立已加密 TLS 連線，而無需鑑別伺服器。 在用戶端與伺服器之間共用且用來加密 TLS 傳輸的主要密碼，是在不交換私密簽署伺服器憑證的情況下建立的。

由於匿名連線不安全，大部分 TLS 實作不會預設為使用匿名 CipherSpecs。 如果 AMQP 通道接受 TLS 連線的用戶端要求，則通道必須具有受通行詞組保護的金鑰儲存庫。 依預設，由於 TLS 實作不使用匿名 CipherSpecs，金鑰儲存庫必須包含用戶端可以鑑別的私密簽章憑證。

如果您使用匿名 CipherSpec，則伺服器金鑰儲存庫必須存在，但是它不必包含任何私密簽章的憑證。

建立加密連線的另一種方法，是利用您自己的實作，取代用戶端上的信任提供者。 您的信任提供者不會鑑別伺服器憑證，但是會加密連線。

V 8.0.0.4 使用 TLS 配置 MQ Light 用戶端

您可以將 MQ Light 用戶端配置為使用 TLS 來保護流經網路的資料，以及鑑別用戶端所連接的佇列管理程式身分。

若要將 TLS 用於從 MQ Light 用戶端到 AMQP 通道的連線，您必須確保佇列管理程式已配置為 TLS。 在佇列管理程式上配置 SSL 說明如何配置佇列管理程式從中讀取 TLS 憑證的金鑰儲存庫。

當佇列管理程式已配置金鑰儲存庫時，您必須在用戶端將連接的 AMQP 通道上配置 TLS 屬性。 AMQP 通道具有四個與 TLS 配置相關的屬性，如下所示:

SSLCAUTH
SSLCAUTH 屬性用來指定佇列管理程式是否應要求 MQ Light 用戶端提供用戶端憑證以驗證其身分。

SSLCIPH
SSLCIPH 屬性指定通道應該用來編碼 TLS 流程中資料的密碼。

SSLPEER
SSLPEER 屬性用來指定如果要容許連線，用戶端憑證必須符合的識別名稱 (DN)。

CERTLABL
CERTLABL 指定佇列管理程式應該呈現給用戶端的憑證。 佇列管理程式的金鑰儲存庫可以包含多個憑證。 此屬性可讓您指定要用於此通道連線的憑證。 如果未指定 CERTLABL，則會使用佇列管理程式金鑰儲存庫中標籤對應於佇列管理程式 CERTLABL 屬性的憑證。

當您已使用 TLS 屬性配置 AMQP 通道時，必須使用下列指令重新啟動 AMQP 服務:

```
STOP SERVICE(SYSTEM.AMQP.SERVICE) START SERVICE(SYSTEM.AMQP.SERVICE)
```

當 MQ Light 用戶端連接至受 TLS 保護的 AMQP 通道時，用戶端會驗證佇列管理程式所提供憑證的身分。 若要這樣做，您必須使用包含佇列管理程式憑證的信任儲存庫來配置 MQ Light 用戶端。 執行此動作的步驟視您使用的 MQ Light 用戶端而定。

- 如需 MQ Light Client for Node JS API 文件，請參閱 https://www.npmjs.com/package/mqlight
- 如需 MQ Light Client for Java API 文件，請參閱 https://mqlight.github.io/java-mqlight/
- 如需 MQ Light Client for Ruby 說明文件，請參閱 https://www.rubydoc.info/github/mqlight/ruby-mqlight/

- 如需 Python 的 MQ Light 用戶端說明文件，請參閱 https://python-mqlight.readthedocs.org/en/latest/

V 8.0.0.4 切斷 MQ Light 用戶端與佇列管理程式的連線

如果您要中斷 MQ Light 與佇列管理程式的連線，請執行 PURGE CHANNEL 指令或停止與 MQ Light 用戶端的連線。

- 執行 **PURGE CHANNEL** 指令。 例如：

```
PURGE CHANNEL(MYAMQP) CLIENTID('recv_28dbb7e')
```

- 或者，完成下列步驟，以停止 MQ Light 用戶端用來中斷用戶端連線的連線：

1. 執行 **DISPLAY CONN** 指令，以尋找用戶端正在使用的連線。 例如：

```
DISPLAY CONN(*) TYPE(CONN) WHERE (CLIENTID EQ 'recv_28dbb7e')
```

指令輸出如下：

```
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. 停止連線。 例如：

```
STOP CONN(707E0A565F2D0020)
```

管理多重播送

使用此資訊來瞭解「IBM MQ 多重播送」管理作業，例如減少多重播送訊息的大小及啟用資料轉換。

開始使用多重播送

使用此資訊來開始使用 IBM MQ Multicast 主題及通訊資訊物件。

關於這項作業

IBM MQ Multicast 傳訊透過將主題對映至群組位址，使用網路來遞送訊息。 下列作業是快速測試所需的 IP 位址和埠是否已正確配置多重播送傳訊的方法。

建立多重播送的 COMMINFO 物件

通訊資訊 (COMMINFO) 物件包含與多重播送傳輸相關聯的屬性。 如需 COMMINFO 物件參數的相關資訊，請參閱 DEFINE COMMINFO。

請使用下列指令行範例來定義多重播送的 COMMINFO 物件：

```
DEFINE COMMINFO(MC1) GRPADDR(group address) PORT(port number)
```

其中 *MC1* 是 COMMINFO 物件的名稱， *group address* 是您的群組多重播送 IP 位址或 DNS 名稱， *port number* 是要傳輸的埠 (預設值為 1414)。

會建立稱為 *MC1* 的新 COMMINFO 物件; 此名稱是您在下一個範例中定義 TOPIC 物件時必須指定的名稱。

建立 TOPIC 物件以進行多重播送

主題是發佈/訂閱訊息中所發佈資訊的主旨，而主題是透過建立 TOPIC 物件來定義。 TOPIC 物件有兩個參數，可定義它們是否可以與多重播送一起使用。 這些參數為: **COMMINFO** 和 **MCAST**。

- **COMMINFO** 此參數指定多重播送通訊資訊物件的名稱。 如需 COMMINFO 物件參數的相關資訊，請參閱 DEFINE COMMINFO。
- **MCAST** 此參數指定主題樹狀結構中的這個位置是否容許多重播送。

使用下列指令行範例來定義多重播送的 TOPIC 物件:

```
DEFINE TOPIC(ALLSPORTS) TOPICSTR('Sports') COMMINFO(MC1) MCAST(ENABLED)
```

即會建立稱為 *ALLSPORTS* 的新 TOPIC 物件。 它具有主題字串 *Sports*，其相關通訊資訊物件稱為 *MC1* (在前一個範例中定義 COMMINFO 物件時指定的名稱) ，並且已啟用多重播送。

測試多重播送發佈/訂閱

建立 TOPIC 和 COMMINFO 物件之後，可以使用 amqspubc 範例和 amqssubc 範例來測試它們。 如需這些範例的相關資訊，請參閱 發佈/訂閱範例程式。

1. 開啟兩個指令行視窗; 第一個指令行用於 amqspubc 發佈範例，第二個指令行用於 amqssubc 訂閱範例。
2. 在指令行 1 輸入下列指令:

   ```
   amqspubc Sports QM1
   ```

   其中 *Sports* 是先前範例中所定義 TOPIC 物件的主題字串，而 *QM1* 是佇列管理程式的名稱。
3. 在指令行 2 輸入下列指令:

   ```
   amqssubc Sports QM1
   ```

   其中 體育 和 *QM1* 與步驟 第 150 頁的『2』中使用的相同。
4. 在指令行 1 輸入 `Hello world`。 如果 COMMINFO 物件中指定的埠和 IP 位址已正確配置; 則 amqssubc 範例會在埠上接聽來自指定位址的發佈，並在指令行 2 輸出 `Hello world`。

IBM MQ 多重播送主題拓蹼

使用此範例來瞭解 IBM MQ Multicast 主題拓蹼。

IBM MQ 多重播送支援需要每個子樹狀結構在總階層內都有自己的多重播送群組和資料串流。

具類別網路 IP 定址方法對於多重播送位址具有指定的位址空間。 完整的多重播送 IP 位址範圍是 `224.0.0.0` 到 `239.255.255.255`，但其中有些位址已保留。 如需保留位址清單，請聯絡您的系統管理者，或參閱 https://www.iana.org/assignments/multicast-addresses 以取得相關資訊。 建議您使用 `239.0.0.0` 至 `239.255.255.255` 範圍內的本端範圍多重播送位址。

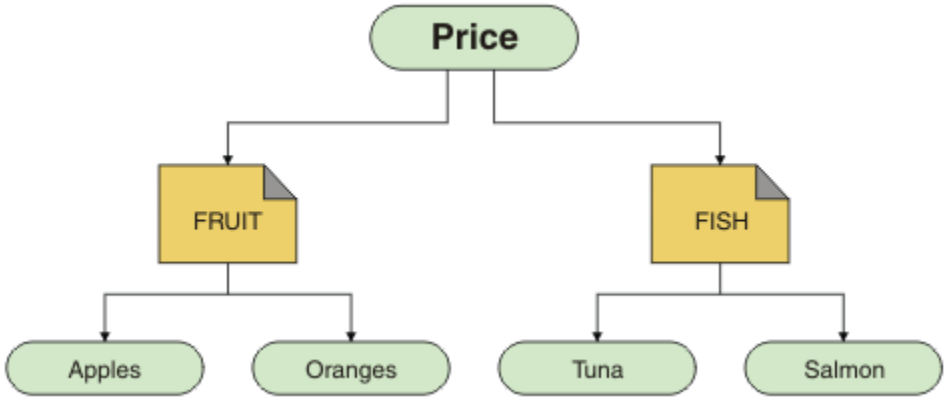
在下圖中，有兩個可能的多重播送資料串流:

```
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* 是有效的多重播送位址。

這些主題定義用來建立主題樹狀結構，如下圖所示:

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)

DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



每一個多重播送通訊資訊 (COMMINFO) 物件都代表不同的資料串流，因為它們的群組位址不同。 在此範例中， FRUIT 主題定義為使用 COMMINFO 物件 MC1 ， FISH 主題定義為使用 COMMINFO 物件 MC2 ， 並且 Price 節點沒有多重播送定義。

IBM MQ 多重播送的主題字串限制為 255 個字元。 此限制表示必須小心處理樹狀結構內節點及葉節點的名稱; 如果節點及葉節點的名稱太長，則主題字串可能會超出 255 個字元，並傳回 2425 (0979) (RC2425) :MQRC_TOPIC_STRING_ERROR 原因碼。 建議儘量縮短主題字串，因為較長的主題字串可能會對效能造成不利影響。

控制多重播送訊息的大小

使用此資訊來瞭解 IBM MQ 訊息格式，並減少 IBM MQ 訊息的大小。

IBM MQ 訊息具有一些相關聯的屬性，這些屬性包含在訊息描述子中。 對於小型訊息，這些屬性可能代表大部分資料流量，並且可能對傳輸速率產生重大不利影響。 IBM MQ Multicast 可讓使用者配置隨訊息一起傳輸這些屬性 (如果有的話)。

訊息屬性 (非主題字串) 的存在取決於 COMMINFO 物件是否指出必須傳送它們。 如果未傳輸屬性，接收端應用程式會套用預設值。 預設 MQMD 值不一定與 MQMD_DEFAULT 值相同，並在 第 152 頁的表 9 中說明。

COMMINFO 物件包含 MCPROP 屬性，可控制有多少 MQMD 欄位及使用者內容與訊息一起流動。 透過將此屬性的值設為適當的層次，您可以控制「IBM MQ 多重播送」訊息的大小:

MCPROP
多重播送內容控制與訊息一起傳送的 MQMD 內容及使用者內容數。

ALL
會傳輸 MQMD 的所有使用者內容及所有欄位。

回覆
只傳輸使用者內容，以及處理訊息回覆的 MQMD 欄位。 這些內容如下:

- MsgType
- MessageId
- CorrelId
- ReplyToQ
- ReplyToQmgr

使用者
只傳輸使用者內容。

無
不傳輸任何使用者內容或 MQMD 欄位。

COMPAT
此值會導致以相容模式將訊息傳輸至 RMM，這容許與現行 XMS 應用程式及 IBM Integration Bus RMM 應用程式進行一些交互作業。

多重播送訊息屬性

訊息屬性可以來自各種地方，例如 MQMD、 MQRFH2 中的欄位，以及訊息內容。

下表顯示傳送受 MCPROP 值限制的訊息時所發生的情況，以及未傳送屬性時所使用的預設值。

表 9: 傳訊屬性及其與多重播送的關係

屬性	使用多重播送時的動作	如果未傳輸，則為預設值
TopicString	一律併入	不適用
MQMQ StrucId	未傳輸	不適用
MQMD 版本	未傳輸	不適用
報告	如果不是預設值則併入	0
MsgType	如果不是預設值則併入	MQMT_DATAGRAM
期限	如果不是預設值則併入	0
意見	如果不是預設值則併入	0
編碼	如果不是預設值則併入	MQENC_NORMAL (equiv)
CodedCharSetId	如果不是預設值則併入	1208
格式	如果不是預設值則併入	MQRFH2
優先順序	如果不是預設值則併入	4
持續性	如果不是預設值則併入	MQPER_NOT_PERSISTENT
MsgId	如果不是預設值則併入	空值
CorrelId	如果不是預設值則併入	空值
BackoutCount	如果不是預設值則併入	0
ReplyToQ	如果不是預設值則併入	Blank
回覆目的地佇列管理程式	如果不是預設值則併入	Blank
UserIdentifier	如果不是預設值則併入	Blank
AccountingToken	如果不是預設值則併入	空值
PutAppIType	如果不是預設值則併入	MQAT_JAVA
PutApp 名稱	如果不是預設值則併入	Blank
PutDate	如果不是預設值則併入	Blank
PutTime	如果不是預設值則併入	Blank
ApplOriginData	如果不是預設值則併入	Blank
GroupID	已排除	不適用
MsgSeqNumber	已排除	不適用
偏移	已排除	不適用
MsgFlags	已排除	不適用
OriginalLength	已排除	不適用
UserProperties	已包括	不適用

相關參考
ALTER COMMINFO
DEFINE COMMINFO

啟用多重播送傳訊的資料轉換

使用此資訊來瞭解 IBM MQ Multicast 傳訊的資料轉換如何運作。

IBM MQ Multicast 是一種共用的無連線通訊協定，因此每一個用戶端都無法提出特定要求來進行資料轉換。每個訂閱相同多重播送串流的用戶端都會接收相同的二進位資料; 因此，如果需要 IBM MQ 資料轉換，則會在每個用戶端本端執行轉換。

在混合平台安裝中，可能是大部分用戶端需要的資料格式不是傳輸應用程式的原始格式。 在此情況下，可以使用多重播送 COMMINFO 物件的 **CCSID** 及 **ENCODING** 值來定義訊息傳輸的編碼，以提高效率。

IBM MQ Multicast 支援下列內建格式的訊息有效負載的資料轉換:

- MQADMIN
- MQEVENT
- MQPCF
- MQRFH
- MQRFH2
- MQSTR

除了這些格式之外，您還可以定義自己的格式，並使用 MQDXP-資料轉換結束程式參數 資料轉換結束程式。

如需程式設計資料轉換的相關資訊，請參閱 MQI for 多重播送傳訊中的資料轉換。

如需資料轉換的相關資訊，請參閱 資料轉換。

如需資料轉換結束程式及 ClientExitPath 的相關資訊，請參閱 用戶端配置檔的 ClientExit 路徑段落。

多重播送應用程式監視

使用此資訊來瞭解管理及監視 IBM MQ Multicast。

多重播送資料流量的現行發佈者及訂閱者的狀態 (例如，傳送及接收的訊息數，或遺失的訊息數) 會定期從用戶端傳輸至伺服器。 收到狀態時， COMMINFO 物件的 COMMEV 屬性會指定佇列管理程式是否將事件訊息放置在 SYSTEM.ADMIN.PUBSUB.EVENT。 事件訊息包含收到的狀態資訊。 此資訊是尋找問題來源的寶貴診斷輔助。

使用 MQSC 指令 **DISPLAY CONN** 可顯示連接至佇列管理程式之應用程式的連線資訊。 如需 **DISPLAY CONN** 指令的相關資訊，請參閱 DISPLAY CONN。

使用 MQSC 指令 **DISPLAY TPSTATUS** 來顯示發佈者和訂閱者的 狀態。 如需 **DISPLAY TPSTATUS** 指令的相關資訊，請參閱 DISPLAY TPSTATUS。

COMMEV 及多重播送訊息可靠性指示器

可靠性指示器與 COMMINFO 物件的 **COMMEV** 屬性一起使用，是監視 IBM MQ Multicast 發佈者和訂閱者的關鍵元素。 可靠性指示器 (在「發佈」或「訂閱」狀態指令上傳回的 **MSGREL** 欄位) 是 IBM MQ 指示器，說明沒有錯誤的傳輸百分比。有時由於傳輸錯誤而必須重新傳輸訊息，這會反映在 **MSGREL** 的值中。 傳輸錯誤的潛在原因包括使用者緩慢、網路忙碌及網路中斷。 **COMMEV** 控制是否針對使用 COMMINFO 物件所建立的多重播送控點，產生事件訊息，並設為下列三個可能值之一:

已停用
不會寫入事件訊息。

ENABLED
一律寫入事件訊息，並在 COMMINFO **MONINT** 參數中定義頻率。

異常狀況
如果訊息可靠性低於可靠性臨界值，則會寫入事件訊息。 90% 或更小的訊息可靠性層次指出網路配置可能有問題，或一個以上「發佈/訂閱」應用程式執行太慢:

- 值 **MSGREL(100,100)** 表示在短期或長期時間範圍內沒有任何問題。
- 值 **MSGREL(80,60)** 表示 20% 的訊息目前有問題，但它也比長期值 60 有所改善。

即使佇列管理程式的單點播送連線中斷，用戶端仍可能繼續傳輸及接收多重播送資料流量，因此資料可能過期。

多重播送訊息可靠性

使用此資訊來瞭解如何設定 IBM MQ Multicast 訂閱及訊息歷程。

克服多重播送傳輸失敗的關鍵元素是 IBM MQ 對已傳輸資料 (要保留在鏈結傳輸端的訊息歷程) 的緩衝。 此處理程序表示在放置應用程式程序中不需要緩衝訊息，因為 IBM MQ 提供可靠性。 此歷程的大小是透過通訊資訊 (COMMINFO) 物件來配置，如下列資訊中所述。 較大的傳輸緩衝意味著在需要時需要重新傳輸的傳輸歷程更多，但由於多重播送的本質，無法支援 100% 的保證遞送。

IBM MQ Multicast 訊息歷程在通訊資訊 (COMMINFO) 物件中由 **MSGHIST** 屬性控制:

MSGHIST
此值是系統保留來處理在 ACK (負值確認通知) 情況下重新傳輸的訊息歷程數量 (以 KB 為單位)。

值 0 會提供最低可靠性層次。 預設值為 100 KB。

IBM MQ Multicast 新訂閱歷程在通訊資訊 (COMMINFO) 物件中由 **NSUBHIST** 屬性控制:

NSUBHIST
新訂閱者歷程控制加入發佈串流的訂閱者是否收到目前所有可用的資料，或只收到訂閱後的發佈。

無
NONE 值會導致轉送器僅傳輸從訂閱時間開始的發佈。 NONE 是預設值。

ALL
ALL 值會導致轉送器重新傳輸已知的主題歷程。 在某些情況下，此狀況會對保留的發佈提供類似的行為。

註: 如果因為重新傳輸所有主題歷程而有大型主題歷程，則使用 ALL 值可能會對效能造成不利影響。

相關資訊
DEFINE COMMINFO
ALTER COMMINFO

進階多重播送作業

使用此資訊來瞭解進階 IBM MQ Multicast 管理作業，例如配置 .ini 檔案以及與 IBM MQ LLM 的交互作業能力。

如需多重播送安裝中的安全考量，請參閱 多重播送安全。

在多重播送與非多重播送發佈/訂閱網域之間橋接

使用此資訊來瞭解當非多重播送發佈者發佈至已啟用 IBM MQ 多重播送的主題時所發生的情況。

如果非多重播送發佈者發佈至定義為 **MCAST** 已啟用且 **BRIDGE** 已啟用的主題，則佇列管理程式會透過多重播送直接將訊息傳輸出至任何可能正在接聽的訂閱者。 多重播送發佈者無法發佈至未啟用多重播送的主題。

透過設定主題物件的 **MCAST** 及 **COMMINFO** 參數，可以啟用多重播送現有主題。 如需這些參數的相關資訊，請參閱 起始多重播送概念 。

COMMINFO 物件 **BRIDGE** 屬性控制來自未使用多重播送之應用程式的發佈。 如果 **BRIDGE** 設為 ENABLED ，且主題的 **MCAST** 參數也設為 ENABLED，則來自未使用多重播送之應用程式的發佈會橋接至所使用的應用程式。 如需 **BRIDGE** 參數的相關資訊，請參閱 DEFINE COMMINFO。

配置「多重播送」的 .ini 檔

使用此資訊來瞭解 .ini 檔案中的 IBM MQ 多重播送欄位。

可以在 ini 檔案中進行其他 IBM MQ 多重播送配置。 您必須使用的特定 ini 檔案取決於應用程式類型:

- 用戶端: 配置 *MQ_DATA_PATH* /mqclient.ini 檔案。
- 佇列管理程式: 配置 *MQ_DATA_PATH* /qmgrs/*QMNAME*/qm.ini 檔案。

其中 *MQ_DATA_PATH* 是 IBM MQ 資料目錄 (/var/mqm/mqclient.ini) 的位置， *QMNAME* 是套用 .ini 檔案的佇列管理程式名稱。

.ini 檔案包含用來細部調整 IBM MQ 多重播送 行為的欄位:

```
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
```

通訊協定

UDP
在此模式中，會使用 UDP 通訊協定來傳送封包。 然而，網路元素無法像在 IP 模式中一樣在多重播送中提供協助。 封包格式仍與 PGM 相容。 這是預設值。

IP
在此模式中，轉送器會傳送原始 IP 封包。 具有 PGM 支援的網路元素可協助進行可靠的多重播送封包配送。 此模式與 PGM 標準完全相容。

IPVersion

IPv4
僅使用 IPv4 通訊協定進行通訊。 這是預設值。

IPv6
僅使用 IPv6 通訊協定進行通訊。

ANY
視可用的通訊協定而定，使用 IPv4 及/或 IPv6 進行通訊。

兩者
支援使用 IPv4 和 IPv6 進行通訊。

LimitTrans 率

已停用
沒有傳輸速率控制。 這是預設值。

靜態
實作靜態傳輸速率控制。 轉送器不會以超出 TransRate 限制參數所指定速率的速率進行傳輸。

動態
發射機根據從接收機獲得的反饋來調整其傳輸速率。 在此情況下，傳輸速率限制不能大於 TransRateLimit 參數指定的值。 傳送器嘗試達到最佳傳輸速率。

TransRate 限制
傳輸速率限制 (以 Kbps 為單位)。

SocketTTL
SocketTTL 的值決定多重播送資料流量是否可以通過路由器，或它可以通過的路由器數目。

批次
控制是批次處理還是立即傳送訊息。有 2 個可能的值:

- *NO* 訊息不會批次處理，會立即傳送。
- *YES* 訊息已批次處理。

重複播放
將值設為 1 可啟用多重播送迴圈。 多重播送迴圈定義傳送的資料是否迴圈回主電腦。

介面
多重播送資料流量在其上流動之介面的 IP 位址。 如需相關資訊及疑難排解，請參閱: 在非多重播送網路上測試多重播送應用程式 及 針對多重播送資料流量設定適當的網路

FeedbackMode

NACK
負確認通知的意見。 這是預設值。

ACK
正面確認通知的意見。

WAIT1
由正面確認通知所提供的回饋，其中轉送器只會等待來自任何接收端的 1 個 ACK。

HeartbeatTimeout
活動訊號逾時 (毫秒)。 值 0 表示主題的一或多個接收端不會產生活動訊號逾時事件。 預設值為 20000。

HeartbeatInterval
活動訊號間隔 (毫秒)。 值 0 表示不傳送活動訊號。 活動訊號間隔必須遠小於 **HeartbeatTimeout** 值，以避免錯誤活動訊號逾時事件。 預設值為 2000。

多重播送與 IBM MQ 低延遲傳訊的交互作業能力

使用此資訊來瞭解 IBM MQ 多重播送與 IBM MQ 低延遲傳訊 (LLM) 之間的交互作業能力。

對於使用 LLM 的應用程式而言，基本有效負載傳送是可能的，而另一個應用程式則使用多重播送來雙向交換訊息。 雖然多重播送使用 LLM 技術，但 LLM 產品本身並未內嵌。 因此，可以同時安裝 LLM 和 IBM MQ Multicast ，並分別操作和服務這兩個產品。

與多重播送通訊的 LLM 應用程式可能需要傳送及接收訊息內容。 IBM MQ 訊息內容及 MQMD 欄位會以 LLM 訊息內容傳輸，並具有下表所示的特定 LLM 訊息內容碼:

表 *10: IBM MQ* 訊息內容至 *IBM MQ LLM* 內容對映

IBM MQ 內容 (property)	**IBM MQ LLM 內容類型**	**LLM 內容類型**	**LLM 內容碼**
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
MQMD.PutDate	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1020
MQMD.PutTime	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1021

表 10: IBM MQ 訊息內容至 IBM MQ LLM 內容對映 (繼續)			
IBM MQ 內容 (property)	**IBM MQ LLM 內容類型**	**LLM 內容類型**	**LLM 內容碼**
MQMD.ApplOriginData	RMM_MSG_PROP_BYTES	LLM_PROP_KIND_String	-1022
MQPubOptions	RMM_MSG_PROP_INT32	LLM_PROP_KIND_int32	-1053

如需 LLM 的相關資訊，請參閱 LLM 產品說明文件: IBM MQ 低延遲傳訊。

管理 HP Integrity NonStop Server

使用此資訊來瞭解 HP Integrity NonStop Server 的 IBM MQ 用戶端的管理作業。

您可以使用兩項管理作業:

1. 從 Pathway 手動啟動 TMF/ 閘道。
2. 從 Pathway 停止 TMF/ 閘道。

從 Pathway 手動啟動 TMF/ 閘道

您可以容許 Pathway 在第一個列入要求時自動啟動 TMF/ 閘道，也可以從 Pathway 手動啟動 TMF/ 閘道。

程序

若要從 Pathway 手動啟動 TMF/ 閘道，請輸入下列 PATHCOM 指令:

```
START SERVER <server_class_name>
```

如果用戶端應用程式在 TMF/ 閘道完成回復不確定的交易之前提出列入要求，則該要求最多會保留 1 秒。 如果回復未在該時間內完成，則會拒絕列入。 然後，用戶端會從使用交易式 MQI 收到 MQRC_UOW_ENLISTMENT_ERROR 錯誤。

從 Pathway 停止 TMF/ 閘道

此作業說明如何從 Pathway 停止 TMF/ 閘道，以及如何在停止之後重新啟動 TMF/ 閘道。

程序

1. 若要防止對 TMF/ 閘道提出任何新的列入要求，請輸入下列指令:

   ```
   FREEZE SERVER <server_class_name>
   ```

2. 若要觸發 TMF/ 閘道完成任何進行中作業並結束，請輸入下列指令:

   ```
   STOP SERVER <server_class_name>
   ```

3. 若要容許 TMF/ 閘道在第一次列入時自動重新啟動或手動重新啟動，請遵循步驟 1 及 2，輸入下列指令:

   ```
   THAW SERVER <server_class_name>
   ```

   系統會阻止應用程式提出新的列入要求，且除非您發出 **THAW** 指令，否則無法發出 **START** 指令。

IBM i 管理 IBM i

介紹可讓您在 IBM i 上管理 IBM MQ 的方法。

管理作業包括建立、啟動、變更、檢視、停止及刪除叢集、處理程序及 IBM MQ 物件 (佇列管理程式、佇列、名稱清單、處理程序定義、通道、用戶端連線通道、接聽器、服務及鑑別資訊物件)。

如需如何管理 IBM MQ for IBM i 的詳細資料，請參閱下列鏈結:

- 第 158 頁的『使用 CL 指令管理 IBM MQ for IBM i』
- 第 170 頁的『管理 IBM MQ for IBM i 的替代方式』
- 第 174 頁的『工作管理』

相關概念

第 180 頁的『可用性、備份、回復及重新啟動』
使用此資訊來瞭解 IBM MQ for IBM i 如何使用 IBM i 日誌登載支援來協助其備份及還原策略。

相關參考

第 217 頁的『靜止 IBM MQ for IBM i』
本節說明如何靜止 (循序結束) IBM MQ for IBM i

相關資訊

變更 IBM i 上的配置資訊
瞭解 IBM MQ for IBM i 佇列管理程式檔案庫名稱
在 IBM i 上設定安全
IBM i 上無法傳送郵件的佇列處理程式
判斷 IBM MQ for IBM i 應用程式的問題
IBM i 上的可安裝服務及元件
IBM i 上的系統及預設物件

使用 CL 指令管理 IBM MQ for IBM i

使用此資訊來瞭解 IBM MQ IBM i 指令。

大部分 IBM MQ 指令群組 (包括與佇列管理程式、佇列、主題、通道、名稱清單、程序定義及鑑別資訊物件相關聯的指令) 都可以使用相關 **WRK*** 指令來存取。

該集中的主體指令是 **WRKMQM**。 例如，此指令可讓您顯示系統上所有佇列管理程式的清單，以及狀態資訊。或者，您可以針對每一個項目使用各種選項來處理所有佇列管理程式特定的指令。

從 **WRKMQM** 指令中，您可以選取每一個佇列管理程式的特定區域 (例如，使用通道、主題或佇列)，並從中選取個別物件。

記錄 IBM MQ 應用程式定義

當您建立或自訂 IBM MQ 應用程式時，保留所建立的所有 IBM MQ 定義的記錄非常有用。 此記錄可用於:

- 回復目的
- 維護
- 推出 IBM MQ 應用程式

您可以使用下列兩種方式之一來記錄 IBM MQ 應用程式定義:

1. 建立 CL 程式以產生伺服器的 IBM MQ 定義。
2. 建立 MQSC 文字檔作為 SRC 成員，以使用跨平台 IBM MQ 指令語言來產生 IBM MQ 定義。

如需定義佇列物件的進一步詳細資料，請參閱 第 67 頁的『Script (MQSC) 指令』 及 第 10 頁的『使用可程式指令格式』。

開始使用 IBM MQ for IBM i 之前，請使用 CL 指令

使用此資訊來啟動 IBM MQ 子系統並建立本端佇列管理程式。

開始之前

請確定 IBM MQ 子系統正在執行中 (使用指令 `STRSBS QMQM/QMQM`)，且未保留與該子系統相關聯的工作佇列。 依預設，在檔案庫 `QMQM` 中， IBM MQ 子系統及工作佇列都命名為 `QMQM`。

關於這項作業
使用 IBM i 指令行啟動佇列管理程式

程序

1. 從 IBM i 指令行發出 CRTMQM 指令來建立本端佇列管理程式。

   當您建立佇列管理程式時，您可以選擇讓該佇列管理程式成為預設佇列管理程式。 如果省略佇列管理程式名稱參數 (MQMNAME)，則預設佇列管理程式 (只能有一個) 是 CL 指令適用的佇列管理程式。
2. 從 IBM i 指令行發出 STRMQM 指令，以啟動本端佇列管理程式。

   如果佇列管理程式啟動花費數秒以上的時間， IBM MQ 會顯示間歇性詳細說明啟動進度的狀態訊息。 如需這些訊息的相關資訊，請參閱 訊息及原因碼。

下一步

您可以從 IBM i 指令行發出 ENDMQM 指令來停止佇列管理程式，並從 IBM i 指令行發出其他 IBM MQ 指令來控制佇列管理程式。

遠端佇列管理程式無法從遠端啟動，但必須由本端操作員在其系統中建立並啟動。 但有遠端作業機能 (IBM MQ for IBM i 外部) 可啟用這類作業的情況例外。

本端佇列管理者無法停止遠端佇列管理程式。

註: 在靜止 IBM MQ 系統的過程中，您必須靜止作用中的佇列管理程式。 這說明於第 217 頁的『靜止 IBM MQ for IBM i』。

建立 IBM MQ for IBM i 物件

使用此資訊來瞭解為 IBM i 建立 IBM MQ 物件的方法。

開始之前
下列作業建議從指令行使用 IBM MQ for IBM i 的各種方式。

關於這項作業

有兩種線上方法可建立 IBM MQ 物件:

程序

1. 使用 Create 指令，例如: **Create MQM Queue** 指令: **CRTMQMQ**
2. 使用「使用 MQM 物件」指令，後面接著 F6，例如: **Work with MQM Queues** 指令: **WRKMQMQ**

下一步

如需所有指令的清單，請參閱 IBM MQ for IBM i CL 指令。

註: 可以從「訊息佇列管理程式指令」功能表提交所有 MQM 指令。 若要顯示此功能表，請在指令行上鍵入 GO CMDMQM，然後按 Enter 鍵。

當您從此功能表中選取指令時，系統會自動顯示提示畫面。 若要顯示您直接在指令行上鍵入之指令的提示畫面，請在按下 Enter 鍵之前按 F4。

使用 *CRTMQMQ* 指令建立本端佇列

程序

1. 在指令行上鍵入 CHGMQM，然後按 F4 鍵。
2. 在 **建立 MQM 佇列畫面**上，在 Queue name 欄位中鍵入您要建立的佇列名稱。 若要指定大小寫混合的名稱，請以單引號括住名稱。

3. 在 Queue type 欄位中鍵入 *LCL 。
4. 除非您使用預設佇列管理程式，否則請指定佇列管理程式名稱，然後按 Enter 鍵。 您可以使用新值改寫任何值。 向前捲動以查看進一步的欄位。 用於叢集的選項位於選項清單的結尾。
5. 當您變更任何值時，請按 Enter 鍵以建立佇列。

使用 *WRKMQMQ* 指令建立本端佇列

程序

1. 在指令行上鍵入 WRKMQMQ 。
2. 輸入佇列管理程式的名稱。
3. 如果您要顯示提示畫面，請按 F4。 提示畫面可用來指定同屬佇列名稱或佇列類型，以減少顯示的佇列數目。
4. 按下 Enter ，即會顯示「 **使用 MQM 佇列** 」畫面。 您可以使用新值來改寫任何值。 向前捲動以查看進一步的欄位。 用於叢集的選項位於選項清單的結尾。
5. 按 F6 以建立新佇列; 這會將您帶到 **CRTMQMQ** 畫面。 如需如何建立佇列的指示，請參閱 第 159 頁的『使用 CRTMQMQ 指令建立本端佇列』 。 當您已建立佇列時，會再次顯示 **使用 MQM 佇列** 畫面。 當您按下 F5=Refresh 時，新佇列會新增至清單中。

變更佇列管理程式屬性

關於這項作業

若要變更在 **CHGMQM** 指令上指定之佇列管理程式的屬性，請指定您要變更的屬性及值。 例如，使用下列選項來變更 jupiter.queue.manager 的屬性:

程序

在指令行上鍵入 **CHGMQM** ，然後按 F4 鍵。

結果

此指令會變更使用的無法傳送郵件的佇列，並啟用禁止事件。

使用本端佇列

本節包含可用來管理本端佇列的部分指令範例。 也可以使用 **WRKMQMQ 指令畫面**中的選項來使用所有顯示的指令。

定義本端佇列

對於應用程式，本端佇列管理程式是應用程式所連接的佇列管理程式。 本端佇列管理程式所管理的佇列被認為是該佇列管理程式的本端佇列。

使用指令 **CRTMQMQ QTYPE *LCL** 來建立本端佇列的定義，以及建立稱為佇列的資料結構。 您也可以修改預設本端佇列的佇列性質。

在此範例中，我們定義的佇列 orange.local.queue 指定為具有下列性質:

- 它會啟用取得、停用放置，並以先進先出 (FIFO) 為基礎來運作。
- 它是 一般 佇列，亦即它不是起始佇列或傳輸佇列，且不會產生觸發訊息。
- 佇列深度上限為 1000 則訊息; 訊息長度上限為 2000 個位元組。

下列指令會在預設佇列管理程式上執行此動作:

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL)
TEXT('Queue for messages from other systems')
PUTENBL(*NO)
GETENBL(*YES)
TRGENBL(*NO)
```

```
MSGDLYSEQ(*FIFO)
MAXDEPTH(1000)
MAXMSGLEN(2000)
USAGE(*NORMAL)
```

註:

1. USAGE *NORMAL 指出此佇列不是傳輸佇列。
2. 如果相同佇列管理程式上已有名稱為 orange.local.queue 的本端佇列，則此指令會失敗。 如果您要改寫佇列的現有定義，請使用 REPLACE *YES 屬性，但另請參閱 第 161 頁的『變更本端佇列屬性』。

定義無法傳送郵件的佇列

每一個佇列管理程式都必須有一個本端佇列作為無法傳送郵件的佇列，以便儲存無法遞送至其正確目的地的訊息，以供稍後擷取。 您必須明確告知佇列管理程式無法傳送郵件的佇列。 您可以在 **CRTMQM** 指令上指定無法傳送郵件的佇列來執行此動作，也可以稍後使用 **CHGMQM** 指令來指定一個佇列。 您也必須先定義無法傳送郵件的佇列，才能使用它。

產品隨附稱為 SYSTEM.DEAD.LETTER.QUEUE 的無法傳送郵件佇列範例。 當您建立佇列管理程式時，會自動建立此佇列。 必要的話，您可以修改此定義。 不需要重新命名它，但如果您喜歡，可以重新命名。

無法傳送郵件的佇列沒有特殊需求，除了:

- 它必須是本端佇列。
- 其 MAXMSGL (訊息長度上限) 屬性必須啟用佇列，以容納佇列管理程式 **加上** 無法傳送郵件的標頭 (MQDLH) 大小所必須處理的最大訊息。

IBM MQ 提供無法傳送郵件的佇列處理程式，可讓您指定如何處理或移除在無法傳送郵件的佇列上找到的訊息。 如需進一步資訊，請參閱 IBM MQ for IBM i 無法傳送郵件的佇列處理程式。

顯示預設物件屬性

當您定義 IBM MQ 物件時，它會採用您未從預設物件指定的任何屬性。 例如，當您定義本端佇列時，佇列會從預設本端佇列 (稱為 SYSTEM.DEFAULT.LOCAL.QUEUE) 繼承您在定義中省略的任何屬性。 若要確切查看這些屬性，請使用下列指令:

```
DSPMQMQ QNAME(SYSTEM.DEFAULT.LOCAL.QUEUE) MQMNAME(MYQUEUEMANAGER)
```

複製本端佇列定義

您可以使用 CPYMQMQ 指令來複製佇列定義。 例如:

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

此指令會建立與原始佇列 orange.local.queue 具有相同屬性的佇列，而不是系統預設本端佇列的佇列。

您也可以使用 **CPYMQMQ** 指令來複製佇列定義，但替換原始屬性的一或多項變更。 例如:

```
CPYMQMQ FROMQ('orange.local.queue') TOQ('third.queue') MQMNAME(MYQUEUEMANAGER)
MAXMSGLEN(1024)
```

此指令會將佇列 orange.local.queue 的屬性複製到佇列 third.queue，但指定新佇列上的訊息長度上限為 1024 個位元組，而不是 2000 個位元組。

註: 當您使用 **CPYMQMQ** 指令時，只會複製佇列屬性，不會複製佇列上的訊息。

變更本端佇列屬性

您可以使用 **CHGMQMQ** 指令或具有 REPLACE *YES 屬性的 **CPYMQMQ** 指令，以兩種方式變更佇列屬性。 在 第 160 頁的『定義本端佇列』中，您已定義佇列 orange.local.queue。 例如，如果您需要將此佇列上的訊息長度上限增加到 10,000 個位元組。

- 使用 **CHGMQMQ** 指令:

```
CHGMQMQ QNAME('orange.local.queue') MQMNAME(MYQUEUEMANAGER) MAXMSGLEN(10000)
```

此指令會變更單一屬性，即訊息長度上限的屬性; 所有其他屬性則維持相同。

- 搭配使用 **CRTMQMQ** 指令與 REPLACE *YES 選項，例如:

```
CRTMQMQ QNAME('orange.local.queue') QTYPE(*LCL) MQMNAME(MYQUEUEMANAGER)
MAXMSGLEN(10000) REPLACE(*YES)
```

此指令不僅會變更訊息長度上限，還會變更所有其他屬性，這些屬性會提供其預設值。 現在佇列已啟用放置，而先前已禁止放置。 啟用放置是由佇列 SYSTEM.DEFAULT.LOCAL.QUEUE 指定的預設值，除非您已變更它。

如果您 **減少** 現有佇列上的訊息長度上限，則現有訊息不會受到影響。 不過，任何新訊息都必須符合新準則。

清除本端佇列

若要從本端佇列 magenta.queue 中刪除所有訊息，請使用下列指令:

```
CLRMQMQ QNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

在下列情況下，您無法清除佇列:

- 在同步點下，有未確定的訊息已放置於佇列上。
- 應用程式目前已開啟佇列。

刪除本端佇列

使用指令 **DLTMQMQ** 來刪除本端佇列。

如果佇列上有未確定的訊息，或佇列正在使用中，則無法刪除。

啟用大型佇列

IBM MQ 支援大於 2 GB 的佇列。 如需如何啟用 IBM i 以支援大型檔案的相關資訊，請參閱作業系統文件。

IBM i 產品說明文件可在這裡找到: https://publib.boulder.ibm.com/iseries/

部分公用程式可能無法處理大於 2 GB 的檔案。 在啟用大型檔案支援之前，請先檢查作業系統說明文件，以取得這類支援的限制相關資訊。

使用別名佇列

本節包含可用來管理別名佇列的部分指令範例。 也可以使用 **WRKMQMQ 指令畫面**中的選項來使用所有顯示的指令。

別名佇列 (有時稱為佇列別名) 提供重新導向 MQI 呼叫的方法。 別名佇列不是實際佇列，而是解析為實際佇列的定義。 別名佇列定義包含由 TGTQNAME 屬性指定的目標佇列名稱。

當應用程式在 MQI 呼叫中指定別名佇列時，佇列管理程式會在執行時期解析實際佇列名稱。

例如，已開發應用程式將訊息放置在稱為 my.alias.queue 的佇列上。 當此佇列提出 **MQOPEN** 要求時，它會指定此佇列的名稱; 如果它將訊息放置在此佇列上，則會間接指定此佇列的名稱。 應用程式不知道佇列是別名佇列。 對於每一個使用此別名的 MQI 呼叫，佇列管理程式會解析實際佇列名稱，該名稱可以是本端佇列，也可以是在此佇列管理程式中定義的遠端佇列。

透過變更 TGTQNAME 屬性的值，您可以將 MQI 呼叫重新導向至另一個佇列，可能是在另一個佇列管理程式上。 這對於維護、移轉及負載平衡非常有用。

定義別名佇列

下列指令會建立別名佇列:

```
CRTMQMQ QNAME('my.alias.queue') QTYPE(*ALS) TGTQNAME('yellow.queue')
MQMNAME(MYQUEUEMANAGER)
```

此指令會將指定 my.alias.queue 的 MQI 呼叫重新導向至佇列 yellow.queue。 指令不會建立目標佇列; 如果在執行時期佇列 yellow.queue 不存在， MQI 呼叫會失敗。

如果您變更別名定義，則可以將 MQI 呼叫重新導向至另一個佇列。 例如:

```
CHGMQMQ QNAME('my.alias.queue') TGTQNAME('magenta.queue') MQMNAME(MYQUEUEMANAGER)
```

此指令會將 MQI 呼叫重新導向至另一個佇列 magenta.queue。

您也可以使用別名佇列，讓單一佇列 (目標佇列) 看起來具有不同應用程式的不同屬性。 作法是定義兩個別名，每一個應用程式一個別名。 假設有兩個應用程式:

- 應用程式 ALPHA 可以在 yellow.queue 上放置訊息，但不容許從它取得訊息。
- 應用程式測試版可以從 yellow.queue 取得訊息，但不容許在其中放置訊息。

您可以使用下列指令來執行此動作:

```
/* This alias is put enabled and get disabled for application ALPHA */

CRTMQMQ 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)
```

A 會在其 MQI 呼叫中使用佇列名稱 alphas.alias.queue ;BETA 會使用佇列名稱 betas.alias.queue。 它們都存取相同的佇列，但使用不同的方式。

當您定義別名佇列時，可以使用 REPLACE *YES 屬性，就像您將這些屬性與本端佇列搭配使用一樣。

搭配使用其他指令與別名佇列

您可以使用適當的指令來顯示或變更別名佇列屬性。 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. */

CHQMQMQ QNAME('alphas.alias.queue') TGTQNAME('orange.local.queue') FORCE(*YES)
MQMNAME(MYQUEUEMANAGER)
```

使用模型佇列

本節包含可用來管理模型佇列的部分指令範例。 也可以使用 **WRKMQMQ 指令畫面**中的選項來使用所有顯示的指令。

如果佇列管理程式從應用程式接收 MQI 呼叫，並指定已定義為模型佇列的佇列名稱，則會建立動態佇列。 建立佇列時，佇列管理程式會產生新動態佇列的名稱。 模型佇列是一個範本，指定從它建立的任何動態佇列的屬性。

模型佇列為應用程式提供方便的方法來建立佇列，因為它們是必要項目。

定義模型佇列

您可以使用定義本端佇列的相同方式，來定義具有一組屬性的模型佇列。 模型佇列和本端佇列具有相同的屬性集，但在模型佇列上，您可以指定所建立的動態佇列是暫時還是永久。 (永久佇列是在佇列管理程式重新啟動之間維護，暫時佇列則不是)。 例如：

```
CRTMQMQ QNAME('green.model.queue') QTYPE(*MDL) DFNTYPE(*PERMDYN)
```

此指令會建立模型佇列定義。 從 DFNTYPE 屬性中，從這個範本建立的實際佇列是永久動態佇列。 未指定的屬性會自動從 SYSYTEM.DEFAULT.MODEL.QUEUE 預設佇列複製。

當您定義模型佇列時，可以使用 REPLACE *YES 屬性，其方式與您將它們與本端佇列搭配使用的方式相同。

搭配使用其他指令與模型佇列

您可以使用適當的指令來顯示或變更模型佇列的屬性。 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. */

CHGMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('blue.model.queue') PUTENBL(*YES)
```

使用觸發

使用此資訊來瞭解觸發及程序定義。

IBM MQ 提供在符合佇列上的特定條件時自動啟動應用程式的機能。 條件的一個範例是佇列上的訊息數達到指定的數目時。 此機能稱為 觸發 ，並在 觸發通道中詳細說明。

什麼是觸發?

佇列管理程式會將某些條件定義為構成觸發事件。 如果佇列已啟用觸發，且發生觸發事件，則佇列管理程式會將觸發訊息傳送至稱為起始佇列的佇列。 起始佇列上的觸發訊息存在指出已發生觸發事件。

佇列管理程式所產生的觸發訊息不會持續存在。 這會減少記載 (從而提高效能) ，並在重新啟動期間將重複項減至最少，從而縮短重新啟動時間。

觸發監視器是什麼?

處理起始佇列的程式稱為觸發監視器應用程式，其功能是讀取觸發訊息，並根據觸發訊息中包含的資訊採取適當的動作。 通常此動作會啟動其他應用程式，以處理導致產生觸發訊息的佇列。 從佇列管理程式的觀點來看，觸發監視器應用程式沒有特殊的功能-它是從佇列 (起始佇列) 讀取訊息的另一個應用程式。

變更觸發監視器的工作提交屬性

以指令 **STRMQMTRM** 提供的觸發監視器會使用系統預設工作說明 QDFTJOBD 提交每一個觸發訊息的工作。 這有一些限制，因為提交的工作一律稱為 QDFTJOBD ，且具有預設工作說明的屬性，包括檔案庫清單 *SYSVAL。 IBM MQ 提供置換這些屬性的方法。 例如，可以自訂提交的工作，以具有更有意義的工作名稱，如下所示：

1. 在工作說明中指定您想要的說明，例如記載值。
2. 指定觸發程序中所使用程序定義的「環境資料」：

   ```
   CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOBD)')
   ```

   「觸發監視器」會使用指定的說明來執行 SBMJOB 。

透過在程序定義的「環境資料」中指定適當的關鍵字和值，可以置換 SBMJOB 的其他屬性。 唯一例外是 CMD 關鍵字，因為此屬性由觸發監視器填入。 用於指定程序定義的「環境資料」，其中要變更工作名稱和說明的指令範例如下:

```
CHGMQMPRC PRCNAME(MY_PROCESS) MQMNAME(MHA3) ENVDATA ('JOBD(MYLIB/TRIGJOB)
JOB(TRIGGER)')
```

定義用於觸發的應用程式佇列

應用程式佇列是應用程式透過 MQI 進行傳訊所使用的本端佇列。 觸發需要在應用程式佇列上定義一些佇列屬性。 觸發本身由 TRGENBL 屬性啟用。

在此範例中，當本端佇列 motor.insurance.queue 上有 100 則優先順序為 5 或更高的訊息時，會產生觸發事件，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) TRGTYPE(*DEPTH) TRGDEPTH(100) TRGMSGPTY(5)
```

其中參數為:

MQMNAME(MYQUEUEMANAGER)
佇列管理程式的名稱。

QNAME('motor.insurance.queue')
正在定義的應用程式佇列名稱。

PRCNAME('motor.insurance.quote.process')
要由觸發監視器程式啟動的應用程式名稱。

MAXMSGLEN(2000)
佇列上訊息的長度上限。

DFTMSGPST(*YES)
依預設，此佇列上的訊息會持續保存。

INITQNAME('motor.ins.init.queue')
佇列管理程式要放置觸發訊息的起始佇列名稱。

TRGENBL(*YES)
觸發屬性值。

TRGTYPE(*DEPTH)
當必要優先順序 (**TRGMSGPTY**) 的訊息數時，會產生觸發事件 達到 **TRGDEPTH** 中指定的數目。

TRGDEPTH(100)
產生觸發事件所需的訊息數目。

TRGMSGPTY(5)
在決定是否產生觸發事件時，由佇列管理程式計算的訊息優先順序。 只會計算優先順序為 5 或以上的訊息。

定義起始佇列

當觸發事件發生時，佇列管理程式會將觸發訊息放置在應用程式佇列定義中指定的起始佇列上。 起始佇列沒有特殊設定，但您可以使用本端佇列 motor.ins.init.queue 的下列定義作為指引:

```
CRTMQMQ MQMNAME(MYQUEUEMANAGER) QNAME('motor.ins.init.queue') QTYPE(*LCL)
GETENBL(*YES) SHARE(*NO) TRGTYPE(*NONE)
MAXMSGL(2000)
MAXDEPTH(1000)
```

建立程序定義

使用 **CRTMQMPRC** 指令來建立程序定義。 程序定義會將應用程式佇列與要處理來自佇列之訊息的應用程式相關聯。 這是透過應用程式佇列 motor.insurance.queue 上的 PRCNAME 屬性來完成。 下列指令會建立此範例中所識別的必要處理程序 motor.insurance.quote.process:

```
CRTMQMPRC MQMNAME(MYQUEUEMANAGER) PRCNAME('motor.insurance.quote.process')
TEXT('Insurance request message processing')
APPTYPE(*OS400) APPID(MQTEST/TESTPROG)
USRDATA('open, close, 235')
```

其中參數為:

MQMNAME(MYQUEUEMANAGER)
佇列管理程式的名稱。

PRCNAME('motor.insurance.quote.process')
程序定義的名稱。

TEXT('Insurance request message processing')
與此定義相關的應用程式說明。 當您使用 **DSPMQMPRC** 指令時，會顯示此文字。 這可協助您識別處理程序執行的動作。 如果您在字串中使用空格，則必須以單引號括住字串。

APPTYPE(*OS400)
要啟動的應用程式類型。

APPID(MQTEST/TESTPROG)
應用程式執行檔的名稱，指定為完整檔名。

USRDATA('open, close, 235')
使用者定義資料，可供應用程式使用。

顯示程序定義

使用 **DSPMQMPRC** 指令來檢查定義的結果。 例如:

```
MQMNAME(MYQUEUEMANAGER) DSPMQMPRC('motor.insurance.quote.process')
```

您也可以使用 **CHGMQMPRC** 指令來變更現有的程序定義，並使用 **DLTMQMPRC** 指令來刪除程序定義。

兩個系統之間的通訊

下列範例說明如何使用 CL 指令來設定兩個 IBM MQ for IBM i 系統，以便它們可以彼此通訊。

系統稱為 SYSTEMA 和 SYSTEMB，所用的通訊協定是 TCP/IP。

執行下列程序:

1. 在 SYSTEMA 上建立佇列管理程式，並將它稱為 QMGRA1。

   ```
   CRTMQM   MQMNAME(QMGRA1) TEXT('System A - Queue +
   Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   ```

2. 啟動此佇列管理程式。

   ```
   STRMQM   MQMNAME(QMGRA1)
   ```

3. 在 SYSTEMA 上定義您需要將訊息傳送至 SYSTEMB 上佇列管理程式的 IBM MQ 物件。

   ```
   /* 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) +
RMTMQMNAME(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') +
CONNAME(SYSTEMB) TMQNAME(XMITQ.TO.QMGRB1)
```

4. 在 SYSTEMB 上建立佇列管理程式，並將它稱為 QMGRB1。

```
CRTMQM   MQMNAME(QMGRB1) TEXT('System B - Queue +
Manager 1') UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
```

5. 在 SYSTEMB 上啟動佇列管理程式。

```
STRMQM   MQMNAME(QMGRB1)
```

6. 在 SYSTEMA 上定義從佇列管理程式接收訊息所需的 IBM MQ 物件。

```
/* 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. 最後，在 SYSTEMB 上啟動 TCP/IP 接聽器，以便可以啟動通道。 此範例使用預設埠 1414。

```
STRMQMLSR MQMNAME(QMGRB1)
```

現在，您已準備好在 SYSTEMA 與 SYSTEMB 之間傳送測試訊息。 使用其中一個提供的範例，將一系列訊息放置到 SYSTEMA 上的遠端佇列。

在 SYSTEMA 上啟動通道，方法是使用指令 STRMQMCHL ，或使用指令 WRKMQMCHL ，並針對傳送端通道輸入啟動要求 (選項 14)。

通道應該進入 RUNNING 狀態，且訊息會傳送至 SYSTEMB 上的 TARGETB 佇列。

發出下列指令來檢查您的訊息:

```
WRKMQMMSG QNAME(TARGETB) MQMNAME(QMGRB1).
```

資源定義範例

此範例包含 AMQSAMP4 範例 IBM i CL 程式。

```
/*********************************************************************/
/*                                                                   */
/* 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 */+
TGTQNAME('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('Transmision 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('QMQM/AMQSINQ4')  /* C       +
/*    APPID('QMQM/AMQ0INQ4')  /* COBOL */    +
/*    APPID('QMQM/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('QMQM/AMQSSET4')  /* C   */    +
/*    APPID('QMQM/AMQ0SET4')  /* COBOL */    +
/*    APPID('QMQM/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('QMQM/AMQSECH4')  /* C   */    +
/*    APPID('QMQM/AMQ0ECH4')  /* COBOL */    +
/*    APPID('QMQM/AMQ3ECH4')  /* RPG - ILE */

/*********************************************************************/
/*                                                                   */
/*  Normal return.                                                   */
/*                                                                   */
/*********************************************************************/
SNDPGMMSG  MSG('AMQSAMP4 Completed creating sample +
objects for ' *CAT &QMGRNAME)
RETURN
ENDPGM

/*********************************************************************/
/*                                                                   */
/* END OF AMQSAMP4                                                   */
/*                                                                   */
/*********************************************************************/
```

管理 IBM MQ for IBM i 的替代方式

使用此資訊來瞭解 IBM MQ for IBM i、MQSC 指令、PCF 指令及遠端管理。

您可以使用 IBM MQ 檢測事件來監視佇列管理程式的作業。 如需 IBM MQ 設備測試事件以及如何使用它們的相關資訊，請參閱 設備測試事件 。

您通常使用 IBM i CL 指令來管理 IBM MQ for IBM i。 如需這些指令的概觀，請參閱 第 158 頁的『使用 CL 指令管理 IBM MQ for IBM i』 。

使用 CL 指令是管理系統的偏好方法。 不過，您可以使用各種其他方法。 本節提供這些方法的概觀，並包括下列主題:

本端和遠端管理

您可以在本端或遠端管理 IBM MQ for IBM i 物件。

本端管理 表示對您在本端系統上定義的任何佇列管理程式執行管理作業。 在 IBM MQ 中，您可以將此視為本端管理，因為不涉及任何 IBM MQ 通道，即通訊由作業系統管理。 若要執行這種類型的作業，您必須登入遠端系統並從該處發出指令，或建立可以為您發出指令的處理程序。

IBM MQ 支援透過所謂 遠端管理從單一點進行管理。 遠端管理包括將可程式化指令格式 (PCF) 控制訊息傳送至目標佇列管理程式上的 SYSTEM.ADMIN.COMMAND.QUEUE 。

有多種產生 PCF 訊息的方法。 它們是:

1. 使用 PCF 訊息寫入程式。 請參閱 第 172 頁的『使用 PCF 指令進行管理』 。
2. 使用 MQAI 撰寫程式，以送出 PCF 訊息。 請參閱 第 19 頁的『使用 MQAI 來簡化 PCF 的使用』 。
3. 使用 IBM MQ for Windows 隨附的「IBM MQ 檔案總管」，可讓您使用圖形使用者介面 (GUI) 並產生正確的 PCF 訊息。 請參閱 第 172 頁的『將 IBM MQ Explorer 與 IBM MQ for IBM i 搭配使用』 。
4. 使用 **STRMQMMQSC** 可間接將指令傳送至遠端佇列管理程式。 請參閱 第 171 頁的『使用 MQSC 指令進行管理』 。

例如，您可以發出遠端指令來變更遠端佇列管理程式上的佇列定義。

有些指令無法以這種方式發出，尤其是建立或啟動佇列管理程式，以及啟動指令伺服器。 若要執行這種類型的作業，您必須登入遠端系統並從該處發出指令，或建立可以為您發出指令的處理程序。

使用 MQSC 指令進行管理

使用此資訊來瞭解 MQSC 指令，以及如何使用它們來管理 IBM MQ for IBM i。

IBM MQ Script (MQSC) 指令以人類可讀的格式 (即 EBCDIC 文字) 撰寫。 您可以使用 MQSC 指令來管理佇列管理程式物件，包括佇列管理程式本身、佇列、程序定義、名稱清單、通道、用戶端連線通道、接聽器、服務、主題及鑑別資訊物件。

您可以使用 **STRMQMMQSC** IBM MQ CL 指令，對佇列管理程式發出 MQSC 指令。 此方法僅是批次方法，從伺服器檔案庫系統中的來源實體檔取得其輸入。 此來源實體檔的預設名稱為 QMQSC。

小心: 請勿使用 QTEMP 程式庫作為 STRMQMQSC 的來源程式庫，因為 QTEMP 程式庫的使用受到限制。 您必須使用另一個檔案庫作為指令的輸入檔。

IBM MQ for IBM i 未提供稱為 QMQSC 的原始檔。 若要處理 MQSC 指令，您必須發出下列指令，在您選擇的程式庫中建立 QMQSC 原始檔:

```
CRTSRCPF FILE(MYLIB/QMQSC) RCDLEN(240) TEXT('IBM MQ - MQSC Source')
```

MQSC 來源保留在此原始檔內的成員中。 若要使用成員，請輸入下列指令:

```
WRKMBRPDM MYLIB/QMQSC
```

現在您可以新增成員並維護現有成員

您也可以透過發出 RUNMQSC 或下列指令，以互動方式輸入 MQSC 指令:

1. 鍵入佇列管理程式名稱，並按 Enter 鍵以存取 **WRKMQM** 結果畫面。
2. 在此畫面上選取 F23=More options 。
3. 在「第 171 頁的圖 31」顯示的畫面上，針對作用中佇列管理程式選取選項 26。

若要結束這類 MQSC 階段作業，請鍵入 end 。

第 171 頁的圖 31 是 MQSC 指令檔的摘錄，顯示 MQSC 指令 (DEFINE QLOCAL) 及其屬性。

```
.
.
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;
.
.
```

圖 *31:* 從 *MQSC* 指令檔 *myprog.in* 解壓縮

為了在 IBM MQ 環境之間實現可攜性，請將 MQSC 指令檔中的行長度限制為 72 個字元。 加號表示指令在下一行繼續執行。

在 MQSC 中指定的物件屬性會以大寫顯示在此區段中 (例如， RQMNAME) ，雖然它們不區分大小寫。

註:

1. MQSC 檔案的格式不取決於其在檔案系統中的位置。

2. MQSC 屬性名稱限制為 8 個字元。
3. MQSC 指令可在其他平台上使用，包括 z/OS。

如需每一個 MQSC 指令及其語法的說明，請參閱 第 67 頁的『Script (MQSC) 指令』。

使用 PCF 指令進行管理

IBM MQ 可程式指令格式 (PCF) 指令的目的是容許將管理作業程式設計到管理程式中。 如此一來，您可以從程式建立佇列和程序定義，以及變更佇列管理程式。

PCF 指令涵蓋 MQSC 指令所提供的相同函數範圍。 不過，與 MQSC 指令不同， PCF 指令及其回覆不是您可以讀取的文字格式。

您可以撰寫程式，從單一節點向網路中的任何佇列管理程式發出 PCF 指令。 透過此方式，您可以將管理作業集中化及自動化。

每一個 PCF 指令都是內嵌在 IBM MQ 訊息的應用程式資料部分中的資料結構。 每一個指令都會以與任何其他訊息相同的方式，使用 MQI 函數 MQPUT 來傳送至目標佇列管理程式。 佇列管理程式上接收訊息的指令伺服器會將它解譯為指令訊息，並執行指令。 為了取得回覆，應用程式會發出 MQGET 呼叫，並以另一個資料結構傳回回覆資料。 然後，應用程式可以處理回覆並相應地採取行動。

簡言之，這些是應用程式設計師必須指定以建立 PCF 指令訊息的部分內容:

訊息描述子
這是標準 IBM MQ 訊息描述子，其中:

- 訊息類型 (*MsgType*) 為 MQMT_REQUEST。
- 訊息格式 (*Format*) 是 MQFMT_ADMIN。

應用程式資料
包含 PCF 訊息 (包括 PCF 標頭) ，其中:

- PCF 訊息類型 (*Type*) 指定 MQCFT_COMMAND。
- 指令 ID 指定指令，例如 *Change Queue* (MQCMD_CHANGE_Q)。

Escape PCF 是在訊息文字內包含 MQSC 指令的 PCF 指令。 您可以使用 PCF 將指令傳送至遠端佇列管理程式。 如需進一步資訊，請參閱第 19 頁的『使用 MQAI 來簡化 PCF 的使用』。

如需 PCF 資料結構及其實作方式的完整說明，請參閱 指令及回應的結構。

將 IBM MQ Explorer 與 IBM MQ for IBM i 搭配使用

使用此資訊，以使用 IBM MQ Explorer 來管理 IBM MQ for IBM i 。

IBM MQ for Windows (x86 平台) 及 IBM MQ for Linux (x86 及 x86-64 平台) 提供稱為「 IBM MQ 探險家」的管理介面，以執行管理作業作為使用 CL、控制或 MQSC 指令的替代方案。

「 IBM MQ 探險家」可讓您從執行 Windows (x86 平台) 或 Linux (x86 及 x86-64 平台) 的電腦執行網路的本端或遠端管理，方法是將「 IBM MQ 探險家」指向您感興趣的佇列管理程式及叢集。 遠端佇列管理程式中說明可使用「 IBM MQ 探險家」來管理的 IBM MQ 平台及層次。

使用「 IBM MQ 檔案總管」，您可以執行下列動作:

- 啟動和停止佇列管理程式 (僅在本端機器上)。
- 定義、顯示及變更 IBM MQ 物件 (例如佇列、主題及通道) 的定義。
- 瀏覽佇列上的訊息。
- 啟動和停止通道。
- 檢視通道的狀態資訊。
- 檢視叢集中的佇列管理程式。
- 請檢查以查看哪些應用程式、使用者或通道已開啟特定佇列。
- 使用「 **建立新的叢集** 」精靈來建立新的佇列管理程式叢集。
- 使用「 **將佇列管理程式新增至叢集** 」精靈，將佇列管理程式新增至叢集。

- 管理搭配 Secure Sockets Layer (SSL) 通道安全使用的鑑別資訊物件。

使用線上指引，您可以:

- 定義及控制各種資源，包括佇列管理程式、佇列、通道、程序定義、用戶端連線通道、接聽器、主題、服務、名稱清單及叢集。
- 啟動或停止佇列管理程式及其相關聯的處理程序。
- 檢視工作站上或其他工作站中的佇列管理程式及其相關聯物件。
- 檢查佇列管理程式、叢集及通道的狀態。

在嘗試使用 IBM MQ Explorer 來管理伺服器機器上的 IBM MQ 之前，請確定您已滿足下列需求。 請檢查:

1. 正在針對受管理的 **任何** 佇列管理程式執行指令伺服器，由 CL 指令 **STRMQMCSVR** 在伺服器上啟動。
2. 每個遠端佇列管理程式都有適當的 TCP/IP 接聽器。 這是由 **STRMQMLSR** 指令啟動的 IBM MQ 接聽器。
3. 伺服器連線通道 (稱為 SYSTEM.ADMIN.SVRCONN) 存在於每一個遠端佇列管理程式上。 您 **必須** 自行建立此通道。 對於每一個受管理的遠端佇列管理程式，這是必要的。 沒有它，無法進行遠端管理。
4. 驗證 SYSTEM.MQEXPLORER.REPLY.MODEL 佇列是否存在。

管理用於遠端管理的指令伺服器

使用此資訊來瞭解 IBM MQ IBM i 指令伺服器的遠端管理。

每一個佇列管理程式都可以有相關聯的指令伺服器。 指令伺服器會處理來自遠端佇列管理程式的任何送入指令，或來自應用程式的 PCF 指令。 它會將指令呈現給佇列管理程式進行處理，並根據指令的原點傳回完成碼或操作員訊息。

對於涉及 PCF 的所有管理、MQAI 以及遠端管理而言，指令伺服器是必要的。

註: 對於遠端管理，您必須確定目標佇列管理程式正在執行中。 否則，包含指令的訊息無法離開從中發出指令的佇列管理程式。 相反地，這些訊息會在提供遠端佇列管理程式的本端傳輸佇列中排入佇列。 如果可能的話，請避免此狀況。

有個別控制指令可用來啟動及停止指令伺服器。 您可以使用「IBM MQ 檔案總管」來執行下列各節中說明的作業。

啟動和停止指令伺服器

若要啟動指令伺服器，請使用下列 CL 指令:

```
STRMQMCSVR MQMNAME('saturn.queue.manager')
```

其中 saturn.queue.manager 是正在啟動指令伺服器的佇列管理程式。

若要停止指令伺服器，請使用下列其中一個 CL 指令:

1. ```
   ENDMQMCSVR MQMNAME('saturn.queue.manager') OPTION(*CNTRLD)
   ```

   以執行受管制的停止，其中 saturn.queue.manager 是正在停止指令伺服器的佇列管理程式。 這是預設選項，表示可以省略 OPTION(*CNTRLD)。
2. ```
   ENDMQMCSVR MQMNAME('saturn.queue.manager') OPTION(*IMMED)
   ```

   執行立即停止，其中 saturn.queue.manager 是要停止指令伺服器的佇列管理程式。

顯示指令伺服器的狀態

對於遠端管理，請確定目標佇列管理程式上的指令伺服器正在執行中。 如果它不在執行中，則無法處理遠端指令。 任何包含指令的訊息都會排入目標佇列管理程式的指令佇列 SYSTEM.ADMIN.COMMAND.QUEUE 中。

若要顯示佇列管理程式的指令伺服器狀態 (在這裡稱為 saturn.queue.manager)， CL 指令為:

```
DSPMQMCSVR MQMNAME('saturn.queue.manager')
```

在目標機器上發出此指令。 如果指令伺服器正在執行中，則會出現 第 174 頁的圖 32 中顯示的畫面:

```
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
```

圖 *32:* 顯示 *MQM* 指令伺服器畫面

工作管理

此資訊說明 IBM MQ 處理工作要求的方式，並詳細說明可用於設定優先順序及控制與 IBM MQ 相關聯之工作的選項。

警告

除非您完全瞭解 IBM i 和 IBM MQ 工作管理的概念，否則請 **不要** 變更 IBM MQ 工作管理物件。

如需子系統及工作說明的相關資訊，請參閱 IBM i 產品說明文件中的 工作管理 。 請特別注意 啟動工作 及 批次工作上的小節。

IBM MQ for IBM i 納入 IBM i UNIX 環境及 IBM i 執行緒。 請 **不要** 對 Integrated File System (IFS) 中的物件進行任何變更。

在正常作業期間， IBM MQ 佇列管理程式會啟動一些批次工作來執行不同的作業。 依預設，這些批次工作在安裝 IBM MQ 時所建立的 QMQM 子系統中執行。

工作管理是指自訂 IBM MQ 作業以從系統取得最佳效能，或簡化管理的程序。

例如，您可以:

- 變更工作的執行優先順序，使一個佇列管理程式比另一個佇列管理程式更能回應。
- 將一些工作的輸出重新導向至特定輸出佇列。
- 讓特定類型的所有工作在特定子系統中執行。
- 將錯誤隔離至子系統。

工作管理是透過建立或變更與 IBM MQ 工作相關聯的工作說明來執行。 您可以配置下列項目的工作管理:

- 整個 IBM MQ 安裝。
- 個別佇列管理程式。
- 個別佇列管理程式的個別工作。

IBM MQ 作業的說明

這是 IBM MQ 工作的表格，以及各工作的簡要說明。

當佇列管理程式正在執行時，您會看到下列部分或所有批次工作在 IBM MQ 子系統中的 QMQM 使用者設定檔下執行。 這些工作在 第 175 頁的表 11 中有簡要說明。

您可以使用「**使用佇列管理程式** (WRKMQM)」畫面上的選項 22 ，檢視連接至佇列管理程式的所有工作。 您可以使用 WRKMQMLSR 指令來檢視接聽器。

表 *11: IBM MQ* 作業。

工作名稱	**函數**
AMQALMPX	定期取得日誌登載檢查點的檢查點處理器。
AMQZMUC0	公用程式管理程式。 此工作執行重要佇列管理程式公用程式，例如日誌鏈管理程式。
AMQZXMA0	執行控制器，它是佇列管理程式所啟動的第一個工作。 它會處理 MQCONN 要求，並啟動代理程式處理程序來處理 IBM MQ API 呼叫。
AMQZFUMA	物件權限管理程式 (OAM)。
AMQZLAA0	針對使用 MQCNO_STANDARD_BINDING 連接至佇列管理程式的應用程式執行大部分工作的佇列管理程式代理程式。
AMQZLSA0	佇列管理程式代理程式。
AMQZMUF0	公用程式管理程式
AMQZMGR0	處理程序控制器。 此工作用於啟動及管理接聽器和服務。
AMQZMUR0	公用程式管理程式。 此工作執行重要佇列管理程式公用程式，例如日誌鏈管理程式。
AMQFQPUB	已將發佈/訂閱常駐程式排入佇列。
AMQFCXBA	分配管理系統工作者工作。
RUNMQBRK	分配管理系統控制工作。
AMQRMPPA	通道處理程序儲存區工作。
AMQCRSTA	TCP/IP 呼叫的通道回應器。
AMQCRS6B	LU62 接收端通道及用戶端連線 (請參閱附註)。
AMQRRMFA	叢集的儲存庫管理程式。
AMQCLMAA	非執行緒 TCP/IP 接聽器。
AMQPCSEA	處理 PCF 及遠端管理要求的 PCF 指令處理器。
RUNMQTRM	觸發監視器。
RUNMQDLQ	無法傳送郵件的佇列處理程式。
RUNMQCHI	通道起始程式。
RUNMQCHL	針對每一個傳送端通道啟動的傳送端通道工作。
RUNMQLSR	執行緒 TCP/IP 接聽器。
AMQRCMLA	通道 MQSC 及 PCF 指令處理器。

註: LU62 接收端工作在通訊子系統中執行，並從用來啟動工作的遞送及通訊項目中取得其執行時期內容。 如需相關資訊，請參閱 起始結束 (接收端) 。

IBM MQ for IBM i 工作管理物件

安裝 IBM MQ 時，會在 QMQM 程式庫中提供各種物件，以協助進行工作管理。 這些物件是 IBM MQ 工作在其自己的子系統中執行所需的物件。

提供兩個 IBM MQ 批次工作的範例工作說明。 如果未提供 IBM MQ 工作的特定工作說明，則它會以預設工作說明 QMQMJOBD 執行。

當您安裝 IBM MQ 時所提供的工作管理物件會列在 第 176 頁的表 12 中，而針對佇列管理程式所建立的物件會列在 第 176 頁的表 13 中。

註: 工作管理物件可以在 QMQM 程式庫中找到，佇列管理程式物件可以在佇列管理程式程式庫中找到。

表 *12:* 工作管理物件

姓名	**類型**	**說明**
AMQALMPX	*JOBD	檢查點處理程序所使用的工作說明
AMQZLAA0	*JOBD	IBM MQ 代理程式處理程序所使用的工作說明
AMQZLSA0	*JOBD	隔離的連結佇列管理程式代理程式
AMQZXMA0	*JOBD	IBM MQ 執行控制器使用的工作說明
QMQM	*SBSD	在其中執行所有 IBM MQ 工作的子系統
QMQM	*JOBQ	連接至所提供子系統的工作佇列
QMQMJOBD	*JOBD	預設 IBM MQ 工作說明，在沒有工作的特定工作說明時使用
QMQMMSG	*MSGQ	IBM MQ 工作的預設訊息佇列。
QMQMRUN20	*CLS	高優先順序 IBM MQ 工作的類別說明
QMQMRUN35	*CLS	中優先順序 IBM MQ 工作的類別說明
QMQMRUN50	*CLS	低優先順序 IBM MQ 工作的類別說明

表 *13:* 為佇列管理程式建立的工作管理物件

姓名	**類型**	**說明**
AMQA000000	*JRNRCV	本端異動日誌接收器
AMQAJRN	*JRN	本端異動日誌
AMQJRNINF	*USRSPC	以佇列管理程式啟動及媒體回復所需的最新異動日誌接收器更新的使用者空間。 應用程式可以查詢此使用者空間，以判斷哪些異動日誌接收器需要保存，哪些可以安全刪除。
AMQAJRNMSG	*MSGQ	本端日誌登載訊息佇列
AMQCRC6B	*PGM	啟動 LU6.2 連線的程式
AMQRFOLD	*FILE	已移轉佇列管理程式通道定義檔
QMQMMSG	*MSGQ	佇列管理程式訊息佇列

IBM MQ 如何使用工作管理物件

此資訊說明 IBM MQ 使用工作管理物件的方式，並提供配置範例。

小心: 請勿變更 QMQM 子系統中的工作佇列登錄設定，以依優先順序限制子系統中容許的工作數目。 如果您嘗試這樣做，則可以在提交必要的 IBM MQ 工作之後停止它們執行，並導致佇列管理程式啟動失敗。

若要瞭解如何配置工作管理，您必須先瞭解 IBM MQ 如何使用工作說明。

用來啟動工作的工作說明控制工作的許多屬性。 例如:

- 將工作排入佇列的工作佇列，以及工作執行所在的子系統。
- 用來啟動工作的遞送資料，以及工作用於其執行時期參數的類別。

- 工作用於列印檔案的輸出佇列。

啟動 IBM MQ 工作的程序可以分為三個步驟:

1. IBM MQ 選取工作說明。

   IBM MQ 使用下列技術來判定要用於批次工作的工作說明:

   a. 請在佇列管理程式檔案庫中尋找與工作同名的工作說明。 如需佇列管理程式庫的進一步詳細資料，請參閱 瞭解 IBM MQ for IBM i 佇列管理程式庫名稱 。

   b. 在佇列管理程式檔案庫中尋找預設工作說明 QMQMJOBD。

   c. 請在 QMQM 程式庫中尋找與工作同名的工作說明。

   d. 使用 QMQM 程式庫中的預設工作說明 QMQMJOBD。

2. 工作已提交至工作佇列。

   依預設， IBM MQ 隨附的工作說明已設定為將工作放入檔案庫 QMQM 中的工作佇列 QMQM。 QMQM 工作佇列會連接至所提供的 QMQM 子系統，因此依預設，工作會在 QMQM 子系統中開始執行。

3. 工作進入子系統並完成遞送步驟。

   當工作進入子系統時，會使用工作說明上指定的遞送資料來尋找工作的遞送登錄。

   遞送資料必須符合 QMQM 子系統中定義的其中一個遞送項目，且這會定義工作使用哪些提供的類別 (QMQMRUN20、 QMQMRUN35 或 QMQMRUN50)。

註: 如果 IBM MQ 工作似乎未啟動，請確定子系統正在執行中，且未保留工作佇列。

如果您已修改 IBM MQ 工作管理物件，請確定所有項目都有正確關聯。 例如，如果您在工作說明上指定 QMQM/QMQM 以外的工作佇列，請確定已針對子系統 (即 QMQM) 執行 ADDJOBQE。

您可以使用下列工作表作為範例，為 第 175 頁的表 11 中所記載的每一個工作建立工作說明:

```
What is the queue manager library name? ___________
Does job description AMQZXMA0 exist in the queue manager library? Yes   No
Does job description QMQMJOBD exist in the queue manager library? Yes   No
Does job description AMQZXMA0 exist in the QMQM library?     Yes   No
Does job description QMQMJOBD exist in the QMQM library?     Yes   No
```

如果您對所有這些問題都回答「否」，請在 QMQM 程式庫中建立廣域工作說明 QMQMJOBD。

IBM MQ 訊息佇列

在每一個佇列管理程式檔案庫中建立 IBM MQ 訊息佇列 QMQMMSG。 當佇列管理程式工作結束，且 IBM MQ 將訊息傳送至佇列時，作業系統訊息會傳送至這個佇列。 例如，報告啟動時需要哪些異動日誌接收器。 請將此訊息佇列中的訊息數保持在可管理的大小，以便更容易監視。

預設系統範例

這些範例顯示在佇列管理程式啟動時提交部分標準工作時，未修改的 IBM MQ 安裝如何運作。

首先，會啟動 AMQZXMA0 執行控制器工作。

1. 針對佇列管理程式 TESTQM 發出 **STRMQM** 指令。
2. IBM MQ 會先搜尋佇列管理程式庫 QMTESTQM ，找出工作說明 AMQZXMA0，然後再搜尋工作說明 QMQMJOBD。

   這兩個工作說明都不存在，因此 IBM MQ 會在產品程式庫 QMQM 中尋找工作說明 AMQZXMA0 。 此工作說明已存在，因此會使用它來提交工作。

3. 工作說明使用 IBM MQ 預設工作佇列，因此會將工作提交至工作佇列 QMQM/QMQM。
4. AMQZXMA0 工作說明上的遞送資料是 QMQMRUN20，因此系統會在子系統遞送項目中搜尋符合該資料的項目。

   依預設，序號為 9900 的遞送登錄具有符合 QMQMRUN20 的比較資料，因此會以該遞送登錄上定義的類別 (也稱為 QMQMRUN20) 來啟動工作。

5. QMQM/QMQMRUN20 類別已將執行優先順序設為 20，因此 AMQZXMA0 工作以與系統上大部分互動式工作相同的優先順序在子系統 QMQM 中執行。

接下來， AMQALMPX 檢查點處理程序工作即會啟動。

1. IBM MQ 會先在佇列管理程式庫 QMTESTQM 中搜尋工作說明 AMQALPMX，然後搜尋工作說明 QMQMJOBD。

   這些工作說明都不存在，因此 IBM MQ 會在產品程式庫 QMQM 中尋找工作說明 AMQALMPX 及 QMQMJOBD。

   工作說明 AMQALMPX 不存在，但 QMQMJOBD 存在，因此會使用 QMQMJOBD 來提交工作。

   註: QMQMJOBD 工作說明一律用於沒有專屬工作說明的 IBM MQ 工作。
2. 工作說明使用 IBM MQ 預設工作佇列，因此會將工作提交至工作佇列 QMQM/QMQM。
3. QMQMJOBD 工作說明上的遞送資料是 QMQMRUN35，因此系統會在子系統遞送項目中搜尋符合該資料的項目。

   依預設，序號為 9910 的遞送登錄具有符合 QMQMRUN35 的比較資料，因此會以該遞送登錄上定義的類別 (也稱為 QMQMRUN35) 來啟動工作。
4. QMQM/QMQMRUN35 類別已將執行優先順序設為 35，因此 AMQALMPX 工作在子系統 QMQM 中執行，其優先順序低於系統上大部分互動式工作，但高於大部分批次工作。

配置工作管理範例

使用此資訊來瞭解如何變更及建立 IBM MQ 工作說明，以變更 IBM MQ 工作的執行時期屬性。

IBM MQ 工作管理彈性的關鍵在於 IBM MQ 搜尋工作說明的兩層方式:

- 如果您在佇列管理程式庫中建立或變更工作說明，則那些變更會置換 QMQM 中的廣域工作說明，但這些變更是 本端，且只會影響該特定佇列管理程式。
- 如果您在 QMQM 程式庫中建立或變更廣域工作說明，除非針對個別佇列管理程式在本端置換，否則這些工作說明會影響系統上的所有佇列管理程式。

1. 下列範例會增加個別佇列管理程式之通道控制工作的優先順序。

   若要讓儲存庫管理程式及通道起始程式工作 AMQRRMFA 及 RUNMQCHI 儘快針對佇列管理程式 TESTQM 執行，請執行下列步驟:

   a. 使用您要在佇列管理程式檔案庫中控制的 IBM MQ 處理程序名稱，建立 QMQM/QMQMJOBD 工作說明的本端重複項。 例如:

   ```
   CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
   NEWOBJ(RUNMQCHI)
   CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
   NEWOBJ(AMQRRMFA)
   ```

   b. 變更工作說明上的遞送資料參數，以確保工作使用 QMQMRUN20 類別。

   ```
   CHGJOBD JOBD(QMTESTQM/RUNMQCHI) RTGDTA('QMQMRUN20')
   CHGJOBD JOBD(QMTESTQM/AMQRRMFA) RTGDTA('QMQMRUN20')
   ```

   現在佇列管理程式 TESTQM 的 AMQRRMFA 及 RUNMQCHI 工作:

   - 使用佇列管理程式檔案庫中新的本端工作說明
   - 以優先順序 20 執行，因為當工作進入子系統時，會使用 QMQMRUN20 類別。
2. 下列範例定義 QMQM 子系統的新執行優先順序類別。

   a. 透過發出下列指令，在 QMQM 程式庫中建立重複的類別，以容許其他佇列管理程式存取該類別:

   ```
   CRTDUPOBJ OBJ(QMQMRUN20) FROMLIB(QMQM) OBJTYPE(*CLS) TOLIB(QMQM)
   NEWOBJ(QMQMRUN10)
   ```

b. 發出下列指令，將類別變更為具有新的執行優先順序:

```
CHGCLS CLS(QMQM/QMQMRUN10) RUNPTY(10)
```

c. 發出下列指令，將新的類別定義新增至子系統:

```
ADDRTGE SBSD(QMQM/QMQM) SEQNBR(8999) CMPVAL('QMQMRUN10') PGM(QSYS/QCMD)
CLS(QMQM/QMQMRUN10)
```

註: 您可以為遞送序號指定任何數值，但這些值必須依序排列。 此序號告訴子系統要在遞送登錄中搜尋遞送資料相符項的順序。

d. 發出下列指令，將本端或廣域工作說明變更為使用新的優先順序類別:

```
CHGJOBD JOBD(QMQMlibname/QMQMJOBD) RTGDTA('QMQMRUN10')
```

現在與 QMlibraryname 相關聯的所有佇列管理程式工作都使用執行優先順序 10。

3. 下列範例會在其自己的子系統中執行佇列管理程式

若要讓佇列管理程式 TESTQM 的所有工作在 QBATCH 子系統中執行，請執行下列步驟:

a. 使用指令在佇列管理程式檔案庫中建立 QMQM/QMQMJOBD 工作說明的本端副本

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMTESTQM)
```

b. 變更工作說明上的工作佇列參數，以確定工作使用 QBATCH 工作佇列。

```
CHGJOBD JOBD(QMTESTQM/QMQMJOBD) JOBQ(*LIBL/QBATCH)
```

註: 工作佇列與子系統說明相關。 如果您發現工作保留在工作佇列上，請驗證工作佇列定義已定義在 SBSD 上。 對子系統使用 DSPSBSD 指令，並採用選項 **6**「工作佇列登錄」。

現在佇列管理程式 TESTQM 的所有工作:

- 在佇列管理程式檔案庫中使用新的本端預設工作說明
- 提交至工作佇列 QBATCH。

若要確保正確遞送工作並設定其優先順序，請執行下列動作:

- 在子系統 QBATCH 中建立 IBM MQ 工作的遞送登錄，或
- 不論使用何種遞送資料，都依賴呼叫 QCMD 的全面遞送登錄。

僅當工作佇列 QBATCH 的最大作用中工作選項設定為 *NOMAX 時，此選項才有效。 系統預設值為 1。

4. 下列範例會建立另一個 IBM MQ 子系統

a. 發出下列指令，在 QMQM 程式庫中建立重複的子系統:

```
CRTDUPOBJ OBJ(QMQM) FROMLIB(QMQM) OBJTYPE(*SBSD) TOLIB(QMQM) NEWOBJ(QMQM2)
```

b. 發出下列指令來移除 QMQM 工作佇列:

```
RMVJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM)
```

c. 發出下列指令，為子系統建立新的工作佇列:

```
CRTJOBQ JOBQ(QMQM/QMQM2) TEXT('Job queue for IBM MQ Queue Manager')
```

d. 發出下列指令，將工作佇列登錄新增至子系統:

```
ADDJOBQE SBSD(QMQM/QMQM2) JOBQ(QMQM/QMQM2) MAXACT(*NOMAX)
```

e. 發出下列指令，在佇列管理程式庫中建立重複的 QMQMJOBD:

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) TOLIB(QMlibraryname)
```

f. 發出下列指令，變更工作說明以使用新的工作佇列:

```
CHGJOBD JOBD(QMlibraryname/QMQMJOBD) JOBQ(QMQM/QMQM2)
```

g. 發出下列指令來啟動子系統:

```
STRSBS SBSD(QMQM/QMQM2)
```

註:

a. 您可以在任何檔案庫中指定子系統。 如果基於任何原因重新安裝產品或取代 QMQM 程式庫，則會移除您所做的任何變更。

b. 與 QMlibraryname 相關聯的所有佇列管理程式工作現在在子系統 QMQM2 下執行。

5. 下列範例會收集工作類型的所有輸出。

若要將多個佇列管理程式的所有檢查點處理程序 AMQALMPX 工作日誌收集至單一輸出佇列，請執行下列步驟:

a. 例如，建立輸出佇列

```
CRTOUTQ OUTQ(MYLIB/CHCKPTLOGS)
```

b. 使用您要控制的 IBM MQ 處理程序名稱來建立 QMQM/QMQMJOBD 工作說明的廣域副本，例如

```
CRTDUPOBJ OBJ(QMQMJOBD) FROMLIB(QMQM) OBJTYPE(*JOBD) NEWOBJ(AMQALMPX)
```

c. 變更工作說明上的輸出佇列參數以指向新的輸出佇列，並變更工作記載層次，以便將所有訊息寫入工作日誌。

```
CHGJOBD JOBD(QMQM/AMQALMPX) OUTQ(MYLIB/CHKPTLOGS) LOG(4 00 *SECLVL)
```

所有佇列管理程式的所有 IBM MQ AMQALMPX 工作都使用新的廣域 AMQALMPX 工作說明，但前提是本端佇列管理程式檔案庫中沒有本端置換工作說明。

這些工作的所有工作日誌排存檔現在都會寫入檔案庫 MYLIB 中的輸出佇列 CHKPTLOGS。

註:

a. 僅當 QPJOBLOG 或任何列印檔的輸出佇列參數值為 *JOB 時，前述範例才有效。 在前述範例中，QSYS/QPDJOBLOG 檔案需要將 OUTQ 設為 *JOB。

b. 若要變更系統列印檔，請使用 CHGPRTF 指令。 例如:

```
CHGPRTF PRTF(QJOBLOG) OUTQ(*JOB)
```

*JOB 選項指出必須使用您的工作說明。

c. 您可以將與 IBM MQ 工作相關聯的任何排存檔傳送至特定輸出佇列。 不過，請驗證所使用的列印檔具有 OUTQ 參數的適當值。

可用性、備份、回復及重新啟動

使用此資訊來瞭解 IBM MQ for IBM i 如何使用 IBM i 日誌登載支援來協助其備份及還原策略。

在閱讀本節之前，您必須先熟悉標準 IBM i 備份及回復方法，以及使用 IBM i 上的異動日誌及其相關異動日誌接收器。 如需這些主題的相關資訊，請參閱 備份及回復。

若要瞭解備份及回復策略，您首先需要瞭解 IBM MQ for IBM i 如何在 IBM i 檔案系統及整合檔案系統 (IFS) 中組織其資料。

IBM MQ for IBM i 會將其資料保留在每一個佇列管理程式實例的個別檔案庫中，以及 IFS 檔案系統中的串流檔中。

佇列管理程式特定檔案庫包含日誌登載、異動日誌接收器及控制佇列管理程式工作管理所需的物件。 IFS 目錄及檔案包含 IBM MQ 配置檔、 IBM MQ 物件的說明，以及它們包含的資料。

在將這些物件套用至適當的物件 之前 ，這些物件的每一項變更都會記錄在異動日誌中 (可透過系統失效來回復)。 這會透過重播日誌中所記錄的資訊來回復這類變更。

您可以將「IBM MQ for IBM i 」配置成使用不同伺服器上的多個佇列管理程式實例，以在伺服器或佇列管理程式失敗時提供更高的佇列管理程式可用性，並加快回復速度。

IBM MQ for IBM i 日誌

使用此資訊來瞭解 IBM MQ for IBM i 如何在其作業中使用日誌登載來控制本端物件的更新。

每一個佇列管理程式檔案庫都包含該佇列管理程式的日誌登載，且日誌登載具有名稱 QM *GRLIB*/AMQ *A* JRN，其中 QM *GRLIB* 是佇列管理程式檔案庫的名稱， *A* 是單一實例佇列管理程式的字母 A ，對於佇列管理程式實例而言是唯一的。

QM *GRLIB* 採用名稱 QM，後面接著唯一格式的佇列管理程式名稱。 例如，名為 TEST 的佇列管理程式具有名為 QMTEST 的佇列管理程式庫。 使用 **CRTMQM** 指令建立佇列管理程式時，可以指定佇列管理程式檔案庫。

異動日誌具有相關的異動日誌接收器，其中包含要登載的資訊。 接收器是只能附加資訊且最終會填滿的物件。

異動日誌接收器會使用過期資訊的寶貴磁碟空間。 不過，您可以將資訊放在永久儲存體中，以將這個問題降到最低。 在任何特定時間，都會將一個異動日誌接收器連接至異動日誌。 如果異動日誌接收器達到其預定的臨界值大小，則會將它分離並由新的異動日誌接收器取代。 當您使用 **CRTMQM** 及 **THRESHOLD** 參數建立佇列管理程式時，可以指定異動日誌接收器的臨界值。

與本端 IBM MQ for IBM i 異動日誌相關聯的異動日誌接收器存在於每一個佇列管理程式檔案庫中，並採用如下的命名慣例:

```
AMQ Arnnnnn
```

其中

A
是字母 A-Z。 對於單一實例佇列管理程式，它是 A 。 它會因多重實例佇列管理程式的不同實例而有所不同。

NNNNN
是順序中下一個異動日誌加 1 的十進位 00000 to 99999 。

r
是十進位 0 to 9，每次還原接收端時都會增加 1。

日誌的順序是根據日期。 不過，下一個異動日誌的命名是根據下列規則:

1. AMQArnnnnn 會移至 AMQAr(nnnnn+1)，並在 nnnnn 到達 99999 時折返。 例如， AMQA099999 會移至 AMQA000000，而 AMQA999999 會移至 AMQA900000。
2. 如果具有規則 1 所產生名稱的日誌已存在，則訊息 CPI7OE3 會傳送至 QSYSOPR 訊息佇列，且自動接收端切換會停止。

   繼續使用目前連接的接收端，直到您調查問題並手動連接新的接收端為止。
3. 如果順序中沒有可用的新名稱 (亦即，所有可能的異動日誌名稱都在系統上) ，則您需要執行下列兩項:

   a. 刪除不再需要的日誌 (請參閱 第 185 頁的『異動日誌管理』)。

b. 使用 (**RCDMQMIMG**) 將異動日誌變更記錄至最新的異動日誌接收器 然後重複前一個步驟。 這容許重複使用舊的異動日誌接收器名稱。

AMQAJRN 異動日誌會使用 MNGRCV(*SYSTEM) 選項，讓作業系統在達到臨界值時自動變更異動日誌接收器。 如需系統如何管理接收器的相關資訊，請參閱 *IBM i* 備份及回復。

異動日誌接收器的預設臨界值為 100,000 KB。 當您建立佇列管理程式時，可以將此值設為較大的值。 LogReceiverSize 屬性的起始值會寫入 mqs.ini 檔案的 LogDefaults 段落。

當異動日誌接收器延伸超出其指定的臨界值時，會分離接收器並建立新的異動日誌接收器，繼承前一個接收器的屬性。 當系統自動連接新的異動日誌接收器時，會忽略在建立佇列管理程式之後對 LogReceiverSize 或 LogASP 屬性所做的變更

如需配置系統的進一步詳細資料，請參閱 變更 IBM i 上的配置資訊 。

在建立佇列管理程式之後，如果您需要變更異動日誌接收器的大小，請使用下列指令建立新的異動日誌接收器，並將其擁有者設為 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
是佇列管理程式檔案庫的名稱

A
是實例 ID (通常是 A)。

rnnnnn
是先前說明的命名順序中的下一個異動日誌接收器

xxxxxx
是新的接收端臨界值 (KB)

註: 接收端的大小上限由作業系統控管。 若要檢查此值，請查看 **CRTJRNRCV** 指令上的 THRESHOLD 關鍵字。

現在，使用下列指令將新的接收器連接至 AMQAJRN 日誌登載:

```
CHGJRN JRN(QMGRLIB/AMQ A JRN) JRNRCV(QMGRLIB/AMQ Annnnnn)
```

如需如何管理這些異動日誌接收器的詳細資料，請參閱 第 185 頁的『異動日誌管理』 。

IBM MQ for IBM i 日誌登載使用情形

使用此資訊來瞭解 IBM MQ for IBM i 如何在其作業中使用日誌登載來控制本端物件的更新。

訊息佇列的持續更新分兩個階段進行。 代表更新的記錄會先寫入日誌登載，然後更新佇列檔。

因此，異動日誌接收器會變得比佇列檔更最新。 為了確保從一致點開始重新啟動處理程序， IBM MQ 會使用檢查點。

檢查點是指異動日誌中說明的記錄與佇列中的記錄相同的時間點。 檢查點本身是由重新啟動佇列管理程式所需的日誌登載記錄系列所組成。 例如，檢查點時所有作用中交易 (即工作單元) 的狀態。

IBM MQ 會自動產生檢查點。 當佇列管理程式啟動及關閉時，以及在記載特定數目的作業之後，即會採用它們。

您可以對佇列管理程式上的所有物件發出 RCDMQMIMG 指令並顯示結果，以強制佇列管理程式取得檢查點，如下所示:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(<Q_MGR_NAME>) DSPJRNDTA(*YES)
```

當佇列處理進一步訊息時，檢查點記錄會變成與佇列的現行狀態不一致。

當 IBM MQ 重新啟動時，它會在日誌中尋找最新的檢查點記錄。 此資訊保留在每個檢查點結束時更新的檢查點檔案中。 檢查點記錄代表日誌與資料之間的最新一致性點。 來自此檢查點的資料用來重建在檢查點時存在的佇列。 當重新建立佇列時，即會向前播放日誌，讓佇列回到系統失效或關閉之前的狀態。

若要瞭解 IBM MQ 如何使用日誌登載，請考量在佇列管理程式 TEST 中名為 TESTQ 的本端佇列的情況。 這由 IFS 檔案代表:

```
/QIBM/UserData/mqm/qmgrs/TEST/queues
```

如果指定的訊息放置在此佇列上，然後從佇列中擷取，則發生的動作會顯示在圖 第 183 頁的圖 33 中。

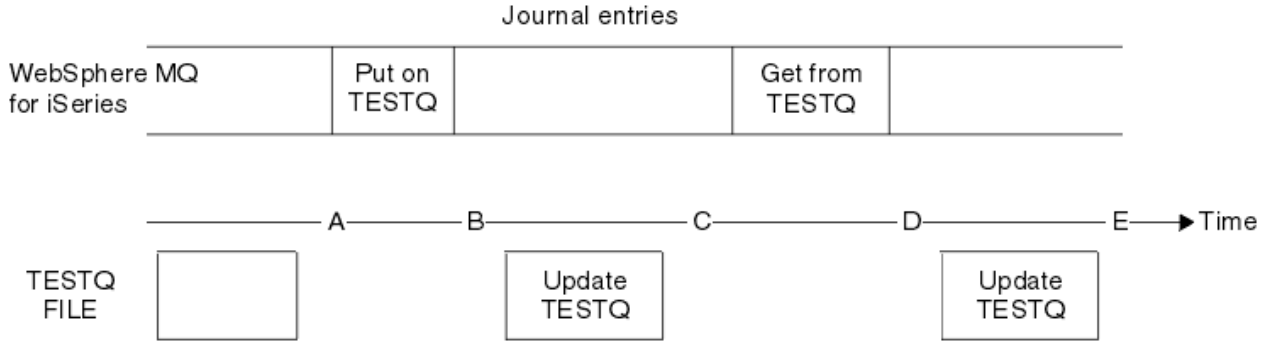


圖 *33:* 更新 *MQM* 物件時的事件順序

圖表中顯示的五個點 A 到 E 代表定義下列狀態的時間點:

A
佇列的 IFS 檔案表示法與異動日誌中包含的資訊一致。

B
異動日誌項目會寫入佇列上定義 Put 作業的異動日誌。

C
對佇列進行適當的更新。

D
日誌登載項目會寫入佇列中定義 Get 作業的日誌登載。

E
對佇列進行適當的更新。

IBM MQ for IBM i 回復功能的關鍵在於使用者可以將 TESTQ 的 IFS 檔案表示法儲存為時間 A，隨後透過還原已儲存的物件並從時間 A 開始重播日誌登載中的項目，以回復 TESTQ 的 IFS 檔案表示法 (即時間 E)。

在系統失效之後， IBM MQ for IBM i 會使用此策略來回復持續訊息。 IBM MQ 會記住異動日誌接收器中的特定項目，並確保在啟動時從此時開始重播異動日誌中的項目。 此啟動項目會定期重新計算，因此 IBM MQ 只需要在下一次啟動時執行必要的重播下限。

IBM MQ 提供物件的個別回復。 與物件相關的所有持續性資訊都會記錄在本端 IBM MQ for IBM i 日誌登載中。 任何變成損壞或毀損的 IBM MQ 物件都可以從保留在異動日誌中的資訊完全重建。

如需系統如何管理接收端的相關資訊，請參閱 第 180 頁的『可用性、備份、回復及重新啟動』。

媒體影像

使用此資訊來瞭解媒體映像檔，以及從媒體映像檔回復。

長持續時間的 IBM MQ 物件可以代表大量日誌登載項目，回到建立它的點。 為了避免此情況， IBM MQ for IBM i 具有物件的 媒體影像 概念。

此媒體影像是記錄在異動日誌中的 IBM MQ 物件完整副本。 如果取得物件的影像，則可以透過重播從此影像開始的異動日誌項目來重建物件。 異動日誌中代表每一個 IBM MQ 物件的重播點的項目稱為其 媒體回復項目。 IBM MQ 會追蹤:

- 每一個佇列管理程式物件的媒體回復項目。
- 此集合內最舊的項目 (如需詳細資料，請參閱 第 185 頁的『異動日誌管理』 中的錯誤訊息 AMQ7462 。

會定期取得 *CTLG 物件及 *MQM 物件的影像，因為這些物件對於佇列管理程式重新啟動非常重要。

其他物件的影像會在方便時拍攝。 依預設，當使用 **ENDMQM** 指令搭配參數 ENDCCTJOB (*YES) 來關閉佇列管理程式時，會取得 **所有** 物件的影像。 對於非常大的佇列管理程式，這項作業可能需要相當長的時間。 如果您需要快速關閉，請指定參數 RCDMQMIMG (*NO) 與 ENDCCTJOB (*YES)。 在這種情況下，建議您使用下列指令，在重新啟動佇列管理程式之後，將完整媒體映像檔記錄在日誌登載中:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(<Q_MGR_NAME>)
```

IBM MQ 會自動記錄物件的影像，如果它發現可由日誌中的小項目緊密說明物件的方便點。 不過，某些物件 (例如，一致包含大量訊息的佇列) 可能永遠不會發生這種情況。

請使用 IBM MQ 指令 RCDMQMIMG，它可讓您手動取得所選取物件的影像，而不是讓最舊媒體回復項目的日期持續一段不必要的長時間。

從媒體映像檔回復

如果發現部分物件已毀損或損壞，則 IBM MQ 會自動從其媒體映像檔回復這些物件。 尤其是在一般佇列管理程式啟動期間，這會套用至特殊 *MQM 及 *CTLG 物件。 如果在前次關閉佇列管理程式時有任何同步點交易未完成，則也會自動回復任何受影響的佇列，以完成啟動作業。

您必須使用 IBM MQ 指令 RCRMQMOBJ 手動回復其他物件。 此指令會重播日誌登載中的項目，以重建 IBM MQ 物件。 如果 IBM MQ 物件損壞，則唯一有效的動作是刪除它或透過此方法重建它。 不過請注意，無法以這種方式回復非持續訊息。

檢查點

在不同時間會採用檢查點，以提供已知的一致起始點來進行回復。

檢查點處理程序 AMQALMPX 負責在下列點取得檢查點:

- 佇列管理程式啟動 (STRMQM)。
- 佇列管理程式關閉 (ENDMQM)。
- 自前次檢查點以來已經過一段時間 (預設期間為 30 分鐘) ，且自前次檢查點以來已寫入日誌記錄數下限 (預設值為 100)。
- 在寫入一些日誌記錄之後。 預設值為 10 000。
- 在超出異動日誌臨界值大小且已自動建立新的異動日誌接收器之後。
- 使用下列項目取得完整媒體映像檔時:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(<Q_MGR_NAME>) DSPJRNDTA(*YES)
```

IBM MQ for IBM i 資料的備份

使用此資訊來瞭解每一個佇列管理程式的兩種 IBM MQ 備份類型。

對於每一個佇列管理程式，有兩種類型的 IBM MQ 備份可供考量:

- 資料及日誌登載備份。

  若要確保兩組資料一致，請僅在關閉佇列管理程式之後執行此動作。

- 日誌登載備份。

  您可以在佇列管理程式處於作用中狀態時執行此動作。

對於這兩種方法，您都需要尋找佇列管理程式 IFS 目錄及佇列管理程式檔案庫的名稱。 您可以在 IBM MQ 配置檔 (mqs.ini) 中找到這些檔案。 如需相關資訊，請參閱 QueueManager 段落。

請使用下列程序來執行這兩種類型的備份:

特定佇列管理程式的資料及日誌登載備份

註: 當佇列管理程式在執行中，請勿使用「作用中時儲存」要求。 除非已確定或回復具有擱置變更的所有確定定義，否則無法完成此類要求。 如果在佇列管理程式作用中時使用此指令，則通道連線可能不會正常結束。 請一律使用下列程序。

1. 使用下列指令建立空的異動日誌接收器:

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. 使用 **RCDMQMIMG** 指令來記錄所有 IBM MQ 物件的 MQM 映像檔，然後使用下列指令強制執行檢查點:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 結束通道並確定佇列管理程式不在執行中。 如果佇列管理程式正在執行中，請使用 **ENDMQM** 指令停止它。
4. 發出下列指令來備份佇列管理程式檔案庫:

```
SAVLIB LIB(QMTEST)
```

5. 發出下列指令來備份佇列管理程式 IFS 目錄:

```
SAV DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/test'))
```

特定佇列管理程式的日誌登載備份

因為所有相關資訊都保留在異動日誌中，只要您在某個時間執行完整儲存，就可以透過儲存異動日誌接收器來執行部分備份。 這些會記錄自完整備份以來的所有變更，並透過發出下列指令來執行:

1. 使用下列指令建立空的異動日誌接收器:

```
CHGJRN JRN(QMTEST/AMQAJRN) JRNRCV(*GEN)
```

2. 使用 **RCDMQMIMG** 指令來記錄所有 IBM MQ 物件的 MQM 映像檔，然後使用下列指令強制執行檢查點:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) MQMNAME(TEST)
```

3. 使用下列指令儲存異動日誌接收器:

```
SAVOBJ OBJ(AMQ*) LIB(QMTEST) OBJTYPE(*JRNRCV) .........
```

簡式備份策略是每週執行 IBM MQ 程式庫的完整備份，並執行每日日誌登載備份。 這當然取決於您如何為企業設定備份策略。

異動日誌管理

作為備份策略的一部分，請注意異動日誌接收器。 由於各種原因，從 IBM MQ 檔案庫中移除異動日誌接收器非常有用:

- 釋放空間; 這會套用至所有異動日誌接收器
- 啟動時增進效能 (STRMQM)
- 增進重建物件的效能 (RCRMQMOBJ)

在刪除異動日誌接收器之前，您必須小心擁有備份副本，且不再需要異動日誌接收器。

異動日誌接收器 在 已從異動日誌分離並儲存之後，只要它們在回復作業需要時可供還原，即可從佇列管理程式檔案庫中移除它們。

日誌登載管理的概念顯示在 第 186 頁的圖 34 中。

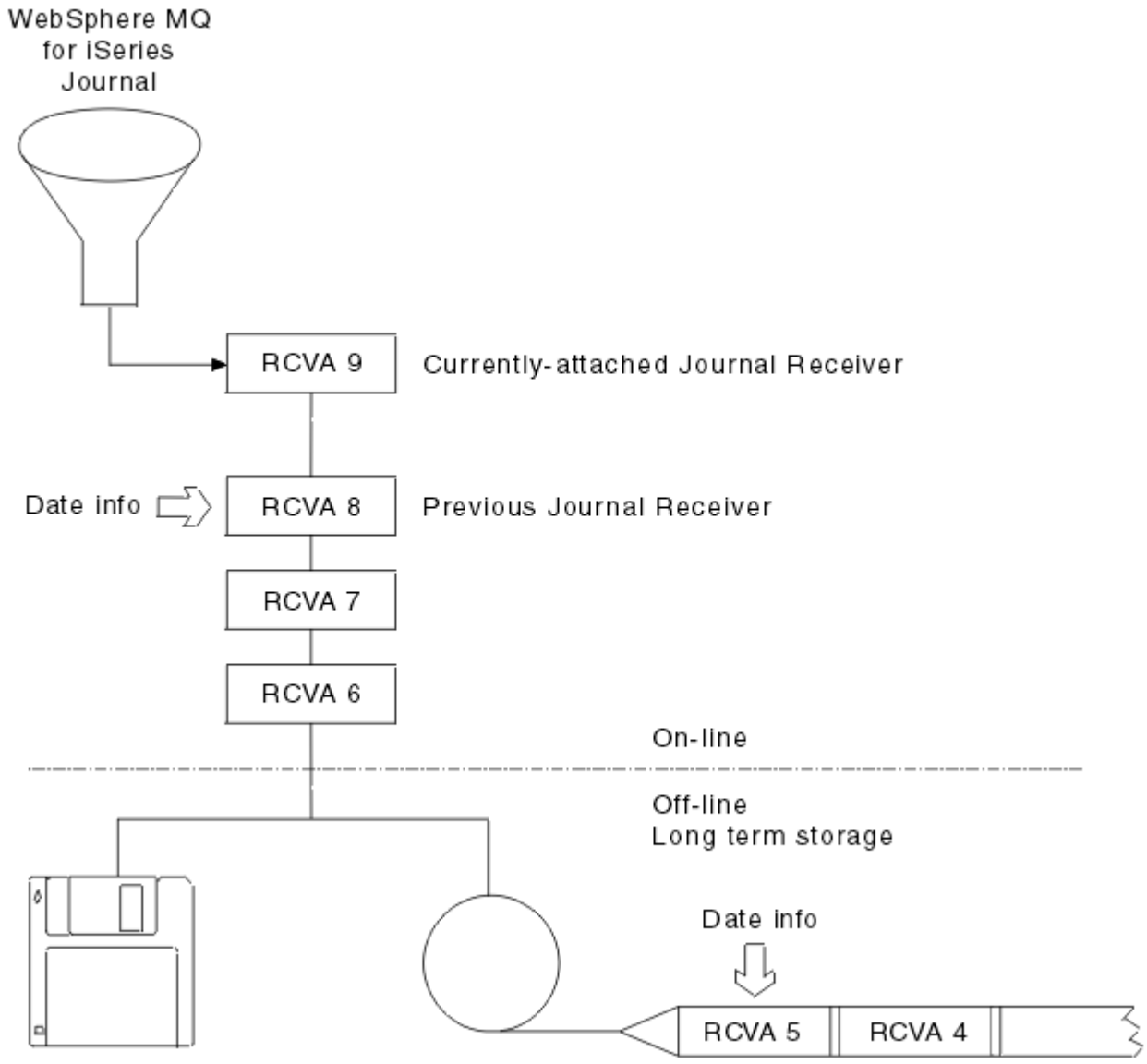


圖 *34: IBM MQ for IBM i* 日誌登載 *(journaling)*

請務必瞭解日誌登載 IBM MQ 可能需要回溯多久，以判定何時可以從佇列管理程式檔案庫中移除已備份的日誌登載接收端，以及何時可以捨棄備份本身。

IBM MQ 會向佇列管理程式訊息佇列 (佇列管理程式檔案庫中的 QMQMMSG) 發出兩則訊息，以協助判斷此時間。 當啟動時，當它變更本端異動日誌接收器，且您使用 RCDMQIMG 來強制檢查點時，即會發出這些訊息。 這兩個訊息如下:

AMQ7460
啟動回復點。 此訊息定義啟動項目的日期和時間， IBM MQ 會在啟動回復通過時從中重播日誌登載。 如果包含此記錄的異動日誌接收器在 IBM MQ 檔案庫中可用，則此訊息也會包含包含此記錄的異動日誌接收器名稱。

AMQ7462
最舊的媒體回復登錄。 此訊息定義用來從其媒體映像檔重建物件的最舊項目的日期和時間。

所識別的異動日誌接收器是所需要的最舊異動日誌接收器。 不再需要任何其他具有較舊建立日期的 IBM MQ 異動日誌接收器。 如果只顯示星星，則您需要從指示的日期還原備份，以判定哪一個是最舊的異動日誌接收器。

當記載這些訊息時， IBM MQ 也會將使用者空間物件寫入只包含一個項目的佇列管理程式檔案庫: 需要保留在系統上的最舊異動日誌接收器名稱。 此使用者空間稱為 AMQJRNINF ，且資料會以下列格式寫入:

```
JJJJJJJJJJLLLLLLLLLLYYYYMMDDHHMMSSmmm
```

其中:

JJJJJJJJJJ
是 IBM MQ 仍需要的最舊接收端名稱。

LLLLLLLLLL
是異動日誌接收器檔案庫名稱。

YYYY
是 IBM MQ 需要的最舊日誌登載項目的年份。

MM
是 IBM MQ 需要的最舊日誌登載項目的月份。

DD
是 IBM MQ 需要的最舊日誌登載項目的日期。

HH
是 IBM MQ 需要的最舊日誌登載項目的小時。

SS
是 IBM MQ 需要的最舊異動日誌項目的秒數。

mmm
是 IBM MQ 需要的最舊日誌登載項目的毫秒數。

從系統中刪除最舊的異動日誌接收器時，此使用者空間包含星號 (*) 作為異動日誌接收器名稱。

註: 定期執行 RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) DSPJRNDTA(*YES) 可以節省 IBM MQ 的啟動時間，並減少為了回復而需要儲存及還原的本端異動日誌接收器數目。

除非 IBM MQ for IBM i 正在對啟動或重建物件執行回復傳遞，否則它不會參照異動日誌接收器。 如果發現其需要的異動日誌不存在，則會向佇列管理程式訊息佇列 (QMQMMSG) 發出訊息 AMQ7432 ，報告完成回復傳遞所需的異動日誌項目的時間及日期。

如果發生此情況，請從備份中還原在此日期之後分離的所有異動日誌接收器，以容許回復傳遞成功。

保留包含啟動項目的異動日誌接收器，以及佇列管理程式檔案庫中可用的任何後續異動日誌接收器。

保留包含最舊 Media Recovery Entry 及任何後續異動日誌接收器的異動日誌接收器 (隨時可用) ，以及存在於佇列管理程式檔案庫或備份中。

當您強制檢查點時:

- 如果 AMQ7460 中所指名的異動日誌接收器未進階，則表示需要確定或回復不完整的工作單元。
- 如果未進階 AMQ7462 中所命名的異動日誌接收器，則表示有一個以上損壞的物件。

還原完整佇列管理程式 (資料及日誌登載)

使用此資訊可從備份或遠端機器還原一或多個佇列管理程式。

如果您需要從備份回復一或多個 IBM MQ 佇列管理程式，請執行下列步驟。

1. 靜止 IBM MQ 佇列管理程式。
2. 尋找最新備份集，由最新完整備份及後續備份的異動日誌接收器組成。
3. 發出下列指令，從完整備份執行 RSTLIB 作業，將 IBM MQ 資料檔案庫還原至完整備份時的狀態:

   ```
   RSTLIB LIB(QMQRLIB1) .........
   RSTLIB LIB(QMQRLIB2) .........
   ```

   如果異動日誌接收器部分儲存在一個異動日誌備份中，且完整儲存在後續的備份中，則只還原完全儲存的異動日誌接收器。 依時間順序個別還原日誌。

4. 使用下列指令執行 RST 作業，將 IBM MQ IFS 目錄還原至 IFS 檔案系統:

```
RST DEV(...) OBJ(('/QIBM/UserData/mqm/qmgrs/testqm')) ...
```

5. 啟動訊息佇列管理程式。 這會重播自完整備份以來寫入的所有日誌登載記錄，並將所有 IBM MQ 物件還原至日誌登載備份時的一致狀態。

如果您要在不同機器上還原完整佇列管理程式，請使用下列程序來還原佇列管理程式檔案庫中的所有項目。 (我們使用 TEST 作為範例佇列管理程式名稱。)

1. CRTMQM TEST
2. DLTLIB LIB(QMTEST)
3. RSTLIB SAVLIB(QMTEST) DEV(*SAVF) SAVF(QMGRLIBSAV)
4. 刪除下列 IFS 檔案:

   /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)

還原特定佇列管理程式的異動日誌接收器

使用此資訊來瞭解還原異動日誌接收器的不同方式。

最常見的動作是如果後續回復功能再次需要已移除的接收器，則將已備份的異動日誌接收器還原至佇列管理程式檔案庫。

這是一項簡式作業，需要使用標準 IBM i RSTOBJ 指令來還原異動日誌接收器:

```
RSTOBJ OBJ(QMQMDATA/AMQA000005) OBJTYPE(*JRNRCV) .........
```

可能需要還原一系列異動日誌接收器，而不是單一接收器。 例如， AMQA000007 是 IBM MQ 程式庫中最舊的接收端，且 AMQA000005 和 AMQA000006 都需要還原。

在此情況下，請依反向時間順序個別還原接收端。 這並不總是必要的，但卻是良好的實踐。 在嚴重狀況下，您可能需要使用 IBM i 指令 WRKJRNA ，將已還原的異動日誌接收器與異動日誌建立關聯。

還原異動日誌時，系統會以異動日誌接收器順序中的新名稱自動建立連接的異動日誌接收器。 不過，產生的新名稱可能與您需要還原的異動日誌接收器相同。 需要人為介入來克服此問題; 依序建立新的名稱異動日誌接收器，並在還原異動日誌接收器之前建立新的異動日誌。

例如，考量已儲存的異動日誌 AMQAJRN 及下列異動日誌接收器的問題:

- AMQA000000
- AMQA100000
- AMQA200000
- AMQA300000
- AMQA400000
- AMQA500000
- AMQA600000

- AMQA700000
- AMQA800000
- AMQA900000

將異動日誌 AMQAJRN 還原至佇列管理程式檔案庫時，系統會自動建立異動日誌接收器 AMQA000000。 此自動產生的接收器與您要還原且無法還原的其中一個現有異動日誌接收器 (AMQA000000) 衝突。

解決方案是:

1. 手動建立下一個異動日誌接收器 (請參閱 第 181 頁的『IBM MQ for IBM i 日誌』):

```
CRTJRNRCV JRNRCV(QMQRLIB/AMQA900001) THRESHOLD(XXXXX)
```

2. 使用異動日誌接收器手動建立異動日誌:

```
CRTJRN JRN(QMGRLIB/AMQAJRN) MNGRCV(*SYSTEM) +
JRNRCV(QMGRLIB/AMQA9000001) MSGQ(QMGRLIB/AMQAJRNMSG)
```

3. 將本端異動日誌接收器 AMQA000000 還原為 AMQA900000。

多重實例佇列管理程式

多重實例佇列管理程式透過在作用中伺服器失敗時自動切換至待用伺服器，來改善可用性。 作用中及待用伺服器是相同佇列管理程式的多重實例；它們共用相同的佇列管理程式資料。 如果作用中實例失敗，您需要將其日誌登載傳送至接管的待命資料庫，以便佇列管理程式可以重建其佇列。

配置您正在執行多重實例佇列管理程式的 IBM i 系統，以便在作用中佇列管理程式實例失敗時，它所使用的日誌登載可供接管的待命實例使用。 您可以設計自己的配置及管理作業，讓作用中實例的日誌登載可供接管的實例使用。 如果您不想遺失訊息，您的設計必須確保待命日誌與失敗點的作用中日誌一致。 您可以從兩個配置中的其中一個來調整您的設計，這些配置會在後續主題中以維護一致性的範例來說明。

1. 將日誌登載從執行作用中佇列管理程式實例的系統鏡映至執行待命實例的系統。
2. 將異動日誌放置在可從執行作用中實例的系統轉移到待命實例的「獨立輔助儲存區 (IASP)」中。

第一個解決方案不需要其他硬體或軟體，因為它使用基本 ASP。 第二個解決方案需要切換式 IASP ，其需要 IBM i 叢集作業支援，可作為個別計價的 IBM i 授權產品 5761-SS1 選項 41。

可靠性和可用性

多重實例佇列管理程式旨在改善應用程式的可用性。 技術和實體限制意味著您需要不同的解決方案來滿足災難回復、備份佇列管理程式及連續作業的需求。

在配置可靠性和可用性時，您會交換大量因素，產生四個不同的設計點:

災難回復
已針對毀損所有本端資產的重大災難之後的回復最佳化。

IBM i 上的災難回復通常基於 IASP 的地理鏡映。

備份
針對在本地化失敗之後的回復進行最佳化，通常是人為錯誤或某些無法預期的技術問題。

「 IBM MQ 」提供備份佇列管理程式，以定期備份佇列管理程式。 您也可以使用佇列管理程式日誌登載的非同步抄寫來改善備份的貨幣。

可用性
已針對快速還原作業進行最佳化，在可預見的技術故障 (例如伺服器或磁碟故障) 之後，提供幾乎不中斷的服務外觀。

回復通常以分鐘為單位來測量，偵測有時需要比回復處理程序更長的時間。 多重實例佇列管理程式可協助您配置 可用性。

連續作業
已針對提供不中斷服務進行最佳化。

連續操作解決方案必須解決偵測問題，幾乎總是涉及透過多個系統提交相同的工作，並使用第一個結果，或者如果正確性是主要考量，則至少比較兩個結果。

多重實例佇列管理程式可協助您配置 可用性。 一次有一個佇列管理程式實例在作用中。 根據系統的配置、載入及調整方式，切換至待命實例所需的時間從 10 秒多一點到 15 分鐘或以上。

如果與可重新連接的 IBM MQ MQI clients 搭配使用，則多重實例佇列管理程式可以提供幾乎不中斷服務的外觀，該佇列管理程式可以繼續處理，而無需應用程式察覺佇列管理程式中斷; 請參閱 自動化用戶端重新連線主題。

高可用性解決方案的元件

使用多重實例佇列管理程式來建構高可用性解決方案，方法是針對佇列管理程式資料提供健全的網路儲存體、佇列管理程式日誌登載的日誌抄寫或健全的 IASP 儲存體，以及使用配置為可重新啟動佇列管理程式服務之應用程式的可重新連接用戶端。

多重實例佇列管理程式會回復在另一部伺服器上啟動另一個佇列管理程式實例，以回應佇列管理程式失敗的偵測。 若要完成其啟動，實例需要存取網路儲存體中的共用佇列管理程式資料，以及其本端佇列管理程式日誌登載的副本。

若要建立高可用性解決方案，您需要管理佇列管理程式資料的可用性、本端佇列管理程式日誌登載的貨幣，以及建置可重新連接的用戶端應用程式，或將應用程式部署為佇列管理程式服務，以在佇列管理程式回復時自動重新啟動。 IBM MQ classes for Java 不支援自動重新連接用戶端。

佇列管理程式資料

將佇列管理程式資料放置在共用、高可用性及可靠的網路儲存體上，可能是使用 RAID 層次 1 磁碟或更高層次的磁碟。 檔案系統需要符合多重實例佇列管理程式之共用檔案系統的需求; 如需共用檔案系統需求的相關資訊，請參閱 共用檔案系統的需求。 「網路檔案系統第 4 版 (NFS4)」是符合這些需求的通訊協定。

佇列管理程式日誌登載

您還需要配置佇列管理程式實例使用的 IBM i 日誌登載，以便待命實例能夠將其佇列管理程式資料還原至一致狀態。 對於不中斷服務，這表示當作用中實例失敗時，您必須將日誌登載還原至其狀態。 與備份或災難回復解決方案不同，將異動日誌還原至較早的檢查點是不夠的。

您無法在網路儲存體上的多個 IBM i 系統之間實際共用日誌登載。 若要在失敗點將佇列管理程式日誌登載還原為一致狀態，您需要將失敗時作用中佇列管理程式實例的本端實體日誌登載，傳送至已啟動的新實例，或在執行中待命實例上維護日誌登載的鏡映。 鏡映異動日誌是已與屬於失敗實例的本端異動日誌保持完全同步的遠端異動日誌抄本。

三個配置是設計如何管理多重實例佇列管理程式的日誌登載的起點。

1. 使用從作用中實例 ASP 到待命實例 ASP 的同步化日誌登載抄寫 (日誌登載鏡映)。
2. 將您已配置為保留佇列管理程式日誌登載的 IASP 從作用中實例傳送至接管作為作用中實例的待命實例。
3. 使用已同步的次要 IASP 鏡映。

如需將佇列管理程式資料放入 iASP 的相關資訊，請參閱 IBM MQ IBM i CRTMQM 指令的 ASP 選項。

此外，請參閱 IBM Documentation 中 IBM i 資訊內的 高可用性 ，以及 管理者> 高可用性。

應用程式

若要建置用戶端以在待命佇列管理程式回復時自動重新連接至佇列管理程式，請使用 MQCONNX 將應用程式連接至佇列管理程式，並在 **MQCNO 選項** 欄位中指定 MQCNO_RECONNECT_Q_MGR 。 如需設計用戶端應用程式以進行回復的相關資訊，請參閱 高可用性範例程式 ，以取得使用可重新連接用戶端的三個範例程式，以及 應用程式回復 。

使用 *NetServer* 建立佇列管理程式資料的網路共用

在 IBM i 伺服器上建立網路共用，以儲存佇列管理程式資料。 設定來自兩部伺服器 (即將管理佇列管理程式實例) 的連線，以存取網路共用。

開始之前

- 此作業需要三部 IBM i 伺服器。 網路共用定義在其中一部伺服器 GAMMA 上。 其他兩部伺服器 (ALPHA 和 BETA) 將連接至 GAMMA。
- 在所有三部伺服器上安裝 IBM MQ 。
- 安裝 System i Navigator; 請參閱 System i Navigator。

關於這項作業

- 在 GAMMA 上建立佇列管理程式目錄，並設定使用者設定檔 QMQM 及 QMQMADM 的正確所有權及許可權。 在 GAMMA 上安裝 IBM MQ 可輕鬆建立目錄及許可權。
- 使用 System i Navigator 來建立與 GAMMA 上佇列管理程式資料目錄的共用。
- 在 ALPHA 和 BETA 上建立指向共用的目錄。

程序

1. 在 GAMMA 上，建立目錄以管理佇列管理程式資料，並以 QMQM 使用者設定檔作為擁有者，並以 QMQMADM 作為主要群組。

   提示:

   建立具有正確許可權的目錄的快速可靠方法是在 GAMMA 上安裝 IBM MQ 。

   稍後，如果您不想在 GAMMA 上執行 IBM MQ ，請解除安裝 IBM MQ。 解除安裝之後，目錄 /QIBM/UserData/mqm/qmgrs 會保留在 GAMMA 上，擁有者為 QMQM 使用者設定檔，以及 QMQMADM 主要群組。

   作業會使用 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 目錄來進行共用。
2. 啟動 System i Navigator **新增連線** 精靈，並連接至 GAMMA 系統。
   a) 按兩下 Windows 桌面上的 **System i Navigator** 圖示。
   b) 按一下 **是** 以建立連線。
   c) 遵循 **新增連線** 精靈中的指示，並建立從 IBM i 系統到 GAMMA 的連線。

      GAMMA 的連線會新增至 **我的連線**。
3. 在 GAMMA 上新增檔案共用。
   a) 在「 **System i Navigator** 」視窗中，按一下 My Connections/GAMMA/File Systems 中的 File Shares 資料夾。
   b) 在「 **我的作業** 」視窗中，按一下 **管理 IBM i NetServer 共用**。

      即會在桌面上開啟新視窗 **IBM i NetServer -GAMMA**，並顯示共用物件。
   c) 用滑鼠右鍵按一下 Shared Objects 資料夾> **檔案** > **新建** > **檔案**。

      即會開啟新視窗 **IBM i NetServer 檔案共用-GAMMA**。
   d) 例如，為共用提供名稱 WMQ 。
   e) 將存取控制設為 Read/Write。
   f) 瀏覽至您先前建立的 /QIBM/UserData/mqm/qmgrs 目錄，以選取 **路徑名稱** ，然後按一下 **確定**。

      IBM i NetServer 檔案共用-GAMMA 視窗會關閉，且 WMQ 會列在共用物件視窗中。
4. 在共用物件視窗中，用滑鼠右鍵按一下 **WMQ** 。 按一下 **檔案** > **許可權**。

   會針對物件 /QIBM/UserData/mqm/qmgrs 開啟 **Qmgrs 許可權-GAMMA** 視窗。
   a) 請檢查 QMQM 的下列許可權 (如果尚未設定的話):

      ```
      Read
      Write
      Execute
      Management
      ```

Existence
Alter
Reference

b) 請檢查 QMQMADM 的下列許可權 (如果尚未設定的話):

Read
Write
Execute
Reference

c) 新增您要授與 /QIBM/UserData/mqm/qmgrs 許可權的其他使用者設定檔。

例如，您可以將預設使用者設定檔 (公用) Read 及 Execute 許可權提供給 /QIBM/UserData/mqm/qmgrs。

5. 請檢查所有獲授與 GAMMA 上 /QIBM/UserData/mqm/qmgrs 存取權的使用者設定檔，其密碼與存取 GAMMA 的伺服器上的密碼相同。

特別是，請確保要存取共用之其他伺服器上的 QMQM 使用者設定檔，與 GAMMA 上的 QMQM 使用者設定檔具有相同的密碼。

提示: 按一下「 System i Navigator 」中的 My Connections/GAMMA/Users and Groups 資料夾，以設定密碼。 或者，使用 **CHFUSRPRF** 和 **CHGPWD** 指令。

結果

確認您可以使用共用從其他伺服器存取 GAMMA。 如果您正在執行其他作業，請檢查您是否可以使用路徑 /QNTC/GAMMA/WMQ 從 ALPHA 和 BETA 存取 GAMMA。 如果 /QNTC/GAMMA 目錄不存在於 ALPHA 或 BETA 上，則您必須建立該目錄。 視 NetServer 網域而定，在建立目錄之前，您可能必須先 IPL ALPHA 或 BETA。

```
CRTDIR DIR('/QNTC/GAMMA')
```

當您檢查是否可以從發出指令的 ALPHA 或 BETA 存取 /QNTC/GAMMA/WMQ 時， CRTMQM MQMNAME('QM1') MQMDIRP('/QNTC/GAMMA/WMQ') 會在 GAMMA 上建立 /QIBM/UserData/mqm/qmgrs/QM1 。

下一步

遵循作業 第 201 頁的『使用日誌登載鏡映及 NetServer 建立多重實例佇列管理程式』 或 第 204 頁的『使用 NetServer 及日誌登載鏡映將單一實例佇列管理程式轉換為多重實例佇列管理程式』中的步驟來建立多重實例佇列管理程式。

失效接手效能

偵測佇列管理程式實例所花費的時間已失敗，然後在待命資料庫上回復處理所花費的時間可能會根據配置而在數十秒至 15 分鐘之間或更長。 在設計及測試高可用性解決方案時，效能需要成為主要考量。

在決定是配置多重實例佇列管理程式來使用日誌登載抄寫還是使用 IASP 時，需要權衡一些優點和缺點。 鏡映需要佇列管理程式同步寫入遠端異動日誌。 從硬體觀點來看，這不需要影響效能，但從軟體觀點來看，寫入遠端異動日誌所涉及的路徑長度大於僅寫入本端異動日誌所涉及的路徑長度，這可能會在某種程度上降低執行中佇列管理程式的效能。 不過，當待命佇列管理程式接管時，在失敗之前從作用中實例所維護的遠端日誌登載同步化其本端日誌登載的延遲，通常與 IBM i 偵測 IASP 並將其傳送至執行佇列管理程式待命實例的伺服器所花費的時間相比較小。 IASP 傳送時間可以多達 10 到 15 分鐘，而不是在秒內完成。 IASP 傳送時間取決於將 IASP 傳送至待命系統時需要 轉接 的物件數，以及需要合併的存取路徑或索引大小。

當待命佇列管理程式接管時，在失敗之前從作用中實例所維護的遠端日誌中同步化其本端日誌登載的延遲，通常與 IBM i 偵測獨立 ASP 並將其傳送至執行佇列管理程式待命實例的伺服器所花費的時間相比較小。 獨立 ASP 傳送時間可以多達 10 到 15 分鐘，而不是在秒內完成。 獨立 ASP 傳送時間取決於當獨立 ASP 傳送至待命系統時需要 轉接 的物件數，以及需要合併的存取路徑或索引大小。

不過，傳送日誌不是影響待命實例完全回復所花費時間的唯一因素。 您也需要考量網路檔案系統釋放佇列管理程式資料鎖定所花費的時間，這些資料會向待命實例發出信號以嘗試繼續其啟動，以及從日誌登載回復佇

列所花費的時間，以便實例能夠重新開始處理訊息。 這些其他延遲來源都會增加啟動待命實例所花費的時間。 切換時間總計由下列元件組成:

失敗偵測時間
NFS 釋放佇列管理程式資料鎖定所花費的時間，以及待命實例繼續其啟動程序所花費的時間。

傳送時間
如果是 HA 叢集，則為 IBM i 將 IASP 從管理作用中實例的系統傳送至待命實例所花費的時間; 如果是日誌登載抄寫，則為使用來自遠端抄本的資料更新待命資料庫的本端日誌登載所花費的時間。

重新啟動時間
新作用中佇列管理程式實例從其還原日誌中的最新檢查點重建其佇列及回復處理訊息所花費的時間。

註:

如果已接管的待命實例配置為同步抄寫至先前作用中的實例，則啟動可能會延遲。 如果遠端日誌位於管理先前作用中實例的伺服器上，且伺服器失敗，則新的已啟動實例可能無法抄寫至其遠端日誌登載。

等待同步回應的預設時間是一分鐘。 您可以配置抄寫逾時之前的延遲上限。 或者，您可以配置待命實例，以開始使用非同步抄寫至失敗的作用中實例。 稍後，當失敗實例再次在待命上執行時，您會將切換至同步抄寫。 相同的考量也適用於使用同步獨立 ASP 鏡映。

您可以針對這些元件進行個別基準線測量，以協助您評量失效接手的整體時間，並將要使用的配置方法納入決策中。 在做出最佳配置決策時，您也需要考量相同伺服器上的其他應用程式如何進行失效接手，以及是否有備份或災難回復處理程序已使用 IASP。

透過調整叢集配置，可以縮短 IASP 傳送時間:

1. 叢集中跨系統的使用者設定檔應該具有相同的 GID 和 UID ，而不需要轉接處理程序來變更 UID 和 GID。
2. 將系統及基本使用者磁碟儲存區中的資料庫物件數目縮至最小，因為需要合併這些物件，以建立磁碟儲存區群組的交互參照表格。
3. 如需進一步的效能提示，請參閱 IBM 紅皮書: *Implementing PowerHA for IBM i(SG24-7405*)。

使用基本 ASP、異動日誌鏡映及小型配置的配置應該以數十秒的時間順序切換。

結合 *IBM i* 叢集功能與 *IBM MQ* 叢集作業的概觀

在 IBM i 上執行 IBM MQ ，並利用 IBM i 叢集作業功能，可以提供比僅使用 IBM MQ 叢集作業更完整的「高可用性」解決方案。

若要具有此功能，您需要設定:

1. IBM i 機器上的叢集; 請參閱 第 193 頁的『IBM i 叢集』
2. 您將佇列管理程式移至其中的獨立輔助儲存區 (IASP); 請參閱 第 193 頁的『獨立輔助儲存區 (IASP)』
3. 叢集資源群組 (CRG); 請參閱 第 194 頁的『裝置叢集資源群組』，您在其中定義:
   - 回復網域
   - IASP
   - 結束程式; 請參閱 第 194 頁的『裝置 CRG 跳出程式』

IBM i 叢集

IBM i 叢集是邏輯上鏈結在一起的實例 (即 IBM i 電腦或分割區) 集合。

此分組的目的是容許備份每一個實例，消除單一失敗點，並增加應用程式及資料備援。 建立叢集之後，可以配置各種叢集資源群組 (CRG) 類型來管理叢集中的應用程式、資料及裝置。

如需進一步資訊，請參閱 建立叢集 及 建立叢集 (CRTCLU) 指令。

獨立輔助儲存區 (IASP)

IASP 是一種使用者 ASP 類型，可作為單一層次儲存體的延伸。 它是儲存體的一部分，由於其獨立於系統儲存體，因此可以輕鬆地操作，而不需要對系統進行 IPL。

IASP 可以輕鬆切換至另一個作業系統實例，或抄寫至另一個作業系統實例上的目標 IASP。 可以使用兩種方法在實例之間切換 IASP:

- 第一種方法需要使用「高速鏈結 (HSL)」迴圈來連接叢集中的所有電腦，以及包含 IASP 的切換式磁碟直立式主機。
- 第二個方法要求作業系統實例必須是可在分割區之間切換輸入/輸出處理器 (IOP) 之相同 IBM i 電腦上的分割區。 不需要特殊硬體即可抄寫 IASP。 透過網路使用 TCP/IP 執行抄寫。

如需相關資訊，請參閱 配置裝置 ASP (CFGDEVASP) 指令。

裝置叢集資源群組

叢集資源群組 (CRG) 有多種類型。 如需不同可用 CRG 類型的相關資訊，請參閱 叢集資源群組。

本主題集中於裝置 CRG。 裝置 CRG:

- 說明及管理裝置資源，例如獨立輔助儲存區 (IASP)。
- 定義叢集節點的回復網域
- 指派裝置，以及
- 指派將處理叢集事件的結束程式。

回復網域表示將哪些叢集節點視為主要節點。 其餘節點會被視為備份。 備份節點也會在回復網域中排序，指定哪個節點是第一個備份、第二個備份等等，視回復網域中有多少節點而定。

如果主要節點失敗，跳出程式會在回復網域中的所有節點上執行。 然後，在第一個備份上執行的跳出程式可以進行必要的起始設定，以讓此節點成為新的主要節點。

如需相關資訊，請參閱 建立裝置 CRG 及 建立叢集資源群組 (CRTCRG) 指令。

裝置 CRG 跳出程式

當回復網域定義的其中一個節點中發生事件時，作業系統叢集資源服務會呼叫裝置 CRG 跳出程式; 例如，失效接手或切換事件。

當叢集的主要節點失敗且 CRG 與它們所管理的所有資源一起切換時，會發生失效接手事件，而當特定的 CRG 從主要節點手動切換至備份節點時，會發生切換事件。

不論任何一種方式，跳出程式都會負責起始設定及啟動在前一個主要節點上執行的所有程式，這會將第一個備份節點轉換成新的主要節點。

例如，使用 IBM MQ 時，結束程式應該負責啟動 IBM MQ 子系統 (QMQM) 及佇列管理程式。 佇列管理程式應該配置成自動啟動接聽器和服務，例如觸發監視器。

切換式 IASP 配置

IBM MQ 可以設定為利用 IBM i 的叢集功能。 若要執行此作業:

1. 在資料中心系統之間建立 IBM i 叢集
2. 將佇列管理程式移至 IASP。

   第 195 頁的『將佇列管理程式移至獨立輔助儲存區或從獨立輔助儲存區移除佇列管理程式』 包含一些範例程式碼，可協助您執行這項作業。
3. 您需要建立 CRG ，以定義回復網域、IASP 及跳出程式。

   第 195 頁的『配置裝置叢集資源群組』 包含一些範例程式碼，可協助您執行這項作業。

相關概念

第 213 頁的『獨立 ASP 及高可用性』

獨立 ASP 可讓應用程式及資料在伺服器之間移動。 獨立 ASP 的彈性表示它們是部分 IBM i 高可用性解決方案的基礎。 在考量對佇列管理程式異動日誌使用 ASP 或獨立 ASP 時，您應該考量根據獨立 ASP 的其他高可用性配置。

配置裝置叢集資源群組
設定裝置叢集資源群組 (CRG) 的範例程式。

關於這項作業

在下列範例中，請注意:

- [PRIMARY SITE NAME] 和 [BACKUP SITE NAME] 可以是八個字元或更少的任何兩個不同字串。
- [PRIMARY IP] 和 [BACKUP IP] 是用於鏡映的 IP。

程序

1. 識別叢集的名稱。
2. 識別 CRG 跳出程式名稱及檔案庫。
3. 決定此 CRG 要定義的主要節點及備份節點名稱。
4. 識別要由此 CRG 管理的 IASP，並確定已在主要節點下建立它。
5. 使用下列指令在備份節點中建立裝置說明:

```
 CRTDEVASP DEVD([IASP NAME]) RSRCNAME([IASP NAME])
```

6. 使用下列指令，將接管 IP 位址新增至所有節點:

```
ADDTCPIFC INTNETADR(' [TAKEOVER IP]') LIND([LINE DESC])
SUBNETMASK('[SUBNET MASK]') AUTOSTART(*NO)
```

7. 使用下列指令，僅在主要節點中啟動接管 IP 位址:

```
STRTCPIFC INTNETADR('[TAKEOVER IP')
```

8. 選擇性的: 如果您的 IASP 是可切換的，請呼叫此指令:

```
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. 選擇性的: 如果要鏡映 IASP，請呼叫下列指令:

```
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]'))
```

將佇列管理程式移至獨立輔助儲存區或從獨立輔助儲存區移除佇列管理程式
將佇列管理程式移至獨立輔助儲存區 (IASP) 的範例程式，以及從 IASP 移除佇列管理程式的指令。

關於這項作業

在下列範例中，請注意:

- [MANAGER NAME] 是佇列管理程式的名稱。
- [IASP NAME] 是 IASP 的名稱。
- [MANAGER LIBRARY] 是佇列管理程式檔案庫的名稱。
- [MANAGER DIRECTORY] 是佇列管理程式目錄的名稱。

程序

1. 識別您的主要節點和備份節點。
2. 在主要節點上執行下列程序:

a) 請確定您的佇列管理程式已結束。

b) 使用指令，確定您的 IASP 是 vary on

```
VRYCFG CFGOBJ([IASP NAME]) CFGTYPE(*DEV) STATUS(*ON)
```

c) 在 IASP 下建立佇列管理程式目錄。
根目錄下將會有一個具有 IASP 名稱的目錄，即:

```
QSH CMD('mkdir -p /[IASP_NAME]/QIBM/UserData/mqm/qmgrs/')
```

d) 使用下列指令，將管理程式的 IFS 物件移至您剛在 IASP 下建立的佇列管理程式目錄:

```
QSH CMD('mv /QIBM/UserData/mqm/qmgrs/[MANAGER NAME]
/[IASP NAME]/QIBM/UserData/mqm/qmgrs')
```

e) 使用下列指令，建立名為 MGRLIB 的暫時儲存檔:

```
CRTSAVF QGPL/MGRLIB
```

f) 使用下列指令，將佇列管理程式檔案庫儲存至 MGRLIB 儲存檔:

```
SAVLIB LIB([MANGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
```

g) 使用下列指令來刪除佇列管理程式檔案庫，並忽略所有查詢訊息:

```
DLTLIB [MANAGER LIBRARY]
```

h) 使用下列指令，將佇列管理程式檔案庫還原至 IASP:

```
RSTLIB SAVLIB([MANAGER LIBRARY]) DEV(*SAVF) SAVF(QGPL/MGRLIB)
RSTASPDEV([IASP NAME])
```

i) 使用下列指令來刪除暫時儲存檔:

```
 DLTF FILE(QGPL/MGRLIB)
```

j) 使用下列指令，建立 IASP 下佇列管理程式 IFS 物件的符號鏈結:

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

k) 使用下列指令連接至 IASP:

```
SETASPGRP [IASP NAME]
```

l) 使用下列指令來啟動佇列管理程式:

```
STRMQM [MANAGER NAME]
```

3. 在一或多個備份節點上執行下列程序:

a) 使用下列指令來建立暫時佇列管理程式目錄:

```
QSH CMD('mkdir -p /[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

b) 使用下列指令，建立佇列管理程式暫存目錄的符號鏈結:

```
ADDLNK OBJ('/[IASP NAME]/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
NEWLNK('/QIBM/UserData/mqm/qmgrs/[MANAGER NAME]')
```

c) 使用下列指令來刪除暫存目錄:

```
QSH CMD('rm -r /[IASP NAME]')
```

d) 在檔案 /QIBM/UserData/mqm/mqs.ini 結尾新增下列:

```
QueueManager:
Name=[MANAGER NAME]
Prefix=/QIBM/UserData/mqm
Library=[MANAGER LIBRARY]
Directory=[MANAGER DIRECTORY]
```

4. 若要從 IASP 移除佇列管理程式，請發出下列指令:
   a) VRYCFG CFGOBJ ([IASP NAME]) CFGTYPE (*DEV) STATUS (*ON)
   b) SETASPGRP [IASP 名稱]
   c) ENDMQM [MANAGER 名稱]
   d) DLTMQM [MANAGER 名稱]

ASP 的鏡映異動日誌配置

使用鏡映日誌登載之間的同步抄寫來配置健全的多重實例佇列管理程式。

鏡映佇列管理程式配置使用在基本或獨立輔助儲存區 (ASP) 中建立的異動日誌。

在 IBM i 上，佇列管理程式資料會寫入日誌登載及檔案系統。 日誌登載包含佇列管理程式資料的正本。 在系統之間使用同步或非同步日誌登載抄寫來共用日誌登載。 需要混合本端及遠端異動日誌，才能重新啟動佇列管理程式實例。 佇列管理程式重新啟動會從伺服器上本端及遠端異動日誌的混合，以及共用網路檔案系統上的佇列管理程式資料中讀取日誌登載記錄。 檔案系統中的資料可加速重新啟動佇列管理程式。 檢查點儲存在檔案系統中，標示檔案系統與日誌登載之間的同步點。 一般佇列管理程式重新啟動時，不需要在檢查點之前儲存的日誌登載記錄。 不過，檔案系統中的資料可能不是最新的，且會使用檢查點之後的日誌登載記錄來完成佇列管理程式重新啟動。 連接至實例的日誌登載中的資料會保持最新，以便重新啟動可以順利完成。

但如果正在非同步抄寫待命伺服器上的遠端日誌，且在同步化之前發生失敗，則即使日誌登載記錄也可能不是最新的。 如果您決定使用未同步的遠端日誌登載來重新啟動佇列管理程式，待命佇列管理程式實例可能會重新處理在作用中實例失敗之前刪除的訊息，或不處理在作用中實例失敗之前收到的訊息。

另一個罕見的可能性是檔案系統包含最新的檢查點記錄，而待命資料庫上未同步的遠端異動日誌則不包含。 在此情況下，佇列管理程式不會自動重新啟動。 您可以選擇等待遠端異動日誌同步化，或從檔案系統冷啟動待命佇列管理程式。 即使在此情況下，檔案系統包含比遠端日誌登載更新的佇列管理程式資料檢查點，它可能未包含作用中實例失敗之前已處理的所有訊息。 在與日誌登載不同步的冷重新啟動之後，部分訊息可能重新處理，部分未處理。

使用多重實例佇列管理程式時，也會使用檔案系統來控制佇列管理程式的哪些實例處於作用中，以及哪些實例處於待命狀態。 作用中實例會獲得佇列管理程式資料的鎖定。 待命資料庫會等待獲得鎖定，當取得鎖定時，它會變成作用中實例。 如果作用中實例正常結束，則會釋放鎖定。 如果檔案系統偵測到作用中實例失敗，或無法存取檔案系統，則檔案系統會釋放鎖定。 檔案系統必須符合偵測失敗的需求; 請參閱 共用檔案系統的需求。

IBM i 上多重實例佇列管理程式的架構提供在伺服器或佇列管理程式失敗之後自動重新啟動。 它也支援在儲存佇列管理程式資料的檔案系統失敗之後還原佇列管理程式資料。

在 第 198 頁的圖 35 中，如果 ALPHA 失敗，您可以使用鏡映日誌登載在測試版上手動重新啟動 QM1 。 透過將多重實例佇列管理程式功能新增至 QM1，如果 ALPHA 上的作用中實例失敗，則 QM1 的待命實例會自動在測試版上回復。 如果伺服器 ALPHA 失敗，則 QM1 也可以自動回復，而不只是 QM1 的作用中實例。 一旦測試版變成作用中佇列管理程式實例的主機，即可在 ALPHA 上啟動待命實例。

第 198 頁的圖 35 顯示使用 NetServer 來鏡映佇列管理程式兩個實例之間的日誌登載，以儲存佇列管理程式資料的配置。 您可以展開型樣以包括更多日誌登載，從而包括更多實例。 遵循 第 181 頁的『IBM MQ for IBM i 日誌』主題中說明的日誌登載命名規則。 目前佇列管理程式的執行中實例數限制為兩個，一個在作用中，另一個在待命中。

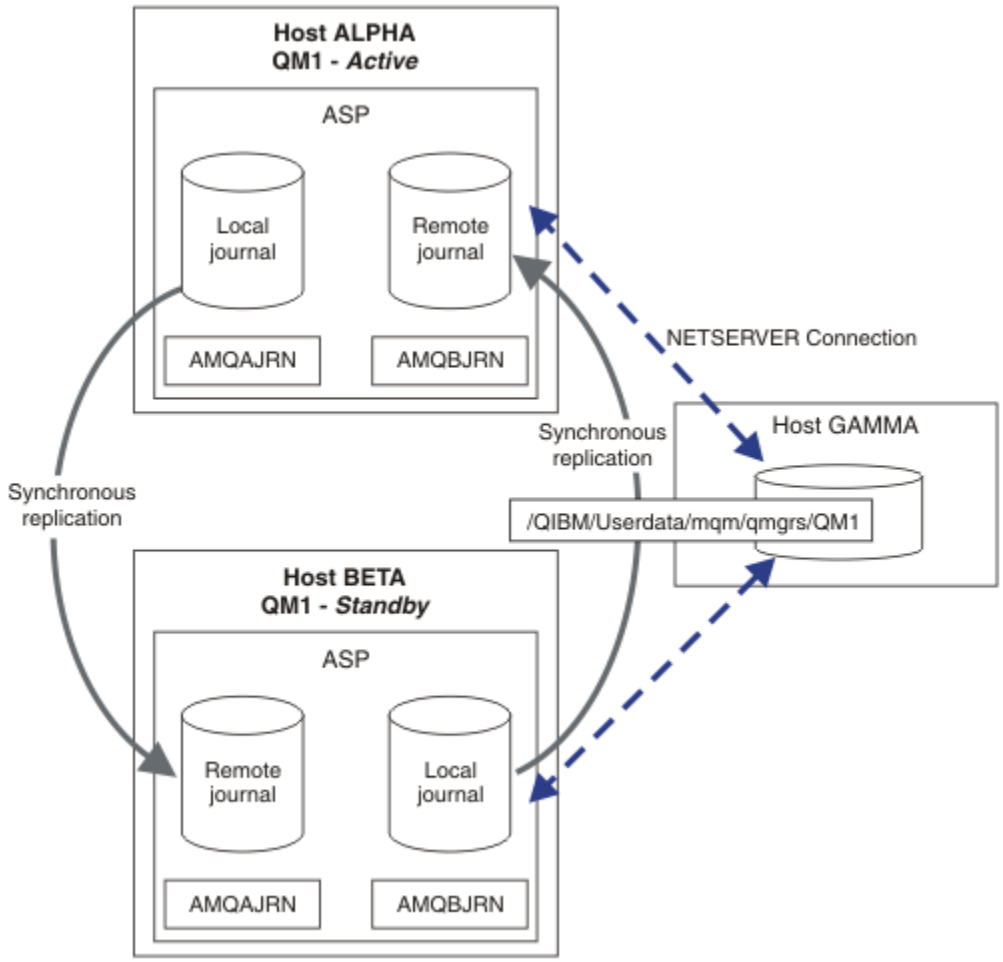


圖 *35:* 鏡映佇列管理程式日誌登載

主機 ALPHA 上 QM1 的本端日誌稱為 AMQAJRN (或更完整的 QMQM1/AMQAJRN)，在測試版上，日誌登載為 QMQM1/AMQBJRN。 每一個本端日誌都會抄寫至佇列管理程式所有其他實例上的遠端日誌登載。 如果佇列管理程式已配置兩個實例，則會將本端日誌登載抄寫至一個遠端日誌登載。

*SYNC 或 *ASYNC 遠端異動日誌抄寫

使用同步 (*SYNC) 來鏡映 IBM i 異動日誌 或非同步 (*ASYNC) 日誌登載; 請參閱 遠端異動日誌管理。

第 198 頁的圖 35 中的抄寫模式是 *SYNC，而不是 *ASYNC。 *ASYNC 更快，但如果遠端異動日誌狀態為 *ASYNCPEND 時發生失敗，則本端與遠端異動日誌不一致。 遠端異動日誌必須趕上本端異動日誌。 如果您選擇 *SYNC，則本端系統會等待遠端異動日誌，然後再從需要完成寫入的呼叫返回。 本端及遠端異動日誌通常會彼此保持一致。 僅當 *SYNC 作業花費的時間超過指定的時間時 [1]，且遠端日誌登載已取消啟動，請執行異動日誌不同步。 錯誤會記載至日誌登載訊息佇列及 QSYSOPR。 佇列管理程式會偵測此訊息，將錯誤寫入佇列管理程式錯誤日誌，並取消啟動佇列管理程式日誌登載的遠端抄寫。 作用中佇列管理程式實例會回復，而不會遠端登載至此日誌登載。 當遠端伺服器再次可用時，您必須手動重新啟動同步遠端異動日誌抄寫。 然後會重新同步化日誌。

第 198 頁的圖 35 中說明的 *SYNC / *SYNC 配置的問題是 BETA 上待命佇列管理程式實例如何控制。 只要 BETA 上的佇列管理程式實例寫入其第一個持續訊息，它就會嘗試更新 ALPHA 上的遠端日誌登載。 如果控制從 ALPHA 傳遞至 BETA 的原因是 ALPHA 失敗，且 ALPHA 仍關閉，則遠端日誌登載至 ALPHA 會失敗。 BETA 會等待 ALPHA 回應，然後取消啟動遠端日誌登載，並僅使用本端日誌登載回復處理訊息。 BETA 必須等待一段時間，才能偵測到 ALPHA 已關閉，導致閒置一段時間。

[1] 在 IBM i 第 5 版上，指定的時間是 60 秒，在 IBM i 6.1 開始的時間範圍是 1-3600 秒。

選擇將遠端日誌登載設定為 *SYNC 或 *ASYNC 是取捨。 第 199 頁的表 14 彙總在一對佇列管理程式之間使用 *SYNC 與 *ASYNC 日誌登載之間的取捨:

表 *14:* 遠端日誌登載選項

作用中 \ 待用	***SYNC**	***ASYNC**
***SYNC**	1. 一致切換及失效接手 2. 失效接手之後，待命實例不會立即回復。 3. 遠端日誌登載必須隨時可用 4. 佇列管理程式效能取決於遠端日誌登載	1. 一致切換及失效接手 2. 當待命伺服器可用時，遠端日誌登載必須切換至 *SYNC 3. 遠端日誌登載在重新啟動之後必須保持可用 4. 佇列管理程式效能取決於遠端日誌登載
***ASYNC**	1. 不是明智的組合	1. 在失效接手或切換之後，部分訊息可能遺失或重複 2. 待命實例不需要隨時可用，作用中實例才能繼續而不延遲。 3. 效能與遠端日誌登載無關

***SYNC / *SYNC**

作用中佇列管理程式實例會使用 *SYNC 日誌登載，當待命佇列管理程式實例啟動時，它會立即嘗試使用 *SYNC 日誌登載。

1. 遠端異動日誌在交易上與作用中佇列管理程式的本端異動日誌一致。 如果佇列管理程式切換至待命實例，則可以立即回復。 待命實例通常會回復，而不會遺失或複製任何訊息。 只有在自前次檢查點以來遠端日誌登載失敗，且無法重新啟動先前作用中的佇列管理程式時，才會遺失或複製訊息。
2. 如果佇列管理程式失效接手至待命實例，則可能無法立即啟動。 使用 *SYNC 日誌登載來啟動待命佇列管理程式實例。 失效接手的原因可能阻止遠端日誌登載至管理待命實例的伺服器。 在處理任何持續訊息之前，佇列管理程式會等待直到偵測到問題為止。 錯誤會記載至日誌登載訊息佇列及 QSYSOPR。 佇列管理程式會偵測此訊息，將錯誤寫入佇列管理程式錯誤日誌，並取消啟動佇列管理程式日誌登載的遠端抄寫。 作用中佇列管理程式實例會回復，而不會遠端登載至此日誌登載。 當遠端伺服器再次可用時，您必須手動重新啟動同步遠端異動日誌抄寫。 然後會重新同步化日誌。
3. 抄寫遠端異動日誌的目標伺服器必須一律可用，才能維護遠端異動日誌。 遠端日誌通常會抄寫至管理待命佇列管理程式的相同伺服器。 伺服器可能變成無法使用。 錯誤會記載至日誌登載訊息佇列及 QSYSOPR。 佇列管理程式會偵測此訊息，將錯誤寫入佇列管理程式錯誤日誌，並取消啟動佇列管理程式日誌登載的遠端抄寫。 作用中佇列管理程式實例會回復，而不會遠端登載至此日誌登載。 當遠端伺服器再次可用時，您必須手動重新啟動同步遠端異動日誌抄寫。 然後會重新同步化日誌。
4. 遠端日誌登載比本端日誌登載更慢，而且如果伺服器之間相隔很遠，則會更慢。 佇列管理程式必須等待遠端日誌登載，這會降低佇列管理程式效能。

一對伺服器之間的 *SYNC / *SYNC 配置在失效接手之後回復待命實例時具有延遲的缺點。 *SYNC / *ASYNC 配置沒有這個問題。

*SYNC / *SYNC 會保證在切換或失效接手之後，只要遠端異動日誌可用，就不會遺失任何訊息。 如果您想要減少失效接手或切換之後訊息遺失的風險，您有兩個選擇。 如果遠端異動日誌變成非作用中，請停止作用中實例，或在多部伺服器上建立遠端異動日誌。

***SYNC / *ASYNC**

作用中佇列管理程式實例使用 *SYNC 日誌登載，當待命佇列管理程式實例啟動時，它會使用 *ASYNC 日誌登載。 在管理新待命實例的伺服器變成可用之後不久，系統操作員必須將作用中實例上的遠端異動日誌切換至 *SYNC。 當操作員將遠端日誌登載從 *ASYNC 切換至 *SYNC 時，如果遠端日誌登載的狀態為 *ASYNCPEND，則作用中實例會暫停。 作用中佇列管理程式實例會等待，直到將剩餘的異動日誌項目傳送至遠端異動日誌為止。 當遠端異動日誌已與本端異動日誌同步時，新的待命資料庫會與新的作用中實例重新交易一致。 從多重實例佇列管理程式管理的角度來看，在 *SYNC / *ASYNC 配置中， IBM i 系統

操作員還有一項額外作業。 除了重新啟動失敗的佇列管理程式實例之外，操作員還必須將遠端日誌登載切換至 *SYNC 。

1. 遠端異動日誌在交易上與作用中佇列管理程式的本端異動日誌一致。 如果作用中佇列管理程式實例已切換或失效接手至待命實例，則待命實例可以立即回復。 待命實例通常會回復，而不會遺失或複製任何訊息。 只有在自前次檢查點以來遠端日誌登載失敗，且無法重新啟動先前作用中的佇列管理程式時，才會遺失或複製訊息。
2. 在管理作用中實例的系統重新變成可用之後，系統操作員必須立即將遠端異動日誌從 *ASYNC 切換至 *SYNC 。 在將遠端異動日誌切換至 *SYNC 之前，操作員可能等待遠端異動日誌擷取並回應。 或者，操作員可以立即將遠端實例切換至 *SYNC ，並強制作用中實例等待直到待命實例日誌登載已擷取為止。 當遠端日誌登載設為 *SYNC 時，待命實例通常與作用中實例在交易上一致。 只有在自前次檢查點以來遠端日誌登載失敗，且無法重新啟動先前作用中的佇列管理程式時，才會遺失或複製訊息。
3. 從切換或失效接手還原配置時，管理遠端異動日誌的伺服器必須隨時可用。

當您想要在失效接手之後快速回復待命佇列管理程式時，請選擇 *SYNC / *ASYNC 。 您必須手動將新作用中實例上的遠端異動日誌設定還原為 *SYNC 。 *SYNC / *ASYNC 配置符合管理一對多重實例佇列管理程式的一般型樣。 在一個實例失敗之後，有一段時間會重新啟動待命實例，在此期間作用中實例無法失效接手。

***ASYNC / *ASYNC**
同時管理作用中及待命佇列管理程式的伺服器均配置為使用 *ASYNC 遠端日誌登載。

1. 發生切換或失效接手時，佇列管理程式會繼續使用新伺服器上的日誌登載。 發生切換或失效接手時，日誌登載可能未同步。 因此，訊息可能遺失或重複。
2. 即使管理待命佇列管理程式的伺服器無法使用，作用中實例仍會執行。 當待命伺服器可用時，會以非同步方式抄寫本端日誌登載。
3. 遠端日誌登載不會影響本端佇列管理程式的效能。

如果效能是您的主體需求，且您準備在失效接手或切換之後釋放或複製部分訊息，請選擇 *ASYNC / *ASYNC 。

***ASYNC / *SYNC**
沒有理由使用此選項組合。

從遠端異動日誌啟動佇列管理程式

日誌登載會同步或非同步抄寫。 遠端異動日誌可能不在作用中，或它可能正在與本端異動日誌擷取並更新。 遠端異動日誌可能正在迎頭趕上，即使它是同步抄寫的，因為它最近可能已啟動。 佇列管理程式在啟動期間套用至其使用之遠端異動日誌狀態的規則如下。

1. 如果待命資料庫必須從待命資料庫上的遠端異動日誌重播，且日誌狀態為 *FAILED 或 *INACTPEND，則待命資料庫啟動會失敗。
2. 當開始啟動待命資料庫時，待命資料庫上的遠端異動日誌狀態必須是 *ACTIVE 或 *INACTIVE。 如果狀態為 *INACTIVE，則啟動可能失敗，如果尚未抄寫所有異動日誌資料。

   如果網路檔案系統上佇列管理程式資料的檢查點記錄比遠端異動日誌中的檢查點記錄還新，則會發生失敗。 只要遠端異動日誌在檢查點之間的預設 30 分鐘間隔上限內順利啟動，就不可能發生失敗。 如果待命佇列管理程式從檔案系統讀取最近的檢查點記錄，則不會啟動。

   您可以選擇: 等到可以還原作用中伺服器上的本端日誌登載，或冷啟動待命佇列管理程式。 如果您選擇冷啟動，則佇列管理程式會在沒有日誌登載資料的情況下啟動，並根據檔案系統中佇列管理程式資料的一致性及完整性。

   註: 如果您冷啟動佇列管理程式，則在最後一個檢查點之後，會有遺失或複製訊息的風險。 訊息交易已寫入日誌登載，但部分交易可能尚未寫入檔案系統中的佇列管理程式資料。 當您冷啟動佇列管理程式時，會啟動全新日誌登載，且會遺失未寫入檔案系統中佇列管理程式資料的交易。
3. 待命佇列管理程式啟動會等待待命資料庫上的遠端異動日誌狀態從 *ASYNCPEND 或 *SYNCPEND 變更為 *ASYNC 或 *SYNC。 訊息會定期寫入執行控制器的工作日誌。

   註: 在此情況下，啟動會在所啟動待命佇列管理程式的本端遠端異動日誌上等待。 在沒有遠端異動日誌的情況下繼續之前，佇列管理程式也會等待一段時間。 當它嘗試同步寫入遠端異動日誌 (或異動日誌) 且異動日誌無法使用時，它會等待。

4. 如果異動日誌狀態變更為 *FAILED 或 *INACTPEND，則啟動會停止。

要在啟動中使用的本端及遠端異動日誌的名稱及狀態會寫入佇列管理程式錯誤日誌。

使用日誌登載鏡映及 *NetServer* 建立多重實例佇列管理程式

建立要在兩部 IBM i 伺服器上執行的多重實例佇列管理程式。 佇列管理程式資料會使用 NetServer 儲存在第三部 IBM i 伺服器上。 使用遠端日誌登載在兩部伺服器之間鏡映佇列管理程式異動日誌。 **ADDMQMJRN** 指令可用來簡化建立遠端異動日誌。

開始之前

1. 此作業需要三部 IBM i 伺服器。 在其中兩個上安裝 IBM MQ : 範例中的 ALPHA 和 BETA。 IBM MQ 必須至少為 7.0.1.1 版。
2. 第三部伺服器是 IBM i 伺服器，由 NetServer 連接至 ALPHA 及 BETA。 它用來共用佇列管理程式資料。 它不需要有 IBM MQ 安裝架構。 在伺服器上作為暫時步驟安裝 IBM MQ 有助於設定佇列管理程式目錄及許可權。
3. 請確定 QMQM 使用者設定檔在這三部伺服器上都有相同的密碼。
4. 安裝 IBM i NetServer; 請參閱 i5/OS NetServer。

關於這項作業

請執行下列步驟來建立 第 203 頁的圖 36 中所示的配置。 佇列管理程式資料是使用 IBM i NetServer 來連接。

- 在要儲存佇列管理程式資料的 GAMMA 上建立從 ALPHA 及 BETA 到目錄共用的連線。 此作業也會設定必要的許可權、使用者設定檔及密碼。
- 將「關聯式資料庫項目 (RDBE)」新增至即將執行佇列管理程式實例的 IBM i 系統。 RDBE 項目是用來連接至用於遠端日誌登載的 IBM i 系統。
- 在 IBM i 伺服器 Alpha 上建立佇列管理程式 QM1 。
- 在其他 IBM i 伺服器 BETA 上新增 QM1 的佇列管理程式控制資訊。
- 在兩個佇列管理程式實例的兩個 IBM i 伺服器上建立遠端日誌登載。 每一個佇列管理程式都會寫入本端日誌登載。 本端異動日誌會複製到遠端異動日誌。 **ADDMQMJRN** 指令可簡化新增日誌登載及連線。
- 啟動佇列管理程式，允許待命實例。

程序

1. 執行作業 第 190 頁的『使用 NetServer 建立佇列管理程式資料的網路共用』。

   因此， ALPHA 和 BETA 具有一個共用 /QNTC/GAMMA/WMQ，指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 。 使用者設定檔 QMQM 和 QMQMADM 具有必要的許可權，且 QMQM 在所有三個系統上都具有相符的密碼。
2. 將「關聯式資料庫項目 (RDBE)」新增至將要管理佇列管理程式實例的 IBM i 系統。

   a) 在 ALPHA 上建立 BETA 的連線。

   ```
   ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
   ```

   b) 在測試版上，建立與 ALPHA 的連線。

   ```
   ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
   ```
3. 在 ALPHA 上建立佇列管理程式 QM1 ， 並將佇列管理程式資料儲存在 GAMMA 上。

```
CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
MQMDIRP(' /QNTC/GAMMA/WMQ ')
```

路徑 /QNTC/GAMMA/WMQ 使用 NetServer 在 /QIBM/UserData/mqm/qmgrs 中建立佇列管理程式資料。

4. 在 ALPHA 上執行 **ADDMQMJRN** 。 此指令會在 BETA for QM1 上新增遠端異動日誌。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(BETA)
```

當 QM1 的作用中實例位於 ALPHA 時， QM1 會在 ALPHA 的本端異動日誌中建立異動日誌項目。 Alpha 上的本端異動日誌會複製到 BETA 上的遠端異動日誌。

5. 使用指令 **DSPF** 來檢查 **CRTMQM** for QM1 在 ALPHA 上建立的 IBM MQ 配置資料。

下一步需要此資訊。

在此範例中，下列配置建立在 QM1 的 Alpha 上的 /QIBM/UserData/mqm/mqs.ini 中:

```
Name=QM1
Prefix=/QIBM/UserData/mqm
Library=QMQM1
Directory=QM1
DataPath= /QNTC/GAMMA/WMQ /QM1
```

6. 使用 **ADDMQMINF** 指令在測試版上建立 QM1 的佇列管理程式實例。 在 BETA 上執行下列指令，以修改 BETA 上 /QIBM/UserData/mqm/mqs.ini 中的佇列管理程式控制資訊。

```
ADDMQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
```

提示: 複製並貼上配置資訊。 Alpha 和 BETA 上的佇列管理程式段落相同。

7. 在測試版上執行 **ADDMQMJRN** 。 此指令會在 BETA 上新增本端異動日誌，並在 ALPHA for QM1 上新增遠端異動日誌。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
```

當 QM1 的作用中實例在 BETA 上時， QM1 會在其 BETA 上的本端異動日誌中建立異動日誌項目。 BETA 上的本端異動日誌會複製到 ALPHA 上的遠端異動日誌。

註: 作為替代方案，您可能想要使用非同步日誌登載來設定從 BETA 到 ALPHA 的遠端日誌登載。

使用此指令來設定從 BETA 到 ALPHA 的非同步日誌登載，而不是步驟 第 202 頁的『7』中的指令。

```
ADDMQMJRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNC)
```

如果 ALPHA 上的伺服器或日誌登載是失敗的來源，則 BETA 會在不等待新的異動日誌項目抄寫至 ALPHA 的情況下啟動。

當 ALPHA 再次連線時，使用 **CHGMQMJRN** 指令將抄寫模式切換至 *SYNC。

使用 第 197 頁的『ASP 的鏡映異動日誌配置』 中的資訊來決定是同步、非同步或兩者的混合鏡映日誌。預設值是同步抄寫，等待來自遠端異動日誌的回應 60 秒。

8. 請驗證 ALPHA 及 BETA 上的異動日誌已啟用，且遠端異動日誌抄寫的狀態為 *ACTIVE。

a) 在 ALPHA 上:

```
WRKMQMJRN MQMNAME(QM1)
```

b) 在測試版上:

```
WRKMQMJRN MQMNAME(QM1)
```

9. 在 ALPHA 和 BETA 上啟動佇列管理程式實例。
   a) 在 ALPHA 上啟動第一個實例，使它成為作用中實例。 啟用切換至待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

   b) 在 BETA 上啟動第二個實例，使它成為待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

結果

使用 **WRKMQM** 來檢查佇列管理程式狀態:

1. Alpha 上佇列管理程式實例的狀態應該是 *ACTIVE。
2. BETA 上佇列管理程式實例的狀態應該是 *STANDBY。

範例

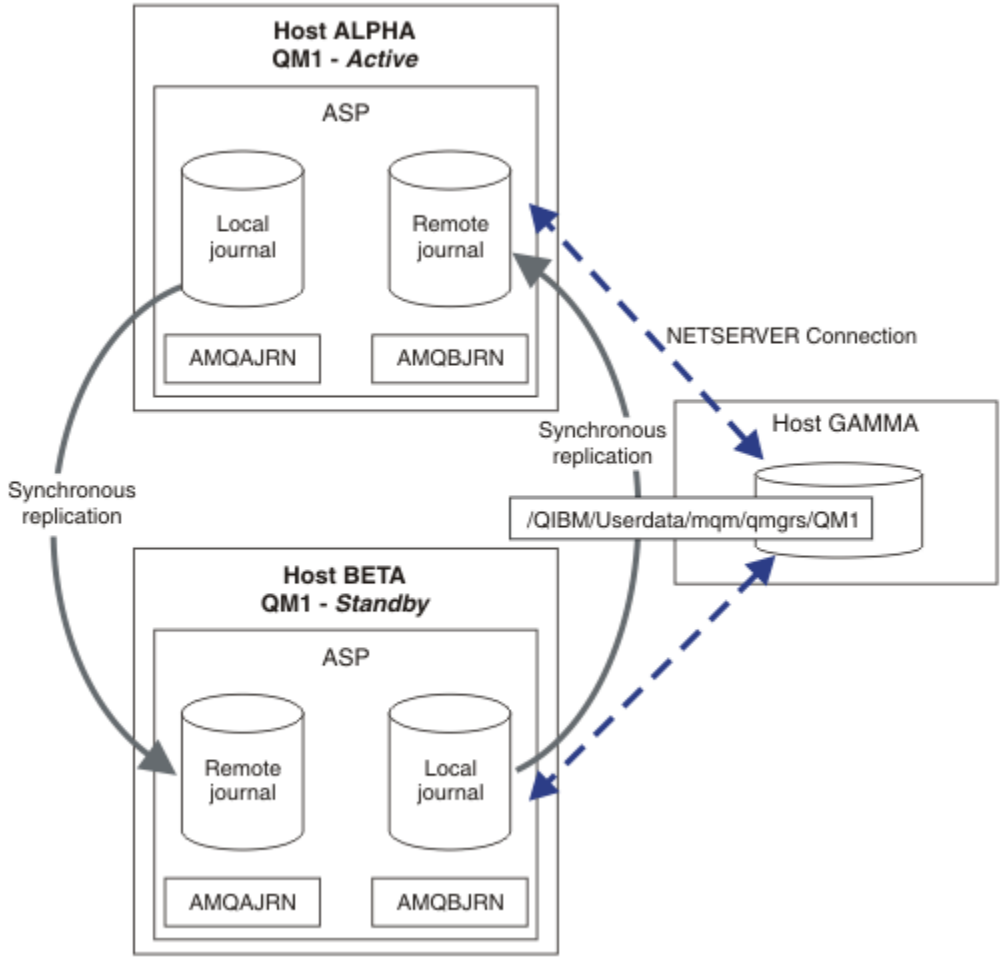


圖 *36:* 鏡映異動日誌配置

下一步

- 驗證作用中及待命實例自動切換。 您可以執行範例高可用性範例程式來測試切換; 請參閱 高可用性範例程式。 範例程式是 ' C' 用戶端。 您可以從 Windows 或 Unix 平台執行它們。
  1. 啟動高可用性範例程式。
  2. 在 ALPHA 上，結束要求切換的佇列管理程式:

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. 檢查 BETA 上的 QM1 實例是否處於作用中。
  4. 在 ALPHA 上重新啟動 QM1

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 查看替代高可用性配置:
  1. 使用 NetServer ，將佇列管理程式資料放置在 Windows 伺服器上。
  2. 不使用遠端日誌登載來鏡映佇列管理程式異動日誌，而是將異動日誌儲存在獨立 ASP 上。 使用 IBM i 叢集作業，將獨立 ASP 從 ALPHA 傳送至 BETA。

使用 *NetServer* 及日誌登載鏡映將單一實例佇列管理程式轉換為多重實例佇列管理程式
將單一實例佇列管理程式轉換為多重實例佇列管理程式。 將佇列管理程式資料移至 NetServer 所連接的網路共用。 使用遠端日誌登載，將佇列管理程式日誌登載鏡映至第二部 IBM i 伺服器。

開始之前

1. 此作業需要三部 IBM i 伺服器。 範例中伺服器 ALPHA 上的現有 IBM MQ 安裝必須至少為 7.0.1 版.1。 ALPHA 正在執行範例中稱為 QM1 的佇列管理程式。
2. 在第二部 IBM i 伺服器上安裝 IBM MQ ，範例中為 BETA。
3. 第三部伺服器是 IBM i 伺服器，由 NetServer 連接至 ALPHA 及 BETA。 它用來共用佇列管理程式資料。 它不需要有 IBM MQ 安裝架構。 在伺服器上作為暫時步驟安裝 IBM MQ 有助於設定佇列管理程式目錄及許可權。
4. 請確定 QMQM 使用者設定檔在這三部伺服器上都有相同的密碼。
5. 安裝 IBM i NetServer; 請參閱 i5/OS NetServer。

關於這項作業

執行下列步驟，將單一實例佇列管理程式轉換為 第 207 頁的圖 37 中顯示的多重實例佇列管理程式。 單一實例佇列管理程式會在作業中刪除，然後重建，並將佇列管理程式資料儲存在 NetServer 所連接的網路共用上。 此程序比使用 **CPY** 指令將佇列管理程式目錄及檔案移至網路共用更為可靠。

- 在要儲存佇列管理程式資料的 GAMMA 上建立從 ALPHA 及 BETA 到目錄共用的連線。 此作業也會設定必要的許可權、使用者設定檔及密碼。
- 將「關聯式資料庫項目 (RDBE)」新增至即將執行佇列管理程式實例的 IBM i 系統。 RDBE 項目是用來連接至用於遠端日誌登載的 IBM i 系統。
- 儲存佇列管理程式日誌和定義，停止佇列管理程式，然後刪除它。
- 重建佇列管理程式，並在 GAMMA 上儲存網路共用上的佇列管理程式資料。
- 將佇列管理程式的第二個實例新增至另一部伺服器。
- 在兩個佇列管理程式實例的兩個 IBM i 伺服器上建立遠端日誌登載。 每一個佇列管理程式都會寫入本端日誌登載。 本端異動日誌會複製到遠端異動日誌。 **ADDMQMJRN** 指令可簡化新增日誌登載及連線。
- 啟動佇列管理程式，允許待命實例。

註:

在作業的步驟 第 205 頁的『4』 中，您刪除單一實例佇列管理程式 QM1。 刪除佇列管理程式會刪除佇列上的所有持續訊息。 因此，在轉換佇列管理程式之前，請先完成處理佇列管理程式所儲存的所有訊息。 如果無法處理所有訊息，請在步驟 第 205 頁的『4』之前備份佇列管理程式庫。 在步驟 第 205 頁的『5』之後還原佇列管理程式庫。

註:

在作業的步驟 第 205 頁的『5』 中，您重建 QM1。 雖然佇列管理程式具有相同的名稱，但它具有不同的佇列管理程式 ID。 佇列管理程式叢集作業會使用佇列管理程式 ID。 若要刪除並重建叢集中的佇列管理程式，您必須先從叢集中移除佇列管理程式; 請參閱 從叢集中移除佇列管理程式: 替代方法 或 從叢集中移除佇列管理程式。 當您重建佇列管理程式時，請將它新增至叢集。 雖然它的名稱與之前相同，但對叢集中其他佇列管理程式而言，它似乎是新的佇列管理程式。

程序

1. 執行作業 第 190 頁的『使用 NetServer 建立佇列管理程式資料的網路共用』。

   因此， ALPHA 和 BETA 具有一個共用 /QNTC/GAMMA/WMQ，指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 。 使用者設定檔 QMQM 和 QMQMADM 具有必要的許可權，且 QMQM 在所有三個系統上都具有相符的密碼。
2. 將「關聯式資料庫項目 (RDBE)」新增至將要管理佇列管理程式實例的 IBM i 系統。
   a) 在 ALPHA 上建立 BETA 的連線。

   ```
   ADDRDBDIRE RDB(BETA) RMTLOCNAME(BETA *IP) RMTAUTMTH(*USRIDPWD)
   ```

   b) 在測試版上，建立與 ALPHA 的連線。

   ```
   ADDRDBDIRE RDB(ALPHA) RMTLOCNAME(ALPHA *IP) RMTAUTMTH(*USRIDPWD)
   ```
3. 建立 Script 來重建佇列管理程式物件。

   ```
   QSAVEQMGR LCLQMGRNAM(QM1) FILENAME('*CURLIB/QMQSC(QM1)')
   OUTPUT(*REPLACE) MAKEAUTH(*YES) AUTHFN('*CURLIB/QMAUT(QM1)')
   ```
4. 停止佇列管理程式並刪除它。

   ```
   ENDMQM MQMNAME(QM1) OPTION(*IMMED) ENDCCTJOB(*YES) RCDMQMIMG(*YES) TIMEOUT(15)
   DLTMQM MQMNAME(QM1)
   ```
5. 在 ALPHA 上建立佇列管理程式 QM1 ， 並將佇列管理程式資料儲存在 GAMMA 上。

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   路徑 /QNTC/GAMMA/WMQ 使用 NetServer 在 /QIBM/UserData/mqm/qmgrs 中建立佇列管理程式資料。
6. 從儲存的定義重建 QM1 的佇列管理程式物件。

   ```
   STRMQMMQSC SRCMBR(QM1) SRCFILE(*CURLIB/QMQSC) MQMNAME(QM1)
   ```
7. 從儲存的資訊套用授權。
   a) 編譯已儲存的授權程式。

   ```
   CRTCLPGM PGM(*CURLIB/QM1) SRCFILE(*CURLIB/QMAUT)
   SRCMBR(QM1) REPLACE(*YES)
   ```

   b) 執行程式以套用授權。

```
CALL PGM(*CURLIB/QM1)
```

c) 重新整理 QM1 的安全資訊。

```
RFRMQMAUT MQMNAME(QM1)
```

8. 在 ALPHA 上執行 **ADDMQMJRN** 。 此指令會在 BETA for QM1 上新增遠端異動日誌。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(BETA)
```

當 QM1 的作用中實例位於 ALPHA 時， QM1 會在 ALPHA 的本端異動日誌中建立異動日誌項目。 Alpha 上的本端異動日誌會複製到 BETA 上的遠端異動日誌。

9. 使用指令 **DSPF** 來檢查 **CRTMQM** for QM1 在 ALPHA 上建立的 IBM MQ 配置資料。

下一步需要此資訊。

在此範例中，下列配置建立在 QM1 的 Alpha 上的 /QIBM/UserData/mqm/mqs.ini 中:

```
Name=QM1
Prefix=/QIBM/UserData/mqm
Library=QMQM1
Directory=QM1
DataPath= /QNTC/GAMMA/WMQ /QM1
```

10. 使用 **ADDMQMINF** 指令在測試版上建立 QM1 的佇列管理程式實例。 在 BETA 上執行下列指令，以修改 BETA 上 /QIBM/UserData/mqm/mqs.ini 中的佇列管理程式控制資訊。

```
ADDMQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
```

提示: 複製並貼上配置資訊。 Alpha 和 BETA 上的佇列管理程式段落相同。

11. 在測試版上執行 **ADDMQMJRN** 。 此指令會在 BETA 上新增本端異動日誌，並在 ALPHA for QM1 上新增遠端異動日誌。

```
ADDMQMJRN MQMNAME(QM1) RMTJRNRDB(ALPHA)
```

當 QM1 的作用中實例在 BETA 上時， QM1 會在其 BETA 上的本端異動日誌中建立異動日誌項目。 BETA 上的本端異動日誌會複製到 ALPHA 上的遠端異動日誌。

註: 作為替代方案，您可能想要使用非同步日誌登載來設定從 BETA 到 ALPHA 的遠端日誌登載。

使用此指令來設定從 BETA 到 ALPHA 的非同步日誌登載，而不是步驟 第 202 頁的『7』中的指令。

```
ADDMQMJRN MQMNAME (QM1) RMTJRNRDB (ALPHA) RMTJRNDLV (*ASYNC)
```

如果 ALPHA 上的伺服器或日誌登載是失敗的來源，則 BETA 會在不等待新的異動日誌項目抄寫至 ALPHA 的情況下啟動。

當 ALPHA 再次連線時，使用 **CHGMQMJRN** 指令將抄寫模式切換至 *SYNC。

使用 第 197 頁的『ASP 的鏡映異動日誌配置』 中的資訊來決定是同步、非同步或兩者的混合鏡映日誌。 預設值是同步抄寫，等待來自遠端異動日誌的回應 60 秒。

12. 請驗證 ALPHA 及 BETA 上的異動日誌已啟用，且遠端異動日誌抄寫的狀態為 *ACTIVE。

a) 在 ALPHA 上:

```
WRKMQMJRN MQMNAME(QM1)
```

b) 在測試版上:

```
WRKMQMJRN MQMNAME(QM1)
```

13. 在 ALPHA 和 BETA 上啟動佇列管理程式實例。
    a) 在 ALPHA 上啟動第一個實例，使它成為作用中實例。 啟用切換至待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

    b) 在 BETA 上啟動第二個實例，使它成為待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

結果

使用 **WRKMQM** 來檢查佇列管理程式狀態:

1. Alpha 上佇列管理程式實例的狀態應該是 *ACTIVE。
2. BETA 上佇列管理程式實例的狀態應該是 *STANDBY。

範例

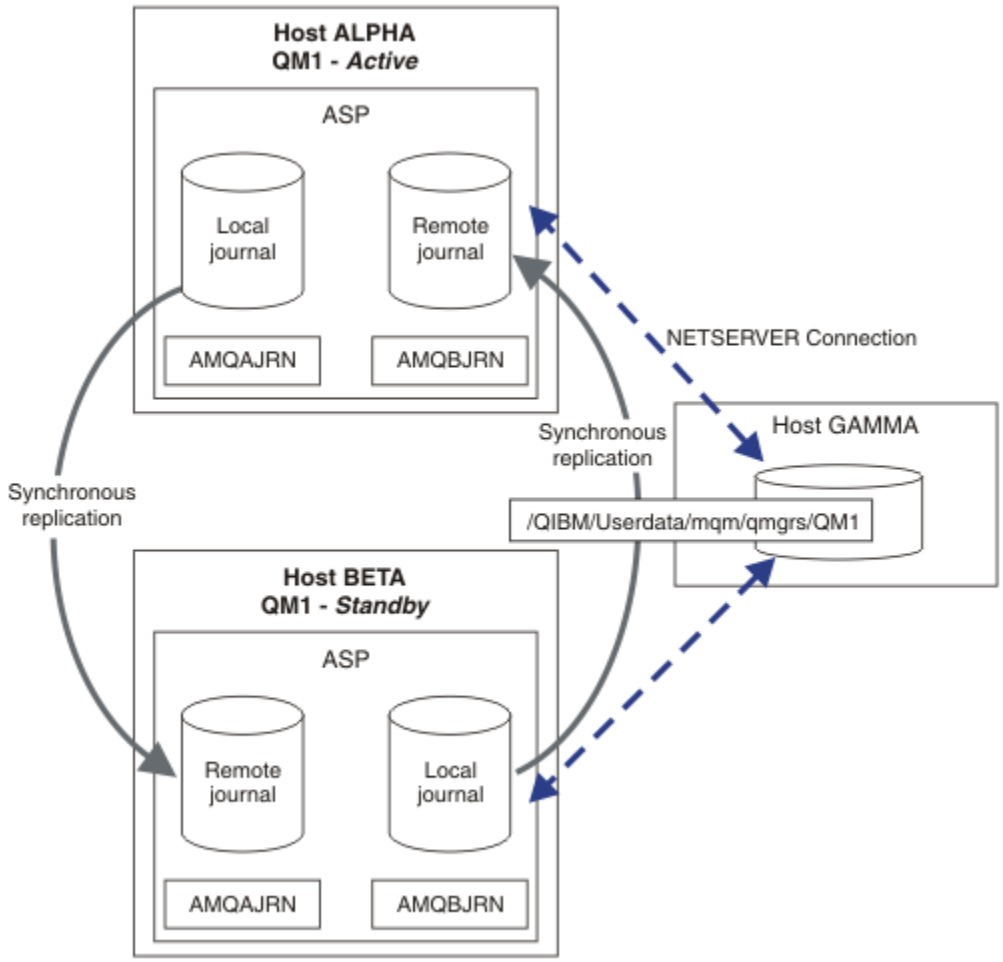


圖 *37:* 鏡映異動日誌配置

下一步

- 驗證作用中及待命實例自動切換。 您可以執行範例高可用性範例程式來測試切換; 請參閱 高可用性範例程式。 範例程式是 ' C' 用戶端。 您可以從 Windows 或 Unix 平台執行它們。
  1. 啟動高可用性範例程式。
  2. 在 ALPHA 上，結束要求切換的佇列管理程式:

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. 檢查 BETA 上的 QM1 實例是否處於作用中。
  4. 在 ALPHA 上重新啟動 QM1

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 查看替代高可用性配置:
  1. 使用 NetServer ，將佇列管理程式資料放置在 Windows 伺服器上。
  2. 不使用遠端日誌登載來鏡映佇列管理程式異動日誌，而是將異動日誌儲存在獨立 ASP 上。 使用 IBM i 叢集作業，將獨立 ASP 從 ALPHA 傳送至 BETA。

交換式獨立 *ASP* 異動日誌配置

您不需要抄寫獨立 ASP 日誌登載，即可建立多重實例佇列管理程式配置。 您確實需要自動化方法，將獨立 ASP 從作用中佇列管理程式傳送至待命佇列管理程式。 有替代高可用性解決方案可以使用獨立 ASP ，但並非所有解決方案都需要使用多重實例佇列管理程式。

使用獨立 ASP 時，您不需要鏡映佇列管理程式異動日誌。 如果您已安裝叢集管理，且管理佇列管理程式實例的伺服器位於相同的叢集資源群組中，則在執行作用中實例的主機失敗時，佇列管理程式日誌登載可以在作用中伺服器的短距離內自動傳送至另一部伺服器。 您也可以在計劃的交換器中手動傳送異動日誌，也可以撰寫指令程序以程式化方式傳送獨立 ASP。

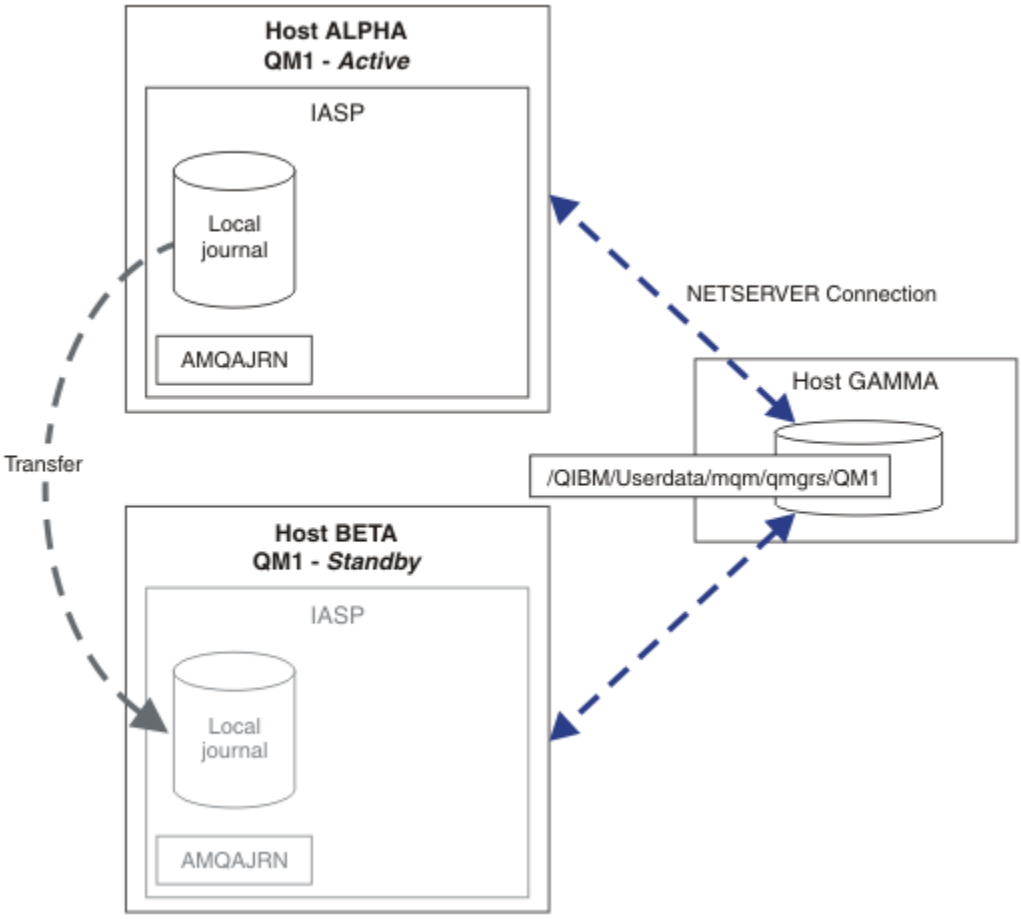


圖 *38:* 使用獨立 *ASP* 傳送佇列管理程式日誌

對於多重實例佇列管理程式作業，佇列管理程式資料必須儲存在共用檔案系統上。檔案系統可以在各種不同的平台上管理。 您無法在 ASP 或獨立 ASP 上儲存多重實例佇列管理程式資料。

共用檔案系統會在配置中執行兩個角色: 在佇列管理程式的所有實例之間共用相同的佇列管理程式資料。 檔案系統必須具有健全的鎖定通訊協定，以確保在啟動佇列管理程式資料之後，只有一個佇列管理程式實例具有佇列管理程式資料的存取權。 如果佇列管理程式失敗，或與檔案伺服器的通訊中斷，則檔案系統必須解除鎖定不再與檔案系統通訊的作用中實例所保留的佇列管理程式資料。 然後，待命佇列管理程式實例可以取得佇列管理程式資料的讀寫存取權。 檔案系統通訊協定必須符合一組規則，才能正確地使用多重實例佇列管理程式; 請參閱 第 190 頁的『高可用性解決方案的元件』。

鎖定機制會序列化啟動佇列管理程式指令，並控制佇列管理程式的作用中實例。 一旦佇列管理程式變成作用中，它會從您或 HA 叢集已傳送至待命伺服器的本端日誌登載重新建置其佇列。 等待重新連線至相同佇列管理程式的可重新連線用戶端會重新連線，且會取消任何進行中的交易。 已啟動配置為作為佇列管理程式服務啟動的應用程式。

您需要透過配置叢集資源管理程式或手動傳送獨立 ASP，確保將獨立 ASP 上失敗作用中佇列管理程式實例的本端日誌傳送至管理新啟動待命佇列管理程式實例的伺服器。 如果您決定使用獨立 ASP 進行備份及災難回復，並對多重實例佇列管理程式配置使用遠端異動日誌鏡映，則使用獨立 ASP 並不會排除配置遠端異動日誌及鏡映。

如果您已選擇使用獨立 ASP，則可以考慮使用替代的高可用性配置。 第 213 頁的『獨立 ASP 及高可用性』中說明這些解決方案的背景。

1. 不使用多重實例佇列管理程式，而是完全在獨立 ASP 上安裝並配置單一實例佇列管理程式，並使用 IBM i 高可用性服務讓佇列管理程式失效接手。 您可能需要使用佇列管理程式監視器來擴增解決方案，以偵測佇列管理程式是否獨立於伺服器而失敗。 這是 *Supportpac MC41:* 配置 *IBM MQ for iSeries for High Availability*。

2. 使用獨立 ASP 及跨站台鏡映 (XSM) 來鏡映獨立 ASP，而不是在本端匯流排上切換獨立 ASP。 這會將獨立 ASP 解決方案的地理範圍延伸到容許長時間寫入日誌記錄所花費的時間。

使用獨立 *ASP* 及 *NetServer* 建立多重實例佇列管理程式
建立要在兩部 IBM i 伺服器上執行的多重實例佇列管理程式。 佇列管理程式資料會使用 NetServer 儲存在 IBM i 伺服器中。 佇列管理程式異動日誌儲存在獨立 ASP 上。 使用 IBM i 叢集作業或手動程序，將包含佇列管理程式日誌登載的獨立 ASP 傳送至其他 IBM i 伺服器。

開始之前

1. 此作業需要三部 IBM i 伺服器。 在其中兩個上安裝 IBM MQ : 範例中的 ALPHA 和 BETA。 IBM MQ 必須至少為 7.0.1.1 版。
2. 第三部伺服器是 IBM i 伺服器，由 NetServer 連接至 ALPHA 及 BETA。 它用來共用佇列管理程式資料。 它不需要有 IBM MQ 安裝架構。 在伺服器上作為暫時步驟安裝 IBM MQ 有助於設定佇列管理程式目錄及許可權。
3. 請確定 QMQM 使用者設定檔在這三部伺服器上都有相同的密碼。
4. 安裝 IBM i NetServer; 請參閱 i5/OS NetServer。
5. 建立程序以將獨立 ASP 從失敗的佇列管理程式傳送至接管的待命資料庫。 您可能會發現 *SupportPac MC41:* 配置 *IBM MQ for iSeries for High Availability* 有助於設計您的獨立 ASP 傳送程序。

關於這項作業

請執行下列步驟來建立 第 212 頁的圖 39 中所示的配置。 佇列管理程式資料是使用 IBM i NetServer 來連接。

- 在要儲存佇列管理程式資料的 GAMMA 上建立從 ALPHA 及 BETA 到目錄共用的連線。 此作業也會設定必要的許可權、使用者設定檔及密碼。
- 在 IBM i 伺服器 Alpha 上建立佇列管理程式 QM1 。
- 在其他 IBM i 伺服器 BETA 上新增 QM1 的佇列管理程式控制資訊。
- 啟動佇列管理程式，允許待命實例。

程序

1. 執行作業 第 190 頁的『使用 NetServer 建立佇列管理程式資料的網路共用』。

   因此， ALPHA 和 BETA 具有一個共用 /QNTC/GAMMA/WMQ，指向 GAMMA 上的 /QIBM/UserData/mqm/qmgrs 。 使用者設定檔 QMQM 和 QMQMADM 具有必要的許可權，且 QMQM 在所有三個系統上都具有相符的密碼。
2. 在 ALPHA 上建立佇列管理程式 QM1 ， 並將佇列管理程式資料儲存在 GAMMA 上。

   ```
   CRTMQM MQMNAME(QM1) UDLMSGQ(SYSTEM.DEAD.LETTER.QUEUE)
   MQMDIRP(' /QNTC/GAMMA/WMQ ')
   ```

   路徑 /QNTC/GAMMA/WMQ 使用 NetServer 在 /QIBM/UserData/mqm/qmgrs 中建立佇列管理程式資料。
3. 使用指令 **DSPF** 來檢查 **CRTMQM** for QM1 在 ALPHA 上建立的 IBM MQ 配置資料。

   下一步需要此資訊。

   在此範例中，下列配置建立在 QM1 的 Alpha 上的 /QIBM/UserData/mqm/mqs.ini 中:

   ```
   Name=QM1
   Prefix=/QIBM/UserData/mqm
   Library=QMQM1
   Directory=QM1
   DataPath= /QNTC/GAMMA/WMQ /QM1
   ```

4. 使用 **ADDMQMINF** 指令在測試版上建立 QM1 的佇列管理程式實例。 在 BETA 上執行下列指令，以修改 BETA 上 /QIBM/UserData/mqm/mqs.ini 中的佇列管理程式控制資訊。

```
ADDMQMINF MQMNAME(QM1)
PREFIX('/QIBM/UserData/mqm')
MQMDIR(QM1)
MQMLIB(QMQM1)
DATAPATH(' /QNTC/GAMMA/WMQ /QM1 ')
```

   提示: 複製並貼上配置資訊。 Alpha 和 BETA 上的佇列管理程式段落相同。
5. 在 ALPHA 和 BETA 上啟動佇列管理程式實例。
   a) 在 ALPHA 上啟動第一個實例，使它成為作用中實例。 啟用切換至待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

   b) 在 BETA 上啟動第二個實例，使它成為待命實例。

```
STRMQM MQMNAME(QM1) STANDBY(*YES)
```

結果

使用 **WRKMQM** 來檢查佇列管理程式狀態:

1. Alpha 上佇列管理程式實例的狀態應該是 *ACTIVE。
2. BETA 上佇列管理程式實例的狀態應該是 *STANDBY。

範例

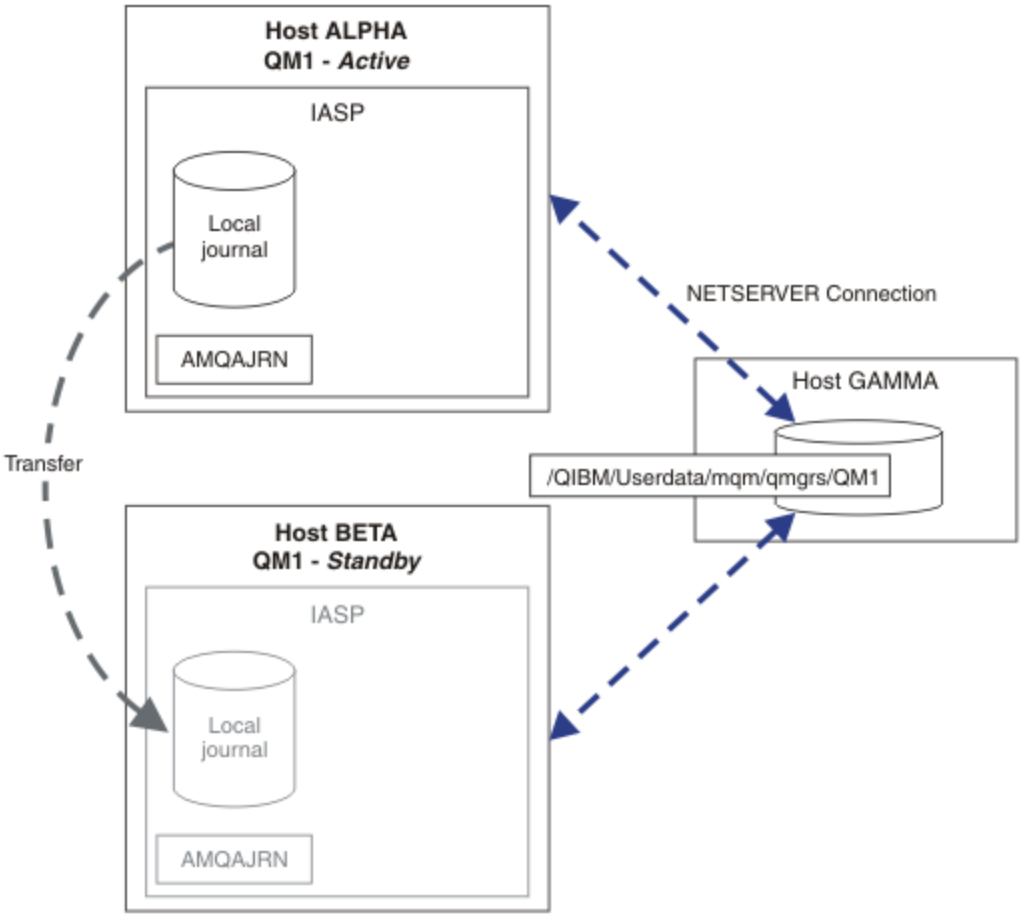


圖 *39:* 使用獨立 *ASP* 傳送佇列管理程式日誌

下一步

- 驗證作用中及待命實例自動切換。 您可以執行範例高可用性範例程式來測試切換; 請參閱 高可用性範例程式。 範例程式是 ' C' 用戶端。 您可以從 Windows 或 Unix 平台執行它們。
  1. 啟動高可用性範例程式。
  2. 在 ALPHA 上，結束要求切換的佇列管理程式:

     ```
     ENDMQM MQMNAME(QM1) OPTION(*IMMED) ALSWITCH(*YES)
     ```

  3. 檢查 BETA 上的 QM1 實例是否處於作用中。
  4. 在 ALPHA 上重新啟動 QM1

     ```
     STRMQM MQMNAME(QM1) STANDBY(*YES)
     ```

- 查看替代高可用性配置:
  1. 使用 NetServer ， 將佇列管理程式資料放置在 IBM i 伺服器上。
  2. 使用遠端日誌登載將日誌登載鏡映至待命伺服器， 而不是使用獨立 ASP 將佇列管理程式日誌登載傳送至待命伺服器。

獨立 *ASP* 及高可用性

獨立 ASP 可讓應用程式及資料在伺服器之間移動。 獨立 ASP 的彈性表示它們是部分 IBM i 高可用性解決方案的基礎。 在考量對佇列管理程式異動日誌使用 ASP 或獨立 ASP 時，您應該考量根據獨立 ASP 的其他高可用性配置。

輔助儲存區 (ASP) 是 IBM i 架構的建置區塊。 硬碟機會分組在一起，以形成單一 ASP。 透過將物件放置在不同的 ASP 中，您可以保護一個 ASP 中的資料不受另一個 ASP 中磁碟故障的影響。

每個 IBM i 伺服器至少有一個 基本 ASP，稱為系統 ASP。 它指定為 ASP1，有時稱為 *SYSBAS。 您最多可以配置 31 個其他基本 使用者 ASP，因為它們共用相同的名稱空間，無法從應用程式的觀點來區分系統 ASP。 透過使用多個基本 ASP 來將應用程式配送至多個磁碟，您可以增進效能並減少回復時間。 使用多個基本 ASP 也可以針對磁碟故障提供某種程度的隔離，但它不會改善整體可靠性。

獨立 ASP 是特殊類型的 ASP。 它們通常稱為獨立磁碟儲存區。 獨立磁碟儲存區是 IBM i 高可用性的主要元件。 您可以儲存在獨立磁碟儲存裝置上視為與它們所連接的現行系統無關的資料及應用程式。 您可以配置可切換或不可切換的獨立 ASP。 從可用性角度來看，您通常只關心可切換的獨立 ASP，它們可以自動從伺服器傳送至伺服器。 因此，您可以將獨立 ASP 上的應用程式及資料從伺服器移至伺服器。

與基本使用者 ASP 不同，獨立 ASP 不會與系統 ASP 共用相同的名稱空間。 使用使用者 ASP 的應用程式需要變更才能使用獨立 ASP。 您需要驗證您的軟體以及您使用的協力廠商軟體在獨立 ASP 環境中運作。

當獨立 ASP 連接至不同的伺服器時，獨立 ASP 的名稱空間必須與系統 ASP 的名稱空間結合。 此處理程序稱為 轉接 獨立 ASP。 您可以在不執行伺服器 IPL 的情況下轉接獨立 ASP。 需要叢集支援，才能自動將獨立 ASP 從一部伺服器傳送至另一部伺服器。

使用獨立 ASP 來建置可靠的解決方案

日誌登載至獨立 ASP，而不是日誌登載至 ASP 並使用日誌登載抄寫，提供替代方法來為待命佇列管理程式提供失敗佇列管理程式實例的本端日誌副本。 若要自動將獨立 ASP 傳送至另一部伺服器，您必須已安裝並配置叢集作業支援。 根據叢集支援及低階磁碟鏡映，有許多獨立 ASP 的高可用性解決方案，您可以使用多重實例佇列管理程式來結合或替代。

下列清單說明根據獨立 ASP 建置可靠解決方案所需的元件。

日誌登載
佇列管理程式及其他應用程式會使用本端日誌登載來安全地將持續資料寫入磁碟，以防止因伺服器故障而遺失記憶體中的資料。 這有時稱為時間點一致性。 它不保證在一段時間內發生的多個更新項目的一致性。

確定控制
透過使用廣域交易，您可以協調訊息及資料庫的更新，以便寫入日誌登載的資料一致。 它使用兩段式確定通訊協定來提供一段時間內的一致性。

交換式磁碟

交換式磁碟由 HA 叢集中的裝置叢集資源群組 (CRG) 管理。 在非計劃性中斷的情況下， CRG 會自動將獨立 ASP 切換至新的伺服器。 CRG 在地理上受限於本端 IO 匯流排的範圍。

透過在切換式獨立 ASP 上配置本端日誌，您可以將日誌傳送至不同的伺服器，並回復處理訊息。 除非獨立 ASP 失敗，否則不會遺失沒有同步點控制或以同步點控制確定的持續訊息變更。

如果您在切換式獨立 ASP 上同時使用日誌登載及確定控制，則可以將資料庫異動日誌及佇列管理程式異動日誌傳送至不同的伺服器，並回復處理異動，而不會失去一致性或已確定異動。

跨站台鏡映 (XSM)
XSM 會透過 TCP/IP 網路將主要獨立 ASP 鏡映至地理上遠端次要獨立 ASP，並在失敗時自動傳送控制。 您可以選擇配置同步或非同步鏡映。 同步鏡映會降低佇列管理程式的效能，因為在正式作業系統上的寫入作業完成之前會鏡映資料，但它會保證次要獨立 ASP 是最新的。 而如果您使用非同步鏡映，則無法保證次要獨立 ASP 是最新的。 非同步鏡映會維護次要獨立 ASP 的一致性。

有三種 XSM 技術。

地理鏡映
地理鏡映是叢集作業的延伸，可讓您在廣域範圍內切換獨立 ASP。 它同時具有同步及非同步模式。 您只能在同步模式下保證高可用性，但分隔獨立 ASP 可能會對效能造成太大影響。 您可以結合地理鏡映與交換式磁碟，以提供本端高可用性及遠端災難回復。

Metro Mirror
Metro Mirror 是一種裝置層次服務，提供比本端匯流排更長距離的快速本端同步鏡映。 您可以將它與多重實例佇列管理程式結合，以提供佇列管理程式的高可用性，並透過具有兩個獨立 ASP 副本，提供佇列管理程式日誌的高可用性。

Global Mirror
Global Mirror 是提供非同步鏡映的裝置層次服務，適合在較長的距離上進行備份及災難回復，但並非高可用性的正常選擇，因為它只會維護時間點的一致性，而不是貨幣。

你應該考慮的關鍵決策點是，

ASP 或獨立 ASP?
您不需要執行 IBM i HA 叢集，即可使用多重實例佇列管理程式。 如果您已使用獨立 ASP，或您具有其他需要獨立 ASP 之應用程式的可用性需求，則可以選擇獨立 ASP。 它可能值得結合獨立 ASP 與多重實例佇列管理程式，以取代佇列管理程式監視，作為偵測佇列管理程式失敗的方法。

可用性?
回復時間目標 (RTO) 是什麼? 如果您需要出現近乎不中斷的行為，則哪個解決方案具有最快的回復時間?

日誌登載可用性?
如何將日誌刪除為單一失敗點。 您可以採用硬體解決方案，使用 RAID 1 裝置或更高版本，或者您可以使用抄本日誌或磁碟鏡映來結合或使用軟體解決方案。

距離?
作用中及待命佇列管理程式實例之間的距離。 您的使用者是否可以容忍在大於大約 250 公尺的距離上同步抄寫的效能降低?

技能?
需要執行一些工作，以將維護及定期執行解決方案所涉及的管理作業自動化。 根據 ASP 及獨立 ASP 的解決方案，執行自動化所需的技能有所不同。

刪除多重實例佇列管理程式

在刪除多重實例佇列管理程式之前，請停止遠端日誌登載，並移除佇列管理程式實例。

開始之前

1. 在此範例中，在伺服器 ALPHA 和 BETA 上定義兩個 QM1 佇列管理程式實例。 A 是作用中實例， BETA 是待命實例。 與佇列管理程式 QM1 相關聯的佇列管理程式資料，會使用 NetServer 儲存在 IBM i 伺服器 GAMMA 上。 請參閱 第 201 頁的『使用日誌登載鏡映及 NetServer 建立多重實例佇列管理程式』。
2. 必須連接 ALPHA 及 BETA， IBM MQ 才能刪除任何已定義的遠端異動日誌。
3. 驗證可以使用系統指令 **EDTF** 或 **WRKLNK** 來存取 /QNTC 目錄及伺服器目錄檔案共用

關於這項作業

在使用 **DLTMQM** 指令從伺服器刪除多重實例佇列管理程式之前，請使用 **RMVMQMINF** 指令移除其他伺服器上的任何佇列管理程式實例。

當您使用 **RMVMQMINF** 指令移除佇列管理程式實例時，會刪除字首為 AMQ 且與實例相關聯的本端及遠端異動日誌。 也會刪除伺服器本端佇列管理程式實例的相關配置資訊。

請勿在保留其餘佇列管理程式實例的伺服器上執行 **RMVMQMINF** 指令。 這樣做會導致 **DLTMQM** 無法正確運作。

使用 **DLTMQM** 指令刪除佇列管理程式。 佇列管理程式資料會從網路共用中移除。 會刪除字首為 AMQ 且與實例相關聯的本端及遠端異動日誌。 **DLTMQM** 也會刪除伺服器本端佇列管理程式實例的相關配置資訊。

在範例中，只有兩個佇列管理程式實例。 IBM MQ 支援具有一個作用中佇列管理程式實例及一個待命實例的執行中多重實例配置。 如果您已建立要在執行中配置中使用的其他佇列管理程式實例，請先使用 **RMVMQMINF** 指令移除它們，然後再刪除其餘實例。

程序

1. 在每一部伺服器上執行 **CHGMQMJRN RMTJRNSTS** (*INACTIVE) 指令，以讓佇列管理程式實例之間的遠端日誌登載成為非作用中。

   a) 在 ALPHA 上:

   ```
   CHGMQMJRN MQMNAME('QM1')
   RMTJRNRDB('BETA') RMTJRNSTS(*INACTIVE)
   ```

   b) 在測試版上:

   ```
   CHGMQMJRN MQMNAME('QM1')
   RMTJRNRDB('ALPHA') RMTJRNSTS(*INACTIVE)
   ```

2. 在 ALPHA (作用中佇列管理程式實例) 上執行 **ENDMQM** 指令，以停止 QM1 的兩個實例。

   ```
   ENDMQM MQMNAME(QM1) OPTION(*IMMED) INSTANCE(*ALL) ENDCCTJOB(*YES)
   ```

3. 在 ALPHA 上執行 **RMVMQMINF** 指令，以從 ALPHA 及 BETA 移除實例的佇列管理程式資源。

   ```
   RMVMQMINF MQMNAME(QM1)
   ```

   RMVMQMINF 會從 ALPHA 移除 QM1 的佇列管理程式配置資訊。 如果日誌名稱以 AMQ 作為字首，則會從 ALPHA 中刪除與 QM1 相關聯的本端日誌。 如果異動日誌名稱以 AMQ 為字首，且已建立遠端異動日誌，則它也會從 BETA 移除遠端異動日誌。

4. 在測試版上執行 **DLTMQM** 指令，以刪除 QM1。

   ```
   DLTMQM MQMNAME(QM1)
   ```

   DLTMQM 會從 GAMMA 上的網路共用中刪除佇列管理程式資料。 它會從測試版中移除 QM1 的佇列管理程式配置資訊。 如果日誌登載名稱以 AMQ 作為字首，則會從測試版中刪除與 QM1 相關聯的本端日誌登載。 如果異動日誌名稱以 AMQ 為字首，且已建立遠端異動日誌，則它也會從 ALPHA 中移除遠端異動日誌。

結果

DLTMQM 和 **RMVMQMINF** 刪除 **CRTMQM** 和 **ADDMQJRN** 所建立的本端和遠端異動日誌。 這些指令也會刪除異動日誌接收器。 異動日誌及異動日誌接收器必須遵循名稱以 AMQ 開頭的命名慣例。 **DLTMQM** 及 **RMVMQMINF** 從 `mqs.ini` 中移除佇列管理程式物件、佇列管理程式資料及佇列管理程式配置資訊。

下一步

替代方法是在步驟 第 215 頁的『1』 中取消啟動日誌登載之後，以及在結束佇列管理程式實例之前，發出下列指令。 或者，如果您未遵循命名慣例，則必須依名稱刪除異動日誌及異動日誌接收器。

1. 在 ALPHA 上:

   ```
   RMVMQMJRN MQMNAME('QM1') RMTJRNRDB('BETA')
   ```

2. 在測試版上:

   ```
   RMVMQMJRN MQMNAME('QM1') RMTJRNRDB('ALPHA')
   ```

刪除日誌之後，請繼續執行其餘步驟。

備份多重實例佇列管理程式

此程序顯示如何備份本端伺服器上的佇列管理程式物件，以及網路檔案伺服器上的佇列管理程式資料。 調整範例以備份其他佇列管理程式的資料。

開始之前

在此範例中，與佇列管理程式 QM1 相關聯的佇列管理程式資料使用 NetServer 儲存在稱為 GAMMA 的 IBM i 伺服器上。 請參閱 第 201 頁的『使用日誌登載鏡映及 NetServer 建立多重實例佇列管理程式』。 IBM MQ 安裝在伺服器上， Alpha 和 BETA。 佇列管理程式 QM1 在 ALPHA 和 BETA 上配置。

關於這項作業

IBM i 不支援從遠端目錄儲存資料。 使用檔案系統伺服器本端的備份程序，將佇列管理程式資料儲存在遠端檔案系統上。 在此作業中，網路檔案系統位於 IBM i 伺服器 GAMMA 上。 佇列管理程式資料會備份在 GAMMA 上的儲存檔中。

如果網路檔案系統位於 Windows 或 Linux 上，您可以將佇列管理程式資料儲存在壓縮檔中，然後儲存它。 如果您有備份系統 (例如 Tivoli Storage Manager) ，請使用它來備份佇列管理程式資料。

程序

1. 在 ALPHA 上為與 QM1 相關聯的佇列管理程式檔案庫建立儲存檔。

   使用佇列管理程式檔案庫名稱來命名儲存檔。

   ```
   CRTSAVF FILE(QGPL/QMQM1)
   ```

2. 將佇列管理程式檔案庫儲存在 ALPHA 上的儲存檔中。

   ```
   SAVLIB LIB(QMQM1) DEV(*SAVF) SAVF(QGPL/QMQM1)
   ```

3. 在 GAMMA 上建立佇列管理程式資料目錄的儲存檔。

   使用佇列管理程式名稱來命名儲存檔。

   ```
   CRTSAVF FILE(QGPL/QMDQM1)
   ```

4. 儲存 GAMMA 上本端目錄中佇列管理程式資料的副本。

   ```
   SAV DEV('/QSYS.LIB/QGPL.LIB/QMDQM1.FILE') OBJ('/QIBM/Userdata/mqm/qmgrs/QM1')
   ```

設定多重實例佇列管理程式的指令

IBM MQ 具有指令可簡化配置日誌登載抄寫、新增佇列管理程式實例，以及配置佇列管理程式以使用獨立 ASP。

建立及管理本端及遠端異動日誌的異動日誌指令為:

ADDMQMJRN
使用此指令，您可以建立佇列管理程式實例的具名本端及遠端日誌登載，並配置抄寫是同步還是非同步、同步逾時為何，以及是否要立即啟動遠端日誌登載。

CHGMQMJRN
此指令會修改影響抄本日誌的逾時、狀態及遞送參數。

RMVMQMJRN
從佇列管理程式實例移除指名的 遠端 日誌登載。

WRKMQMJRN
列出本端佇列管理程式實例的本端及遠端日誌登載狀態。

使用下列指令來新增及管理其他佇列管理程式實例，這些指令會修改 `mqs.ini` 檔案。

ADDMQMINF
此指令會使用您使用 DSPMQMINF 指令從 `mqs.ini` 檔案擷取的資訊，在不同的 IBM i 伺服器上新增佇列管理程式實例。

RMVMQMINF
移除佇列管理程式實例。 請使用這個指令來移除現有佇列管理程式的實例，或移除已從不同伺服器刪除之佇列管理程式的配置資訊。

CRTMQM 指令有三個參數可協助配置多重實例佇列管理程式，

MQMDIRP (*DFT | 目錄字首)
使用此參數來選取對映至網路儲存體上佇列管理程式資料的裝載點。

ASP (*SYSTEM | *ASPDEV | *auxiliary-storage-pool-number*)
指定 *SYSTEM 或 *auxiliary-storage-pool-number* ，以將佇列管理程式異動日誌放置在系統或基本使用者 ASP 上。 選取 *ASPDEV 選項，並使用 **ASPDEV** 參數設定裝置名稱，以將佇列管理程式日誌登載放置在獨立 ASP 上。

ASPDEV (*ASP | *device-name*)
指定主要或次要獨立 ASP 裝置的 *device-name* 。 選取 *ASP 的結果與指定 **ASP** (*SYSTEM) 的結果相同。

效能及磁碟失效接手考量

使用不同的輔助儲存區來增進效能和可靠性。

如果您在應用程式中使用大量持續訊息或大量訊息，則將這些訊息寫入磁碟所花費的時間會成為系統效能的重要因素。

請確定您有足夠的磁碟啟動來處理此可能性，或考量在其中保留佇列管理程式異動日誌接收器的個別「輔助儲存區 (ASP)」。

當您使用 **CRTMQM** 的 ASP 參數建立佇列管理程式時，您可以指定儲存佇列管理程式檔案庫及日誌登載的 ASP。 依預設，佇列管理程式檔案庫和異動日誌及 IFS 資料會儲存在系統 ASP 中。

ASP 容許隔離一或多個特定硬碟機上的物件。 這也可以減少由於磁碟媒體故障而造成的資料流失。 在大部分情況下，只會遺失受影響 ASP 中硬碟機上儲存的資料。

建議您將佇列管理程式檔案庫及日誌登載資料儲存在個別使用者 ASP 與根 IFS 檔案系統的 ASP 中，以提供失效接手並減少磁碟競用。

如需相關資訊，請參閱 備份及回復。

使用 SAVLIB 來儲存 IBM MQ 檔案庫

您無法使用 SAVLIB LIB(*ALLUSR) 來儲存 IBM MQ 檔案庫，因為這些檔案庫的名稱以 Q 開頭。

您可以使用 SAVLIB LIB(QM*) 來儲存所有佇列管理程式檔案庫，但前提是您是使用 *SAVF 以外的儲存裝置。 對於 DEV(*SAVF)，您必須對系統上每一個佇列管理程式檔案庫使用 SAVLIB 指令。

靜止 IBM MQ for IBM i

本節說明如何靜止 (循序結束) IBM MQ for IBM i

若要靜止 IBM MQ for IBM i:

1. 登入新的互動式 IBM MQ for IBM i 階段作業，確定您沒有存取任何物件。
2. 確保您已執行下列作業:
   - *ALLOBJ 權限，或 QMQM 檔案庫的物件管理權限
   - 有足夠權限使用 ENDSBS 指令
3. 通知所有使用者您將停止 IBM MQ for IBM i。
4. 然後如何繼續取決於您是否要關閉 (靜止) 單一佇列管理程式 (其他佇列管理程式可能存在的位置) (請參閱 第 218 頁的『關閉 IBM MQ for IBM i 的單一佇列管理程式』) 或所有佇列管理程式 (請參閱 第 219 頁的『關閉 IBM MQ for IBM i 的所有佇列管理程式』)。

ENDMQM 參數 ENDCCTJOB (*YES)

與舊版相比， ENDMQM 參數 ENDCCTJOB (*YES) 在 IBM MQ for IBM i V6.0 以及更新版本中的運作方式不同。

在舊版上，當您指定 ENDCCTJOB (*YES) 時， MQ 會強制終止您的應用程式。

在 IBM MQ for IBM i V6.0 或更新版本上，當您指定 ENDCCTJOB (*YES) 時，您的應用程式不會終止，而是與佇列管理程式切斷連線。

如果您指定 ENDCCTJOB (*YES) ，且您有應用程式未寫入以偵測佇列管理程式是否正在結束，則下次發出新的 MQI 呼叫時，該呼叫將會傳回 MQRC_CONNECTION_BROKEN (2009) 錯誤。

作為使用 ENDCCTJOB (*YES) 的替代方案，請使用參數 ENDCCTJOB (*NO) 並使用 WRKMQM 選項 22 (使用工作) 來手動結束將阻止佇列管理程式重新啟動的任何應用程式工作。

關閉 IBM MQ for IBM i 的單一佇列管理程式

使用此資訊來瞭解三種關機類型。

在接下來的程序中，我們使用範例佇列管理程式名稱 QMgr1 及範例子系統名稱 SUBX。 必要的話，請將這些名稱取代為您自己的值。

計劃的關閉

計劃在 IBM i 上關閉佇列管理程式

1. 關閉之前，請執行:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
```

2. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

   如果 QMgr1 未結束，則通道或應用程式可能忙碌中。
3. 如果您必須立即關閉 QMgr1 ，請執行下列指令:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

意外關閉

1. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

   如果 QMgr1 未結束，則通道或應用程式可能忙碌中。
2. 如果您需要立即關閉 QMgr1 ，請執行下列指令:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

在異常狀況下關閉

1. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

如果 QMgr1 未結束，請在下列情況下繼續步驟 3:

- QMgr1 位於其自己的子系統中，或
- 您可以結束與 QMgr1 共用相同子系統的所有佇列管理程式。 對所有這類佇列管理程式使用非計劃的關閉程序。

2. 當您針對共用子系統的所有佇列管理程式採取程序中的所有步驟 (在我們的範例中，是 SUBX) 時，請執行:

```
ENDSBS SUBX *IMMED
```

如果此指令無法完成，請使用非計劃性關閉程序關閉所有佇列管理程式，並在機器上執行 IPL。

警告: 對於因 ENDJOB 或 ENDSBS 而無法結束的 IBM MQ 工作，請不要使用 ENDJOBABN ，除非您準備在之後立即在機器上執行 IPL。

3. 執行下列指令來啟動子系統:

```
STRSBS SUBX
```

4. 執行下列指令，立即關閉佇列管理程式:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(10)
```

5. 執行下列指令來重新啟動佇列管理程式:

```
STRMQM MQMNAME(QMgr1)
```

如果失敗，且您:

- 執行 IPL 來重新啟動您的機器，或
- 只有單一佇列管理程式

執行下列指令來清理 IBM MQ 共用記憶體:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

在重複步驟 5 之前。

如果佇列管理程式重新啟動花費數秒以上的時間， IBM MQ 會間歇性將狀態訊息新增至工作日誌，以詳述啟動進度。

如果您在重新啟動佇列管理程式時仍有問題，請聯絡 IBM 支援中心。 您可能採取的任何進一步動作可能會損壞佇列管理程式，導致 IBM MQ 無法回復。

關閉 IBM MQ for IBM i 的所有佇列管理程式

使用此資訊來瞭解三種關機類型。

這些程序幾乎與單一佇列管理程式相同，但可能的話，會使用 *ALL 來取代佇列管理程式名稱，否則會輪流重複使用指令來使用每一個佇列管理程式名稱。 在整個程序中，我們使用範例佇列管理程式名稱 QMgr1 及範例子系統名稱 SUBX。 將這些取代為您自己的。

計劃的關閉

1. 關機前一小時，執行:

```
RCDMQMIMG OBJ(*ALL) OBJTYPE(*ALL) MQMNAME(QMgr1) DSPJRNDTA(*YES)
```

針對您要關閉的每一個佇列管理程式，重複此步驟。

2. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*CNTRLD)
```

對您要關閉的每一個佇列管理程式重複此步驟; 個別指令可以平行執行。

如果任何佇列管理程式未在合理時間 (例如 10 分鐘) 內結束，請繼續步驟 3。

3. 若要立即關閉所有佇列管理程式，請執行下列動作:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

意外關閉

1. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

對您要關閉的每一個佇列管理程式重複此步驟; 個別指令可以平行執行。

如果佇列管理程式未結束，則通道或應用程式可能忙碌。

2. 如果您需要立即關閉佇列管理程式，請執行下列動作:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

在異常狀況下關閉

1. 若要關閉佇列管理程式，請執行:

```
ENDMQM MQMNAME(QMgr1) OPTION(*IMMED)
```

對您要關閉的每一個佇列管理程式重複此步驟; 個別指令可以平行執行。

2. 透過執行下列指令來結束子系統 (在我們的範例中為 SUBX):

```
ENDSBS SUBX *IMMED
```

針對您要關閉的每一個子系統重複此步驟; 個別指令可以平行執行。

如果此指令無法完成，請在系統上執行 IPL。

警告: 對於因 ENDJOB 或 ENDSBS 而無法結束的工作，請勿使用 ENDJOBABN ，除非您準備在之後立即在系統上執行 IPL。

3. 執行下列指令來啟動子系統:

```
STRSBS SUBX
```

針對您要啟動的每個子系統重複此步驟。

4. 執行下列指令，立即關閉佇列管理程式:

```
ENDMQM MQMNAME(*ALL) OPTION(*IMMED)
ENDCCTJOB(*YES) TIMEOUT(15)
```

5. 執行下列指令來重新啟動佇列管理程式:

```
STRMQM MQMNAME(QMgr1)
```

針對您要啟動的每一個佇列管理程式，重複此步驟。

如果任何佇列管理程式重新啟動需要幾秒鐘以上的時間， IBM MQ 會顯示間歇性詳細說明啟動進度的狀態訊息。

如果您在重新啟動任何佇列管理程式時仍有問題，請聯絡 IBM 支援中心。 您可能採取的任何進一步動作可能會損壞佇列管理程式，導致 MQSeries 或 IBM MQ 無法回復。

z/OS 管理 IBM MQ for z/OS

管理佇列管理程式及相關聯資源包括您經常執行以啟動及管理那些資源的作業。 選擇您偏好用來管理佇列管理程式及相關聯資源的方法。

IBM MQ for z/OS 可以由產品隨附的一組公用程式來控制及管理。 您可以使用 IBM MQ Script (MQSC) 指令或「可程式指令格式 (PCF)」來管理 IBM MQ for z/OS。 如需對 IBM MQ for z/OS 使用指令的相關資訊，請參閱 第 221 頁的『對 IBM MQ for z/OS 發出指令』。

IBM MQ for z/OS 也提供一組公用程式來協助您進行系統管理。 如需不同公用程式及其使用方式的相關資訊，請參閱 第 228 頁的『IBM MQ for z/OS 公用程式』。

如需如何管理 IBM MQ for z/OS 以及您可能必須執行的不同管理作業的詳細資料，請參閱下列鏈結:

相關概念

第 62 頁的『管理本端 IBM MQ 物件』
本節告訴您如何管理本端 IBM MQ 物件，以支援使用「訊息佇列介面 (MQI)」的應用程式。 在此環境定義中，本端管理是指建立、顯示、變更、複製及刪除 IBM MQ 物件。

第 114 頁的『管理遠端 IBM MQ 物件』

第 5 頁的『管理 IBM MQ』
管理佇列管理程式及相關聯資源包括您經常執行以啟動及管理那些資源的作業。 選擇您偏好用來管理佇列管理程式及相關聯資源的方法。

相關資訊

IBM MQ for z/OS 概念
規劃
在 z/OS 上規劃 IBM MQ 環境
配置
配置 z/OS
可程式化指令格式參照
MQSC 參照
使用 IBM MQ for z/OS 公用程式

對 IBM MQ for z/OS 發出指令

您可以在批次或互動模式下使用 IBM MQ Script 指令 (MQSC) 來控制佇列管理程式。

IBM MQ for z/OS 支援 MQSC 指令，這些指令可以從下列來源發出:

- z/OS 主控台或對等項目 (例如 SDSF/TSO)。
- 起始設定輸入資料集。
- 提供的批次公用程式 CSQUTIL 會處理循序資料集中的指令清單。
- 適當授權的應用程式，將指令當作訊息傳送至指令輸入佇列。 應用程式可以是下列任何一項:
  - 批次區域程式
  - CICS 應用程式
  - IMS 應用程式

- TSO 應用程式
- 另一個 IBM MQ 系統上的應用程式或公用程式

第 224 頁的表 16 彙總 MQSC 指令及可從中發出指令的來源。

這些指令的大部分功能都可以從 IBM MQ for z/OS 作業及控制台中方便地取得。

在重新啟動 IBM MQ 子系統之後，會保留使用指令 (直接或間接) 對佇列管理程式的資源定義所做的變更。

IBM MQ for z/OS 也支援「可程式化指令格式 (PCF)」指令。 這些可簡化建立應用程式以管理 IBM MQ。 MQSC 指令是人類可讀的文字形式，而 PCF 可讓應用程式建立要求並讀取回覆，而無需剖析字串。 與 MQSC 指令一樣，應用程式也會將 PCF 指令當作訊息傳送至指令輸入佇列，以發出 PCF 指令。 如需使用 PCF 指令的相關資訊，以及指令的詳細資料，請參閱 可程式化指令格式參照 說明文件。

專用和廣域定義

當您在 IBM MQ for z/OS 上定義物件時，可以選擇是要與其他佇列管理程式共用該定義 (廣域 定義) ，還是只由一個佇列管理程式使用該物件定義 (專用 定義)。 這稱為物件 處置。

廣域定義

如果您的佇列管理程式屬於佇列共用群組，您可以選擇與群組其他成員共用您所建立的任何物件定義。 這表示您必須只定義物件一次，以減少整個系統所需的定義總數。

廣域物件定義保留在 共用儲存庫 (Db2 共用資料庫) 中，可供佇列共用群組中的所有佇列管理程式使用。 這些物件的處置方式為 GROUP。

專用定義

如果您只想建立一個佇列管理程式所需要的物件定義，或您的佇列管理程式不是佇列共用群組的成員，您可以建立不與佇列共用群組其他成員共用的物件定義。

專用物件定義保留在定義佇列管理程式的頁集零上。 這些物件具有 QMGR 處置。

您可以為所有類型的 IBM MQ 物件建立專用定義，但 CF 結構除外 (亦即，通道、名稱清單、程序定義、佇列、佇列管理程式、儲存類別定義及鑑別資訊物件) ，以及所有類型的物件 (佇列管理程式除外) 的廣域定義。

IBM MQ 會自動將群組物件的定義複製到使用它的每一個佇列管理程式的頁集零。 您可以視需要暫時變更定義的副本， IBM MQ 可讓您在必要時重新整理儲存庫副本中的頁面集副本。

IBM MQ 一律在啟動時嘗試從儲存庫副本重新整理頁面集副本 (對於通道指令，這是在通道起始程式重新啟動時執行) ，或如果群組物件已變更。

註: 只有在您建立定義副本之後群組的定義已變更時，才會從群組的定義重新整理定義副本。

這可確保頁面集副本反映儲存庫上的版本，包括佇列管理程式非作用中時所做的任何變更。 會產生 DEFINE REPLACE 指令來重新整理副本，因此在某些情況下不會執行重新整理，例如:

- 如果開啟佇列副本，則變更佇列使用情形的重新整理會失敗。
- 如果佇列副本中有訊息，則刪除該佇列的重新整理會失敗。
- 如果佇列副本需要 ALTER 搭配 FORCE 來變更它。

在這些情況下，不會對該副本上執行重新整理，而是對所有其他佇列管理程式上的副本上執行重新整理。

如果佇列管理程式關閉，然後獨立式重新啟動，則會刪除任何物件的本端副本，除非例如，佇列具有相關聯的訊息。

第三個物件處置僅適用於本端佇列。 這可讓您建立共用佇列。 共用佇列的定義會保留在共用儲存庫上，且可供佇列共用群組中的所有佇列管理程式使用。 此外，共用佇列上的訊息也可供佇列共用群組中的所有佇列管理程式使用。 這在 共用佇列及佇列共用群組中有說明。 共用佇列的物件處置為 SHARED。

下表彙總已獨立式啟動且作為佇列共用群組成員之佇列管理程式的物件處置選項的效果。

處置	獨立式佇列管理程式	佇列共用群組的成員
QMGR	保留在頁集零上的物件定義。	保留在頁集零上的物件定義。

處置	獨立式佇列管理程式	佇列共用群組的成員
GROUP	不容許。	保留在共用儲存庫中的物件定義。 在群組中每一個佇列管理程式的頁集零上保留的本端副本。
共用	不容許。	保留在共用儲存庫中的佇列定義。 可供群組中任何佇列管理程式使用的訊息。

操作廣域定義

如果您要變更共用儲存庫中所保留物件的定義，則需要指定是要變更儲存庫上的版本，還是頁集零上的本端副本。 請使用物件處置作為指令的一部分來執行此動作。

將指令導向至不同的佇列管理程式

您可以使用 指令範圍 來控制指令執行所在的佇列管理程式。

您可以選擇在輸入指令的佇列管理程式上執行指令，或在佇列共用群組中的不同佇列管理程式上執行指令。 您也可以選擇在佇列共用群組中的所有佇列管理程式上平行發出特定指令。 這適用於 MQSC 指令及 PCF 指令。

這是由 指令範圍所決定。 指令範圍與物件處置一起使用，以判斷您要使用的物件版本。

例如，您可能想要變更物件的部分屬性，其定義保留在共用儲存庫中。

- 您可能只想變更一個佇列管理程式上的版本，而不要對儲存庫上的版本或其他佇列管理程式正在使用的版本進行任何變更。
- 您可能想要為未來使用者變更共用儲存庫中的版本，但保留現有副本不變。
- 您可能想要變更共用儲存庫中的版本，但也希望您所做的變更立即反映在佇列共用群組中的所有佇列管理程式上，這些佇列管理程式在其頁集零上保留物件的副本。

使用指令範圍來指定指令是在此佇列管理程式、另一個佇列管理程式或所有佇列管理程式上執行。 使用物件處置來指定您正在操作的物件是在共用儲存庫中 (群組物件)，還是在頁集零上的本端副本 (佇列管理程式物件)。

您不需要指定指令範圍和物件處置來使用共用佇列，因為佇列共用群組中的每個佇列管理程式會將共用佇列當作單一佇列來處理。

指令摘要

請使用本主題作為主要 MQSC 及 PCF 指令的參照。

第 223 頁的表 15 彙總 IBM MQ for z/OS 上可用來變更、定義、刪除及顯示 IBM MQ 物件的 MQSC 及 PCF 指令。

表 *15:* 依物件類型列出的主要 *MQSC* 及 *PCF* 指令摘要

MQSC 指令	**ALTER**	**定義**	**顯示畫面**	**DELETE**
PCF 指令	**變更**	**建立/複製**	**查詢**	**刪除**
AUTHINFO	X	X	X	X
CFSTATUS			X	
CFSTRUCT	X	X	X	X
CHANNEL	X	X	X	X
CHSTATUS			X	
名稱清單	X	X	X	X

表 *15:* 依物件類型列出的主要 *MQSC* 及 *PCF* 指令摘要 *(繼續)*				
MQSC 指令	**ALTER**	**定義**	**顯示畫面**	**DELETE**
PROCESS	X	X	X	X
QALIAS	M	M	M	M
QCLUSTER			M	
QLOCAL	M	M	M	M
QMGR	X		X	
QMODEL	M	M	M	M
QREMOTE	M	M	M	M
佇列	P	P	X	P
QSTATUS			X	
STGCLASS	X	X	X	X

表格符號的索引鍵:

- M = 僅限 MQSC
- P = 僅限 PCF
- X = 兩者

除了 第 223 頁的表 15 中彙總的那些動作之外，還有許多其他 MQSC 及 PCF 指令可讓您管理其他 IBM MQ 資源，以及執行其他動作。

第 224 頁的表 16 顯示每一個 MQSC 指令，以及可從中發出每一個指令的位置:

- CSQINP1 起始設定輸入資料集
- CSQINP2 起始設定輸入資料集
- z/OS 主控台 (或對等項目)
- SYSTEM.COMMAND.INPUT 佇列及指令伺服器 (來自應用程式、CSQUTIL 或 CSQINPX 起始設定輸入資料集)

表 *16:* 從中執行 *MQSC* 指令的來源				
指令	**CSQINP1**	**CSQINP2**	**z/OS 主控台**	**指令輸入佇列及伺服器**
ALTER AUTHINFO		X	X	X
ALTER BUFFPOOL		X	X	X
ALTER CFSTRUCT		X	X	X
ALTER CHANNEL		X	X	X
ALTER NAMELIST		X	X	X
ALTER PSID			X	X
ALTER PROCESS		X	X	X
ALTER QALIAS		X	X	X
ALTER QLOCAL		X	X	X
ALTER QMGR		X	X	X

表 16: 從中執行 *MQSC* 指令的來源 (繼續)

指令	CSQINP1	CSQINP2	z/OS 主控台	指令輸入佇列及伺服器
ALTER QMODEL		X	X	X
ALTER QREMOTE		X	X	X
ALTER SECURITY	X	X	X	X
ALTER STGCLASS		X	X	X
ALTER SUB		X	X	X
ALTER TOPIC		X	X	X
ALTER TRACE	X	X	X	X
保存日誌	X	X	X	X
備份 CFSTRUCT			X	X
CLEAR QLOCAL		X	X	X
DEFINE AUTHINFO		X	X	X
DEFINE BUFFPOOL	X	X		
DEFINE CFSTRUCT		X	X	X
定義通道		X	X	X
DEFINE LOG			X	X
DEFINE NAMELIST		X	X	X
DEFINE PROCESS		X	X	X
DEFINE PSID	X		X	X
DEFINE QALIAS		X	X	X
DEFINE QLOCAL		X	X	X
DEFINE QMODEL		X	X	X
DEFINE QREMOTE		X	X	X
DEFINE STGCLASS		X	X	X
DEFINE SUB			X	X
DEFINE TOPIC		X	X	X
DELETE AUTHINFO		X	X	X
DELETE BUFFPOOL			X	X
DELETE CFSTRUCT		X	X	X
刪除通道			X	X
刪除名單		X	X	X
刪除處理程序		X	X	X
DELETE PSID			X	X
DELETE QALIAS		X	X	X
DELETE QLOCAL		X	X	X

表 16: 從中執行 *MQSC* 指令的來源 (繼續)

指令	CSQINP1	CSQINP2	z/OS 主控台	指令輸入佇列及伺服器
DELETE QMODEL		X	X	X
DELETE QREMOTE		X	X	X
DELETE STGCLASS		X	X	X
DELETE SUB		X	X	X
刪除主題		X	X	X
顯示保存檔	X	X	X	X
DISPLAY AUTHINFO		X	X	X
DISPLAY CFSTATUS			X	X
DISPLAY CFSTRUCT		X	X	X
顯示通道		X	X	X
DISPLAY CHSTATUS			X	X
DISPLAY CLUSQMGR			X	X
DISPLAY CMDSERV	X	X	X	X
DISPLAY CONN		X	X	X
顯示 CHINIT		X	X	X
顯示群組		X	X	X
DISPLAY LOG	X	X	X	X
顯示名單		X	X	X
DISPLAY PROCESS		X	X	X
DISPLAY QALIAS		X	X	X
DISPLAY QCLUSTER		X	X	X
DISPLAY QLOCAL		X	X	X
DISPLAY QMGR		X	X	X
DISPLAY QMODEL		X	X	X
DISPLAY QREMOCE		X	X	X
DISPLAY QSTATUS		X	X	X
顯示佇列		X	X	X
DISPLAY SECURITY			X	X
DISPLAY STGCLASS		X	X	X
DISPLAY SUB		X	X	X
顯示主題		X	X	X
DISPLAY SYSTEM	X	X	X	X
顯示執行緒		X	X	X
顯示追蹤	X	X	X	X

表 16: 從中執行 *MQSC* 指令的來源 (繼續)

指令	CSQINP1	CSQINP2	z/OS 主控台	指令輸入佇列及伺服器
顯示使用情形		X	X	X
MOVE QLOCAL		X	X	X
Ping 通道			X	X
回復 BSDS	X	X	X	X
回復 CFSTRUCT			X	X
重新整理叢集		X	X	X
重新整理佇列管理程式		X	X	X
REFRESH SECURITY		X	X	X
重設通道			X	X
重設叢集		X	X	X
重設 QSTATS		X	X	X
重設 Tpipe			X	X
解析通道			X	X
解析不確定		X	X	X
回復佇列管理程式			X	X
RVerify 安全		X	X	X
設定保存	X	X	X	X
設定日誌	X	X	X	X
設定系統	X	X	X	X
啟動通道			X	X
開始 CHINIT		X	X	X
START CMDSERV	X	X	X	
啟動接聽器			X	X
開始佇列管理程式			X	
啟動追蹤	X	X	X	X
停止通道			X	X
停止 CHINIT			X	X
STOP CMDSERV	X	X	X	
停止接聽器			X	X
停止佇列管理程式			X	X
停止追蹤	X	X	X	X
SUSPEND 佇列管理程式			X	X

在 MQSC 指令中，每一個指令說明會識別可從中執行該指令的來源。

起始設定指令

起始設定指令可用來控制佇列管理程式啟動。

在佇列管理程式啟動時起始設定 IBM MQ 時，會處理起始設定輸入資料集中的指令。 可以從起始設定輸入資料集發出三種類型的指令:

- 這些指令用來定義無法在其他位置定義的 IBM MQ 實體，例如 DEFINE BUFFPOOL。

  這些指令必須位於 DD 名稱 CSQINP1 所識別的資料集中。 在起始設定的重新啟動階段之前，會先處理它們。 無法透過主控台、作業及控制台或應用程式來發出它們。 這些指令的回應會寫入已啟動作業程序的 CSQOUT1 陳述式中所參照的循序資料集。
- 這些指令用來定義在重新啟動之後可回復的 IBM MQ 物件。 這些定義必須在 DD 名稱 CSQINP2 所識別的資料集中指定。 它們儲存在頁集零中。 在起始設定的重新啟動階段之後，會處理 CSQINP2 。 這些指令的回應會寫入已啟動作業程序的 CSQOUT2 陳述式中所參照的循序資料集。
- 用來操作 IBM MQ 物件的指令。 這些指令也必須在 DD 名稱 CSQINP2 所識別的資料集中指定。 例如，IBM MQ 提供的範例包含 ALTER QMGR 指令，以指定子系統的無法傳送郵件的佇列。 這些指令的回應會寫入 CSQOUT2 輸出資料集。

註: 如果 IBM MQ 物件定義在 CSQINP2 中，則 IBM MQ 會在每次啟動佇列管理程式時嘗試重新定義它們。 如果物件已存在，則嘗試定義它們會失敗。 如果您需要在 CSQINP2 中定義物件，您可以使用 DEFINE 指令的 REPLACE 參數來避免此問題，不過，這會置換前次執行佇列管理程式期間所做的任何變更。

IBM MQ for z/OS 隨附範例起始設定資料集成員。 它們在 IBM MQ 提供的範例定義中說明。

分散式佇列作業的起始設定指令

您也可以使用 START CHINIT 指令的 CSQINP2 起始設定資料集。 如果您需要一系列其他指令來定義分散式佇列作業環境 (例如，啟動接聽器)， IBM MQ 會提供稱為 CSQINPX 的第三個起始設定輸入資料集，該資料集在通道起始程式啟動作業程序中處理。

資料集中包含的 MQSC 指令會在通道起始程式起始設定結束時執行，並將輸出寫入 CSQOUTX DD 陳述式指定的資料集。 例如，您可以使用 CSQINPX 起始設定資料集來啟動接聽器。

IBM MQ for z/OS 提供了通道起始程式起始設定資料集成員範例。 它在 IBM MQ 提供的範例定義中說明。

發佈/訂閱的起始設定指令

如果您需要一系列指令來定義發佈/訂閱環境 (例如，定義訂閱時)， IBM MQ 會提供第四個起始設定輸入資料集，稱為 CSQINPT。

資料集中包含的 MQSC 指令會在發佈/訂閱起始設定結束時執行，並將輸出寫入 CSQOUTT DD 陳述式指定的資料集。 例如，您可以使用 CSQINPT 起始設定資料集來定義訂閱。

IBM MQ for z/OS 隨附了發佈/訂閱起始設定資料集成員範例。 它在 IBM MQ 提供的範例定義中說明。

IBM MQ for z/OS 公用程式

IBM MQ for z/OS 提供一組公用程式，您可以用來協助進行系統管理。

IBM MQ for z/OS 提供一組公用程式來協助您執行各種管理作業，包括下列各項:

- 管理訊息安全原則。
- 執行備份、還原及重組作業。
- 發出指令並處理物件定義。
- 產生資料轉換結束程式。
- 修改引導資料集。
- 列出日誌的相關資訊。
- 列印日誌。

- 設定 Db2 表格及其他 Db2 公用程式。
- 處理無法傳送郵件的佇列上的訊息。

訊息安全原則公用程式

訊息安全原則公用程式 (CSQ0UTIL) 會以獨立式公用程式來執行，以管理訊息安全原則。 如需相關資訊，請參閱 訊息安全原則公用程式 (CSQ0UTIL) 。

CSQUTIL 公用程式

這是提供來協助您執行備份、還原及重組作業的公用程式。 如需相關資訊，請參閱 CSQUTIL 公用程式 。

資料轉換結束程式公用程式

IBM MQ for z/OS 資料轉換結束程式公用程式 (**CSQUCVX**) 作為獨立式公用程式來執行，以建立資料轉換結束常式。

變更日誌庫存公用程式

IBM MQ for z/OS 變更日誌庫存公用程式 (**CSQJU003**) 以獨立式公用程式來執行，以變更引導資料集 (BSDS)。 您可以使用此公用程式來執行下列功能:

- 新增或刪除作用中或保存日誌資料集。
- 提供保存日誌的密碼。

列印日誌對映公用程式

IBM MQ for z/OS 列印日誌對映公用程式 (**CSQJU004**) 會以獨立式公用程式來執行，以列出下列資訊:

- 所有作用中及保存日誌資料集的兩個副本的日誌資料集名稱及日誌 RBA 關聯。 如果雙重記載不在作用中，則只有一個資料集副本。
- 可用於新日誌資料的作用中日誌資料集。
- 引導資料集 (BSDS) 中檢查點記錄佇列的內容。
- 保存日誌指令歷程記錄的內容。
- 系統及公用程式時間戳記。

日誌列印公用程式

日誌列印公用程式 (**CSQ1LOGP**) 作為獨立式公用程式執行。 您可以指定下列指令來執行公用程式:

- 引導資料集 (BSDS)
- 作用中日誌 (不含 BSDS)
- 保存日誌 (不含 BSDS)

佇列共用群組公用程式

佇列共用群組公用程式 (**CSQ5PQSG**) 作為獨立式公用程式來執行，以設定 Db2 表格，並執行佇列共用群組所需的其他 Db2 作業。

作用中日誌預先格式化公用程式

作用中日誌預先格式化公用程式 (**CSQJUFMT**) 會先格式化作用中日誌資料集，然後再由佇列管理程式使用它們。 如果公用程式已預先格式化作用中日誌資料集，則佇列管理程式第一次通過作用中日誌的日誌寫入效能會有所改善。

無法傳送郵件的佇列處理程式公用程式

無法傳送郵件的佇列處理程式公用程式 (**CSQUDLQH**) 作為獨立式公用程式執行。 它會檢查無法傳送郵件的佇列上的訊息，並根據您提供給公用程式的一組規則來處理它們。

CSQUTIL 公用程式

IBM MQ for z/OS 隨附了 CSQUTIL 公用程式，可協助您執行備份、還原及重組作業，以及發出指令和處理物件定義。

如需 CSQUTIL 公用程式的相關資訊，請參閱 IBM MQ 公用程式 (CSQUTIL)。 透過使用此公用程式，您可以呼叫下列函數:

指令
發出 MQSC 指令、記錄物件定義，以及建立用戶端通道定義檔。

COPY
讀取具名 IBM MQ for z/OS 訊息佇列的內容或具名頁集之所有佇列的內容，並將它們放入循序檔並保留原始佇列。

COPYPAGE
將整個頁面集複製到較大的頁面集。

空
若要刪除具名 IBM MQ for z/OS 訊息佇列的內容或具名頁集之所有佇列的內容，請保留佇列的定義。

格式
格式化 IBM MQ for z/OS 頁集。

載入
從 COPY 函數建立的循序檔還原具名 IBM MQ for z/OS 訊息佇列的內容或具名頁集的所有佇列的內容。

PAGEINFO
從一或多個頁集擷取頁集資訊。

重設頁面
將整個頁面集複製到其他頁面集資料集，並重設副本中的日誌資訊。

SCOPY
在佇列管理程式離線時，將佇列內容複製到資料集。

SDEFS
在佇列管理程式離線時，為物件產生一組定義指令。

SLOAD
從先前 COPY 或 SPROTICT 作業的目的地資料集還原訊息。 SLOAD 會處理單一佇列。

switch
切換或查詢與叢集傳送端通道相關聯的傳輸佇列。

XPARM
將通道起始程式參數載入模組轉換成佇列管理程式屬性 (用於移轉)。

操作 IBM MQ for z/OS

使用這些基本程序來操作 IBM MQ for z/OS。

您也可以使用與 IBM MQ for Windows、 IBM MQ for Linux (x86 及 x86-64 平台) 及 SupportPac MS0T 一起配送的 IBM MQ Explorer 來執行本節中說明的作業。 如需相關資訊，請參閱 第 54 頁的『使用 MQ Explorer 進行管理』 及 IBM 支援與下載。

本節包含下列主題的相關資訊:

發出佇列管理程式指令

您可以從 z/OS 主控台或使用公用程式 CSQUTIL 來發出 IBM MQ 控制指令。 指令可以使用指令字首字串 (CPF) 來指出哪個 IBM MQ 子系統處理指令。

您可以使用 IBM MQ 指令來控制 IBM MQ 的大部分作業環境。 IBM MQ for z/OS 同時支援這些指令的 MQSC 及 PCF 類型。 本主題說明如何使用 MQSC 指令來指定屬性，因此它會使用 MQSC 指令名稱而非 PCF 名稱來參照那些指令及屬性。 如需 MQSC 指令語法的詳細資料，請參閱 MQSC 指令。 如需 PCF 指令語法的詳細資料，請參閱 第 10 頁的『使用可程式指令格式』。 如果您是適當授權的使用者，則可以從下列發出 IBM MQ 指令:

- 起始設定輸入資料集 (如 第 228 頁的『起始設定指令』 所述)。
- z/OS 主控台或對等項目，例如 SDSF
- z/OS 主要 get 指令常式 MGCRE (SVC 34)
- IBM MQ 公用程式 CSQUTIL (在 IBM MQ 公用程式中說明。)
- 使用者應用程式，可以是:
  - CICS 程式
  - TSO 程式
  - z/OS 批次程式
  - IMS 程式

  如需相關資訊，請參閱 第 248 頁的『撰寫程式以管理 IBM MQ』 。

這些指令的大部分功能由作業及控制面板以方便的方式提供，可從 TSO 及 ISPF 存取，並在 第 235 頁的『介紹作業及控制面板』中說明。

如需進一步資訊，請參閱

- 第 231 頁的『從 z/OS 主控台或其對等項目發出指令』
  - 指令字首字串
  - 使用 z/OS 主控台發出指令
  - 指令回應
- 從公用程式 CSQUTIL 發出指令

從 z/OS 主控台或其對等項目發出指令

您可以從 z/OS 主控台或其對等項目發出所有 IBM MQ 指令。 您也可以從可以發出 z/OS 指令 (例如 SDSF) 的任何地方發出 IBM MQ 指令，或透過使用 MGCRE 巨集的程式來發出。

在主控台鍵入指令後可顯示的資料數量上限為 32 KB。

註:

1. 您無法從 IMS 終端機使用 IMS/SSR 指令格式來發出 IBM MQ 指令。 IMS 配接器不支援此功能。
2. SDSF 提供的輸入欄位對於部分指令 (特別是通道的那些指令) 可能不夠長。

指令字首字串

每一個 IBM MQ 指令都必須以指令字首字串 (CPF) 作為字首，如 第 232 頁的圖 40 所示。

因為多個 IBM MQ 子系統可以在 z/OS 下執行，所以 CPF 用來指出哪個 IBM MQ 子系統處理指令。 例如，針對稱為 CSQ1 的子系統啟動佇列管理程式，其中 CPF 是 ' +CSQ1 ' ，您從操作員主控台發出指令 +CSQ1 START QMGR。 此 CPF 必須定義在子系統名稱表格中 (針對子系統 CSQ1)。 這在 定義指令字首字串 (CPF)中有說明。 在範例中，字串 ' +CSQ1 ' 作為指令字首。

使用 z/OS 主控台來發出指令

您可以從 z/OS 主控台鍵入簡式指令，例如 第 232 頁的圖 40 中的 DISPLAY 指令。 不過，對於複式指令或您經常發出的指令集，發出指令的其他方法會更好。

```
+CSQ1 DISPLAY QUEUE(TRANSMIT.QUEUE.PROD) TYPE(QLOCAL)
```

圖 *40:* 從 *z/OS* 主控台發出 *DISPLAY* 指令

指令回應

指令的直接回應會傳送至發出指令的主控台。 IBM MQ 支援 z/OS 中可用的 延伸主控台支援 (EMCS) 功能，因此可以使用具有 4 位元組 ID 的主控台。 此外，當指令由使用 MGCRE 巨集的程式發出時，START QMGR 及 STOP QMGR 以外的所有指令都支援使用「指令及回應記號 (CART)」。

從公用程式 CSQUTIL 發出指令

您可以使用公用程式 CSQUTIL 的 COMMAND 函數，從循序資料集發出指令。 此公用程式會將指令以訊息形式傳送至 系統指令輸入佇列 ，並等待回應，該回應與 SYSPRINT 中的原始指令一起列印。 如需詳細資料，請參閱 IBM MQ 公用程式。

啟動及停止佇列管理程式

請使用本主題作為停止及啟動佇列管理程式的簡介。

本節說明如何啟動和停止佇列管理程式。 它包含下列主題的相關資訊:

- 第 232 頁的『開始之前 IBM MQ』
- 第 232 頁的『啟動佇列管理程式』
- 第 234 頁的『停止佇列管理程式』

啟動和停止佇列管理程式相對直接明確。 當佇列管理程式在正常狀況下停止時，其最後一個動作是採取終止檢查點。 此檢查點及日誌會提供佇列管理程式重新啟動所需的資訊。

本節包含 START 及 STOP 指令的相關資訊，並包含在發生異常終止之後啟動的簡要概觀。

開始之前 IBM MQ

安裝 IBM MQ 之後，它會定義為正式 z/OS 子系統。 在 z/OS: 的任何起始程式載入 (IPL) 期間會出現此訊息

```
CSQ3110I +CSQ1 CSQ3UR00 - SUBSYSTEM ssnm INITIALIZATION COMPLETE
```

其中 *ssnm* 是 IBM MQ 子系統名稱。

從現在開始，您可以 從已授權發出系統控制指令的任何 *z/OS* 主控台 啟動該子系統的佇列管理程式; 即 z/OS SYS 指令群組。 您必須從授權主控台發出 START 指令，無法透過 JES 或 TSO 來發出。

如果您使用佇列共用群組，則必須先啟動 RRS ，然後再啟動 Db2，然後再啟動佇列管理程式。

啟動佇列管理程式

您可以發出 START QMGR 指令來啟動佇列管理程式。 不過，除非您具有適當的權限，否則無法順利使用 START 指令。 如需 IBM MQ 安全的相關資訊，請參閱 在 z/OS 上設定安全 。 第 233 頁的圖 41 顯示 START 指令的範例。 (請記住，您必須在 IBM MQ 指令前面加上指令字首字串 (CPF)。)

```
+CSQ1  START QMGR
+CSQ1  START QMGR PARM(NEWLOG)
```

圖 *41:* 從 *z/OS* 主控台啟動佇列管理程式

如需 START QMGR 指令語法的相關資訊，請參閱 START QMGR 。

您無法將佇列管理程式當作批次工作來執行，或使用 z/OS 指令 START 來啟動它。 這些方法可能會啟動 IBM MQ 的位址空間，然後異常結束。 您也無法從 CSQUTIL 公用程式或類似使用者應用程式啟動佇列管理程式。

不過，您可以透過將 START QMGR 指令傳遞至 z/OS MGCRE (SVC 34) 服務，從 APF 授權程式啟動佇列管理程式。

如果您使用佇列共用群組，當您啟動佇列管理程式時，相關聯的 Db2 系統和 RRS 必須在作用中。

啟動選項

當您啟動佇列管理程式時，會載入系統參數模組。 您可以使用下列兩種方式之一來指定系統參數模組的名稱:

- 使用 /cpf START QMGR 指令的 PARM 參數，例如

```
/cpf START QMGR PARM(CSQ1ZPRM)
```

- 使用啟動程序中的參數，例如，將 JCL EXEC 陳述式撰寫為

```
//MQM  EXEC PGM=CSQYASCP,PARM='ZPARM(CSQ1ZPRM)'
```

系統參數模組提供自訂佇列管理程式時指定的資訊。

您也可以使用 ENVPARM 選項，將 JCL 程序中的一或多個參數替換為佇列管理程式。

例如，您可以更新佇列管理程式啟動程序，讓 DDname CSQINP2 成為變數。 這表示您可以在不變更啟動程序的情況下變更 CSQINP2 DDname。 這適用於實作變更，為操作員提供取消，以及佇列管理程式作業。

假設佇列管理程式 CSQ1 的啟動程序看起來像 第 233 頁的圖 42。

```
//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=*
```

圖 *42:* 啟動程序範例

如果您接著使用下列指令來啟動佇列管理程式:

```
+CSQ1  START QMGR
```

使用的 CSQINP2 是稱為 CSQ1NORM 的成員。

不過，假設您要將新的程式套組放入正式作業，以便下次啟動佇列管理程式 CSQ1 時，會從成員 CSQ1NEW 取得 CSQINP2 定義。 若要這樣做，您可以使用下列指令來啟動佇列管理程式:

```
+CSQ1  START QMGR ENVPARM('INP2=NEW')
```

且會使用 CSQ1NEW 來取代 CSQ1NORM。 附註: z/OS 會將符號參數的 KEYWORD=value 規格 (如 INP2=NEW) 限制為 255 個字元。

在異常終止之後啟動

IBM MQ 會自動偵測在正常關機或異常終止之後重新啟動。

在異常結束之後啟動佇列管理程式不同於在發出 STOP QMGR 指令之後啟動佇列管理程式。 在 STOP QMGR 之後，系統會依序完成其工作，並在停止之前取得終止檢查點。 當您重新啟動佇列管理程式時，它會使用來自系統檢查點及回復日誌的資訊來判定關機時的系統狀態。

不過，如果佇列管理程式異常結束，則會終止而無法完成其工作或取得終止檢查點。 當您在異常終止之後重新啟動佇列管理程式時，它會使用日誌中的資訊來重新整理其在終止時狀態的知識，並通知您各種作業的狀態。 通常，重新啟動程序會解決所有不一致狀態。 但是，在某些情況下，您必須採取特定步驟來解決不一致。

啟動時的使用者訊息

當您順利啟動佇列管理程式時，佇列管理程式會產生一組啟動訊息。

停止佇列管理程式

在停止佇列管理程式之前，所有 IBM MQ 相關的「使用回覆寫入操作員 (WTOR)」訊息都必須接收回覆，例如，取得日誌要求。 第 234 頁的圖 43 中的每一個指令都會終止執行中的佇列管理程式。

```
+CSQ1  STOP QMGR
+CSQ1  STOP QMGR MODE(QUIESCE)
+CSQ1  STOP QMGR MODE(FORCE)

+CSQ1  STOP QMGR MODE(RESTART)
```

圖 *43:* 停止佇列管理程式

指令 STOP QMGR 預設為 STOP QMGR MODE (QUIESCE)。

在 QUIESCE 模式中， IBM MQ 不容許建立任何新的連線執行緒，但容許現有的執行緒繼續; 只有在所有執行緒都結束時，它才會終止。 應用程式可以要求在佇列管理程式靜止時收到通知。 因此，請儘可能使用 QUIESCE 模式，讓已要求通知的應用程式有機會中斷連線。 如需詳細資料，請參閱 終止期間發生的狀況 。

如果佇列管理程式未在合理時間內終止以回應 STOP QMGR MODE (QUIESCE) 指令，請使用 DISPLAY CONN 指令來判斷是否存在任何連線執行緒，並採取必要步驟來終止相關聯的應用程式。 如果沒有執行緒，請發出 STOP QMGR MODE (FORCE) 指令。

STOP QMGR MODE (QUIESCE E) 及 STOP QMGR MODE (FORCE) 指令會從 MVS 自動重新啟動管理程式 (ARM) 取消登錄 IBM MQ，防止 ARM 自動重新啟動佇列管理程式。 STOP QMGR MODE (RESTART) 指令的運作方式與 STOP QMGR MODE (FORCE) 指令相同，但它不會從 ARM 取消登錄 IBM MQ 。 這表示佇列管理程式可以立即自動重新啟動。

如果 IBM MQ 子系統未向 ARM 登錄，則會拒絕 STOP QMGR MODE (RESTART) 指令，並將下列訊息傳送至 z/OS 主控台:

```
CSQY205I ARM element arm-element is not registered
```

如果未發出此訊息，則會自動重新啟動佇列管理程式。 如需 ARM 的相關資訊，請參閱 第 297 頁的『使用 z/OS Automatic Restart Manager (ARM)』。

只有在 STOP QMGR MODE (FORCE) 未終止佇列管理程式時，才會取消佇列管理程式位址空間。

如果透過取消位址空間或使用指令 STOP QMGR MODE (FORCE) 來停止佇列管理程式，則會維護與已連接 CICS 或 IMS 系統的一致性。 資源的重新同步會在佇列管理程式重新啟動時啟動，並在建立與 CICS 或 IMS 系統的連線時完成。

註: 當您停止佇列管理程式時，可能會發現發出 IEF352I 訊息。 如果 z/OS 偵測到未將位址空間標示為無法使用會導致完整性暴露，則會發出此訊息。 您可以忽略這個訊息。

停止訊息

發出 STOP QMGR 指令之後，您會收到訊息 CSQY009I 及 CSQY002I，例如:

```
CSQY009I +CSQ1 ' STOP QMGR' COMMAND ACCEPTED FROM
USER(userid), STOP MODE(FORCE)
CSQY002I +CSQ1 QUEUE MANAGER STOPPING
```

其中 `userid` 是發出 STOP QMGR 指令的使用者 ID， MODE 參數取決於指令中指定的使用者 ID。

當 STOP 指令順利完成時， z/OS 主控台上會顯示下列訊息:

```
CSQ9022I +CSQ1 CSQYASCP ' STOP QMGR' NORMAL COMPLETION
CSQ3104I +CSQ1 CSQ3EC0X - TERMINATION COMPLETE
```

如果您使用 ARM，且未指定 MODE (RESTART)，則也會顯示下列訊息:

```
CSQY204I +CSQ1 ARM DEREGISTER for element arm-element type
arm-element-type successful
```

在顯示下列訊息之前，您無法重新啟動佇列管理程式:

```
CSQ3100I +CSQ1 CSQ3EC0X - SUBSYSTEM ssnm READY FOR START COMMAND
```

介紹作業及控制面板

您可以使用 IBM MQ 作業及控制台，對 IBM MQ 物件執行管理作業。 請使用本主題作為指令及控制台的簡介。

您可以使用這些畫面來定義、顯示、變更或刪除 IBM MQ 物件。 使用這些畫面來進行日常管理，以及對物件進行小型變更。 如果您要設定或變更許多物件，請使用 CSQUTIL 公用程式的 COMMAND 函數。

作業及控制面板支援通道起始程式 (例如，啟動通道或 TCP/IP 接聽器)、叢集作業及安全的控制項。 它們也可讓您顯示執行緒及頁集使用情形的相關資訊。

這些畫面的運作方式是透過系統指令輸入佇列，將 MQSC 類型 IBM MQ 指令傳送至佇列管理程式。

註:

1. z/OS IBM MQ 作業及控制台 (CSQOREXX) 可能不支援從第 7 版開始新增的所有新功能及參數。 例如，沒有用於直接操作主題物件或訂閱的畫面。

   使用下列其中一個支援的機制，可讓您管理發佈/訂閱定義及其他無法直接從其他畫面使用的其他系統控制項:

   a. IBM MQ 瀏覽者

   b. z/OS 主控台

   c. 可程式指令格式 (PCF) 訊息

   d. CSQUTIL 的 COMMAND 函數

   請注意， CSQOREXX 畫面中的一般 **Command** 動作可讓您發出任何有效的 MQSC 指令，包括 SMDS 相關指令。 您可以使用 CSQUTIL 的 COMMAND 函數發出的所有指令。
2. 您無法直接從畫面中的指令行發出 IBM MQ 指令。
3. 如果要使用作業和控制台，您必須具備正確的安全授權; 這在 指令安全和指令資源安全的使用者 ID 中有說明。
4. 您無法使用 CSQUTIL 或 CSQOREXX 畫面來提供使用者 ID 和密碼。 相反地，如果您的使用者 ID 對 MQCONN 中的 BATCH 設定檔具有 UPDATE 權限，則可以略過 **CHCKLOCL**(*REQUIRED* 設定。 如需相關資訊，請參閱 在本端連結的應用程式上使用 **CHCKLOCL** 。

作業及控制台的呼叫及規則

您可以透過 ISPF 畫面來控制 IBM MQ 及發出控制指令。

如何存取 IBM MQ 作業及控制台

如果已更新 IBM MQ 的 ISPF/PDF 主要選項功能表，您可以從該功能表存取 IBM MQ 作業和控制面板。 如需更新功能表的詳細資料，請參閱 作業 20: 設定作業及控制台。

您可以從 TSO 指令處理器畫面 (通常是 ISPF/PDF 主要選項功能表上的選項 6) 存取 IBM MQ 作業和控制面板。 您執行以執行此動作的執行程式名稱是 CSQOREXX。 它有兩個參數: `thlqual` 是要使用之 IBM MQ 程式庫的高階限定元， `langletter` 是識別要使用之國家語言程式庫的字母 (例如， E 代表 U.S)。 英文)。 如果 IBM MQ 程式庫永久安裝在 ISPF 設定中，則可以省略這些參數。 或者，您可以從 TSO 指令行發出 CSQOREXX。

這些面板設計為操作員及管理者使用，並提供最少的正式訓練。 在畫面執行的情況下閱讀這些指示，並試用建議的不同作業。

註: 使用畫面時，暫時動態佇列的名稱格式為 SYSTEM.CSQOREXX.* 已建立。

作業及控制面板的規則

請參閱 IBM MQ 物件的命名規則 ，以瞭解 IBM MQ 字串和名稱的一般規則。 不過，有些規則只適用於作業和控制台:

- 請勿以單引號或雙引號括住字串，例如說明。
- 如果您在文字欄位中併入單引號或引號，則不需要重複它或新增跳出字元。 所儲存的字元與您鍵入的字元完全相同; 例如:

```
This is Maria's queue
```

畫面處理器會將它們加倍，讓您將它們傳遞至 IBM MQ。 不過，如果它必須截斷您的資料才能執行此動作，則會執行此動作。

- 您可以在大部分欄位中使用大寫或小寫字元，當您按 Enter 鍵時，它們會變成大寫字元。 異常狀況如下:
  - 儲存類別名稱和連結機能結構名稱，必須以大寫 A 到 Z 開頭，後面接著大寫 A 到 Z 或數值字元。
  - 未翻譯的特定欄位。 其中包括:
    - 應用程式 ID
    - 說明
    - 環境資料
    - 物件名稱 (但如果您使用小寫物件名稱，則可能無法在 z/OS 主控台輸入它)
    - 遠端系統名稱
    - 觸發資料
    - 使用者資料
- 在名稱中，會忽略前導空白和前導底線。 因此，物件名稱不能以空白或底線開頭。
- 底線用來顯示空白欄位的範圍。 當您按 Enter 鍵時，尾端底線會取代為空白。
- 許多說明和文字欄位會呈現在多個組件中，每一個組件由 IBM MQ 獨立處理。 這表示會保留尾端空白，且文字不連續。

空白欄位

當您指定 IBM MQ 物件的 **定義** 動作時，定義畫面上的每一個欄位都包含一個值。 如需 IBM MQ 取得值之位置的相關資訊，請參閱顯示畫面的一般說明 (延伸說明)。 如果您鍵入含有空白的欄位，且不容許空白，則 IBM MQ 會將安裝預設值放置在欄位中，或提示您輸入必要值。

當您為 IBM MQ 物件指定 **變更** 動作時，變更畫面上的每一個欄位都會包含該欄位的現行值。 如果您鍵入含有空白的欄位，且不容許空白，則該欄位的值不會變更。

物件和動作

作業及控制台提供您許多不同類型的物件，以及您可以對它們執行的許多動作。

這些動作會列在起始畫面上，可讓您操作物件並顯示它們的相關資訊。 這些物件包括所有 IBM MQ 物件，以及一些額外的物件。 物件屬於下列種類。

- 佇列、處理程序、鑑別資訊物件、名稱清單、儲存類別及 CF 結構
- 通道
- 叢集物件
- 佇列管理程式及安全
- 連線
- 系統

請參閱 動作，以取得可對 IBM MQ 物件採取之動作的交互參照表格。

佇列、處理程序、鑑別資訊物件、名稱清單、儲存類別及 CF 結構

這些是基本 IBM MQ 物件。 每一種類型可以有許多種。 可以使用 LIST 或 DISPLAY、LIST 及 FILTER、DEFINE LIKE、MANAGE 和 ALTER 動作來列出、列出過濾器、定義及刪除它們，並具有可顯示及變更的屬性。 (使用 MANAGE 動作刪除物件。)

此種類由下列物件組成:

QLOCAL	本端佇列
QREMOTE	遠端佇列
QALIAS	佇列間接參照的別名佇列

QMODEL	動態定義佇列的模型佇列
佇列	任何類型的佇列
QSTATUS	本端佇列的狀態
PROCESS	發生觸發事件時要啟動之應用程式的相關資訊
AUTHINFO	鑑別資訊: 使用 LDAP 伺服器執行「憑證撤銷清冊 (CRL)」檢查所需的定義
名稱清單	名稱清單，例如佇列或叢集
STGCLASS	儲存體類別
CFSTRUCT	連結機能 (CF) 結構
CFSTATUS	CF 結構的狀態

通道

通道用於分散式佇列作業。 每一種類型可以有許多種，它們可以列出、列出，以及過濾器、已定義、已刪除、已顯示及已變更。 它們也可以使用 START、STOP 和 PERFORM 動作來使用其他功能。 PERFORM 提供重設、連線測試及解析通道功能。

此種類由下列物件組成:

CHANNEL	任何類型的通道
傳送者	傳送端通道
SERVER	伺服器通道
接收者	接收端通道
要求端	要求端通道
CLUSRCVR	叢集接收端通道
CLUSSDR	叢集傳送端通道
SVRCONN	伺服器連線通道
CLNTCONN	用戶端連線通道
CHSTATUS	通道連線的狀態

叢集物件

叢集物件是針對屬於叢集的佇列及通道自動建立的。 基本佇列及通道定義可以位於另一個佇列管理程式上。 每一種類型可以有許多種，且名稱可以重複。 它們可以列出、與過濾器一起列出，以及顯示。 PERFORM、START 及 STOP 也可以透過 LIST 動作來使用。

此種類由下列物件組成:

CLUSQ	叢集佇列，針對屬於叢集的佇列所建立
CLUSCHL	叢集通道，針對屬於叢集的通道建立
CLUSQMGR	叢集佇列管理程式，與叢集通道相同，但由其佇列管理程式名稱識別

叢集通道和叢集佇列管理程式確實有 PERFORM、START 和 STOP 動作，但只能透過 DISPLAY 動作間接執行。

佇列管理程式及安全

佇列管理程式及安全物件具有單一實例。 它們可以列出，並具有可顯示及變更的屬性 (使用 LIST 或 DISPLAY ，以及 ALTER 動作) ，以及具有可使用 PERFORM 動作的其他功能。

此種類由下列物件組成:

管理員	佇列管理程式 :PERFORM 動作提供暫停及回復叢集功能

安全	安全功能 :PERFORM 動作提供重新整理及重新驗證功能

連線

可以列出連線，並列出過濾器及顯示連線。

此種類僅包含連線物件 CONNECT。

系統

其他函數的集合。 此種類由下列物件組成:

系統	系統功能
CONTROL	SYSTEM 的同義字

可用的功能如下:

LIST 或 DISPLAY	顯示佇列共用群組、分散式佇列、頁集或資料集使用情形資訊。
執行	重新整理或重設叢集作業
開始	啟動通道起始程式或接聽器
停止	停止通道起始程式或接聽器

動作

下表顯示您可以對每一種物件類型執行的動作:

表 *17: IBM MQ* 物件的有效作業及控制面板動作

物件	變更	定義相似	管理 (1)	清單或顯示畫面	含有過濾器的清單	執行	開始	停止(S)
AUTHINFO	X	X	X	X	X			
CFSTATUS				X				
CFSTRUCT	X	X	X	X	X			
CHANNEL	X	X	X	X	X	X	X	X
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
CLUSSDR	X	X	X	X	X	X	X	X
連接				X	X			
CONTROL				X		X	X	X
管理員	X			X		X		
名稱清單	X	X	X	X	X			
PROCESS	X	X	X	X	X			
QALIAS	X	X	X	X	X			

表 17: *IBM MQ* 物件的有效作業及控制面板動作 (繼續)								
物件	**變更**	**定義相似**	**管理 (1)**	**清單或顯示畫面**	**含有過濾器的清單**	**執行**	**開始**	**停止(S)**
QLOCAL	X	X	X	X	X			
QMODEL	X	X	X	X	X			
QREMOTE	X	X	X	X	X			
QSTATUS				X	X			
佇列	X	X	X	X	X			
接收者	X	X	X	X	X	X	X	X
要求端	X	X	X	X	X	X	X	X
安全	X			X		X		
傳送者	X	X	X	X	X	X	X	X
SERVER	X	X	X	X	X	X	X	X
SVRCONN	X	X	X	X	X		X	X
STGCLASS	X	X	X	X	X			
系統				X		X	X	X

附註:

1. 提供 `Delete` 及其他功能
2. 使用 `List or Display` 動作

物件處置

您可以指定需要使用之物件的 處置 。 disposition 表示物件 **定義** 的保留位置，以及物件的行為方式。

只有在您使用下列任何物件類型時，此處置才有意義:

- 佇列
- 通道
- 程序
- 名單
- 儲存類別
- 鑑別資訊物件

如果您使用其他物件類型，則不會處理處置。

允許的值如下:

Q
QMGR。 物件定義位於佇列管理程式的頁集上，且只能由佇列管理程式存取。

C
收到 物件定義位於佇列管理程式的頁集上，且只能由佇列管理程式存取。 它們是定義為具有 GROUP 處置的物件本端副本。

P
專用。 物件定義位於佇列管理程式的頁集上，且只能由佇列管理程式存取。 物件已定義為具有 QMGR 或 COPY 處置。

G
GROUP。 物件定義位於共用儲存庫中，且可供佇列共用群組中的所有佇列管理程式存取。

S

共用。 此處置僅適用於本端佇列。 佇列定義位於共用儲存庫中，且可供佇列共用群組中的所有佇列管理程式存取。

A

好吧 如果動作佇列管理程式是目標佇列管理程式或 *，則會併入 **所有** 處置的物件; 否則，只會併入 QMGR 及 COPY 處置的物件。 這是預設值。

使用 *ISPF* 控制台選取佇列管理程式、預設值及層次

您可以在 ISPF 中使用 CSQOREXX 執行程式來控制佇列管理程式。

當您檢視起始畫面時，未連接任何佇列管理程式。 不過，只要按 Enter 鍵，即會連接至佇列管理程式，或連接至 **連接名稱** 欄位中指名之佇列共用群組中的佇列管理程式。 您可以將此欄位保留空白; 這表示您正在使用批次應用程式的預設佇列管理程式。 這在 CSQBDEFV 中定義 (如需相關資訊，請參閱 作業 19: 設定批次、TSO 和 RRS 配接器)。

使用 **目標佇列管理程式** 欄位來指定要執行您所要求動作的佇列管理程式。 如果您將此欄位保留空白，則會預設為 **連接名稱** 欄位中指定的佇列管理程式。 您可以指定不是您所連接的目標佇列管理程式。 在此情況下，您通常會指定遠端佇列管理程式物件的名稱，該物件提供佇列管理程式別名定義 (在開啟指令輸入佇列時，該名稱會用作 *ObjectQMgr* 名稱)。 如果要這麼做，您必須設定適當的佇列和通道來存取遠端佇列管理程式。

動作佇列管理程式 可讓您指定佇列管理程式，該佇列管理程式與 **目標佇列管理程式** 欄位中指定的佇列管理程式位於相同的佇列共用群組中，以作為您要求執行動作的佇列管理程式。 如果您在此欄位中指定 *，則會對佇列共用群組中的所有佇列管理程式執行您要求的動作。 如果將此欄位保留空白，則會預設為 **目標佇列管理程式** 欄位中指定的值。 **動作佇列管理程式** 欄位對應於使用 MQSC 指令中說明的 CMDSCOPE 指令修飾元。

佇列管理程式預設值

如果您將任何佇列管理程式欄位保留空白，或選擇連接至佇列共用群組，則當您按 Enter 鍵時，會開啟次要視窗。 這個視窗會確認您將使用的佇列管理程式名稱。 按 Enter 鍵繼續。 當您在提出一些要求之後回到起始畫面時，您會找到以實際名稱完成的欄位。

佇列管理程式層次

只有在 z/OS 上執行的佇列管理程式，以及符合畫面 (目前 710 或 800) 的指令層次時，「作業」和「控制」畫面才能令人滿意地運作。

如果不符合這些條件，則動作可能只能局部、不正確或完全無法運作，且無法辨識來自佇列管理程式的回覆。

如果動作佇列管理程式不是指令層次 800，則不會顯示部分欄位，且無法輸入部分值。 不允許一些物件和動作。 在這種情況下，會開啟次要視窗，要求您確認要繼續進行。

搭配使用功能鍵和指令行與 *ISPF* 控制台

如果要使用畫面，您必須使用功能鍵，或在 ISPF 控制面板指令區中輸入對等指令。

- 功能鍵
  - 處理您的動作
  - 第 242 頁的『顯示 IBM MQ 使用者訊息』
  - 取消動作
  - 取得說明
- 使用指令行

功能鍵

功能鍵具有 IBM MQ 的特殊設定。 (這表示您無法對功能鍵使用 ISPF 預設值; 如果您先前已在任何位置使用 KEYLIST OFF ISPF 指令，則必須在任何作業及控制台的指令區域中鍵入 KEYLIST ON ，然後按 Enter 鍵以啟用 IBM MQ 設定。)

這些功能鍵設定可以顯示在畫面上，如 第 243 頁的圖 44 所示。 如果未顯示設定，請在任何作業及控制面板的指令區域中鍵入 PFSHOW ，然後按 Enter 鍵。 若要移除設定的顯示畫面，請使用指令 PFSHOW OFF。

作業及控制面板中的功能鍵設定符合 CUA 標準。 雖然您可以透過一般 ISPF 程序 (例如 KEYLIST 公用程式) 來變更金鑰設定，但不建議您這麼做。

註: 使用 PFSHOW 和 KEYLIST 指令會影響您擁有的任何其他邏輯 ISPF 畫面，當您離開作業和控制面板時，它們的設定仍會保留。

處理您的動作
按 Enter 鍵以在畫面上執行所要求的動作。 畫面中的資訊會傳送至佇列管理程式進行處理。

每次在畫面中按 Enter 鍵時， IBM MQ 會產生一或多個操作員訊息。 如果作業成功，您會收到確認訊息 CSQ9022I，否則會收到一些錯誤訊息。

顯示 IBM MQ 使用者訊息
在任何畫面中按功能鍵 F10 ，以查看 IBM MQ 使用者訊息。

取消動作
在起始畫面上， F3 和 F12 會結束作業和控制面板，並讓您回到 ISPF。 不會將任何資訊傳送至佇列管理程式。

在任何其他畫面上，按功能鍵 F3 或 F12 ，以離開現行畫面 **忽略自前次按 Enter 鍵以來所鍵入的任何資料**。 同樣地，不會將任何資訊傳送至佇列管理程式。

- F3 會直接回到起始畫面。
- F12 帶您回到前一個畫面。

取得說明

每一個畫面都有相關聯的說明畫面。 說明畫面使用 ISPF 通訊協定:

- 在任何畫面上按功能鍵 F1 ，以查看作業的一般說明 (延伸說明)。
- 按功能鍵 F1 ，游標停在任何欄位上，以查看該欄位的特定說明。
- 從任何欄位說明畫面中按功能鍵 F5 ，以取得一般說明。
- 按功能鍵 F3 ，回到基本畫面，亦即您按下功能鍵 F1 的畫面。
- 從任何說明畫面中按功能鍵 F6 ，以取得功能鍵的說明。

如果說明資訊進入第二個或後續頁面，畫面右上方會顯示 **其他** 指示器。 請使用下列功能鍵來導覽說明頁面:

- F11 ，以進入下一個說明頁面 (如果有的話)。
- F10 ，以回到前一個說明頁面 (如果有的話)。

使用指令行

您絕不需要使用指令行來發出作業及控制面板所使用的指令，因為它們可從功能鍵取得。 所提供的指令行可讓您輸入一般 ISPF 指令 (如 PFSHOW)。

ISPF 指令 PANELID ON 顯示現行 CSQOREXX 畫面的名稱。

不論您有哪些 ISPF 設定，指令行一開始都會顯示在畫面底端的預設位置。 您可以從任何作業及控制台使用 SETTINGS ISPF 指令來變更指令行的位置。 這些設定會針對作業及控制台的後續階段作業而被記住。

使用作業及控制面板

請利用這個主題來調查從 CSQOREXX 顯示的起始控制面板

第 243 頁的圖 44 顯示當您啟動畫面階段作業時所顯示的畫面。

```
                    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
```

圖 *44: IBM MQ* 作業及控制起始畫面

從這個畫面中，您可以執行下列動作:

- 選擇您要的本端佇列管理程式，以及您要在該佇列管理程式、遠端佇列管理程式，還是在與本端佇列管理程式相同佇列共用群組中的另一個佇列管理程式上發出指令。 如果您需要變更佇列管理程式名稱，請改寫它。
- 在 **動作** 欄位中鍵入適當的數字，以選取您要執行的動作。
- 指定您要使用的物件類型。 如果您不確定物件類型為何，請按功能鍵 F1 以取得物件類型的說明。
- 指定您要使用之物件類型的處置方式。
- 顯示指定類型的物件清單。 在 **名稱** 欄位中鍵入星號 (*)，然後按 Enter 鍵以顯示已在動作佇列管理程式上定義的物件 (指定類型) 清單。 然後，您可以依序選取一或多個要使用的物件。 所有動作都可以從清單中取得。

註: 建議您選擇會導致顯示物件清單的選項，然後從該清單中執行。 請使用 **顯示** 動作，因為所有物件類型都容許這樣做。

使用指令機能

使用編輯器來輸入或修正要傳遞至佇列管理程式的 MQSC 指令。

從主要畫面 CSQOPRIA 中，選取選項 **8 指令**，以啟動「指令機能」。

您會看到循序檔 *prefix*.CSQUTIL.COMMANDS，用作 CSQUTIL COMMAND 函數的輸入; 請參閱 將指令發出至 IBM MQ。

您不需要以指令字首字串 (CPF) 作為指令的字首。

您可以透過以接續字元 **+** 或 **-**終止現行行，在後續行上繼續 MQSC 指令。 或者，使用行編輯模式來提供長 MQSC 指令或指令內長屬性值的值。

行編輯

若要使用行編輯，請將游標移至編輯畫面中的適當行，並使用 **F4** 在可捲動畫面中顯示單一行。 單行最多可以有 32 760 個位元組的資料。

若要保留行編輯，請執行下列動作:

- **F3 結束程式** 會儲存對行和結束程式所做的變更
- **F12 取消** 會回到編輯畫面，捨棄對行所做的變更。

若要捨棄在編輯階段作業中所做的變更，請使用 **F12 取消** 來終止編輯階段作業，讓檔案的內容保持不變。 未執行指令。

執行指令

當您完成輸入 MQSC 指令時，請使用 **F3 結束** 來終止編輯階段作業，以儲存檔案的內容，並呼叫 CSQUTIL 以將指令傳遞至佇列管理程式。 指令處理的輸出保留在檔案 *prefix*.CSQUTIL.OUTPUT。 編輯階段作業會自動開啟此檔案，讓您可以檢視回應。 按 **F3 結束** 以結束此階段作業，並回到主功能表。

使用 IBM MQ 物件

本文件中說明的許多作業都涉及操作 IBM MQ 物件。 物件類型是佇列管理程式、佇列、程序定義、名稱清單、通道、用戶端連線通道、接聽器、服務及鑑別資訊物件。

- 定義簡式佇列物件
- 定義其他類型的物件
- 使用物件定義
- 使用名稱清單

定義簡式佇列物件

若要定義新物件，請使用現有定義作為其基礎。 您可以透過下列三種方式之一來執行此動作:

- 透過選取物件，該物件是在起始畫面上所選取選項的結果所顯示清單的成員。 然後輸入動作類型 2 (**定義相似**) 在所選取物件旁的動作欄位中。 您的新物件具有所選物件的屬性，但處置除外。 然後，您可以視需要變更新物件中的任何屬性。
- 在起始畫面上，選取 **定義相似** 動作類型，在 **物件類型** 欄位中輸入您要定義的物件類型，然後在 **名稱** 欄位中輸入特定現有物件的名稱。 新物件的屬性與您在 **名稱** 欄位中指定的物件相同，但處置除外。 然後，您可以視需要變更新物件定義中的任何屬性。
- 選取 **定義相似** 動作類型，指定物件類型，然後將 **名稱** 欄位留白。 然後，您可以定義新物件，且它會為您的安裝定義預設屬性。 然後，您可以視需要變更新物件定義中的任何屬性。

註: 您不是在起始畫面上輸入您要定義之物件的名稱，而是在「 **定義** 」畫面上呈現您所定義的物件。

下列範例示範如何使用現有佇列作為範本來定義本端佇列。

定義本端佇列

如果要從作業和控制台定義本端佇列物件，請使用現有的佇列定義作為新定義的基礎。 有數個畫面要完成。 當您完成所有畫面且滿意屬性正確時，請按 Enter 鍵將您的定義傳送至佇列管理程式，然後該佇列管理程式會建立實際佇列。

在起始畫面上，或針對清單中的物件項目使用 **定義相似** 動作，該清單顯示為起始畫面上所選取選項的結果。

例如，從起始畫面開始，完成下列欄位:

動作	2 (定義相似)
物件類型	QLOCAL
名稱	QUEUE.YOU.LIKE. 這是為新佇列提供屬性的佇列名稱。

按 Enter 鍵以顯示 **定義本端佇列** 畫面。 佇列名稱欄位為空白，因此您可以提供新佇列的名稱。 說明是您以此新定義為基礎之佇列的說明。 以您自己的新佇列說明來輸入此欄位。

其他欄位中的值是您以此新佇列為基礎之佇列的值，但處置除外。 您可以視需要來重新輸入這些欄位。 例如，如果適當授權的應用程式可以將訊息放置在此佇列上，請在 **啟用放置** 欄位中鍵入 Y (如果它還不是 Y)。

將游標移至欄位並按功能鍵 F1，即可取得欄位說明。 欄位說明提供可用於每一個屬性的值相關資訊。

當您完成第一個畫面時，請按功能鍵 F8 以顯示第二個畫面。

提示:

1. 請 不要 在此階段按 Enter 鍵，否則會在您有機會完成其餘欄位之前建立佇列。 (如果您過早按 Enter 鍵，請不要擔心; 您隨時可以稍後變更您的定義。)
2. 請勿按功能鍵 F3 或 F12，否則您鍵入的資料將會遺失。

重複按功能鍵 F8 ，以查看並完成其餘畫面，包括觸發程式定義、事件控制及取消報告畫面。

當本端佇列定義完成時

當定義完成時，請按 Enter 鍵將資訊傳送至佇列管理程式以進行處理。 佇列管理程式會根據您提供的定義來建立佇列。 如果您不想要建立佇列，請按功能鍵 F3 以結束並取消定義。

定義其他類型的物件

若要定義其他類型的物件，請使用現有定義作為新定義的基礎，如 定義本端佇列中所述。

在起始畫面上，或針對清單中的物件項目使用 **定義相似** 動作，該清單顯示為起始畫面上所選取選項的結果。

例如，從起始畫面開始，完成下列欄位:

動作	2 (定義相似)
物件類型	QALIAS、NAMELIST、PROCESS、CHANNEL 及其他資源物件。
名稱	保留空白或輸入相同類型的現有物件名稱。

按 Enter 鍵以顯示對應的 DEFINE 畫面。 視需要完成欄位，然後再次按 Enter 鍵，將資訊傳送至佇列管理程式。

就像定義本端佇列一樣，定義另一種類型的物件通常需要完成數個畫面。 定義名單需要一些其他工作，如第 245 頁的『使用名稱清單』中所述。

使用物件定義

已定義物件時，您可以在 **動作** 欄位中指定動作，以變更、顯示或管理它。

在每一種情況下，您可以:

- 從清單中選取您要使用的物件，該清單會因為在起始畫面上選取的選項而顯示。 例如，在 **動作** 欄位中輸入 1 以顯示物件，在 **物件類型** 欄位中輸入 Queue ，在 **名稱** 欄位中輸入 * ，即會呈現系統中定義的所有佇列清單。 然後，您可以從此清單中選取您需要使用的佇列。
- 從起始畫面開始，您可以透過完成 **物件類型** 及 **名稱** 欄位來指定您正在使用的物件。

變更物件定義

若要變更物件定義，請指定動作 3 ，然後按 Enter 鍵以查看 ALTER 畫面。 這些面板與 DEFINE 面板非常類似。 您可以變更您想要的值。 當變更完成時，請按 Enter 鍵將資訊傳送至佇列管理程式。

顯示物件定義

如果您想要查看物件的詳細資料，但無法變更它們，請指定動作 1 ，然後按 Enter 鍵以查看 DISPLAY 畫面。 同樣地，這些畫面與 DEFINE 畫面類似，但您無法變更任何欄位。 請變更物件名稱，以顯示另一個物件的詳細資料。

刪除物件

若要刪除物件，請指定動作 4 (管理) ，並且 **刪除** 動作是產生的功能表上呈現的其中一個動作。 選取 **刪除** 動作。

系統會要求您確認您的要求。 如果您按功能鍵 F3 或 F12，則會取消要求。 如果您按 Enter 鍵，則會確認要求並傳遞至佇列管理程式。 然後會刪除您指定的物件。

註: 除非已啟動通道起始程式，否則無法刪除大部分類型的通道物件。

使用名稱清單

使用名稱清單時，請繼續進行其他物件的作業。

對於動作 DEFINE LIKE 或 ALTER，請按功能鍵 F11，將名稱新增至清單或變更清單中的名稱。 這涉及使用 ISPF 編輯器，且所有一般 ISPF 編輯指令都可供使用。 在個別行上輸入名稱清單中的每一個名稱。

當您以此方式使用 ISPF 編輯器時，功能鍵設定是一般 ISPF 設定， **不是** 其他作業和控制台所使用的設定。

如果您需要在清單中指定小寫名稱，請在編輯器畫面指令行上指定 CAPS (OFF)。 當您執行此動作時，在您指定 CAPS (ON) 之前，您未來編輯的所有名稱清單都是小寫。

當您完成編輯名單時，請按功能鍵 F3，以結束 ISPF 編輯階段作業。 然後按 Enter 鍵，將變更傳送至佇列管理程式。

請注意: 如果您在此階段未按 Enter 鍵，但改為按功能鍵 F3，則會遺失您鍵入的任何更新項目。

使用多個叢集傳輸佇列來實作系統

如果在單一叢集或重疊叢集中使用通道，則不會有任何不同。 當選取並啟動通道時，通道會根據定義來選取傳輸佇列。

關於這項作業

如果您使用 DEFCLXQ 選項，請參閱 第 246 頁的『使用佇列及切換的自動定義』。

如果您使用暫置方法，請參閱 第 246 頁的『使用階段式方法變更叢集傳送端通道』。

使用佇列及切換的自動定義

如果您計劃使用 DEFCLXQ 選項，請使用此選項。 將為每個通道及每個新通道建立一個佇列。

程序

1. 檢閱 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 並視需要變更屬性。
   此佇列定義在成員 SCSQPROC(csq4insx)中。
2. 建立 SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE 模型佇列。
3. 套用此模型佇列及 SYSTEM.CLUSTER.TRANSMIT. * * 佇列。
   對於 z/OS，通道起始程式已啟動作業使用者 ID 需要:
   - 控制對 CLASS (MQADMIN) for 的存取權

     ```
     ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channelname
     ```

   - 更新的 CLASS (MQQUEUE) 存取權

     ```
     ssid.SYSTEM.CLUSTER.TRANSMIT.channelname
     ```

使用階段式方法變更叢集傳送端通道

此處理程序可讓您在不同時間移至新的叢集傳送端通道，以符合您企業的需求。

開始之前

- 識別您的商業應用程式，以及使用哪些通道。
- 對於您使用的佇列，顯示它們在其中的叢集。
- 顯示通道以顯示連線名稱、遠端佇列管理程式的名稱，以及通道支援的叢集。

關於這項作業

- 建立傳輸佇列。 在 z/OS 上，您可能想要考量用於佇列的頁集。
- 設定佇列的安全原則。

- 請變更任何佇列監視，以併入此佇列名稱。
- 決定要使用此傳輸佇列的通道。 通道應該具有類似名稱，因此一般字元 '*' 在 CLCHNAME 中識別通道。
- 當您準備好使用新功能時，請變更傳輸佇列，以指定使用此傳輸佇列的通道名稱。 例如，CLUSTER1.TOPARIS 或 CLUSTER1.* 或 *.TOPARIS
- 啟動通道

程序

1. 使用 DIS CLUSQMGR(xxxx) XMITQ 指令來顯示叢集中定義的叢集傳送端通道，其中 *xxxx* 是遠端佇列管理程式的名稱。
2. 設定傳輸佇列的安全設定檔，並提供通道起始程式的佇列存取權。
3. 定義要使用的傳輸佇列，並指定 USAGE (XMITQ) INDXTYPE (CORRELID) SHARE 及 CLCHNAME (*value*)

   通道起始程式啟動作業使用者 ID 需要下列存取權:

   ```
   alter class(MQADMIN) ssid.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.channel
   update class(MQQUEUE ssid.SYSTEM.CLUSTER.TRANSMIT.channel
   ```

   且使用 SWITCH 指令的使用者 ID 需要下列存取權:

   ```
   alter cl(MQADMIN) ssid.QUEUE.queuename
   ```

4. 停止並重新啟動通道。

   當通道使用 MQSC 指令啟動時，或您使用 CSQUTIL 時，會發生通道變更。 您可以使用 CSQUTIL 的 SWITCH CHANNEL(*)STATUS 來識別需要重新啟動哪些通道

   如果您在啟動通道時發生問題，請停止通道，解決問題，然後重新啟動通道。

   請注意，您可以根據需要經常變更 CLCHNAME 屬性。

   所使用的 CLCHNAME 值是通道啟動時的值，因此您可以在通道從啟動時開始繼續使用定義時變更 CLCHNAME 定義。 通道會在重新啟動時使用新定義。

復原變更

如果變更結果不是您預期的結果，則您需要有一個處理程序來取消變更。

什麼會出問題?

如果新的傳輸佇列不是您預期的:

1. 請檢查 CLCHNAME 是否如您預期
2. 請檢閱工作日誌，以檢查交換器處理程序是否已完成。 如果沒有，請稍後等待並檢查通道的新傳輸佇列。

如果您使用多個叢集傳輸佇列，請務必明確設計傳輸佇列定義，並避免複雜的重疊配置。 如此一來，您可以確定如果有問題，您可以回到原始佇列和配置。

如果您在移至使用不同傳輸佇列期間遇到問題，則必須先解決所有問題，才能繼續進行變更。

必須先完成現有的變更要求，才能提出新的變更要求。 例如，您:

1. 定義新的傳輸佇列，其深度上限為 1 ，且有 10 則訊息等待傳送。
2. 請變更傳輸佇列，以在 CLCHNAME 參數中指定通道名稱。
3. 停止並重新啟動通道。 嘗試移動訊息失敗並報告問題。
4. 將傳輸佇列上的 CLCHNAME 參數變更為空白。
5. 停止並重新啟動通道。 通道會繼續嘗試並完成原始要求，因此通道會繼續使用新的傳輸佇列。
6. 需要解決問題並重新啟動通道，以便移動訊息順利完成。

下次重新啟動通道時，它會挑選任何變更，因此如果您將 CLCHNAME 設為空白，通道將不會使用指定的傳輸佇列。

在此範例中，將傳輸佇列上的 CLCHNAME 變更為空白，不一定表示通道使用 SYSTEM.CLUSTER.TRANSMIT 佇列，因為可能有其他傳輸佇列的 CLCHNAME 參數符合通道名稱。 例如，一般名稱或佇列管理程式屬性 DEFCLXQ 可能設為通道，因此通道會使用動態佇列而非 SYSTEM.CLUSTER.TRANSMIT 佇列。

撰寫程式以管理 IBM MQ

您可以撰寫自己的應用程式來管理佇列管理程式。 請利用這個主題來瞭解撰寫您自己的管理程式的需求。

啟動一般用途程式設計介面資訊

這組主題包含提示和指引，可讓您從 IBM MQ 應用程式發出 IBM MQ 指令。

註: 在本主題中，使用 C 語言表示法來說明 MQI 呼叫。 如需 COBOL、PL/I 及組譯語言中呼叫的一般呼叫，請參閱 函數呼叫 手冊。

瞭解一切如何運作

在大綱中，從應用程式發出指令的程序如下:

1. 將 IBM MQ 指令建置成 IBM MQ 訊息類型，稱為 要求訊息。 指令可以是 MQSC 或 PCF 格式。
2. 傳送 (使用 MQPUT) 將此訊息傳送至稱為系統指令輸入佇列的特殊佇列。 IBM MQ 指令處理器會執行指令。
3. 擷取 (使用 MQGET) 在回覆目的地佇列上作為 回覆訊息 的指令結果。 這些訊息包含使用者訊息，您需要這些訊息來判斷您的指令是否成功，如果成功，結果為何。

然後由您的應用程式來處理結果。

這組主題包含:

準備管理程式的佇列

管理程式需要一些預先定義的佇列，才能進行系統指令輸入及接收回應。

本節適用於 MQSC 格式的指令。 如需 PCF 中的對等項目，請參閱 第 10 頁的『使用可程式指令格式』。

您必須先定義要使用的佇列，然後開啟，才能發出任何 MQPUT 或 MQGET 呼叫。

定義系統指令輸入佇列

系統指令輸入佇列是稱為 SYSTEM.COMMAND.INPUT。 提供的 CSQINP2 起始設定資料集 thlqual.SCSQPROC(CSQ4INSG) 包含系統指令輸入佇列的預設定義。 為了與其他平台上的 IBM MQ 相容，這個佇列的別名稱為 SYSTEM.ADMIN.COMMAND.QUEUE 。 如需相關資訊，請參閱 IBM MQ 隨附的範例定義 。

定義回覆目的地佇列

您必須定義回覆目的地佇列，以接收來自 IBM MQ 指令處理器的回覆訊息。 它可以是任何具有容許在其上放置回覆訊息之屬性的佇列。 不過，對於一般作業，請指定下列屬性:

- USAGE (NORMAL)
- NOTRIGGER (除非您的應用程式使用觸發程式)

避免對指令使用持續訊息，但如果您選擇這樣做，則回覆目的地佇列不能是暫時動態佇列。

所提供的 CSQINP2 起始設定資料集 thlqual.SCSQPROC(CSQ4INSG) 包含稱為 SYSTEM.COMMAND.REPLY.MODEL。 您可以使用此模型來建立動態回覆目的地佇列。

註: 指令處理器所產生的回覆長度最多為 15 000 個位元組。

如果您使用永久動態佇列作為回覆目的地佇列，則您的應用程式應該容許在嘗試刪除佇列之前完成所有 PUT 及 GET 作業的時間，否則可能會傳回 MQRC2055 (MQRC_Q_NOT_EMPTY)。 如果發生此情況，請在幾秒後重試佇列刪除。

開啟系統指令輸入佇列

您的應用程式必須先連接至佇列管理程式，才能開啟系統指令輸入佇列。 請使用 MQI 呼叫 MQCONN 或 MQCONNX 來執行此動作。

然後使用 MQI 呼叫 MQOPEN 來開啟系統指令輸入佇列。 若要使用此呼叫，請執行下列動作:

1. 將 *Options* 參數設為 MQOO_OUTPUT
2. 設定 MQOD 物件描述子欄位，如下所示:

   ObjectType
   MQOT_Q (物件是佇列)

   ObjectName
   SYSTEM.COMMAND.INPUT

   ObjectQMgrName
   如果您要將要求訊息傳送至本端佇列管理程式，請將此欄位保留空白。 這表示會在本端處理您的指令。

   如果您想要在遠端佇列管理程式上處理 IBM MQ 指令，請在這裡放置其名稱。 您也必須設定正確的佇列及鏈結，如 分散式佇列及叢集中所述。

開啟回覆目的地佇列

若要從 IBM MQ 指令擷取回覆，您必須開啟回覆目的地佇列。 作法之一是在 MQOPEN 呼叫中指定模型佇列 SYSTEM.COMMAND.REPLY.MODEL ，以建立永久動態佇列作為回覆目的地佇列。 若要使用此呼叫，請執行下列動作:

1. 將 *Options* 參數設為 MQOO_INPUT_SHARED
2. 設定 MQOD 物件描述子欄位，如下所示:

   ObjectType
   MQOT_Q (物件是佇列)

   ObjectName
   回覆目的地佇列的名稱。 如果您指定的佇列名稱是模型佇列物件的名稱，則佇列管理程式會建立動態佇列。

   ObjectQMgrName
   若要在本端佇列管理程式上接收回覆，請將此欄位保留空白。

   DynamicQName
   指定要建立的動態佇列名稱。

使用指令伺服器

指令伺服器是與指令處理器元件搭配使用的 IBM MQ 元件。 您可以將格式化訊息傳送至指令伺服器，以解譯訊息、執行管理要求，以及將回應傳回給管理應用程式。

指令伺服器會從系統指令輸入佇列讀取要求訊息，驗證它們，並將有效的訊息當作指令傳遞至指令處理器。 指令處理器會處理指令，並將任何回覆作為回覆訊息放置在您指定的回覆目的地佇列上。 第一個回覆訊息包含使用者訊息 CSQN205I。 如需相關資訊，請參閱第 253 頁的『解譯來自指令伺服器的回覆訊息』。 不論從何處發出通道起始程式及佇列共用群組指令，指令伺服器也會處理這些指令。

識別處理指令的佇列管理程式

處理您從管理程式發出的指令的佇列管理程式是擁有系統指令輸入佇列的佇列管理程式。

啟動指令伺服器

通常，當佇列管理程式啟動時，指令伺服器會自動啟動。 只要從 START QMGR 指令傳回訊息 CSQ9022I 'START QMGR' NORMAL COMPLETION 即可使用。 在系統終止階段期間，當所有連接的作業都已斷線時，指令伺服器即會停止。

您可以使用 START CMDSERV 及 STOP CMDSERV 指令自行控制指令伺服器。 若要防止指令伺服器在 IBM MQ 重新啟動時自動啟動，您可以將 STOP CMDSERV 指令新增至 CSQINP1 或 CSQINP2 起始設定資料集。 不過，不建議這樣做，因為它會防止處理任何通道起始程式或佇列共用群組指令。

STOP CMDSERV 指令會在完成處理現行訊息之後立即停止指令伺服器，如果未處理任何訊息，則會立即停止指令伺服器。

如果指令伺服器已由程式中的 STOP CMDSERV 指令停止，則無法處理程式中的其他指令。 若要重新啟動指令伺服器，您必須從 z/OS 主控台發出 START CMDSERV 指令。

如果您在佇列管理程式執行時停止並重新啟動指令伺服器，則在指令伺服器停止時，系統指令輸入佇列上的所有訊息都會在指令伺服器重新啟動時處理。 不過，如果您在指令伺服器停止之後停止並重新啟動佇列管理程式，則在指令伺服器重新啟動時，只會處理系統指令輸入佇列上的持續訊息。 系統指令輸入佇列上的所有非持續訊息都會遺失。

將指令傳送至指令伺服器

對於每一個指令，您可以建置包含指令的訊息，然後將它放在系統指令輸入佇列中。

建置包含 IBM MQ 指令的訊息

您可以建置包含必要指令的要求訊息，以在應用程式中納入 IBM MQ 指令。 對於每一個這類指令，您可以執行下列動作:

1. 建立緩衝區，其中包含代表指令的字串。
2. 在呼叫的 *buffer* 參數中指定緩衝區名稱，以發出 MQPUT 呼叫。

在 C 中，最簡單的作法是使用 'char' 來定義緩衝區。 例如:

```
char message_buffer[ ] = "ALTER QLOCAL(SALES) PUT(ENABLED)";
```

當您建置指令時，請使用以空值結尾的字串。 請勿在以這種方式定義的指令開頭指定指令字首字串 (CPF)。 這表示如果您想要在另一個佇列管理程式上執行指令 Script ，則不需要變更它們。 不過，您必須考量 CPF 已包含在任何放入回覆目的地佇列的回應訊息中。

指令伺服器會將所有小寫字元全部轉換成大寫，除非它們是在引號內。

指令可以是任何長度，最多 32 762 個字元。

將訊息放置在系統指令輸入佇列上

使用 MQPUT 呼叫，將包含指令的要求訊息放置在系統指令輸入佇列上。 在此呼叫中，您指定已開啟的回覆目的地佇列名稱。

如果要使用 MQPUT 呼叫，請執行下列動作:

1. 設定下列 MQPUT 參數:

   Hconn
   MQCONN 或 MQCONNX 呼叫所傳回的連線控點。

   Hobj
   系統指令輸入佇列的 MQOPEN 呼叫所傳回的物件控點。

   BufferLength
   格式化指令的長度。

Buffer
包含指令的緩衝區名稱。

2. 設定下列 MQMD 欄位:

MsgType
MQMT_REQUEST

Format
MQFMT_STRING 或 MQFMT_NONE

如果您未使用與佇列管理程式相同的字碼頁，請適當地設定 *CodedCharSetId* 並設定 MQFMT_STRING，以便指令伺服器可以轉換訊息。 請勿設定 MQFMT_ADMIN，因為這會導致您的指令解譯為 PCF。

ReplyToQ
回覆目的地佇列的名稱。

ReplyToQMgr
如果您想要將回覆傳送至本端佇列管理程式，請將此欄位保留空白。 如果您要將 IBM MQ 指令傳送至遠端佇列管理程式，請在這裡放置其名稱。 您也必須設定正確的佇列及鏈結，如 分散式佇列及叢集中所述。

3. 視需要設定任何其他 MQMD 欄位。 您通常應該對指令使用非持續訊息。
4. 視需要設定任何 *PutMsgOpts* 選項。

如果您指定 MQPMO_SYNCPOINT (預設值)，則必須在 MQPUT 呼叫之後加上同步點呼叫。

使用 MQPUT1 及系統指令輸入佇列
如果您只想在系統指令輸入佇列中放置一則訊息，您可以使用 MQPUT1 呼叫。 此呼叫結合 MQOPEN 的功能，後面接著一個訊息的 MQPUT，後面接著 MQCLOSE，全部在一個呼叫中。 如果您使用此呼叫，請相應地修改參數。 如需詳細資料，請參閱 使用 MQPUT1 呼叫將一則訊息放入佇列。

擷取指令的回覆

指令伺服器會針對它所接收的每一則要求訊息，將回應傳送至回覆佇列。 任何管理應用程式都必須接收並處理回覆訊息。

當指令處理器處理您的指令時，任何回覆訊息都會放入 MQPUT 呼叫中指定的回覆目的地佇列。 指令伺服器會傳送回覆訊息，其持續性與收到的指令訊息相同。

等待回覆

使用 MQGET 呼叫來擷取要求訊息中的回覆。 一個要求訊息可以產生數個回覆訊息。 如需詳細資料，請參閱第 253 頁的『解譯來自指令伺服器的回覆訊息』。

您可以指定 MQGET 呼叫等待產生回覆訊息的時間間隔。 如果您未取得回覆，請使用主題 第 253 頁的『如果您沒有收到回覆』中開始的核對清單。

如果要使用 MQGET 呼叫，請執行下列動作:

1. 設定下列參數:

Hconn
MQCONN 或 MQCONNX 呼叫所傳回的連線控點。

Hobj
MQOPEN 呼叫針對回覆目的地佇列所傳回的物件控點。

Buffer
接收回覆的區域名稱。

BufferLength
接收回覆的緩衝區長度。 至少必須有 80 個位元組。

2. 若要確保僅從您發出的指令取得回應，您必須指定適當的 *MsgId* 及 *CorrelId* 欄位。 這些取決於您在 MQPUT 呼叫中指定的報告選項 MQMD_REPORT:

MQRO_NONE
二進位零， '00 ... 00' (24 個空值)。

MQRO_NEW_MSG_ID
二進位零， '00 ... 00' (24 個空值)。

如果未指定任何這些選項，則這是預設值。

MQRO_PASS_MSG_ID
MQPUT 中的 *MsgId* 。

MQRO_NONE
來自 MQPUT 呼叫的 *MsgId* 。

MQRO_COPY_MSG_ID_TO_CORREL_ID
來自 MQPUT 呼叫的 *MsgId* 。

如果未指定任何這些選項，則這是預設值。

MQRO_PASS_CORREL_ID
來自 MQPUT 呼叫的 *CorrelId* 。

如需報告選項的詳細資料，請參閱 報告選項及訊息旗標。

3. 設定下列 *GetMsgOpts* 欄位:

Options
MQGMO_WAIT

如果您未使用與佇列管理程式相同的字碼頁，請在 MQMD 中適當設定 MQGMO_CONVERT ，並設定 *CodedCharSetId* 。

WaitInterval
對於來自本端佇列管理程式的回覆，請嘗試 5 秒。 以毫秒來編碼，這會變成 5000。 對於來自遠端佇列管理程式的回覆，以及通道控制和狀態指令，請嘗試 30 秒。 以毫秒來編碼，這會變成 30 000。

捨棄的訊息數

如果指令伺服器發現要求訊息無效，它會捨棄此訊息，並將訊息 CSQN205I 寫入指名的回覆目的地佇列。 如果沒有回覆目的地佇列，則會將 CSQN205I 訊息放入無法傳送郵件的佇列中。 此訊息中的回覆碼顯示原始要求訊息無效的原因:

00D5020F	它不是 MQMT_REQUEST 類型。
00D50210	它的長度為零。
00D50212	它超過 32 762 個位元組。
00D50211	它包含所有空白。
00D5483E	需要轉換，但 *Format* 不是 MQFMT_STRING。
其他	請參閱 指令伺服器代碼

指令伺服器回覆訊息描述子

對於任何回覆訊息，會設定下列 MQMD 訊息描述子欄位:

MsgType	MQMT_REPLY
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
Priority	至於您發出的訊息中的 MQMD。

Persistence 至於您發出的訊息中的 MQMD。

CorrelId 取決於 MQPUT 報告選項。

ReplyToQ 無。

指令伺服器會將 MQPMO 結構的 *Options* 欄位設為 MQPMO_NO_SYNCPOINT。 這表示您可以在建立回覆時擷取回覆，而不是在下一個同步點以群組方式擷取。

解譯來自指令伺服器的回覆訊息

IBM MQ 正確處理的每一個要求訊息至少會產生兩個回覆訊息。 每一個回覆訊息都包含單一 IBM MQ 使用者訊息。

回覆的長度視發出的指令而定。 您可以從 DISPLAY NAMELIST 取得的最長回覆，長度最多可達 15000 個位元組。

第一個使用者訊息 CSQN205I 一律包含:

- 回覆計數 (以十進位表示)，您可以用作迴圈中的計數器，以取得其餘回覆。 計數包括此第一則訊息。
- 來自指令前置處理器的回覆碼。
- 原因碼，這是來自指令處理器的原因碼。

此訊息不包含 CPF。

例如:

```
CSQN205I    COUNT=    4, RETURN=0000000C, REASON=00000008
```

COUNT 欄位長度為 8 個位元組，且向右對齊。 它一律從位置 18 開始，即緊接在 'COUNT =' 之後。 RETURN 欄位的字元十六進位長度為 8 個位元組，且緊接在位置 35 的 'RETURN =' 之後。 REASON 欄位長度為 8 個位元組 (十六進位字元)，且緊接在位置 52 的 'REASON =' 之後。

如果 RETURN= 值為 00000000 且 REASON= 值為 00000004，則回覆訊息集不完整。 擷取 CSQN205I 訊息指出的回覆之後，請發出進一步 MQGET 呼叫以等待進一步的回覆集。 下一組回覆中的第一個訊息是 CSQN205I，指出有多少回覆，以及是否還有更多回覆。

如需個別訊息的詳細資料，請參閱 IBM MQ for z/OS 訊息、完成及原因碼 文件。

如果您使用非英文語言特性，則回覆的文字及版面與這裡顯示的不同。 不過，訊息 CSQN205I 中計數與回覆碼的大小與位置相同。

如果您沒有收到回覆

如果您未收到對指令伺服器的要求的回應，則可以採取一系列步驟。

如果您未收到要求訊息的回覆，請執行下列核對清單:

- 指令伺服器是否在執行中?
- *WaitInterval* 夠長嗎?
- 是否正確定義系統指令輸入及回覆目的地佇列?
- 對這些佇列的 MQOPEN 呼叫是否成功?
- 是否同時針對 MQPUT 和 MQGET 呼叫啟用系統指令輸入及回覆目的地佇列?
- 您是否考慮增加佇列的 MAXDEPTH 及 MAXMSGL 屬性?
- 您是否正確使用 *CorrelId* 和 *MsgId* 欄位?
- 佇列管理程式是否仍在執行中?

- 指令建置正確嗎?
- 您的所有遠端鏈結是否都已定義並正確運作?
- 是否已正確定義 MQPUT 呼叫?
- 是否已將回覆目的地佇列定義為暫時動態佇列，而非永久動態佇列? (如果要求訊息持續存在，您必須使用永久動態佇列來進行回覆。)

當指令伺服器產生回覆但無法將它們寫入您指定的回覆目的地佇列時，它會將它們寫入無法傳送郵件的佇列。

使用 MGCRE 傳遞指令

在適當的授權下，應用程式可以使用 z/OS 服務常式向多個佇列管理程式提出要求。

如果您具有正確的授權，則可以透過 MGCRE (SVC 34) z/OS 服務將 IBM MQ 指令從程式傳遞至多個佇列管理程式。 CPF 的值可識別指令所導向的特定佇列管理程式。 如需 CPF 的相關資訊，請參閱 指令安全及指令資源安全的使用者 ID 及 第 231 頁的『發出佇列管理程式指令』。

如果您使用 MGCRE，則可以使用「指令及回應記號 (CART)」來取得指令的直接回應。

指令及其回覆的範例

請使用這個主題作為指令伺服器的一系列指令範例，以及指令伺服器的回應。

以下是一些可以內建在 IBM MQ 訊息中的指令範例，以及作為回覆的使用者訊息。 除非另有說明，否則回覆的每一行都是個別訊息。

- 來自 DEFINE 指令的訊息
- DELETE 指令的訊息
- DISPLAY 指令的訊息
- 來自具有 CMDSCOPE 之指令的訊息
- 來自使用 CMDSCOPE 產生指令之指令的訊息

來自 DEFINE 指令的訊息

the following command:

```
DEFINE QLOCAL(Q1)
```

會產生下列訊息:

```
CSQN205I    COUNT=    2, RETURN=00000000, REASON=00000000
CSQ9022I +CSQ1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
```

這些回覆訊息會在正常完成時產生。

DELETE 指令的訊息

the following command:

```
DELETE QLOCAL(Q2)
```

會產生下列訊息:

```
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
```

這些訊息指出稱為 Q2 的本端佇列不存在。

DISPLAY 指令的訊息

下列範例顯示部分 DISPLAY 指令的回覆。

找出無法傳送郵件的佇列名稱

如果您想要找出佇列管理程式的無法傳送郵件的佇列名稱，請從應用程式發出下列指令:

```
DISPLAY QMGR DEADQ
```

會傳回下列三則使用者訊息，您可以從中擷取必要的名稱:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM409I +CSQ1 QMNAME(CSQ1) DEADQ(SYSTEM.DEAD.QUEUE          )
CSQ9022I +CSQ1 CSQMDRTS ' DISPLAY QMGR' NORMAL COMPLETION
```

來自 DISPLAY QUEUE 指令的訊息

下列範例顯示指令的結果如何取決於該指令中指定的屬性。

範例 1

您可以使用下列指令來定義本端佇列:

```
DEFINE QLOCAL(Q1) DESCR('A sample queue') GET(ENABLED) SHARE
```

如果您從應用程式發出下列指令:

```
DISPLAY QUEUE(Q1) SHARE GET DESCR
```

會傳回下列三則使用者訊息:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM401I +CSQ1 QUEUE(Q1                       ) TYPE(
QLOCAL ) QSGDISP(QMGR   )
DESCR(A sample queue
) SHARE  GET(ENABLED  )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

註: 這裡顯示第二個訊息 CSQM401I，佔用四行。

範例 2

兩個佇列的名稱以字母 A 開頭:

- A1 是本端佇列，其 PUT 屬性設為 DISABLED。
- A2 是遠端佇列，其 PUT 屬性設為 ENABLED。

如果您從應用程式發出下列指令:

```
DISPLAY QUEUE(A*) PUT
```

會傳回下列四則使用者訊息:

```
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 CSQMDMSG ' DISPLAY QUEUE' NORMAL COMPLETION
```

註: 這裡顯示第二個及第三個訊息 (CSQM401I 及 CSQM406I) ，佔用三行及兩行。

來自 DISPLAY NAMELIST 指令的訊息

您可以使用下列指令來定義名稱清單:

```
DEFINE NAMELIST(N1) NAMES(Q1,SAMPLE_QUEUE)
```

如果您從應用程式發出下列指令:

```
DISPLAY NAMELIST(N1) NAMES NAMCOUNT
```

會傳回下列三個使用者訊息:

```
CSQN205I    COUNT=    3, RETURN=00000000, REASON=00000000
CSQM407I +CSQ1 NAMELIST(N1                       ) QS
GDISP(QMGR   ) NAMCOUNT(     2) NAMES(Q1
,SAMPLE_QUEUE                 )
CSQ9022I +CSQ1 CSQMDMSG ' DISPLAY NAMELIST' NORMAL COMPLETION
```

註: 這裡顯示第二則訊息 CSQM407I，佔用三行。

來自具有 CMDSCOPE 之指令的訊息

下列範例顯示已使用 CMDSCOPE 屬性輸入之指令的回覆。

來自 ALTER PROCESS 指令的訊息

the following command:

```
ALT PRO(V4) CMDSCOPE(*)
```

會產生下列訊息:

```
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
```

這些訊息告訴您指令已在佇列管理程式 MQ25 上輸入，並傳送至兩個佇列管理程式 (MQ25 及 MQ26)。 指令在 MQ25 上順利完成，但程序定義在 MQ26 上不存在，因此指令在該佇列管理程式上失敗。

DISPLAY PROCESS 指令的訊息

the following command:

```
DIS PRO(V*) CMDSCOPE(*)
```

會產生下列訊息:

```
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
```

這些訊息告訴您指令已在佇列管理程式 MQ25 上輸入，並傳送至兩個佇列管理程式 (MQ25 及 MQ26)。 會顯示每一個佇列管理程式上名稱以字母 V 開頭的所有處理程序的相關資訊。

DISPLAY CHSTATUS 指令的訊息

the following command:

```
DIS CHS(VT) CMDSCOPE(*)
```

會產生下列訊息:

```
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
```

這些訊息告訴您指令已在佇列管理程式 MQ25 上輸入，並傳送至兩個佇列管理程式 (MQ25 及 MQ26)。 會顯示每一個佇列管理程式上通道狀態的相關資訊。

STOP CHANNEL 指令的訊息

the following command:

```
STOP CHL(VT) CMDSCOPE(*)
```

會產生下列訊息:

```
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
```

這些訊息告訴您指令已在佇列管理程式 MQ25 上輸入，並傳送至兩個佇列管理程式 (MQ25 及 MQ26)。 已在每一個佇列管理程式上停止通道 VT。

來自使用 CMDSCOPE 產生指令之指令的訊息

the following command:

```
DEF PRO(V2) QSGDISP(GROUP)
```

會產生下列訊息:

```
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
```

這些訊息告訴您已在佇列管理程式 MQ25 上輸入指令。 在共用儲存庫上建立物件時，會產生另一個指令，並傳送至佇列共用群組 (MQ25 及 MQ26) 中的所有作用中佇列管理程式。

z/OS 在 z/OS 上管理 IBM MQ 資源

請使用本主題中的鏈結，以瞭解如何管理 IBM MQ for z/OS 所使用的資源，例如，管理日誌檔、資料集、頁集、緩衝池及連結機能結構。

請使用下列鏈結，以取得您在使用 IBM MQ for z/OS 時可能必須完成的不同管理作業的詳細資料:

- 第 259 頁的『管理日誌』
- 第 266 頁的『管理引導資料集 (BSDS)』
- 第 273 頁的『管理頁面集』
- 第 278 頁的『如何備份及回復頁集』
- 第 281 頁的『如何使用 CSQUTIL 備份及還原佇列』
- 第 281 頁的『管理緩衝池』
- 第 282 頁的『管理佇列共用群組及共用佇列』

相關概念

第 221 頁的『管理 IBM MQ for z/OS』
管理佇列管理程式及相關聯資源包括您經常執行以啟動及管理那些資源的作業。 選擇您偏好用來管理佇列管理程式及相關聯資源的方法。

第 221 頁的『對 IBM MQ for z/OS 發出指令』
您可以在批次或互動模式下使用 IBM MQ Script 指令 (MQSC) 來控制佇列管理程式。

第 289 頁的『回復及重新啟動』
請利用這個主題來瞭解 IBM MQ 所使用的回復和重新啟動機制。

相關參考

第 228 頁的『IBM MQ for z/OS 公用程式』
IBM MQ for z/OS 提供一組公用程式，您可以用來協助進行系統管理。

相關資訊

IBM MQ for z/OS 概念
在 z/OS 上規劃 IBM MQ 環境
配置 z/OS
可程式化指令格式參照
MQSC 參照
使用 IBM MQ for z/OS 公用程式

管理日誌

請利用這個主題來瞭解如何管理 IBM MQ 日誌檔，包括日誌保存程序、使用日誌記錄壓縮、日誌記錄回復及列印日誌記錄。

本主題說明管理 IBM MQ 日誌所涉及的作業。 它包含下列區段:

使用 *ARCHIVE LOG* 指令保存日誌

每當需要使用 ARCHIVE LOG 指令時，授權操作員可以保存現行 IBM MQ 作用中日誌資料集。

當您發出 ARCHIVE LOG 指令時， IBM MQ 會截斷現行作用中日誌資料集，然後執行非同步卸載處理程序，並以卸載處理程序的記錄來更新 BSDS。

ARCHIVE LOG 指令具有 MODE (QUIESCE) 選項。 使用此選項， IBM MQ 工作及使用者會在確定點之後靜止，且在卸載之前，會在現行作用中日誌中擷取產生的一致性點。

在規劃離站回復的備份策略時，請考慮使用 MODE (QUIESCE) 選項。 它會建立全系統一致性點，當在回復期間將保存日誌與最新備份頁集副本搭配使用時，可將資料不一致的數目降至最低。 例如：

```
ARCHIVE LOG MODE(QUIESCE)
```

如果您發出 ARCHIVE LOG 指令但未指定 TIME 參數，則靜止時段會預設為 CSQ6ARVP 巨集的 QUIESCE 參數值。 如果完成 ARCHIVE LOG MODE (QUIESCE) 所需的時間小於指定的時間，指令會順利完成; 否則，指令會在時段到期時失敗。 您可以使用 TIME 選項來明確指定時段，例如：

```
ARCHIVE LOG MODE(QUIESCE) TIME(60)
```

此指令指定在進行 ARCHIVE LOG 處理之前最多 60 秒的靜止期間。

請注意: 當時間很重要時，使用 TIME 選項會大幅中斷所有使用 IBM MQ 資源之工作及使用者的 IBM MQ 可用性。

依預設，會從您提交指令時開始非同步處理指令。 (若要與其他 IBM MQ 指令同步處理指令，請使用 WAIT (YES) 選項與 QUIESCE ，但請注意 z/OS 主控台在整個 QUIESCE 期間已從 IBM MQ 指令輸入鎖定。)

在靜止期間:

- 佇列管理程式上的工作及使用者可以進行確定處理，但如果在確定之後嘗試更新任何 IBM MQ 資源，則會暫停。
- 只讀取資料的工作及使用者可能會受到影響，因為他們可能正在等待已暫停的工作或使用者所保留的鎖定。
- 新作業可以啟動，但無法更新資料。

DISPLAY LOG 指令的輸出會使用訊息 CSQV400I 來指出靜止生效。 例如：

```
CSQJ322I +CSQ1 DISPLAY LOG report ...
Parameter   Initial value          SET value
----------- ---------------------- ----------------------
INBUFF      60
OUTBUFF     4000
MAXRTU      2
MAXARCH     2
TWOACTV     YES
TWOARCH     YES
TWOBSDS     YES
OFFLOAD     YES
WRTHRSH     20
DEALLCT     0
End of LOG report
CSQJ370I +CSQ1 LOG status report ...
Copy %Full DSName
1      68  VICY.CSQ1.LOGCOPY1.DS01
2      68  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
```

當所有更新項目都靜止時， BSDS 中的靜止歷程記錄會更新為截斷作用中日誌資料集的日期和時間，以及現行作用中日誌資料集中前次寫入 RBA 的日期和時間。 IBM MQ 會截斷現行作用中日誌資料集，切換至下一個可用的作用中日誌資料集，並發出訊息 CSQJ311I，指出卸載處理程序已啟動。

如果在靜止期間到期之前無法靜止更新項目， IBM MQ 會發出訊息 CSQJ317I，且 ARCHIVE LOG 處理程序會終止。 現行作用中日誌資料集不會截斷，也不會切換至下一個可用的日誌資料集，且卸載處理程序不會啟動。

不論靜止是否成功，都會回復所有已暫停的使用者及工作，且 IBM MQ 會發出訊息 CSQJ312I，指出靜止已結束並回復更新活動。

當現行作用中日誌是最後一個可用的作用中日誌資料集時，如果發出 ARCHIVE LOG，則不會處理指令，且 IBM MQ 會發出下列訊息:

```
CSQJ319I - csect-name CURRENT ACTIVE LOG DATA SET IS THE LAST
AVAILABLE ACTIVE LOG DATA SET. ARCHIVE LOG PROCESSING
WILL BE TERMINATED
```

如果在另一個 ARCHIVE LOG 指令已在進行中時發出 ARCHIVE LOG，則不會處理新指令，且 IBM MQ 會發出下列訊息:

```
CSQJ318I - ARCHIVE LOG COMMAND ALREADY IN PROGRESS
```

如需在保存期間發出的訊息的相關資訊，請參閱 IBM MQ for z/OS 的訊息。

在失敗之後重新啟動日誌保存處理程序

如果在日誌保存處理程序期間發生問題 (例如，配置或磁帶裝載的問題)，則可能會暫停保存作用中日誌。 您可以使用 ARCHIVE LOG CANCEL OFFLOAD 指令來取消保存程序並重新啟動它。 這個指令會取消目前進行中的任何卸載處理程序，並重新啟動保存處理程序。 它會從尚未保存的最舊日誌資料集開始，並繼續進行所有需要卸載的作用中日誌資料集。 已暫停的任何日誌保存作業都會重新啟動。

只有在您確定現行日誌保存作業不再運作，或您想要重新啟動先前失敗的嘗試時，才使用這個指令。 這是因為指令可能會導致卸載作業異常終止，這可能會導致傾出。

控制保存及記載

您可以使用 CSQ6LOGP、 CSQ6ARVP 及 CSQ6SYSP 巨集來控制壓縮、列印、保存、回復及記載。 請注意，專用物件的變更只會記載在 IBM MQ 日誌中。 也會記載 GROUP 物件 (例如共用入埠通道) 的變更，因為定義會在群組周圍傳播並在本端保留。

當自訂佇列管理程式時，保存及記載的許多層面是由使用系統參數模組的 CSQ6LOGP、 CSQ6ARVP 及 CSQ6SYSP 巨集所設定的參數所控制。 如需這些巨集的詳細資料，請參閱 作業 17: 自訂系統參數模組 。

當佇列管理程式使用 IBM MQ MQSC SET LOG、SET SYSTEM 及 SET ARCHIVE 指令執行時，可以變更其中部分參數。 它們顯示在 第 262 頁的表 18 中:

表 *18:* 在佇列管理程式執行時可以變更的保存及記載參數	
設定指令	**參數**
日誌	WRTHRSH、MAXARCH、DEALLCT、MAXRTU、COMPLOG
保存式	全部
系統	LOGLOAD

您可以使用 MQSC DISPLAY LOG、 DISPLAY ARCHIVE 及 DISPLAY SYSTEM 指令來顯示所有參數的設定。 這些指令也會顯示保存及記載的相關狀態資訊。

控制日誌壓縮

您可以使用下列任一項來啟用及停用日誌記錄的壓縮:

- MQSC 中的 SET 和 DISPLAY LOG 指令; 請參閱 MQSC 指令
- 正在呼叫 PCF 介面。 請參閱 第 9 頁的『可程式指令格式簡介』
- 在系統參數模組中使用 CSQ6LOGP 巨集; 請參閱 使用 CSQ6LOGP

列印日誌記錄

您可以使用 CSQ1LOGP 公用程式來擷取及列印日誌記錄。 如需指示，請參閱 日誌列印公用程式。

回復日誌

通常，您不需要備份及還原 IBM MQ 日誌，尤其是當您使用雙重記載時。 不過，在極少數情況下 (例如日誌上的 I/O 錯誤) ，您可能需要回復日誌。 使用「存取方法服務」來刪除並重新定義資料集，然後將對應的雙重日誌複製到其中。

捨棄保存日誌資料集

您可以捨棄保存日誌資料集，並選擇自動或手動捨棄日誌。

您必須保留足夠的日誌資料，才能執行工作單元回復、頁集媒體回復 (如果頁集遺失) 或 CF 結構媒體回復 (如果 CF 結構遺失)。 請勿捨棄回復可能需要的保存日誌資料集; 如果您捨棄這些保存日誌資料集，則可能無法執行必要的回復作業。

如果您已確認可以捨棄保存日誌資料集，則可以使用下列其中一種方式來執行此動作:

- 自動刪除保存日誌資料集
- 手動刪除保存日誌資料集

自動刪除保存日誌資料集

您可以使用 DASD 或磁帶管理系統來自動刪除保存日誌資料集。 IBM MQ 保存日誌資料集的保留期間由 CSQ6ARVP 安裝巨集中的保留期間欄位 ARCRETN 指定 (如需相關資訊，請參閱 使用 CSQ6ARVP)。

保留期的預設值指定將保存日誌保留 9999 天 (上限)。 **您可以變更保留期間，但必須確保您可以容納已針對**規劃的備份週期數。

建立保存日誌資料集時， IBM MQ 會使用保留期間值作為 JCL 參數 RETPD 的值。

此 IBM MQ 參數可以置換 MVS/DFP 儲存體管理子系統 (SMS) 所設定的保留期間。 一般而言，保留期間會設為 IBM MQ 或 SMS 所指定的較小值。 儲存體管理者和 IBM MQ 管理者必須同意適用於 IBM MQ 的保留期間值。

註: IBM MQ 沒有自動化方法可從 BSDS 中刪除保存日誌資料集的相關資訊，因為部分磁帶管理系統提供保留期間的外部手動置換。 因此，保存日誌資料集的相關資訊在資料集保留期已過期且磁帶管理系統已刮回資料集之後很長時間仍可以在 BSDS 中。 相反地，可能已超出保存日誌資料集數目上限，且在資料集達到其到期日之前，可能已捨棄 BSDS 中的資料。

如果自動刪除保存日誌資料集，請記住作業不會更新 BSDS 中的保存日誌清單。 您可以使用變更日誌庫存公用程式來更新 BSDS ，如 第 267 頁的『變更 BSDS』中所述。 更新並不重要。 記錄舊的保存日誌會浪費 BSDS 中的空間，但不會造成其他傷害。

手動刪除保存日誌資料集

您必須將所有日誌記錄保留到訊息 CSQI024I 及 CSQI025I 中所識別的最低 RBA 為止。 此 RBA 是使用您在使用 方法 1: 完整備份建立回復點時發出的 DISPLAY USAGE 指令來取得。

請先閱讀 建立非共用資源的回復點，再捨棄任何日誌。

尋找並捨棄保存日誌資料集

建立回復所需的日誌 RBA 下限之後，您可以執行下列程序來尋找只包含較早日誌記錄的保存日誌資料集:

1. 使用列印日誌對映公用程式來列印 BSDS 的內容。 如需輸出的範例，請參閱 列印日誌對映公用程式。
2. 尋找標題為 "ARCHIVE LOG COPY n DATA SETS" 的輸出區段。 如果您使用雙重記載，則有兩個區段。 標籤為 STARTRBA 和 ENDRBA 的直欄會顯示每一個磁區中包含的 RBA 範圍。 尋找其範圍包括您找到之訊息 CSQI024I 及 CSQI025I 的 RBA 下限的磁區。 這些是您需要保留的最早磁區。 如果您使用雙重記載，則有兩個這類磁區。

   如果沒有磁區具有適當的範圍，則適用下列其中一種情況:

   - 尚未保存最小 RBA ，您可以捨棄所有保存日誌磁區。
   - 當磁區數超出 CSQ6LOGP 巨集的 MAXARCH 參數所容許的數目時， BSDS 中折返的保存日誌磁區清單。 如果 BSDS 未登錄保存日誌磁區，則該磁區無法用於回復。 因此，請考量將現有磁區的相關資訊新增至 BSDS。 如需相關指示，請參閱第 269 頁的『保存日誌的變更』。

     也請考量增加 MAXARCH 的值。 如需相關資訊，請參閱 使用 CSQ6LOGP。
3. 刪除 ENDRBA 值小於您要保留之最早磁區的 STARTRBA 值的任何保存日誌資料集或磁區。 如果您使用雙重記載，請刪除這兩個副本。

   因為 BSDS 項目會折返，所以 BSDS 保存日誌區段中的前幾個項目可能比底端的項目更新。 查看日期和時間的組合，並比較其年齡。 請勿假設您可以捨棄 上方 包含最低 LOGRBA 之保存日誌項目的所有項目。

   刪除資料集。 如果保存檔是在磁帶上，請清除磁帶。 如果它們位於 DASD 上，請執行 z/OS 公用程式來刪除每一個資料集。 然後，如果您想要 BSDS 只列出現有的保存磁區，請使用變更日誌庫存公用程式 (CSQJU003) 來刪除捨棄磁區的項目。 如需範例，請參閱 第 269 頁的『保存日誌的變更』。

日誌分流的影響

長時間執行交易可能會導致工作日誌記錄的單元跨越日誌資料集。 IBM MQ 會使用日誌分流來處理此實務範例，這項技術會移動日誌記錄以最佳化所保留日誌資料的數量，以及佇列管理程式重新啟動時間。

當工作單元被視為較長時，每一個日誌記錄的表示法會進一步寫下日誌。 這稱為 日誌分流。 它在 日誌檔中有更完整的說明。

在失敗之後，佇列管理程式會使用這些延遲的日誌記錄，而非原始記錄，以確保工作單元完整性。 這有兩個好處:

- 減少工作單元協調必須保留的日誌資料量
- 必須在佇列管理程式重新啟動時遍訪較少日誌資料，因此佇列管理程式會更快速重新啟動

延遲的日誌記錄未包含足夠的媒體回復作業資訊。

日誌中保留的資料有兩個不同用途: 媒體回復和工作單元協調。 如果發生會影響 CF 結構或頁集的媒體故障，佇列管理程式可以透過還原先前的副本並使用日誌中包含的資料更新此副本，將媒體回復至故障點。 在工作單元中執行的持續性活動會記錄在日誌上，以便在發生失敗時，可以取消或在變更的資源上回復鎖定。 這兩個元素會影響您為了啟用佇列管理程式回復而需要保留的日誌資料數量。

對於媒體回復，您必須保留足夠的日誌資料，才能至少從最新的媒體副本執行媒體回復，並且能夠回復。 (您的網站可能會規定從較舊備份回復的能力。) 針對工作單元完整性，您必須保留最舊的進行中或不確定工作單元的日誌資料。

為了協助您管理系統，佇列管理程式會在每一個日誌保存檔中偵測舊的工作單元，並在訊息 CSQJ160 和 CSQJ161 中報告它們。 內部作業會讀取這些舊工作單元的工作日誌資訊，並以更簡潔的形式將它重新寫入日誌中的現行位置。 訊息 CSQR026 指出何時發生此情況。 MQSC 指令 DISPLAY USAGE TYPE (DATASET) 也可以協助您管理日誌資料的保留。 指令會報告 3 項回復資訊:

1. 工作單元回復必須保留多少日誌
2. 必須保留多少日誌才能進行頁集的媒體回復
3. 對於佇列共用群組中的佇列管理程式，必須保留多少日誌才能進行 CF 結構的媒體回復

對於其中每一個，會嘗試將所需的最舊日誌資料對映至資料集。 當新的工作單元開始和停止時，我們預期 (1) 會移至日誌中較新的位置。 如果未移動，長時間執行的 UOW 訊息會警告您發生問題。 (2) 如果現在要關閉並重新啟動佇列管理程式，則與頁集媒體回復相關。 它不知道您前次備份頁面集的時間，或如果頁面集失敗，您可能必須使用哪一個備份。 在檢查點處理程序期間，它通常會移至日誌中較新的位置，因為緩衝池中保留的變更會寫入頁集。 在 (3) 中，佇列管理程式確實知道在此佇列管理程式或佇列共用群組中其他佇列管理程式上所執行的 CF 結構備份。 不過， CF 結構回復需要合併佇列共用群組中自前次備份以來與 CF 結構互動的所有佇列管理程式的日誌資料。 這表示日誌資料由日誌記錄序號 (或 LRSN) 識別，這是時間戳記型，因此適用於整個佇列共用群組，而不是在佇列共用群組中不同佇列管理程式上不同的 RBA。 它通常會移至日誌中較新的位置，因為 BACKUP CFSTRUCT 指令是在佇列共用群組中的這個或其他佇列管理程式上執行。

重設佇列管理程式的日誌

請利用這個主題來瞭解如何重設佇列管理程式的日誌。

您不得容許佇列管理程式日誌 RBA 從日誌 RBA 範圍結尾折返至 0 ，因為這會導致佇列管理程式中斷，且所有持續資料將變成無法回復。 日誌 RBA 的結尾是 FFFFFFFFFFFFFFFFFFFF 值 (如果使用 6 個位元組的 RBA) 或 FFFFFFFFFFFFFFFFFFFFFFFF (如果使用 8 個位元組的 RBA)。

佇列管理程式會發出訊息 CSQI045I、 CSQI046E、 CSQI047E、 CSQJ031D 及 CSQJ032E ，以指出使用的日誌範圍很重要，您應該計劃採取動作來避免意外中斷執行。

當 RBA 值達到 FFF800000000 (如果使用 6 位元組日誌 RBA) 或 FFFFFFC000000000 (如果使用 8 位元組日誌 RBA) 時，佇列管理程式會終止，原因碼為 00D10257 。

如果 6 位元組日誌 RBA 正在使用中，請遵循 實作較大的日誌相對位元組位址中說明的處理程序，考慮將佇列管理程式轉換為使用 8 位元組日誌 RBA ，而不是重設佇列管理程式的日誌。 將佇列管理程式轉換為使用 8 位元組日誌 RBA 所需要的中斷時間比重設日誌所需的中斷時間更短，並增加您必須重設日誌之前的時段。

在佇列管理程式起始設定期間發出的訊息 CSQJ034I 指出已配置佇列管理程式的日誌 RBA 範圍結束，可用來判斷 6 位元組或 8 位元組日誌 RBA 是否在使用中。

重設佇列管理程式日誌所遵循的程序如下:

1. 解決任何未解決的工作單元。 在佇列管理程式啟動時，未解決的工作單元數目會顯示在訊息 CSQR005I 中，作為 INDOUBT 計數。 在每一個檢查點，以及在佇列管理程式關閉時，佇列管理程式會自動發出指令

DISPLAY CONN(*) TYPE(CONN) ALL WHERE(UOWSTATE EQ UNRESOLVED)，提供未解決工作單元的相關資訊。

如需解決回復單元的相關資訊，請參閱 如何解決不確定的回復單元 。 最終手段是使用 **RESOLVE INDOUBT** MQSC 指令來手動解決不確定的回復單元。

2. 完全關閉佇列管理程式。

   您可以使用 **STOP QMGR** 或 **STOP QMGR MODE(FORCE)**，因為這兩個指令都會將任何已變更的頁面從緩衝池清除至頁集。

3. 如果佇列管理程式是佇列共用群組的一部分，請針對佇列共用群組中的所有結構，在其他佇列管理程式上執行 CFSTRUCT 備份。 這可確保最近的備份不在此佇列管理程式的日誌中，且 CFSTRUCT 回復不需要此佇列管理程式的日誌。
4. 使用 CSQJU003 定義新的日誌和 BSDS (如需使用變更日誌庫存公用程式的相關資訊，請參閱 變更日誌庫存公用程式)。
5. 針對此佇列管理程式的所有頁集執行 **CSQUTIL RESETPAGE** (如需使用此功能的相關資訊，請參閱 複製頁面並重設日誌)。 請注意，頁集 RBA 可以獨立重設，因此可以提交多個並行工作 (例如，每個頁集一個)，以減少此步驟的經歷時間。
6. 重新啟動佇列管理程式

相關概念

第 265 頁的『實作較大的日誌相對位元組位址』

舊版 IBM MQ for z/OS 使用 6 個位元組的日誌 RBA 來識別資料在日誌內的位置。 在 IBM MQ 8.0 中，日誌 RBA 可以有 8 個位元組的長度，增加您必須重設日誌之前的時段。

實作較大的日誌相對位元組位址

舊版 IBM MQ for z/OS 使用 6 個位元組的日誌 RBA 來識別資料在日誌內的位置。 在 IBM MQ 8.0 中，日誌 RBA 可以有 8 個位元組的長度，增加您必須重設日誌之前的時段。

這個新特性需要明確啟用。 如需規劃啟用 8 位元組日誌 RBA 時的考量，請參閱 規劃增加可定址日誌範圍上限 。

依照顯示的順序執行下列指示，以在單一 IBM MQ for z/OS 佇列管理程式上啟用 8 個位元組的日誌 RBA:

1. 使用 OPMODE 啟用 IBM MQ 8.0 新功能。

   對於佇列共用群組中的佇列管理程式，您不需要採取佇列共用群組中斷總計。 您可以依序停止每一個佇列管理程式，啟用 OPMODE = (NEWFUNC,800) 並重新啟動它。

   一旦佇列共用群組中的所有佇列管理程式都以 OPMODE (NEWFUNC,800) 執行，請針對佇列共用群組中的每一個佇列管理程式執行下列步驟，直到所有佇列管理程式都以新的 BSDS 執行為止。

2. 將具有類似屬性的新 BSDS 資料集配置給現行 BSDS。 您可以自訂範例 CSQ4BSDS 並刪除任何不相關的陳述式，也可以使用現有的 JCL，但將 BSDS 名稱變更為類似 `++HLQ++.NEW.BSDS01` 的名稱。

   附註:

   a. 請檢查新 BSDS 的屬性。 唯一可能變更的屬性是 BSDS 的大小。

   b. 新的 BSDS 包含現行 BSDS 的更多資料，因此您必須確保為新資料集配置足夠的可用空間。 如需定義新 BSDS 時的建議值，請參閱 規劃記載環境及相關聯的主題。

3. 完全關閉佇列管理程式。
4. 執行 BSDS 轉換公用程式 (CSQJUCNV)，將現有的 BSDS 轉換成新的 BSDS 資料集。 這通常需要幾秒鐘才能執行。

   在此處理程序期間將不會變更您現有的 BSDS，且在轉換失敗的情況下，您可以使用該 BSDS 來起始設定佇列管理程式。

5. 將現行 BSDS 重新命名為舊的 BSDS，並將新的 BSDS 重新命名為現行 BSDS，以便在您下次重新啟動佇列管理程式時使用新的資料集。 您可以使用 DFSMS Access Method Services ALTER 指令，例如:

```
ALTER '++HLQ++.BSDS01' NEWNAME('++HLQ++.OLD.BSDS01')
ALTER '++HLQ++.NEW.BSDS01' NEWNAME('++HLQ++.BSDS01')
```

請確定您也發出指令來重新命名 VSAM 叢集的資料和索引部分。

6. 重新啟動佇列管理程式。 它應該在使用 6 個位元組日誌 RBA 時所執行的相同時間量啟動。

   如果佇列管理程式由於無法存取已轉換的 BSDS 而無法順利重新啟動，請嘗試識別失敗的原因，解決問題並重試作業。 必要的話，請聯絡 IBM 支援中心，以取得協助。

   必要的話，可以透過下列方式在此時取消變更:

   a. 將現行 BSDS 重新命名為新的 BSDS。

   b. 將舊 BSDS 重新命名為現行 BSDS。

   c. 正在重新啟動佇列管理程式。

   使用已轉換的 BSDS 順利重新啟動佇列管理程式之後，請勿嘗試使用舊的 BSDS 來啟動佇列管理程式。

7. 在佇列管理程式起始設定期間發出訊息 CSQJ034I ，以指出所配置佇列管理程式的日誌 RBA 結束。 確認所顯示日誌 RBA 範圍的結尾是 FFFFFFFFFFFFFFFFFFFF。 這指出正在使用 8 個位元組的日誌 RBA。

註: 若要在新的 IBM MQ 8.0 佇列管理程式上啟用 8 個位元組的日誌 RBA ，在第一次啟動之前，您必須先建立空的第 1 版格式 BSDS ，並使用該格式作為 BSDS 轉換公用程式的輸入，以產生第 2 版格式 BSDS。 如需如何執行此處理程序的相關資訊，請參閱 建立引導及日誌資料集 。

相關資訊

規劃增加可定址日誌範圍上限

較大的日誌相對位元組位址

BSDS 轉換公用程式 (CSQJUCNV)

管理引導資料集 (BSDS)

引導資料集 (BSDS) 用來參照日誌資料集及日誌記錄。 請利用這個主題來瞭解如何檢查、變更及回復 BSDS。

如需相關資訊，請參閱 引導資料集。

這個主題說明管理引導資料集所涉及的作業。 它包含下列區段:

- 第 266 頁的『找出 BSDS 包含的內容』
- 第 267 頁的『變更 BSDS』
- 第 271 頁的『回復 BSDS』

找出 *BSDS* 包含的內容

您可以使用列印日誌對映公用程式 (CSQJU004) 來檢查 BSDS 的內容。

列印日誌對映公用程式 (CSQJU004) 是一個批次公用程式，列出儲存在 BSDS 中的資訊。 如需執行它的指示，請參閱 列印日誌對映公用程式。

BSDS 包含:

- 時間戳記
- 作用中日誌資料集狀態

BSDS 中的時間戳記

列印日誌對映公用程式的輸出會顯示時間戳記，用來記錄儲存在 BSDS 中的各種系統事件的日期和時間。

下列時間戳記包含在報告的標頭區段中:

系統時間戳記
反映前次更新 BSDS 的日期和時間。 在下列情況下，可以更新 BSDS 時間戳記:

- 即會啟動佇列管理程式。

- 在日誌寫入活動期間達到寫入臨界值。 視您指定的輸出緩衝區數目及系統活動速率而定， BSDS 可能每秒更新數次，或數秒、分鐘甚至數小時都不會更新。 如需寫入臨界值的詳細資料，請參閱使用 CSQ6LOGP 中 CSQ6LOGP 巨集的 WRTHRSH 參數。
- 由於發生錯誤， IBM MQ 從其正常雙重 BSDS 模式進入單一 BSDS 模式。 當取得、插入、指向、更新或刪除 BSDS 記錄的要求不成功時，可能會發生這種情況。 發生此錯誤時， IBM MQ 會更新剩餘 BSDS 中的時間戳記，以強制時間戳記與已停用 BSDS 不符。

公用程式時間戳記
變更日誌庫存公用程式 (CSQJU003) 變更 BSDS 內容的日期和時間。

報告的作用中和保存日誌資料集部分包含下列時間戳記:

作用中日誌日期
在 BSDS 中建立作用中日誌項目的日期，即完成 CSQJU003 NEWLOG 時。

作用中日誌時間
在 BSDS 中建立作用中日誌項目的時間，即完成 CSQJU003 NEWLOG 時。

保存日誌日期
在 BSDS 中建立保存日誌項目的日期，即完成 CSQJU003 NEWLOG 或完成保存本身的時間。

保存日誌時間
在 BSDS 中建立保存日誌項目的時間，即完成 CSQJU003 NEWLOG 或完成保存本身的時間。

作用中日誌資料集狀態

BSDS 會將作用中日誌資料集的狀態記錄為下列其中一項:

新建
已定義資料集，但從未由 IBM MQ 使用，或日誌已截斷至第一次使用資料集之前的點。 在任一情況下，資料集啟動及結束 RBA 值都會重設為零。

可重複使用
資料集已定義但從未由 IBM MQ 使用，或資料集已卸載。 在列印日誌對映輸出中，最後一個可重複使用資料集的開始 RBA 值等於最後一個保存日誌資料集的開始 RBA 值。

不可重複使用
資料集包含尚未卸載的記錄。

STOPPED
卸載處理器在讀取記錄時發現錯誤，且無法從作用中日誌的另一個副本取得該記錄。

已截斷
您可以:

- 發生 I/O 錯誤， IBM MQ 已停止寫入此資料集。 會從啟動 RBA 開始卸載作用中日誌資料集，並繼續到截斷的作用中日誌資料集中最後一個有效記錄區段。 最後一個有效記錄區段的 RBA 低於作用中日誌資料集的結束 RBA。 記載會切換至下一個可用的作用中日誌資料集，並繼續不中斷。

or

- 已呼叫 ARCHIVE LOG 函數，已截斷作用中日誌。

狀態會出現在列印日誌對映公用程式的輸出中。

變更 *BSDS*
您不需要採取特殊步驟，即可使用記載事件的記錄來更新 BSDS ，因為 IBM MQ 會自動這麼做。

不過，如果您執行下列任何動作，則可能想要變更 BSDS:

- 新增更多作用中日誌資料集。
- 例如，提供較大的作用中日誌配置時，將作用中日誌資料集複製到新配置的資料集。
- 將日誌資料集移至其他裝置。
- 回復損壞的 BSDS。
- 捨棄過期的保存日誌資料集。

您可以執行變更日誌庫存公用程式 (CSQJU003) 來變更 BSDS。 只有在佇列管理程式非作用中，或您可能得到不一致的結果時，才執行此公用程式。 公用程式的動作是由 SYSIN 資料集中的陳述式所控制。 本節顯示數個範例。 如需完整指示，請參閱 變更日誌庫存公用程式。

只有在佇列管理程式非作用中時，您才能複製作用中日誌資料集，因為在佇列管理程式啟動時， IBM MQ 會將作用中日誌資料集配置成專用 (DISP = OLD)。

作用中日誌的變更
請利用這個主題來瞭解如何使用 BSDS 來變更作用中日誌。

您可以使用變更日誌公用程式，在 BSDS 中新增、刪除及記錄作用中日誌的項目。 這裡只顯示範例; 將顯示的資料集名稱取代為您要使用的資料集名稱。 如需公用程式的詳細資料，請參閱 變更日誌庫存公用程式。

如需相關資訊，請參閱下列各節:

- 將記錄項目新增至 BSDS
- 從 BSDS 刪除作用中日誌資料集的相關資訊
- 記錄 BSDS 中日誌資料集的相關資訊
- 增加作用中日誌的大小
- 使用 CSQJUFMT

將記錄項目新增至 BSDS

如果作用中日誌已標示為「已停止」，則不會重複用於記載; 不過，它會繼續用於讀取。 使用存取方法服務來定義新的作用中日誌資料集，然後使用變更日誌庫存公用程式在 BSDS 中登錄新的資料集。 例如，使用:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY1
NEWLOG DSNAME=MQM111.LOGCOPY2.DS10,COPY2
```

如果您要將舊作用中日誌資料集的內容複製到新的內容集，您也可以提供 RBA 範圍，以及 NEWLOG 函數的開始和結束時間戳記。

從 BSDS 刪除作用中日誌資料集的相關資訊

若要從 BSDS 刪除作用中日誌資料集的相關資訊，您可以使用:

```
DELETE DSNAME=MQM111.LOGCOPY1.DS99
DELETE DSNAME=MQM111.LOGCOPY2.DS99
```

記錄 BSDS 中日誌資料集的相關資訊

若要記錄 BSDS 中現有作用中日誌資料集的相關資訊，請使用:

```
NEWLOG DSNAME=MQM111.LOGCOPY1.DS10,COPY2,STARTIME=19930212205198,
ENDTIME=19930412205200,STARTRBA=6400,ENDRBA=94FF
```

您可能需要在 BSDS 中插入包含此類型資訊的記錄，因為:

- 已刪除資料集的項目，但再次需要。
- 您正在將一個作用中日誌資料集的內容複製到另一個資料集。
- 您正在從備份副本回復 BSDS。

增加作用中日誌的大小

實現這一程序有兩種方法。

1. 當佇列管理程式處於作用中狀態時:
   a. 使用 JCL 來定義新的較大日誌資料集。
   b. 使用 MQSC DEFINE LOG 指令，將新的日誌資料集新增至作用中佇列管理程式。
   c. 請使用 MQSC ARCHIVE LOG 指令來移動現行作用中日誌，以成為新的較大日誌。
   d. 等待較小作用中日誌資料集的保存完成。
   e. 使用 CSQJU003 公用程式來移除舊的小型作用中日誌，以關閉佇列管理程式。
   f. 重新啟動佇列管理程式。
2. 當佇列管理程式處於非作用中狀態時:
   a. 停止佇列管理程式。 這是必要步驟，因為 IBM MQ 會配置所有作用中日誌資料集，以供其在作用中時專用。
   b. 搭配使用 Access Method Services ALTER 與 NEWNAME 選項，以重新命名作用中日誌資料集。
   c. 使用 Access Method Services DEFINE 來定義較大的作用中日誌資料集。

      透過重複使用舊資料集名稱，您無需執行變更日誌庫存公用程式，即可在 BSDS 中建立新名稱。 舊資料集名稱及正確的 RBA 範圍已在 BSDS 中。
   d. 使用 Access Method Services REPRO 將舊 (重新命名) 資料集複製到其適當的新資料集。

      註: 此步驟可能需要很長時間，因此您的企業在此期間可能無法運作。
   e. 啟動佇列管理程式。

如果您的所有日誌資料集都是相同的大小，則您的系統作業會更一致且更有效率。 如果日誌資料集的大小不同，則更難以追蹤系統的日誌，因此可能會浪費空間。

使用 CSQJUFMT

增加作用中日誌的大小時，請勿執行 CSQJUFMT 格式。

如果您執行 CSQJUFMT (為了在佇列管理程式第一次寫入新的作用中日誌時提供效能優勢) ，您會收到下列訊息:

```
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，如果您使用 Access Method Services REPRO ，請確保定義新的空日誌。

如果您使用 REPRO 將舊 (重新命名) 資料集複製到其個別的新資料集，則預設值為 NOREPLACE。

這表示 REPRO 不會取代已在指定資料集上的記錄。 對資料集執行格式化時，會重設 RBA 值。 淨結果是格式化之後不是空的資料集。

保存日誌的變更

請利用這個主題來瞭解如何變更保存日誌。

您可以在保存日誌的 BSDS 中新增、刪除及變更項目的密碼。 這裡只顯示範例; 將顯示的資料集名稱取代為您要使用的資料集名稱。 如需公用程式的詳細資料，請參閱 變更日誌庫存公用程式。

- 新增保存日誌
- 刪除保存日誌
- 變更保存日誌的密碼

新增保存日誌

當物件的回復取決於讀取現有的保存日誌資料集時， BSDS 必須包含該資料集的相關資訊，讓 IBM MQ 可以找到它。 若要登錄 BSDS 中現有保存日誌資料集的相關資訊，請使用:

```
NEWLOG DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04,
UNIT=TAPE,STARTRBA=3A190000,ENDRBA=3A1F0FFF,CATALOG=NO
```

刪除保存日誌

若要刪除一或多個磁區上的整個保存日誌資料集，請使用:

```
DELETE DSNAME=CSQARC1.ARCHLOG1.E00021.T2205197.A0000015,COPY1VOL=CSQV04
```

變更保存日誌的密碼

如果您變更現有保存日誌資料集的密碼，則也必須變更 BSDS 中的資訊。

1. 使用列印日誌對映公用程式列出 BSDS。
2. 使用 CSQJU003 公用程式的 DELETE 功能，刪除具有已變更密碼之保存日誌資料集的項目 (請參閱 變更日誌庫存公用程式 主題)。
3. 將資料集命名為新的保存日誌資料集。 使用 CSQJU003 公用程式的 NEWLOG 函數 (請參閱主題 變更日誌庫存公用程式)，並提供新密碼、開始及結束 RBA，以及磁區序號 (可在列印日誌對映公用程式輸出中找到，請參閱 列印日誌對映公用程式)。

若要變更新保存日誌資料集的密碼，請使用:

```
ARCHIVE PASSWORD= password
```

若要停止將密碼放置在新的保存日誌資料集上，請使用:

```
ARCHIVE NOPASSWD
```

註: 只有在您沒有外部安全管理程式時，才使用 ARCHIVE 公用程式功能。

變更日誌和 *BSDS* 的高階限定元 *(HLQ)*
請利用這個主題來瞭解變更高階限定元 (HLQ) 所需的程序。

開始之前

在將任何日誌或資料集複製到新的資料集之前，您必須正常結束佇列管理程式。 這是為了確保資料一致，且在重新啟動期間不需要任何回復。

關於這項作業

此作業提供如何變更日誌及 BSDS 之 HLQ 的相關資訊。 為達成此目的，請遵循下列步驟:

程序

1. 執行日誌列印公用程式 CSQJU004，以記錄日誌資料集資訊。 稍後需要此資訊。
2. 您可以執行下列任一動作:
   a) 在要重新命名的日誌和 BSDS 資料集上執行 DSS 備份及還原，或
   b) 使用 AMS DEFINE 和 REPRO 來建立 HLQ 資料集，並從舊資料集複製資料。
3. 修改 MSTR 和 CHIN 程序以指向新的資料集。

4. 使用 CSQJU003 刪除 BSDS 新副本中的舊日誌資訊。
5. 使用 CSQJU003 的 NEWLOG 函數，將新的日誌資料集定義至新的 BSDS。
   除了 HLQ 之外，每一個日誌的所有相關資訊都保持相同。
6. 新的 BSDS 應該反映針對舊 BSDS 中舊日誌所記錄的相同資訊。
   HLQ 應該是唯一改變的東西。

下一步

在啟動佇列管理程式之前，請比較新舊 BSDS 的 CSQJU004 輸出，以確定它們看起來完全相同 (HLQ 除外)。

註: 執行這些作業時必須小心。 不正確的動作可能會導致無法復原的狀況。 請檢查 PRINT LOG MAP UTILITY 輸出，並確定已包含回復或重新啟動所需的所有資訊。

回復 *BSDS*

如果 IBM MQ 以雙重 BSDS 模式運作，且有一個 BSDS 損壞，則強制 IBM MQ 進入單一 BSDS 模式， IBM MQ 會繼續運作而沒有問題 (直到下次重新啟動為止)。

若要讓環境回到雙重 BSDS 模式，請執行下列動作:

1. 使用「存取方法服務」來重新命名或刪除損壞的 BSDS ，以及定義與損壞 BSDS 同名的新 BSDS。 在 thlqual.SCSQPROC 的工作 CSQ4BREC 中可以找到控制陳述式範例。
2. 發出 IBM MQ 指令 RECOVER BSDS ，以在新配置的資料集中建立有效 BSDS 的副本，並恢復雙重 BSDS 模式。

如果 IBM MQ 以單一 BSDS 模式運作且 BSDS 損壞，或如果 IBM MQ 以雙重 BSDS 模式運作且兩個 BSDS 都損壞，則佇列管理程式會停止且不會重新啟動，直到 BSDS 資料集修復為止。 在此情況下:

1. 尋找與最新保存日誌資料集相關聯的 BSDS。 最近保存日誌的資料集名稱會出現在工作日誌最後一次出現的訊息 CSQJ003I 中，指出卸載處理已順利完成。 在準備此程序的其餘部分時，最好保留該訊息所指出的所有成功保存檔的日誌:
   - 如果保存日誌位於 DASD 上，則會在任何可用的 DASD 上配置 BSDS。 BSDS 名稱類似於對應的保存日誌資料集名稱; 只將最後一個限定元的第一個字母從 A 變更為 B ，如下列範例所示:

     保存日誌名稱
     CSQ.ARCHLOG1. ***A*** 0000001

     BSDS 副本名稱
     CSQ.ARCHLOG1. ***B*** 0000001
   - 如果保存日誌位於磁帶上，則 BSDS 是第一個保存日誌磁區的第一個資料集。 在後續磁區上不會重複 BSDS。
2. 如果最新保存日誌資料集沒有 BSDS 的副本 (例如，因為卸載時發生錯誤) ，請從先前的卸載處理中尋找先前的 BSDS 副本。
3. 搭配使用 Access Method Services ALTER 指令與 NEWNAME 選項，以重新命名 損壞的 BSDS。 如果您要刪除損壞的 BSDS ，請使用「存取方法服務 DELETE」指令。 對於每一個損壞的 BSDS ，請使用「存取方法服務」來定義新的 BSDS 作為取代資料集。 thlqual.SCSQPROC 中的工作 CSQ4BREC 包含存取方法服務控制陳述式，可定義新的 BSDS。
4. 使用 Access Method Services REPRO 指令，將保存日誌中的 BSDS 複製到您在步驟 第 271 頁的『3』中定義的其中一個取代 BSDS。 請勿將任何資料複製到第二個取代 BSDS ，您可以在步驟 第 272 頁的『5』中這樣做。
   a. 列印取代 BSDS 的內容。

      使用列印日誌對映公用程式 (CSQJU004) 來列印取代 BSDS 的內容。 這可讓您在繼續回復工作之前檢閱取代 BSDS 的內容。

   b. 更新取代 BSDS 中的保存日誌資料集庫存。

      檢查列印日誌對映公用程式的輸出，並檢查取代的 BSDS 是否不包含從中複製 BSDS 的保存日誌記錄。 如果取代 BSDS 是舊副本，則其庫存可能不會包含最近建立的所有保存日誌資料集。 必須更新保存日誌資料集的 BSDS 庫存，以反映現行子系統庫存。

使用變更日誌庫存公用程式 (CSQJU003) NEWLOG 陳述式來更新取代 BSDS，並新增從中複製 BSDS 的保存日誌記錄。 如果保存日誌資料集受密碼保護，請使用 NEWLOG 函數的 PASSWORD 選項。 此外，如果已編目保存日誌資料集，請確定 NEWLOG 函數的 CATALOG 選項已適當地設為 CATALOG=YES。 使用 NEWLOG 陳述式來新增任何比 BSDS 副本晚建立的其他保存日誌資料集。

c. 更新取代 BSDS 中的密碼。

BSDS 包含保存日誌資料集及作用中日誌資料集的密碼。 若要確保替代 BSDS 中的密碼反映您安裝所使用的現行密碼，請搭配使用變更日誌庫存 ARCHIVE 公用程式功能與 PASSWORD 選項。

d. 更新取代 BSDS 中的作用中日誌資料集庫存。

在特殊情況下，自複製 BSDS 之後，您的安裝可能已新增、刪除或重新命名作用中日誌資料集。 在此情況下，取代的 BSDS 不會反映您安裝目前使用中的作用中日誌資料集實際數目或名稱。

如果您需要從取代 BSDS 日誌庫存中刪除作用中日誌資料集，請使用變更日誌庫存公用程式 DELETE 功能。

如果您需要將作用中日誌資料集新增至取代 BSDS 日誌庫存，請使用變更日誌庫存公用程式 NEWLOG 功能。 請確定已在 NEWLOG 函數上正確地指定 RBA 範圍。 如果作用中日誌資料集受密碼保護，請使用 PASSWORD 選項。

如果您需要重新命名取代 BSDS 日誌庫存中的作用中日誌資料集，請使用變更日誌庫存公用程式 DELETE 函數，後面接著 NEWLOG 函數。 請確定已在 NEWLOG 函數上正確地指定 RBA 範圍。 如果作用中日誌資料集受密碼保護，請使用 PASSWORD 選項。

e. 更新取代 BSDS 中的作用中日誌 RBA 範圍。

稍後，當佇列管理程式重新啟動時，它會比較 BSDS 中列出的作用中日誌資料集的 RBA 與實際作用中日誌資料集中找到的 RBA。 如果 RBA 不同意，則佇列管理程式不會重新啟動。 使用 BSDS 的舊副本時，問題會放大。 若要解決此問題，請使用變更日誌庫存公用程式 (CSQJU003)，使用實際作用中日誌資料集中的 RBA 來調整 BSDS 中找到的 RBA。 您可以透過下列方式執行此動作:

- 使用列印日誌記錄公用程式 (CSQ1LOGP) 來列印作用中日誌資料集的摘要報告。 這會顯示開始及結束 RBA。
- 已知所有作用中日誌資料集的 RBA 時，比較實際 RBA 範圍與您剛列印的 RBA 範圍。

  如果所有作用中日誌資料集的 RBA 範圍相等，您可以繼續進行下一個回復步驟，而不需要任何其他工作。

  如果 RBA 範圍不相等，請調整 BSDS 中的值以反映實際值。 對於需要調整 RBA 範圍的每一個作用中日誌資料集，請使用變更日誌庫存公用程式 DELETE 功能，從取代 BSDS 中的庫存刪除作用中日誌資料集。 然後使用 NEWLOG 函數，將作用中日誌資料集重新定義為 BSDS。 如果作用中日誌資料集受密碼保護，請使用 NEWLOG 函數的 PASSWORD 選項。

f. 如果每一個作用中日誌副本只指定兩個作用中日誌資料集，則在重新啟動佇列管理程式期間， IBM MQ 會有困難。 當其中一個作用中日誌資料集已滿且尚未卸載，而第二個作用中日誌資料集即將填滿時，可能會發生此問題。 在此情況下，請為作用中日誌的每一個副本新增作用中日誌資料集，並在取代 BSDS 日誌庫存中定義每一個新的作用中日誌資料集。

使用「存取方法服務 DEFINE」指令為作用中日誌的每一個副本定義新的作用中日誌資料集，並使用變更日誌庫存公用程式 NEWLOG 功能在取代 BSDS 中定義新的作用中日誌資料集。 您不需要在 NEWLOG 陳述式上指定 RBA 範圍。 不過，如果作用中日誌資料集受到密碼保護，請使用 NEWLOG 函數的 PASSWORD 選項。 可以在 thlqual.SCSQPROC 中的工作 CSQ4LREC 中找到完成此作業的控制陳述式範例。

5. 將更新的 BSDS 複製到第二個新的 BSDS 資料集。 現在 BSDS 是相同的。

   此時使用列印日誌對映公用程式 (CSQJU004) 來列印第二個取代 BSDS 的內容。

6. 如需您遺失現行作用中日誌資料集時要執行之動作的相關資訊，請參閱 作用中日誌問題 。
7. 使用新建構的 BSDS 來重新啟動佇列管理程式。 IBM MQ 決定現行 RBA 以及需要保存哪些作用中日誌。

管理頁面集

請利用這個主題來瞭解如何管理與佇列管理程式相關聯的頁面集。

本主題說明如何新增、複製及一般管理與佇列管理程式相關聯的頁面集。 它包含下列區段:

如需頁集、儲存類別、緩衝區及緩衝池的說明，以及一些適用的效能考量，請參閱 頁集 。

如何變更頁面集的高階限定元 (HLQ)

此作業提供如何變更頁面集的 HLQ 的相關資訊。 若要執行此作業，請執行下列動作:

1. 定義新的 HLQ 頁面集。
2. 如果大小配置與舊頁集相同，請使用 REPRO 將現有頁集複製到空的新 HLQ 頁集。 如果您要增加頁集的大小，請使用 CSQUTIL 的 FORMAT 函數來格式化目的地頁集。 如需相關資訊，請參閱 格式化頁集 (FORMAT)。
3. 使用 CSQUTIL 的 COPYPAGE 函數，將來源頁集的所有訊息複製到目的地頁集。 如需相關資訊，請參閱 展開頁集 (COPYPAGE)。
4. 變更佇列管理程式程序中的 CSQP00xx DD 陳述式，以指向新的 HLQ 頁面集。

重新啟動佇列管理程式，並驗證對頁面集所做的變更。

如何將頁面集新增至佇列管理程式

此說明假設您具有已在執行中的佇列管理程式。 例如，如果您的佇列管理程式必須使用新的佇列來處理新的應用程式，則您可能需要新增頁集。

若要新增頁集，請使用下列程序:

1. 定義並格式化新頁集。 您可以使用 thlqual.SCSQPROC(CSQ4PAGE) 中的範例 JCL 作為基準。 如需相關資訊，請參閱 格式化頁集 (FORMAT)。

   請小心不要格式化任何使用中的頁集，除非這是您想要的。 若是如此，請使用 FORMAT 公用程式函數的 FORCE 選項。
2. 搭配使用 DEFINE PSID 指令與 DSN 選項，以建立頁面集與緩衝池的關聯。
3. 透過發出 DEFINE STGCLASS 指令，為您的頁集新增適當的儲存類別定義。
4. 選擇性地記載佇列管理程式的配置方式:
   a. 將新頁集新增至佇列管理程式的已啟動作業程序。
   b. 將新頁集的定義新增至 CSQINP1 起始設定資料集。
   c. 將新儲存類別的定義新增至 CSQ4INYR 起始設定資料集成員。

如需 DEFINE PSID 及 DEFINE STGCLASS 指令的詳細資料，請參閱 DEFINE PSID 及 DEFINE STGCLASS。

當其中一個頁面集已滿時要執行的動作

您可以使用 IBM MQ 指令 DISPLAY USAGE，以瞭解頁集的使用率。 例如，指令:

```
DISPLAY USAGE PSID(03)
```

顯示頁集 03 的現行狀態。 這會告訴您此頁集有多少可用頁面。

如果您已為頁面集定義次要延伸範圍，則每次填滿時都會動態展開這些延伸範圍。 最終會使用所有次要延伸範圍，或沒有進一步可用的磁碟空間。 如果發生這種情況，應用程式會收到回覆碼 MQRC_STORAGE_MEDIUM_FULL。

如果應用程式收到來自 MQI 呼叫的回覆碼 MQRC_STORAGE_MEDIUM_FULL，則表示頁集上沒有足夠的剩餘空間。 如果問題持續或可能再次發生，您必須執行一些動作來解決它。

您可以使用下列數種方式來處理此問題:

- 將佇列從一個頁集移至另一個頁集，以平衡頁集之間的負載。
- 展開頁面集。 請參閱第 276 頁的『如何增加頁集的大小』以取得相關指示。
- 請重新定義頁集，讓它可以擴充到超過 4 GB 的大小上限 (64 GB)。 如需指示，請參閱 定義大於 4 GB 的頁集 。

如何平衡頁面集上的負載

頁集上的負載平衡表示將與一個以上佇列相關聯的訊息從一個頁集移至另一個頁集 (較少使用)。 如果展開頁集並不實際，請使用此技術。

若要識別哪些佇列正在使用頁集，請使用適當的 IBM MQ 指令。 例如，若要找出哪些佇列對映至頁集 02，請先使用下列指令找出哪些儲存類別對映至頁集 02:

```
DISPLAY STGCLASS(*) PSID(02)
```

然後使用下列指令，找出哪些佇列使用哪個儲存類別:

```
DISPLAY QUEUE(*) TYPE(QLOCAL) STGCLASS
```

移動非共用佇列

若要將佇列及其訊息從一個頁集移至另一個頁集，請使用 MQSC MOVE QLOCAL 指令 (在 MOVE QLOCAL 中說明)。 當您已識別要移至新頁集的一或多個佇列時，請針對下列每一個佇列遵循下列程序:

1. 請確定您要移動的佇列未被任何應用程式使用 (亦即， DISPLAY QSTATUS 指令中的 IPPROCS 及 OPPROCS 值為零)，且沒有未確定的訊息 (DISPLAY QSTATUS 指令中的 UNCOM 值為 NO)。

   註: 確保此狀態繼續的唯一方法是暫時變更佇列的安全授權。 如需相關資訊，請參閱 佇列安全的設定檔 。

   如果您無法執行此動作，則在應用程式開始使用佇列時，即使採取諸如設定 PUT (DISABLED) 之類的預防措施步驟，此程序中的後續階段也可能會失敗。 不過，此程序絕不會遺失訊息。
2. 透過變更佇列定義以停用 MQPUT，防止應用程式將訊息放置在佇列上。 將佇列定義變更為 PUT (DISABLED)。
3. 使用下列指令，定義與要移動的佇列具有相同屬性的暫時佇列:

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED)
```

註: 如果前次執行時已存在此暫時佇列，請先刪除它，然後再執行定義。

4. 使用下列指令，將訊息移至暫時佇列:

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. 使用下列指令，刪除您要移動的佇列:

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. 定義對映至所需頁集的新儲存類別，例如:

```
DEFINE STGCLASS(NEW) PSID(nn)
```

將新的儲存類別定義新增至 CSQINP2 資料集，以供下次重新啟動佇列管理程式時使用。

7. 透過變更儲存類別屬性，重新定義您要移動的佇列:

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) STGCLASS(NEW)
```

重新定義佇列時，它會以步驟 第 274 頁的『3』中建立的暫時佇列為基礎。

8. 使用下列指令，將訊息移回新佇列:

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

9. 不再需要在步驟 第 274 頁的『3』 中建立的佇列。 請使用下列指令來刪除它:

```
DELETE QL(TEMP_QUEUE)
```

10. 如果要移動的佇列定義在 CSQINP2 資料集中，請在 CSQINP2 資料集中變更適當 DEFINE QLOCAL 指令的 STGCLASS 屬性。 新增 REPLACE 關鍵字，以取代現有的佇列定義。

第 276 頁的圖 45 顯示從負載平衡工作擷取。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.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)
/*
```

圖 *45:* 從頁面集的負載平衡工作擷取

如何增加頁集的大小

您可以起始配置大於 4 GB 的頁集，請參閱 將頁集定義為大於 4 GB

您可以指定 EXPAND (SYSTEM) 或 EXPAND (USER)，將頁集定義成在已滿時自動展開。 如果您的頁集是以 EXPAND (NONE) 來定義，您可以用下列兩種方式之一來展開它:

- 變更其定義以容許自動擴充。 請參閱 變更頁集以容許自動擴充
- 建立新的較大頁集，並將訊息從舊頁集複製到新頁集。 請參閱 將訊息移至新的較大頁集

將頁集定義為大於 4 GB

IBM MQ 可以使用大小最多為 64 GB 的頁面集，前提是資料集定義為 VSAM 的「延伸定址能力」。 延伸定址能力是 SMS 資料類別所提供的屬性。 在下列範例 JCL 中顯示的範例中，管理類別 'EXTENDED' 定義給具有 'Extended addressability' 的 SMS。 如果您的現有頁集目前未定義為具有延伸定址能力，請使用下列方法來移轉至延伸定址能力格式資料集。

1. 停止佇列管理程式。
2. 使用「存取方法服務」來重新命名現有的頁面集。
3. 定義目的地頁集，其大小與現有頁集相同，但具有 DATACLas (EXTENDED)。

   註: 延伸格式資料集必須由 SMS 管理。 以下是 VSAM 資料集的要求延伸格式的機制:

   - 使用 DSNTYPE 值為 EXT 及子參數 R 或 P 的資料類別，以指出必要或偏好。
   - 在 DD 陳述式上編碼 DSNTYPE=EXTREQ (需要延伸格式) 或 DSNTYPE=EXTPREF (偏好延伸格式)。
   - 在 DD 陳述式上撰寫 LIKE= 參數的程式碼，以參照現有的延伸格式資料集。

   如需相關資訊，請參閱 定義延伸格式資料集的限制。
4. 使用 CSQUTIL 的 COPYPAGE 函數，將來源頁集的所有訊息複製到目的地頁集。 如需詳細資料，請參閱 展開頁面集 (COPYPAGE) 。
5. 重新啟動佇列管理程式。
6. 變更頁集以使用系統擴充，以容許它在現行配置之外繼續成長。

下列 JCL 顯示「存取方法服務」指令範例:

```
//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))
/*
```

變更頁集以容許自動擴充

搭配使用 ALTER PSID 指令與 EXPAND (USER) 或 EXPAND (SYSTEM) 選項。 如需展開頁集的一般資訊，請參閱 ALTER PSID 及 展開頁集 (COPYPAGE) 。

將訊息移至新的較大頁集

這項技術涉及停止及重新啟動佇列管理程式。 這會刪除重新啟動時不在共用佇列上的任何非持續訊息。 如果您有不想刪除的非持續訊息，請改用負載平衡。 如需詳細資料，請參閱第 274 頁的『如何平衡頁面集上的負載』。 在此說明中，您要展開的頁集稱為 來源 頁集; 新的較大頁集稱為 目的地 頁集。

請遵循下列步驟:

1. 停止佇列管理程式。
2. 定義目的地頁集，以確保它大於來源頁集，並具有較大的次要延伸範圍值。
3. 使用 CSQUTIL 的 FORMAT 函數來格式化目的地頁集。 如需詳細資料，請參閱 格式化頁集 (FORMAT) 。
4. 使用 CSQUTIL 的 COPYPAGE 函數，將來源頁集的所有訊息複製到目的地頁集。 如需詳細資料，請參閱 展開頁面集 (COPYPAGE) 。
5. 執行下列其中一項，以使用目的地頁集來重新啟動佇列管理程式:
   - 請變更佇列管理程式啟動型作業程序，以參照目的地頁集。
   - 使用「存取方法服務」來刪除來源頁集，然後重新命名目的地頁集，使其名稱與來源頁集的名稱相同。

請注意:

在刪除任何 IBM MQ 頁集之前，請確定您已建立必要的備份副本。

如何減少頁面集

防止 IBM MQ 管理者以外的所有使用者使用佇列管理程式。 例如，透過變更存取安全設定。

如果您的大型頁集大部分是空的 (如 DISPLAY USAGE 指令所示) ，您可能想要減少其大小。 執行此動作的程序涉及使用 CSQUTIL 的 COPY、FORMAT 及 LOAD 函數 (請參閱 IBM MQ 公用程式)。 此程序不適用於頁集零 (0) ，因為減少此頁集的大小並不實際; 唯一的方法是重新起始設定佇列管理程式 (請參閱 第 296 頁的『重新起始設定佇列管理程式』)。 此程序的必要條件是嘗試從系統中移除所有使用者，以便所有 UOW 都完成且頁面集一致。

1. 搭配使用 STOP QMGR 指令與 QUIESCE 或 FORCE 屬性，以停止佇列管理程式。
2. 使用 PSID 選項執行 CSQUTIL 的 SCOPY 函數，以複製大型頁面集中的所有訊息資料，並將它們儲存在循序資料集中。
3. 定義新的較小頁面集資料集，以取代大型頁面集。
4. 針對您在步驟 第 277 頁的『3』 中建立的頁集，執行 CSQUTIL 的 FORMAT TYPE (NEW) 函數。
5. 使用在步驟 第 277 頁的『3』 中建立的頁集來重新啟動佇列管理程式。
6. 執行 CSQUTIL 的 LOAD 函數，以重新載入在步驟 第 277 頁的『2』 期間儲存的所有訊息。
7. 容許所有使用者存取佇列管理程式。

8. 刪除舊的大型頁面集。

如何重新建立頁集

在某些情況下，將舊頁集重新連線至佇列管理程式會很有用。 除非採取特定動作，否則當舊頁集上線時，佇列管理程式會辨識出頁集本身及檢查點記錄中所儲存的頁集回復 RBA 是舊的，因此會自動啟動頁集的媒體回復，使其保持最新。

此類媒體回復只能在佇列管理程式重新啟動時執行，且可能需要相當長的時間，尤其是必須讀取保留在磁帶上的保存日誌時。 不過，在此情況下，頁集在中間期間已離線，因此日誌不包含與頁集回復相關的資訊。

可用的選項有下列三個:

容許執行完整媒體回復。

1. 停止佇列管理程式。
2. 請確定佇列管理程式的已啟動作業程序及 CSQINP1 起始設定資料集中的頁集都有可用的定義。
3. 重新啟動佇列管理程式。

容許毀損頁集上的任何訊息。

如果頁面集已離線很長時間 (例如幾個月)，且現在已決定將它重複用於不同的用途，則此選項非常有用。

1. 使用具有 TYPE (NEW) 選項之 CSQUTIL 的 FORMAT 函數來格式化頁集。
2. 將頁集的定義新增至佇列管理程式及 CSQINP1 起始設定資料集的已啟動作業程序。
3. 重新啟動佇列管理程式。

使用 TYPE (NEW) 選項進行格式化會清除頁集的現行內容，並告知佇列管理程式忽略頁集相關檢查點中的任何歷程資訊。

讓頁集上線，避免媒體回復處理程序。

只有在您確定佇列管理程式完全關閉之後頁集已離線時，才使用此技術。 此選項最適合在頁面集短暫離線時使用，通常是由於作業問題 (例如啟動佇列管理程式時執行備份)。

1. 使用具有 TYPE (REPLACE) 選項之 CSQUTIL 的 FORMAT 函數來格式化頁集。
2. 請使用 DEFINE PSID 指令搭配 DSN 選項，以動態方式將頁集新增回佇列管理程式，或容許在佇列管理程式重新啟動時新增該頁集。

使用 TYPE (REPLACE) 選項來格式化佇列管理程式已完全關閉頁集的檢查，並將它標示為不會執行媒體回復。 不會對頁面集的內容進行其他變更。

如何備份及回復頁集

有不同的機制可用於備份及回復。 請利用這個主題來瞭解這些機制。

本節說明下列主題:

- 第 278 頁的『建立非共用資源的回復點』
- 第 279 頁的『備份頁面集』
- 第 280 頁的『回復頁集』
- 如何刪除頁面集

如需如何為共用資源建立回復點的相關資訊，請參閱 第 284 頁的『回復共用佇列』。

建立非共用資源的回復點

在下列情況下， IBM MQ 可以將物件及非共用持續訊息回復至其現行狀態:

1. 先前點的頁集副本已存在。
2. 所有 IBM MQ 日誌都可用來從該點執行回復。

這些代表非共用資源的回復點。

物件和訊息都保留在頁集上。 相同頁集中可以存在來自不同佇列的多個物件和訊息。 基於回復目的，無法單獨備份物件及訊息，因此必須整體備份頁集，以確保正確回復資料。

IBM MQ 回復日誌包含所有持續訊息及對物件所做變更的記錄。 如果 IBM MQ 失敗 (例如，由於頁集上的 I/O 錯誤)，您可以透過還原備份副本並重新啟動佇列管理程式來回復頁集。 IBM MQ 會將日誌變更從備份副本點套用至頁集。

有兩種方法可以建立回復點:

完整備份
停止佇列管理程式，這會強制對頁集進行所有更新。

這可讓您從回復點重新啟動，只使用已備份的頁面集資料集，以及從該點開始的日誌。

模糊備份
在不停止佇列管理程式的情況下取得頁面集的 模糊 備份副本。

如果您使用此方法，且稍後相關聯的日誌已損壞或遺失，則無法使用模糊頁集備份副本進行回復。 這是因為模糊頁集備份副本包含不一致的佇列管理程式狀態視圖，且取決於可用的日誌。 如果日誌無法使用，您需要回到子系統非作用中時所取得的最後一組備份頁集副本 (方法 1) 並接受從那時開始的資料流失。

方法 1: 完整備份

此方法涉及關閉佇列管理程式。 這會強制對頁面集進行所有更新，以便頁面集處於一致狀態。

1. 停止所有正在使用佇列管理程式的 IBM MQ 應用程式 (讓它們先完成)。 例如，可以透過變更存取安全或佇列設定來完成此動作。
2. 當所有活動都已完成時，顯示並解決任何不確定的回復單元。 (請依照 DISPLAY CONN 和 RESOLVE INDOUBT 中的說明，使用 DISPLAY CONN 和 RESOLVE INDOUBT 指令。)

   這會使頁面集進入一致狀態; 如果您不這麼做，則頁面集可能不一致，且您實際上正在執行模糊備份。
3. 發出 ARCHIVE LOG 指令，以確保將最新的日誌資料寫出至日誌資料集。
4. 發出 STOP QMGR MODE (QUIESCE) 指令。 記錄 CSQI024I 或 CSQI025I 訊息中的最低 RBA 值 (如需相關資訊，請參閱 CSQI024I 及 CSQI025I)。 您應該將日誌資料集從 RBA 值所指示的日誌資料集，保留到現行日誌資料集為止。
5. 取得所有佇列管理程式頁集的備份副本 (請參閱 第 279 頁的『備份頁面集』)。

方法 2: 模糊備份

此方法不涉及關閉佇列管理程式。 因此，在備份處理程序期間，虛擬儲存體緩衝區中可能有更新項目。 這表示頁面集不是處於一致狀態，只能用於日誌的回復。

1. 發出 DISPLAY USAGE TYPE (ALL) 指令，並在 CSQI024I 或 CSQI025I 訊息中記錄 RBA 值 (如需相關資訊，請參閱 CSQI024I 及 CSQI025I)。
2. 取得頁面集的備份副本 (請參閱 第 279 頁的『備份頁面集』)。
3. 發出 ARCHIVE LOG 指令，以確保將最新的日誌資料寫出至日誌資料集。 若要從回復點重新啟動，您必須將日誌資料集從 RBA 值所指示的日誌資料集啟動，直到現行日誌資料集為止。

備份頁面集

若要回復頁集， IBM MQ 需要知道日誌中要回溯多久。 IBM MQ 會維護每個頁集第 0 頁中的日誌 RBA 號碼，稱為 回復日誌序號 (LSN)。 此數字是日誌中的起始 RBA， IBM MQ 可以從中回復頁集。 當您備份頁集時，也會複製此數字。

如果稍後使用副本來回復頁集，則 IBM MQ 必須具有從此 RBA 值到現行 RBA 的所有日誌記錄的存取權。 這表示您必須保留足夠的日誌記錄，才能讓 IBM MQ 從您想要保留之頁集的最舊備份副本回復。

使用 ADRDSSU COPY 功能來複製頁集。

如需相關資訊，請參閱 COPY DATASET 邏輯資料集的指令語法 文件。

例如:

```
//STEP2 EXEC PGM=ADRDSSU,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)
/*
//
```

如果您在佇列管理程式執行時複製頁集，則必須先使用複製頁集第零頁的複製公用程式。 如果您不這麼做，可能會毀損頁集中的資料。

如果動態擴充頁集的處理程序被岔斷 (例如，因為失去系統電源)，您仍然可以使用 ADRDSSU 來備份頁集。

如果您執行 Access Method Services IDCAMS `LISTCAT ENT('page set data set name') ALLOC`，您會看到 `HI-ALLOC-RBA` 高於 `HI-USED-RBA`。

下次此頁面設定填滿時，它會重新延伸 (可能的話)，並使用高使用 RBA 與最高配置 RBA 之間的頁面，以及另一個新的延伸範圍。

備份物件定義

您也應該備份物件定義的副本。 如果要這麼做，請使用 CSQUTIL COMMAND 函數的 MAKEDEF 特性 (如 發出指令至 IBM MQ (COMMAND) 中所說明)。

每當您取得佇列管理程式的備份副本時，請備份物件定義，並保留最新版本。

回復頁集

如果佇列管理程式因失敗而終止，則通常可以重新啟動佇列管理程式，並在重新啟動期間執行所有回復。 不過，如果沒有任何頁集或日誌資料集可用，則無法進行這類回復。 您現在可以回復的範圍取決於頁集及日誌資料集備份副本的可用性。

若要從回復點重新啟動，您必須具有:

- 要回復之頁集的備份副本。
- 如果您已使用 第 279 頁的『方法 2: 模糊備份』中說明的 "模糊" 備份處理程序，則包含記錄 RBA 值的日誌資料集、ARCHIVE LOG 指令所建立的日誌資料集，以及這些日誌資料集之間的所有日誌資料集。
- 如果您已使用完整備份，但沒有遵循 ARCHIVE LOG 指令所建立的日誌資料集，則 **不** 需要對所有頁集執行 CSQUTIL 公用程式的 FORMAT TYPE (REPLACE) 函數。

若要將頁集回復至其現行狀態，您還必須具有自 ARCHIVE LOG 指令以來的所有日誌資料集及記錄。

有兩種方法可回復頁集。 若要使用任一方法，必須停止佇列管理程式。

簡式回復

這是較簡單的方法，適用於大部分回復狀況。

1. 刪除您要從備份還原的頁集。
2. 使用 ADRDSSU COPY 功能，從備份副本回復頁集。

   或者，您可以將備份副本重新命名為原始名稱，或變更佇列管理程式程序中的 CSQP00xx DD 陳述式，以指向備份頁集。 不過，如果您隨後遺失或毀損頁面集，則將不再具有要從中還原的備份副本。
3. 重新啟動佇列管理程式。
4. 當佇列管理程式順利重新啟動時，您可以重新啟動應用程式

5. 恢復已還原頁面的一般備份程序。

進階回復

如果您要回復大型頁集，或自前次備份以來頁集有許多活動，則此方法會提供效能優點。 不過，它需要比簡式方法更多的人為介入，這可能會增加錯誤的風險，以及執行回復所花費的時間。

1. 刪除並重新定義您要從備份還原的頁集。
2. 使用 ADRDSSU ，將頁集的備份副本複製到新的頁集。 使用次要範圍值來定義新的頁面集，以便可以動態展開它。

   或者，您可以將備份副本重新命名為原始名稱，或變更佇列管理程式程序中的 CSQP00xx DD 陳述式，以指向備份頁集。 不過，如果您隨後遺失或毀損頁面集，則將不再具有要從中還原的備份副本。
3. 變更佇列管理程式的 CSQINP1 定義，使與要回復之頁集相關聯的緩衝池儘可能大。 藉由使緩衝池變得很大，您可以讓所有已變更的頁面常駐在緩衝池中，並減少頁集的 I/O 量。
4. 重新啟動佇列管理程式。
5. 當佇列管理程式順利重新啟動時，請停止它 (使用靜止) ，然後使用該頁集的一般緩衝池定義來重新啟動它。 在第二次重新啟動順利完成之後，您可以重新啟動應用程式
6. 恢復已還原頁面的一般備份程序。

重新啟動佇列管理程式時會發生什麼情況

當佇列管理程式重新啟動時，它會將所有變更套用至日誌中登錄的頁集，從頁集的重新啟動點開始。 IBM MQ 可以用這種方式回復多個頁集。 必要的話，會在媒體回復期間動態展開頁集。

在重新啟動期間， IBM MQ 會採用下列最低值來決定要從中啟動的日誌 RBA:

- 從每一個頁集的檢查點日誌記錄回復 LSN。
- 從每個頁集中的第零頁回復 LSN。
- 備份時系統中最舊的不完整回復單元的 RBA。

所有物件定義都儲存在頁集零上。 訊息可以儲存在任何可用的頁集中。

註: 如果頁集零無法使用，則佇列管理程式無法重新啟動。

如何刪除頁面集

您可以使用 DELETE PSID 指令來刪除頁集; 如需此指令的詳細資料，請參閱 DELETE PSID 。

您無法刪除仍由任何儲存類別參照的頁集。 請使用 DISPLAY STGCLASS 來找出哪些儲存類別參照頁集。

已從 IBM MQ 取消配置資料集，但未刪除。 它仍可供未來使用，或者可以使用 z/OS 機能來刪除。

從佇列管理程式的已啟動作業程序中移除頁集。

從 CSQINP1 起始設定資料集中移除頁集的定義。

如何使用 CSQUTIL 備份及還原佇列

如需使用 CSQUTIL 進行備份及還原的進一步相關資訊，請使用本主題作為參照。

您可以使用 CSQUTIL 公用程式函數來備份及還原佇列。 若要備份佇列，請使用 COPY 或 SCOPY 函數，將佇列中的訊息複製到資料集。 若要還原佇列，請使用補充函數 LOAD 或 SLOAD。 如需相關資訊，請參閱 IBM MQ 公用程式。

管理緩衝池

如果您要變更或刪除緩衝池，請使用本主題。

本主題說明如何變更及刪除緩衝池。 它包含下列區段:

- 第 282 頁的『如何變更緩衝池中的緩衝區數目』
- 第 282 頁的『如何刪除緩衝池』

在佇列管理程式起始設定期間，會使用從起始設定輸入資料集 CSQINP1 發出的 DEFINE BUFFPOOL 指令來定義緩衝池。 在佇列管理程式執行時，可以使用本主題中詳述的程序來變更其屬性，以回應商業需求。 佇列管理程式會在檢查點日誌記錄中記錄現行緩衝池屬性。 除非 CSQINP1 中的緩衝池定義包括 REPLACE 屬性，否則這些會在後續佇列管理程式重新啟動時自動還原。

使用 DISPLAY USAGE 指令來顯示現行緩衝區屬性。

您也可以搭配使用 DEFINE PSID 指令與 DSN 選項，來動態定義緩衝池。

如果您動態變更緩衝池，則也應該在起始設定資料集 CSQINP1 中更新其定義。

如需頁集、儲存類別、緩衝區及緩衝池的說明，以及適用的部分效能考量，請參閱 在 z/OS 上規劃 。

註: 緩衝池使用大量儲存體。 當您增加緩衝池大小或定義新的緩衝池時，請確保有足夠的儲存體可用。 如需相關資訊，請參閱 位址空間儲存體。

如何變更緩衝池中的緩衝區數目

如果緩衝池太小，此狀況可能會導致主控台上出現訊息 CSQP020E ，您可以使用 ALTER BUFFPOOL 指令配置更多緩衝區給它，如下所示:

1. 查看日誌中的 CSQY220I 訊息，以判斷新緩衝區可用的空間量。 報告可用空間 (以 MB 為單位)。 因為緩衝區大小為 4 KB ，所以每 MB 可用空間可讓您配置 256 個緩衝區。 請勿將所有可用空間配置給緩衝區，因為其他作業需要一些可用空間。

   如果緩衝池使用固定 4 KB 頁面 (亦即，其 PAGECLAS 屬性為 FIXED4KB) ，請確定 LPAR 上有足夠的實際儲存體可用。
2. 如果報告的可用空間不足，請使用指令從另一個緩衝池釋放部分緩衝區

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   其中 *buf-pool-id* 是您要從中收回空間的緩衝池，而 *integer* 是要配置給此緩衝池的新緩衝區數目，它必須小於配置給它的原始緩衝區數目。
3. 使用指令將緩衝區新增至您要擴充的緩衝池

   ```
   ALTER BUFFPOOL(buf-pool-id) BUFFERS(integer)
   ```

   其中 *buf-pool-id* 是要擴充的緩衝池，而 *integer* 是要配置給此緩衝池的新緩衝區數目，它必須大於配置給它的原始緩衝區數目。

如何刪除緩衝池

當緩衝池不再由任何頁集使用時，請刪除它，以釋放配置給它的虛擬儲存體。

您可以使用 DELETE BUFFPOOL 指令來刪除緩衝池。 如果任何頁集正在使用此緩衝池，則指令會失敗。

如需如何刪除頁集的相關資訊，請參閱 第 281 頁的『如何刪除頁面集』 。

管理佇列共用群組及共用佇列

IBM MQ 可以使用不同類型的共用資源，例如佇列共用群組、共用佇列及連結機能。 請利用這個主題來檢閱管理這些共用資源所需的程序。

本節包含下列主題的相關資訊:

- 第 283 頁的『管理佇列共用群組』
- 第 284 頁的『管理共用佇列』
- 第 288 頁的『管理群組物件』
- 第 288 頁的『管理連結機能』

管理佇列共用群組

您可以在佇列共用群組中新增或移除佇列管理程式，以及管理相關聯的 Db2 表格。

本主題包含下列作業的相關章節:

- 第 283 頁的『將佇列共用群組新增至 Db2 表格』
- 第 283 頁的『將佇列管理程式新增至佇列共用群組』
- 第 283 頁的『從佇列共用群組移除佇列管理程式』
- 第 284 頁的『從 Db2 表格中移除佇列共用群組』
- 第 284 頁的『驗證 Db2 定義的一致性』

將佇列共用群組新增至 Db2 表格

若要將佇列共用群組新增至 Db2 表格，請使用佇列共用群組公用程式 (CSQ5PQSG) 的 ADD QSG 功能。 此程式在 佇列共用群組公用程式中有說明。 thlqual.SCSQPROC(CSQ45AQS) 中提供了範例。

將佇列管理程式新增至佇列共用群組

佇列管理程式可以新增至佇列共用群組，本主題說明部分限制。

若要將佇列管理程式新增至佇列共用群組，請使用佇列共用群組公用程式 (CSQ5PQSG) 的 ADD QMGR 功能。 此程式在 佇列共用群組公用程式中有說明。 thlqual.SCSQPROC(CSQ45AQM) 中提供範例。

佇列共用群組必須存在，您才能將佇列管理程式新增至其中。

佇列共用群組有一個至多四個字元的名稱。 該名稱在您的網路中必須是唯一的，且必須不同於任何佇列管理程式名稱。

佇列管理程式只能是一個佇列共用群組的成員。

註: 若要將佇列管理程式新增至包含執行舊版 IBM MQ 之佇列管理程式的現有佇列共用群組，您必須先將群組中最高版本 IBM MQ 的共存性 PTF 套用至群組中每一個舊版佇列管理程式。

從佇列共用群組移除佇列管理程式

如果另一個處理程序不需要佇列管理程式的日誌，則只能從佇列共用群組中移除佇列管理程式。 如果日誌包含下列項目，則需要這些日誌:

- 佇列共用群組所使用的其中一個連結機能 (CF) 應用程式結構的最新備份
- 未來還原處理程序所需的資料，亦即，自前次備份排除間隔值所說明的時間以來，佇列管理程式已使用可回復的結構。

如果其中一個或兩個點都適用，則無法移除佇列管理程式。 若要判斷未來還原處理程序所需的佇列管理程式日誌，請搭配使用 MQSC DISPLAY CFSTATUS 指令與 TYPE (BACKUP) 選項 (如需此指令的詳細資料，請參閱 DISPLAY CFSTATUS)。

如果不需要佇列管理程式日誌，請採取下列步驟，從佇列共用群組中移除佇列管理程式:

1. 解決涉及此佇列管理程式的任何不確定工作單元。
2. 使用 STOP QMGR MODE (QUIESCE) 完全關閉佇列管理程式。
3. 等待間隔至少等於您將在下一個步驟中於 BACKUP CFSTRUCT 指令中指定的 EXCLINT 參數值。
4. 在另一個佇列管理程式上，使用 MQSC BACKUP CFSTRUCT 指令並指定前一個步驟所需的 EXCLINT 值，對每一個可回復 CF 結構執行 CF 結構備份。
5. 使用 CSQ5PQSG 公用程式的 REMOVE QMGR 功能，可以從佇列共用群組中移除佇列管理程式。 此程式在 佇列共用群組公用程式中有說明。 thlqual.SCSQPROC(CSQ45RQM) 中提供範例。

6. 在重新啟動佇列管理程式之前，請將 QSGDATA 系統參數重設為其預設值。 如需如何自訂系統參數的相關資訊，請參閱 使用 CSQ6SYSP 。

請注意，移除佇列共用群組中的最後一個佇列管理程式時，您必須使用 FORCE 選項，而非 REMOVE。 這會從佇列共用群組中移除佇列管理程式，但不會執行回復所需的佇列管理程式日誌一致性檢查。 只有在刪除佇列共用群組時，才應該執行這項作業。

從 Db2 表格中移除佇列共用群組

若要從 Db2 表格中移除佇列共用群組，請使用佇列共用群組公用程式 (CSQ5PQSG) 的 REMOVE QSG 功能。 此程式在 佇列共用群組公用程式中有說明。 thlqual.SCSQPROC(CSQ45RQS) 中提供範例。

從佇列共用群組中移除所有佇列管理程式之後，您只能從一般 Db2 資料共用群組表格中移除佇列共用群組 (如 第 283 頁的『從佇列共用群組移除佇列管理程式』 中所述)。

從佇列共用群組管理表格中刪除佇列共用群組記錄時，與該佇列共用群組相關的所有物件及管理資訊都會從其他 IBM MQ Db2 表格中刪除。 這包括共用佇列及群組物件資訊。

驗證 Db2 定義的一致性

如果 Db2 物件定義因任何原因而不一致，則佇列共用群組內的共用佇列可能會發生問題。

若要驗證佇列管理程式、CF 結構及共用佇列的 Db2 物件定義一致性，請使用佇列共用群組公用程式 (CSQ5PQSG) 的 Verify QSG 函數。 此程式在 佇列共用群組公用程式中有說明。

管理共用佇列

請利用這個主題來瞭解如何回復、移動及移轉共用佇列。

本節說明下列作業:

- 第 284 頁的『回復共用佇列』
- 第 285 頁的『移動共用佇列』
- 第 287 頁的『將非共用佇列移轉至共用佇列』
- 暫停 Db2 連線

回復共用佇列

在下列情況下， IBM MQ 可以回復共用佇列上的持續訊息:

- 已執行包含訊息之 CF 結構的備份。
- 佇列共用群組中所有佇列管理程式的所有日誌都可用，以從建立備份的點執行回復。
- Db2 可用，且結構備份表格比最新 CF 結構備份還要新。

共用佇列上的訊息儲存在連結機能 (CF) 結構中。 持續訊息可以放置在共用佇列上，如同非共用佇列上的持續訊息一樣，它們會複製到佇列管理程式日誌。 提供 MQSC 備份 CFSTRUCT 及 回復 CFSTRUCT 指令，以容許在不太可能的連結機能失敗事件中回復 CF 結構。 在這種情況下，受影響結構中儲存的任何非持續訊息都會遺失，但持續訊息可以回復。 在回復結構之前，會防止使用該結構的任何進一步應用程式活動。

若要啟用回復，您必須經常使用 MQSC BACKUP CFSTRUCT 指令來備份連結機能清單結構。 CF 結構中的訊息會寫入建立備份之佇列管理程式的作用中日誌資料集。 它會將備份的記錄寫入 Db2: 正在備份的 CF 結構名稱、執行備份的佇列管理程式名稱、該佇列管理程式日誌上此備份的 RBA 範圍，以及備份時間。 備份 CF 清單結構，即使您不主動使用共用佇列 (例如，如果您已設定佇列共用群組，打算在未來使用它)。

您可以對可執行回復的佇列管理程式發出 MQSC RECOVER CFSTRUCT 指令來回復 CF 結構; 您可以使用佇列共用群組中的任何佇列管理程式。 您可以指定要回復的單一 CF 結構，也可以同時回復數個 CF 結構。

如前所述，請務必經常備份 CF 清單結構，否則回復 CF 結構可能需要很長時間。 此外，無法取消回復處理程序。

共用佇列的定義會保留在 Db2 資料庫中，因此可以在必要時使用標準 Db2 資料庫程序來回復。 如需相關資訊，請參閱 共用佇列及佇列共用群組 。

移動共用佇列

本節說明如何將共用佇列從一個連結機能結構移至另一個連結機能結構，以執行負載平衡。 它也說明如何將非共用佇列移至共用佇列，以及如何將共用佇列移至非共用佇列。

當您移動佇列時，需要在程序中定義暫時佇列。 這是因為每一個佇列都必須有唯一名稱，因此您不能有兩個同名的佇列，即使佇列有不同的佇列處置。 IBM MQ 容許有兩個同名 (在步驟 第 285 頁的『2』 中) 的佇列，但您無法使用這些佇列。

- 將佇列從一個連結機能結構移至另一個連結機能結構
- 將非共用佇列移至共用佇列
- 將共用佇列移至非共用佇列

將佇列從一個連結機能結構移至另一個連結機能結構

若要將佇列及其訊息從一個 CF 結構移至另一個 CF 結構，請使用 MQSC MOVE QLOCAL 指令。 當您已識別要移至新 CF 結構的一或多個佇列時，請使用下列程序來移動每一個佇列:

1. 請確定您要移動的佇列未被任何應用程式使用，亦即，佇列共用群組中所有佇列管理程式上的佇列屬性 IPPROCS 及 OPPROCS 均為零。
2. 透過變更佇列定義以停用 MQPUT ，防止應用程式將訊息放置在佇列上。 將佇列定義變更為 PUT (DISABLED)。
3. 使用下列指令，定義與所移動佇列具有相同屬性的暫時佇列:

```
DEFINE QL(TEMP_QUEUE) LIKE(QUEUE_TO_MOVE) PUT(ENABLED) GET(ENABLED) QSGDISP(QMGR)
```

   註: 如果此暫時佇列存在先前的執行，請在執行定義之前先刪除它。
4. 使用下列指令，將訊息移至暫時佇列:

```
MOVE QLOCAL(QUEUE_TO_MOVE) TOQLOCAL(TEMP_QUEUE)
```

5. 使用下列指令，刪除您要移動的佇列:

```
DELETE QLOCAL(QUEUE_TO_MOVE)
```

6. 使用下列指令，重新定義要移動的佇列，變更 CFSTRUCT 屬性:

```
DEFINE QL(QUEUE_TO_MOVE) LIKE(TEMP_QUEUE) CFSTRUCT(NEW) QSGDISP(SHARED)
```

   重新定義佇列時，它會以步驟 第 285 頁的『3』 中建立的暫時佇列為基礎。
7. 使用下列指令將訊息移回新佇列:

```
MOVE QLOCAL(TEMP) TOQLOCAL(QUEUE_TO_MOVE)
```

8. 不再需要在步驟 第 285 頁的『3』 中建立的佇列。 請使用下列指令來刪除它:

```
DELETE QL(TEMP_QUEUE)
```

9. 如果要移動的佇列定義在 CSQINP2 資料集中，請在 CSQINP2 資料集中變更適當 DEFINE QLOCAL 指令的 CFSTRUCT 屬性。 新增 REPLACE 關鍵字，以取代現有的佇列定義。

第 286 頁的圖 46 顯示將佇列從一個 CF 結構移至另一個 CF 結構的範例工作。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.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)
/*
```

圖 *46:* 將佇列從一個 *CF* 結構移至另一個 *CF* 結構的範例工作

將非共用佇列移至共用佇列

將非共用佇列移至共用佇列的程序與將佇列從一個 CF 結構移至另一個 CF 結構的程序類似 (請參閱 第 285 頁的『將佇列從一個連結機能結構移至另一個連結機能結構』)。 第 286 頁的圖 47 提供範例工作來執行此動作。

註: 請記住，共用佇列上的訊息會受到訊息大小上限、訊息持續性及佇列索引類型的特定限制，因此您可能無法將部分非共用佇列移至共用佇列。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.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)
/*
```

圖 *47:* 將非共用佇列移至共用佇列的範例工作

將共用佇列移至非共用佇列

將共用佇列移至非共用佇列的程序與將佇列從一個 CF 結構移至另一個 CF 結構的程序類似 (請參閱 第 285 頁的『將佇列從一個連結機能結構移至另一個連結機能結構』)。

第 287 頁的圖 48 提供範例工作來執行此動作。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.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)
/*
```

圖 *48:* 將共用佇列移至非共用佇列的範例工作

將非共用佇列移轉至共用佇列

將非共用佇列移轉至共用佇列有兩個階段:

- 移轉佇列共用群組中的第一個 (或唯一) 佇列管理程式
- 移轉佇列共用群組中的任何其他佇列管理程式

移轉佇列共用群組中的第一個 (或唯一) 佇列管理程式

第 286 頁的圖 47 顯示將非共用佇列移至共用佇列的範例工作。 針對每一個需要移轉的佇列執行此動作。

註:

1. 共用佇列上的訊息受限於訊息大小上限、訊息持續性及佇列索引類型的特定限制，因此您可能無法將部分非共用佇列移至共用佇列。
2. 您必須對共用佇列使用正確的索引類型。 如果您將傳輸佇列移轉為共用佇列，則索引類型必須是 MSGID。

如果佇列是空的，或您不需要保留其中的訊息，則移轉佇列會更簡單。 第 287 頁的圖 49 顯示在這些情況下要使用的範例工作。

```
//UTILITY EXEC PGM=CSQUTIL,PARM=('CSQ1')
//STEPLIB DD  DSN=thlqual.SCSQANLE,DISP=SHR
//     DD  DSN=thlqual.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)
/*
```

圖 *49:* 將無訊息的非共用佇列移至共用佇列的範例工作

移轉佇列共用群組中的任何其他佇列管理程式

1. 對於與現有共用佇列沒有相同名稱的每一個佇列，請依照 第 286 頁的圖 47 或 第 287 頁的圖 49 中的說明來移動佇列。
2. 對於與現有共用佇列同名的佇列，請使用 第 288 頁的圖 50 中所示的指令將訊息移至共用佇列。

```
MOVE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR) TOQLOCAL(QUEUE_TO_MOVE)
DELETE QLOCAL(QUEUE_TO_MOVE) QSGDISP(QMGR)
```

圖 *50:* 將訊息從非共用佇列移至現有的共用佇列

暫停與 Db2 的連線

如果您想要將維護或服務套用至與共用佇列相關的 Db2 表格或套件，而不停止佇列管理程式，則必須暫時中斷資料共用群組 (DSG) 中佇列管理程式與 Db2 的連線。

若要執行此作業：

1. 使用 MQSC 指令 SUSPEND QMGR FACILITY (Db2)。
2. 進行連結。
3. 使用 MQSC 指令 RESUME QMGR FACILITY (Db2) 重新連接 Db2

請注意，使用這些指令有一些限制。

小心: 當 Db2 連線暫停時，下列作業將無法使用。 因此，在企業最不忙碌的時期，您需要執行這項工作。

- 存取共用佇列物件以進行管理 (定義、刪除、變更)
- 啟動共用通道
- 將訊息儲存在 Db2 中
- 備份或回復 CFSTRUCT

管理群組物件

請利用這個主題來瞭解如何使用群組物件。

IBM MQ 會自動將群組物件的定義複製到使用它的每一個佇列管理程式的頁集零。 您可以暫時變更定義的副本， IBM MQ 可讓您重新整理儲存庫副本中的頁面集副本。 IBM MQ 一律在啟動時嘗試從儲存庫副本重新整理頁面集副本 (對於通道物件，這是在通道起始程式重新啟動時完成)。 這可確保頁面集副本反映儲存庫上的版本，包括佇列管理程式非作用中時所做的任何變更。

在某些情況下，不會執行重新整理，例如：

- 如果開啟佇列副本，則會變更佇列使用情形的重新整理會失敗。
- 如果佇列副本中有訊息，則會刪除該佇列的重新整理會失敗。

在這些情況下，不會對該副本上執行重新整理，而是對所有其他佇列管理程式上的副本上執行重新整理。 在新增、變更或刪除群組物件之後，以及在佇列管理程式或通道起始程式重新啟動時，檢查並更正副本物件的任何問題。

管理連結機能

請利用這個主題來瞭解如何新增或移除連結機能 (CF) 結構。

本節說明下列作業：

- 第 289 頁的『新增連結機能結構』
- 第 289 頁的『移除連結機能結構』

新增連結機能結構

當您新增連結機能結構時，不需要任何 IBM MQ 動作。 作業 10: 設定連結機能 中設定連結機能的相關資訊說明命名連結機能結構的規則，以及如何在 CFRM 原則資料集中定義結構。

移除連結機能結構

若要移除連結機能結構，請遵循下列程序:

- 使用下列指令，以取得使用您要刪除之連結機能結構的所有佇列清單:

```
DISPLAY QUEUE(*) QSGDISP(SHARED) CFSTRUCT(structure-name)
```

- 刪除所有使用該結構的佇列。
- 依序停止並重新啟動佇列共用群組中的每一個佇列管理程式，以中斷 IBM MQ 及 Db2 與結構的連線，並刪除其相關資訊。 (您不需要一次停止所有佇列管理程式; 一次一個。)
- 從 CFRM 原則資料集中移除結構定義，然後執行 IXCMIAPU 公用程式。 (這與 作業 10: 設定連結機能中說明的自訂作業 10 (設定連結機能) 相反。)

回復及重新啟動

請利用這個主題來瞭解 IBM MQ 所使用的回復和重新啟動機制。

正在重新啟動 IBM MQ

在佇列管理程式終止之後，視佇列管理程式終止的方式而定，需要不同的重新啟動程序。 請利用這個主題來瞭解您可以使用的不同重新啟動程序。

本主題包含如何在下列情況下重新啟動佇列管理程式的相關資訊:

- 第 289 頁的『在正常關機之後重新啟動』
- 第 289 頁的『在異常終止之後重新啟動』
- 第 290 頁的『如果您遺失頁集，請重新啟動』
- 第 290 頁的『如果您遺失日誌資料集，則會重新啟動』
- 如果您遺失 CF 結構，請重新啟動

在正常關機之後重新啟動

如果已使用 STOP QMGR 指令停止佇列管理程式，則系統會依序完成其工作，並在停止之前取得終止檢查點。 當您重新啟動佇列管理程式時，它會使用來自系統檢查點及回復日誌的資訊來判定關機時的系統狀態。

若要重新啟動佇列管理程式，請發出 START QMGR 指令，如 第 232 頁的『啟動及停止佇列管理程式』中所述。

在異常終止之後重新啟動

IBM MQ 會自動偵測在正常關機或異常終止之後重新啟動。

在異常終止之後啟動佇列管理程式不同於在發出 STOP QMGR 指令之後啟動佇列管理程式。 如果佇列管理程式異常終止，則會在無法完成其工作或取得終止檢查點的情況下終止。

若要重新啟動佇列管理程式，請發出 START QMGR 指令，如 第 232 頁的『啟動及停止佇列管理程式』中所述。 當您在異常終止之後重新啟動佇列管理程式時，它會使用日誌中的資訊重新整理其在終止時狀態的知識，並通知您各種作業的狀態。

通常，重新啟動程序會解決所有不一致狀態。 但是，在某些情況下，您必須採取特定步驟來解決不一致。 這說明於第 301 頁的『手動回復工作單元』。

如果您遺失頁集，請重新啟動

如果您遺失頁集，則需要從備份副本還原它們，然後才能重新啟動佇列管理程式。 這說明於第 278 頁的『如何備份及回復頁集』。

在這些情況下，由於媒體回復所需的時間長度，佇列管理程式可能需要很長的時間來重新啟動。

如果您遺失日誌資料集，則會重新啟動

在停止佇列管理程式 (使用 STOP QMGR 指令) 之後，如果日誌的兩個副本都遺失或損壞，您可以重新啟動佇列管理程式，但前提是您有一組一致的頁集 (使用 方法 1: 完整備份 產生)。

請遵循下列程序:

1. 定義新的頁面集，以對應佇列管理程式中的每一個現有頁面集。 如需頁集定義的相關資訊，請參閱 作業 15: 定義頁集 。

   請確定每一個新頁集都大於對應的來源頁集。
2. 使用 CSQUTIL 的 FORMAT 函數來格式化目的地頁集。 如需詳細資料，請參閱 格式化頁面集 。
3. 使用 CSQUTIL 的 RESETPAGE 函數來複製現有頁集或就地重設頁集，並重設每一頁中的日誌 RBA。 如需此功能的相關資訊，請參閱 複製頁集並重設日誌 。
4. 使用 CSQJU003 重新定義佇列管理程式日誌資料集及 BSDS (請參閱 變更日誌庫存公用程式)。
5. 使用新的頁集來重新啟動佇列管理程式。 若要執行此動作，請執行下列其中一項:
   - 請變更佇列管理程式啟動型作業程序，以參照新的頁集。 如需相關資訊，請參閱 作業 6: IBM MQ 佇列管理程式的建立程序 。
   - 使用「存取方法服務」來刪除舊的頁面集，然後重新命名新的頁面集，讓它們與舊的頁面集同名。

請注意: 在刪除任何 IBM MQ 頁集之前，請確定您已建立必要的備份副本。

如果佇列管理程式是佇列共用群組的成員，則遺失或損壞的日誌通常不會影響 GROUP 及 SHARED 物件定義。 不過，如果遺失或損壞的日誌所涵蓋的工作單元中涉及任何共用佇列訊息，則無法預期對這類未確定的訊息的影響。

註: 如果日誌已損壞，且佇列管理程式是佇列共用群組的成員，則回復共用持續訊息的能力可能會遺失。 針對具有 RECOVER (YES) 屬性的所有 CF 結構，在佇列共用群組中的另一個作用中佇列管理程式上立即發出 BACKUP CFSTRUCT 指令。

如果您遺失 CF 結構，請重新啟動

如果您失去 CF 結構，則不需要重新啟動，因為佇列管理程式不會終止。

z/OS 替代站台回復

您可以回復單一佇列管理程式或佇列共用群組，或考量磁碟鏡映。

如需詳細資料，請參閱下列各節:

- 在替代站台回復單一佇列管理程式
- 回復佇列共用群組。
  - CF 結構媒體回復
  - 備份主要站台上的佇列共用群組
  - 回復替代網站上的佇列共用群組
- 使用磁碟鏡映

回復替代站台上的單一佇列管理程式

如果完全失去 IBM MQ 運算中心，您可以在回復站台的另一個佇列管理程式或佇列共用群組上回復。 (如需佇列共用群組的替代站台回復程序，請參閱 第 294 頁的『在替代站台回復佇列共用群組』 。)

若要在回復站台的另一個佇列管理程式上回復，您必須定期備份頁集及日誌。 與所有資料回復作業一樣，災難回復的目標是盡可能減少資料、工作量處理 (更新) 及時間。

在回復站台:

- 回復佇列管理程式 **必須** 具有與遺失佇列管理程式相同的名稱。
- 每一個回復佇列管理程式上使用的系統參數模組 (例如， CSQZPARM) 必須包含與對應的遺失佇列管理程式相同的參數。

當您完成此動作時，請重新建立所有佇列管理程式，如下列程序中所述。 這可用來在單一佇列管理程式的回復站台執行災難回復。 它假設所有可用的項目如下:

- 在主要網站上正常執行所建立的保存日誌及 BSDS 副本 (作用中日誌將與主要網站上的佇列管理程式一起遺失)。
- 來自主要網站之佇列管理程式的頁面集副本，其經歷時間與可用的最新保存日誌副本相同或更舊。

您可以對作用中及保存日誌使用雙重記載，在此情況下，您需要將 BSDS 更新項目套用至這兩個副本:

1. 定義新的頁面集資料集，並使用主要網站的頁面集副本中的資料來載入它們。
2. 定義新的作用中日誌資料集。
3. 定義新的 BSDS 資料集，並使用 Access Method Services REPRO 將 最新 保存的 BSDS 複製到其中。
4. 使用列印日誌對映公用程式 CSQJU004 來列印來自這個最新 BSDS 的資訊。 在保存此 BSDS 時，您所擁有的最新保存日誌會被截斷為作用中日誌，而不會顯示為保存日誌。 記錄此日誌的 STARTRBA 和 ENDRBA。
5. 使用變更日誌庫存公用程式 CSQJU003，利用步驟 第 291 頁的『4』 中所記錄的 STARTRBA 和 ENDRBA ，在您剛還原的 BSDS 中登錄這個最新的保存日誌資料集。
6. 使用 CSQJU003 的 DELETE 選項，從 BSDS 移除所有作用中日誌資訊。
7. 使用 CSQJU003 的 NEWLOG 選項將作用中日誌新增至 BSDS ，不要指定 STARTRBA 或 ENDRBA。
8. 使用 CSQJU003 將重新啟動控制記錄新增至 BSDS。 指定 `CRESTART CREATE,ENDRBA=highrba`，其中 `highrba` 是可用的最新保存日誌 (在步驟 第 291 頁的『4』 中找到) 的高 RBA ，加上 1。

   BSDS 現在說明所有作用中日誌都是空的、您有可用的所有保存日誌，且日誌結尾之後沒有任何檢查點。
9. 使用 START QMGR 指令重新啟動佇列管理程式。 在起始設定期間，會發出操作員回覆訊息，如下所示:

   ```
   CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
   REPLY Y TO CONTINUE, N TO CANCEL
   ```

   輸入 Y 以啟動佇列管理程式。 佇列管理程式會啟動，並將資料回復至 CRESTART 陳述式中指定的 ENDRBA。

如需使用 CSQJU003 及 CSQJU004 的相關資訊，請參閱 使用 IBM MQ 公用程式 。

第 292 頁的圖 51 顯示步驟 6、7 及 8 之 CSQJU003 的輸入陳述式範例:

```
* 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
```

圖 *51: CSQJU003* 的輸入陳述式範例

在回復網站上重新啟動通道起始程式時需要考量的事項，與使用 ARM 在不同 z/OS 映像檔上重新啟動通道起始程式時所面臨的事項類似。 如需相關資訊，請參閱第 299 頁的『在 IBM MQ 網路中使用 ARM』。 您的回復策略也應該涵蓋回復 IBM MQ 產品程式庫，以及使用 IBM MQ (例如 CICS) 的應用程式設計環境。

變更日誌庫存公用程式 (CSQJU003) 的其他功能也可以在災難回復實務範例中使用。 HIGHRBA 函數容許更新引導資料集內最高 RBA 寫入值及最高 RBA 卸載值。 CHECKPT 功能容許在 BSDS 中新增新的檢查點佇列記錄或刪除現有的檢查點佇列記錄。

請注意: 這些函數可能會影響 IBM MQ 資料的完整性。 僅在 IBM 服務人員的指引下，在災難回復實務範例中使用它們。

快速複製技術

如果在佇列管理程式凍結時建立所有頁面集和日誌的副本，則副本將是一個一致集，可用來在替代網站上重新啟動佇列管理程式。 它們通常會啟用更快速的佇列管理程式重新啟動，因為幾乎沒有要執行的媒體回復。

請使用 SUSPEND QMGR LOG 指令來凍結佇列管理程式。 此指令會將緩衝池清除至頁集，取得檢查點，並停止任何進一步的日誌寫入活動。 一旦暫停日誌寫入活動，佇列管理程式實際上會凍結，直到您發出 RESUME QMGR LOG 指令為止。 當佇列管理程式凍結時，可以複製頁面集和日誌。

透過使用複製工具 (例如 FLASHCOPY 或 SNAPSHOT) 來快速複製頁集和日誌，可將佇列管理程式凍結的時間縮短至最短。

不過，在佇列共用群組內， SUSPEND QMGR LOG 指令可能不是很好的解決方案。 若要生效，日誌副本必須全部包含相同的回復時間點，這表示必須同時在佇列共用群組內的所有佇列管理程式上發出 SUSPEND QMGR LOG 指令，因此整個佇列共用群組將凍結一段時間。

回復佇列共用群組

如果發生主要站台災難，您可以使用主要站台中的備份資料集，在遠端站台上重新啟動佇列共用群組。 若要回復佇列共用群組，您需要協調佇列共用群組中所有佇列管理程式的回復，並與其他資源 (主要是 Db2) 協調。 本節詳細說明這些作業。

- CF 結構媒體回復
- 備份主要站台上的佇列共用群組
- 回復替代網站上的佇列共用群組

CF 結構媒體回復

用來在共用佇列上保留持續訊息之 CF 結構的媒體回復，取決於是否有媒體的備份，可透過套用已記載的更新項目來回復。 使用 MQSC BACKUP CFSTRUCT 指令定期備份 CF 結構。 共用佇列 (MQGET 及 MQPUT) 的所有更新項目都會寫入執行更新之佇列管理程式的日誌中。 若要執行 CF 結構的媒體回復，您必須從已使用該 CF 結構的 **所有** 佇列管理程式的日誌中，將已記載的更新項目套用至該備份。 當您使用 MQSC RECOVER CFSTRUCT 指令時， IBM MQ 會自動合併相關佇列管理程式中的日誌，並將更新項目套用至最新備份。

CF 結構備份會寫入處理 BACKUP CFSTRUCT 指令之佇列管理程式的日誌中，因此沒有要收集並傳輸至替代站台的其他資料集。

備份主要站台上的佇列共用群組

在主要站台上，您需要定期建立一組一致的備份，可在發生災難時用來在替代站台上重建佇列共用群組。 對於單一佇列管理程式，回復可以是任意時間點，通常是遠端站台可用的日誌結尾。 不過，如果持續訊息已儲存在共用佇列上，則必須合併佇列共用群組中所有佇列管理程式的日誌，才能回復共用佇列，因為佇列共用群組中的任何佇列管理程式可能已在佇列上執行更新 (MQPUT 或 MQGET)。

若要回復佇列共用群組，您需要建立一個在所有佇列管理程式的日誌資料日誌範圍內的復原點。 不過，因為您只能從日誌 **轉遞** 回復媒體，所以這個時間點必須在發出 BACKUP CFSTRUCT 指令之後，以及執行任何頁集備份之後。 (通常，回復的時間點可能對應於工作日或週的結束。)

下圖顯示佇列共用群組中兩個佇列管理程式的時間表。 對於每一個佇列管理程式，會建立頁集的模糊備份 (請參閱 方法 2: 模糊備份)。 在佇列管理程式 A 上，發出 BACKUP CFSTRUCT 指令。 隨後，會在每一個佇列管理程式上發出 ARCHIVE LOG 指令來截斷作用中日誌，並將它從佇列管理程式離線複製到媒體，它可以傳輸至替代站台。 日誌結尾會識別發出 ARCHIVE LOG 指令的時間，因此會標示替代站台通常可用的日誌資料範圍。 回復的時間點必須介於任何頁集或 CF 結構備份的結尾，以及替代位置可用的日誌的最早結尾之間。

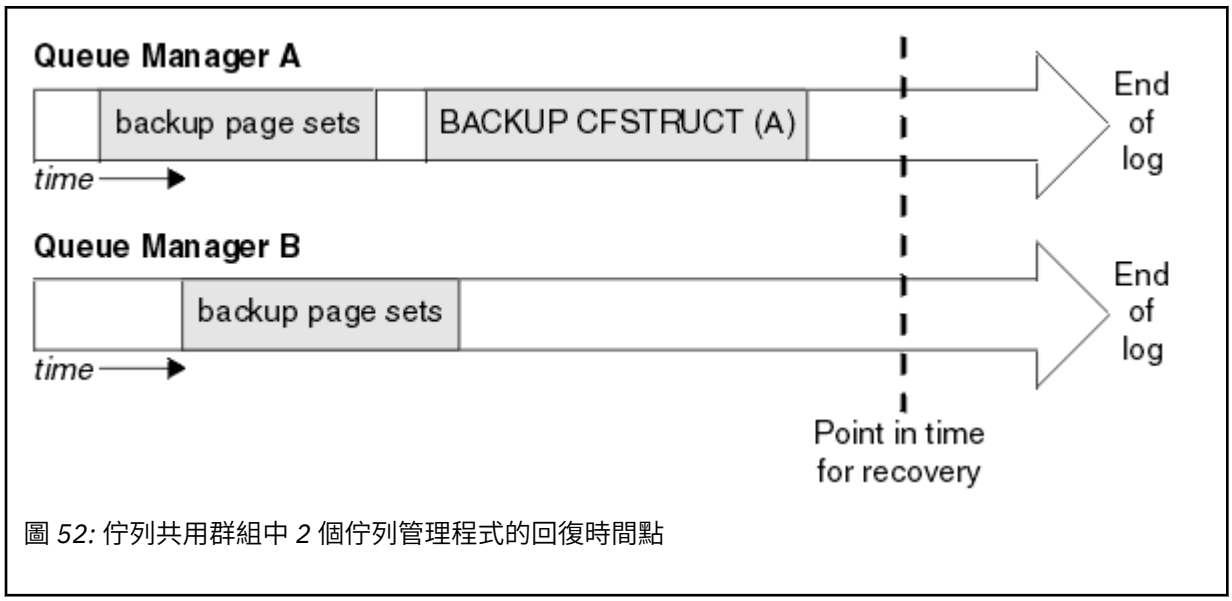


圖 *52:* 佇列共用群組中 *2* 個佇列管理程式的回復時間點

IBM MQ 會在 Db2 的表格中記錄與 CF 結構備份相關聯的資訊。 視您的需求而定，您可能想要將 IBM MQ 回復的復原點與 Db2 的復原點進行協調，或者取得 IBM MQ CSQ.ADMIN_B_STRBACKUP 表格。

若要準備回復，請執行下列動作:

1. 為佇列共用群組中的每一個佇列管理程式建立頁面集備份。
2. 對具有 RECOVER (YES) 屬性的每一個 CF 結構發出 BACKUP CFSTRUCT 指令。 您可以從單一佇列管理程式或從佇列共用群組內的不同佇列管理程式發出這些指令，以平衡工作量。
3. 完成所有備份之後，請發出 ARCHIVE LOG 指令來切換作用中日誌，並建立佇列共用群組中每一個佇列管理程式的日誌和 BSDS 副本。

4. 傳輸頁面集備份、保存日誌、佇列共用群組中所有佇列管理程式的保存 BSDS，以及您選擇的 Db2 備份資訊 (離站)。

在替代站台回復佇列共用群組

您需要先準備環境，才能回復佇列共用群組:

1. 當您安裝佇列共用群組時，如果您在連結機能中有來自實際啟動的舊資訊，則需要先清除下列項目:

   註: 如果您在連結機能中沒有舊資訊，則可以省略此步驟。

   a. 輸入下列 z/OS 指令，以顯示此佇列共用群組的 CF 結構:

   ```
   D XCF,STRUCTURE,STRNAME= qsgname
   ```

   b. 對於以佇列共用群組名稱開頭的所有結構，請使用 z/OS 指令 SETXCF FORCE CONNECTION 來強制關閉這些結構的連線:

   ```
   SETXCF FORCE,CONNECTION,STRNAME= strname,CONNAME=ALL
   ```

   c. 針對每一個結構使用下列指令，刪除所有 CF 結構:

   ```
   SETXCF FORCE,STRUCTURE,STRNAME= strname
   ```

2. 還原 Db2 系統及資料共用群組。
3. 回復 CSQ.ADMIN_B_STRBACKUP 表格，以便它包含在主要網站上取得的最新結構備份的相關資訊。

   註: STRBACKUP 表格必須包含最新的結構備份資訊。 較舊的結構備份資訊可能需要您因最近的 DISPLAY USAGE TYPE (DATASET) 指令所提供的資訊而捨棄的資料集，這表示已回復的 CF 結構將不會包含正確的資訊。
4. 針對佇列共用群組中的每個佇列管理程式，執行 CSQ5PQSG 公用程式的 ADD QMGR 指令。 這會還原每一個佇列管理程式的 XCF 群組項目。

   在此實務範例中執行公用程式時，下列訊息是正常的:

```
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
```

如果要回復佇列共用群組中的佇列管理程式，請執行下列動作:

1. 定義新的頁面集資料集，並使用主要網站的頁面集副本中的資料來載入它們。
2. 定義新的作用中日誌資料集。
3. 定義新的 BSDS 資料集，並使用 Access Method Services REPRO 將 最新 保存的 BSDS 複製到其中。
4. 使用列印日誌對映公用程式 CSQJU004 來列印來自這個最新 BSDS 的資訊。 在保存此 BSDS 時，您所擁有的最新保存日誌會被截斷為作用中日誌，而不會顯示為保存日誌。 記錄此日誌的 STARTRBA、STARTLRSN、ENDRBA 及 ENDRSN 值。
5. 使用變更日誌庫存公用程式 CSQJU003，利用步驟 第 294 頁的『4』 中所記錄的值，在您剛還原的 BSDS 中登錄這個最新的保存日誌資料集。
6. 使用 CSQJU003 的 DELETE 選項，從 BSDS 移除所有作用中日誌資訊。
7. 使用 CSQJU003 的 NEWLOG 選項將作用中日誌新增至 BSDS，不要指定 STARTRBA 或 ENDRBA。

8. 計算佇列共用群組的 *recoverylrsn* 。 *recoverylrsn* 是佇列共用群組中所有佇列管理程式的最低 ENDRSNs (如步驟 第 294 頁的『4』 中所記錄)，減 1。 比方說，如果佇列共用群組中有兩個佇列管理程式，其中一個是 B713 3C72 22C5，另一個是 B713 3D45 2123 ，則 *recoverylrsn* 是 B713 3C72 22C4。
9. 使用 CSQJU003 將重新啟動控制記錄新增至 BSDS。 指定：

```
CRESTART CREATE,ENDLRSN= recoverylrsn
```

   其中 *recoverylrsn* 是您在步驟 第 295 頁的『8』 中記錄的值。

   BSDS 現在說明所有作用中日誌都是空的、您有可用的所有保存日誌，且日誌結尾之後沒有任何檢查點。

   您必須針對佇列共用群組內的每一個佇列管理程式，將 CRESTART 記錄新增至 BSDS。
10. 使用 START QMGR 指令重新啟動佇列共用群組中的每一個佇列管理程式。 在起始設定期間，會發出操作員回覆訊息，如下所示：

```
CSQJ245D +CSQ1 RESTART CONTROL INDICATES TRUNCATION AT RBA highrba.
REPLY Y TO CONTINUE, N TO CANCEL
```

    回覆 Y 以啟動佇列管理程式。 佇列管理程式會啟動，並將資料回復至 CRESTART 陳述式中指定的 ENDRBA。

    在 IBM WebSphere MQ 7.0.1 以及更新版本中，第一個啟動的佇列管理程式可以為佇列共用群組的其他成員及其自己的成員重建管理結構分割區，而且在此階段不再需要重新啟動佇列共用群組中的每一個佇列管理程式。
11. 當已重建所有佇列管理程式的管理結構資料時，請針對每一個 CF 應用程式結構發出 RECOVER CFSTRUCT 指令。

    如果您針對單一佇列管理程式上的所有結構發出 RECOVER CFSTRUCT 指令，則日誌合併處理程序只會執行一次，因此比針對每一個 CF 結構在不同佇列管理程式上發出指令來得快，其中每一個佇列管理程式都必須執行日誌合併步驟。

在佇列共用群組中使用條件式重新啟動處理程序時， IBM WebSphere MQ 7.0.1 以及更新版本的佇列管理程式會執行對等節點管理重建，請檢查對等節點 BSDS 是否包含與其自己相同的 CRESTART LRSN。 這是為了確保重建的管理結構的完整性。 因此，在群組的任何成員下一次無條件重新啟動之前，重新啟動 QSG 中的其他對等節點是很重要的，因此它們可以處理自己的 CRESTART 資訊。

使用磁碟鏡映

現在許多安裝都使用磁碟鏡映技術 (例如 IBM Metro Mirror (舊稱為 PPRC)) ，在替代站台上建立資料集的同步副本。 在這種情況下，由於替代網站上的 IBM MQ 頁面集和日誌實際上與主要網站上的那些頁面集和日誌完全相同，因此許多詳細的步驟變得不需要。 在使用這類技術的情況下，在替代網站上重新啟動佇列共用群組的步驟可以彙總為：

- 清除替代位置上的 IBM MQ CF 結構。 (這些通常包含先前任何災難回復作業的剩餘資訊)。
- 還原 Db2 系統及 IBM MQ 佇列共用群組所使用資料庫中的所有表格。
- 重新啟動佇列管理程式。 在 IBM WebSphere MQ 7.0.1 之前，必須重新啟動佇列共用群組中定義的每一個佇列管理程式，因為在佇列管理程式重新啟動期間，每一個佇列管理程式都會回復其自己的管理結構分割區。 重新啟動每一個佇列管理程式之後，可以再次關閉不在其「起始」LPAR 上的那些佇列管理程式。 對於 IBM WebSphere MQ 7.0.1 以及更新版本，第一個已啟動的佇列管理程式會為佇列共用群組的其他成員及其自己的成員重新建置管理結構分割區，而且不再需要重新啟動佇列共用群組中的每一個佇列管理程式。
- 重建管理結構之後，請回復應用程式結構。

重新起始設定佇列管理程式

如果佇列管理程式已異常終止，您可能無法重新啟動它。 這可能是因為您的頁面集或日誌已遺失、截斷或毀損。 如果發生這種情況，您可能必須重新起始設定佇列管理程式 (執行冷啟動)。

請注意

只有在無法以任何其他方式重新啟動佇列管理程式時，才會執行冷啟動。 執行冷啟動可讓您回復佇列管理程式及物件定義; 您 **無法** 回復訊息資料。 在執行此動作之前，請確認本主題中說明的任何其他重新啟動實務範例都不適用於您。

當您重新啟動時，所有 IBM MQ 物件都已定義且可供使用，但沒有訊息資料。

註: 當佇列管理程式是叢集的一部分時，請勿重新起始設定它。 您必須先從叢集中移除佇列管理程式 (在叢集中的其他佇列管理程式上使用 RESET CLUSTER 指令)，然後重新起始設定它，最後將它重新引進叢集作為新的佇列管理程式。

這是因為在重新起始設定期間，佇列管理程式 ID (QMID) 會變更，因此必須從叢集中移除任何具有舊佇列管理程式 ID 的叢集物件。

如需進一步資訊，請參閱下列各節:

- 重新起始設定不在佇列共用群組中的佇列管理程式
- 重新起始設定佇列共用群組中的佇列管理程式

重新起始設定不在佇列共用群組中的佇列管理程式

若要重新起始設定佇列管理程式，請遵循下列程序:

1. 準備要在重新啟動佇列管理程式時使用的物件定義陳述式。 若要執行此動作，請執行下列任一動作:
   - 如果頁集零可用，請使用 CSQUTIL SDEFS 函數 (請參閱 產生 IBM MQ 定義指令清單)。 您必須取得所有物件類型 (鑑別資訊物件、CF 結構、通道、名稱清單、處理程序、佇列及儲存類別) 的定義。
   - 如果無法使用頁集零，請使用前次備份物件定義時的定義。
2. 重新定義佇列管理程式資料集 (在完成步驟 第 296 頁的『1』 之前，請不要這麼做)。

   如需相關資訊，請參閱 建立引導及日誌資料集 及 定義頁面集 。
3. 使用新定義且已起始設定的日誌資料集、BSDS 及頁面集，重新啟動佇列管理程式。 使用您在步驟 第 296 頁的『1』 中建立的物件定義輸入陳述式，作為 CSQINP2 起始設定輸入資料集的輸入。

重新起始設定佇列共用群組中的佇列管理程式

在佇列共用群組中，重新起始設定佇列管理程式更為複雜。 由於頁集或日誌問題，可能需要重新起始設定一或多個佇列管理程式，但 Db2 或要處理的連結機能也可能有問題。 因此，有許多替代方案:

冷啟動 (cold start)
重新起始設定整個佇列共用群組包括強制所有連結機能結構、從 Db2 中清除佇列共用群組的所有物件定義、刪除或重新定義日誌及 BSDS，以及格式化佇列共用群組中所有佇列管理程式的頁集。

保留的共用定義
刪除或重新定義日誌和 BSDS，佇列共用群組中所有佇列管理程式的格式頁集，並強制所有連結機能結構。 重新啟動時，將會刪除所有訊息。 佇列管理程式會重建 COPY 物件，對應於仍然存在於 Db2 資料庫中的 GROUP 物件。 任何共用佇列仍然存在且可以使用。

已重新起始設定單一佇列管理程式
刪除或重新定義日誌和 BSDS，以及單一佇列管理程式的格式頁集 (這會刪除其所有專用物件和訊息)。 重新啟動時，佇列管理程式會重建 COPY 物件，這些物件對應於仍然存在於 Db2 資料庫中的 GROUP 物件。 任何共用佇列仍然存在，就像它們上的訊息一樣，可以使用。

佇列共用群組的復原點回復
這是替代站台災難回復實務範例。

共用物件會回復至 Db2 回復 (在 A Db2 系統失效 中說明) 所達到的復原點。 每一個佇列管理程式都可以從替代站台可用的備份副本回復至某個復原點。

持續訊息可以在佇列共用群組中使用，並且可以使用 MQSC RECOVER CFSTRUCT 指令來回復。 請注意，這個指令會回復到失敗的時間。 不過，不會回復非持續性共用佇列訊息; 除非您使用 CSQUTIL 公用程式的 COPY 功能獨立製作備份副本，否則它們會遺失。

不需要嘗試將每一個佇列管理程式還原至相同的時間點，因為不同佇列管理程式上的本端物件 (實際上正在回復的物件) 之間沒有交互相依關係，且重新啟動時佇列管理程式與 Db2 重新同步會根據佇列管理程式在佇列管理程式上的需要建立或刪除 COPY 物件。

使用 z/OS Automatic Restart Manager (ARM)

請利用這個主題來瞭解如何使用 ARM 來自動重新啟動佇列管理程式。

本節包含下列主題的相關資訊:

- 第 297 頁的『什麼是 ARM?』
- 第 297 頁的『ARM 原則』
- 第 299 頁的『在 IBM MQ 網路中使用 ARM』

什麼是 ARM?

z/OS Automatic Restart Manager (ARM) 是一種 z/OS 回復功能，可改善佇列管理程式的可用性。 當工作或作業失敗，或執行它的系統失敗時， ARM 可以重新啟動工作或作業，不需要操作員介入。

如果佇列管理程式或通道起始程式失敗， ARM 會在相同的 z/OS 映像檔上重新啟動它。 如果 z/OS 以及整個相關子系統和應用程式群組都失敗， ARM 可以在 Sysplex 內的另一個 z/OS 映像檔上以預先定義的順序自動重新啟動所有失敗的系統。 這稱為 跨系統重新啟動。

只有在異常情況下，才由 ARM 重新啟動通道起始程式。 如果 ARM 重新啟動佇列管理程式，請從 CSQINP2 起始設定資料集重新啟動通道起始程式 (請參閱 第 299 頁的『在 IBM MQ 網路中使用 ARM』)。

如果 z/OS 失敗，您可以使用 ARM ，在 Sysplex 內的不同 z/OS 映像檔上重新啟動佇列管理程式。 第 299 頁的『在 IBM MQ 網路中使用 ARM』 中說明不同 z/OS 映像檔上 IBM MQ ARM 重新啟動的網路含意。

若要啟用自動重新啟動，請執行下列動作:

- 設定 ARM 連結資料集。
- 在 *ARM* 原則中定義您要 z/OS 執行的自動重新啟動動作。
- 啟動 ARM 原則。

此外， IBM MQ 必須在啟動時向 ARM 登錄 (這會自動發生)。

註: 如果您想要自動重新啟動不同 z/OS 映像檔中的佇列管理程式，您必須在每一個 z/OS 映像檔中，將每一個佇列管理程式定義為可能重新啟動該佇列管理程式的子系統，並使用 Sysplex 層面唯一的四個字元子系統名稱。

ARM 連結資料集

在啟動您想要 ARM 支援的任何佇列管理程式之前，請確定您已定義 ARM 所需的連結資料集，且它們在線上且在作用中。 如果在佇列管理程式啟動時無法使用這兩個資料集，則 IBM MQ 自動 ARM 登錄會失敗。 在此情況下， IBM MQ 會假設缺少連結資料集，表示您不想要 ARM 支援，且會繼續起始設定。

如需 ARM 連結資料集的相關資訊，請參閱 *z/OS MVS Setting up a Sysplex* 手冊。

ARM 原則

「自動重新啟動管理程式」原則是使用者定義的規則，可控制 ARM 函數，可控制佇列管理程式的任何重新啟動。

ARM 函數由使用者定義的 *ARM* 原則控制。 執行由 ARM 重新啟動之佇列管理程式實例的每一個 z/OS 映像檔，必須連接至具有作用中 ARM 原則的 ARM 聯結資料集。

IBM 提供預設 ARM 原則。 您可以定義新的原則，或使用 z/OS 隨附的 管理資料公用程式 (IXCMIAPU) 來置換原則預設值。 *z/OS MVS Setting up a Sysplex* 手冊說明此公用程式，並包含如何定義 ARM 原則的完整詳細資料。

第 298 頁的圖 53 顯示 ARM 原則的範例。 如果佇列管理程式失敗或整個系統失敗，此範例原則會重新啟動 Sysplex 內的任何佇列管理程式。

```
//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(SYSMQMGRMQ*)  /* These jobs to be restarted by ARM */
/*
```

圖 *53: ARM* 原則範例

如需相關資訊，請參閱:

- 定義 ARM 原則
- 啟動 ARM 原則
- 向 ARM 登錄

定義 ARM 原則

設定 ARM 原則，如下所示:

- 為每一個佇列管理程式實例定義 RESTART_GROUPs ，這些佇列管理程式實例也包含連接至該佇列管理程式實例的任何 CICS 或 IMS 子系統。 如果您使用子系統命名慣例，則可能可以使用 '?' 以及元素名稱中的 '*' 萬用字元，以最少的定義工作量來定義 RESTART_GROUPS。
- 指定通道起始程式的 TERMTYPE (ELEMTERM) ， 以指出只有在通道起始程式失敗且 z/OS 映像檔未失敗時，才會重新啟動它們。
- 指定佇列管理程式的 TERMTYPE (ALLTERM) ， 以指出如果佇列管理程式失敗或 z/OS 映像檔失敗，將會重新啟動它們。
- 同時為佇列管理程式及通道起始程式指定 RESTART_METHOD (BOTH ， PERSIST)。 這會告知 ARM 在前次啟動期間使用它所儲存的 JCL 來重新啟動 (在系統符號解析之後)。 不論個別元素是否失敗，或 z/OS 映像檔是否失敗，都會告知 ARM 來執行這個動作。
- 接受所有其他 ARM 原則選項的預設值。

啟動 ARM 原則

若要啟動自動重新啟動管理原則，請發出下列 z/OS 指令:

```
SETXCF START,POLICY,TYPE=ARM,POLNAME= mypol
```

啟動原則時，所有連接至 ARM 聯結資料集的系統都會使用相同的作用中原則。

請使用 SETXCF STOP 指令來停用自動重新啟動。

向 ARM 登錄

在佇列管理程式啟動期間， IBM MQ 會自動登錄為 *ARM* 元素 (取決於 ARM 可用性)。 除非要求不取消登錄，否則它會在其關閉階段期間取消登錄。

啟動時，佇列管理程式會判斷 ARM 是否可用。 如果是， IBM MQ 會使用名稱 SYSMQMGR *ssid* 登錄，其中 *ssid* 是四個字元的佇列管理程式名稱，而 SYSMQMGR 是元素類型。

STOP QMGR MODE (QUIESCE) 及 STOP QMGR MODE (FORCE) 指令會從 ARM 取消登錄佇列管理程式 (如果在啟動時已向 ARM 登錄)。 這會防止 ARM 重新啟動此佇列管理程式。 STOP QMGR MODE (RESTART) 指令不會從 ARM 取消登錄佇列管理程式，因此它適合立即自動重新啟動。

每一個通道起始程式位址空間都會決定 ARM 是否可用，如果可用，則會以元素名稱 SYSMQCH *ssid* 登錄，其中 *ssid* 是佇列管理程式名稱，而 SYSMQCH 是元素類型。

通道起始程式在正常停止時一律會從 ARM 取消登錄，且只有在異常結束時才會維持登錄。 如果佇列管理程式失敗，通道起始程式一律會取消登錄。

在 *IBM MQ* 網路中使用 *ARM*

您可以設定佇列管理程式，以便在重新啟動佇列管理程式時自動啟動通道起始程式及相關聯的接聽器。

若要確保在 LU 6.2 及 TCP/IP 通訊協定的相同 z/OS 映像檔上完全自動重新啟動佇列管理程式，請執行下列動作:

- 將適當的 START LISTENER 指令新增至 CSQINPX 資料集，以自動啟動接聽器。
- 將適當的 START CHINIT 指令新增至 CSQINP2 資料集，以自動啟動通道起始程式。

若要使用 TCP/IP 或 LU6.2 重新啟動佇列管理程式，請參閱

- 第 299 頁的『使用 TCP/IP 在不同的 z/OS 映像檔上重新啟動』
- 第 300 頁的『在具有 LU 6.2 的不同 z/OS 映像檔上重新啟動』

如需 CSQINP2 及 CSQINPX 資料集的相關資訊，請參閱 作業 13: 自訂起始設定輸入資料集 。

使用 TCP/IP 在不同的 z/OS 映像檔上重新啟動

如果您使用 TCP/IP 作為通訊協定，並且使用虛擬 IP 位址，則可以將這些配置為在其他 z/OS 映像檔上回復，容許連接至該佇列管理程式的通道重新連接，而無需進行任何變更。 否則，只有在您使用叢集或使用「WLM 動態網域名稱系統 (DNS)」邏輯群組名稱連接至佇列共用群組時，才可以在將佇列管理程式移至不同的 z/OS 映像檔之後重新配置 TCP/IP 位址。

- 使用叢集作業時
- 連接至佇列共用群組時

使用叢集作業時

z/OS ARM 會在相同 Sysplex 的不同 z/OS 映像檔上重新啟動佇列管理程式，以回應系統故障; 這個系統與原始 z/OS 映像檔有不同的 TCP/IP 位址。 下列說明在 ARM 重新啟動將佇列管理程式的 TCP/IP 位址移至不同的 z/OS 映像檔之後，如何使用 IBM MQ 叢集來重新指派它。

當用戶端佇列管理程式偵測到佇列管理程式失敗 (作為通道失敗) 時，它會回應將其叢集傳輸佇列上的適當訊息重新配置給管理目標叢集佇列不同實例的不同伺服器佇列管理程式。 不過，它無法重新配置因親緣性限制而連結至原始伺服器的訊息，或因為伺服器佇列管理程式在批次結束處理期間失敗而不確定的訊息。 若要處理這些訊息，請執行下列動作:

1. 配置不同的叢集接收端通道名稱及不同的 TCP/IP 埠給每一個 z/OS 佇列管理程式。 每一個佇列管理程式都需要不同的埠，以便兩個系統可以在 z/OS 映像檔上共用單一 TCP/IP 堆疊。 其中一個是最初在該 z/OS 映像檔上執行的佇列管理程式，另一個是 ARM 在系統失效之後將在該 z/OS 映像檔上重新啟動的佇列管理程式。 在每一個 z/OS 映像檔上配置每一個埠，以便 ARM 可以在任何 z/OS 映像檔上重新啟動任何佇列管理程式。
2. 針對要在通道起始程式啟動期間參照的每一個佇列管理程式及 z/OS 映像檔組合，建立不同的通道起始程式指令輸入檔 (CSQINPX)。

每一個 CSQINPX 檔案都必須包括該佇列管理程式特有的 START LISTENER PORT (port) 指令，以及該佇列管理程式及 z/OS 映像檔組合特有之叢集接收端通道的 ALTER CHANNEL 指令。 ALTER CHANNEL 指令需要將連線名稱設為重新啟動所在之 z/OS 映像檔的 TCP/IP 名稱。 它必須包含特定於重新啟動的佇列管理程式的埠號，作為連線名稱的一部分。

每一個佇列管理程式的啟動 JCL 都可以具有此 CSQINPX 檔案的固定資料集名稱，且每一個 z/OS 映像檔在非共用 DASD 磁區上必須具有每一個 CSQINPX 檔案的不同版本。

如果發生 ARM 重新啟動， IBM MQ 會將已變更的通道定義通告至叢集儲存庫，然後再將它發佈至表示對伺服器佇列管理程式感興趣的所有用戶端佇列管理程式。

用戶端佇列管理程式會將伺服器佇列管理程式失敗視為通道失敗，並嘗試重新啟動失敗的通道。 當用戶端佇列管理程式得知新的伺服器連線名稱時，通道重新啟動會將用戶端佇列管理程式重新連接至重新啟動的伺服器佇列管理程式。 然後，用戶端佇列管理程式可以重新同步化其訊息，解決用戶端佇列管理程式的傳輸佇列上任何不確定的訊息，而正常處理可以繼續進行。

連接至佇列共用群組時

透過 TCP/IP 動態網域名稱系統 (DNS) 邏輯群組名稱連接至佇列共用群組時，通道定義中的連線名稱會指定佇列共用群組的邏輯群組名稱，而不是實體機器的主機名稱或 IP 位址。 當此通道啟動時，它會連接至動態 DNS ，然後連接至佇列共用群組中的其中一個佇列管理程式。 此程序在 使用佇列共用群組來設定 IBM MQ for z/OS 的通訊中有說明。

在映像檔失敗的不太可能事件中，會發生下列其中一項:

- 從 Sysplex 上執行的動態 DNS 取消登錄失敗映像檔上的佇列管理程式。 通道會進入 RETRYING 狀態以回應連線失敗，然後連接至在 Sysplex 上執行的動態 DNS。 動態 DNS 會將入埠要求配置給仍在剩餘映像檔上執行的佇列共用群組的其中一個剩餘成員。
- 如果佇列共用群組中沒有其他佇列管理程式在作用中，且 ARM 會在不同映像檔上重新啟動佇列管理程式和通道起始程式，則群組接聽器會從這個新映像檔向動態 DNS 登錄。 這表示邏輯群組名稱 (來自通道的連線名稱欄位) 會連接至動態 DNS ，然後連接至現在在不同映像檔上執行的相同佇列管理程式。 不需要變更通道定義。

若要進行這種類型的回復，必須注意下列要點:

- 在 z/OS 上，動態 DNS 會在 Sysplex 的其中一個 z/OS 映像檔上執行。 如果此映像檔失敗，則需要配置動態 DNS ，以便在 Sysplex 中有作用中的次要名稱伺服器，作為主要名稱伺服器的替代方案。 如需主要及次要動態 DNS 伺服器的相關資訊，請參閱 *OS/390 SecureWay CS IP* 配置 手冊。
- TCP/IP 群組接聽器可能已在此 z/OS 映像檔上無法使用的特定 IP 位址上啟動。 如果是這樣，則可能需要在新映像檔上的不同 IP 位址上啟動接聽器。 如果您使用虛擬 IP 位址，則可以將這些配置為在其他 z/OS 映像檔上回復，以便不需要變更 START LISTENER 指令。

在具有 LU 6.2 的不同 z/OS 映像檔上重新啟動

如果您只使用 LU 6.2 通訊協定，請執行下列程序，在 Sysplex 內不同 z/OS 映像檔上自動重新啟動佇列管理程式之後啟用網路重新連接:

- 請使用唯一子系統名稱來定義 Sysplex 內的每一個佇列管理程式。
- 使用唯一 LUNAME 來定義 Sysplex 內的每一個通道起始程式。 這同時在佇列管理程式屬性及 START LISTENER 指令中指定。

  註: LUNAME 會命名 APPC 端表格中的登錄，然後將此對映至實際 LUNAME。
- 設定共用 APPC 端表格，供 Sysplex 內每一個 z/OS 映像檔參照。 這應該包含每一個通道起始程式的 LUNAME 的項目。 如需相關資訊，請參閱 *MVS Planning: APPC/MVS Management* 手冊。
- 針對 Sysplex 內的每一個通道起始程式設定 SYS1.PARMLIB 的 APPCPM *xx* 成員，以包含 LUADD 來啟動該通道起始程式的 APPC 端表格項目。 這些成員應該由每一個 z/OS 映像檔共用。 適當的 SYS1.PARMLIB 成員由 z/OS 指令 SET APPC= *xx* 啟動，在不同 z/OS 映像檔上 ARM 重新啟動佇列管理程式 (及其通道起始程式) 期間會自動發出此指令，如下列文字中所述。
- 使用 LU62ARM 佇列管理程式屬性，為每一個通道起始程式指定此 SYS1.PARMLIB 成員的 *xx* 字尾。 這會導致通道起始程式發出必要的 z/OS 指令 SET APPC= *xx* ，以啟動其 LUNAME。

定義 ARM 原則，以便只有在通道起始程式的 z/OS 映像檔保持啟動時失敗時，它才會重新啟動; 與 XCFAS 位址空間相關聯的使用者 ID 必須獲得授權，才能發出 IBM MQ 指令 START CHINIT。 如果通道起始程式的 z/OS 映像檔也失敗，請不要自動重新啟動通道起始程式，請改用 CSQINP2 及 CSQINPX 資料集中的指令來啟動通道起始程式及接聽器。

手動回復工作單元

您可以手動回復佇列共用群組中的工作單元 CICS、 IMS、RRS 或其他佇列管理程式。 您可以使用佇列管理程式指令，來顯示與佇列管理程式的每一個連線相關聯的工作單元狀態。

本主題包含下列主題的相關資訊:

顯示連線和執行緒

您可以使用 DISPLAY CONN 指令來取得佇列管理程式及其相關聯工作單元的連線相關資訊。 您可以顯示作用中的工作單元，以查看目前發生的情況，或查看需要終止以容許佇列管理程式關閉的項目，並且您可以顯示未解決的工作單元，以協助進行回復。

作用中工作單元

若只要顯示作用中的工作單元，請使用

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ ACTIVE)
```

未解決的工作單元

未解決的工作單元 (也稱為「不確定的執行緒」) 是在兩段式確定作業的第二次傳遞中的工作單元。 資源代表其保留在 IBM MQ 中。 如果要顯示尚未解析的工作單元，請使用

```
DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

需要外部介入來解決未解決工作單元的狀態。 這可能只涉及啟動回復協調程式 (CICS、 IMS 或 RRS) ，也可能涉及更多，如下列各節所述。

手動回復 *CICS* 回復單元

使用本主題來瞭解當 CICS 配接器重新啟動時所發生的情況，然後說明如何處理任何產生的未解決回復單元。

當 CICS 配接卡重新啟動時發生的情況

每當連線中斷時，配接器必須在 重新連接處理程序期間經歷 重新啟動階段 。 重新啟動階段會重新同步化資源。 CICS 與 IBM MQ 之間的重新同步會啟用要識別並解決的不確定工作單元。

重新同步的原因可能是:

- 來自分散式佇列元件的明確要求
- 對 IBM MQ 建立連線時的隱含要求

如果連接至 IBM MQ 導致重新同步，則事件順序為:

1. 連線程序會從 IBM MQ 擷取不確定的工作單元 (UOW) ID 清單。
2. UOW ID 會顯示在主控台的 CSQC313I 訊息中。
3. UOW ID 會傳遞至 CICS。
4. CICS 會針對每一個不確定的 UOW ID 起始重新同步作業 (CRSY)。
5. 主控台上會顯示每一個不確定 UOW 的作業結果。

您需要檢查在連接程序期間顯示的訊息:

CSQC313I
顯示 UOW 不確定。

CSQC400I
識別 UOW , 後面接著下列其中一則訊息:

- CSQC402I 或 CSQC403I 顯示已順利解析 UOW (已確定或已取消)。
- CSQC404E、 CSQC405E、 CSQC406E 或 CSQC407E 顯示未解析 UOW。

CSQC409I
顯示已順利解析所有 UOW。

CSQC408I
顯示未順利解析所有 UOW。

CSQC314I
警告不會自動解析以 * 強調顯示的 UOW ID。 這些 UOW 必須在重新啟動時由分散式佇列元件明確解析。

第 302 頁的圖 54 顯示 z/OS 主控台上顯示的一組重新啟動訊息範例。

```
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
```

圖 *54:* 重新啟動訊息範例

CSQC313I 訊息總數應該等於 CSQC402I 加上 CSQC403I 訊息總數。 如果總計不相等，則有連線處理程序無法解析的 UOW。 無法解決的那些 UOW 是由 CICS (例如，冷開機) 或 IBM MQ 的問題或配送佇列作業所造成。 當這些問題已修正時，您可以中斷連線，然後重新連接，以起始另一個重新同步。

或者，您可以使用 RESOLVE INDOUBT 指令及訊息 CSQC400I 中顯示的 UOW ID，自行解決每一個未完成的 UOW。 然後，您必須起始斷線及連接，以清除 CICS 中的 回復單元描述子。 您需要知道 UOW 的正確結果，才能手動解析 UOW。

IBM MQ 會鎖定所有與未解析的 UOW 相關聯的訊息，且沒有任何「批次」、TSO 或 CICS 作業可以存取它們。

如果 CICS 失敗，且需要緊急重新啟動， 請勿 改變 CICS 系統的 GENERIC APPLID。 如果您這樣做，然後重新連接至 IBM MQ，則無法保證 IBM MQ 的資料完整性。 這是因為 IBM MQ 會將新的 CICS 實例視為不同的 CICS (因為 APPLID 不同)。 然後，不確定的解決方案會根據錯誤的 CICS 日誌。

如何手動解析 CICS 回復單元

如果配接卡異常結束，則 CICS 及 IBM MQ 建置不確定清單會動態或在重新啟動期間，視導致異常終止的子系統而定。

註: 如果您使用 DFH$INDB 範例程式來顯示工作單元，您可能會發現它並非一律正確地顯示 IBM MQ UOW。

當 CICS 連接至 IBM MQ 時，可能有一個以上尚未解析的回復單元。

下列其中一則訊息會傳送至主控台:

- CSQC404E
- CSQC405E
- CSQC406E
- CSQC407E
- CSQC408I

如需這些訊息意義的詳細資料，請參閱 CICS 配接器及橋接器訊息 訊息。

CICS 會保留連線啟動期間未解析之回復單元的詳細資料。 當項目不再出現在 IBM MQ 所呈現的清單上時，即會清除該項目。

任何 CICS 無法解析的回復單元都必須使用 IBM MQ 指令來手動解析。 此手動程序很少在安裝內使用，因為只有在作業錯誤或軟體問題導致無法自動解決時才需要。 必須調查在不確定解決期間發現的任何不一致。

若要解析回復單元，請執行下列動作:

1. 使用下列指令，從 IBM MQ 取得回復單元的清單:

```
+CSQ1  DISPLAY CONN( * ) WHERE(UOWSTATE EQ UNRESOLVED)
```

   您會收到下列訊息:

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC85772CBE3E0001)
EXTCONN(C3E2D8C3C7D9F0F94040404040404040)
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
```

對於 CICS 連線， NID 由 CICS 應用程式 ID 和 CICS 在寫入同步點日誌項目時提供的唯一號碼組成。 這個唯一數字儲存在同步點處理時間同時寫入 CICS 系統日誌和 IBM MQ 日誌的記錄中。 在 CICS 中，此值稱為 回復記號。

2. 掃描 CICS 日誌，以找出與特定回復單元相關的項目。

   尋找作業相關安裝的 PREPARE 記錄，其中回復記號欄位 (JCSRMTKN) 等於從網路 ID 取得的值。 網路 ID 由 IBM MQ 在 DISPLAY CONN 指令輸出中提供。

   CICS 日誌中回復單元的 PREPARE 記錄提供 CICS 作業號碼。 使用此數字可以找到此 CICS 作業日誌上的所有其他項目。

   掃描日誌時，您可以使用 CICS 日誌登載列印公用程式 DFHJUP。 如需使用此程式的詳細資料，請參閱 *CICS* 作業和公用程式手冊。

3. 掃描 IBM MQ 日誌，以找出具有與特定回復單元相關之 NID 的記錄。 然後使用此記錄中的 URID 來取得此回復單元的其餘日誌記錄。

   掃描 IBM MQ 日誌時，請注意 IBM MQ 啟動訊息 CSQJ001I 提供此階段作業的啟動 RBA。

   列印日誌記錄程式 (CSQ1LOGP) 可用於該目的。

4. 如果您需要，請在 IBM MQ 中執行不確定的解決方案。

   可以使用 IBM MQ RESOLVE INDOUBT 指令，引導 IBM MQ 採取回復單元的回復動作。

   若要回復與特定 *connection-name* 相關聯的所有執行緒，請使用 NID (*) 選項。

   此指令會產生下列其中一則訊息，指出執行緒是已確定還是已取消:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED
CSQV415I +CSQ1 THREAD network-id ABORT SCHEDULED
```

執行不確定的解決方案時， CICS 和配接器不知道 IBM MQ 用來確定或取消回復單元的指令，因為只會影響 IBM MQ 資源。 不過， CICS 會保留 IBM MQ 無法解決之不確定執行緒的詳細資料。 當呈現的清單是空的，或當清單不包括 CICS 具有詳細資料的回復單元時，即會清除此資訊。

手動回復 *IMS* 回復單元

使用本主題來瞭解當 IMS 配接器重新啟動時所發生的情況，然後說明如何處理任何產生的未解決回復單元。

當 IMS 配接卡重新啟動時發生的情況

每當重新啟動與 IBM MQ 的連線時，在佇列管理程式重新啟動或 IMS /START SUBSYS 指令之後， IMS 會起始下列重新同步處理程序:

1. IMS 會以「確定」或「取消」的解析參數來呈現它認為 IBM MQ IMS 配接卡不確定的工作單元 (UOW) ID 清單 (一次一個)。
2. IMS 配接器會將解析要求傳遞至 IBM MQ ， 並將結果回報給 IMS。
3. 在處理完所有 IMS 解析要求之後， IMS 配接器會從 IBM MQ 取得 IBM MQ 仍處於不確定狀態且由 IMS 系統起始的所有 UOW 的清單。 這些會以訊息 CSQQ008I 向 IMS 主要終端機報告。

註: 當 UOW 不確定時， IBM MQ 會鎖定任何相關聯的 IBM MQ 訊息，且無法供任何應用程式使用。

如何手動解析 IMS 回復單元

當 IMS 連接至 IBM MQ 時， IBM MQ 可能有一個以上不確定的回復單元尚未解決。

如果 IBM MQ 具有 IMS 未解析的不確定回復單元，則會在 IMS 主要終端機上發出下列訊息:

```
CSQQ008I nn units of recovery are still in doubt in queue manager qmgr-name
```

如果發出此訊息，則表示 IMS 已冷啟動或以不完整的日誌磁帶啟動。 如果 IBM MQ 或 IMS 因軟體錯誤或其他子系統失敗而異常終止，也會發出此訊息。

收到 CSQQ008I 訊息之後:

- 連線會保持作用中。
- IMS 應用程式仍然可以存取 IBM MQ 資源。
- 部分 IBM MQ 資源仍被鎖定。

如果未解析不確定的執行緒， IMS 訊息佇列可以開始建置。 如果 IMS 佇列填滿容量，則 IMS 會終止。 您必須瞭解這個潛在的困難，且必須監視 IMS ，直到完全解決不確定的回復單元為止。

回復程序

使用下列程序來回復 IMS 工作單元:

1. 使用 /SWI OLDS 強制關閉 IMS 日誌，然後保存 IMS 日誌。 使用 DFSERA10 公用程式來列印前一個 IMS 日誌磁帶中的記錄。 類型 X'3730 '日誌記錄指出 phase-2 確定要求，而類型 X'38' 日誌記錄指出中斷要求。 記錄每一個相依區域中最後一個交易所要求的動作。
2. 執行 DL/I 批次工作，以取消涉及未達到確定點的每一個 PSB。 處理程序可能需要一些時間，因為交易仍在處理中。 它也可能會鎖定一些記錄，這可能會影響處理的其餘部分及訊息佇列的其餘部分。
3. 使用下列指令，從 IBM MQ 產生不確定的回復單元清單:

   ```
   +CSQ1 DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
   ```

   您會收到下列訊息:

```
CSQM201I +CSQ1 CSQMDRTC DISPLAY CONN DETAILS
CONN(BC45A794C4290001)
EXTCONN(C3E2D8C3E2C5C3F24040404040404040)
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)
```

對於 IMS， NID 由 IMS 連線名稱和 IMS 提供的唯一號碼組成。 在 IMS 中，此值稱為 回復記號。 如需相關資訊，請參閱 *IMS* 自訂作業手冊。

4. 比較 DISPLAY THREAD 訊息中顯示的 NID (IMSID 加上十六進位的 OASN) 與 DFSERA10 輸出中顯示的 OASN (4 位元組十進位)。 決定要確定還是取消。
5. 使用 RESOLVE INDOUBT 指令在 IBM MQ 中執行不確定的解析，如下所示:

```
RESOLVE INDOUBT( connection-name )
ACTION(COMMIT|BACKOUT)
NID( network-id )
```

若要回復與 *connection-name* 相關聯的所有執行緒，請使用 NID (*) 選項。 此指令會產生下列其中一則訊息，指出執行緒是否已確定或取消:

```
CSQV414I  THREAD network-id COMMIT SCHEDULED
CSQV415I  THREAD network-id BACKOUT SCHEDULED
```

執行不確定的解析時， IMS 及配接器不知道 IBM MQ 的指令，無法確定或取消不確定的回復單元，因為只會影響 IBM MQ 資源。

手動回復 *RRS* 回復單元

請利用這個主題來瞭解如何判斷是否有不確定的 RRS 回復單元，以及如何手動解決這些回復單元。

當 RRS 連接至 IBM MQ 時， IBM MQ 可能有一個以上不確定的回復單元尚未解決。 如果 IBM MQ 具有 RRS 未解析的不確定回復單元，則會在 z/OS 主控台發出下列其中一則訊息:

- CSQ3011I
- CSQ3013I
- CSQ3014I
- CSQ3016I

IBM MQ 和 RRS 都提供工具來顯示不確定回復單元的相關資訊，以及手動解決它們的技術。

在 IBM MQ 中，使用 DISPLAY CONN 指令來顯示不確定 IBM MQ 執行緒的相關資訊。 指令的輸出包括 RRS 作為協調程式之 IBM MQ 執行緒的 RRS 回復單元 ID。 這可用來判斷回復單元的結果。

請使用 RESOLVE INDOUBT 指令來手動解決 IBM MQ 不確定執行緒。 在您決定正確的決策之後，此指令可用來確定或取消回復單元。

在佇列共用群組中的另一個佇列管理程式上回復單元

請利用這個主題來識別及手動回復佇列共用群組中其他佇列管理程式的回復單元。

如果屬於佇列共用群組成員的佇列管理程式失敗且無法重新啟動，則群組中的其他佇列管理程式可以執行同層級回復，並從中接管。 不過，佇列管理程式可能有無法由同層級回復解決的不確定回復單元，因為只有失敗的佇列管理程式才知道該回復單元的最終處置。 當佇列管理程式最終重新啟動時，會解析這些回復單元，但在此之前，它們仍無法確定。

這表示某些資源 (例如訊息) 可能已鎖定，使群組中的其他佇列管理程式無法使用它們。 在此狀況下，您可以使用 DISPLAY THREAD 指令在非作用中佇列管理程式上顯示這些工作單元。 如果您要手動解析這些回復單元，使訊息可供群組中的其他佇列管理程式使用，您可以使用 RESOLVE INDOUBT 指令。

當您發出 DISPLAY THREAD 指令以顯示不確定的回復單元時，您可以使用 QMNAME 關鍵字來指定非作用中佇列管理程式的名稱。 例如，如果您發出下列指令:

```
+CSQ1 DISPLAY THREAD(*) TYPE(INDOUBT) QMNAME(QM01)
```

您會收到下列訊息:

```
CSQV436I +CSQ1 INDOUBT THREADS FOR QM01 -
NAME   THREAD-XREF    URID NID
USER1  00000000000000000000000000 CSQ:0001.0
USER2  00000000000000000000000000 CSQ:0002.0
DISPLAY THREAD REPORT COMPLETE
```

如果指定的佇列管理程式處於作用中，則 IBM MQ 不會傳回不確定執行緒的相關資訊，但會發出下列訊息:

```
CSQV435I CANNOT USE QMNAME KEYWORD, QM01 IS ACTIVE
```

使用 IBM MQ 指令 RESOLVE INDOUBT 來手動解決不確定的執行緒。 使用 QMNAME 關鍵字，在指令中指定非作用中佇列管理程式的名稱。

此指令可用來確定或取消回復單元。 此指令只會解析回復單元的共用部分; 任何本端訊息都不會受到影響，且會保持鎖定狀態，直到佇列管理程式重新啟動或重新連接至 CICS、 IMS 或 RRS 批次為止。

IBM MQandIMS

IBM MQ 提供兩個元件來與 IMS、 IBM MQ - IMS 配接器及 IBM MQ - IMS 橋接器連接。 這些元件通常稱為 IMS 配接器，以及 IMS 橋接器。

操作 IMS 配接器

請利用這個主題來瞭解如何操作 IMS 配接器，以將 IBM MQ 連接至 IMS 系統。

註: IMS 配接器不會納入任何作業和控制面板。

本主題包含下列各節:

- 第 314 頁的『控制 IMS 觸發監視器』

控制 *IMS* 連線

請利用這個主題來瞭解控制及監視 IBM MQ 連線的 IMS 操作員指令。

IMS 提供下列操作員指令，以控制及監視與 IBM MQ 的連線:

/CHANGE SUBSYS
從 IMS 刪除不確定的回復單元。

/DISPLAY OASN SUBSYS
顯示未執行的回復元素。

/DISPLAY SUBSYS
顯示連線狀態及執行緒活動。

/START SUBSYS
將 IMS 控制區域連接至佇列管理程式。

/STOP SUBSYS
中斷 IMS 與佇列管理程式的連線。

/TRACE
控制 IMS 追蹤。

如需這些指令的相關資訊，請參閱您正在使用之 IMS 層次的 *IMS/ESA Operator 's Reference* 手冊。

IMS 指令回應會傳送至從中發出指令的終端機。 發出 IMS 指令的授權基於 IMS 安全。

從 *IMS* 控制區域連接

請利用這個主題來瞭解可從 IMS 連接至 IBM MQ 的機制。

IMS 會從其控制區域建立一個連線，以連接至每一個使用 IMS 的佇列管理程式。 必須啟用 IMS，才能以下列其中一種方式建立連線:

- 在下列任一期間自動執行:
  - 冷啟動起始設定。
  - 如果在 IMS 關閉時 IBM MQ 連線處於作用中狀態，則 IMS 會暖啟動。
- 為了回應 IMS 指令:

```
/START SUBSYS sysid
```

其中 *sysid* 是佇列管理程式名稱。

不論佇列管理程式是否在作用中，都可以發出指令。

在對佇列管理程式進行第一次 MQ API 呼叫之前，不會建立連線。 在該時間之前， IMS 指令 /DIS SUBSYS 會將狀態顯示為 'NOT CONN'。

您啟動 IMS 及佇列管理程式的順序並不重要。

IMS cannot re-enable the connection to the queue manager automatically if the queue manager is stopped with a STOP QMGR command, the IMS command /STOP SUBSYS, or an abnormal end. 因此，您必須使用 IMS 指令 /START SUBSYS 來建立連線。

起始設定配接器並連接至佇列管理程式

配接器是一組模組，使用 IMS 外部子系統連接機能載入至 IMS 控制和相依區域。

此程序會起始設定配接器並連接至佇列管理程式:

1. 從 IMS 讀取子系統成員 (SSM)。PROCLIB。 選擇的 SSM 是 IMS EXEC 參數。 對於「IMS」可以連接的每一個佇列管理程式，成員中有一個項目。 每一個項目都包含 IBM MQ 配接卡的相關控制資訊。
2. 載入 IMS 配接器。

   註: IMS 會為 SSM 成員中定義的每一個 IBM MQ 實例載入一個配接器模組副本。
3. 連接 IBM MQ 的外部子系統作業。
4. 以 CTL EXEC 參數 (IMSID) 作為連線名稱來執行配接器。

無論連線是起始設定的一部分，還是 IMS 指令 /START SUBSYS 的結果，處理程序都是相同的。

當 IMS 嘗試建立連線時，如果佇列管理程式處於作用中，則會傳送下列訊息:

- 至 z/OS 主控台:

```
DFS3613I ESS TCB INITIALIZATION COMPLETE
```

- 至 IMS 主要終端機:

```
CSQQ000I IMS/TM imsid connected to queue manager ssnm
```

當 IMS 嘗試建立連線且 佇列管理程式不在作用中時，每次應用程式發出 MQI 呼叫時，下列訊息會傳送至 IMS 主要終端機:

```
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
```

如果您在啟動與 IMS 的連線時或在系統啟動時收到 DFS3607I 訊息，則表示佇列管理程式無法使用。 若要防止產生大量訊息，您必須執行下列其中一項:

1. 啟動相關佇列管理程式。
2. 發出 IMS 指令:

   ```
   /STOP SUBSYS
   ```

   因此 IMS 不會預期連接至佇列管理程式。

如果您都不這麼做，則每次在區域中排定工作時，以及每次應用程式對佇列管理程式提出連線要求時，都會發出 DFS3607I 訊息及相關聯的 CSQQ001I 訊息。

執行緒附件

在 MPP 或 IFP 區域中，當第一個應用程式排定到該區域時，即使該應用程式未發出 IBM MQ 呼叫， IMS 也會建立執行緒連線。 在 BMP 區域中，當應用程式第一次發出 IBM MQ 呼叫 (MQCONN 或 MQCONNX) 時， 會建立執行緒連線。 此執行緒會在區域期間保留，或直到連線停止為止。

對於訊息驅動及非訊息驅動區域，與執行緒相關聯的回復執行緒交互參照 ID *Thread-xref* 為:

```
PSTid + PSBname
```

其中:

PSTid
分割區規格表格區域 ID

PSBname
指定程式區塊名稱

您可以在 IBM MQ 指令中使用連線 ID 作為唯一 ID，在此情況下， IBM MQ 會自動將這些 ID 插入它所產生的任何操作員訊息中。

顯示不確定的回復單元

您可以顯示不確定的回復單元，並嘗試回復它們。

本主題中用來列出及回復不確定回復單元的作業步驟僅適用於相對簡單的情況。 如果佇列管理程式在連接至 IMS 時異常結束，則 IMS 可能會確定或取消工作，而 IBM MQ 不知道它。 當佇列管理程式重新啟動時，該工作稱為 不確定。 必須針對工作狀態做出決策。

若要顯示不確定的回復單元清單，請發出下列指令:

```
+CSQ1  DISPLAY CONN(*) WHERE(UOWSTATE EQ UNRESOLVED)
```

IBM MQ 會回應如下的訊息:

```
CSQM201I +CSQ1 CSQMDRTC DIS CONN DETAILS
CONN(BC0F6125F5A30001)
EXTCONN(C3E2D8C3C3E2D8F14040404040404040)
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(000000000000001F000000007472616E5F6964547565204E6F762020...)
QMURID(000000026232)
URTYPE(XA)
USERID( )
APPLTAG(CSQ1CHIN)
ASID(0000)
APPLTYPE(CHINIT)
CHANNEL( )
CONNAME( )
END CONN DETAILS
```

如需此訊息中屬性的說明，請參閱 DISPLAY CONN 指令的說明。

回復不確定的回復單元

若要回復不確定的回復單元，請發出下列指令:

```
+CSQ1  RESOLVE INDOUBT( connection-name ) ACTION(COMMIT|BACKOUT)
NID( net-node.number )
```

其中:

connection-name
IMS 系統 ID。

ACTION
指出要確定 (COMMIT) 或取消 (BACKOUT) 此回復單元。

net-node.number
相關聯的 net-node.number。

當您發出 RESOLVE INDOUBT 指令時，會顯示下列其中一則訊息:

```
CSQV414I +CSQ1 THREAD network-id COMMIT SCHEDULED

CSQV415I +CSQ1 THREAD network-id BACKOUT SCHEDULED
```

解析剩餘回復項目

在給定時間， IMS 會建置殘留回復項目 (RRE) 的清單。 RRE 是 IBM MQ 可能不確定的回復單元。 它們在幾種情況下產生:

- 如果佇列管理程式非作用中，則 IMS 具有在佇列管理程式作用中之前無法解析的 RRE。 這些 RRE 不是問題。
- 如果佇列管理程式處於作用中並連接至 IMS，且 IMS 取消 IBM MQ 已確定的工作，則 IMS 配接器會發出訊息 CSQQ010E。 如果兩個系統中的資料必須一致，則會發生問題。 如需解決此問題的相關資訊，請參閱 第 304 頁的『手動回復 IMS 回復單元』。
- 如果佇列管理程式處於作用中並連接至 IMS，則即使沒有任何訊息通知您此問題，仍可能有 RRE。 建立與 IMS 的 IBM MQ 連線之後，您可以發出下列 IMS 指令，以找出是否有問題:

```
/DISPLAY OASN SUBSYS sysid
```

若要清除 RRE ，請發出下列其中一個 IMS 指令:

```
/CHANGE SUBSYS sysid RESET
/CHANGE SUBSYS sysid RESET OASN nnnn
```

其中 *nnnn* 是為了回應 +CSQ1 DISPLAY 指令而列出的原始應用程式序號。 這是程式實例的排程號碼，提供它自前次 IMS 冷啟動以來在該程式呼叫順序中的位置。 IMS 不能有兩個不確定的回復單元具有相同的排程號碼。

這些指令會重設 IMS 的狀態; 它們不會導致與 IBM MQ 進行任何通訊。

控制 *IMS* 相依區域連線

您可以控制、監視及在必要時終止 IMS 與 IBM MQ 之間的連線。

控制 IMS 相依區域連線涉及下列活動:

- 從相依區域連接
- 區域錯誤選項
- 監視連線上的活動
- 切斷與相依區域的連線

從相依區域連接

控制區域中使用的 IMS 配接器也會載入至相依區域。 會建立從每一個相依區域到 IBM MQ 的連線。 此連線用於協調 IBM MQ 和 IMS 工作的確定。 若要起始設定並建立連線， IMS 會執行下列動作:

1. 從 IMS 讀取子系統成員 (SSM)。PROCLIB。

   可以在相依區域 EXEC 參數上指定子系統成員。 如果未指定，則會使用控制區域 SSM。 如果區域永不可能連接至 IBM MQ，為了避免載入配接器，請指定不含任何項目的成員。
2. 載入 IBM MQ 配接器。

   對於批次訊息程式，除非應用程式發出其第一個傳訊指令，否則不會完成載入。 此時， IMS 會嘗試建立連線。

   對於訊息處理程式區域或 IMS 捷徑區域，會在起始設定區域時進行嘗試。

區域錯誤選項

如果佇列管理程式不在作用中，或從應用程式傳送第一個傳訊指令時無法使用資源，則所採取的動作取決於 SSM 項目上指定的錯誤選項。 選項有:

R
適當的回覆碼會傳送至應用程式。

Q
應用程式異常結束，異常終止碼為 U3051。 輸入訊息已重新排入佇列。

A
應用程式異常結束，異常終止碼為 U3047。 已捨棄輸入訊息。

監視連線上的活動

當應用程式提出其第一個成功 IBM MQ 要求時，會從相依區域建立執行緒。 您可以從 IBM MQ 發出下列指令，以顯示連線及目前使用它們的應用程式的相關資訊:

```
+CSQ1 DISPLAY CONN(*) ALL
```

指令會產生如下所示的訊息:

```
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)
```

對於控制區域， *thread-xref* 是特殊值 CONTROL。 對於相依區域，它是與 PSBname 連結的 PSTid。 *auth-id* 是工作卡中的使用者欄位，或 z/OS 啟動程序表格中的 ID。

如需所顯示清單的說明，請參閱 IBM MQ for z/OS 訊息、完成及原因碼 文件中訊息 CSQV402I 的說明。

IMS 提供顯示指令來監視與 IBM MQ 的連線。 它會顯示每一個相依區域連線上的作用中程式、LTERM 使用者名稱及控制區域連線狀態。 指令為:

```
/DISPLAY SUBSYS name
```

IMS 與 IBM MQ 之間的連線狀態顯示為下列其中一項:

```
CONNECTED
NOT CONNECTED
CONNECT IN PROGRESS
STOPPED
STOP IN PROGRESS
INVALID SUBSYSTEM NAME= name
SUBSYSTEM name NOT DEFINED BUT RECOVERY OUTSTANDING
```

每一個相依區域的執行緒狀態是下列其中一項:

```
CONN
CONN, ACTIVE (includes LTERM of user)
```

中斷與相依區域的連線

變更 IMS 的 SSM 成員中的值。PROCLIB , 您中斷相依區域的連線。 若要執行此動作，您必須:

1. 發出 IMS 指令:

```
/STOP REGION
```

2. 更新 SSM 成員。
3. 發出 IMS 指令:

```
/START REGION
```

中斷連線 *IMS*

當 IMS 或佇列管理程式終止時，會結束連線。 或者， IMS 主要終端機操作員可以明確中斷連線。

若要終止 IMS 與 IBM MQ 之間的連線，請使用下列 IMS 指令:

```
/STOP SUBSYS sysid
```

該指令會將下列訊息傳送至發出該訊息的終端機，通常是主要終端機操作員 (MTO):

```
DFS058I STOP COMMAND IN PROGRESS
```

IMS 指令:

```
/START SUBSYS sysid
```

需要重新建立連線。

註: 如果 IMS 觸發監視器正在執行中，則 IMS 指令 /STOP SUBSYS 未完成。

控制 *IMS* 觸發監視器

您可以使用 CSQQTRMN 交易來停止並啟動 IMS 觸發監視器。

設定 IMS 觸發監視器中說明 IMS 觸發監視器 (CSQQTRMN 交易)。

若要控制 IMS 觸發監視器，請參閱:

- 啟動 CSQQTRMN
- 停止 CSQQTRMN

啟動 CSQQTRMN

1. 針對您要監視的每一個起始佇列，啟動執行程式 CSQQTRMN 的批次導向 BMP。
2. 修改批次 JCL 以新增 DDname CSQQUT1 ， 其指向包含下列資訊的資料集:

```
QMGRNAME=q_manager_name    Comment: queue manager name
INITQUEUENAME=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	佇列管理程式的名稱 (如果這是空白，則會採用 CSQQDEFV 中指定的預設值)
init_q_name	要監視的起始佇列名稱
lterm	錯誤訊息目的地的 IMS LTERM 名稱 (如果此為空白，則預設值為 MASTER)。
CONSOLEMESSAGES=YES	傳送至指定 IMS LTERM 的訊息也會傳送至 z/OS 主控台的要求。 如果省略或拼錯此參數，則預設值為「不」將訊息傳送至主控台。

3. 如果您想要 CSQQUT1 輸入處理程序的列印報告，請新增 DD 名稱 CSQQUT2 。

註:

1. 使用 LRECL=80 定義資料集 CSQQUT1 。 其他 DCB 資訊取自資料集。 資料集 CSQQUT2 的 DCB 是 RECFM=VBA 和 LRECL=125。
2. 您只能在每一筆記錄上放置一個關鍵字。 關鍵字值以關鍵字後面的第一個空白區隔; 這表示您可以併入註解。 直欄 1 中的星號表示整個輸入記錄是註解。
3. 如果您拼錯 QMGRNAME 或 LTERM 關鍵字，則 CSQQTRMN 會使用該關鍵字的預設值。
4. 提交觸發監視器 BMP 工作之前，請確定已在 IMS 中啟動子系統 (由 /START SUBSYS 指令啟動)。 如果未啟動，則觸發監視器工作會終止，異常終止碼為 U3042。

正在停止 CSQQTRMN

啟動之後， CSQQTRMN 會一直執行，直到 IBM MQ 與 IMS 之間的連線因下列其中一個事件而中斷:

- 佇列管理程式結束中
- IMS 結束

或輸入 z/OS STOP **jobname** 指令。

控制 IMS 橋接器

請利用這個主題來瞭解您可以用來控制 IMS 橋接器的 IMS 指令。

沒有 IBM MQ 指令可控制 IBM MQ-IMS 橋接器。 不過，您可以使用下列方式來停止將訊息遞送至 IMS :

- 若為非共用佇列，請針對所有橋接器佇列使用 ALTER QLOCAL (xxx) GET (DISABLED) 指令。
- 若為叢集佇列，請使用 SUSPEND QMGR CLUSTER (xxx) 指令。 這只有在另一個佇列管理程式也管理叢集橋接器佇列時才有效。
- 對於叢集佇列，請使用 SUSPEND QMGR FACILITY (IMSBRIDGE) 指令。 不會將進一步訊息傳送至 IMS，但會從 IMS 收到任何未完成交易的回應。

  若要重新開始傳送訊息至 IMS ，請發出 RESUME QMGR FACILITY (IMSBRIDGE) 指令。

您也可以使用 MQSC 指令 DISPLAY SYSTEM 來顯示橋接器是否已暫停。

如需這些指令的詳細資料，請參閱 MQSC 指令 。

如需進一步資訊，請參閱:

- 第 316 頁的『啟動和停止 IMS 橋接器』
- 第 316 頁的『控制 IMS 連線』
- 控制橋接器佇列

- 第 317 頁的『重新同步化 IMS 橋接器』
- 使用 tpipe 名稱
- 從 IMS 刪除訊息
- 刪除 tp 管道
- 第 319 頁的『IMS 交易有效期限』

啟動和停止 IMS 橋接器

透過啟動 OTMA 來啟動 IBM MQ 橋接器。 請使用 IMS 指令:

```
/START OTMA
```

或在 IMS 系統參數中指定 OTMA=YES 來自動啟動它。 如果 OTMA 已啟動，當佇列管理程式啟動完成時，橋接器會自動啟動。 啟動 OTMA 時會產生 IBM MQ 事件訊息。

使用 IMS 指令:

```
/STOP OTMA
```

來停止 OTMA 通訊。 當發出這個指令時，會產生 IBM MQ 事件訊息。

控制 IMS 連線

IMS 提供下列操作員指令，以控制及監視與 IBM MQ 的連線:

/DEQUEUE TMEMBER *tmember* TPIPE *tpipe*
從 Tpipe 移除訊息。 指定 PURGE1 以移除所有訊息，或指定 PURGE1 以僅移除第一個訊息。

/DISPLAY OTMA
顯示 OTMA 伺服器及用戶端的相關摘要資訊，以及用戶端狀態。

/DISPLAY TMEMBER *name*
顯示 OTMA 用戶端的相關資訊。

/DISPLAY TRACE TMEMBER 名稱
顯示正在追蹤之內容的相關資訊。

/SECURE OTMA
設定安全選項。

/START OTMA
透過 OTMA 啟用通訊。

/START TMEMBER *tmember* TPIPE *tpipe*
啟動具名 Tpipe。

/STOP OTMA
停止透過 OTMA 的通訊。

/STOP TMEMBER *tmember* TPIPE *tpipe*
停止具名 Tpipe。

/TRACE
控制 IMS 追蹤。

如需這些指令的相關資訊，請參閱您使用之 IMS 層次的 *IMS/ESA Operators Reference* 手冊。

IMS 指令回應會傳送至從中發出指令的終端機。 發出 IMS 指令的授權基於 IMS 安全。

控制橋接器佇列

若要透過橋接器停止與具有 XCF 成員名稱 *tmember* 的佇列管理程式通訊，請發出下列 IMS 指令:

```
/STOP TMEMBER tmember TPIPE ALL
```

若要回復通訊，請發出下列 IMS 指令:

```
/START TMEMBER tmember TPIPE ALL
```

可以使用 MQ DISPLAY QUEUE 指令來顯示佇列的 Tpes。

若要在單一 Tpipe 上停止與佇列管理程式的通訊，請發出下列 IMS 指令:

```
/STOP TMEMBER tmember TPIPE tpipe
```

會為每一個作用中橋接器佇列建立一或兩個 Tp 管道，因此發出此指令會停止與 IBM MQ 佇列的通訊。 若要回復通訊，請使用下列 IMS 指令:

```
/START TMEMBER tmember TPIPE tpipe
```

或者，您可以變更 IBM MQ 佇列的屬性，讓它被禁止。

重新同步化 IMS 橋接器

每當佇列管理程式、 IMS 或 OTMA 重新啟動時，都會自動重新啟動 IMS 橋接器。

IMS 橋接器所承擔的第一個作業是與 IMS 重新同步化。 這涉及 IBM MQ 和 IMS 檢查每個同步化 Tpipe 上的序號。 使用確定模式零 (commit-then-send) 從 IBM MQ - IMS 橋接器佇列傳送持續訊息至 IMS 時，會使用 synchronized Tpipe。

如果橋接器無法與 IMS 重新同步化，則會在訊息 CSQ2023E 中傳回 IMS 感應碼，且會停止與 OTMA 的連線。 如果橋接器無法與個別 IMS Tpipe 重新同步化，則會在訊息 CSQ2025E 中傳回 IMS 感應碼，且 Tpipe 已停止。 如果已冷啟動 Tpipe ，則可回復的序號會自動重設為 1。

如果橋接器在與 Tpipe 重新同步化時探索到不符的序號，則會發出 CSQ2020E 訊息。 使用 IBM MQ 指令 RESET TPIE 來起始與 IMS Tpipe 的重新同步。 您需要提供 XCF 群組和成員名稱，以及 Tpipe 的名稱; 此資訊由訊息提供。

您也可以指定:

- 要在 Tpipe 中設定 IBM MQ 所傳送訊息的新可回復序號，並設為夥伴的接收序號。 如果您未指定此項，則夥伴的接收序號會設為現行 IBM MQ 傳送序號。
- 要在 Tpipe 中為 IBM MQ 接收的訊息設定新的可回復序號，並設為夥伴的傳送序號。 如果您未指定此項，則夥伴的傳送序號會設為現行 IBM MQ 接收序號。

如果有未解析的回復單元與 Tpipe 相關聯，則也會在訊息中通知此回復單元。 請使用 IBM MQ 指令 RESET TPIPE 來指定是要確定回復單元，還是取消回復單元。 如果您確定回復單元，則訊息批次已傳送至 IMS，並從橋接器佇列中刪除。 如果您支援回復單元輸出，則訊息會傳回橋接器佇列，稍後會傳送至 IMS。

確定模式 1 (send-then-commit) Tp 管道未同步。

確定模式 1 交易的考量

在 IMS 中，確定模式 1 (CM1) 交易會在同步點之前傳送其輸出回覆。

CM1 交易可能無法傳送其回覆，例如:

- 要傳送回覆的 Tpipe 已停止
- OTMA 已停止
- OTMA 用戶端 (即佇列管理程式) 已消失
- 無法使用回覆目的地佇列及無法傳送郵件的佇列

基於這些原因， IMS 應用程式傳送訊息虛擬異常終止，程式碼為 U0119。 在此情況下，不會停止 IMS 交易及程式。

這些原因通常會阻止訊息傳送至 IMS，以及阻止從 IMS 遞送回覆。 在下列情況下，可能會發生 U0119 異常終止:

- 訊息位於 IMS 時， Tpipe、OTMA 或佇列管理程式會停止
- IMS 在不同的 Tpipe 上回覆送入訊息，且 Tpipe 已停止
- IMS 會回覆不同的 OTMA 用戶端，且該用戶端無法使用。

每當發生 U0119 異常終止時， IMS 的送入訊息和 IBM MQ 的回覆訊息都會遺失。 如果 CM0 交易的輸出因上述任何原因而無法遞送，則會在 IMS 內的 Tpipe 上排入佇列。

使用 tpipe 名稱

許多用來控制 IBM MQ - IMS 橋接器的指令都需要 *tpipe* 名稱。 請利用這個主題來瞭解如何尋找 tpipe 名稱的進一步詳細資料。

對於許多控制 IBM MQ - IMS 橋接器的指令，您需要 *tpipe* 名稱。 您可以從 DISPLAY QUEUE 指令取得 tpipe 名稱，並注意下列要點:

- 當定義本端佇列時，會指派 tpipe 名稱
- 本端佇列有兩個 tpipe 名稱，一個用於同步，另一個用於非同步
- 在 IMS 與 IBM MQ 之間特定於該特定本端佇列的某些通訊發生之後， IMS 才會知道 tpipe 名稱
- 若要讓 tpipe 可供 IBM MQ - IMS 橋接器使用，必須將其相關聯佇列指派給已完成正確 XCF 群組及成員名稱欄位的「儲存類別」

刪除其中的訊息 IMS

如果 Tmember/Tpipe 已停止，則可以刪除透過 IMS 橋接器以 IBM MQ 為目的地的訊息。 若要針對具有 XCF 成員名稱 *tmember* 的佇列管理程式刪除一則訊息，請發出下列 IMS 指令:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE1
```

若要刪除 Tpipe 上的所有訊息，請發出下列 IMS 指令:

```
/DEQUEUE TMEMBER tmember TPIPE tpipe PURGE
```

正在刪除 tp 管道

您無法自行刪除 IMS tp 管道。 IMS 會在下列時間刪除它們:

- 當 IMS 冷啟動時，會刪除已同步的 tp 管道。

- 當 IMS 重新啟動時，會刪除未同步的 tp 管道。

IMS 交易有效期限

有效期限與交易相關聯; 任何 IBM MQ 訊息都可以有相關聯的有效期限。 使用 MQMD.Expiry 欄位，將有效期限間隔從應用程式傳遞至 IBM MQ。 時間是訊息到期之前的持續時間，以十分之一秒的值表示。 嘗試執行訊息的 MQGET (晚於其過期) 會導致從佇列中移除訊息，並執行到期處理程序。 當訊息在 IBM MQ 網路上的佇列管理程式之間流動時，有效期限會減少。 當 IMS 訊息透過 IMS 橋接器傳遞至 OTMA 時，剩餘的訊息到期時間會作為交易到期時間傳遞至 OTMA。

如果交易已指定有效期限，則 OTMA 會在 IMS 中的三個不同位置使輸入交易到期:

- 從 XCF 接收輸入訊息
- 輸入訊息移入佇列時間
- 應用程式 GU 時間

在 GU 時間之後不會執行任何到期。

交易 EXPRTIME 可以由下列提供:

- IMS 交易定義 (transaction definition)
- IMS OTMA 訊息標頭
- IMS DFSINSX0 使用者結束程式
- IMS CREATE 或 UPDATE TRAN 指令

IMS 表示交易已過期，方法為異常終止交易 0243 ，並發出訊息。 發出的訊息為非共用佇列環境中的 DFS555I ，或共用佇列環境中的 DFS2224I 。

操作 IBM MQ Advanced Message Security

若要輸入 IBM MQ Advanced Message Security 位址空間的指令，請使用 z/OS MODIFY 指令。

例如:

```
F qmgr AMSM, cmd
```

接受下列 MODIFY 指令:

表 *19: IBM MQ Advanced Message Security* 位址空間 *MODIFY* 指令

指令	選項	說明
顯示畫面		顯示版本資訊
重新整理	keyRing 原則 ALL	重新整理金鑰環憑證及/或安全原則。
SMFAUDIT	成功 失敗 ALL	設定當 AMS 順利保護/解除保護訊息及/或 AMS 無法保護/解除保護訊息時，是否需要 SMF 審核。
SMFTYPE	0 - 255	設定當 AMS 保護/取消保護訊息時要產生的 SMF 記錄類型。 如果要停用 SMF 審核，請指定記錄類型 0。

註: 若要指定選項，必須以逗點區隔。 例如:

```
F qmgrAMSM,REFRESH KEYRING
```

```
F qmgrAMSM,SMFAUDIT ALL
F qmgrAMSM,SMFTYPE 180
```

REFRESH 指令。

發出 MQOPEN 呼叫的應用程式將採用變更。 現有應用程式會繼續使用該應用程式開啟佇列時的選項。 若要讓變更生效，應用程式必須關閉並重新開啟佇列。

注意事項

本資訊係針對 IBM 在美國所提供之產品與服務所開發。

在其他國家中，IBM 可能不會提供本書中所提的各項產品、服務或功能。 請洽當地 IBM 業務代表，以取得當地目前提供的產品和服務之相關資訊。 這份文件在提及 IBM 的產品、程式或服務時，不表示或暗示只能使用 IBM 的產品、程式或服務。 只要未侵犯 IBM 的智慧財產權，任何功能相當的產品、程式或服務都可以取代 IBM 的產品、程式或服務。 不過，任何非 IBM 的產品、程式或服務，使用者必須自行負責作業的評估和驗證責任。

本文件所說明之主題內容，IBM 可能擁有其專利或專利申請案。 提供本文件不代表提供這些專利的授權。 您可以書面提出授權查詢，來函請寄到：

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

如果是有關雙位元組 (DBCS) 資訊的授權查詢，請洽詢所在國的 IBM 智慧財產部門，或書面提出授權查詢，來函請寄到：

智慧財產權授權
法務部與智慧財產權法律
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

下列段落不適用於英國，若與任何其他國家之法律條款抵觸，亦不適用於該國： International Business Machines Corporation 只依 "現況" 提供本出版品，不提供任何明示或默示之保證，其中包括且不限於不侵權、可商用性或特定目的之適用性的隱含保證。 有些地區在特定交易上，不允許排除明示或暗示的保證，因此，這項聲明不一定適合您。

這項資訊中可能會有技術上或排版印刷上的訛誤。 因此，IBM 會定期修訂；並將修訂後的內容納入新版中。 IBM 隨時會改進及/或變更本出版品所提及的產品及/或程式，不另行通知。

本資訊中任何對非 IBM 網站的敘述僅供參考，IBM 對該網站並不提供任何保證。 這些網站所提供的資料不是 IBM 本產品的資料內容，如果要使用這些網站的資料，您必須自行承擔風險。

IBM 得以各種適當的方式使用或散布由您提供的任何資訊，無需對您負責。

如果本程式的獲授權人為了 (i) 在個別建立的程式和其他程式（包括本程式）之間交換資訊，以及 (ii) 相互使用所交換的資訊，因而需要相關的資訊，請洽詢：

IBM Corporation
軟體交互作業能力協調程式，部門 49XA
3605 公路 52 N
Rochester， MN 55901
U.S.A.

在適當條款與條件之下，包括某些情況下（支付費用），或可使用此類資訊。

IBM 基於雙方之 IBM 客戶合約、IBM 國際程式授權合約或任何同等合約之條款，提供本資訊所提及的授權程式與其所有適用的授權資料。

本文件中所含的任何效能資料都是在受管制的環境下判定。 因此不同作業環境之下所得的結果，可能會有很大的差異。 有些測定已在開發階段系統上做過，不過這並不保證在一般系統上會出現相同結果。 甚至有部分的測量，是利用插補法而得的估計值， 實際結果可能有所不同。 本文件的使用者應驗證其特定環境適用的資料。

本文件所提及之非 IBM 產品資訊，取自產品的供應商，或其發佈的聲明或其他公開管道。 IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、相容性或任何對產品的其他主張是否完全無誤。 有關非 IBM 產品的性能問題應直接洽詢該產品供應商。

有關 IBM 未來方針或目的之所有聲明，僅代表 IBM 的目標與主旨，隨時可能變更或撤銷，不必另行通知。

這份資訊含有日常商業運作所用的資料和報告範例。 為了要使它們儘可能完整，範例包括個人、公司、 品牌和產品的名稱。 這些名稱全屬虛構，如與實際公司的名稱和住址雷同，純屬巧合。

著作權授權:

本資訊含有原始語言之範例應用程式，用以說明各作業平台中之程式設計技術。 您可以基於研發、使用、銷售或散布符合作業平台（撰寫範例程式的作業平台）之應用程式介面的應用程式等目的，以任何形式複製、修改及散布這些範例程式，而不必向 IBM 付費。 這些範例並未在所 有情況下完整測試。 因此，IBM 不保證或暗示這些程式的可靠性、有用性或功能。

若 貴客戶正在閱讀本項資訊的電子檔，可能不會有照片和彩色說明。

程式設計介面資訊

程式設計介面資訊 (如果有提供的話) 旨在協助您建立與此程式搭配使用的應用軟體。

本書包含預期程式設計介面的相關資訊，可讓客戶撰寫程式以取得 WebSphere MQ 的服務。

不過，本資訊也可能包含診斷、修正和調整資訊。 提供診斷、修正和調整資訊，是要協助您進行應用軟體的除錯。

重要: 請勿使用此診斷、修改及調整資訊作為程式設計介面，因為它可能會變更。

商標

IBM、 IBM 標誌 ibm.com 是 IBM Corporation 在全球許多適用範圍的商標。 IBM 商標的最新清單可在 Web 的 "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml 中找到。 其他產品和服務名稱，可能是 IBM 或其他公司的商標。

Microsoft 及 Windows 是 Microsoft Corporation 在美國及/或其他國家或地區的商標。

UNIX 是 The Open Group 在美國及/或其他國家/地區的註冊商標。

Linux 是 Linus Torvalds 在美國及/或其他國家或地區的註冊商標。

本產品包含 Eclipse Project (http://www.eclipse.org/) 所開發的軟體。

Java 和所有以 Java 為基礎的商標及標誌是 Oracle 及/或其子公司的商標或註冊商標。

產品編號:

(1P) P/N: