7.5

管理 *IBM WebSphere MQ*

附註

使用本資訊及其支援的產品之前，請先閱讀第 141 頁的『注意事項』中的資訊。

除非新版中另有指示，否則此版本適用於 IBM® WebSphere MQ 7.5 版及所有後續版次與修訂。

當您將資訊傳送至 IBM 時，您授與 IBM 非專屬權利，以任何其認為適當的方式使用或散佈資訊，而無需對您負責。

目錄

管理 IBM WebSphere 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 WebSphere 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 WebSphere 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 WebSphere MQ：

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

還有下列其他選項可用來建立及管理 IBM WebSphere MQ 物件:

- 第 6 頁的『IBM WebSphere MQ Explorer』
- 第 6 頁的『Windows 預設配置應用程式』
- 第 6 頁的『Microsoft Cluster Service (MSCS)』

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 WebSphere 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 7 頁的『自動化管理作業』。

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

控制指令在 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 WebSphere MQ 控制指令。

IBM WebSphere 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。 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， MQSC 可以建置在您從 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 MQSC 參照。

第 66 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 65 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 WebSphere 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 8 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 WebSphere MQ Explorer

使用 IBM WebSphere 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 及 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 IBM WebSphere MQ Explorer 。

在 Linux 上，若要順利啟動 IBM WebSphere 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 WebSphere MQ Explorer 在其他平台 (包括 z/OS) 上管理遠端佇列管理程式，以取得詳細資料並下載 SupportPac MS0T，請參閱 https://www.ibm.com/support/docview.wss?uid=swg24021041。

如需相關資訊，請參閱第 51 頁的『使用 IBM WebSphere MQ Explorer 進行管理』 。

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 WebSphere 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 Cluster Service (MSCS)

Microsoft Cluster Service (MSCS) 可讓您將伺服器連接至 叢集，提高資料及應用程式的可用性，並讓系統更容易管理。 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 WebSphere MQ 叢集。 區別是:

IBM WebSphere 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 Cluster Service (MSCS) 提供如何配置 IBM WebSphere MQ for Windows 系統以使用 MSCS 的詳細資訊。

相關概念

WebSphere MQ 技術概觀

第 62 頁的『管理本端 IBM WebSphere MQ 物件』

本節告訴您如何管理本端 IBM WebSphere 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 WebSphere MQ 物件。

第 92 頁的『管理遠端 IBM WebSphere MQ 物件』

失去與 XA 資源管理程式的聯絡時的考量

相關工作

規劃

配置

相關參考

交易式支援實務

本端和遠端管理

您可以在本端或遠端管理 WebSphere 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 WebSphere MQ 中，您可以將此視為本端管理，因為不涉及通道，即通訊由作業系統管理。

WebSphere 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，而且也適用於「WebSphere MQ 探險家」。 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「WebSphere MQ 探險家」。

第 92 頁的『管理遠端 IBM WebSphere MQ 物件』 更詳細地說明遠端管理的主題。

如何使用 IBM WebSphere MQ 控制指令

本節說明如何使用 IBM WebSphere MQ 控制指令。

如果您要發出控制指令，則您的使用者 ID 必須是 mqm 群組的成員。 如需此作業的相關資訊，請參閱 在 UNIX、 Linux 及 Windows 系統上管理 WebSphere MQ 的權限。 此外，請注意下列環境特定資訊:

WebSphere 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。

WebSphere MQ for UNIX and Linux 系統

所有 WebSphere 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 WebSphere MQ 檔案總管」來發出控制指令的子集。

如需相關資訊，請參閱 WebSphere 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 WebSphere MQ 物件的經驗。

PCF 指令

WebSphere MQ 可程式化指令格式 (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 WebSphere 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 WebSphere 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 8 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 WebSphere MQ for AIX
- IBM WebSphere MQ for HP-UX
- IBM WebSphere MQ for Linux
- IBM WebSphere MQ for Solaris
- IBM WebSphere MQ for Windows
- 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。

WebSphere 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，以管理 WebSphere MQ 物件: 鑑別資訊物件、通道、通道接聽器、名稱清單、處理程序定義、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 WebSphere MQ 遠端管理的系統管理程式中使用 PCF。

本節包括:

- 第 9 頁的『PCF 指令訊息』
- 第 11 頁的『回應』
- IBM WebSphere MQ 物件的命名規則
- 第 13 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。

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 WebSphere MQ COPY、標頭、併入和模組檔案。

PCF 指令的訊息描述子

WebSphere MQ 訊息描述子已完整記載在 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 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 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。 通話之後，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; WebSphere MQ Windows Explorer 是其中一個此類應用程式 (不過， 6.0 版或更新版本 IBM WebSphere MQ Explorer 在 z/OS 上支援 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。 此結構在回應的定義 (可程式指令格式的定義) 中顯示為 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。

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 WebSphere 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 WebSphere MQ 適用於 Windows， UNIX and Linux 系統

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 WebSphere MQ 物件權限檢查，如 第 14 頁的表 1 所示。

若要處理下列任何指令 ，使用者 ID 必須屬於群組 *mqm*。

註: 對於 Windows **僅**，使用者 ID 可以屬於群組 管理者 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

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 WebSphere MQ 物件權限檢查，如 第 14 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

WebSphere MQ 物件權限

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

指令	**WebSphere 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

表 *1: Windows*、*HP Integrity NonStop Server*、 *UNIX and Linux* 系統-物件權限 (繼續)		
指令	**WebSphere MQ 物件權限**	**類別權限 (適用於物件類型)**
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
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

表 *1: Windows*、*HP Integrity NonStop Server*、 *UNIX and Linux* 系統-物件權限 (繼續)		
指令	**WebSphere MQ 物件權限**	**類別權限 (適用於物件類型)**
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
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。

WebSphere 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。

使用 MQAI 來簡化 PCF 的使用

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

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)。

以下列方式使用 MQAI:

簡化 PCF 訊息的使用
MQAI 是管理 WebSphere MQ 的簡單方法; 您不需要撰寫自己的 PCF 訊息，從而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 訊息必須包含指令及字串或整數資料的詳細資料。 若要執行此動作，您需要在程式中為每個結構提供數個陳述式，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 資料工具袋可讓您不需要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，但 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，該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 17 頁的『IBM WebSphere MQ 管理介面 (MQAI) 簡介』。

IBM WebSphere MQ 管理介面 (MQAI) 簡介

IBM WebSphere MQ 管理介面 (MQAI) 是 IBM WebSphere MQ 的程式設計介面。 它在 IBM WebSphere 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 WebSphere MQ 的程式設計介面，使用 C 語言及 Visual Basic for Windows。 它可在 z/OS 以外的平台上使用。

它會使用資料工具袋在 WebSphere 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 43 頁的『資料工具袋』。 如需 PCF 的相關資訊，請參閱 第 8 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 WebSphere 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 WebSphere 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。 如果訊息資料由整數和字串組成，您可以使用 MQAI 來利用 PCF 資料的 WebSphere 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 WebSphere MQ 以及在應用程式之間交換資料的相關資訊，請參閱 第 17 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 18 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 22 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 34 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 29 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 來建置應用程式，您可以鏈結至與 WebSphere MQ 相同的程式庫。 如需如何建置 WebSphere MQ 應用程式的相關資訊，請參閱 建置 WebSphere MQ 應用程式。

使用 MQAI 配置 WebSphere 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。 您可以在 第 38 頁的『配置 IBM WebSphere MQ 的提示和要訣』 中找到使用 MQAI 來配置 WebSphere MQ 的提示

IBM WebSphere MQ 管理介面 (MQAI)

IBM WebSphere MQ for Windows、 AIX、 Linux、 HP-UX 及 Solaris 支援 IBM WebSphere MQ 管理介面 (MQAI)。 MQAI 是 IBM WebSphere 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 WebSphere 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 */
/*               WebSphere 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, 2024.                        */
/*                                                                          */
/****************************************************************************/
/*                                                                          */
/* 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 WebSphere 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,                    /* WebSphere 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 WebSphere MQ Admin Interface (MQAI).          */
/* Licensed Materials - Property of IBM                                  */
/*                                                                       */
/* 63H9336                                                               */
/* (c) Copyright IBM Corp. 1999, 2024. 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 needs 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 WebSphere MQ Administration Interface (MQAI)     */
/*                                                                          */
/* <N_OCO_COPYRIGHT>                                                        */
/* Licensed Materials - Property of IBM                                     */
/*                                                                          */
/* 63H9336                                                                  */
/* (c) Copyright IBM Corp. 2008, 2024. All Rights Reserved.                 */
/*                                                                          */
/* US Government Users Restricted Rights - Use, duplication or              */
/* disclosure restricted by GSA ADP Schedule Contract with                  */
/* IBM Corp.                                                                */
/* <NOC_COPYRIGHT>                                                          */
/****************************************************************************/
/*                                                                          */
/* Function:                                                                */
/*    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 WebSphere 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, 2024.                         */
/*                                                                           */
/*****************************************************************************/
/*                                                                           */
/* 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 WebSphere 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,                  /* WebSphere 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 WebSphere MQ 的提示和要訣

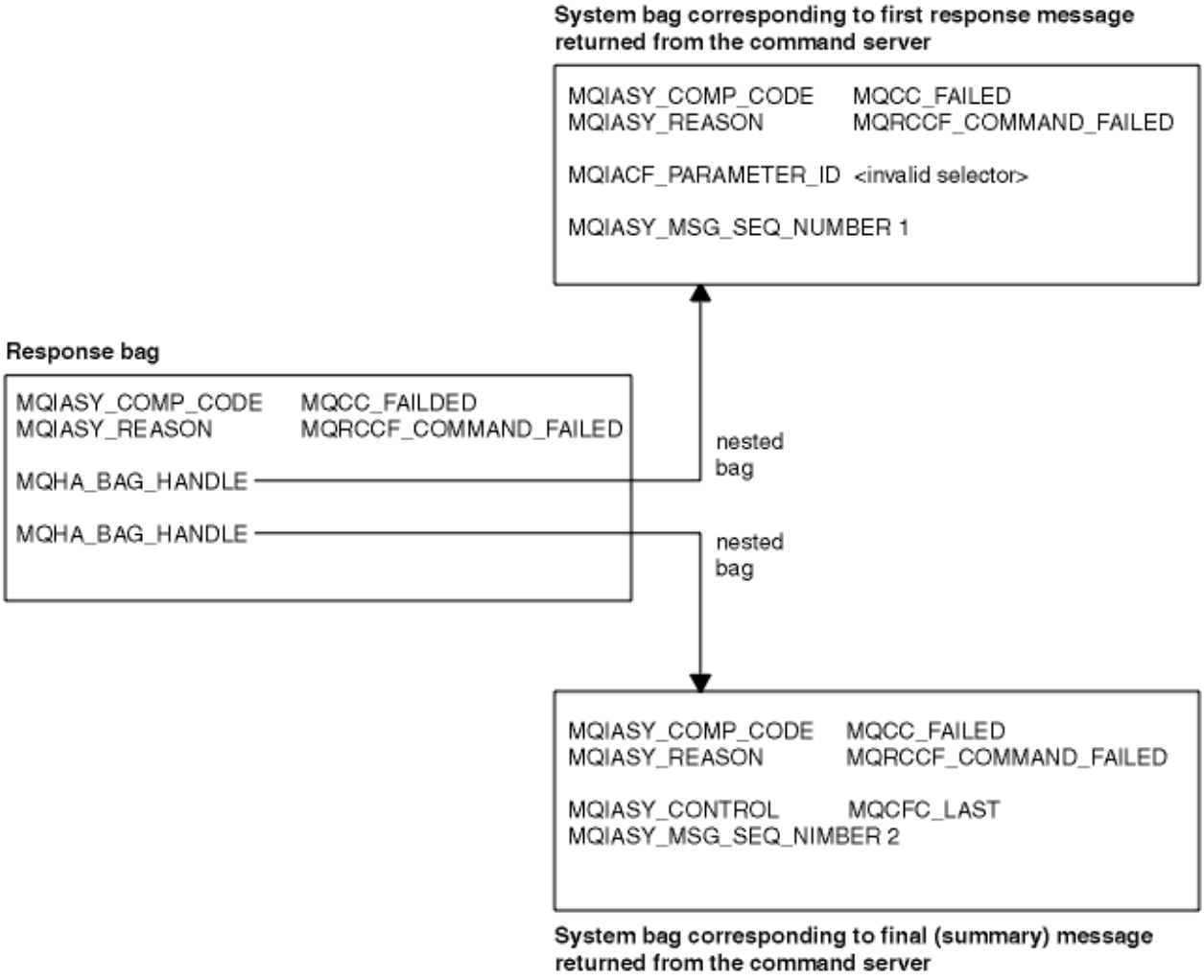
使用 MQAI 時的程式設計提示。

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。 以下是使用 MQAI 來配置 WebSphere MQ 的一些提示:

- WebSphere 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 WebSphere 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 50 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 MQAI 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 39 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 40 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 41 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 40 頁的圖 1 所示。

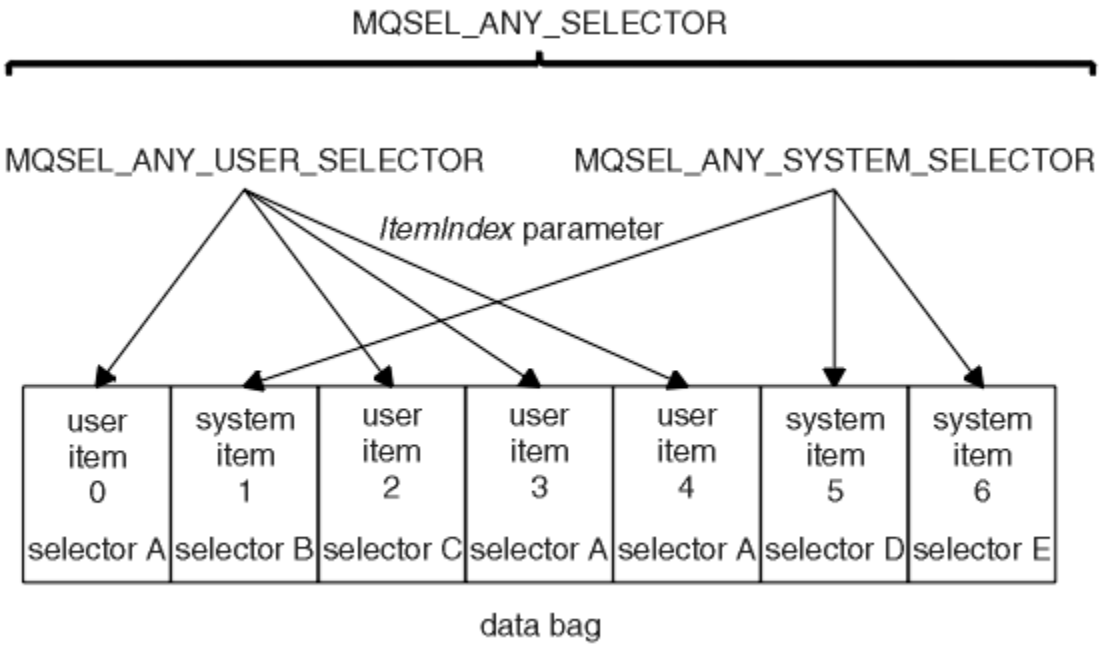


圖 *1:* 編製索引

在圖 第 40 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 40 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 41 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 42 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 42 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 42 頁的表 4。
Expiry	30 秒 (附註 第 43 頁的『1』)
Feedback	MQFB_NONE
Encoding	MQENC_NATIVE
CodedCharSetId	取決於工具袋 CCSID (附註 第 43 頁的『2』)
Format	請參閱第 42 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 43 頁的『3』
ReplyToQMgr	空白

表 5: 訊息描述子值 (繼續)	
參數	**值**
附註: 1. 在 **mqExecute** 呼叫中，可以使用 **OptionsBag** 參數來置換此值。 如需相關資訊，請參閱 **mqExecute**。 2. 請參閱 第 40 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 45 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 39 頁的『在 MQAI 中編製索引』中說明。

  圖 1 顯示這些概念的階層。

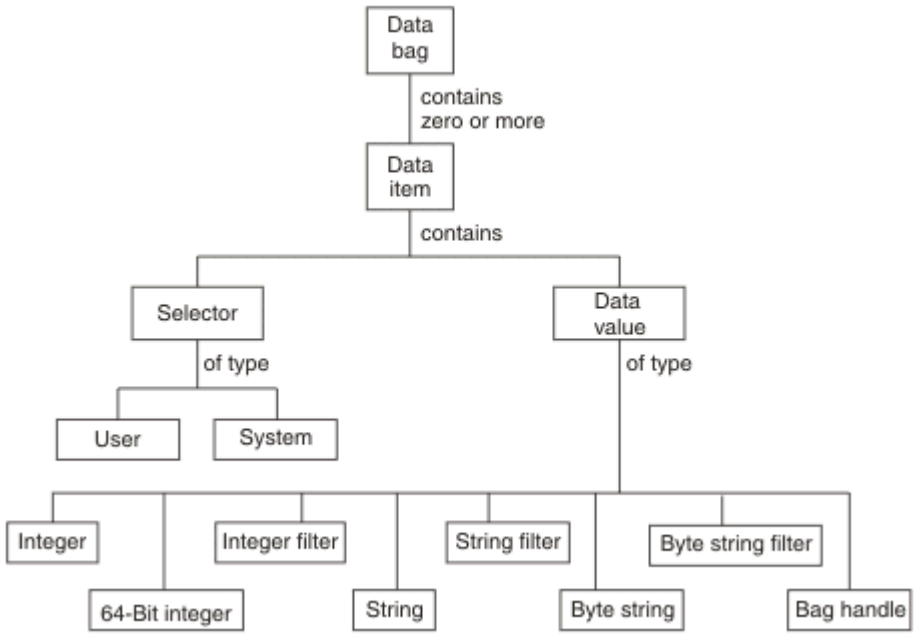


圖 2: *MQAI* 概念的階層

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 WebSphere 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 44 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 WebSphere MQ 物件的指令建立工具袋。 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 44 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 WebSphere 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 44 頁的『建立及刪除資料工具袋』。
- 您可以使用資料工具袋 第 45 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 46 頁的『將資料項目新增至工具袋』。
- 您可以在資料工具袋 第 46 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 47 頁的『在資料袋內查詢』內查詢。
- 您可以對資料工具袋 第 49 頁的『計數資料項目』內的資料項目進行計數。
- 您可以在資料工具袋 第 47 頁的『變更工具袋內的資訊』內變更資訊。
- 您可以清除資料工具袋 第 48 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 48 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 49 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 45 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 50 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

資料項目是用來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 45 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 49 頁的『計數資料項目』.
- 第 49 頁的『刪除資料項目』.
- 第 46 頁的『將資料項目新增至工具袋』.
- 第 46 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 47 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 49 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 45 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 46 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。

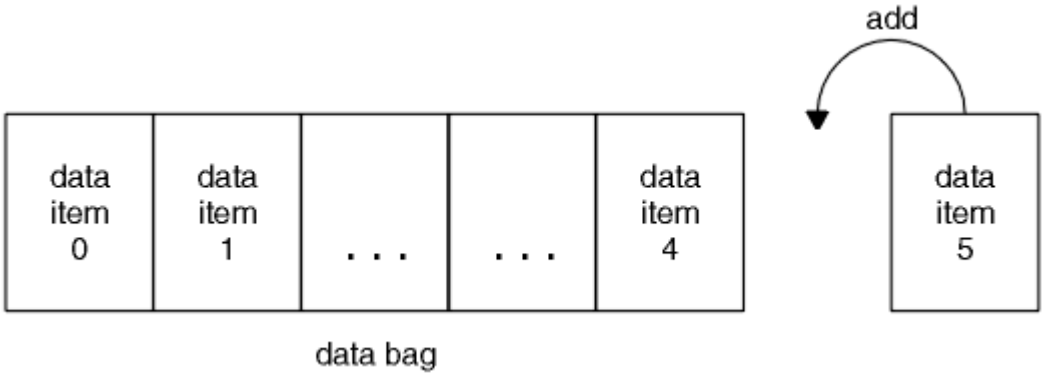


圖 *3:* 新增資料項目

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 45 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 WebSphere 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 WebSphere 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 8 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 18 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 48 頁的圖 4)。

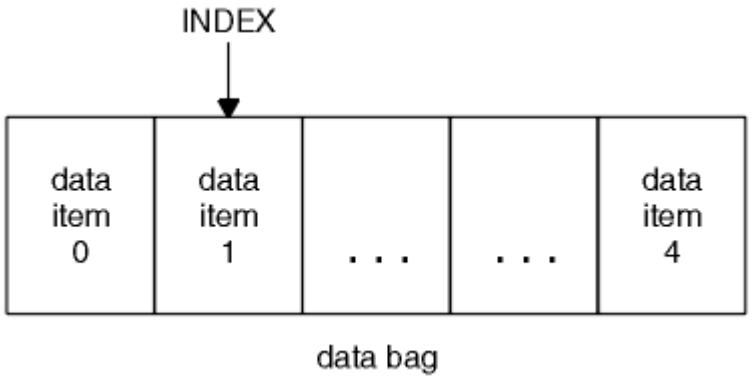


圖 *4:* 修改單一資料項目

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 48 頁的圖 5。) 特殊索引值容許取代參數的 **所有** 實例。

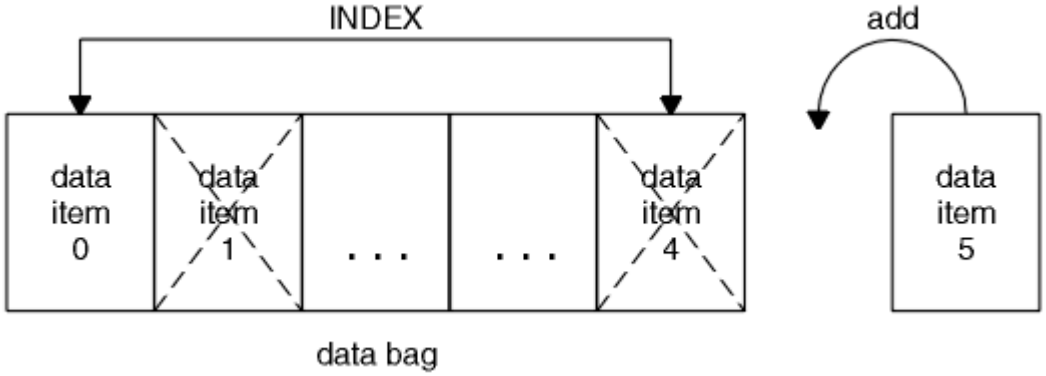


圖 *5:* 修改所有資料項目

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。

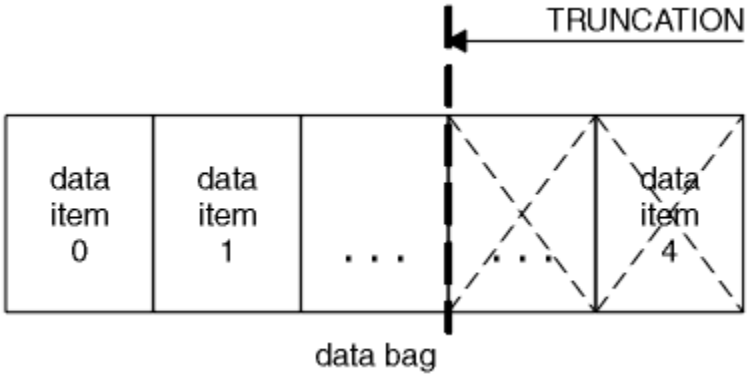


圖 *6:* 截斷工具袋

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。 此圖顯示在 第 49 頁的圖 7 中。 如需 mqBagToBuffer 呼叫的完整說明，請參閱 mqBagToBuffer。

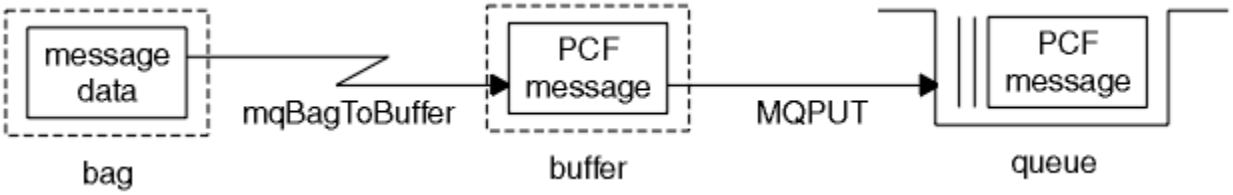


圖 *7:* 將工具袋轉換成 *PCF* 訊息

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 49 頁的圖 8 中顯示。 如需 mqBufferToBag 呼叫的完整說明，請參閱 mqBufferToBag。

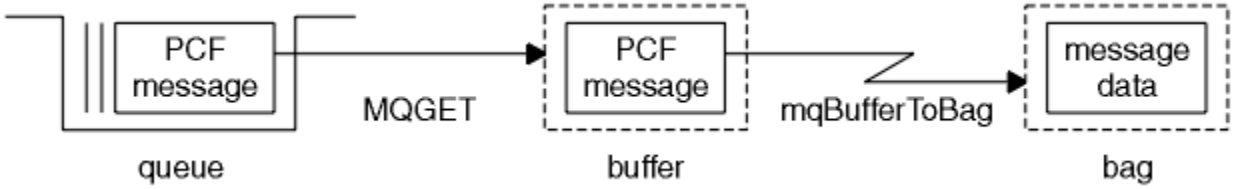


圖 *8:* 將 *PCF* 訊息轉換成工具袋表單

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 50 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 48 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 48 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 50 頁的圖 9。)

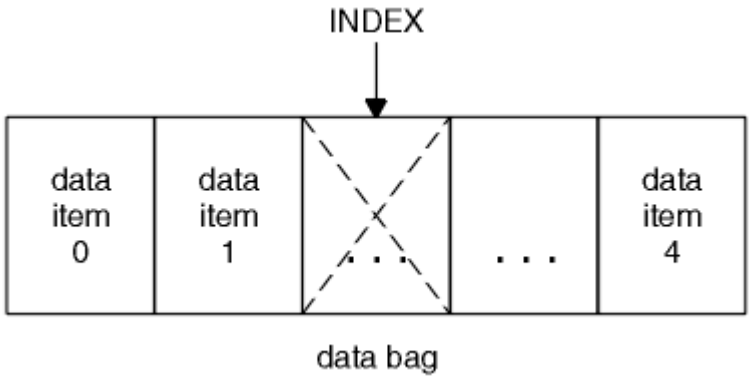


圖 *9:* 刪除單一資料項目

or

2. 指定選取元的所有出現項目。 (請參閱 第 50 頁的圖 10。)

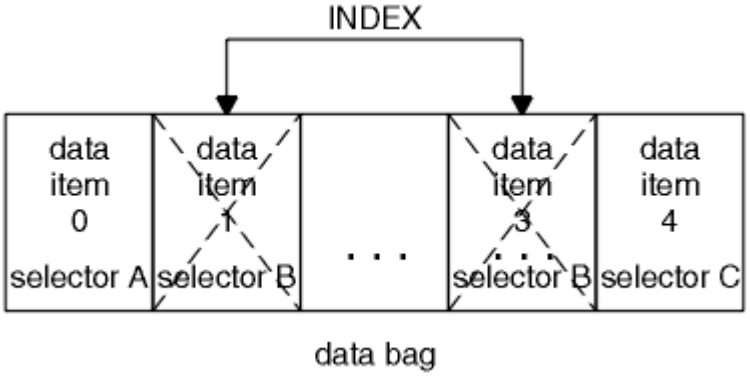


圖 *10:* 刪除所有資料項目

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 WebSphere 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。

第 51 頁的圖 11 中顯示 巢狀 的概念。

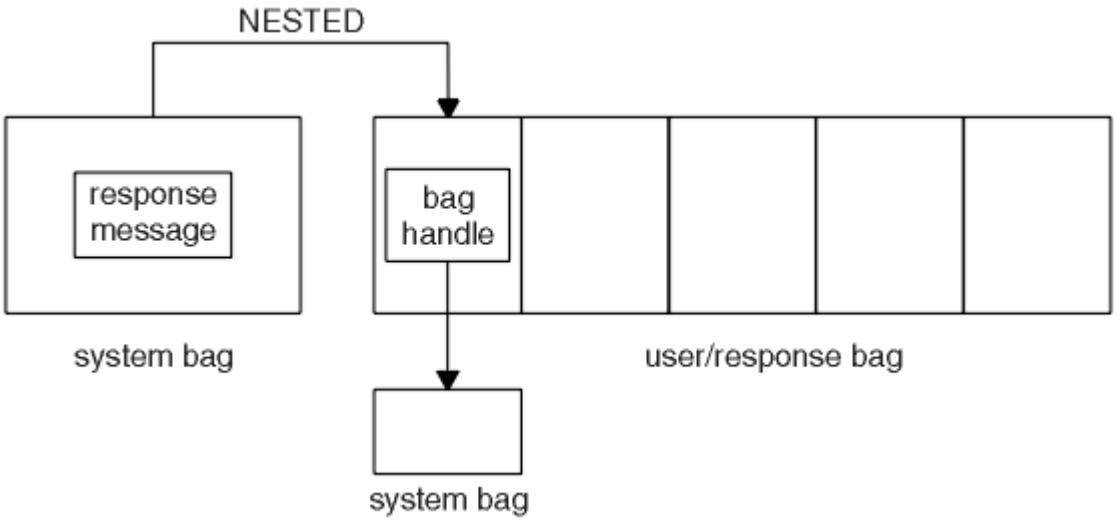


圖 *11:* 巢狀*(N)*

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 IBM WebSphere MQ Explorer 進行管理

IBM WebSphere MQ Explorer 只容許您從執行 Windows 或 Linux (x86 及 x86-64 平台) 的電腦執行網路的本端或遠端管理。

IBM WebSphere MQ for Windows 和 IBM WebSphere MQ for Linux (x86 和 x86-64 平台) 提供稱為 IBM WebSphere 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 IBM WebSphere MQ Explorer 執行的動作。

IBM WebSphere MQ Explorer 可讓您從執行 Windows 或 Linux (x86-64 平台) 的電腦執行網路的本端或遠端管理，方法是將 IBM WebSphere MQ Explorer 指向您感興趣的佇列管理程式及叢集。 第 53 頁的『遠端佇列管理程式』中說明可以使用 IBM WebSphere MQ Explorer 來管理的 IBM WebSphere MQ 平台及層次。

若要配置遠端 IBM WebSphere MQ 佇列管理程式，以便 IBM WebSphere MQ Explorer 可以管理它們，請參閱 第 53 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 及 x86-64 平台) 系統網域本端或遠端設定及細部調整 IBM WebSphere MQ 的工作環境相關聯。

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

在 Linux 上，若要順利啟動 IBM WebSphere 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 WebSphere MQ Explorer 執行的動作

這是您可以使用 IBM WebSphere MQ Explorer 執行的作業清單。

使用「 IBM WebSphere 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 WebSphere 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 ikeyman GUI 來管理 Secure Socket 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 WebSphere MQ 物件建立 JMS 物件，從 JMS 物件建立 WebSphere 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 WebSphere 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「 WebSphere MQ 探險家」。 「 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，「WebSphere 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:

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

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

如果您想要使用 WebSphere MQ V6.0 或更新版本電腦上的「WebSphere MQ 探險家」，在 iSeries 上遠端管理 V5.3 佇列管理程式， 您必須在 WebSphere MQ for iSeries V5.3 電腦上安裝 Fix Pack 11 (CSD11) 或更新版本。 此修正套件會更正「WebSphere 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 WebSphere MQ Explorer

在決定是否在安裝時使用 IBM WebSphere 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
如果您使用「IBM WebSphere MQ 探險家」對佇列管理程式及其他物件使用小寫名稱，當您使用 MQSC 指令處理物件時，必須以單引號括住物件名稱，否則 WebSphere MQ 無法辨識它們。

大型佇列管理程式
「IBM WebSphere 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「WebSphere 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
WebSphere MQ 叢集可能包含數百或數千個佇列管理程式。 「WebSphere MQ 探險家」會使用樹狀結構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 WebSphere MQ 探險家」的速度，因為在您選取之前，「IBM WebSphere MQ 探險家」不會連接至叢集中的佇列管理程式。

設定 IBM WebSphere MQ Explorer

本節概述設定 IBM WebSphere MQ Explorer 所需的步驟。

- 第 53 頁的『必備軟體及定義』
- 第 54 頁的『安全』
- 第 57 頁的『顯示及隱藏佇列管理程式和叢集』
- 第 57 頁的『叢集成員關係』
- 第 58 頁的『資料轉換』

必備軟體及定義

在嘗試使用 IBM WebSphere MQ Explorer 之前，請確定您滿足下列需求。

「IBM WebSphere 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 WebSphere MQ 接聽器，也可以是在 UNIX and 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 WebSphere MQ ， 您可能需要考量使用「 IBM WebSphere MQ 探險家」的安全層面。

使用 *IBM WebSphere MQ Explorer* 的授權

任何使用者都可以使用 IBM WebSphere MQ Explorer，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「 WebSphere MQ 探險家」來執行本端管理作業，使用者必須具備必要的權限，才能執行管理作業。 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「 WebSphere 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「 WebSphere 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 WebSphere MQ for z/OS 上的遠端佇列管理程式，並使用「 IBM WebSphere 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「 WebSphere MQ 探險家」的使用者必須具備下列權限:

- RACF 對系統佇列 SYSTEM.MQEXPLORER.REPLY.MODEL
- RACF 對佇列 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 WebSphere MQ 物件權限的相關資訊，請參閱 授與對 UNIX 或 Linux 系統及 Windows 上 WebSphere 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。

「 WebSphere MQ 探險家」中的預設過濾器是顯示所有 WebSphere MQ 物件。 如果有任何使用者沒有 DISPLAY 權限的 WebSphere 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 WebSphere MQ 探險家」與每一個遠端佇列管理程式之間的通道安全。

「IBM WebSphere 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「*WebSphere MQ* 探險家」

您可以使用「WebSphere 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。

您可以定義預設安全結束程式，它可用於來自「WebSphere MQ 探險家」的所有新用戶端連線。 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「WebSphere MQ 探險家」來指定結束程式。 如需相關資訊，請參閱 WebSphere MQ 說明中心。

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

「IBM WebSphere 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 WebSphere MQ MQI 用戶端概觀。

當您使用用戶端通道定義表建立通道時，可以使用「IBM WebSphere MQ Explorer 」，利用啟用 SSL 的 MQI 通道來連接遠端佇列管理程式，如 第 55 頁的『管理遠端佇列管理程式之系統上的作業』 及 第 55 頁的『管理 IBM WebSphere MQ Explorer 之系統上的作業』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和用戶端連線配對，並在兩個通道的伺服器連線上指定適當的 *SSLCIPH* 變數值。 如需 *SSLCIPH* 變數的相關資訊，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 IBM WebSphere 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 and Linux 系統)
   - `C:\Program Files\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。 透過檢查 iKeyman GUI 中的選項，或搭配使用 `-stash` 選項與 **runmqckm** 指令，將密碼隱藏在檔案中。
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 IBM WebSphere MQ Explorer 之系統上的作業

在管理 IBM WebSphere 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。

IBM WebSphere MQ Explorer 使用 Java 金鑰儲存庫檔 (JKS) 來確保 SSL 安全，因此為了配置 IBM WebSphere 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 及 Linux 系統上，使用系統功能表、 `MQExplorer` 執行檔或 **strmqcfg** 指令來啟動「MQ 探險家」。
5. 從 IBM WebSphere MQ Explorer 工具列中，按一下 **視窗-> 喜好設定**，然後展開 **WebSphere MQ 探險家**，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 55 頁的『管理 IBM WebSphere MQ Explorer 之系統上的作業』步驟 1 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 55 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格檔案的位置。
9. 按一下**完成**。 您現在可以從「IBM WebSphere 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 WebSphere MQ 探險家」可讓您透過「IBM WebSphere 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 WebSphere 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。

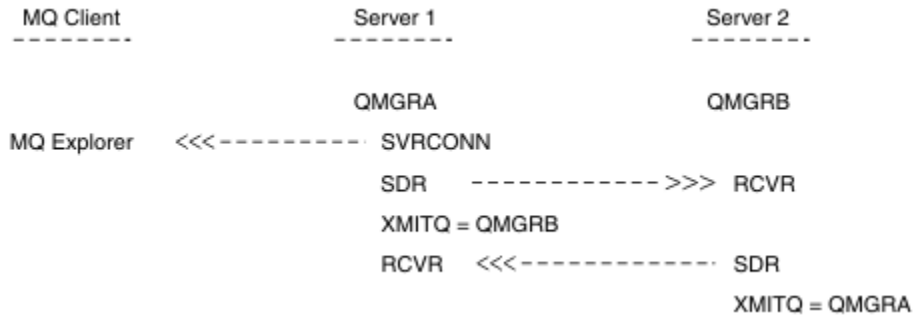
如果接著使用連線透過中繼佇列管理程式連接至目標佇列管理程式，則會再次在訊息描述子 (MQMD) 的 *UserIdentifier* 參數中傳送 userId。 為了讓目標佇列管理程式上的 MCA 接聽器接受此訊息，必須設定 MCAUSER 屬性，或 userId 必須已存在且具有放置權限。

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「WebSphere 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 WebSphere 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 WebSphere MQ 探險家」
- 使用「WebSphere MQ 探險家」連接至 QMGRB 時，請選取 QMGRA 作為中間佇列管理程式

在此狀況下，沒有從「IBM WebSphere 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 95 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

「IBM WebSphere MQ 探險家」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 Queue Managers 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「IBM WebSphere MQ 探險家」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「WebSphere MQ 探險家」的「導覽」窗格中，用滑鼠右鍵按一下頂端 **IBM WebSphere MQ** 樹狀結構節點，然後選取 **匯出 MQ 探險家設定**
3. 按一下 **MQ 探險家> MQ 探險家設定**
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「WebSphere MQ 探險家」的「導覽」窗格中，用滑鼠右鍵按一下頂端 **IBM WebSphere MQ** 樹狀結構節點，然後選取 **匯入 MQ 探險家設定** 。
2. 按一下 **MQ 探險家> MQ 探險家設定**
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「WebSphere MQ 探險家」。

叢集成員關係

「IBM WebSphere 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 WebSphere 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 WebSphere 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。為了這樣做，「WebSphere 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)

利用此資訊，「WebSphere MQ 探險家」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。「WebSphere MQ 探險家」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。
- 所選擇的儲存庫正在「WebSphere 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 WebSphere 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 WebSphere MQ Explorer 以 CCSID 1208 (UTF-8) 運作。這可讓「IBM WebSphere 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 WebSphere 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 WebSphere MQ 探險家」與具有「IBM WebSphere 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。

Windows 上的安全

「準備 WebSphere MQ」精靈會建立特殊使用者帳戶，以便需要使用它的處理程序可以共用 Windows 服務。

Windows 服務在 IBM WebSphere MQ 安裝的用戶端程序之間共用。每個安裝都會建立一個服務。每一個服務都命名為 `MQ_InstallationName`，且顯示名稱為 `IBM WebSphere MQ(InstallationName)`。在 Version 7.1 之前 (在伺服器上只有一個安裝)， Windows 服務命名為 `MQSeriesServices`，顯示名稱為 `IBM MQSeries`。

因為每一個服務必須在非互動式及互動式登入階段作業之間共用，所以您必須在特殊使用者帳戶下啟動每一個服務。您可以對所有服務使用一個特殊使用者帳戶，或建立不同的特殊使用者帳戶。每一個特殊使用者帳戶都必須具有 "以服務方式登入"的使用者權限，如需相關資訊，請參閱 第 59 頁的『IBM WebSphere MQ Windows 服務所需的使用者權限』。如果使用者 ID 沒有執行服務的權限，則服務不會啟動，且會在 Windows 系統事件日誌中傳回錯誤。一般而言，您必須執行「準備 IBM WebSphere MQ」精靈，並正確設定使用者 ID。不過，如果您已手動配置使用者 ID，是否可能有您需要解決的問題。

當您第一次安裝 IBM WebSphere MQ 並執行「準備 IBM WebSphere MQ」精靈時，它會為稱為 MUSR_MQADMIN 的服務建立本端使用者帳戶，並具備必要的設定和許可權，包括 "以服務方式登入"。

對於後續安裝，「準備 IBM WebSphere MQ」精靈會建立名為 MUSR_MQADMINx 的使用者帳戶，其中 x 是下一個可用的號碼，代表不存在的使用者 ID。MUSR_MQADMINx 的密碼是在建立帳戶時隨機產生，用來配置服務的登入環境。產生的密碼不會到期。

此 IBM WebSphere MQ 帳戶不受系統上設定的任何帳戶原則所影響，這些原則會要求在特定期間之後變更帳戶密碼。

密碼在此一次性處理之外不明，並由 Windows 作業系統儲存在登錄的安全部分中。

使用 Active Directory (僅限 Windows)

在某些網路配置中，當使用者帳戶定義在使用 Active Directory 的網域控制站上時，執行中的本端使用者帳戶 IBM WebSphere MQ 可能沒有查詢其他網域使用者帳戶的群組成員資格所需的權限。 「準備 IBM WebSphere MQ 精靈」會執行測試並詢問使用者有關網路配置的問題，以識別是否有這種情況。

如果執行所在的本端使用者帳戶 IBM WebSphere MQ 沒有必要的權限，「準備 IBM WebSphere MQ 精靈」會提示使用者輸入具有特定使用者權限之網域使用者帳戶的帳戶詳細資料。 如需網域使用者帳戶需要的使用者權限，請參閱 第 59 頁的『IBM WebSphere MQ Windows 服務所需的使用者權限』。 當使用者在「準備 IBM WebSphere MQ 精靈」中輸入網域使用者帳戶的有效帳戶詳細資料之後，它會將 IBM WebSphere MQ Windows 服務配置成在新帳戶下執行。 帳戶詳細資料保留在「登錄」的安全部分中，使用者無法讀取。

當服務在執行中，只要 IBM WebSphere MQ Windows 服務在執行中，該服務就會啟動並保持執行中。 在啟動 Windows 服務之後登入伺服器的 IBM WebSphere MQ 管理者可以使用 IBM WebSphere MQ Explorer 來管理伺服器上的佇列管理程式。 這會將 IBM WebSphere MQ Explorer 連接至現有的 Windows 服務程序。 這兩個動作需要不同的許可權層次才能運作:

- 啟動程序需要啟動許可權。
- IBM WebSphere MQ 管理者需要存取權。

IBM WebSphere MQ Windows 服務所需的使用者權限

本主題中的表格列出執行 IBM WebSphere MQ 安裝的 Windows 服務所使用的本端及網域使用者帳戶所需的使用者權限。

以批次工作登入	啟用 IBM WebSphere MQ Windows 服務，以在此使用者帳戶下執行。
以服務方式登入	可讓使用者設定 IBM WebSphere MQ Windows 服務，以使用已配置的帳戶登入。
關閉系統	如果配置為在服務回復失敗時重新啟動伺服器，則容許 IBM WebSphere MQ Windows 服務重新啟動伺服器。
增加配額	作業系統 `CreateProcessAsUser` 呼叫的必要項目。
作為作業系統的一部分	作業系統 `LogonUser` 呼叫的必要項目。
略過遍訪檢查	作業系統 `LogonUser` 呼叫的必要項目。
更換程序層記號	作業系統 `LogonUser` 呼叫的必要項目。

註: 在執行 ASP 及 IIS 應用程式的環境中可能需要除錯程式權限。

您的網域使用者帳戶必須將這些 Windows 使用者權限設為「本機安全性原則」應用程式中列出的有效使用者權限。 如果沒有，請在伺服器本端使用「本機安全性原則」應用程式，或使用「網域安全性應用程式」網域範圍來設定它們。

變更與 IBM WebSphere MQ 服務相關聯的使用者名稱

您可能需要將與 IBM WebSphere MQ 服務相關聯的使用者名稱從 MUSR_MQADMIN 變更為其他名稱。 (例如，如果佇列管理程式與 DB2 相關聯，且不接受超過 8 個字元的使用者名稱，則您可能需要執行此動作。)

程序

1. 建立新的使用者帳戶 (例如 **NEW_NAME**)
2. 使用「準備 IBM WebSphere MQ 精靈」來輸入新使用者帳戶的詳細資料。

變更 IBM WebSphere MQ Windows 服務使用者帳戶的密碼

關於這項作業
若要變更 IBM WebSphere MQ Windows 服務本端使用者帳戶的密碼，請執行下列步驟:

程序
1. 識別執行服務的使用者。
2. 從「電腦管理」畫面停止 IBMIBM WebSphere MQ 服務。
3. 變更所需密碼的方式與您變更個人密碼的方式相同。
4. 從「電腦管理」畫面移至 IBM WebSphere MQ 服務的內容。
5. 選取 **登入** 頁面。
6. 請確認指定的帳戶名稱符合已修改密碼的使用者。
7. 在 **密碼** 和 **確認密碼** 欄位中鍵入密碼，然後按一下 **確定**。

以網域使用者帳戶執行安裝的 *IBM WebSphere MQ Windows* 服務

關於這項作業
如果安裝的 IBM WebSphere MQ Windows 服務在網域使用者帳戶下執行，您也可以變更帳戶的密碼，如下所示:

程序
1. 變更網域控制站上網域帳戶的密碼。 您可能需要要求網域管理者為您執行此動作。
2. 遵循步驟來修改 IBMIBM WebSphere MQ 服務的「 **登入** 」頁面。

   執行 IBM WebSphere MQ Windows 服務的使用者帳戶會執行使用者介面應用程式所發出的任何 MQSC 指令，或在系統啟動、關閉或服務回復時自動執行的任何 MQSC 指令。 因此，此使用者帳戶必須具有 IBM WebSphere MQ 管理權限。 依預設，它會新增至伺服器上的本端 **mqm** 群組。 如果移除此成員資格，則 IBM WebSphere MQ Windows 服務無法運作。 如需使用者權限的相關資訊，請參閱 第 59 頁的『IBM WebSphere MQ Windows 服務所需的使用者權限』

   如果執行 IBM WebSphere MQ Windows 服務的使用者帳戶發生安全問題，系統事件日誌中會出現錯誤訊息和說明。

相關概念

第 59 頁的『使用 Active Directory (僅限 Windows)』
在某些網路配置中，當使用者帳戶定義在使用 Active Directory 的網域控制站上時，執行中的本端使用者帳戶 IBM WebSphere MQ 可能沒有查詢其他網域使用者帳戶的群組成員資格所需的權限。 「準備 IBM WebSphere MQ 精靈」會執行測試並詢問使用者有關網路配置的問題，以識別是否有這種情況。

IBM WebSphere MQ 與作為資源管理程式的 Db2 協調

如果您從「 IBM WebSphere MQ Explorer」啟動佇列管理程式，或正在使用 IBM WebSphere MQ V7，且在協調 Db2 時發生問題，請檢查佇列管理程式錯誤日誌。

請檢查佇列管理程式錯誤日誌，以找出類似下列的錯誤:

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzxma0.exe)
Host(HOST_1) Installation(Installation1)
VMRF(7.1.0.0) QMgr(A.B.C)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available when called
    for xa_open. The queue manager is continuing without this resource manager.
```

說明: 執行 IBM WebSphere MQ 服務處理程序 `amqsvc.exe` 的使用者 ID (預設名稱為 MUSR_MQADMIN) 仍在執行中，其存取記號不包含群組 DB2USERS 的群組成員資格資訊。

解決: 確定 IBM WebSphere MQ 服務使用者 ID 是 DB2USERS 的成員之後，請使用下列指令順序:

- 停止該服務。
- 停止在相同使用者 ID 下執行的任何其他處理程序。

- 重新啟動這些程序。

將機器重新開機可確保先前的步驟，但並非必要。

延伸 IBM WebSphere MQ Explorer

IBM WebSphere MQ for Windows 及 IBM WebSphere MQ for Linux (x86 及 x86-64 平台) 提供稱為 IBM WebSphere 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 WebSphere MQ for Windows 及 WebSphere MQ for Linux (x86 及 x86-64 平台)。

「IBM WebSphere MQ 檔案總管」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「IBM WebSphere MQ 探險家」，系統管理者能夠自訂「WebSphere MQ 探險家」，以改善其管理 WebSphere MQ 的方式。

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

使用 IBM WebSphere MQ 工作列應用程式 (僅限 Windows)

「IBM WebSphere 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 WebSphere 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， WebSphere MQ 圖示位於伺服器上的系統匣中，並以彩色編碼狀態符號覆蓋，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; WebSphere 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 WebSphere 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 WebSphere MQ 警示監視器
- 按一下 **結束** 以結束 WebSphere MQ 工作列應用程式
- 按一下 **WebSphere MQ 探險家** ，以啟動「IBM WebSphere MQ 探險家」
- 按一下 **停止 WebSphere MQ** 以停止 WebSphere MQ
- 按一下 **關於 WebSphere MQ** ，以顯示 WebSphere MQ 警示監視器的相關資訊

IBM WebSphere MQ 警示監視器應用程式 (僅限 Windows)

IBM WebSphere 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 WebSphere MQ 的問題。

警示監視器會顯示 WebSphere MQ 伺服器本端安裝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 WebSphere MQ 警示監視器中，您可以:

- 直接存取 IBM WebSphere MQ Explorer
- 檢視所有未解決警示的相關資訊
- 關閉本端機器上的 WebSphere 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 WebSphere MQ 物件

本節告訴您如何管理本端 IBM WebSphere 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 WebSphere MQ 物件。

除了本節中詳述的方法之外，您還可以使用「 IBM WebSphere MQ 探險家」來管理本端 WebSphere MQ 物件; 請參閱 第 51 頁的『使用 IBM WebSphere MQ Explorer 進行管理』。

本節包含下列資訊:

- 使用 MQI 的應用程式
- 第 65 頁的『使用 MQSC 指令執行本端管理作業』
- 第 72 頁的『使用佇列管理程式』
- 第 74 頁的『使用本端佇列』
- 第 78 頁的『使用別名佇列』
- 第 79 頁的『使用模型佇列』
- 第 85 頁的『使用服務』
- 第 91 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 WebSphere MQ for Windows 及 WebSphere 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 WebSphere 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， WebSphere MQ 會間歇性發出參考訊息，詳述啟動進度。

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 WebSphere MQ for Windows 中，當系統使用「 WebSphere 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 51 頁的『使用 IBM WebSphere 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 WebSphere MQ for Windows 及 WebSphere MQ for Linux (x86 及 x86-64 平台) 系統上，您可以如下停止佇列管理程式:

1. 開啟 IBM WebSphere 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 63 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 建立及管理佇列管理程式。

在 Windows 上停止佇列管理程式

如何在 IBM WebSphere MQ for Windows 中結束處理程序及 IBM WebSphere MQ 服務，以停止佇列管理程式。

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

1. 使用「Windows 工作管理員」，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** 指令來結束處理程序，順序如下 (如果它們在執行中):

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)
AMQZDMAA	延遲訊息處理器
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
AMQZTRCN	追蹤

3. 從 Windows 控制台上的 **管理工具** > **服務** 停止 WebSphere 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 UNIX and Linux 系統上停止佇列管理程式

如何結束處理程序及 IBM WebSphere MQ 服務，以停止 IBM WebSphere MQ for UNIX and 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 WebSphere MQ for UNIX and 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。

   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)
阿姆克茲德馬	延遲訊息處理器
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，則可能會取得 FFST ， 並將 FDC 檔放置在 /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。

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 WebSphere MQ for Windows 或 IBM WebSphere MQ for Linux (x86 及 x86-64 平台) ，則也可以使用「 IBM WebSphere MQ 檔案總管」來執行本節中說明的作業。 如需相關資訊，請參閱第 51 頁的『使用 IBM WebSphere 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; 如需管理通道、用戶端連線通道及接聽器物件的相關資訊，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 66 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 69 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。

MQSC 指令在 MQSC 參照 小節的主題集合中彙總。

Script (MQSC) 指令

MQSC 指令提供在 WebSphere 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 8 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。
- 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)。
- 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:
  - 以星號
  - 單一星號 (例如 TRACE (*))
  - 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。

- 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 WebSphere MQ 的概觀，請參閱 第 65 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 MQSC 指令的完整清單，請參閱 MQSC 指令。

相關工作
建置指令 Script

WebSphere 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 MQSC 參照。

在某些 WebSphere MQ 環境中，與所有 WebSphere MQ 控制指令相同的 runmqsc 指令呼叫會區分大小寫。 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 WebSphere 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: WebSphere 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 參照 以取得正確的語法。

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 67 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 69 頁的圖 12 顯示 MQSC 指令檔的擷取 `myprog.in` ，而 第 70 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 69 頁的圖 12 是從 MQSC 指令檔中擷取的，其中顯示具有其屬性的 MQSC 指令 (DEFINE QLOCAL)。 MQSC 參照 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 WebSphere 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 報告，並傳送至 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 70 頁的圖 13 中所示。 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: WebSphere 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: WebSphere MQ queue created.
       :
 .
 .
```

圖 *13:* 從 *MQSC* 指令報告檔擷取

執行提供的 MQSC 指令檔

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

在 UNIX and Linux 系統上，這些檔案位於 *MQ_INSTALLATION_PATH*/samp 目錄中。
MQ_INSTALLATION_PATH 代表 WebSphere 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 70 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 WebSphere 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。

如果您無法讓 MQSC 指令執行，請使用下列資訊來查看是否有任何這些一般問題適用於您。 當您讀取所產生的錯誤時，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: WebSphere 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: WebSphere 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 73 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(750)                           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

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。

相關參考
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 參照 。

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 75 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 75 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。 您必須告知佇列管理程式無法傳送郵件的佇列。

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。

WebSphere 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 使用 WebSphere MQ 無法傳送郵件的佇列處理程式來處理無法遞送的訊息 。

顯示預設物件屬性

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

當您定義 WebSphere 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 74 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

WebSphere MQ 提供範例佇列瀏覽器，可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 WebSphere MQ 安裝所在的高階目錄。

在 WebSphere 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 WebSphere MQ for UNIX and 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 77 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'0000000000000000000000000000000000000000000000000000000000000000'
  BackoutCount : 0
  ReplyToQ       : '                                                '
  ReplyToQMgr    : 'saturn.queue.manager                          '
  ** Identity Context
  UserIdentifier : '            '
  AccountingToken :
   X'0000000000000000000000000000000000000000000000000000000000000000'
  ApplIdentityData : '                                        '
  ** Origin Context
  PutApplType    : '7'
  PutApplName    : 'saturn.queue.manager        '
  PutDate  : '19970417'    PutTime  : '15115208'
  ApplOriginData : '    '

  GroupId : X'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

啟用大型佇列

IBM WebSphere 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 及 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 WebSphere MQ 網站，以取得平台專用效能報告:

```
https://www.ibm.com/software/integration/ts/mqseries/
```

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 7.5.0.8

小心: 發佈清單不支援使用指向主題物件的別名佇列。 從 Version 7.5.0, Fix Pack 8 開始，如果別名佇列指向發佈清單中的主題物件，則 IBM WebSphere 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 74 頁的『定義本端佇列』)。
- 遠端佇列的本端定義 (請參閱 第 99 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 TARGET 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 QALIAS (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) TARGET(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。 如需此及其他別名佇列指令的相關資訊，請參閱 MQSC 參照 。

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 參照 。

相關概念

管理主題物件

第 81 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 81 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 81 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 82 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 82 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。

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 。 如需主題樹狀結構的相關資訊，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 81 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 81 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。 如需主題樹狀結構的相關資訊，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 參照 。

相關概念
第 83 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 83 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 84 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 85 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 85 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。
- 除了說明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 83 頁的『顯示訂閱的屬性』。
- 如果您已在名稱為 TEST 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 84 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: WebSphere 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: WebSphere 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: WebSphere 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 83 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 76 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 WebSphere MQ 服務。

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 86 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 87 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 and 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 and Linux 系統上: /var/mqm/qmgrs/QMNAME/service.env
  - 在 Windows 系統上: C:\Program Files\IBM\WebSphere 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 WebSphere 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, 2024.                       *#
#*                                                                 *#
#* <NOC_COPYRIGHT>                                                 *#
#*                                                                 *#
#*******************************************************************#
#*******************************************************************************#
#* Module Name: service.env                                                      *#
#* Type       : WebSphere 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 88 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
WebSphere MQ 的安裝位置。

MQ_DATA_PATH
WebSphere MQ 資料目錄的位置:

- 在 UNIX and Linux 系統上， WebSphere MQ 資料目錄位置是 /var/mqm/
- 在 Windows 系統上， WebSphere MQ 資料目錄的位置是在安裝 WebSphere 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 WebSphere 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 89 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 WebSphere MQ 提供的範例程式，要求佇列管理程式中斷程序 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 88 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 89 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 and 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 WebSphere 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。

此範例使用 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

WebSphere 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。

使用觸發程式來啟動 WebSphere MQ 應用程式中詳細說明的觸發。

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。

管理遠端 IBM WebSphere 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 WebSphere 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:

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 WebSphere 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 WebSphere 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 95 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 WebSphere 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。

如需叢集、其屬性及如何設定它們的相關資訊，請參閱 佇列管理程式叢集。

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 9 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 WebSphere MQ 環境中遠端管理佇列管理程式，包括 UNIX and Linux 系統、 Windows 系統 IBM i 及 z/OS。

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 95 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。

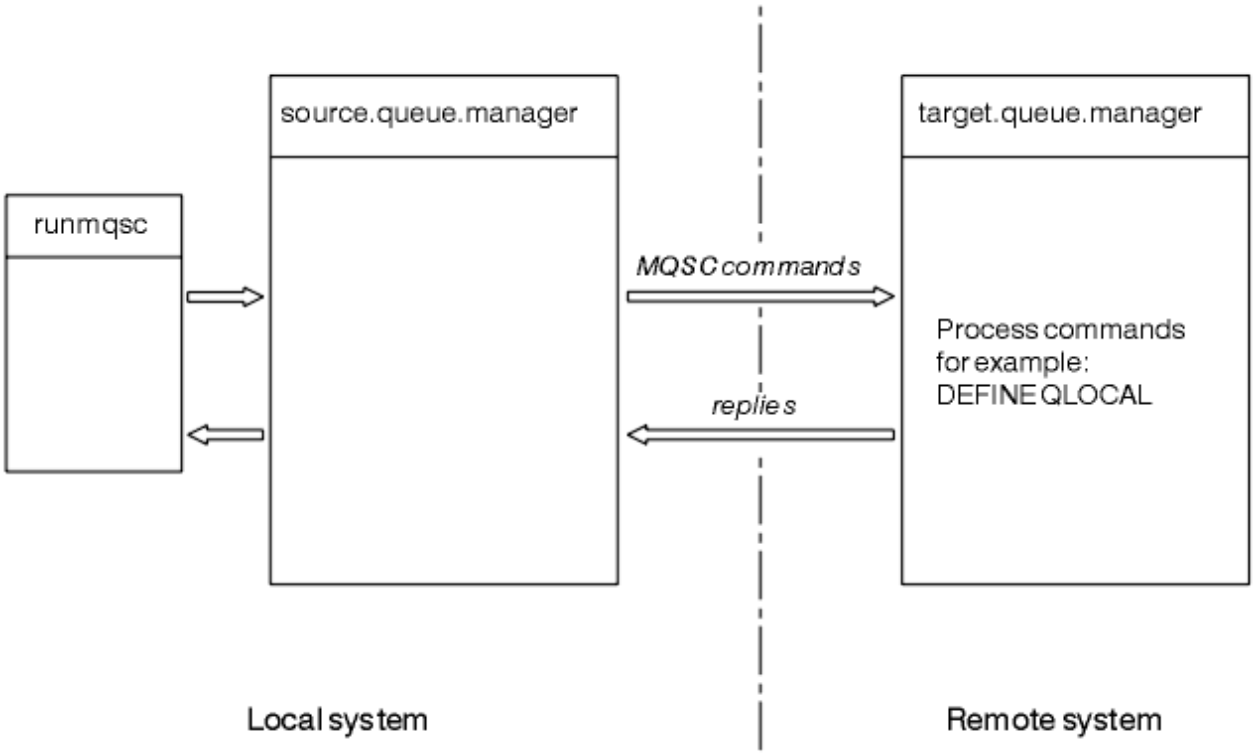


圖 *17:* 使用 *MQSC* 指令進行遠端管理

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 96 頁的圖 18 彙總此配置。

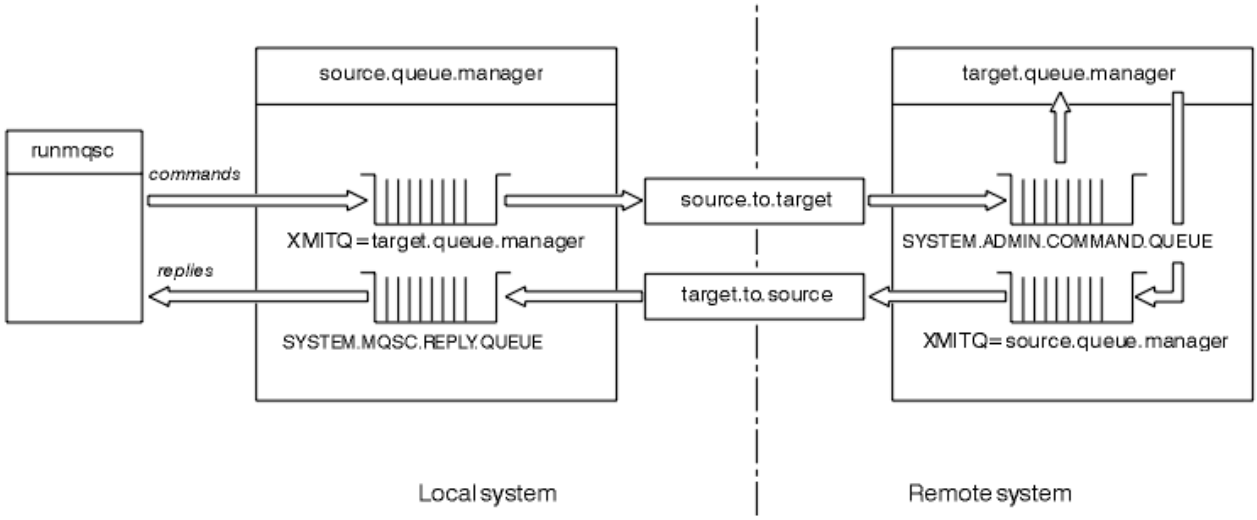


圖 *18:* 設定遠端管理的通道及佇列

如需設定通道的相關資訊，請參閱 使用分散式佇列連接應用程式 。

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 WebSphere 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。自動定義基於 WebSphere 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 自動定義叢集通道。

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 WebSphere MQ for Windows 或 WebSphere MQ for Linux (x86 及 x86-64 平台) 的使用者可以使用「IBM WebSphere MQ 探險家」來執行下列各節中說明的作業。 如需相關資訊，請參閱第 51 頁的『使用 IBM WebSphere 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 97 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 72 頁的『解決 MQSC 指令的問題』。

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 99 頁的『建立遠端佇列的本端定義』。

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 。

編碼字集之間的資料轉換

佇列管理程式可以將 WebSphere MQ 定義格式 (也稱為 內建格式) 中的訊息資料從一個編碼字集轉換成另一個編碼字集，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 102 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 WebSphere 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 and Linux 平台，受支援的字碼集保留在 WebSphere MQ 所提供的轉換表中。
- 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 WebSphere MQ for Windows 中，依預設 ccsid.tbl 位於 `C:\Program Files\IBM\WebSphere MQ\conv\table` 目錄中。

在 WebSphere MQ for UNIX and 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 WebSphere 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。 如果您只使用適用於 WebSphere MQ 物件名稱 (如 命名 IBM WebSphere MQ 物件中所定義) 的字元集，則通常會滿足此需求。 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 WebSphere MQ Telemetry

使用 IBM WebSphere MQ Explorer 或在指令行管理 IBM WebSphere 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 WebSphere MQ 的 MQTT 用戶端。 使用 JAAS、SSL 和 IBM WebSphere MQ 物件權限管理程式來配置 IBM WebSphere MQ Telemetry 的安全。

使用 IBM WebSphere 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 WebSphere MQ 的 MQTT 用戶端。 使用 JAAS、SSL 和 IBM WebSphere MQ 物件權限管理程式來配置 IBM WebSphere MQ Telemetry 的安全。

使用指令行管理

可以在指令行使用 IBM WebSphere MQ MQSC 指令完全管理 IBM WebSphere MQ Telemetry。

IBM WebSphere MQ Telemetry 文件也有範例 Script，示範 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 WebSphere MQ Telemetry 小節中 IBM WebSphere MQ Telemetry 範例程式 中的範例。

相關概念

WebSphere MQ Telemetry

第 107 頁的『配置分散式佇列以將訊息傳送至 MQTT 用戶端』

IBM WebSphere 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。

第 109 頁的『MQTT 用戶端識別、授權和鑑別』
第 114 頁的『使用 SSL 進行遙測通道鑑別』
第 116 頁的『遙測通道上的發佈保密』
第 117 頁的『MQTT 用戶端和遙測通道的 SSL 配置』
第 121 頁的『遙測通道 JAAS 配置』

配置 JAAS 以鑑別用戶端傳送的 Username。

第 123 頁的『用於裝置的 IBM WebSphere MQ Telemetry 常駐程式概念』

適用於裝置的 IBM WebSphere MQ Telemetry 常駐程式是進階 MQTT V3 用戶端應用程式。 可以利用它來儲存及轉遞來自其他 MQTT 用戶端的訊息。 它可以如 MQTT 用戶端一樣連接至 IBM WebSphere MQ，但您也可以將它與其他 MQTT 用戶端連接。

相關工作

第 104 頁的『在 Linux 和 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 WebSphere MQ Telemetry。 您可以執行自動化程序，以使用 IBM WebSphere MQ Explorer 的 IBM WebSphere MQ Telemetry 支援來設定更簡單的配置。

第 105 頁的『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 WebSphere MQ Telemetry。 您可以執行自動化程序，以使用 IBM WebSphere MQ Explorer 的 IBM WebSphere MQ Telemetry 支援來設定更簡單的配置。

相關參考

MQXR 內容

在 Linux 和 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 WebSphere MQ Telemetry。 您可以執行自動化程序，以使用 IBM WebSphere MQ Explorer 的 IBM WebSphere 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 WebSphere MQ 及 IBM WebSphere MQ Telemetry 特性的相關資訊，請參閱 安裝 IBM WebSphere 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

IBM WebSphere MQ Explorer 的 IBM WebSphere 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 IBM WebSphere MQ Explorer 驗證 IBM WebSphere MQ Telemetry 的安裝 和 IBM WebSphere MQ Telemetry 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 WebSphere 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 IBM WebSphere 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 IBM WebSphere MQ 訊息的用戶端新增遠端佇列定義; 請參閱 第 108 頁的『將訊息直接傳送至用戶端』。

4. 遵循 第 110 頁的『授權 MQTT 用戶端存取 WebSphere MQ 物件』 中的程序來建立一或多個使用者 ID。 使用者 ID 有權發佈、訂閱及傳送發佈至 MQTT 用戶端。
5. 安裝遙測 (MQXR) 服務

   ```
   cat /opt/<install_dir>/mqxr/samples/installMQXRService_unix.mqsc | runmqsc qMgr
   ```

   另請參閱 第 105 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 IBM WebSphere 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 WebSphere MQ Telemetry 程式範例。

範例

第 105 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 WebSphere MQ Telemetry。 您可以執行自動化程序，以使用 IBM WebSphere MQ Explorer 的 IBM WebSphere 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 WebSphere MQ 及 IBM WebSphere MQ Telemetry 特性的相關資訊，請參閱 安裝 IBM WebSphere 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 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

IBM WebSphere MQ Explorer 的 IBM WebSphere 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 IBM WebSphere MQ Explorer 驗證 IBM WebSphere MQ Telemetry 的安裝 和 IBM WebSphere MQ Telemetry 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 WebSphere 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 IBM WebSphere MQ Explorer 或 第 106 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 WebSphere MQ 訊息的用戶端新增遠端佇列定義; 請參閱 第 108 頁的『將訊息直接傳送至用戶端』。
4. 遵循 第 110 頁的『授權 MQTT 用戶端存取 WebSphere MQ 物件』 中的程序來建立一或多個使用者 ID。 使用者 ID 有權發佈、訂閱及傳送發佈至 MQTT 用戶端。
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 IBM WebSphere 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 WebSphere MQ Telemetry 程式範例。

手動建立 SYSTEM.MQXR.SERVICE

第 107 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*

配置分散式佇列以將訊息傳送至 MQTT 用戶端

IBM WebSphere 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 用戶端會連接至 WebSphere MQ 作為佇列管理程式，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 WebSphere MQ 物件名稱解析， *ClientIdentifier* 解析如下; 請參閱 第 108 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 WebSphere 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 WebSphere 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，也可以直接將訊息傳送至 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 WebSphere MQ 應用程式) 可以。

WebSphere MQ 應用程式必須知道 MQTT v3 用戶端的 ClientIdentifier 。 由於 MQTT v3 用戶端沒有佇列，因此會將目標佇列名稱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

如果應用程式是使用 JMS 來撰寫，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:* 將訊息傳送至 *MQTT v3* 用戶端的 *JMS* 目的地

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 109 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，則訊息會直接傳送至 MQTT 用戶端，其會呼叫 `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 Stateless 及 Stateful 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，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、授權和鑑別

遙測 (MQXR) 服務使用 MQTT 通道，代表 MQTT 用戶端發佈至 WebSphere MQ 主題，或訂閱該主題。 WebSphere MQ 管理者會配置用於 WebSphere 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 `Username` 和 `Password`。 它們還可以設定加密和鑑別連線所需的 SSL 內容。 管理者決定是否以及如何鑑別 MQTT 通道。

若要授權 MQTT 用戶端存取 WebSphere MQ 物件，請授權用戶端的 `ClientIdentifier` 或 `Username`，或者授權一般用戶端身分。 若要允許用戶端連接至 WebSphere 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 `Password`，請使用 VPN 加密連線，或者配置 MQTT 通道以使用 SSL，來讓密碼保持私密。

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。 而是使用用戶端憑證鑑別裝置，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 用戶端身分和授權

使用 `ClientIdentifier`、`Username` 或用於授權的一般用戶端身分來存取 WebSphere MQ 物件。

WebSphere MQ 管理者有三個選取 MQTT 通道身分的選項。 管理者在定義或修改用戶端使用的 MQTT 通道時進行選擇。 身分用於授與 WebSphere MQ 主題的存取權。 您可以選取:

1. 用戶端 ID。
2. 管理者提供給通道的身分。
3. 從 MQTT 用戶端傳遞的 Username。

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。 其預設值是空值。

使用 WebSphere MQ **setmqaut** 指令，以選取要授權給與 MQTT 通道相關聯的身分使用的物件和動作。 例如，若要授權佇列管理程式 QM1 管理者所提供的通道身分 MQTTClient:

```
setmqaut -m QM1 -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p MQTTClient -all +put
setmqaut -m QM1 -t topic -n SYSTEM.BASE.TOPIC -p MQTTClient -all +pub +sub
```

授權 *MQTT* 用戶端存取 *WebSphere MQ* 物件

請遵循下列步驟，授權 MQTT 用戶端發佈及訂閱 WebSphere 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，以獲授權存取 WebSphere MQ 中的物件。
「WebSphere MQ 管理者」會使用「WebSphere MQ 探險家」來配置遙測通道，以提供三種身分類型之一給用戶端:

1. ClientIdentifier
2. **使用者名稱**
3. 管理者指派給通道的名稱。

不論使用哪種類型，已安裝的授權服務必須將身分定義給 WebSphere 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 WebSphere MQ 中所定義主題的許可權。 如果 MQTT 用戶端已訂閱某個主題，請使用身分為其提供接收所產生發佈資訊的許可權。

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。 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 用戶端結合至一般身分:

1. 定義多個遙測通道，每個通道都具有管理者使用「WebSphere 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 111 頁的『無存取控制』
- 第 111 頁的『粗略存取控制』
- 第 111 頁的『中階存取控制』
- 第 111 頁的『精細存取控制』

模型的結果是指派 *mqttUsers* 許可權集來發佈和訂閱 WebSphere MQ，以及從 WebSphere MQ 接收發佈。

無存取控制
MQTT 用戶端獲授與 WebSphere MQ 管理權限，並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，以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 WebSphere 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 PublishX -all +pub
setmqaut -m qMgr -t topic -n topic1 -p SubscribeX -all +pub +sub
setmqaut -m qMgr -t q -n SYSTEM.MQTT.TRANSMIT.QUEUE -p mqtt -all +put
```

使用密碼進行 MQTT 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 會使用 MQTT 用戶端提供的 Password。

WebSphere MQ 管理者透過配置用戶端連接至的 MQTT 通道，決定是鑑別 Username，還是或者根本不執行鑑別。 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. WebSphere MQ 管理者使用「WebSphere MQ 探險家」，將 MQTT 通道身分設為任一名稱，例如 MQTTClientUser。
2. WebSphere MQ 管理者授權 MQTTClient 發佈至和訂閱任何主題：

```
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. MQTT 用戶端應用程式開發者會建立 MqttConnectOptions 物件，並在連接至伺服器之前設定 Username 及 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. WebSphere MQ 管理者配置 MQTT 通道，以使用 JAAS 鑑別用戶端的 UserName。

使用 SSL 進行 MQTT 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署數位憑證，您可以向 IBM WebSphere MQ 鑑別 MQTT 用戶端。 IBM 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。

-
-

使用 *SSL* 進行 *MQTT* 用戶端鑑別的遙測通道配置

IBM WebSphere 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。

WebSphere 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` 與用戶端 **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 WebSphere 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 WebSphere 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。以下是目前支援的 SSL 密碼組合清單（依英文字母排序）：

- `SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`
- `SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`
- `SSL_DH_anon_WITH_AES_128_CBC_SHA`
- `SSL_DH_anon_WITH_DES_CBC_SHA`
- `SSL_DH_anon_WITH_RC4_128_MD5`
- `SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_DSS_WITH_AES_128_CBC_SHA`
- `SSL_DHE_DSS_WITH_DES_CBC_SHA`
- `SSL_DHE_DSS_WITH_RC4_128_SHA`
- `SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA`

- SSL_DHE_RSA_WITH_AES_128_CBC_SHA
- SSL_DHE_RSA_WITH_DES_CBC_SHA
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5
- SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA
- SSL_KRB5_EXPORT_WITH_RC4_40_MD5
- SSL_KRB5_EXPORT_WITH_RC4_40_SHA
- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 如果您計劃使用 SHA-2 密碼組合，請參閱 使用 SHA-2 密碼組合與 MQTT 通道的系統需求。

相關概念

第 115 頁的『使用 SSL 進行通道鑑別的遙測通道配置』

IBM WebSphere 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。

CipherSpecs 和 CipherSuites

相關參考

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

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。 在某些使用 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 WebSphere 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 WebSphere MQ 探險家來配置遙測通道，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。以下是目前支援的 SSL 密碼組合清單（依英文字母排序）：

- `SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DH_anon_EXPORT_WITH_RC4_40_MD5`
- `SSL_DH_anon_WITH_3DES_EDE_CBC_SHA`
- `SSL_DH_anon_WITH_AES_128_CBC_SHA`
- `SSL_DH_anon_WITH_DES_CBC_SHA`
- `SSL_DH_anon_WITH_RC4_128_MD5`
- `SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_DSS_WITH_AES_128_CBC_SHA`
- `SSL_DHE_DSS_WITH_DES_CBC_SHA`
- `SSL_DHE_DSS_WITH_RC4_128_SHA`
- `SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA`
- `SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA`
- `SSL_DHE_RSA_WITH_AES_128_CBC_SHA`
- `SSL_DHE_RSA_WITH_DES_CBC_SHA`
- `SSL_KRB5_EXPORT_WITH_DES_CBC_40_MD5`
- `SSL_KRB5_EXPORT_WITH_DES_CBC_40_SHA`
- `SSL_KRB5_EXPORT_WITH_RC4_40_MD5`
- `SSL_KRB5_EXPORT_WITH_RC4_40_SHA`

- SSL_KRB5_WITH_3DES_EDE_CBC_MD5
- SSL_KRB5_WITH_3DES_EDE_CBC_SHA
- SSL_KRB5_WITH_DES_CBC_MD5
- SSL_KRB5_WITH_DES_CBC_SHA
- SSL_KRB5_WITH_RC4_128_MD5
- SSL_KRB5_WITH_RC4_128_SHA
- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- SSL_RSA_EXPORT_WITH_RC4_40_MD5
- SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_FIPS_WITH_AES_256_CBC_SHA256
- SSL_RSA_FIPS_WITH_DES_CBC_SHA
- SSL_RSA_WITH_3DES_EDE_CBC_SHA
- SSL_RSA_WITH_AES_128_CBC_SHA
- V7.5.0.2 SSL_RSA_WITH_AES_128_CBC_SHA256
- V7.5.0.2 SSL_RSA_WITH_AES_256_CBC_SHA256
- SSL_RSA_WITH_DES_CBC_SHA
- SSL_RSA_WITH_NULL_MD5
- SSL_RSA_WITH_NULL_SHA
- V7.5.0.2 SSL_RSA_WITH_NULL_SHA256
- SSL_RSA_WITH_RC4_128_MD5
- SSL_RSA_WITH_RC4_128_SHA

V7.5.0.2 如果您計劃使用 SHA-2 密碼組合，請參閱 使用 SHA-2 密碼組合與 MQTT 通道的系統需求。

相關概念

第 113 頁的『使用 SSL 進行 MQTT 用戶端鑑別的遙測通道配置』

IBM WebSphere 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。

CipherSpecs 和 CipherSuites

相關參考

DEFINE CHANNEL (MQTT)

ALTER CHANNEL (MQTT)

遙測通道上的發佈保密

使用 SSL 加密透過連線的傳輸，可保護在遙測通道之間以任一方向所傳送 MQTT 發佈保密的安全。

連接至遙測通道的 MQTT 用戶端，會使用 SSL 透過對稱金鑰加密法來維護在通道上所傳輸發佈保密的安全。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 114 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。

MQTT 用戶端和遙測通道的 SSL 配置

MQTT 用戶端及 WebSphere MQ Telemetry (MQXR) 服務使用 Java Secure Socket Extension (JSSE) ， 以使用 SSL 來連接遙測通道。 適用於裝置的 IBM WebSphere MQ Telemetry 常駐程式不支援 SSL。

配置 SSL 以鑑別遙測通道、MQTT 用戶端，並加密用戶端與遙測通道之間的訊息傳送。

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. 僅允許信任的 MQTT 用戶端連接。 僅將 MQTT 用戶端連接至信任的遙測通道。 加密用戶端與佇列管理程式之間的訊息; 請參閱 第 112 頁的『使用 SSL 進行 MQTT 用戶端鑑別』
2. 僅將 MQTT 用戶端連接至信任的遙測通道。 加密用戶端和佇列管理程式之間的訊息；請參閱第 114 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 116 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 WebSphere MQ 遙測通道
2. MQTT Java 用戶端
3. JRE

使用「IBM WebSphere 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 WebSphere MQ 遙測通道

使用「WebSphere MQ 探險家」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。 對通道命名，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。 透過指定定義給通道的埠，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。

Telemetry 服務和 MQTT 用戶端根據在每端上啟用的所有 CipherSpec，協議一般 CipherSpecs。 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 WebSphere 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 and 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 WebSphere MQ 指令 strmqikm. 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** 指令啟動 **ikeyman**，或者從 IBM 開發者套件安全資訊下載嚴密但適用範圍受限制的檔案。

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

範例說明如何新增信任提供者，以及如何從 MQTT 用戶端程式碼中參照它。 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 120 頁的圖 25 中的程式碼 Snippet 會為 MQTT 用戶端設定 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。

WebSphere MQ 管理者會配置哪些 MQTT 通道需要使用 JAAS 進行用戶端鑑別。 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 122 頁的圖 29。

JAAS 提供四個標準「登入」模組。 標準 NT 和 UNIX「登入」模組具有受限制的值。

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。 WebSphere MQ Telemetry 隨附了範例 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。 請勿將類別放置在類別路徑中的 WebSphere MQ 目錄上。 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，它們會傳回連接至 MQTT 通道的用戶端的 Username 及 Password。 **使用者名稱** 和 **密碼** 設定在 MqttConnectOptions 物件中。 請參閱第 122 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，編寫目的是接收 MQTT 用戶端提供的 Username 和 Password。

```
  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

用於裝置的 IBM WebSphere MQ Telemetry 常駐程式概念

適用於裝置的 IBM WebSphere MQ Telemetry 常駐程式是進階 MQTT V3 用戶端應用程式。 可以利用它來儲存及轉遞來自其他 MQTT 用戶端的訊息。 它可以如 MQTT 用戶端一樣連接至 IBM WebSphere MQ，但您也可以將它與其他 MQTT 用戶端連接。

常駐程式是一種發佈/訂閱分配管理系統。 MQTT 第 3 版用戶端連接至該常駐程式來發佈至及訂閱主題，使用主題字串進行發佈，使用主題過濾器進行訂閱。 主題字串是階層式的，主題層次以 / 分割。 主題過濾器是可以包含單一層次 + 萬用字元的主題字串，並包含多層次 # 萬用字元作為主題字串的最後一部分。

註: 常駐程式中的萬用字元遵循 WebSphere Message Broker 第 6 版中較為嚴格的規則。 IBM WebSphere MQ 則不同。 它支援多個多層次萬用字元；萬用字元可以代表任何數目的階層層次，並且可以位於主題字串的任何位置。

多個 MQTT 第 3 版用戶端使用接聽器埠連接至常駐程式。 可以修改預設接聽器埠。 您可以定義多個接聽器埠並為其配置不同的名稱空間，請參閱第 129 頁的『用於裝置的 WebSphere MQ Telemetry 常駐程式接聽器埠』。 常駐程式自身是一個 MQTT 第 3 版用戶端。 配置常駐程式橋接器連線，可將常駐程式連接至另一個常駐程式的接聽器埠，或連接至 WebSphere MQ Telemetry (MQXR) 服務。

您可以為適用於裝置的 WebSphere MQ Telemetry 常駐程式配置多個橋接器。 使用橋接器可將能夠交換發佈的常駐程式網路連接在一起。

每個橋接器都可以發佈至及訂閱其本端常駐程式上的主題。 它還可以發佈至及訂閱其他常駐程式、WebSphere MQ 發佈/訂閱分配管理系統或其連接的任何其他 MQTT 第 3 版分配管理系統上的主題。 透過使用主題過濾器，您可以選取要從一個分配管理系統傳送至另一個分配管理系統的發佈。 您可以任一方向傳送發佈。 您可以將發佈從本端常駐程式傳送至其連接的每個遠端分配管理系統，或從連接的任一分配管理系統傳送至本端常駐程式；請參閱第 123 頁的『用於裝置的 IBM WebSphere MQ Telemetry 常駐程式橋接器』。

用於裝置的 IBM WebSphere MQ Telemetry 常駐程式橋接器

用於裝置的 IBM WebSphere MQ Telemetry 常駐程式橋接器，可使用 MQTT 第 3 版通訊協定連接兩個發佈/訂閱分配管理系統。 該橋接器可以任一方向將發佈從一個分配管理系統傳送至另一個分配管理系統。 一端為用於裝置的 WebSphere MQ Telemetry 常駐程式橋接器連線，另一端可以是佇列管理程式或另一個常駐程式。 使用遙測通道可將佇列管理程式連接至橋接器連線。 使用常駐程式接聽器可將常駐程式連接至橋接器連線。

裝置的 IBM WebSphere MQ Telemetry 常駐程式同時間可只有一條對其他分配管理系統的連線，也可有多條連線。 來自常駐程式的連線稱為橋接器，並由常駐配置檔中的連線項目定義。 使用 IBM WebSphere MQ 遙測通道來建立 IBM WebSphere MQ 連線，如下圖所示：

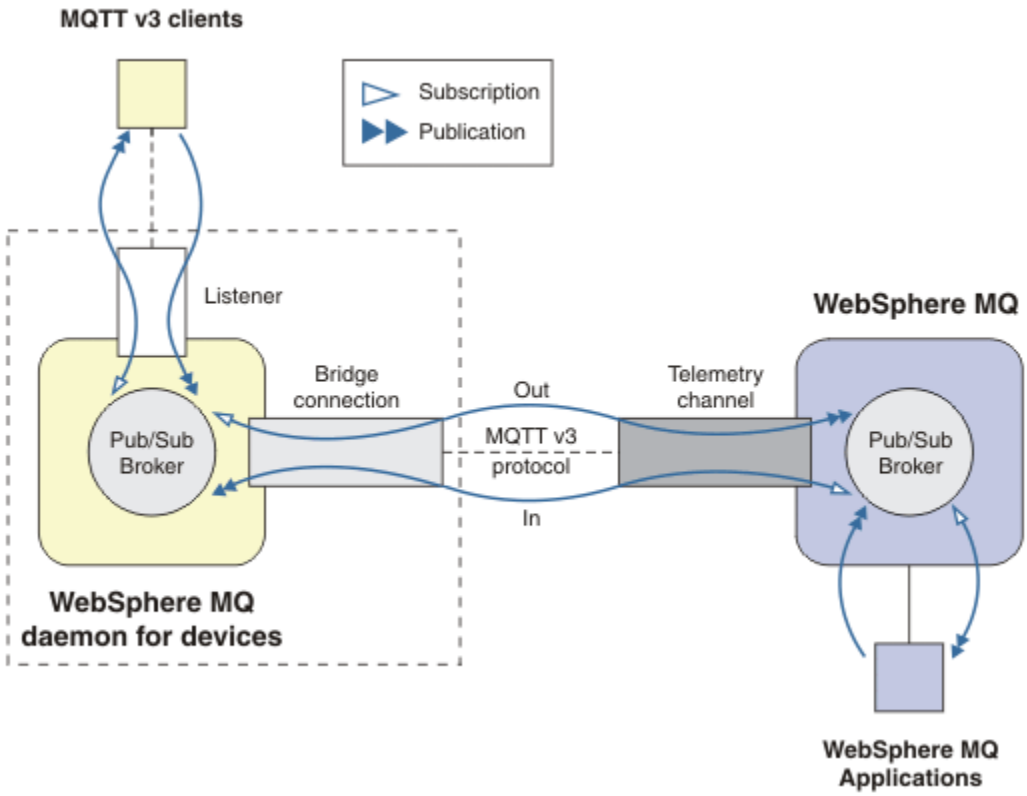


圖 *31:* 將 *IBM WebSphere MQ Telemetry daemon for devices* 連接至 *IBM WebSphere MQ*

橋接器可將常駐程式以 MQTT v3 用戶端形式連接至另一個分配管理系統。 橋接器參數可鏡映 MQTT 第 3 版用戶端的屬性。

橋接器不只是一個連線。 它還可作為兩個發佈/訂閱分配管理系統之間的發佈與訂閱代理程式。 本端分配管理系統為裝置的 IBM WebSphere MQ Telemetry 常駐程式，遠端分配管理系統則是支援 MQTT 第 3 版通訊協定的任何發佈/訂閱分配管理系統。 遠端分配管理系統一般是另一個常駐程式或 IBM WebSphere MQ。

橋接器的工作是在兩個分配管理系統之間傳送發佈。 橋接器是雙向的。 它可以任一方向傳送發佈。 第 124 頁的圖 31 說明橋接器如何將裝置的 IBM WebSphere MQ Telemetry 常駐程式連接至 IBM WebSphere MQ。 第 125 頁的『橋接器的主題設定範例』利用範例說明如何使用主題參數來配置橋接器。

第 124 頁的圖 31 中的 In 和 Out 箭頭指出橋接器的雙向方向性。 在箭頭的其中一端建立訂閱。 符合訂閱的發佈會發佈至箭頭另一端的分配管理系統。 箭頭是根據發佈流向標示的。 發佈流向 In 表示流入常駐程式，Out 則表示流出常駐程式。 標籤的重要性是它們會在指令語法中使用。 請記住，In and Out 參照發佈流向，而非訂閱的傳送方向。

其他用戶端、應用程式或分配管理系統可以連接至 IBM WebSphere MQ 或適用於裝置的 WebSphere MQ Telemetry 常駐程式。 它們可發佈至及訂閱其所連接的分配管理系統上的主題。 如果分配管理系統是 IBM WebSphere MQ，主題可能為叢集或分散式，並且未在本端佇列管理程式上明確定義。

橋接器的使用

使用橋接器連線及接聽器將常駐程式連接在一起。 使用橋接器連線及遙測通道，將常駐程式及佇列管理程式連接在一起。 將多個分配管理系統連接在一起時，可以建立迴圈。 請注意，發佈可能會在分配管理系統迴圈中不斷地循環且無法偵測到。

使用常駐程式橋接至 IBM WebSphere MQ 的部分原因如下所示：

減少與 WebSphere MQ 的 MQTT 用戶端連線數目
透過使用常駐程式階層，您可以將多個用戶端連接至 WebSphere MQ；一次可以連接的用戶端數目多於單一佇列管理程式。

在 MQTT 用戶端與 WebSphere MQ 之間儲存及轉遞訊息
如果用戶端沒有自己的儲存體，您可以使用儲存及轉遞來避免維持用戶端與 IBM WebSphere MQ 之間的連續連線。 您可以在 MQTT 用戶端與 WebSphere MQ 之間使用多種連線類型；請參閱遙測概念及監視與控制實務範例。

過濾 MQTT 用戶端與 WebSphere MQ 之間交換的發佈
發佈通常分為在本端處理的訊息，以及涉及其他應用程式的訊息。 本端發佈可能包括控制感應器與掣動器之間的流向，而遠端發佈則包括讀數、狀態及配置指令要求。

變更發佈的主題空間
避免來自連接至不同接聽器埠之用戶端的主題字串，彼此發生衝突。 範例使用常駐程式來標示來自不同大廈的計量讀數；請參閱分隔不同用戶端群組的主題字串。

橋接器的主題設定範例

將所有內容發佈至遠端分配管理系統 - 使用預設值

預設方向稱為 out，橋接器會將主題發佈至遠端分配管理系統。 topic 參數使用主題過濾器來控制傳送的主題。

橋接器會使用 第 125 頁的圖 32 中的 topic 參數，來訂閱 MQTT 用戶端或其他分配管理系統發佈至本端常駐程式的所有內容。 橋接器會將主題發佈至橋接器所連接的遠端分配管理系統。

```
connection Daemon1
topic #
```

圖 *32:* 將所有內容發佈至遠端分配管理系統

將所有內容發佈至遠端分配管理系統 - 明確

下列程式碼片段中的 topic 設定提供與使用預設值相同的結果。 唯一的差異是 **direction** 參數是明確的。 使用 out 方向可訂閱本端分配管理系統、常駐程式，並發佈至遠端分配管理系統。 橋接器訂閱的於本常駐程式上建立的發佈，會在遠端分配管理系統上進行發佈。

```
connection Daemon1
topic # out
```

圖 *33:* 將所有內容發佈至遠端分配管理系統 - 明確

將所有內容發佈至本端分配管理系統

您可以設定相反的方向 in，來取代使用方向 out。 下列程式碼片段可將橋接器配置為訂閱橋接器所連接遠端分配管理系統上發佈的所有內容。 橋接器會將主題發佈至本端分配管理系統，即常駐程式。

```
connection Daemon1
topic # in
```

圖 *34:* 將所有內容發佈至本端分配管理系統

將所有內容從本端分配管理系統上的 export 主題，發佈至遠端分配管理系統上的 import 主題

使用另外兩個參數 **local_prefix** 及 **remote_prefix**，來修改前一個範例中的主題過濾器 #。 一個參數用於修改訂閱中所使用的主題過濾器，另一個參數用於修改將發佈發佈至其中的主題。 如此一來，其中一個分配管理系統中使用的主題字串開頭，會由另一個分配管理系統上的另一個主題字串所取代。

根據主題指令的方向，**local_prefix** 及 **remote_prefix** 的意義是相反的。 如果方向為 out（預設值），則 **local_prefix** 用作主題訂閱的一部分，**remote_prefix** 會取代遠端訂閱中主題字串的 **local_prefix** 部分。 如果方向為 in，**remote_prefix** 會變成遠端訂閱的一部分，**local_prefix** 則會取代主題字串的 **remote_prefix** 部分。

在定義主題空間時，通常需要考量主題字串的第一部分。 可使用其他參數，來變更將主題發佈至其中的主題空間。 您可以透過此舉來避免要傳送的主題與目標分配管理系統上的另一個主題發生衝突，或透過此舉移除裝載點主題字串。

例如，在下列程式碼片段中，會將對常駐程式上主題字串 `export/#` 的所有發佈，重新發佈至遠端分配管理系統上的 `import/#`。

```
topic # out export/ import/
```

圖 *35:* 將所有內容從本端分配管理系統上的 *export* 主題，發佈至遠端分配管理系統上的 *import* 主題

將所有內容從遠端分配管理系統上的 export 主題，發佈至本端分配管理系統上的 import 主題

下列程式碼片段顯示相反的配置，橋接器會訂閱以遠端分配管理系統上的 `export/#` 主題字串發佈的所有內容，並將該內容發佈至本端分配管理系統上的 `import/#`。

```
connection Daemon1
topic # in import/ export/
```

圖 *36:* 將所有內容從遠端分配管理系統上的 *export* 主題，發佈至本端分配管理系統上的 *import* 主題

將來自 1884/ 裝載點的所有內容，以原始主題字串發佈至遠端分配管理系統

在下列程式碼片段中，橋接器會訂閱連接至本端常駐程式裝載點 1884/ 的用戶端發佈的所有內容。 橋接器會將發佈至該裝載點的所有內容，發佈至遠端分配管理系統。 發佈至遠端分配管理系統的主題中會移除裝載點字串 1884/。 *local_prefix* 與裝載點字串 1884/ 相同，而 *remote_prefix* 為空白字串。

```
listener 1884
mount_point 1884/
connection Daemon1
topic # out 1884/ ""
```

圖 *37:* 將來自 *1884/* 裝載點的所有內容，以原始主題字串發佈至遠端分配管理系統。

分隔不同用戶端（連接至不同常駐程式）的主題空間

針對電力計量器撰寫應用程式，以發佈建築的計量讀數。 使用 MQTT 用戶端將讀數發佈至位於同一棟建築物的常駐程式。 選取的發佈主題為 `power`。 將相同的應用程式部署至綜合大樓中的多棟建築物。 為了實現網站監視及資料儲存，會使用橋接器連線聚集來自所有建築物的讀數。 這些連線會將大廈常駐程式鏈結至位於集中位置的 WebSphere MQ。

每一個建置中的用戶端應用程式都相同，但必須透過建置來區分資料。 每一個讀數都有一個 `power` 主題，且必須以建築號碼作為字首才能加以識別。 來自綜合大樓第一棟建築物的橋接器使用字首 `meters/building01/`，來自第二棟建築物的字首為 `meters/building02/`。 來自其他建築物的讀數遵循相同的模式。 WebSphere MQ 會接收主題類似 `meters/building01/power` 的讀數。

範例是人為的; 實際上，應用程式發佈至的主題空間可能是可配置的。

每個常駐程式的配置檔都具有遵循下列程式碼片段中型樣的主題陳述式:

```
connection Daemon1
topic power out "" meters/building01/
```

圖 *38:* 分隔用戶端（連接至不同常駐程式）的主題空間

請指定空字串作為未用 local_prefix 參數的位置保留元。

分隔用戶端（連接至相同常駐程式）的主題空間

假設使用單一常駐程式來連接所有電源計量器。 假設在應用程式中可以配置為連接至不同埠，您可以透過將來自不同建築物的計量器連接至不同接聽器埠來識別建築物；如下列程式碼片段中所示。 再次聲明，範例是刻意設計的；它說明可以如何使用裝載點。

```
listener 1884
mount_point meters/building01/
listener 1885
mount_point meters/building02/
connection Daemon1
topic meters/+/power out
```

圖 *39:* 分隔用戶端（連接至相同常駐程式）的主題空間

重新對映在兩個方向流動的發佈的不同主題

在下列程式碼片段的配置中，橋接器會訂閱遠端分配管理系統上的單一主題 b ，並將 b 的相關發佈轉遞至本端常駐程式，將主題變更為 a。 橋接器也會訂閱本端分配管理系統上的單一主題 x ，並將 x 的相關出版品轉遞至遠端分配管理系統，將主題變更為 y。

```
connection Daemon1
topic "" in a b
topic "" out x y
```

圖 *40:* 重新對映在兩個方向流動的發佈的不同主題

此範例的重點是訂閱及發佈至兩個分配管理系統上的不同主題。 兩個分配管理系統上的主題空間未連接。

重新對映在兩個方向流動的發佈的相同主題（迴圈）

與前一個範例不同，第 127 頁的圖 41 中的配置通常會導致迴圈。 在主題陳述式 `topic "" in a b` 中，橋接器會從遠端訂閱 b，並在本端發佈至 a。 在另一個主題陳述式中，橋接器會在本端訂閱 a，並從遠端發佈至 b。 可以按第 127 頁的圖 42 所示撰寫相同的配置。

一般結果是如果用戶端從遠端發佈至 b ，則發佈會以主題 a 的發佈方式傳送至本端常駐程式。 不過，當橋接器發佈至主題 a 上的本端常駐程式時，發佈會比對橋接器對本端主題 a 所進行的訂閱。 訂閱為 `topic "" out a b`。 因此，發佈資訊會作為主題 b 的發佈資訊傳回遠端分配管理系統。 橋接器現在已訂閱遠端主題 b，且循環重新開始。

部分分配管理系統會實作迴圈偵測，以防止發生迴圈。 但是，將不同類型的分配管理系統橋接在一起時，迴圈偵測機制必須能正常運作。 如果將 WebSphere MQ 橋接至用於裝置的 WebSphere MQ Telemetry 常駐程式，則迴圈偵測無法運作。 橋接的兩端必須是裝置的 IBM WebSphere MQ Telemetry 常駐程式，此機制才能運作。 依預設會開啟迴圈偵測；請參閱 try_private。

```
connection Daemon1
topic "" in a b
topic "" out a b
```

圖 *41:* **!**針對雙向流動的發佈重新對映相同的主題

```
connection Daemon1
topic "" both a b
```

圖 *42:* **!**使用 *both*，針對雙向流動的發佈重新對映相同的主題。

第 127 頁的圖 40 中的配置與第 127 頁的圖 41 相同。

IBM WebSphere MQ Telemetry daemon for devices 橋接器連線的可用性

配置多個 IBM WebSphere MQ Telemetry daemon for devices 橋接器連線位址，以連接至第一個可用的遠端分配管理系統。 如果分配管理系統是多重實例佇列管理程式，請提供其兩個 TCP/IP 位址。 將主要連線配置為在主要伺服器可用時，連接或重新連接至該伺服器。

連線橋接器參數 addresses，是 TCP/IP 通訊端位址清單。 橋接器會依次嘗試連接至每個位址，直到其成功建立連線為止。 round_robin 及 start_type 連線參數，用於控制成功建立連線後位址的使用方式。

如果 start_type 為 auto、manual 或 lazy，則如果連線失敗，橋接器便會嘗試進行重新連接。 它會依次使用每個位址，每個連線嘗試之間會有大約 20 秒的延遲。 如果 `start_type` 為 once，則如果連線失敗，橋接器不會自動嘗試進行重新連接。

如果 round_robin 為 true，則橋接器連線會從清單中的第一個位址開始嘗試，並依次嘗試清單中的每一個位址。 當清單用盡時，它會重新從第一個位址開始嘗試。 如果清單中只有一個位址，它便會每隔 20 秒重新嘗試一次。

如果 round_robin 為 false，則清單中的第一個位址（該位址稱為主要伺服器）具有優先權。 如果第一次嘗試連接至主要伺服器失敗，橋接器會在背景繼續嘗試重新連接至主要伺服器。 同時，橋接器會嘗試使用清單中的其他位址進行連接。 如果背景嘗試連接至主要伺服器成功，橋接器便會中斷現行連線，並切換至主要伺服器連線。

如果連線主動中斷（例如透過發出 **connection_stop** 指令），則重新啟動連線時，它會嘗試仍使用相同的位址。 如果連線因連接失敗或遠端分配管理系統捨棄連線而中斷，橋接器會等待 20 秒。 之後，它便會嘗試連接至清單中的下一個位址，如果清單中只有一個位址，它會嘗試連接至相同位址。

連接至多重實例佇列管理程式

在多重實例佇列管理程式配置中，佇列管理程式會在兩個具有不同 IP 位址的不同伺服器上執行。 一般來說，不會以特定 IP 位址來配置遙測通道。 它們僅以埠號進行配置。 遙測通道啟動後，依預設它會選取本端伺服器上第一個可用的網址。

以佇列管理程式使用的兩個 IP 位址，來配置橋接器連線的 addresses 參數。 將 round_robin 設定為 true。

如果作用中佇列管理程式實例失敗，佇列管理程式便會切換至待命實例。 常駐程式可偵測到作用中實例的連線已中斷，並嘗試重新連接至待命實例。 它會使用為橋接器連線配置的位址清單中的其他 IP 位址。

橋接器連接的佇列管理程式仍為同一個佇列管理程式。 佇列管理程式會回復其自己的狀態。 如果將 cleansession 設定為 false，橋接器連線會恢復為失效接手之前的相同狀態。 連線會在延遲之後回復。 具有 "至少一次" 或 "最多一次" 服務品質的訊息不會遺失，且訂閱會繼續運作。

重新連線時間取決於待命實例啟動時重新啟動的通道及用戶端數，以及進行中的訊息數。 在重新建立連線之前，橋接器連線可能會嘗試重新連接至這兩個 IP 位址次數。

請勿以特定的 IP 位址配置多重實例佇列管理程式遙測通道。 IP 位址僅在一部伺服器上有效。

如果您是使用替代的高可用性解決方案來管理 IP 位址，則它可能更正為以特定的 IP 位址配置遙測通道。

cleansession

橋接器連線為 MQTT v3 用戶端階段作業。 您可以控制連線是否啟動新階段作業，或者它是否還原現有階段作業。 如果它還原現有階段作業，則橋接器連線會保留前一個階段作業的訂閱及保留的發佈。

如果 addresses 列出多個 IP 位址，且 IP 位址連接至不同佇列管理程式所管理的遙測通道，或連接至不同遙測常駐程式，則不要將 cleansession 設為 false。 不會在佇列管理程式或常駐程式之間傳送階段作業狀態。 如果嘗試在不同佇列管理程式或常駐程式上重新啟動現有階段作業，會導致啟動新的階段作業。 不確定的訊息會遺失，並且訂閱可能會發生未預期的行為。

notifications

透過使用通知，應用程式可以追蹤橋接器連線是否在執行中。 通知是具有值 1（已連接），或 0>（已中斷連線）的發佈。 它會發佈至 notification_topic 參數所定義的 *topicString* 。 *topicString* 的預設值是 $SYS/broker/connection/ *clientIdentifier*/state。 預設 *topicString* 包含字首 $SYS。 透過定義開頭

為 $SYS 的主題過濾器，來訂閱開頭為 $SYS 的主題。 主題過濾器 # 可訂閱所有主題，但不會訂閱常駐程式上開頭為 $SYS 的主題。 請將 $SYS 想像成定義不同於應用程式主題空間的特殊系統主題空間。

通知 可讓 IBM WebSphere MQ Telemetry daemon for devices 在橋接器已連接或中斷連線時通知 MQTT 用戶端。

keepalive_interval

keepalive_interval 橋接器連線參數，可設定橋接器將 TCP/IP 連線測試傳送至遠端伺服器的間隔。 預設間隔為 60 秒。 連線測試可防止遠端伺服器或防火牆（會偵測連線的閒置時間）關閉 TCP/IP 階段作業。

clientId

橋接器連線為 MQTT 第 3 版用戶端階段作業，並且具有橋接器連線參數 clientid 所設定的 `clientIdentifier`。 如果您想要透過將 cleansession 參數設定為 false 進行重新連線以回復前一個階段作業，則每個階段作業中使用的 `clientIdentifier` 必須相同。 clientid 的預設值保持不變仍為 *hostname.connectionName*。

安裝、驗證、配置及控制用於裝置的 WebSphere MQ Telemetry 常駐程式

安裝、配置及控制常駐程式以檔案為基礎。

將 Software Development Kit 複製到您要執行常駐程式 的裝置，以安裝常駐程式。

例如，執行 MQTT 用戶端公用程式，並以發佈/訂閱分配管理系統身分連接至適用於裝置的 WebSphere MQ Telemetry 常駐程式; 請參閱 將訊息發佈至特定的 MQTT v3 用戶端 。

透過建立配置檔配置常駐程式；請參閱用於裝置的 WebSphere MQ Telemetry 常駐程式配置檔。

透過在檔案 `amqtdd.upd` 中建立指令來控制執行中的常駐程式。 每隔 5 秒常駐程式就會讀取該檔案、執行指令並刪除檔案；請參閱用於裝置的 WebSphere MQ Telemetry 常駐程式指令檔。

用於裝置的 WebSphere MQ Telemetry 常駐程式接聽器埠

透過使用接聽器埠，將 MQTT 第 3 版用戶端連接至用於裝置的 WebSphere MQ Telemetry 常駐程式。 您可以使用裝載點及連線數目上限來限定接聽器埠。

接聽器埠必須對應於連接至此埠之用戶端的 MQTT 用戶端 `connect(serverURI)` 方法上指定的埠號。 它在用戶端及常駐程式上均使用預設值 1883。

透過設定常駐配置檔中的廣域定義 `port`，您可以變更常駐程式的預設埠。 透過在常駐配置檔中新增 `listener` 定義，您可以設定特定的埠。

對於預設埠以外的每個接聽器埠，您都可以指定裝載點以隔離用戶端。 連接至具有裝載點之埠的用戶端會與其他用戶端隔離；請參閱第 129 頁的『用於裝置的 WebSphere MQ Telemetry 常駐程式裝載點』。

您可以限制可以連接至任一埠的用戶端數目。 請設定廣域定義 `max_connections` 以限制預設埠的連線，或使用 `max_connections` 限定每個接聽器埠。

範例

以下配置檔範例可將預設埠由 1883 變更為 1880，並將埠 1880 的連線數限制為 10000。 埠 1884 的連線數限制為 1000。 連接至埠 1884 的用戶端與連接至其他埠的用戶端隔離。

```
port 1880
max_connections 10000
listener 1884
mount_point 1884/
max_connections 1000
```

用於裝置的 WebSphere MQ Telemetry 常駐程式裝載點

您可以將裝載點與 MQTT 用戶端所使用的接聽器埠關聯，以連接至用於裝置的 WebSphere MQ Telemetry 常駐程式。 裝載點會將使用一個接聽器埠的 MQTT 用戶端所交換的發佈及訂閱，與連接至其他接聽器埠的 MQTT 用戶端隔離。

連接至具有裝載點之接聽器埠的用戶端，無法與連接至任何其他接聽器埠的用戶端直接交換主題。 連接至沒有裝載點之接聽器埠的用戶端，可以發佈至或訂閱任何用戶端的主題。 用戶端無法知曉它們是否透過裝載點連接；這對用戶端所建立的主題字串來說沒有差別。

裝載點是加在發佈及訂閱主題字串前面的文字字串。 它會加在用戶端（連接至具有裝載點的接聽器埠）建立的所有主題字串的前面。 傳送至連接接聽器埠之用戶端的所有主題字串中會移除該文字字串。

如果接聽器埠沒有裝載點，則不會變更連接至該埠之用戶端所建立及接收的發佈及訂閱的主題字串。

建立尾端為 / 的裝載點字串。 如此一來，裝載點便成為裝載點主題樹狀結構的上層主題。

範例

配置檔包含下列接聽器埠：

```
listener 1883
mount_point 1883/
listener 1884 127.0.0.1
mount_point 1884/
listener 1885
```

連接至埠 1883 的用戶端會建立對 MyTopic 的訂閱。 常駐程式會將該訂閱登錄為 1883/MyTopic。 連接至埠 1883 的另一個用戶端會在主題 MyTopic 上發佈訊息。 常駐程式會將主題字串變更為 1883/MyTopic，並搜尋相符的訂閱。 埠 1883 上的訂閱者會接收含有原始主題字串 MyTopic 的發佈。 常駐程式已從主題字串中移除裝載點字首。

連接至埠 1884 的另一個用戶端也會在主題 MyTopic 上進行發佈。 此時，常駐程式會將主題登錄為 1884/MyTopic。 埠 1883 上的訂閱者不會收到發佈，因為不同的裝載點會導致含有不同主題字串的訂閱。

連接至埠 1885 的用戶端會在主題 1883/MyTopic 上進行發佈。 常駐程式不會變更主題字串。 埠 1883 上的訂閱者會接收 MyTopic 的發佈。

用於裝置的 WebSphere MQ Telemetry 常駐程式服務品質、可延續訂閱及保留的發佈

服務品質設定僅適用於執行中的常駐程式。 如果常駐程式停止（透過採用受控方式或因發生故障），進行中訊息的狀態會遺失。 如果常駐程式停止，則無法保證遞送訊息至少一次，或至多一次。 用於裝置的 WebSphere MQ Telemetry 常駐程式支援有限的持續性。 設定 **retained_persistence** 配置參數，可在常駐程式關閉時儲存保留的發佈及訂閱。

與 WebSphere MQ 不同， 適用於裝置的 WebSphere MQ Telemetry 常駐程式不會日誌登載持續資料。 不會以交易方式儲存階段作業狀態、訊息狀態及保留的發佈。 依預設，常駐程式在其停止時會捨棄所有資料。 您可以設定一個選項，以定期儲存訂閱及保留的發佈之檢查點。 常駐程式停止時，訊息狀態會一律遺失。 所有非保留的發佈都會遺失。

將常駐程式配置選項 Retained_persistence 設為 true，可定期將保留的發佈儲存至檔案。 常駐程式重新啟動時，即會恢復前次自動儲存的保留的發佈。 依預設，常駐程式重新啟動時不會恢復由用戶端建立的保留訊息。

將常駐程式配置選項 Retained_persistence 設為 true，可定期將持續性階段作業中建立的訂閱儲存至檔案。 如果 Retained_persistence 設為 true，則會還原用戶端在 CleanSession 設為 false("持續性階段作業") 的階段作業中建立的訂閱。 常駐程式會在其重新啟動後開始接收發佈時還原訂閱。 在 CleanSession 設為 false 的情況下，用戶端會在其重新啟動時接收發佈。 依預設，在常駐程式停止時不會儲存用戶端階段作業狀態，因此不會還原訂閱，即使用戶端將 CleanSession 設為 false 也是如此。

Retained_persistence 是一種自動儲存機制。 它可能無法儲存最新保留的發佈或訂閱。 您可以變更儲存保留的發佈及訂閱的頻率。 使用配置選項 autosave_on_changes 及 autosave_interval，可設定儲存間隔或儲存之間的變更次數。

設定持續性的範例配置

```
# Sample configuration
# Daemon listens on port 1882 with persistence in /tmp
# Autosave every minute
port 1882
```

```
persistence_location /tmp/
retained_persistence true
autosave_on_changes false
autosave_interval 60
```

用於裝置的 WebSphere MQ Telemetry 常駐程式安全

用於裝置的 WebSphere MQ Telemetry 常駐程式可以鑑別它所連接的用戶端、使用認證連接至其他分配管理系統，以及控制主題的存取。 該常駐程式提供的安全有所限制，因為它是使用不提供 SSL 支援的 WebSphere MQ Telemetry C 用戶端進行建置。 因此，進出常駐程式的連線不會加密，並且無法使用憑證進行鑑別。

依預設，不會開啟任何安全。

用戶端鑑別

MQTT 用戶端可以使用 `MqttConnectOptions.setUserName` 及 `MqttConnectOptions.setPassword` 方法，來設定使用者名稱及密碼。

透過針對密碼檔中的項目檢查用戶端所提供的使用者名稱及密碼，來鑑別連接至常駐程式的用戶端。 若要啟用鑑別，請建立密碼檔並設定常駐配置檔中的 `password_file` 參數；請參閱 password_file。

設定常駐配置檔中的 `allow_anonymous` 參數，可容許沒有使用者名稱或密碼的用戶端進行連接，以連接至檢查鑑別的常駐程式；請參閱 allow_anonymous。 如果已設定 `password_file` 參數，則在用戶端未提供使用者名稱或密碼時會一律針對密碼檔對其進行檢查。

設定常駐配置檔中的 `clientid_prefixes` 參數，可限制對特定用戶端的連線。 用戶端必須具有以 `clientid_prefixes` 參數中所列其中一個字首為開頭的 `clientIdentifiers`；請參閱 clientid_prefixes。

橋接器連線安全

每個用於裝置的 WebSphere MQ Telemetry 常駐程式橋接器連線自身都是一個 MQTT 第 3 版用戶端。 您可以在常駐配置檔中，針對每個橋接器連線設定 username 及 password 作為橋接器連線參數；請參閱 username 及 password。 之後，橋接器即可向分配管理系統鑑別自身。

主題的存取控制

如果正在鑑別用戶端，該常駐程式還可以針對每位使用者提供對主題的存取控制。 常駐程式會根據用戶端發佈至或訂閱的主題是否與存取控制檔中的存取主題字串相符，來授與存取控制權；請參閱 acl_file。

存取控制清單包含兩部分。 第一部分用於控制對所有用戶端（包括匿名用戶端）的存取。 第二部分包含針對密碼檔中任何使用者的區段。 它會列出每位使用者的特定存取控制。

範例

下列範例顯示了安全參數。

```
acl_file c:\WMQTDaemon\config\acl.txt
password_file c:\WMQTDaemon\config\passwords.txt
allow_anonymous true
connection Daemon1
username daemon1
password deamonpassword
```

圖 *43:* 常駐配置檔

```
Fred:Fredpassword
Barney:Barneypassword
```

圖 *44:* 密碼檔，*passwords.txt*

```
topic home/public/#
topic read meters/#
user Fred
topic write meters/fred
topic home/fred/#
user Barney
topic write meters/barney
topic home/barney/#
```

圖 *45:* 存取控制檔，*acl.txt*

管理多重播送

使用此資訊來瞭解 WebSphere 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 WebSphere MQ 多重播送主題及通訊資訊物件。

關於這項作業

WebSphere MQ 多重播送傳訊會將主題對映至群組位址，以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 133 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 WebSphere MQ 多重播送主題拓蹼

使用此範例來瞭解 IBM WebSphere MQ Multicast 主題拓蹼。

IBM WebSphere 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。 如需保留位址的清單，請聯絡您的系統管理者，或參閱 IPv4 多重播送位址空間登錄，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。

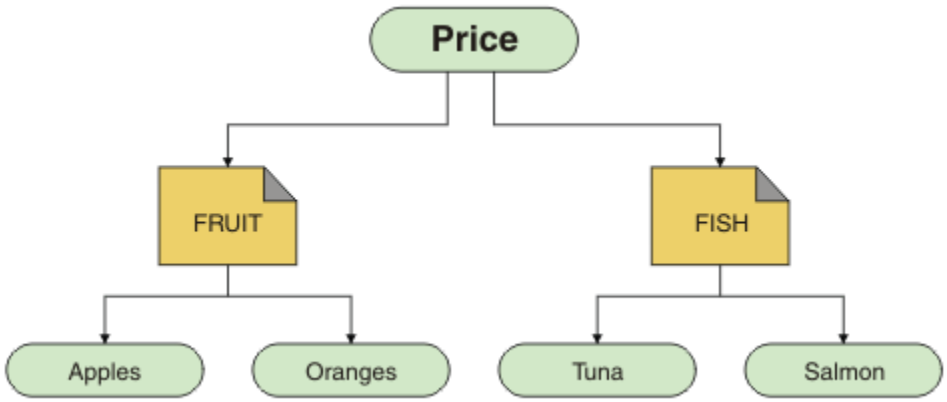
在下圖中，有兩個可能的多重播送資料串流:

```
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。

WebSphere MQ 多重播送對主題字串有 255 個字元限制。 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 WebSphere MQ 訊息格式，並減少 WebSphere MQ 訊息的大小。

WebSphere MQ 訊息有一些相關聯的屬性，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。 WebSphere MQ 多重播送可讓使用者配置這些屬性 (如果有的話) 隨訊息一起傳輸。

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 134 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 WebSphere 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 WebSphere Message Broker 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

表 9: 傳訊屬性及其與多重播送的關係 (繼續)		
屬性	**使用多重播送時的動作**	**如果未傳輸，則為預設值**
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 WebSphere MQ 多重播送傳訊的資料轉換運作方式。

WebSphere MQ 多重播送是一種共用的無連線通訊協定，因此無法讓每一個用戶端提出特定的資料轉換要求。 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 WebSphere 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生格式。 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。

WebSphere MQ Multicast 支援下列內建格式的訊息有效負載資料轉換:

- 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

使用此資訊來瞭解如何管理及監視 WebSphere MQ 多重播送。

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 WebSphere MQ 多重播送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 WebSphere 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，則會撰寫事件訊息。 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)** 表示目前有 20% 的訊息有問題，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 WebSphere MQ 多重播送訂閱及訊息歷程。

克服多重播送傳輸失敗的關鍵元素是 WebSphere MQ 的傳輸資料緩衝 (要保留在鏈結傳輸端的訊息歷程)。 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 WebSphere 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。

WebSphere MQ 多重播送訊息歷程由 **MSGHIST** 屬性控制在通訊資訊 (COMMINFO) 物件中:

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。

WebSphere MQ 多重播送新訂閱歷程是由 **NSUBHIST** 屬性在通訊資訊 (COMMINFO) 物件中控制:

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。

相關參考
DEFINE COMMINFO
ALTER COMMINFO

進階多重播送作業

使用此資訊來瞭解進階 WebSphere MQ 多重播送管理作業，例如配置 .ini 檔案以及與 WebSphere 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 WebSphere MQ 啟用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 WebSphere MQ 多重播送欄位。

其他 WebSphere MQ 多重播送配置可以在 ini 檔案中建立。 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 WebSphere MQ 資料目錄 (/var/mqm/mqclient.ini) 的位置， *QMNAME* 是 .ini 檔案適用的佇列管理程式名稱。

.ini 檔案包含用來細部調整 WebSphere 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
同時支援使用 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。

與 WebSphere MQ 低延遲傳訊的多重播送交互作業能力

使用此資訊來瞭解 WebSphere MQ Multicast 與 WebSphere MQ Low Latency Messaging (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 WebSphere 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。 WebSphere MQ 訊息內容及 MQMD 欄位會以 LLM 訊息內容傳輸，並具有特定的 LLM 訊息內容碼，如下表所示:

表 *10: WebSphere MQ* 訊息內容至 *WebSphere MQ LLM* 內容對映

WebSphere MQ 內容	**WebSphere 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
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: WebSphere MQ Low Latency Messaging。

管理 HP Integrity NonStop Server

使用此資訊來瞭解 HP Integrity NonStop Server 的 IBM WebSphere 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。

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 之產品、程式或服務，使用者必須自行負責作業之評估和驗證責任。

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：

Intellectual Property Licensing Legal and Intellectual Property Law 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 Software Interoperability Coordinator ， Department 49XA 3605 Highway 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。 如果您對非 IBM 產品的性能有任何的疑問，請逕向該產品的供應商查詢。

所有關於 IBM 未來方針或目的之聲明，隨時可能更改或撤銷，不必另行通知，且僅代表目標與主旨。

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 IBM 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: